Smart Community Platform for Taman Suria Tropika

Ibnu Firas bin Ahmad Fadzuli

(78039)

Bachelor of Software Engineering with Honours

2024/2025

Smart Community Platform for Taman Suria Tropika

IBNU FIRAS BIN AHMAD FADZULI

This project is submitted in partial fulfillment of the requirements for the degree of Bachelor of Software Engineering with Honours

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2024/2025

Smart Community Platform for Taman Suria Tropika

IBNU FIRAS BIN AHMAD FADZULI

Projek ini merupakan salah satu keperluan untuk Ijazah Sarjana Muda
Kejuruteraan Perisian dengan Kepujian

Fakulti Sains Komputer dan Teknologi Maklumat
UNIVERSITI MALAYSIA SARAWAK
2024/2025

UNIVERSITI MALAYSIA SARAWAK
THESIS STATUS ENDORSEMENT FORM

TITLE Smart Community Platform for Taman Suria Tropika

ACADEMIC SESSION: 2024/2025

(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
☐	UNRESTRICTED	

Validated by

Firas

(AUTHOR'S SIGNATURE)

(SUPERVISOR'S SIGNATURE)

Permanent Address

LOT 713, JALAN SEKOLAH ANCHI,
PIASAU JAYA FASA 1
98000 MIRI, SARAWAK

Date: 24/7/2025

Date:

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

DECLARATION

I hereby declare that this project is entirely my own original work. It has not been copied from any other student's work or external sources, except where proper acknowledgment is provided. No portion of this project was written by another individual on my behalf.

……*Firas*……

(IBNU FIRAS BIN AHMAD FADZULI) 2025

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to my supervisor, Madam/Ts. Nurfauza binti Jali, for the guidance and support she had given me throughout my Final Year Project. The comprehensive reviews she had given me greatly contributed to the refinement of this project. Secondly, I wish to express my appreciation to my examiner, Sir Muhammad Asyraf**,** who have endeavoured to examine and give helpful comments on my work.

My thanks also go to the Final Year Project coordinator, Sir Wang Yin Chai, who had presented useful and informative guidelines during his classes, which were crucial for navigating the project. Furthermore, I am grateful to Sir Azlan, the secretary of Surau Al-Ansar, for his willingness to share relevant information and for his cooperation throughout the development of this project. Finally, I wish to thank my family and friends who had always supported me as I went through all the work required of me for this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MVC	Model-View-Controller
RAD	Rapid Application Development
UI	User Interface

ABSTRACT

This project presents the development of a Smart Community Platform for Taman Suria Tropika, designed to support the management and communication needs of the Surau Al Ansar and its surrounding residential community. The system aims to improve administrative efficiency, community engagement, and digital communication through a user-friendly, web-based platform. It integrates key modules including event management, financial tracking, facility booking, community voting, a communication hub, user profile management, and notification handling. The platform was developed using Laravel (PHP framework), MySQL, CSS, and JavaScript with Alpine.js to handle interactive front-end components. It follows a modular architecture with role-based access control for both administrators and general users. Usability testing was conducted using a Google Forms-based questionnaire to gather real user feedback, which confirmed that the system was easy to navigate, visually clear, and met user expectations. Functional validation was carried out through unit testing using the Pest framework and system testing. Despite certain limitations, the project successfully achieved its objectives and demonstrates strong potential for adoption in other residential communities seeking to modernize their operations and foster a more connected environment.

ABSTRAK

Projek ini membentangkan pembangunan Platform Komuniti Pintar untuk Taman Suria Tropika, yang direka bagi menyokong keperluan pengurusan dan komunikasi Surau Al Ansar serta komuniti perumahan di sekitarnya. Sistem ini bertujuan untuk meningkatkan kecekapan pentadbiran, penglibatan komuniti, dan komunikasi digital melalui platform berasaskan web yang mesra pengguna. Ia merangkumi modul-modul utama termasuk pengurusan acara, penjejakan kewangan, tempahan kemudahan, pengundian komuniti, hab komunikasi, pengurusan profil pengguna, dan pengendalian notifikasi. Platform ini dibangunkan menggunakan Laravel (kerangka kerja PHP), MySQL, CSS, dan JavaScript bersama Alpine.js untuk mengurus komponen interaktif di bahagian hadapan. Sistem ini mengikuti seni bina modular dengan kawalan akses berasaskan peranan bagi kedua-dua pentadbir dan pengguna umum. Ujian kebolehgunaan telah dijalankan menggunakan borang soal selidik Google Forms untuk mendapatkan maklum balas pengguna sebenar, yang mengesahkan bahawa sistem ini mudah untuk dilayari, jelas dari segi visual, dan memenuhi jangkaan pengguna. Pengesahan fungsi telah dilaksanakan melalui ujian unit menggunakan kerangka Pest serta ujian sistem. Walaupun terdapat beberapa kekangan, projek ini berjaya mencapai objektifnya dan menunjukkan potensi besar untuk diguna pakai dalam komuniti perumahan lain yang ingin memodenkan operasi mereka dan menggalakkan persekitaran yang lebih berhubung.

CHAPTER 1: INTRODUCTION

1.1 Introduction

In a world where digital technologies are driving change for the betterment of mankind, it has become more common to see individuals and organisations leveraging technology to improve certain aspects of their everyday life or businesses. Places like mosques and small religious centres are of no exception and its people are willing to explore how they can utilise digital tools to improve their efficiency in management and even communication. As a Software Engineering student, the contribution to this shift is emphasized through the development of systems that facilitate such advancements.

Surau Al Ansar, a religious centre located in Taman Suria Tropika, Seri Kembangan, Selangor, currently faces challenges in managing events, tracking finances, and maintaining effective communication with the community. The surau relies on manual announcements and WhatsApp to disseminate information, which often leads to inefficiencies and missed communication. Furthermore, the surau's financial records are kept track using tools like spreadsheet in Microsoft Excel, which might make financial tracking difficult for those unused to it.

This project seeks to develop a web-based management and communication system for Surau Al Ansar, aiming to address these issues by offering a more streamlined approach to event management, financial tracking, and community interaction. The system will also serve as a communication hub for the wider Taman Suria Tropika community, allowing the residents to stay updated with any needed information and enable them to participate in community-related activities.

In addition to these core functions, the system will feature a Facility Booking System. This feature will allow the surau and the wider community to reserve event facilities. For instance, community members will be able to book the playground area, which doubles as an event space, while Surau Al-Ansar can book the surau for special events, such as hosting a guest speaker from outside the community to deliver a khutbah. This not only streamlines the booking process but also encourages more organised and efficient use of communal spaces.

Another key feature of the system is the Voting System, which will empower both the surau and the community to make decisions more democratically. This system can be used in various contexts: the community might use it for certain competitions that get organised

whenever an occasion arises, while the surau could employ it to vote on decisions such as what food to serve after Friday prayers or during special occasions. By facilitating greater participation in decision-making, this system encourages a sense of community ownership and engagement.

While Surau Al Ansar is the primary client for this project, the decision to extend the platform's functionality to serve and support the rest of the Taman Suria Tropika community came from the realisation that a shared communication platform could significantly improve community interaction, cooperation, and engagement. The system aims not only to improve the surau's operations, but also to foster a stronger and better-connected community.

1.2 Problem Statement

Surau Al Ansar in Taman Suria Tropika lacks the means to efficiently manage events, track financials, and communicate with the wider local community. The currently existing methods to spread information on issues and events are manual announcements at the religious centre and information sharing through WhatsApp, and they are likely to cause some people to miss out on what has been shared, thus leaving some members of the community uninformed about important activities and announcements.

Moreover, financial management is handled using the less than formal budgeting practice like using Microsoft Excel spreadsheets. It is admittedly functional, but far from optimal. Such a method can be very inconvenient and cumbersome when trying to manage and keep track of the money, more so for those unfamiliar with Excel, potentially leading to confusion and errors in tracking the surau's income and expenses.

Other than that, Surau Al Ansar and the rest of the community lack a unified platform where they can interact and participate in decision making processes, leading to limited engagement. The lack of a centralised system for communication and decision making creates inefficiencies that hinder the success to function as effectively as a community centre that should involve and collaborate more with each other. The surau and the community are also missing a streamlined method for booking its facilities for events, adding further complexity to their operations.

1.3 Scope

This project aims to develop a web-based management system and communication system for Surau Al Ansar and the Taman Suria Tropika community as a whole in Seri Kembangan, Selangor. The scope of the project is as follows:

i. Development of an event management module, enabling administrators to post and manage upcoming events.
ii. Integration of financial tools for tracking the surau's income and expenses, proving transparency and ease of financial management.
iii. Display of the surau's organizational information, including the surau's management structure and other relevant details.
iv. Development of a notification system to distribute important announcements and urgent updates to the community.
v. Implementation of an online facility booking system that will allow members to reserve the surau's facilities or certain location in the housing areas for events or activities, such as hosting a guest speaker for a khutbah or a Merdeka celebration.
vi. Development of a secure voting system for community members to participate in decision making.
vii. Ensuring the system functions as a communication hub for the Taman Suria Tropika community, encouraging greater engagement and promoting information sharing among residents.

1.4 Aims and Objective

The aim of this project is to develop a user-friendly and efficient web-based management and communication system for Surau Al Ansar, which will improve event management, financial tracking, decision-making processes, and communication within the Taman Suria Tropika community. To achieve this aim, the objectives of the project are as follows:

1. To study and analyse existing systems and methods used for event management, financial tracking, facility booking, voting and communication, in order to identify key features and areas for improvement.
2. To design and implement a web-based management and communication system for Surau Al Ansar and Taman Suria Tropika as a whole, incorporating modules for event

announcements, budget management, facility booking, communications, notifications, and voting.

3. To test and evaluate the functionality and usability of the developed system through user feedback and usability testing, ensuring that it meets the needs of the administrators and community members.

1.5 Brief Methodology

Software development is the process of creating, designing, testing, and maintaining software systems, programs, and applications. It involves using various programming languages, tools, and techniques to build software that performs specific functions, solves problems, or provides services. While the process can be complex and demanding, following a systematic approach ensures smoother and more successful development. As a result, applying a structured software development methodology is a crucial element in this project. To be more explicit, in the endeavour to satisfy all the necessities and requirements requested by the client in developing the system, Rapid Application Development (RAD) has been selected as the methodology to be used for this project.

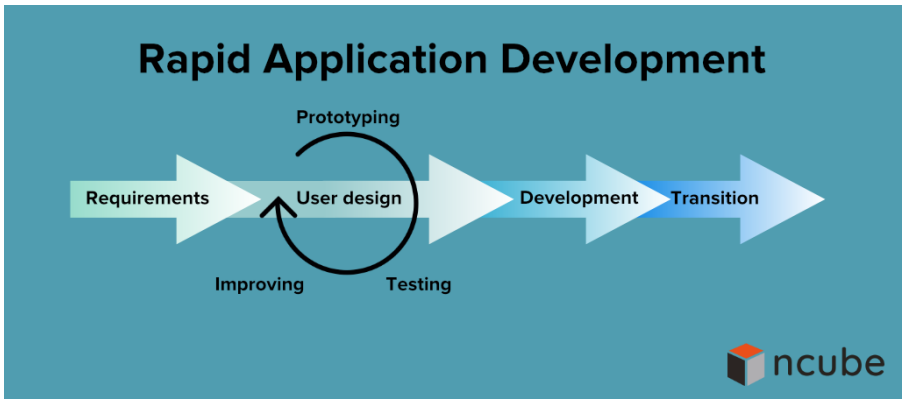


Figure 1.1: Rapid Application Development Methodology Cycles (Demchenko, 2020)

Despite there being other potential methodologies to choose from, such as Waterfall and Agile, Rapid Application Development (RAD) has been determined to be the most appealing. One of the primary reasons why the Rapid Application Development (RAD) methodology is being favoured over the rest is because it would be fitting to utilise a methodology whose approach places a strong emphasis on how end users will interact with the system, prioritising continuous feedback from end users throughout the development process. RAD is driven by user interface needs and is known for its agility, flexibility, and ability to

deliver quick results through visual interface tools and pre-built modules (Kissflow, 2024). Moreover, RAD reduces the risk of project failure by delivering functional prototypes early, ensuring alignment with user needs and minimising costly revisions later in the development cycle (Kissflow, 2024). Since the goal of this project is to develop a system that will benefit the community, the iterative and feedback-driven approach of RAD makes it the most suitable methodology.

1.6.1 Phases in Rapid Application Development (RAD)

1.6.1.1 **Requirements Planning**

In this first phase of development, Rapid Application Development (RAD) differentiates itself from other software development methodologies. Unlike traditional approaches that usually require detailed specifications from end users, RAD puts a focus on gathering the broad requirements instead. This gives way to a more iterative process, allowing developers to take the time and opportunity to refine specific details at different points of development (Kissflow, 2024).

1.6.1.2 **User Design**

In this next phase, developers may go on to create prototypes. To be more concise, this phase is where iterative prototyping happens and where user feedback is crucial in order to refine the design. The developers will create a series of iterations with different functions and features, then showcase them to the clients who will share their feedback in return (Kissflow, 2024).

1.6.1.3 **Development**

Here in the third phase, developers are expected to construct and deliver a working product. Fortunately, a lot of the work has already been done in the last phase, such as designing, testing, and developing, so this phase might take less time. But, such as the last phase, feedback from the clients are still welcomed (Kissflow, 2024).

1.6.1.4 **Transition**

In the final phase, the developed system is deployed into a live production environment where thorough testing and user training take place to ensure smooth operation (Demchenko, 2020). This stage includes comprehensive testing to verify scalability and performance, while also documenting technical details for future reference.

During this process, developers focus on identifying and fixing any remaining issues, implementing final customisations, and simulating real-world conditions. Additionally, teams dedicate time to debugging, applying updates, and completing essential maintenance tasks before officially launching the system (Kissflow, 2024).

1.6 Significance of Project

This project ensures that there will be a system which addresses several operational and communication challenges faced by Surau Al Ansar and the wider Taman Suria Tropika community. With the help of a web-based management and communication system, Surau Al Ansar will be able to operate more efficiently. The system will streamline operations, reduce reliance on manual or difficult methods, and enhance not only the surau's engagement with the Taman Suria Tropika community but also strengthen the connections within the entire neighborhood. Additionally, the inclusion of a facility booking system and voting system will not only enhance the surau's functionality but also promote greater community participation and better coordination of shared spaces. All in all, this project ultimately strengthens both the surau's management and the community's connection.

1.7 Project Schedule

Refer to the Appendices A, B, C and D to see the project schedule.

1.8 Expected Outcome

The expected outcome of this project is the successful development and deployment of the web-based management and communication system made to meet the requirements and needs of not only Surau Al Ansar but also the wider Taman Suria Tropika community. The system will help streamline tasks and operations like managing events, keeping track of financials, and facilitating communication. As a result, Surau Al Ansar will experience improved operational efficiency, enabling easier management of events, better financial reporting, and more effective information dissemination. The rest of the Taman Suria Tropika community will also possess better means of communicating and engaging with each other.

Moreover, the facility booking system will allow both the surau and the community to efficiently reserve spaces and facilities in the neighbourhood. For instance, the surau or the wide playground area located in there, ensuring smoother coordination and planning. Meanwhile, the voting system will enable the community members and surau attenders to participate in decision making processes, which will foster a greater sense of ownership and engagement in both surau and community affairs.

By centralising these functionalities into a single digital platform, this project will be able to create a more well-connected and interactive community, improving communication, participation, and collaboration amongst the Taman Suria Triopika residents. All in all, this system will not only improve the operations of Surau Al Ansar, but also serve as a very beneficial communication hub for the entire neighbourhood.

1.9 Summary

In this chapter, the Smart Community Platform for Taman Suria Tropika gets introduced. This system aims to address the operational and communication challenges faced by Surau Al Ansar and the wider Taman Suria Tropika community through a web-based management and communication system. Currently, the surau relies on inefficient methods for event management, financial tracking, and communication, while lacking a centralised platform for decision-making and facility booking. By utilising the Rapid Application Development (RAD) methodology, the system will incorporate modules for event announcements, financial management, facility booking, notifications, and voting, while serving as a communication hub for the community. Said methodology prioritises user feedback and iterative prototyping, and its process has been outlined in four phases: Requirements Planning, User Design, Development, and Transition, ensuring a user-focused and agile development approach. The project's significance lies in streamlining operations, improving communication, and fostering greater community participation. Expected outcomes include enhanced operational efficiency for Surau Al Ansar, better community engagement, and a strengthened sense of collaboration among residents.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, Chapter 2, the examination of three different existing systems that possesses similar intended features as the proposed application will be carried out, alongside the exploration of the tools and technologies that will be in use. The detailed review of such similar systems will help identify their strength and limitations, thereby presenting useful information that can be used to facilitate this project. Furthermore, the tools and technologies needed for developing the system will be looked into and undergo evaluation to make sure that they will be the most appropriate to be utilized.

2.2 Overview of Objectives

The objectives of the Literature Review section are as follows:

1. Review similar existing systems before comparing them with each other and the proposed system in order to identify their strength and limitations so that advantageous knowledge that can further improve the proposed system can be obtained.
2. Analyse the tools and technologies utilized by current systems and assess their relevance to the design and development of the proposed system.
3. Justify why it is necessary to develop Smart Community Platform for Taman Suria Tropika with the features that it is meant to possess.

2.3 Review on Similar Existing Systems

2.3.1 Management Information System for the Jami Al-Kautsar Mosque

The Management Information System for the Jami Al-Kautsar Mosque was designed and developed by Sri Rezeki Candra and Pratseyo Linuwih, two people who are affiliated with the Informatics Engineering Program at the Faculty of Engineering, Universitas Pancasila, Jakarta, aimed at improving efficiency, transparency, and accountability in mosque operations (Nursari & Linuwih, 2021). They created this system in order to address the challenges faced by the mosque when it comes to managing mosque operations, such as disorganized records and inefficiencies in managing activities, finances, and inventory. Here is a look into the core features, arranged into a table for better clarity:

Table 2.1: Core features and descriptions (Nursari & Linuwih, 2021)

Features	Description
User Authentication (Login)	User can login into the system. The user will only be directed to the home page once the authentication process is successful.
Activity Management	Enables proper documentation of events and prevents conflict in schedule.
Financial Management	Provides the ability to manage income and expenses, including generating transparent financial reports.
Rental Management	Helps make the process of renting facilities easier while keeping track of the details.
Inventory Management	Aid in keeping track of inventory inflows and outflows.
User Management	Keep records of mosque administrators and congregation members.
Reporting System	Generates detailed reports for activities, financials, attendance, and asset usage.

The homepage of the Jami Al-Kautsar Mosque Management Information System serves as a central dashboard, where information such as the mosque's final financial balance, expenses for the current month, and the total number of congregation members are displayed. Other than that, it includes a calendar feature that highlights upcoming activities, allowing users to view and be informed of the mosque's financial status and planned events (Nursari & Linuwih, 2021).

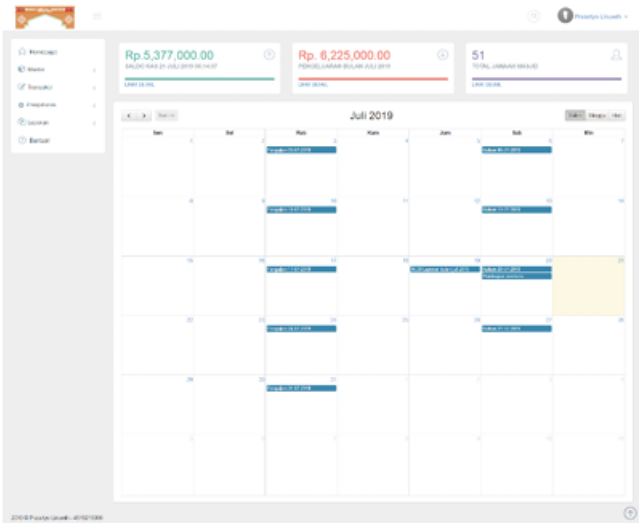


Figure 2.1: Display of Homepage (Nursari & Linuwih, 2021)

The system includes a user management feature, which allows administrators to manage transaction data effectively. This module is equipped with functionalities for adding, editing, and deleting transaction records (Nursari & Linuwih, 2021).

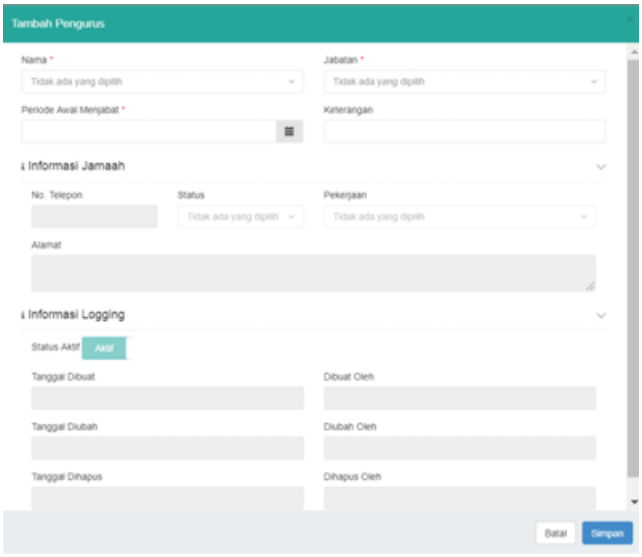


Figure 2.2: Display of Management Input of Transactions (Nursari & Linuwih, 2021)

Another notable component is the activity management feature. This module enables administrators to manage data related to various mosque activities. It provides functionalities for adding, modifying, and deleting activity records (Nursari & Linuwih, 2021).

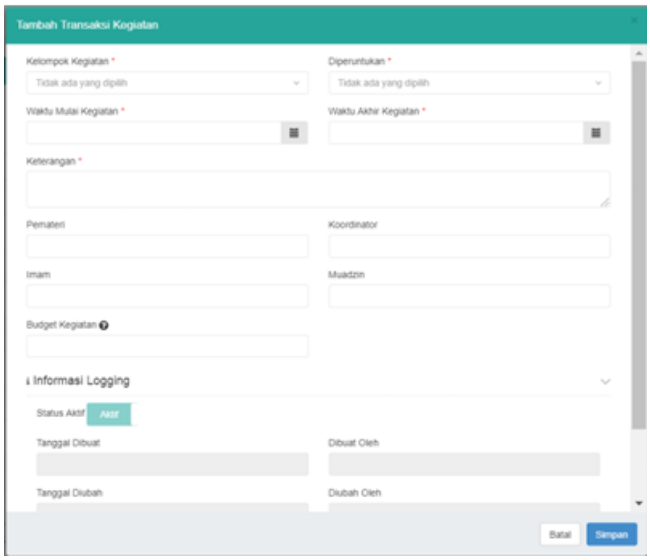


Figure 2.3: Display of Input of Activity Transaction (Nursari & Linuwih, 2021)

There is also the rental management feature which makes it possible for administrators to manage rental transactions for facilities and infrastructure. This includes tracking the use of mosque inventory and resources that are rented out to the community. The feature provides capabilities for adding, modifying, and deleting transaction records, making sure that there is an up-to-date information on facility rentals (Nursari & Linuwih, 2021).

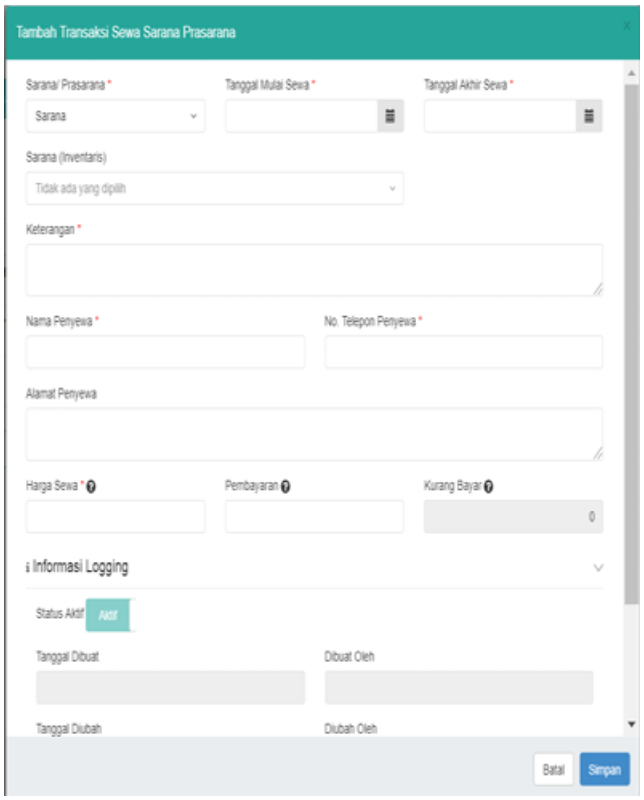


Figure 2.4: Display of Input of Rental Transactions (Nursari & Linuwih, 2021)

The financial management feature supports the management of the mosque's financial activities, including income from donations, rental payments, and other sources, as well as expenses related to mosque operations. This module offers functionalities to add, edit, and delete financial transaction records, promoting effective financial oversight and transparency (Nursari & Linuwih, 2021).

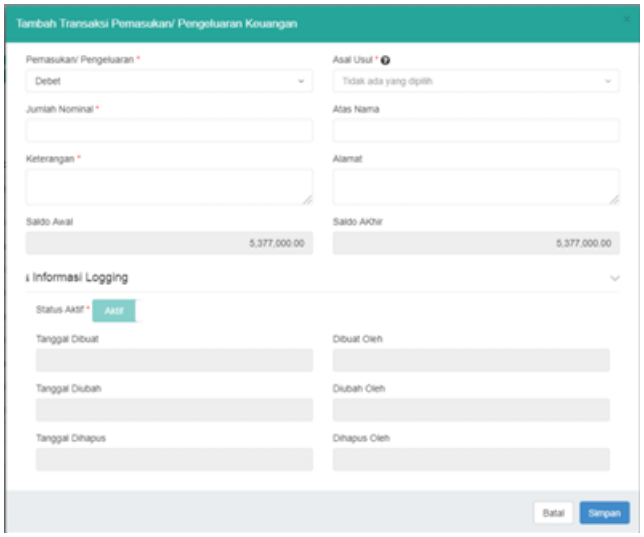


Figure 2.5: Display of Input of Financial Income and Expense Transaction (Nursari & Linuwih, 2021)

The inventory management module is designed to handle the management of goods-related transactions. It allows administrators to track inventory inflows and outflows, ensuring proper documentation of items received or distributed. The feature supports the addition, modification, and deletion of transaction data, enabling efficient inventory management (Nursari & Linuwih, 2021).

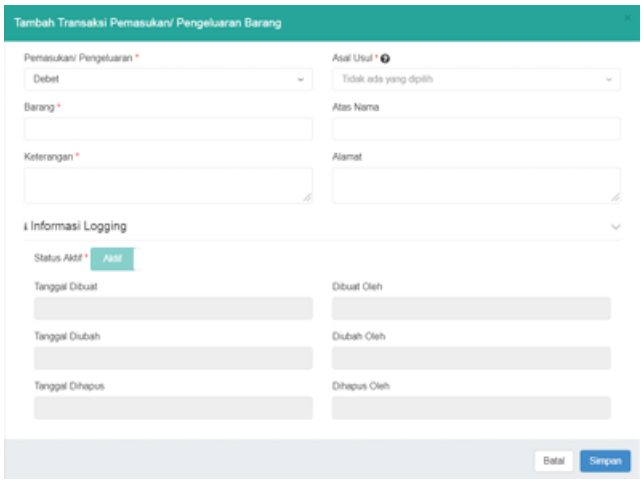


Figure 2.6: Display of Input of Transaction for the Entry and Issuance of Goods (Nursari & Linuwih, 2021)

2.3.2 Mosque Management Information System for FELCRA Bukit Kepong

The Mosque Management Information System was designed and developed by Muhammad Norman Hakim Talib and Rozlini Mohamed from the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia (UTHM), in order to improve the management and operations of the Masjid Jamek Felcra Bukit Kepong located in Felcra Bukit Kepong, Muar, Johor. According to their investigations, the mosque committee members had been managing their operations through outdated means, where everything was done manually. For instance, matters like records, schedules, services, information on staff members, finances, and facilities were either kept in handwritten records or Excel spreadsheets (Talib & Mohamed, 2023).

Moreover, the system also serves to assist in overcoming communication issues faced by the mosque, such as depending on verbal announcements after Friday prayers and advertising posters in order to spread information. These methods were most likely not very effective when it comes to spreading word to everyone (Talib & Mohamed, 2023). Additionally, the system was also made to assist in facilitating fundraising efforts by extending the mosque's reach through online means. The matter of collecting money for activities and maintenance had been an issue for the mosque, because they were only relying on donations boxes to do so, thus were limited only to people who would come to the mosque to donate in person (Talib & Mohamed, 2023).

Overall, the system streamlines the mosque's management, improves operations, enhances accessibility, and made it possible for the community to be more involved with matters pertaining to the mosque, making it a very important and useful tool for a mosque in these modern times. Here is a look into the core features, arranged into a table for better clarity:

Table 2.2.2: Core features and descriptions (Talib & Mohamed, 2023)

Features	Description
Login and Registration Module	Enables users to register an account and login. Modified with error handling and will display alerts for invalid login attempts.

Donation Module	Makes it possible for the mosque committee members to spread information in regard to funding assistance. Every donation detail will be displayed and viewable here. Moreover, this module will also make donation via online banking possible.
Duty Roster Module	Here, the mosque committee members can edit and publish duty schedules. The congregants are also allowed to view the schedules in order to remain informed.
Event Module	With this, the system should allow mosque committee members to insert and manage event details, then have it displayed for the congregation to see.
Expenses Module	Enable the mosque committee members to input and display financial data, such as expenditures. The congregation are also allowed to view the mosque's financial information.
Feedback Module	Congregants can use it to submit feedback or comments. Meanwhile, the mosque committee members are also allowed to review those feedback/comments.
User Management Module	Through this, the system should display all information on registered users. Besides that, administrator also has the ability to edit and delete user data.

The system can be operated by a user through the use of three different types of users, which are mosque committee, administrator, and congregation. The new congregants can register into the system by clciking the register button. After obtaining an account, they are given access to several features, such as the donation module, event module, duty roster

module, and the feedback module. The donation module lets them view details on funding request from the mosque and contribute through online donation. Other than that, the event module presents to them details on upcoming activities related to the mosque. Besides that, the duty roster module is there so that they can remain informed on committee schedules and responsibilities. Furthermore, the feedback module provides them the opportunity and option to voice their thoughts and suggestions to the mosque committee (Talib & Mohamed, 2023).

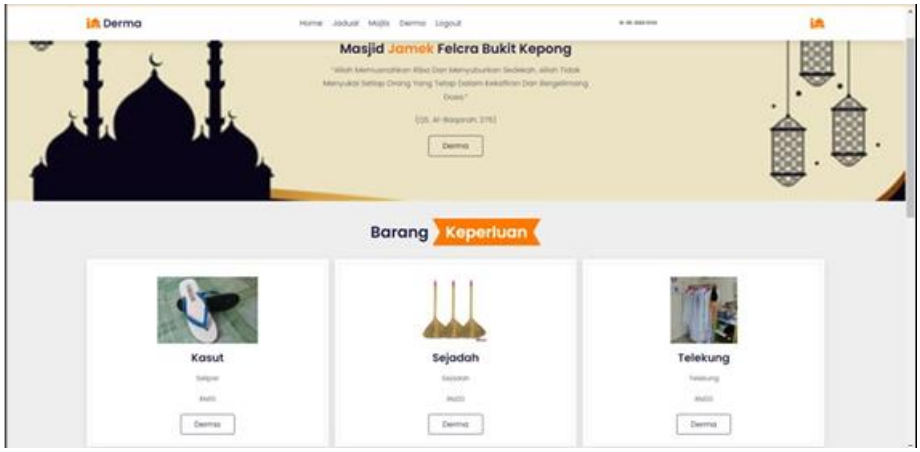


Figure 2.7: Donation Module for the Congregation (Talib & Mohamed, 2023)

Figure 2.8: Event Module for the Congregation (Talib & Mohamed, 2023)

As for the mosque committee members, they can login as two different types of users, which are the treasurer and the secretary. With access to the treasurer account, they can update, record, and delete the mosque's expense data through the expense module. They may also view all information pertaining to the expense. Moreover, they also have access to

the donation module, where they can manage and oversee all donation records and donor details (Talib & Mohamed, 2023).

Figure 2.9: Expenses Module for the Treasurer (Talib & Mohamed, 2023)

Figure 2.10: Donation Module for the Treasurer (Talib & Mohamed, 2023)

The other type of user, secretary, affords them the option to manage event information through the use of the event module. The secretary can either edit or delete the information on posted events. Other than that, they can also manage the duty roster of the mosque committee (Talib & Mohamed, 2023).

Figure 2.11: Event Module for the Secretary (Talib & Mohamed, 2023)

Figure 2.12: Duty Roster Module for the Secretary (Talib & Mohamed, 2023)

The last user option is administrator. With the role, a committe member can edit or delete all the information of a user. Besides that, they can also view the feedback given by other users (Talib & Mohamed, 2023).

2.3.3 Mosque Financial Management Portal

The web-based Mosque Financial Management Portal was designed by Fadelis Sukya and Ellya Nurfarida from the Department of Information Technology, Malang State Polytechnic, Malang, Indonesia, in order to assist in addressing the challenges of transparency and efficiency in mosque financial management (Sukya & Nurfarida, 2024). By forgoing the old, traditional manual methods, like using whiteboards as tool to present reports, to a modern web-based solution, the system ensures the accurate recording and reporting of financial data (Sukya & Nurfarida, 2024). Assuredly, it will also assist in enhancing the accountability of mosque administrators and foster trust amongst the congregates. Here is a look into the core features, arranged into a table for better clarity:

Table 2.3: Core features and descriptions (Sukya & Nurfarida, 2024

Feature	Description
Financial Management	Provides the ability to manage income (e.g., donations, alms, qurban receipts) and expenses.
Announcement Module	Administrators can use the announcement page to send WhatsApp notifications. To be more concise, they can use this feature in order to send messages or announcements, attach PDFs or images, and even select the recepients.

The dashboard displays an organized interface for the users. Through it, they can directly gain access to features such as donations, zakat, qurban, and other financial management tools, ensuring ease of navigation and efficient use of the system (Sukya & Nurfarida, 2024).

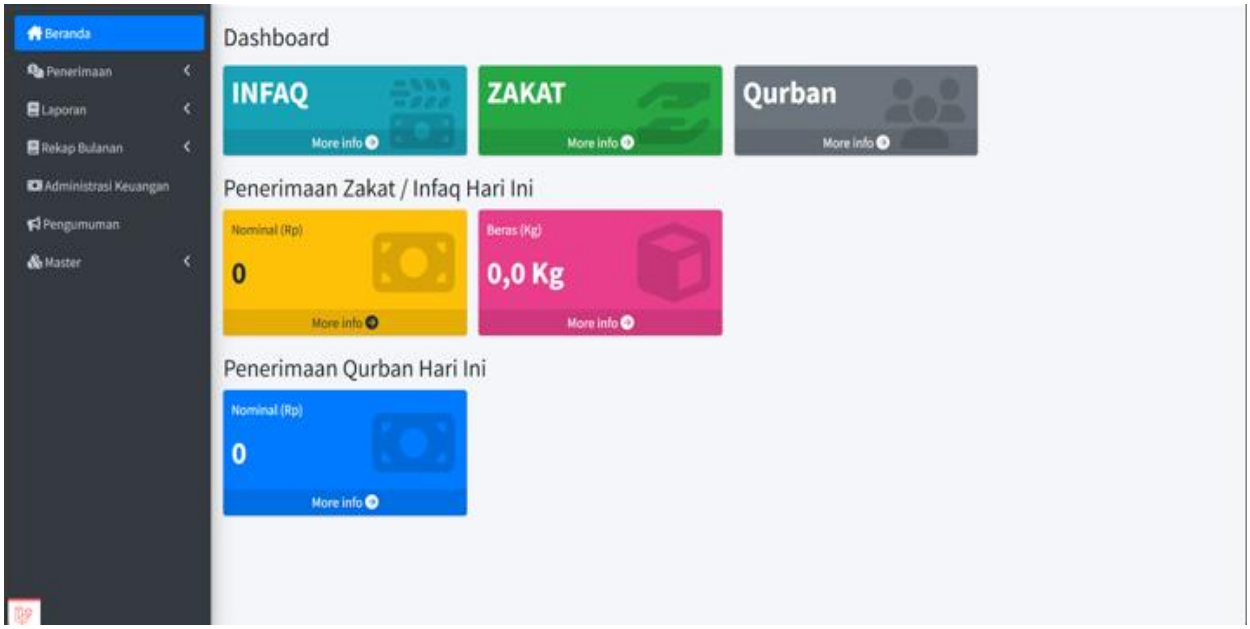


Figure 2.13: Dashboard (Sukya & Nurfarida, 2024)

The configuration page is dedicated to the task of manaing the mosque's profile and data of the officials (takmir) who will use the application. When a mosque account is first added, what needs to be done is to configure key details, such as takmir information, the mosque's WhatsApp contact number, header photo, and address. These details will enable

the system to send notifications to congregants when making donations or zakat payments (Sukya & Nurfarida, 2024).

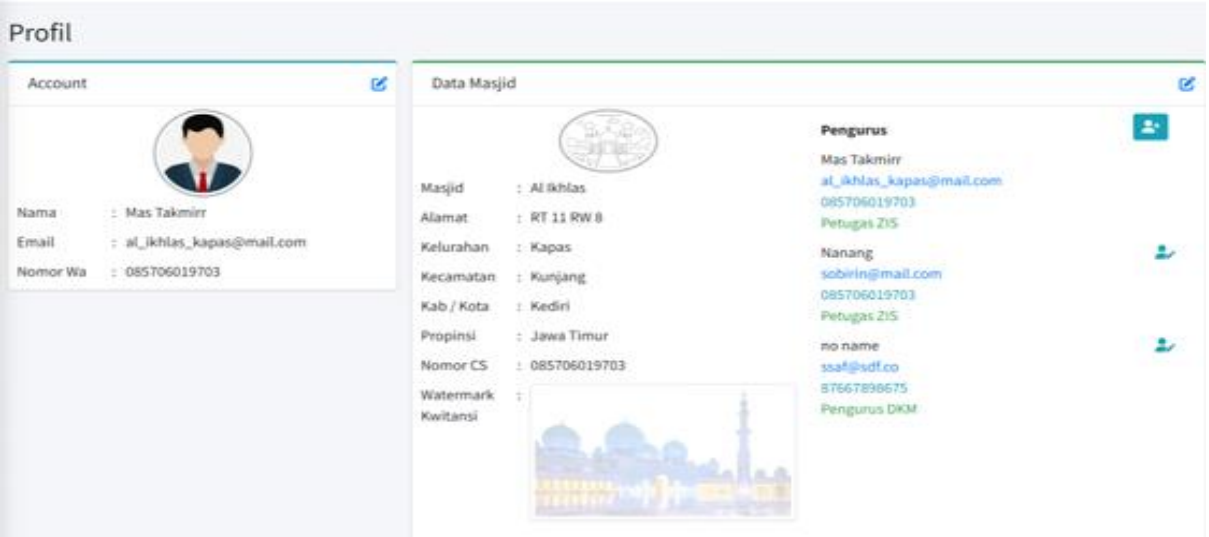


Figure 2.14: Mosque configuration (Sukya & Nurfarida, 2024)

The donation page facilitates the management of donation receipts from congregants/donors. Users can add, edit, and delete donation data efficiently, thus ensuring accurate tracking of financial contributions. Furthermore, the zakat, qurban, and expenditure pages hold the same mechanism and function (Sukya & Nurfarida, 2024).

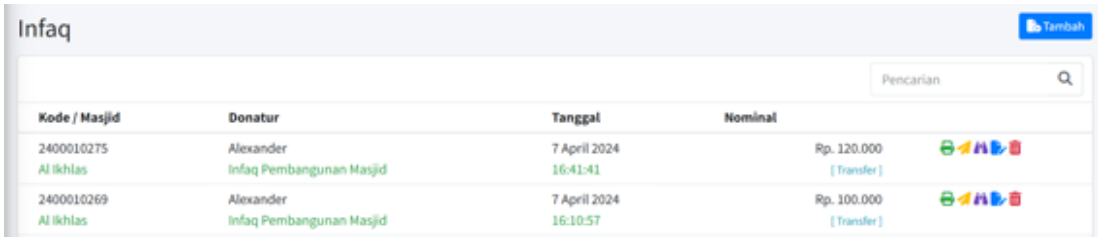


Figure 2.15: Infaq page (Sukya & Nurfarida, 2024)

Figure 2.16: Zakat page (Sukya & Nurfarida, 2024)

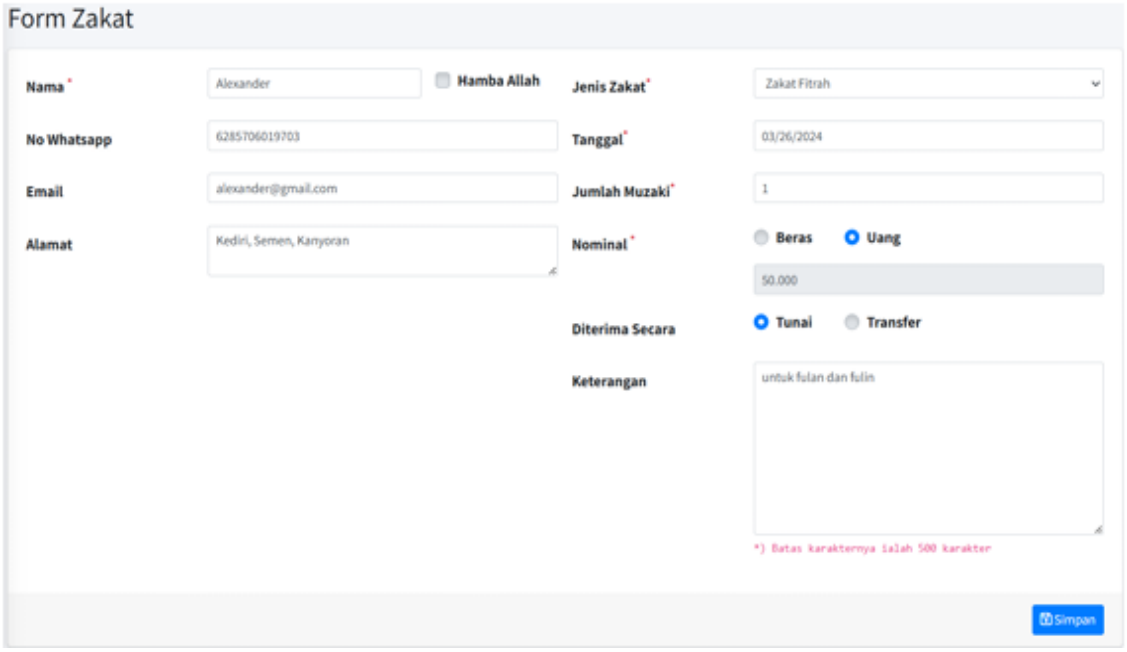


Figure 2.17: Zakat form (Sukya & Nurfarida, 2024)

Figure 2.18: Qurban process (Sukya & Nurfarida, 2024)

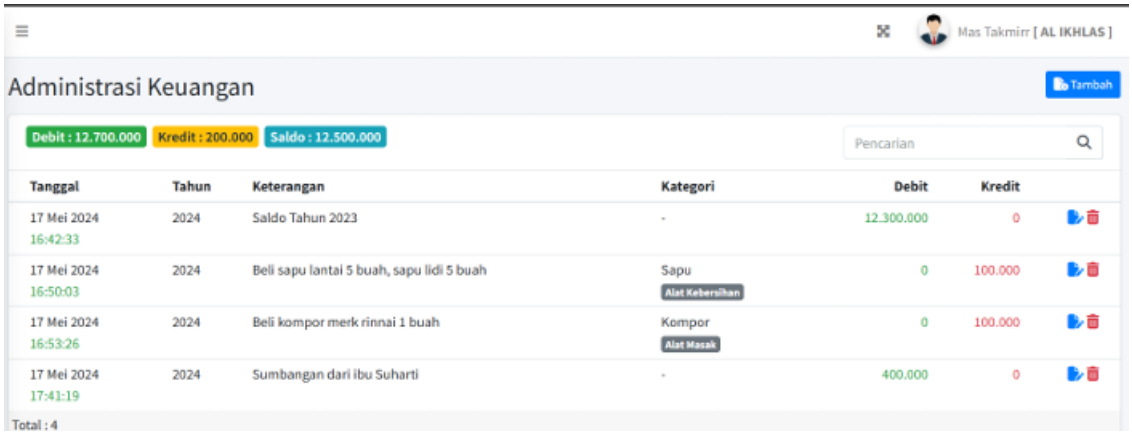


Figure 2.19: Qurban data and receipts to congregants (Sukya & Nurfarida, 2024)

The announcement page is used to allow a form of communication, where it can be used to send Whatsapp notifications to selected recipients. The user can

customize their messages, attach files in PDF or image formats, and distribute announcements effectively (Sukya & Nurfarida, 2024).

Figure 2.20: Announcement (Sukya & Nurfarida, 2024)

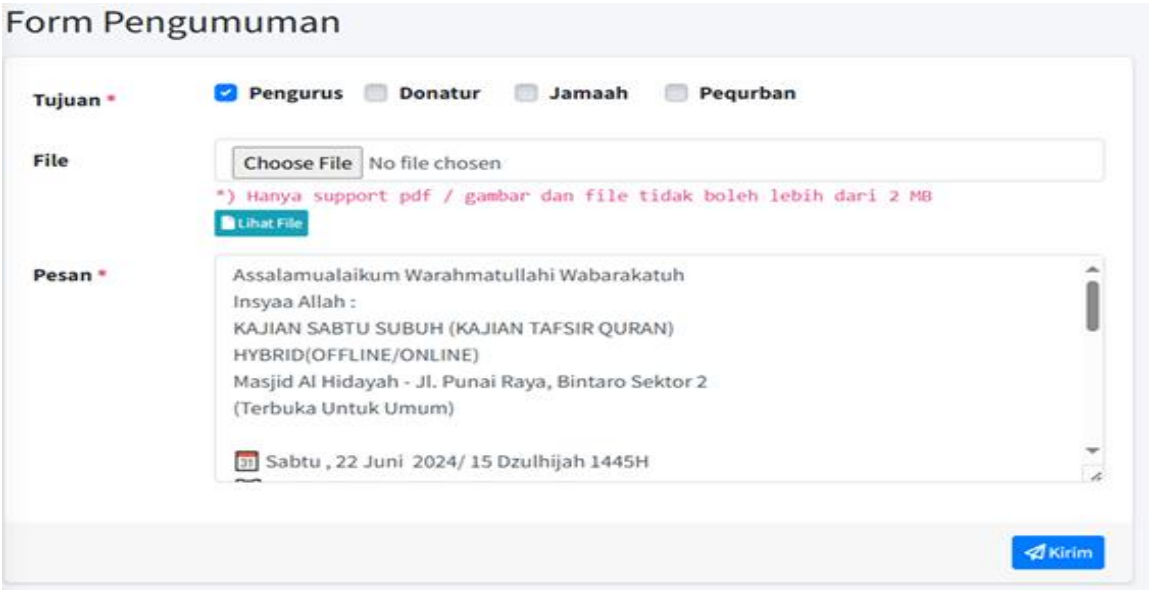


Figure 2.21: Form to create announcement (Sukya & Nurfarida, 2024)

2.3.4 Table of comparison between Systems

Table 2.4: Comparison between systems

Features / System	Management Information System for the Jami Al-Kautsar Mosque	Mosque Management Information System for Felcra Bukit Kepong	Mosque Financial Management Portal	Smart Community Platform for Taman Suria Tropika (Proposed System)

Login and Registration Module	✓ (Only login)	✓	✓	✓
User Management	✓	✓	×	✓
Financial Management (e.g., expense, income, and donation tracking)	✓	✓	✓	✓
Event/Activity Management	✓	✓	×	✓
Rental Management	✓	×	×	✓
Inventory Management	✓	×	×	×
Feedback Module	×	✓	×	×
Duty Roster Module	×	✓	×	×
Reporting System	✓	×	×	×
Announcement/Notification Module	×	×	✓	✓
Voting Module/System	×	×	×	✓
Communication Hub	×	×	×	✓

Based on what has been reviewed and analysed, there are several core features across the discussed systems, such as login and registration modules, as well as financial

management capabilities. Certain systems, like the Management Information System for Jami Al-Kautsar Mosque and the Mosque Management Information System for Felcra Bukit Kepong, also incorporate noteworthy features like event/activity management, user management, and even inventory and duty roster modules that should probably be present in all similar systems due to how useful they are.

However, although the existing systems demonstrate notable advancements in certain areas, they still lack key features and functionalities that are essential for addressing more diverse community needs. None of the reviewed systems provide an all-encompassing platform that not only focuses on administrative matters, but also the community-focused functionalities.

The proposed Smart Community Platform for Taman Suria Tropika seeks to address these issues by providing an integrated solution. It aims to incorporate modern features such as a voting module for democratic decision-making and a communication hub to foster community engagement. Enhancements to existing functionalities, such as financial and event management, will also be prioritized to ensure better usability and transparency. By uniting these features into a single platform, the proposed system will hopefully be able to support both mosque operations and broader community needs, making it a comprehensive and user-centric solution.

2.4 Review on Tools and Technology

This section examines the tools and technologies primarily utilized in the systems reviewed earlier, namely the Mosque Management Information System for Felcra Bukit Kepong and the Mosque Financial Management Portal. These systems demonstrate the strategic use of modern frameworks, programming languages, and database technologies to address the unique challenges of mosque management. Notably, the technologies they employ align closely with the requirements of effective and scalable web-based solutions, making them highly relevant as reference points for developing similar systems, namely the proposed system for this project.

2.4.1 Software

2.4.1.1 **Laravel Framework**

Laravel is known as an advanced PHP framework made to enhance software quality and facilitate in development by making it simpler. It possesses elegant syntax and offers very useful and convenient features such as built-in security, simplified authentication, robust routing, and powerful database handling. Furthermore, it provides tools that enables efficient password management, encryption, and validation, proving itself to be a very excellent choice when it comes to creating secure and scalable systems (Amini et al., 2021).

Further mentions in regard to Laravel by Amini et al. (2021), states that the framework's design also align well with the needs of small-to-medium projects, putting a focus on ease of adoption and cost efficiency. Its support for Model-View-Controller (MVC) architecture helps to ensure that codes be more organized, aiding in better collaboration and maintainability during development. To further elaborate on this, the architectural approach divides the application into three core components: the Model, which is responsible for handling data and business rules; the View, which takes care of the user interface; and the Controller, which functions as a bridge between the Model and the View. By organizing the code in this way, it encourages cleaner and more manageable code while improving teamwork, as developers can focus on different parts of the application without stepping on each other's toes. Additionally, the MVC design supports scalability and simplifies maintenance, making it easier to adjust and update the application as requirements change (Setyawan et al., 2023).

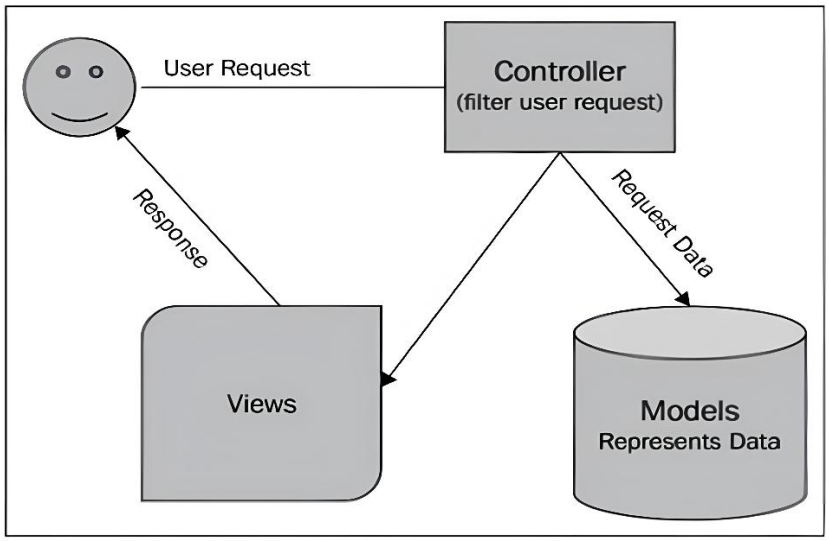


Figure 2.22: MVC paradigm (Setyawan et al., 2023)

All in all, findings indicates that Laravel's utilization can be credited to its relative advantages, compatibility with existing technologies, and ability to reduce development complexity, which makes it suitable for the objectives of my project (Amini et al., 2021).

2.4.1.2 **PHP**

PHP is among one of the most popular and time-tested languages for web-projects, known for being a simple yet very versatile language that is constantly evolving. It has become a welcomed and widely used scripting programming language due to cross-platform compatibility with all major operating systems, high processing speed, allowing developers to easily modify, extend, or remove functionality, offering flexibility for evolving project needs (Sotnik, Manakov, & Lyashenko, 2023). There was also a mention by Sotnik, Manakov, and Lyashenko (2023), that PHP integrates well with MySQL, which is very fortunate since there is a plan on utilizing the database management system. Said compatibility also has the benefit of ensuring efficient server-side development, enabling smooth implementation of essential features like user authentication, secure data handling, and personalized content delivery. Furthermore, PHP's ability to quickly change and update code without significant overhead makes it particularly advantageous for iterative development work processes (Sotnik, Manakov, & Lyashenko, 2023).

2.4.1.3 **MySQL**

The database management system, MySQL, is quite popular among most web programmers and is widely used in conjunction with PHP in order to build data-driven web

applications. It is also known as a tool that gets actively used for dynamic web projects, where things get even more interactive depending on the user and what they need from it. Major factor to its fame and encouragement of use can be credited to how efficient it is when it comes to the matter of storage, retrieval, and manipulation of data. Furthermore, its reliable security system and its quality of being a multi-platform open-source database platform makes it useable for both small and large-scale applications (Sotnik, Manakov, & Lyashenko, 2023).

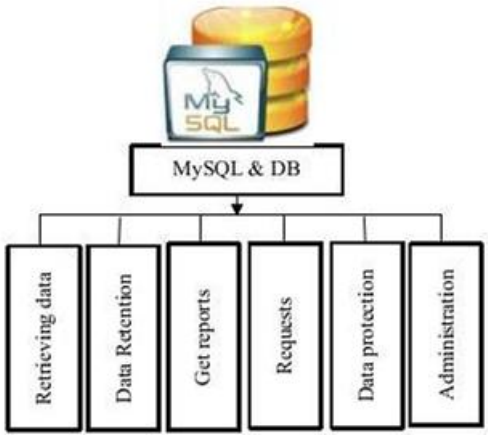


Figure 2.23: Basic MySQL functions (Sotnik, Manakov, & Lyashenko, 2023)

Overall, MySQL is a very excellent choice when it comes to selecting a database management system for a project, and its compatibility with modern development environments and tools, including integration with PHP frameworks like Laravel, as I have picked up from my review of the other systems, further solidifies its position as an ideal choice for my proposed project. Altogether, these features ensure that MySQL not only meets the project's current requirements but also provides the flexibility to scale and evolve as future needs arise.

2.5 Summary

Chapter 2 reviews existing systems and evaluates tools and technologies relevant to the development of the Smart Community Platform for Taman Suria Tropika. The review highlights features such as login, registration, financial management, event/activity management, and user management, which are common across current systems like the Mosque Management Information System for Felcra Bukit Kepong. However, these systems

lack an integrated platform that addresses both administrative tasks and community-focused functionalities. The proposed platform already has plans to incorporate features such as a voting module, a communication hub, and enhanced management tools to address these gaps. Regardless, the review of the three similar, existing systems had been very informative.

Moreover, the review also explores the tools and technologies that will support the system, focusing on the Laravel framework, PHP, and MySQL. Laravel's MVC architecture promotes code organization, collaboration, and scalability, making it suitable for the project. PHP's cross-platform compatibility and versatility, combined with MySQL's efficient data handling and security features, further enhance the proposed system's development. These tools are well-suited to meet the project's current and future needs, ensuring a scalable and secure solution.

CHAPTER 3: REQUIREMENT ANALYSIS & DESIGN

3.1 Introduction

The purpose of this chapter is to delve into the details pertaining to the requirement analysis and design process for developing the Smart Community Platform for Taman Suria Tropika. The development approach is guided by the Rapid Application Development methodology, and its illustration here will make clear for why it has been chosen for its iterative and user-centric nature. Furthermore, this structured approach will provide a clear foundation for the coding and development phase, making the process more efficient and manageable while also ensuring that everything remains relevant and effective for the Taman Suria Tropika community.

3.2 Rapid Application Development

The Rapid Application Development (RAD) methodology was selected for the development of the Smart Community Platform for Taman Suria Tropika due to its iterative and user-centric approach. Unlike traditional methods, RAD emphasizes gathering broad requirements early in the process, allowing for flexibility and refinement throughout development. This methodology focuses on rapid prototyping and continuous user feedback, making it particularly suited for a community-oriented project like this one.

Among the available methodologies, such as Waterfall and Agile, RAD stands out as the most suitable for this project. It emphasizes user interaction and prioritizes continuous feedback throughout the development cycle. RAD's flexibility and agility allow for the quick delivery of functional prototypes, ensuring alignment with user needs and reducing the risk of costly revisions later. By focusing on user interface design and iterative improvements, RAD delivers quick results while maintaining high adaptability (Kissflow, 2024).

The RAD process consists of four key phases. The Requirements Planning phase focuses on gathering broad objectives and identifying core features, leaving room for adjustments as the project evolves. In the User Design phase, prototypes are created and refined based on stakeholder feedback, ensuring that the final system is user-friendly and meets the community's needs. The Development phase involves building the working system, leveraging insights from earlier phases to streamline construction and align with user expectations.

Finally, the Transition phase deploys the system in a live environment, involving rigorous testing, user training, and final customizations to ensure smooth operation.

By emphasizing adaptability and continuous improvement, the RAD methodology minimizes risks and ensures that the platform aligns with user requirements. This approach not only accelerates development but also delivers a system that effectively meets the needs of both Surau Al Ansar and the Taman Suria Tropika community.

3.2.1 Planning

This first phase, the planning phase, is a pivotal part of the development process, as it lays the foundation for the entire project. During the duration of this stage, initial requirements and useful information will be gathered through a combination of interviews and personal observations. Interviews will be conducted with selected key stakeholders, including the secretary of Surau Al Ansar, Mr. Azlan Ithnin, and an active community member, Ms. Amirah, to gain insights into their challenges, expectations, and desired functionalities. These interviews will focus on identifying issues with current methods of communication, event management, financial tracking, and facility bookings.

Additionally, as a resident of Taman Suria Tropika, I will incorporate my own research, observations, and experiences to further inform the system's design. This approach ensures a more comprehensive understanding of the community's needs, as it combines first-hand knowledge with input from other stakeholders.

By combining stakeholder input with personal experience and thorough analysis, the planning phase ensures that the proposed platform is aligned with the community's expectations and addresses their specific challenges effectively. This comprehensive approach lays the groundwork for the subsequent phases of the Rapid Application Development methodology.

3.2.2 Analysis

In this second phase of the RAD methodology, an extensive analysis is conducted to thoroughly examine the current methods, system requirements, and proposed solutions. This phase is critical for identifying inefficiencies in the existing processes and ensuring that the proposed system aligns with user needs and project objectives.

The analysis involves evaluating the current methods used by Surau Al Ansar and the broader Taman Suria Tropika community, understanding their limitations, and gathering insights through interviews and personal observation. Furthermore, the software and hardware requirements for the system are defined, providing a clear understanding of the technical foundation needed for development.

This phase also includes the initial design and prototyping efforts, leveraging tools such as UML, and creating wireframes to visualize the system's structure and functionality. Each subsection in this phase contributes to building a comprehensive blueprint for the proposed Smart Community Platform.

3.2.2.1 **Analysis on Current Method**

Currently, Surau Al Ansar currently rely on a combination of manual processes and basic tools to manage their operations. Event announcements are primarily disseminated through WhatsApp groups or made verbally during prayers. While these methods are familiar and widely accessible, they often lead to inconsistencies and missed information, leaving some community members uninformed about important activities.

Other than that, the method of financial management is another area where inefficiencies are can be seen. The surau tracks income and expenses using Microsoft Excel spreadsheets. Although it works, this approach is prone to human error and lacks transparency, especially for individuals who are not proficient with the software. This can lead to confusion and difficulty in maintaining accurate financial records.

In addition to these challenges, there is no formalized system for booking facilities or gathering community feedback or for the community to communicate with each other better. Facility reservations, such as those for the park or the large pavilion or surau, are managed through ad hoc arrangements, which can cause scheduling conflicts and inefficiencies. Similarly, the absence of a structured feedback mechanism like voting limits the surau's ability to gauge community opinions and involve members in decision-making processes. Also, the community could do well to have a centralized communication hub where they can share information, either it be for issues that the community should be aware of or a simple yet useful thing like wanting make sales within the community itself.

These current methods, while cost-effective, have significant limitations. The lack of a centralized platform hinders effective communication and coordination, making it challenging to ensure that everyone is informed and engaged. Manual processes also increase the likelihood of errors and reduce overall efficiency in managing the surau's activities and finances.

Such inefficiencies highlight the urgent need for a robust, centralized system that can streamline operations and foster better engagement within the community. By addressing these issues, the proposed Smart Community Platform aims to improve the surau's operations while creating a stronger and more connected community in Taman Suria Tropika**.**

3.2.2.2 **Analysis on Proposed Application**

The Smart Community Platform for Taman Suria Tropika is designed to address the inefficiencies and challenges faced by Surau Al Ansar and the broader community, as identified in the analysis of the current methods. The proposed system introduces a centralized, user-friendly platform that integrates key functionalities to streamline operations and enhance community engagement. The key features of the proposed system:

a) Event Management Module
b) Financial Management Module
c) Facility Booking Module
d) View Organization Information.
e) Voting Module
f) Notification Module
g) Communication Hub Module

The Smart Community Platform is specifically tailored to meet the needs of both the surau administration and the Taman Suria Tropika community. Its centralized nature eliminates the inefficiencies of needing to rely on different, separate tools and manual processes, improving operational efficiency. All in all, by integrating features like financial tracking, facility booking, and a voting system, the platform promotes transparency, accountability, and inclusivity, alongside making sure that sure that the surau administrators will have an easier operation henceforth.

3.2.2.3 **Interview and Personal Observation**

Interviews were conducted with two key stakeholders to gain a deeper understanding of their challenges, expectations, and requirements for the proposed system: Mr. Azlan Ithnin, the secretary of Surau Al Ansar, and Ms. Amirah, an active member of the Taman Suria Tropika community. Their insights helped shape the system's features and priorities. The questions asked and the answers collected can be found under Appendix C.

The interviews provided valuable insights into the community's needs:

- Both stakeholders emphasized the importance of different modules, that are Budget Management and the Communication Hub, for different reasons.
- Facility Booking System was seen as somewhat useful by Ms Amirah, but not so much by Mr. Azlan Ithnin. So, it is a secondary priority.
- Current reliance on tools like WhatsApp, Facebook, and Excel creates inefficiencies that the proposed system aims to address.

By integrating these findings into the system design, the proposed Smart Community Platform for Taman Suria Tropika can effectively address the unique needs of Surau Al Ansar and the community as a whole.

3.2.2.4 **Software Requirements**

When it comes to the matter of software tools required to be utilised for the development of the system, it is something which had mostly already been discussed extensively in Chapter 2. Based on the findings in Chapter 2, the following software components will be utilized:

Table 3.1: Software Requirements

Software	**Reasoning/Description**
Laravel Framework	The Laravel framework is central to the development of this project. It offers an advanced PHP framework designed to enhance software quality while simplifying

	the development process. Its elegant syntax and robust feature set, including built-in security, simplified authentication, and powerful database handling, make it a suitable choice for creating secure and scalable systems (Amini et al., 2021). Furthermore, its Model-View-Controller (MVC) architecture promotes a more organized and maintainable coding, which will help keep track and doing work much easier and efficient.
PHP	PHP, a versatile and time-tested scripting language, will be used to implement the back-end functionalities of the system. It is widely recognized for its cross-platform compatibility, high processing speed, and ability to integrate seamlessly with MySQL (Sotnik, Manakov, & Lyashenko, 2023). This integration facilitates efficient server-side development, enabling the smooth implementation of key features like user authentication, secure data handling, and personalized content delivery. Furthermore, PHP's ability to adapt and update code quickly without significant overhead makes it particularly advantageous for iterative development workflows, which align with the RAD methodology being used in this project (Sotnik, Manakov, & Lyashenko, 2023).

MySQL	The database management system, MySQL, is essential for handling the system's data requirements. Widely used alongside PHP, MySQL is known for its efficiency in storing, retrieving, and manipulating data, making it an ideal choice for dynamic, data-driven web applications (Sotnik, Manakov, & Lyashenko, 2023). Its strengths, like its reliable security system, its multi-platform open-source nature, and compatibility with modern development environtment and tools, including integration with Laravel, will ensure that MySQL meets the current needs of the project while offering the flexibility to scale and adapt as future requirements evolve (Sotnik, Manakov, & Lyashenko, 2023).

The other tools that will be utilized for the development of the proposed system are as outlined in the table below:

Table 3.2: Tools

Tools	**Description/Reasoning**
HTML	For structuring the web pages
CSS	For styling and designing the user interface
JavaScript	For adding interactivity and dynamic behaviours to the front-end.
Laragon	A development environment that I am familiar with that can run local servers, including PHP and MySQL.

3.2.2.5 **Hardware Requirements**

The hardware requirements for the back-end system are as stated in the table below:

Table 3.3: Hardware Requirements

Hardware	**Description/Reasoning**
Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz	It is a processor designed for efficiency and performance, made to be able to handle demanding tasks. Moreover, it will be adequate for handling software development tasks like running local servers (e.g., Laragon) and big database queries that can usually cause a lot of lag or unresponsiveness. All in all, the processor will help ensure a responsive back-end development.
16.0 GB RAM	16.0 GB of RAM is sufficient for ensuring that the system remains effective when working with resource-intensive tools like Laravel, PHP, and MySQL.
64-bit operating system, x64-based processor	A 64-bit operating system enables efficient memory utilization and enhances the processing speed of complex applications. It is also better than a 32-bit system.

3.2.2.6 **Design and Prototyping (UML/SSADM)**

3.2.2.6.1 Activity Diagram

The activity diagrams for the Admin and User roles are presented separately in Figures 3.1 and 3.7, respectively.

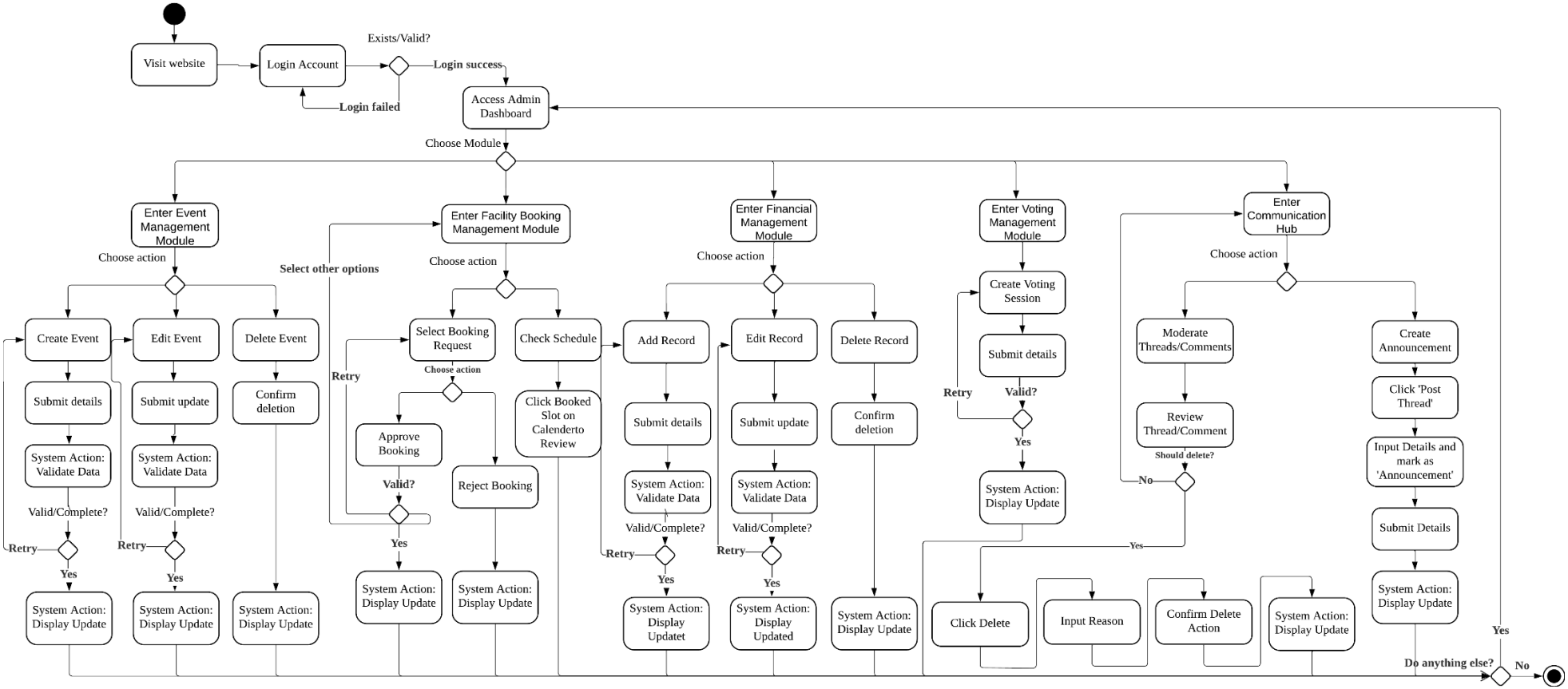


Figure 3.1: Smart Community Platform for Taman Suria Tropika Activity Diagram (Admin)

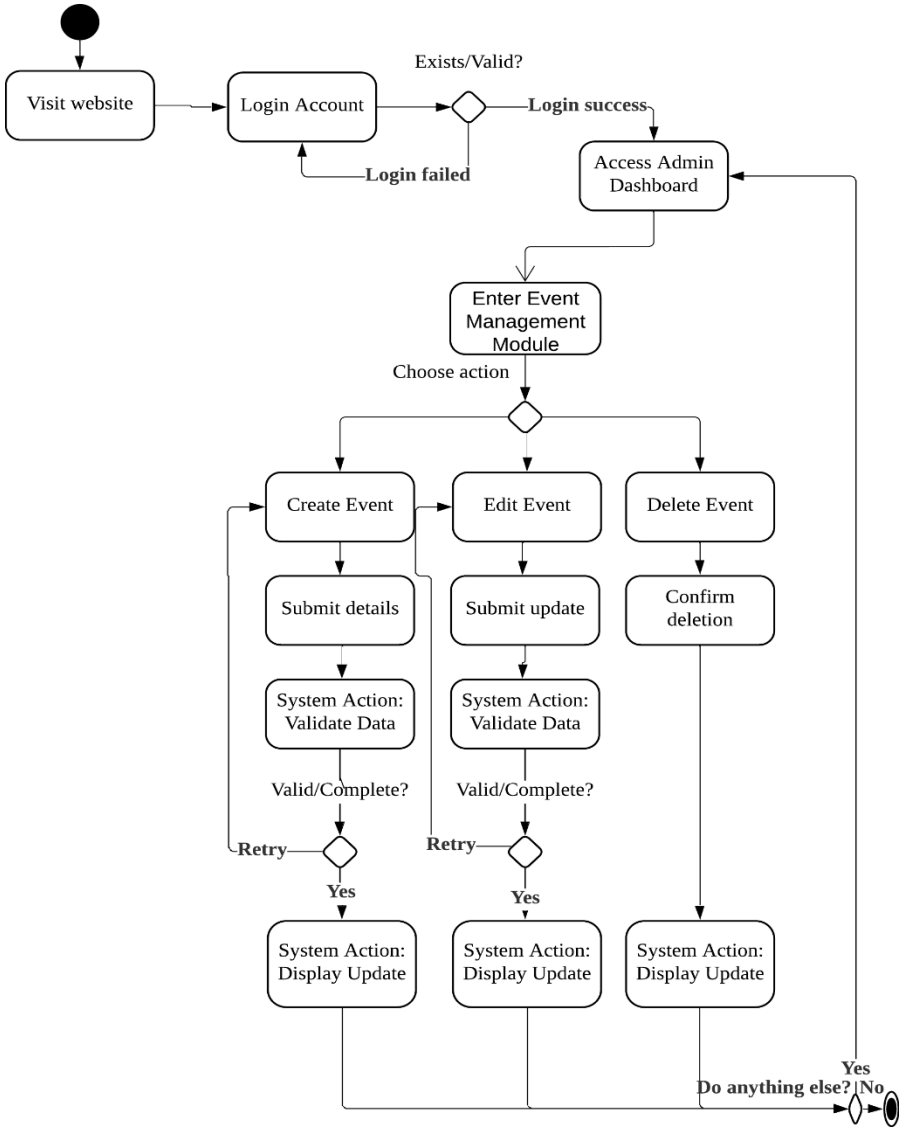


Figure 3.2: Activity Diagram Focused on the Event Management Module (Admin)

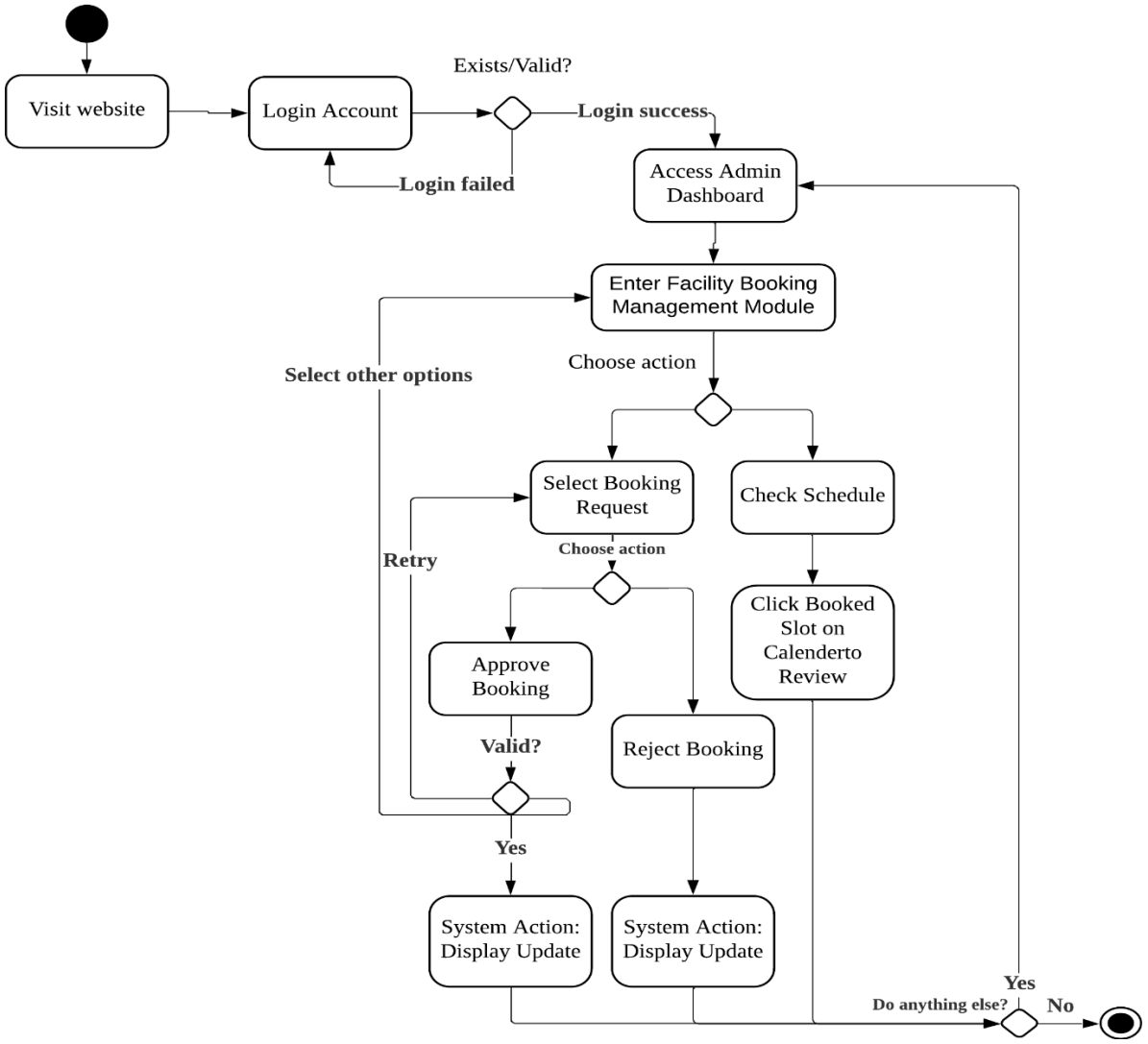


Figure 3.3: Activity Diagram Focused on the Facility Booking Management Module (Admin)

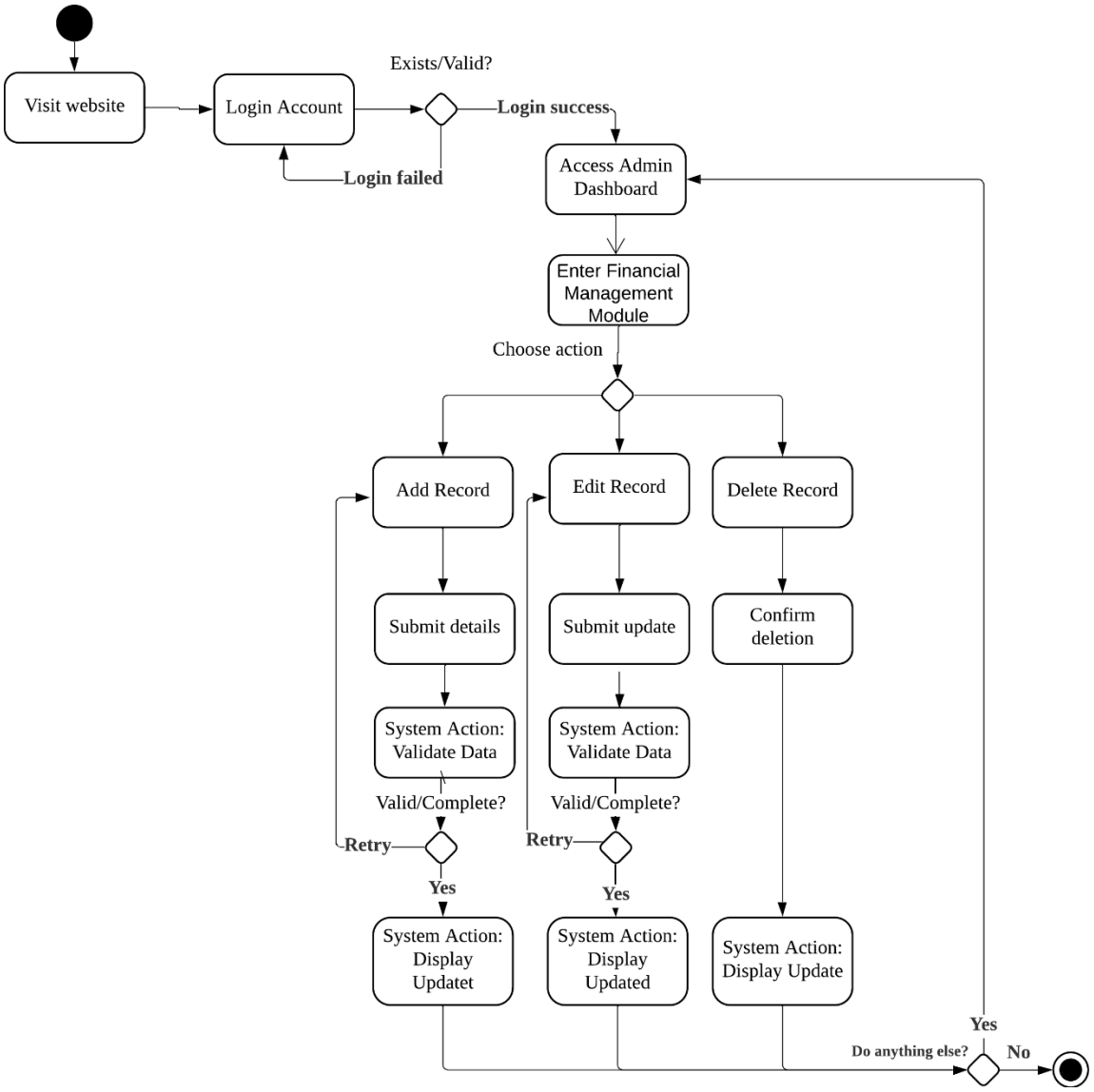


Figure 3.4: Activity Diagram Focused on the Financial Management Module (Admin)

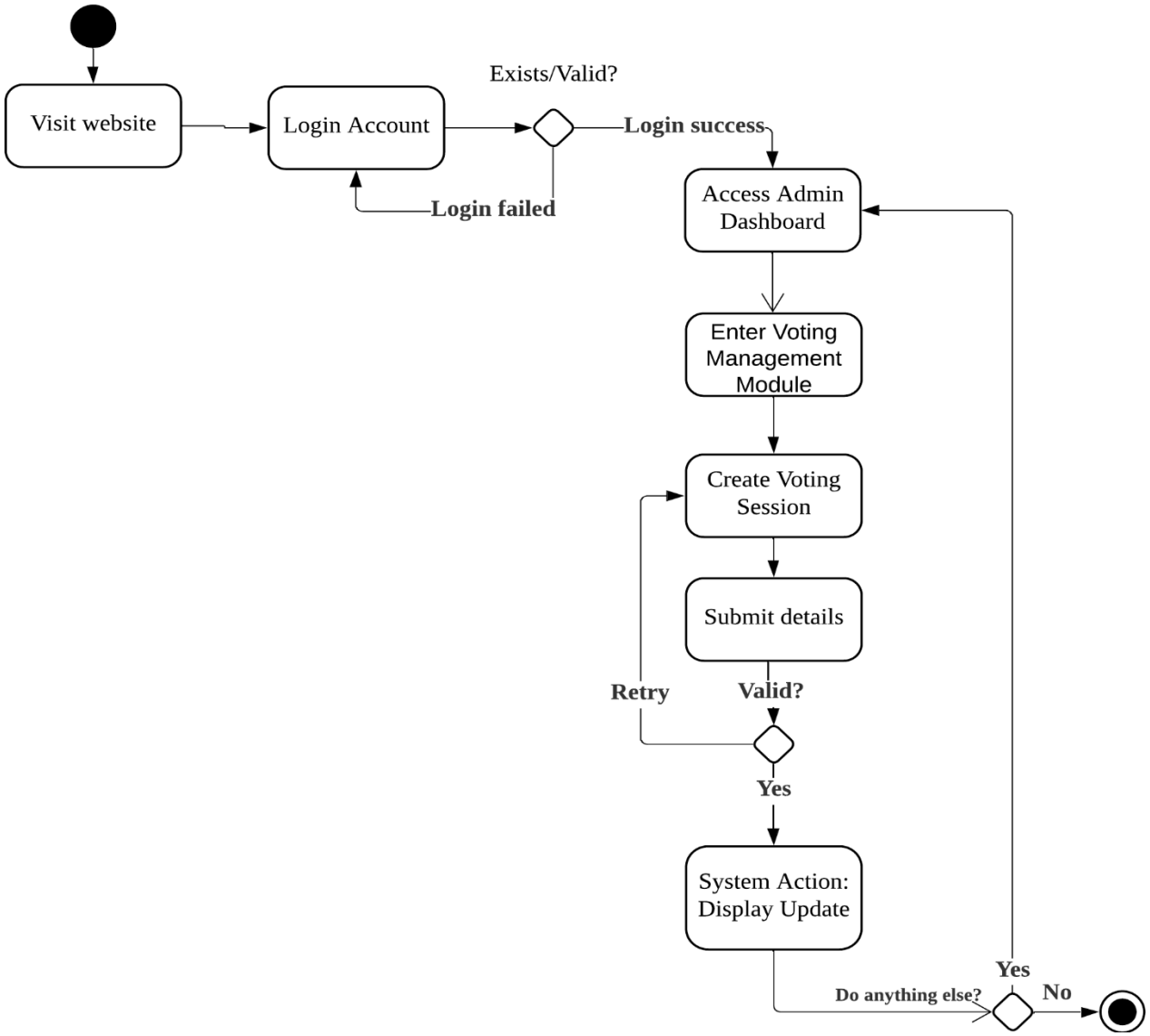


Figure 3.5: Activity Diagram Focused on the Voting Management Module (Admin)

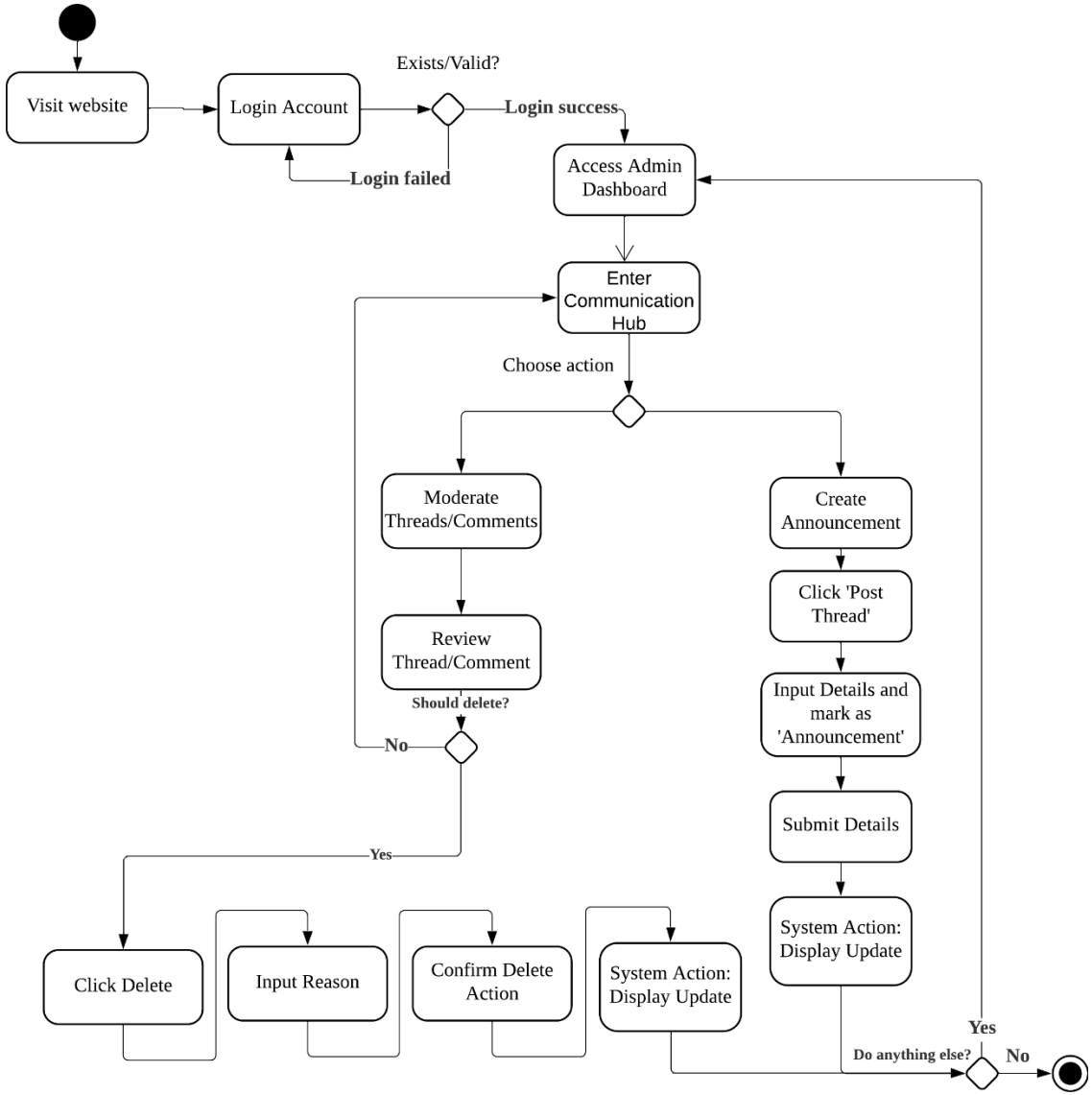


Figure 3.6: Activity Diagram Focused on the Communication Hub (Admin)

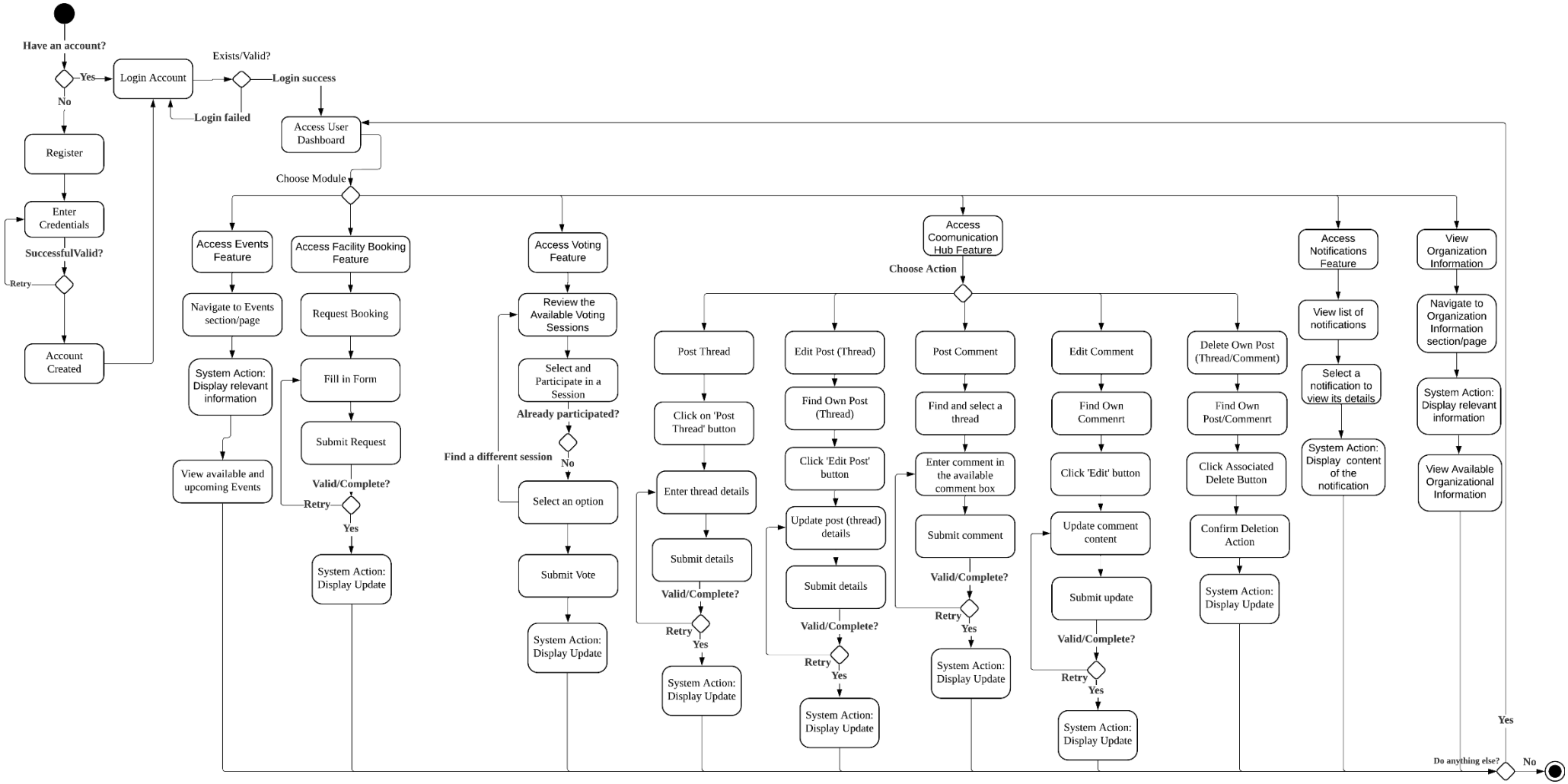


Figure 3.7: Smart Community Platform for Taman Suria Tropika Activity Diagram (User)

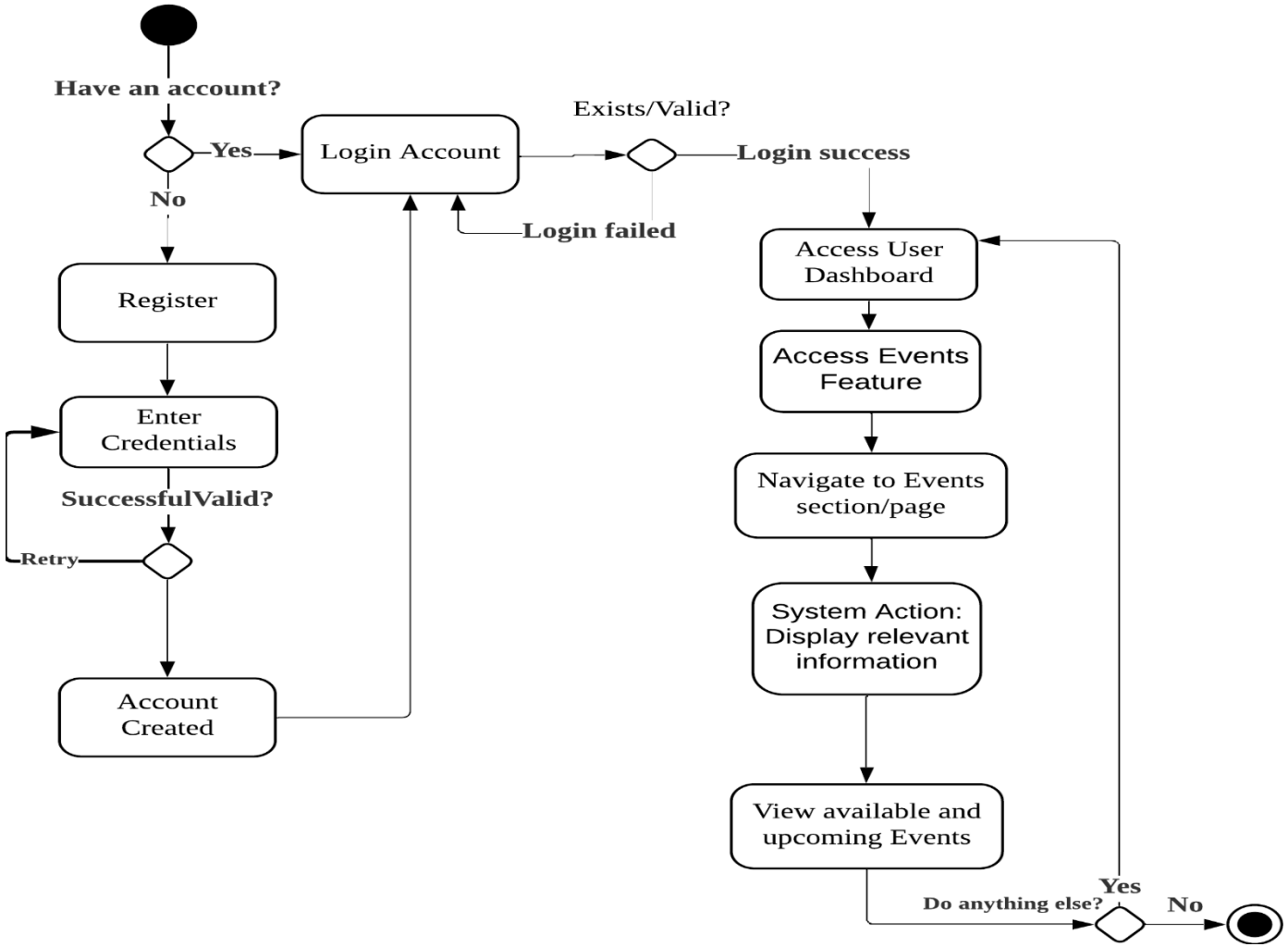


Figure 3.8: Activity Diagram Focused on the Events Feature (User)

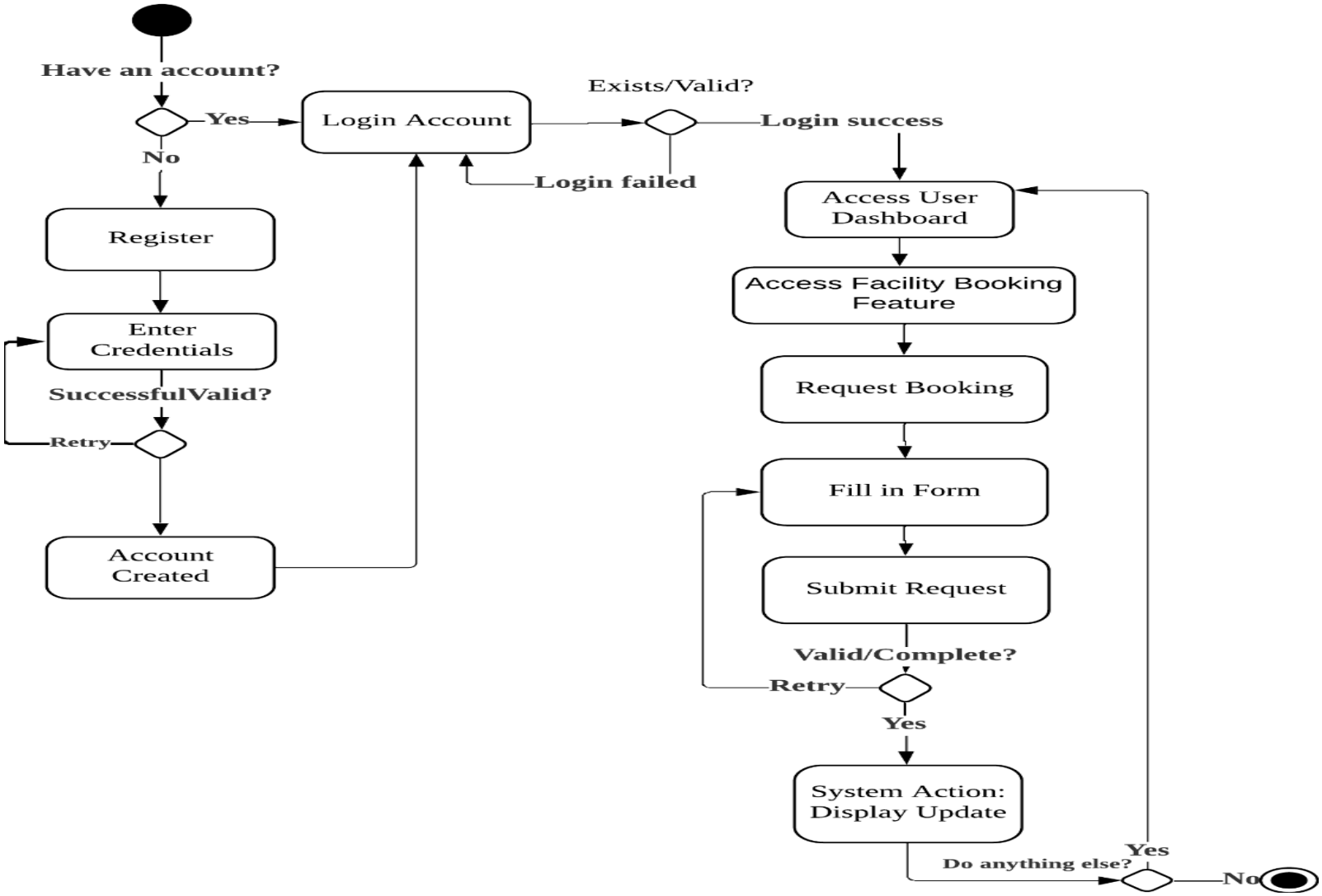


Figure 3.9: Activity Diagram Focused on the Facility Booking Feature (User)

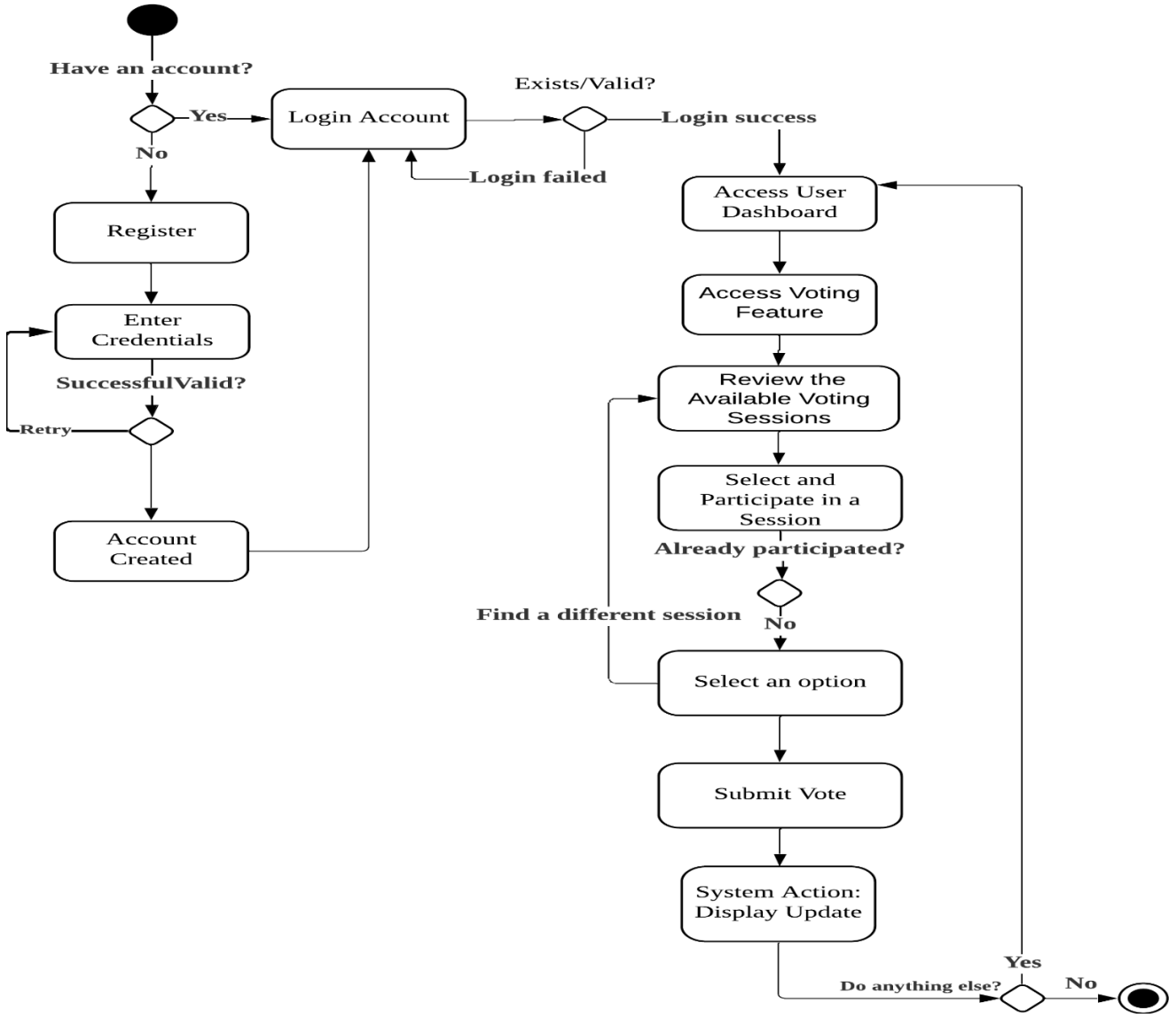


Figure 3.10: Activity Diagram Focused on the Voting Feature (User)

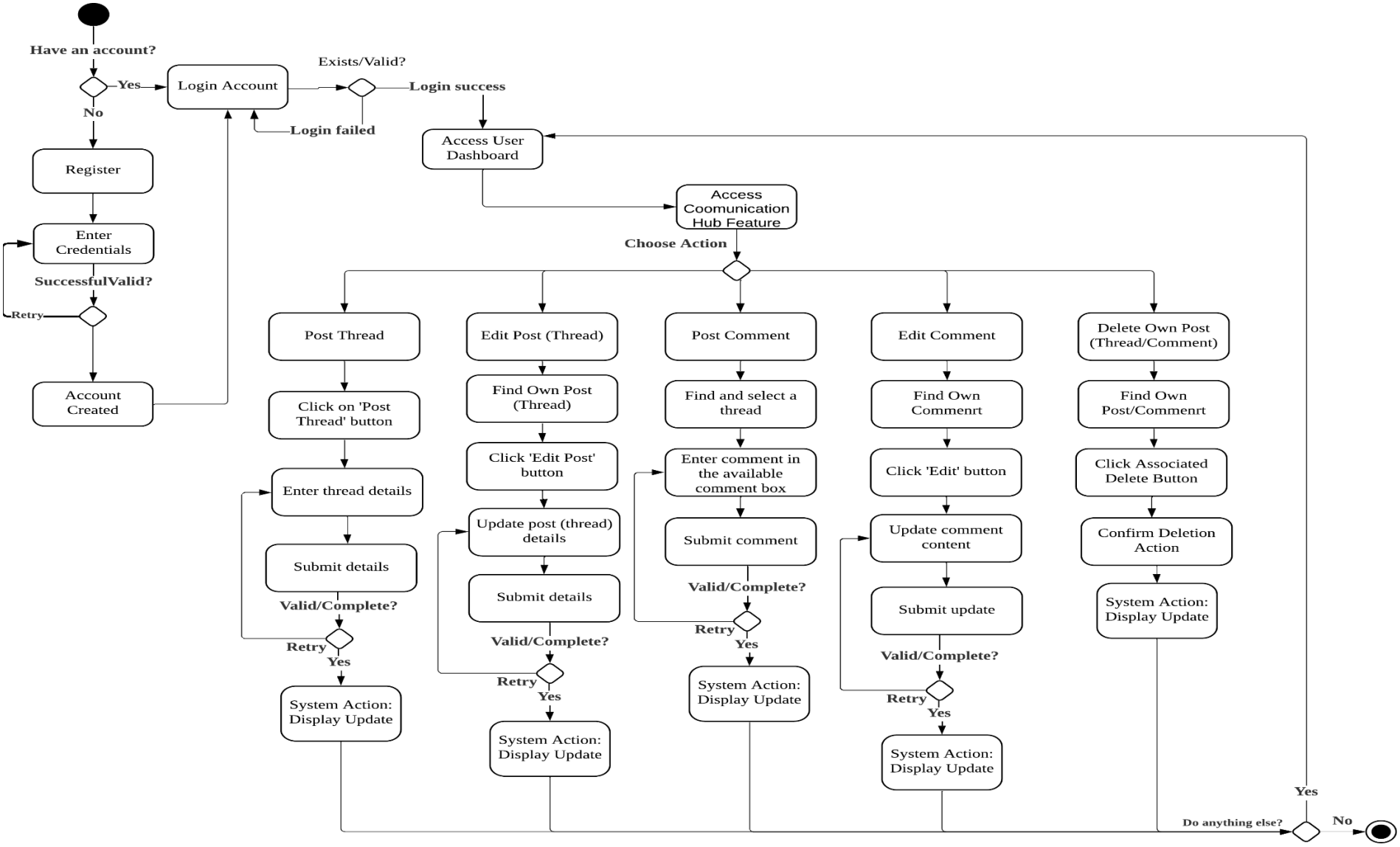


Figure 3.11: Activity Diagram Focused on the Communication Hub Feature (User)

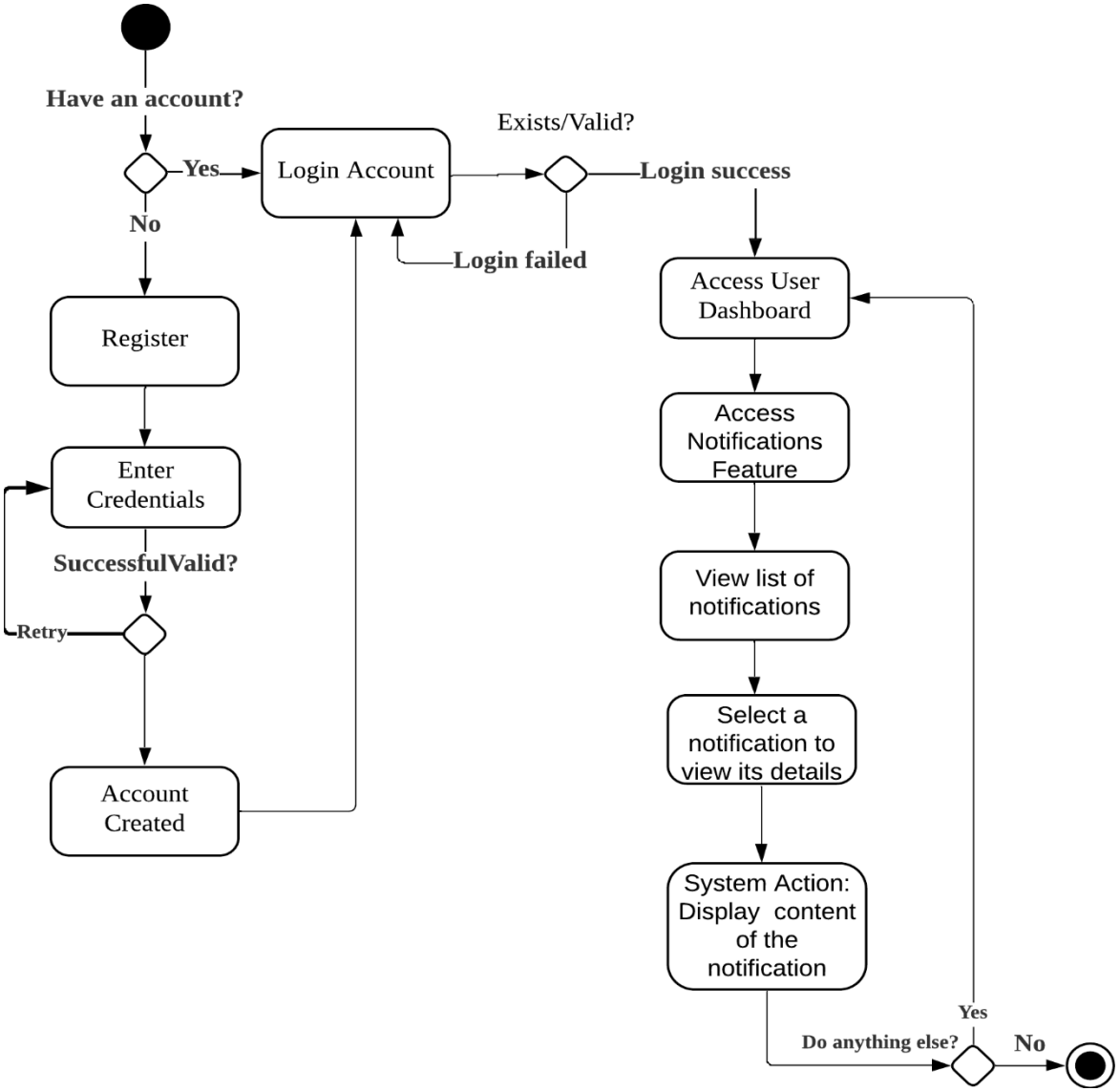


Figure 3.12: Activity Diagram Focused on the Notifications Feature (User)

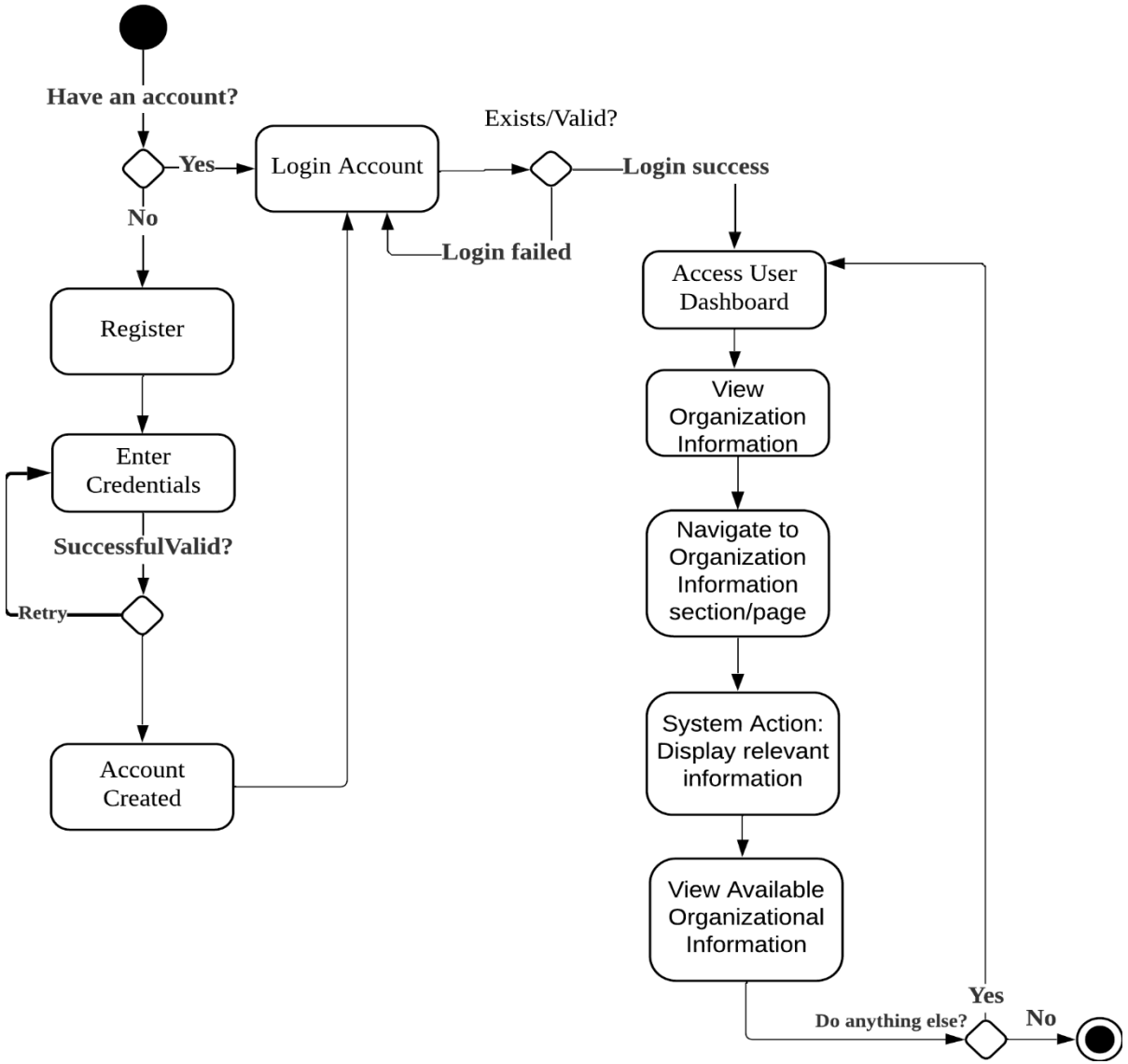


Figure 3.13: Activity Diagram Focused on the View Organization Information (User)

3.2.2.6.2 Use Case Diagram of the Proposed System

Figure 3.14: The Use Case Diagram of the Proposed System

3.2.2.6.3 Sequence Diagram of Login

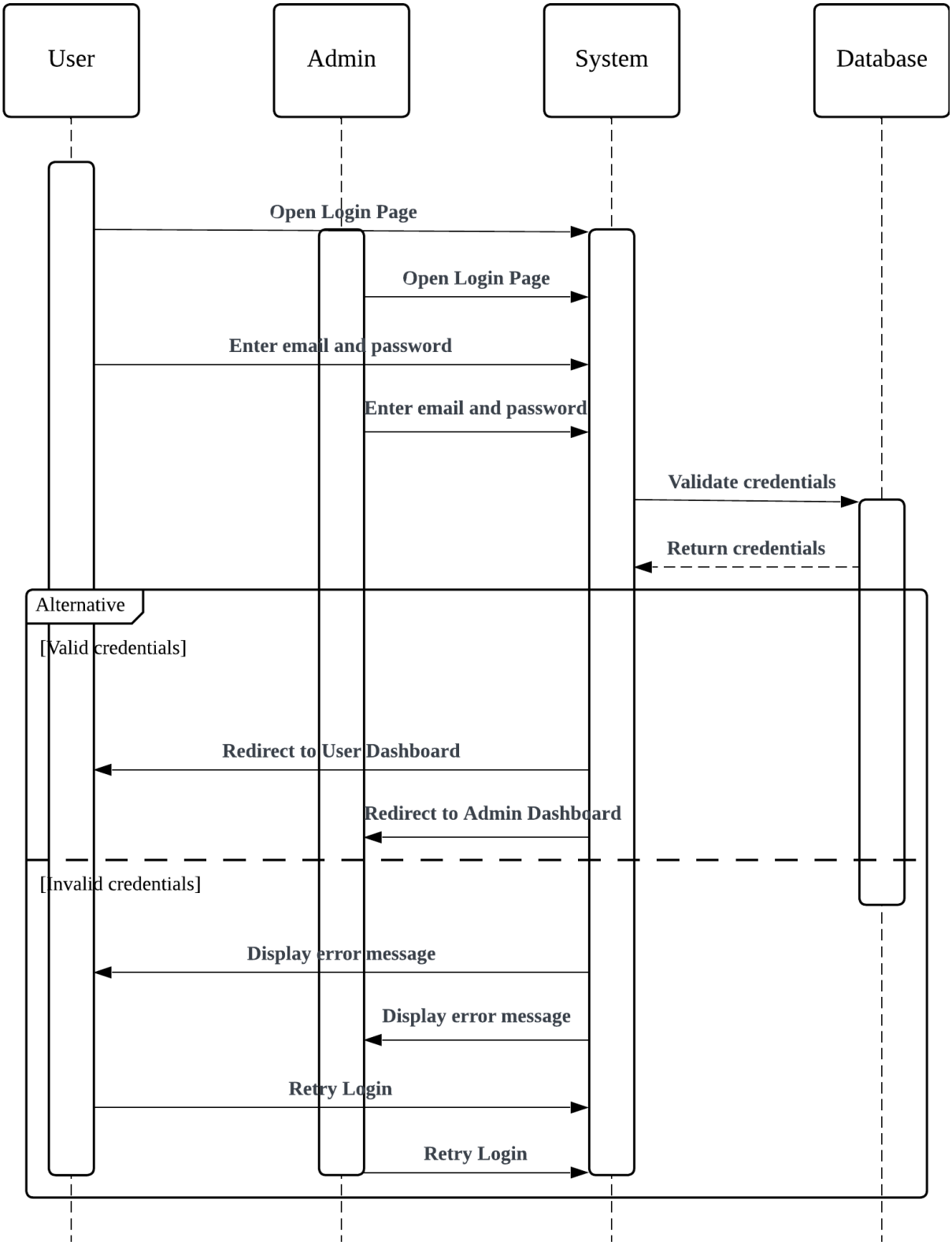


Figure 3.15: Sequence Diagram of Login

Table 3.4: Use Case Description for Login

Use Case	Login
Short Description	User and Admin wants to log into the system.
Actors	User, Admin
Preconditions	Actors must already have a pre-existing account with valid credentials.
Postconditions	The actor, be it normal User or Admin, gets to log into the system.
Main Flow	1. The actor initiates by navigating to the Login Page. 2. The actor will then proceed by entering their email and password. 3. The system will then attempt to validate the credentials by querying the database. 4. Should the credentials be valid, the actor will be granted access and redirected to their perspective dashboards. The User will be redirected to the User Dashboard, while the Admin will be redirected to the Admin Dashboard. Thus, it is a successful login for the actor.
Exception Flow	4a. If the credentials are invalid, the system will display an error message, and the actor will be prompted to try logging in once more or recover their password should they forget.

3.2.2.6.4 Register

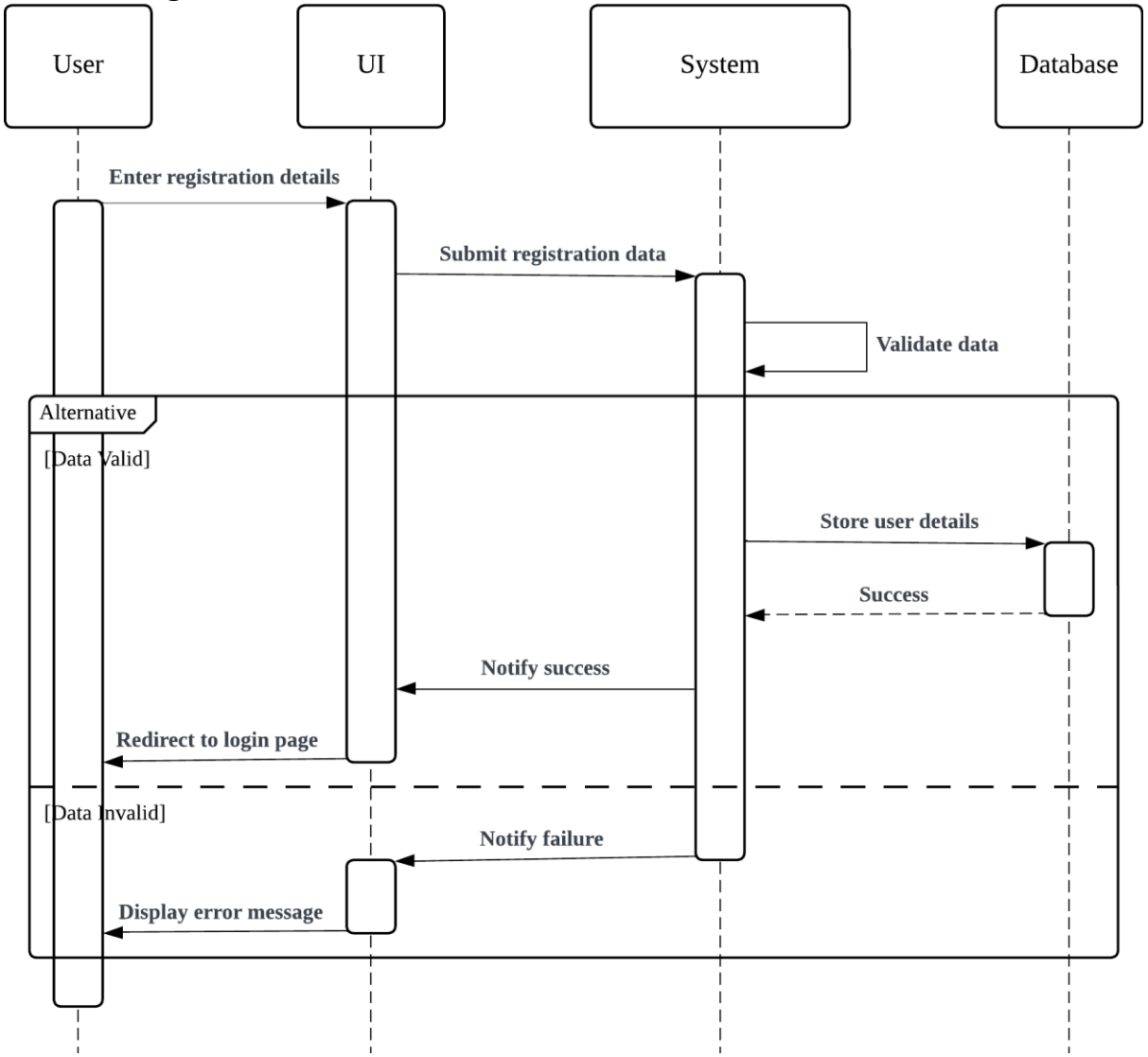


Figure 3.16: Sequence Diagram for Register

Table 3.5: Use Case Description for Register

Use Case	Register
Short Description	User wants to register a new account into the system.
Actors	User
Preconditions	• User must access the registration page through the user interface. • User need to possess an email.
Postconditions	User possess a registered account and can now log in to the system.

Main Flow	1. The actor initiates by navigating to the Registration Page. 2. The actor fills in the form which asks for their name, email, and password. 3. The system validates the provided data. 4. If the provided data is valid, then the system will confirm the creation of a new account and store it in the database, followed by redirecting the actor to the Login Page.
Exception Flow	4a. If the providede information is invalid, then the system will display an error message and prompt the user to correct the errors before submitting the form once more.

3.2.2.6.5 Manage Events

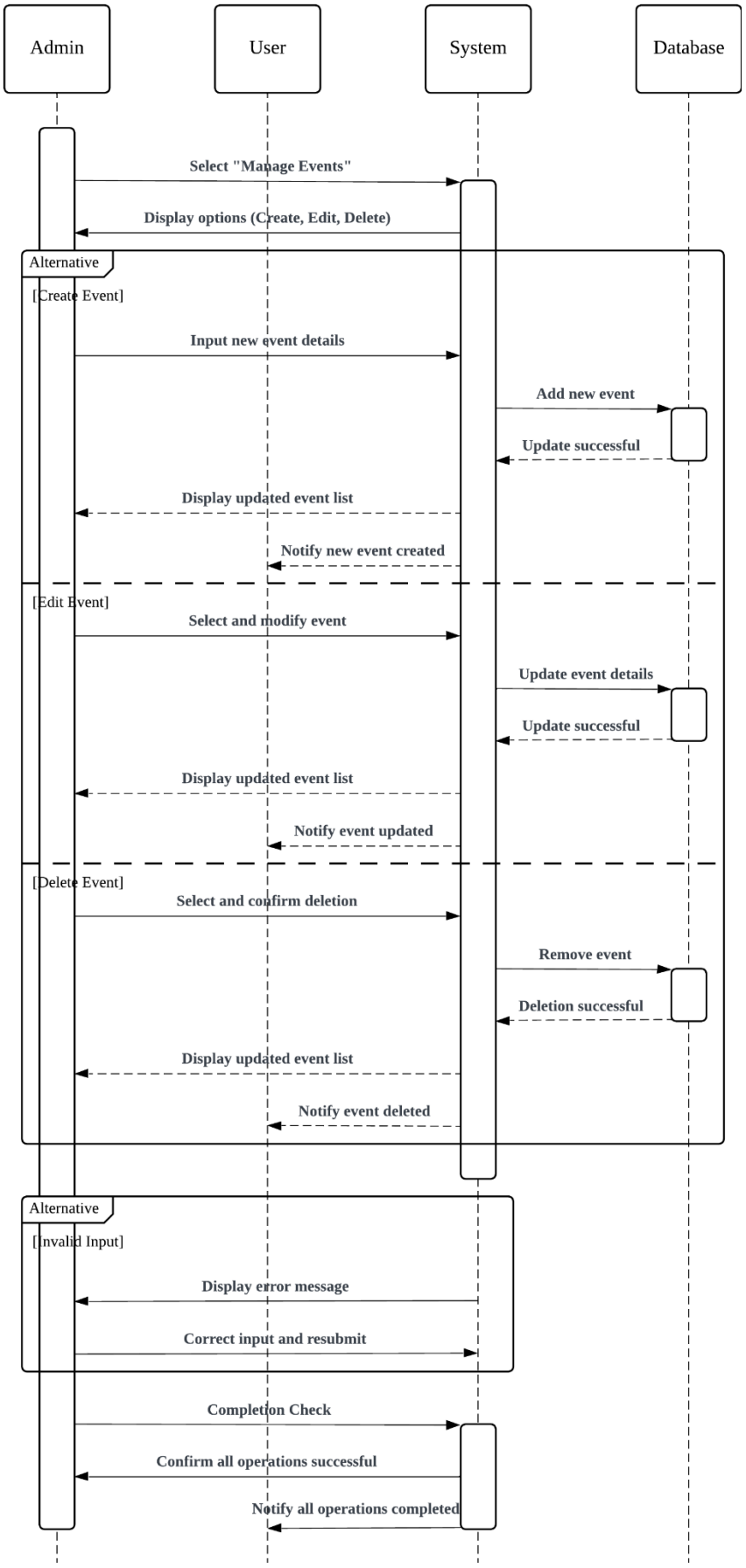


Figure 3.17: Sequence Diagram for Manage Events

Table 3.6: Use Case Description for Manage Events

Use Case	Manage Events (Parent Use Case)
Short Description	Admin uses the system to manage events, mainly utilising functionalities like create, edit, and delete events.
Actors	Admin
Preconditions	The admin must already be logged into the system and the system itself must already have a functional database to store and retrieve event details.
Postconditions	• The database gets updated based on the entered event data. • The updated event list is displayed to the admin. • The updated even gets displayed on the Events page. • Users are notified of the Event.

Main Flow	1. The admin selects the Event Management option available on their dashboard.
	2. The system presents the admin with three options:
	  • Create Event: The admin inputs specific details like picture, event name, the date, time and some description to add a new event.
	  • Edit Event: The admin selects an existing event, modifies its details, and saves the changes.
	  • Delete Event: The admin selects an event and confirms its deletion from the system.
	3. The system processes the admin's request and updates the database accordingly.
	4. The system confirms the success of the operation and displays the updated list of events.
	5. The event also gets displayed on the Events page of the system.

Exception Flow	4a. If the admin provides incomplete or invalid data, then the system will display an error message and prompt them to correct the errors before submitting the form once more.

3.2.2.6.6 Manage Facility Booking

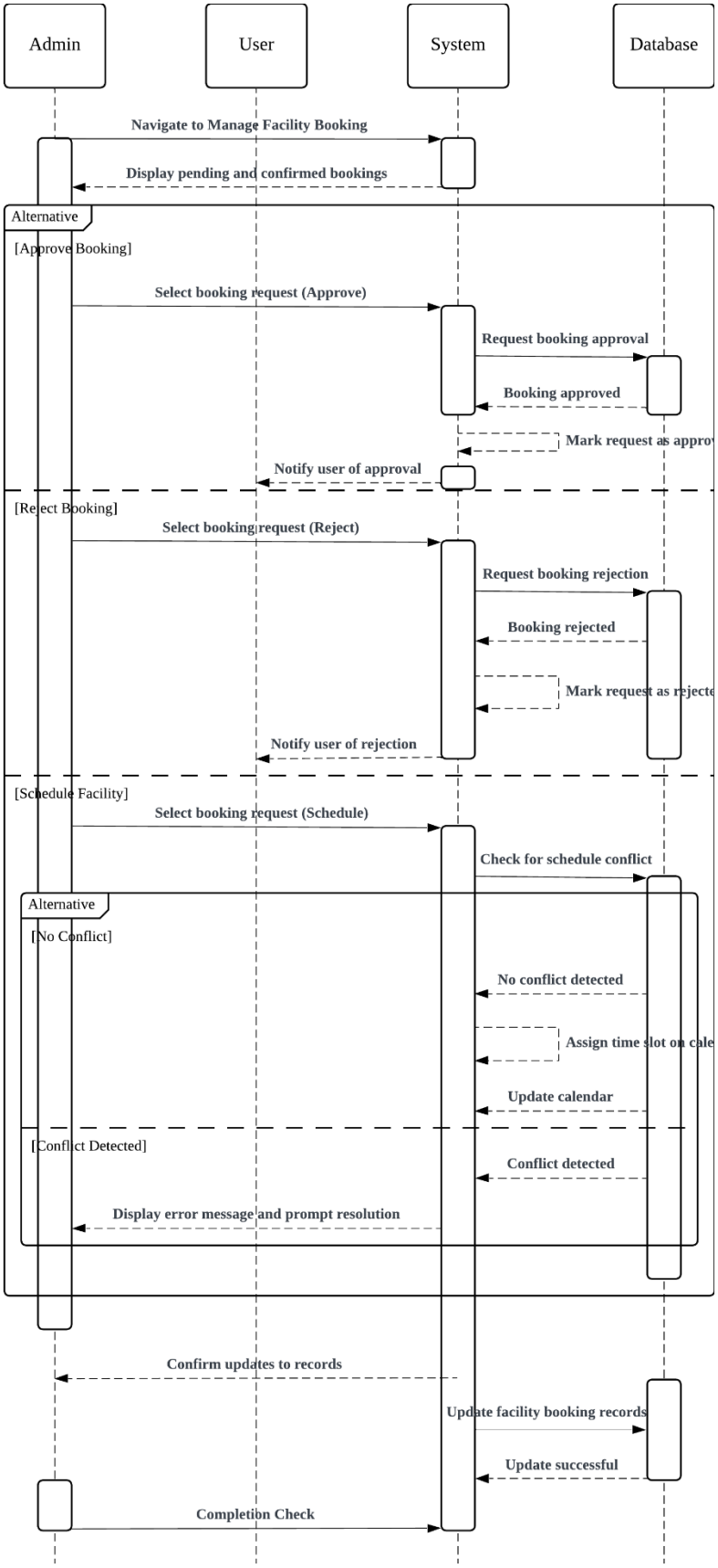


Figure 3.18: Sequence Diagram for Manage Facility Booking

Table 3.7: Use Case Description for Manage Facility Booking

Use Case	Manage Facility Booking (Parent Use Case)
Short Description	Admin uses the system to manage facility bookings submitted by users. The Admin can either approve or reject booking requests and schedule the facility for confirmed bookings.
Actors	Admin, User
Preconditions	• The admin must already be logged into the system. • There must already be booking requests made by users in the system. • The system must maintain a calendar and database of facility schedules.
Postconditions	• The facility booking records are updated with the approved, rejected, or scheduled status. • Users are notified of the outcome of their booking requests.

Main Flow	1. The actor navigates to the Manage Facility Booking section/page.
	2. The system displays a list of pending and confirmed booking requests.
	3. The actor will proceed to select a booking request then proceed with a type of action:
	  • Approve Booking: The booking request gets marked as approved.
	  • Reject Booking: The booking request gets marked as rejected and the user who made the booking request gets notified.
	4. The system validates and updates the facility booking records and reflects the changes in the schedule.

Exception Flow	4a. If the Admin attempts to approve a booking that conflicts with an existing reservation, then the system will display an error message and prompt the Admin to resolve the conflicting issue.

3.2.2.6.7 Check Schedule

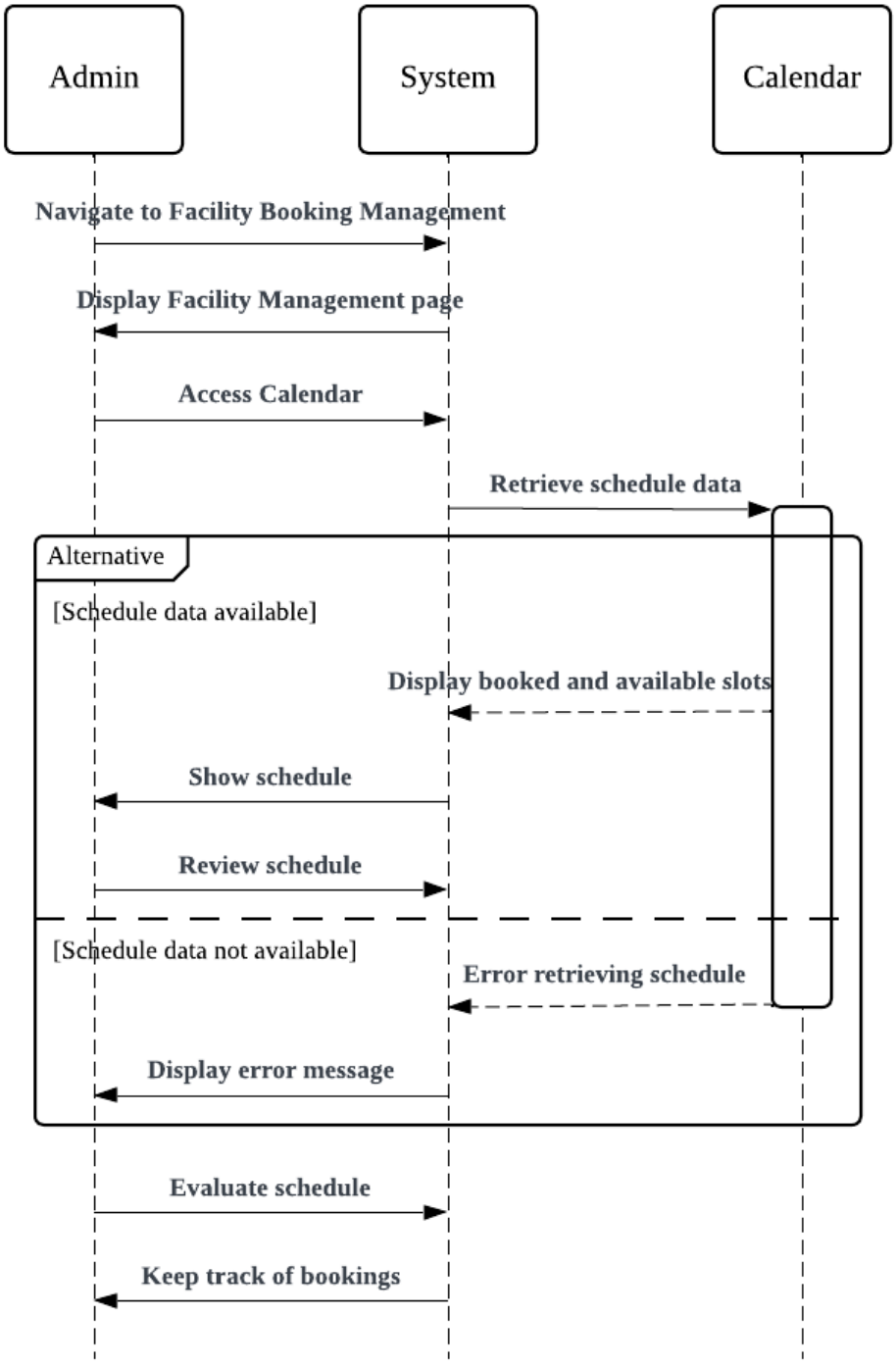


Figure 3.19: Sequence Diagram for Check Schedule

Table 3.8: Use Case Description for Check Schedule

Use Case	Check Schedule

Short Description	Admin and users can view the facility's schedule to check for available time slots and existing bookings.
Actors	Admin
Preconditions	• The actor must already be logged into the system. • The system must maintain a calendar and database of facility schedules.
Postconditions	• The actor is able to view and keep track of scheduled bookings.

Main Flow	1. The actor navigates to the Facility Booking Management section/page.
	2. The actor navigates to the calender in the system.
	3. The calender in the system displays:
	  • Booked time slots, and clicking on one will display its details.
	  • Available time slots.
	4. The actor reviews the schedule to determine availability or identify conflicts.

Exception Flow	3a. If the schedule data cannot be retrieved, the system will display an error message and prompt the actor to try again later.

3.2.2.6.8 Manage Finances

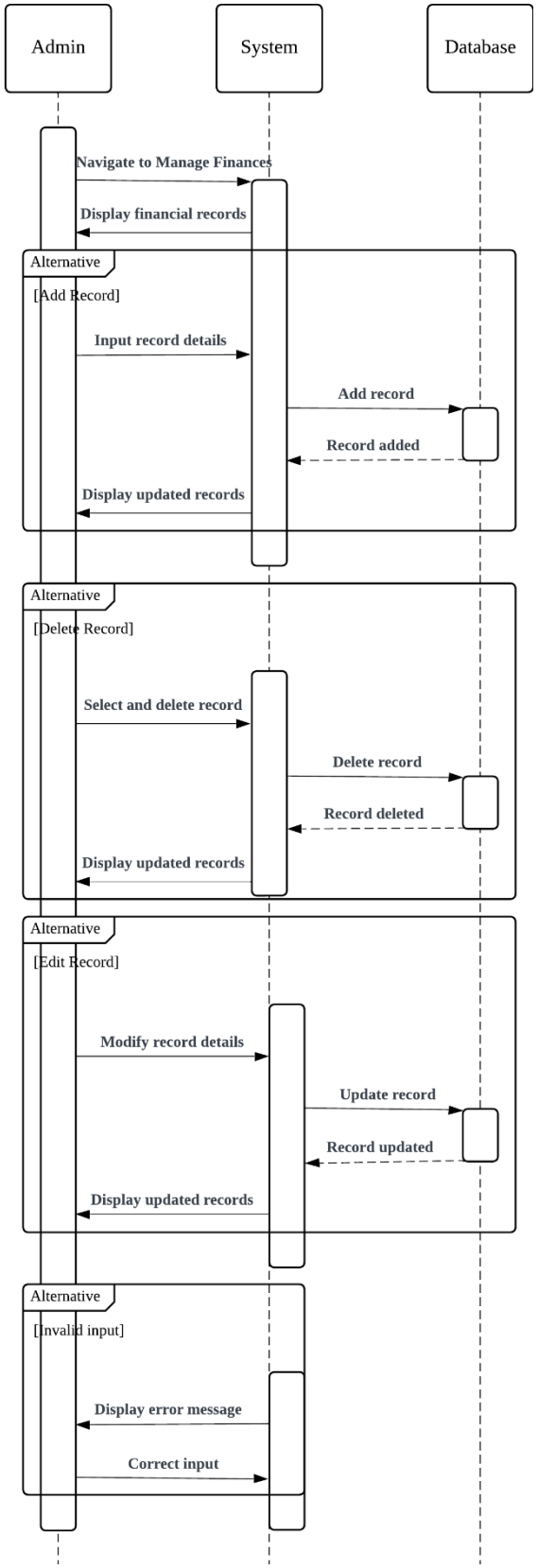


Figure 3.20: Sequence Diagram for Manage Finances

Table 3.9: Use Case Description for Manage Finances

Use Case	Manage Finances (Parent Use Case)
Short Description	Admin uses the system to manage the financial records of the surau, which includes adding, deleting, and editing income or expense records.
Actors	Admin
Preconditions	• The actor must already be logged into the system. • The system must already have a functional database to store financial records.
Postconditions	• The update to the financial records gets displayed on the system. • The database for the financial records gets updated.

Main Flow	1. The actor navigates to the Finance Management section/page.
	2. The system displays the available display or list of financial records.
	3. The actor will proceed to select an action:
	  • Add Record: The actor will first choose whether it will be 'expense' or 'income', then input necessary details such as the category, description, amount, and date.
	  • Delete Record: The actor selects a record and deletes it.
	  • Edit Record: The actor selects a record and modifies its details.
	4. The system validates and updates the financial records, and the changes can be seen in the page.

Exception Flow	4a. If the actor inputs invalid or incomplete data, then an error message will appear, and the actor will be prompted to resolve the issue by correcting the input or completing the details.

3.2.2.6.9 View Events

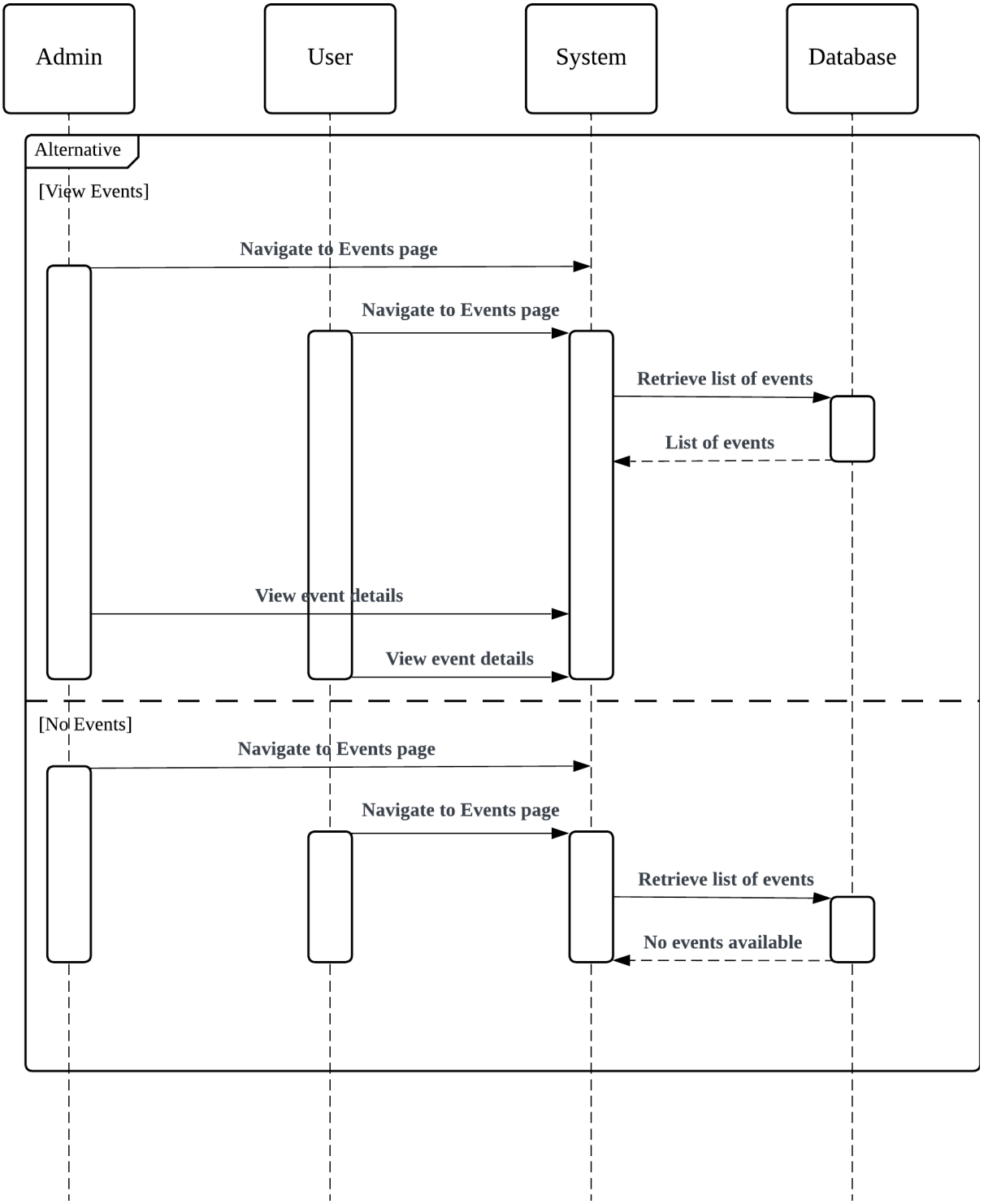


Figure 3.21: Sequence Diagram for View Events

Table 3.10: Use Case Description for View Events

Use Case	View Events

Short Description	Admin and User uses the system to view the details of the events that are being managed in the system
Actors	Admin, User
Preconditions	• The system must at least have one event available in the database.
Postconditions	• The actor manages to view the available event details as intended.
Main Flow	1. The actor navigates to the Events section/page. 2. The system will retrieve a list of past and upcoming events from the database. 3. The actor will proceed to look through the available events on display, with details including: • Name of Event • Date • Time • Location • Description
Exception Flow	3a. There are no available events, so there will be nothing for the actor to view.

3.2.2.6.10 Request Booking

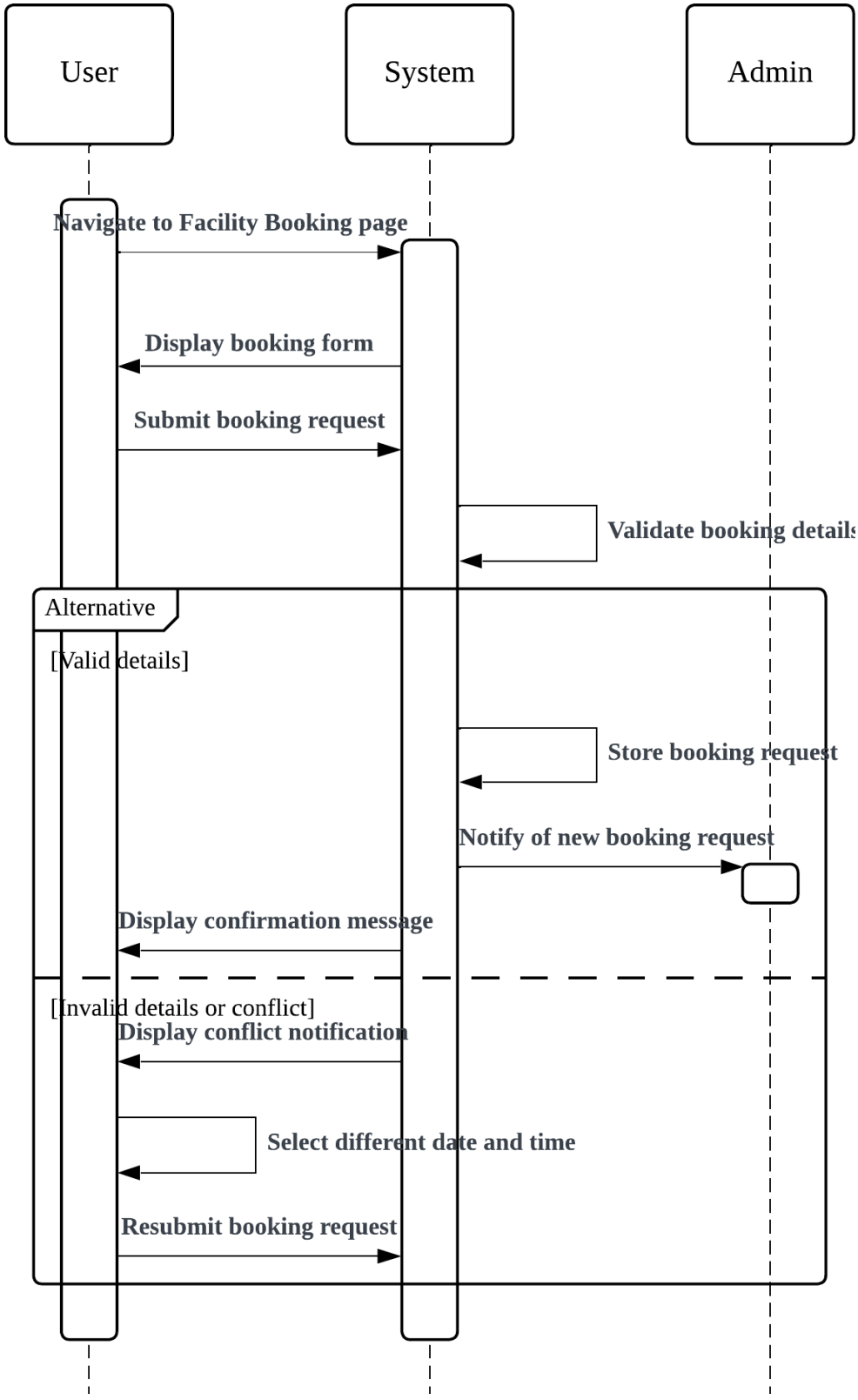


Figure 3.22: Sequence Diagram for Request Booking

Table 3.11: Use Case Description for Request Booking

Use Case	Request Booking
Short Description	User uses the system to submit a booking request for the use of a facility, such as the surau or playground area.
Actors	User
Preconditions	• The actor must already be logged into the system. • The system maintains a list of available facilities available for booking.
Postconditions	• The booking request is successfully submitted and stored in the system. • The Admin gets notified of the new booking request.
Main Flow	1. The actor navigates to the Facility Booking page/section. 2. The actor clicks on the 'Create' button, leading to the system to display a form that the actor has to fill in, complete with the name of the facility, date and time of booking, and purpose or description of the booking. 3. The actor submits the booking request. 4. The system validates the booking details. 5. If the details of the system are valid, the system will store the request and notify the Admin for review. 6. The system will display a confirmation message.
Exception Flow	4a. If the submitted request form is incomplete or the booking's date and time conflicts with an existing booking, then the system will display a conflict notification, and the actor will be prompted to select a different date and time before resubmitting the request.

3.2.2.6.11 Setup Voting

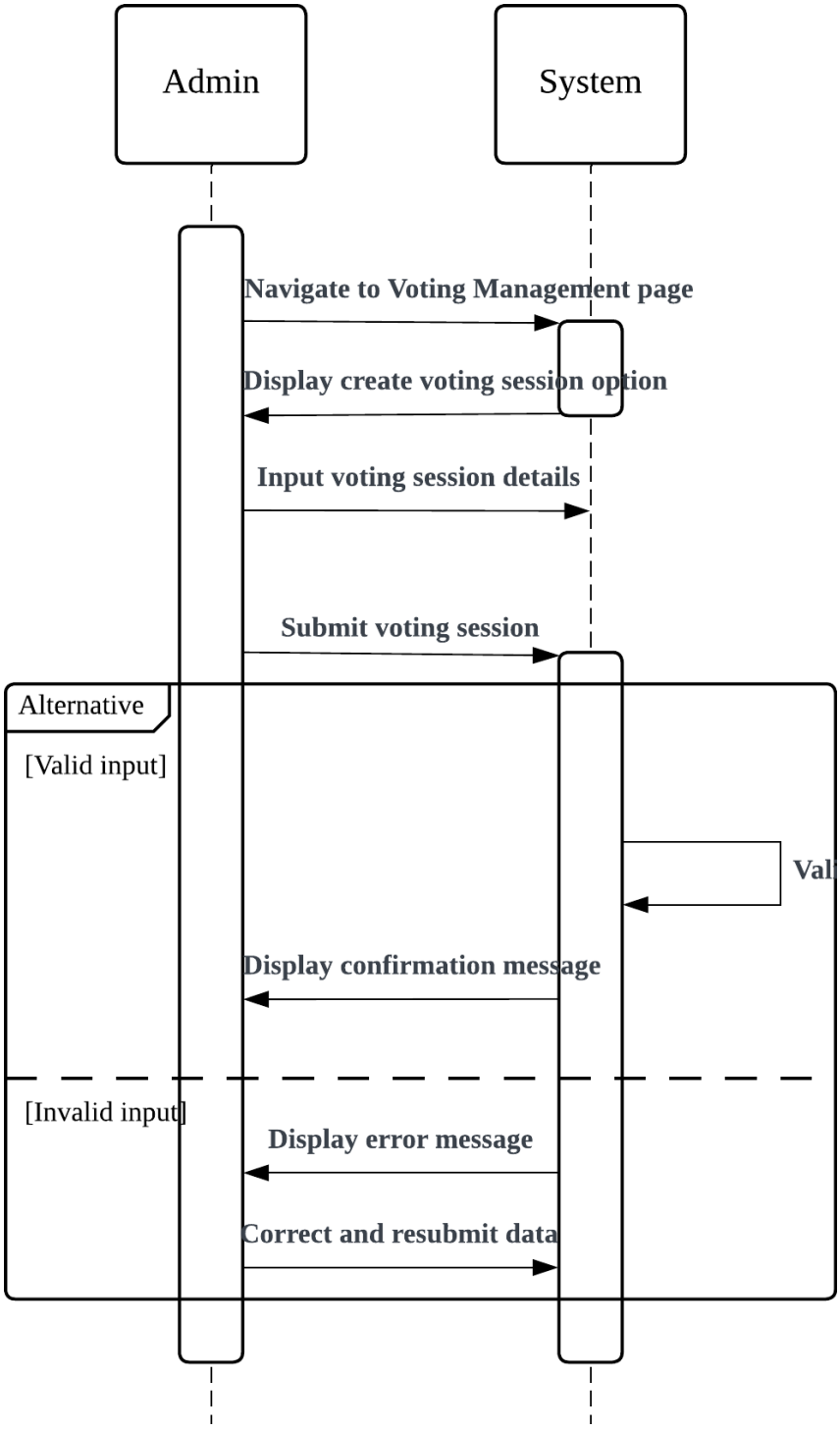


Figure 3.23: Sequence Diagram for Setup Voting

Table 3.12: Use Case Description for Setup Voting

Use Case	Setup Voting
Short Description	Admin uses the system to create and configure voting sessions.
Actors	Admin

Preconditions	• The actor must already be logged into the system.
Postconditions	• The voting session becomes available
Main Flow	1. The actor navigates to the Voting Management page/section. 2. The system displays an option to create a new voting session. 3. The actor inputs details on the voting session, which includes: • Title of the vote. • Description of the vote. • Start and end time for the voting session. • Voting options. 4. The Admin submits the voting session configuration. 5. The system validates the input data. 6. If the input is valid, the system saves the voting session details and the system will display a confirmation message. 7. The voting session is made available to other system users.
Exception Flow	5a. If the data submitted is invalid or incomplete, then the system will display an error message and prompt the actor to correct the input before resubmitting the voting session configuration.

3.2.2.6.12 Participate in Voting

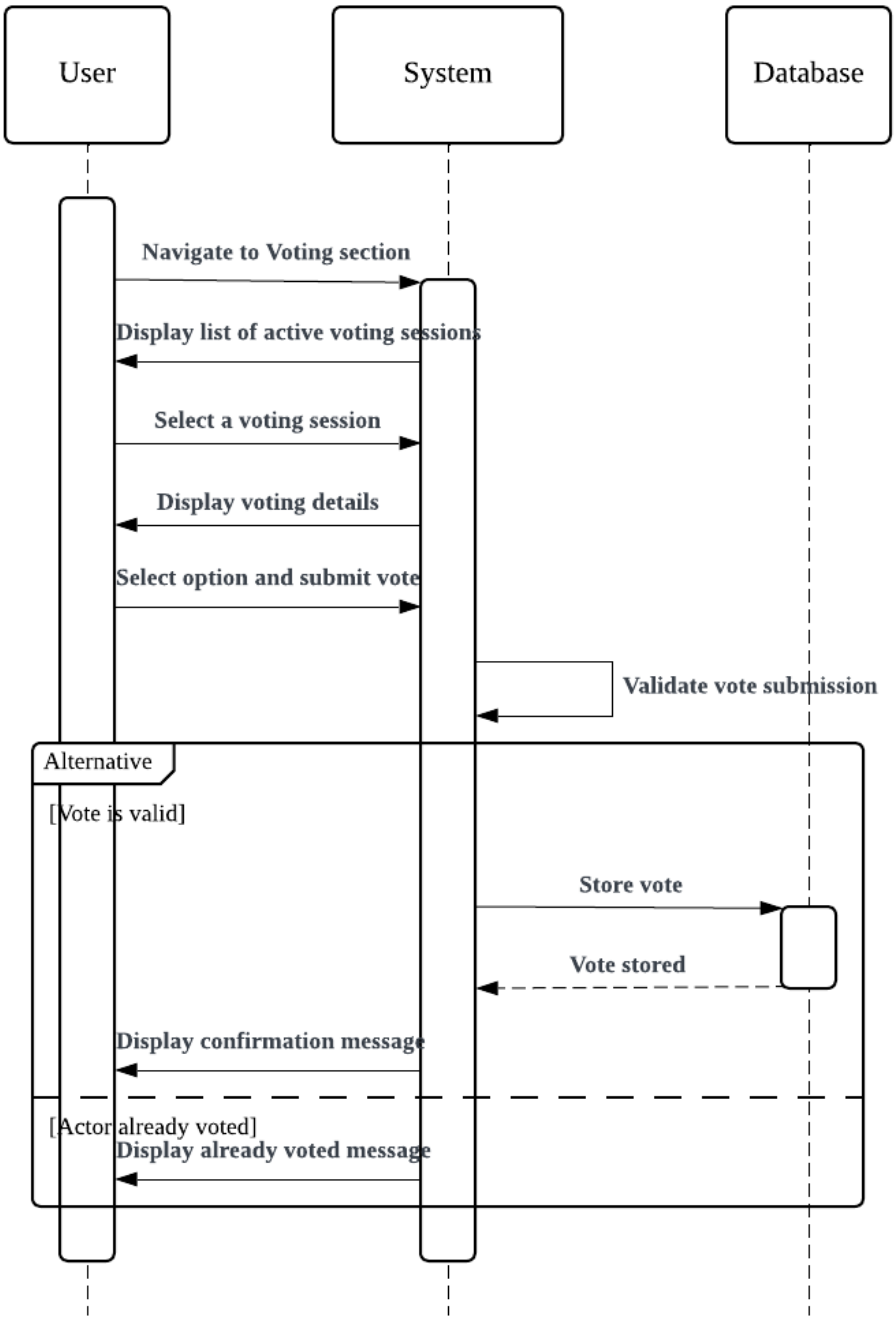


Figure 3.24: Sequence Diagram for Participate in Voting

Table 3.13: Use Case Description for Participate in Voting

Use Case	Participate in Voting
Short Description	User uses the system to view and cast their votes in active voting sessions.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system.

	• At least one voting session must be available.
Postconditions	• The actor's vote is recorded in the database and counted in the results for the voting session. • The actor cannot cast another vote for the same session.
Main Flow	1. The actor navigates to the Voting section/page. 2. The system displays a list of available and active voting sessions. 3. The actor selects a voting session to participate in. 4. The system retrieves and displays the voting details, including the title, description, duration, and voting options. 5. The actor selects an option and submits their vote. 6. The system validates the vote submission. 7. If valid, the system stores the vote in the database and displays a confirmation message.
Exception Flow	6a. If the actor has already participated in the selected voting session, then the system will display a message stating that the actor already voted for the session.

3.2.2.6.13 View Threads

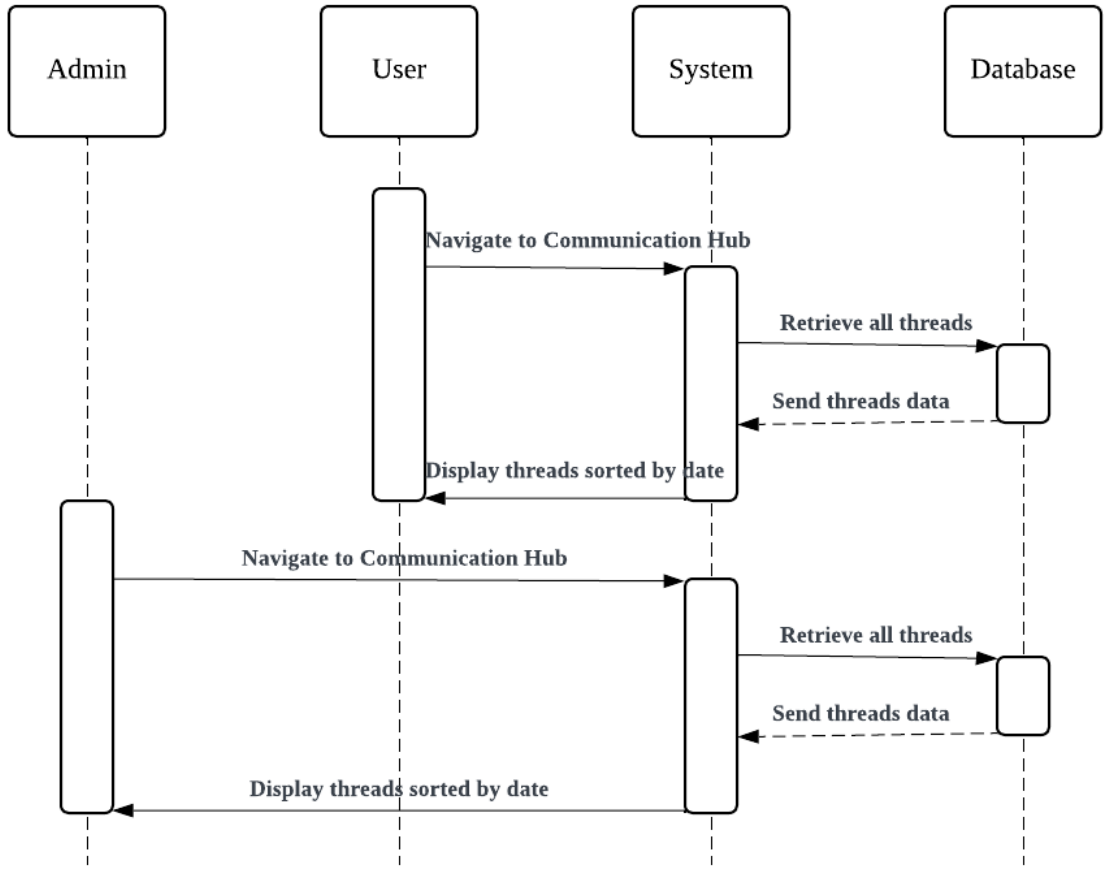


Figure 3.25: Sequence Diagram for View Threads

Table 3.14: Use Case Description for View Threads

Use Case	View Threads
Short Description	The actors use the system to view all the available threads in the Communication Hub.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system.
Postconditions	• The actor can view all the available threads.
Main Flow	1. The actor navigates to the Communication Hub section/page. 2. The system retrieves all threads from the database. 3. The system displays all the available threads sorted by date.

3.2.2.6.14 View Thread

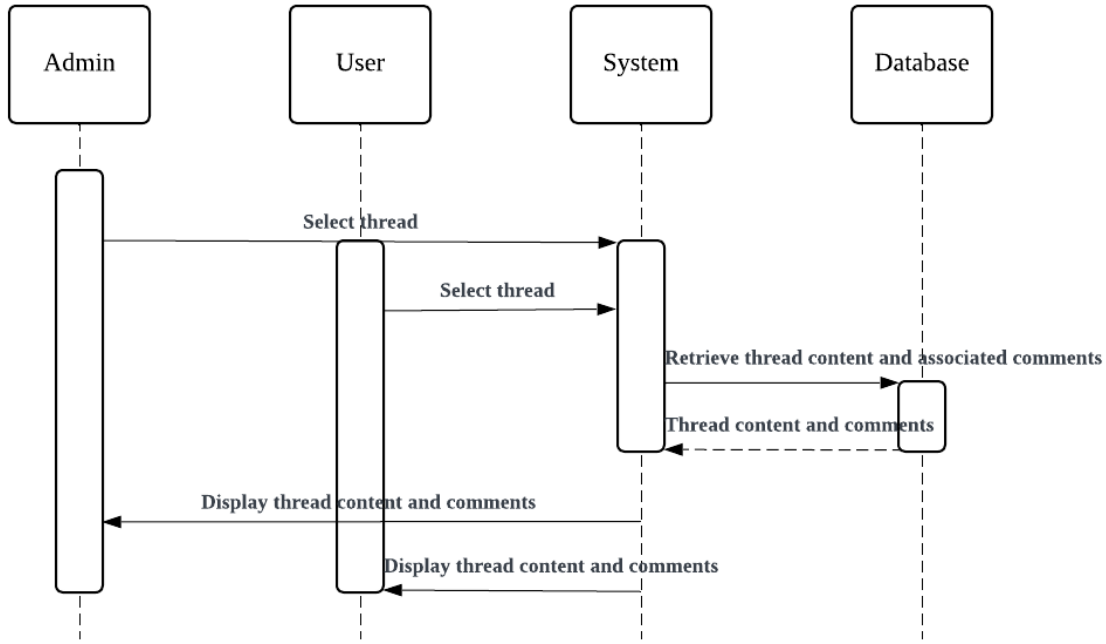


Figure 3.26: Sequence Diagram for View Thread

Table 3.15: Use Case Description for View Thread

Use Case	View Thread
Short Description	The actors use the system to view the content and comments of a specific thread.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system. • The actor has already navigated to the Communication Hub section/page where they can see the list of available threads.
Postconditions	• The actor can read the thread and its existing comments.
Main Flow	1. The actor selects a thread from the list of available threads. 2. The system retrieves the selected thread's content alongside all the comments associated with it from the database. 3. The system displays the thread's content and comments.

3.2.2.6.15 Moderate Threads and Comments

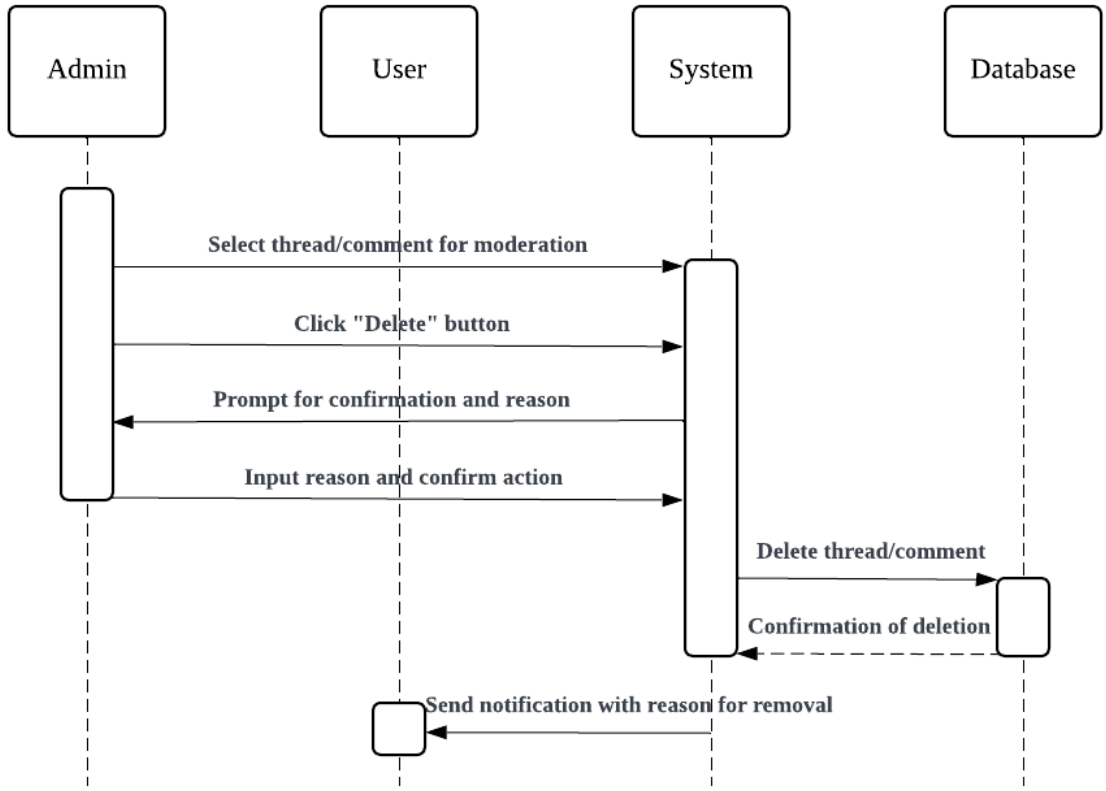


Figure 3.27: Sequence Diagram for Moderate Threads and Comments

Table 3.16: Use Case Description for Moderate Threads and Comments

Use Case	Moderate Threads and Comments
Short Description	The actors use the system to remove threads or comments that have been deemed inappropriate.
Actors	Admin
Preconditions	• The actor must already be viewing a selected thread or comment for moderation.
Postconditions	• The thread or comment is removed from the Communication Hub. • The user who had created the thread mark or comment will receive a message through the notification, detailing why their thread or comment was removed.
Main Flow	1. The actor selects a thread or comment for moderation. 2. The actor clicks the “Delete” button. 3. The system prompts for confirmation and reason. 4. The actor inputs the reason and confirms the action.

	5. The system deletes the thread or comment and updates the database.

3.2.2.6.16 Post Thread

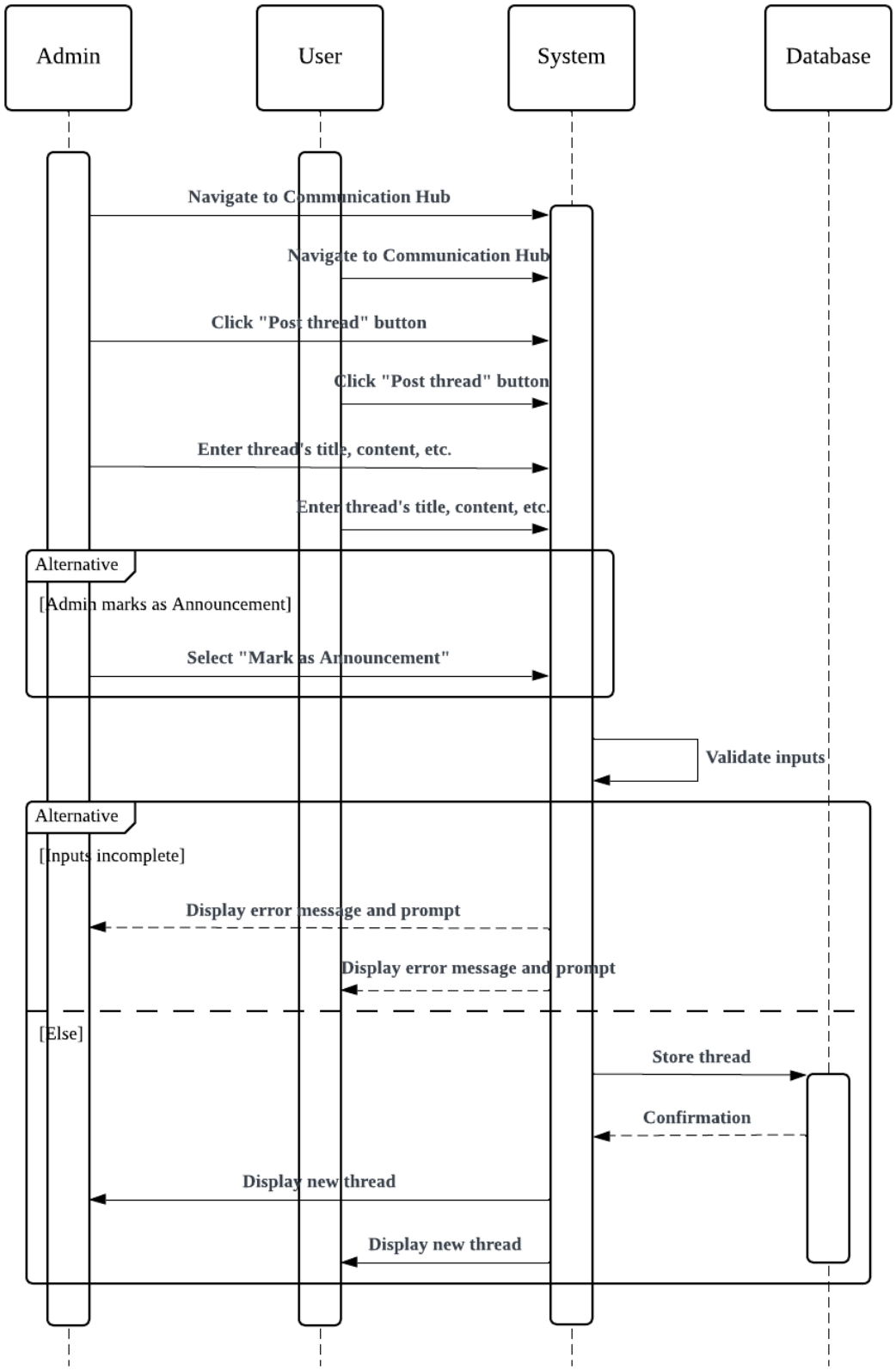


Figure 3.28: Sequence Diagram for Post Thread

Table 3.17: Use Case Description for Post Thread

Use Case	Post Thread
Short Description	The actors use the system to create a new discussion thread. Admins have the additional option to specify that the thread is an Announcement.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system.
Postconditions	• A new thread is created. • If the thread is marked as an Announcement, it is displayed with special formatting or in a designated section for Announcements.
Main Flow	1. The actor navigates to the Communication Hub section/page of the system. 2. The actor navigates to the page where they can create a thread by clicking on the “Post thread” button. 3. The actor enters the thread’s title, content, and any other relevant information before submitting. 4. The system validates the inputs and stores the thread in the database. 5. The system displays the new thread in the list of threads, and it becomes available for all to see.
Exception Flow	3a. Besides the normal thread mark details, an Admin has the option to mark the thread as an Announcement by selecting a checkbox or toggle labelled "Mark as Announcement." 4a. If the actor’s submission is incomplete, then the system will display an error message and prompt the user to correct the issue before resubmitting. 5a. The system displays the new thread which has been marked as an Announcement in the section designated for announcements only.

3.2.2.6.17 Post Comment

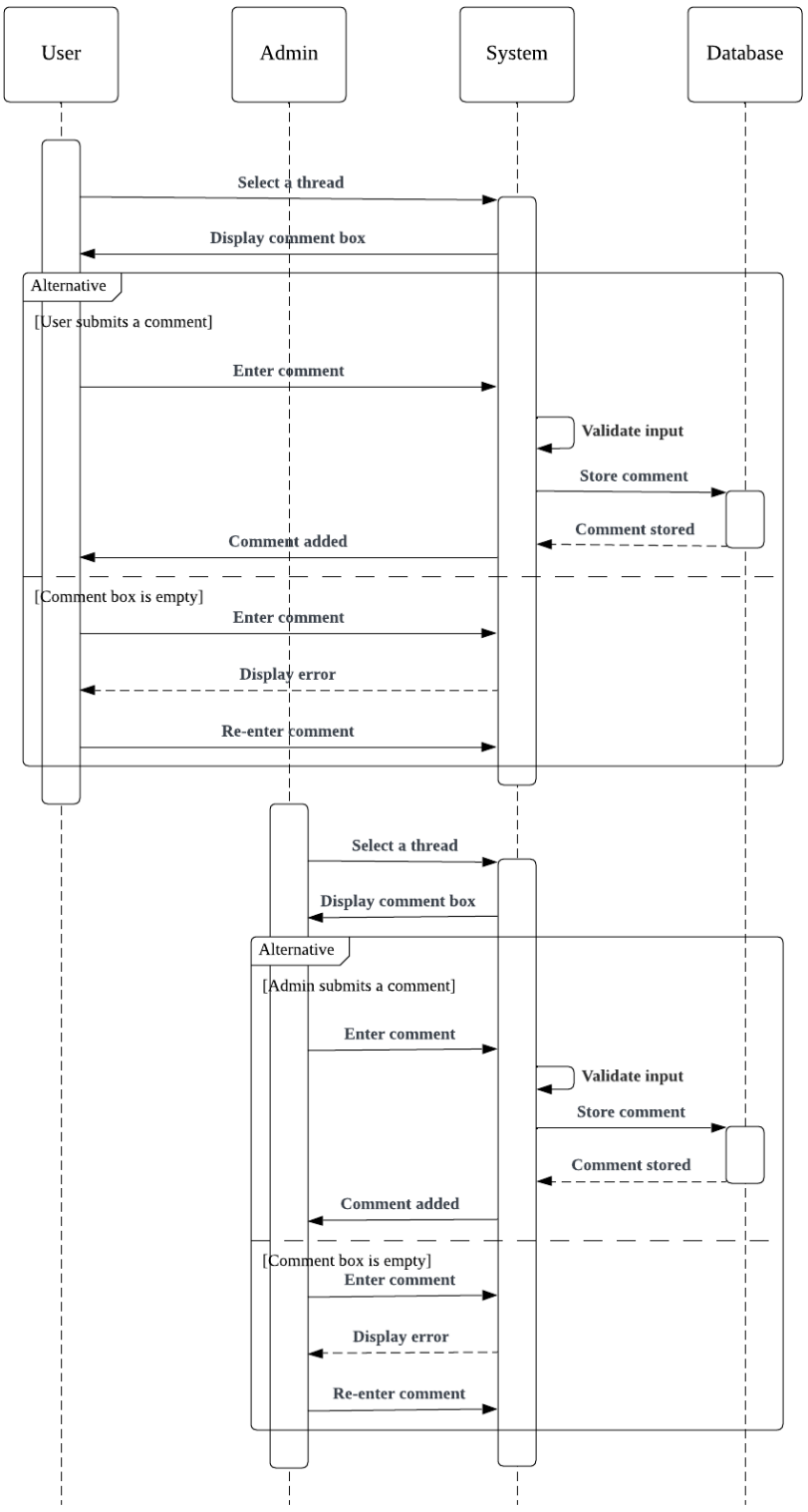


Figure 3.29: Sequence Diagram for Post Comment

Table 3.18: Use Case Description for Post Comment

Use Case	Post Comment
Short Description	The actors use the system to comment on a thread.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system. • The actor has already navigated to the Communication Hub section/page where they can see the list of available threads
Postconditions	• A new comment is added to the thread.
Main Flow	1. The user selects a thread. 2. The actor enters a comment in the available comment box. 3. The actor submits the comment. 4. The system validates the comment and stores it in the database.
Exception Flow	4a. The system detects that the comment box is empty during validation, causing the system to display an error message and prompt the user to re-enter their comment before resubmitting so that the system may validate the comment and store it in the database.

3.2.2.6.18 Edit Post

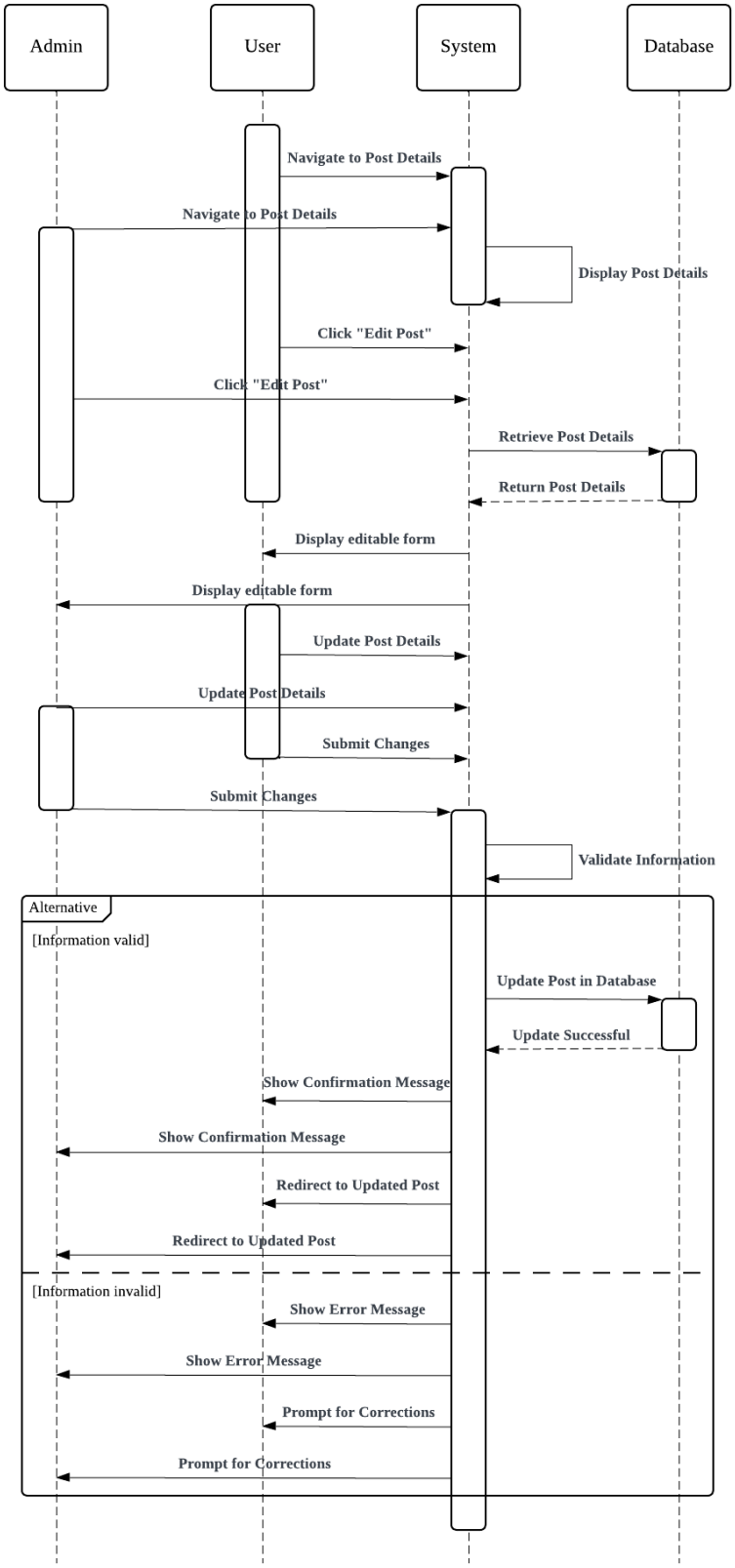


Figure 3.30: Sequence Diagram for Edit Post

Table 3.19: Use Case Description for Edit Post

Use Case	Edit Post
Short Description	The actors edit the post that they have created in the Communication Hub.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system. • The actor can only edit posts that they have created.
Postconditions	• The post is updated in the database and displayed with the new content in the system.
Main Flow	1. The actor navigates to the Post Details page or the page of their post. 2. The actor clicks on the "Edit Post" button. 3. The system retrieves the post details and displays them in editable form. 4. The actor updates the post's details. 5. The actor submits the changes made. 6. The system validates the updated information. 7. If valid, the system updates the post in the database. 8. The system displays a confirmation of success message and redirects the actor to the updated post.
Exception Flow	6a. If the updated information is invalid (e.g., empty title or excessive length), then the system will display and error message and prompt the actor to correct the issue before resubmitting.

3.2.2.6.19 Edit Comment

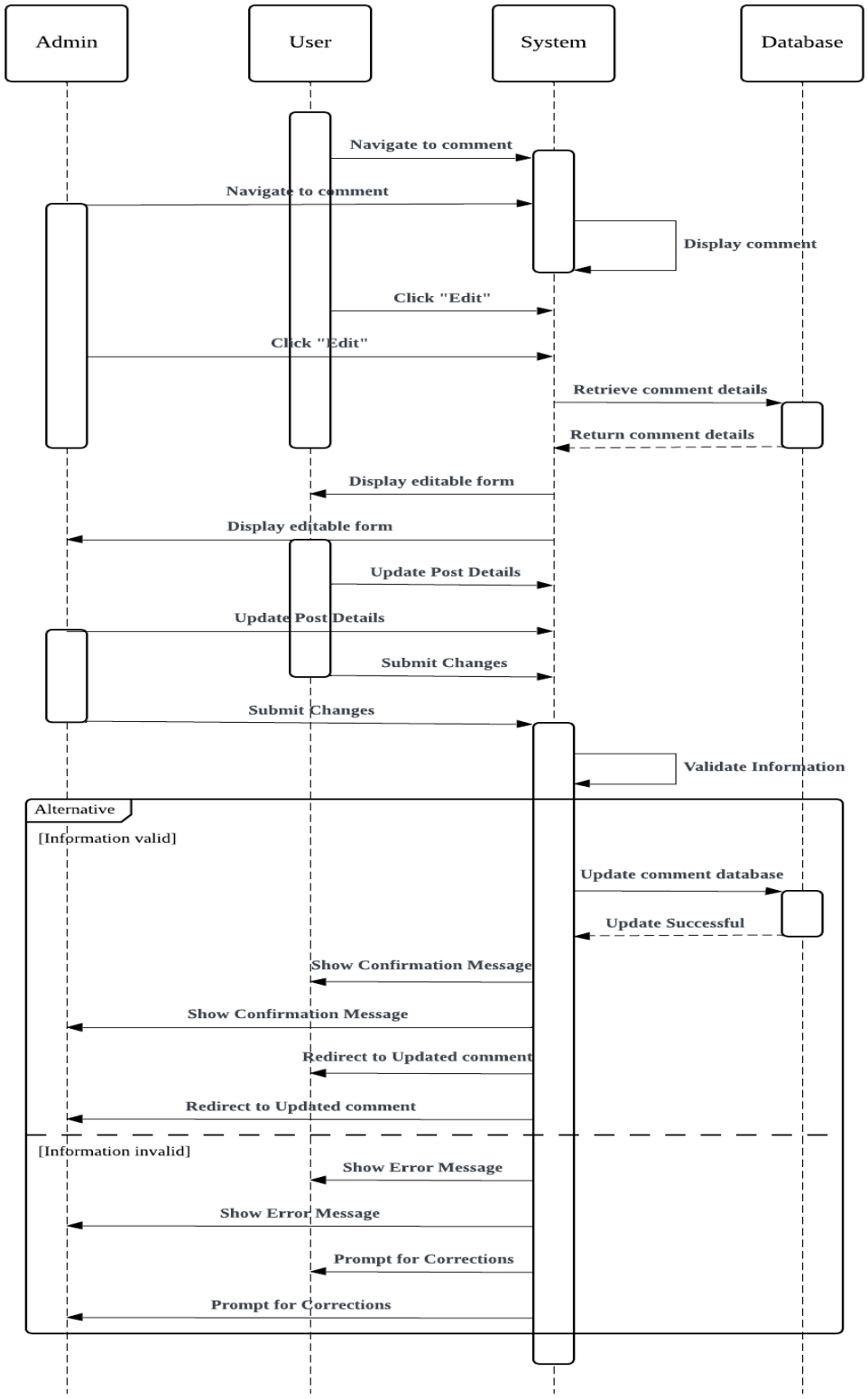


Figure 3.31: Sequence Diagram for Edit Comment

Table 3.20: Use Case Description for Edit Comment

Use Case	Edit Comment
Short Description	The actors edit the comment that they have created in the Communication Hub.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system. • The actor can only edit posts that they have created.
Postconditions	• The comment is updated in the database and displayed with the new content in the system.
Main Flow	1. The actor navigates to the comment that they want to edit. 2. The actor clicks on the "Edit" button. 3. The system retrieves the comment details and displays them in editable form. 4. The actor updates the comment content. 5. The actor submits the changes made. 6. The system validates the updated information. 7. The system updates the comment in the database. 8. The system displays a confirmation of success message and redirects the actor to the updated comment.
Exception Flow	6a. If the updated information is invalid (e.g., empty title or excessive length), then the system will display and error message and prompt the actor to correct the issue before resubmitting.

3.2.2.6.20 Delete Own Comment

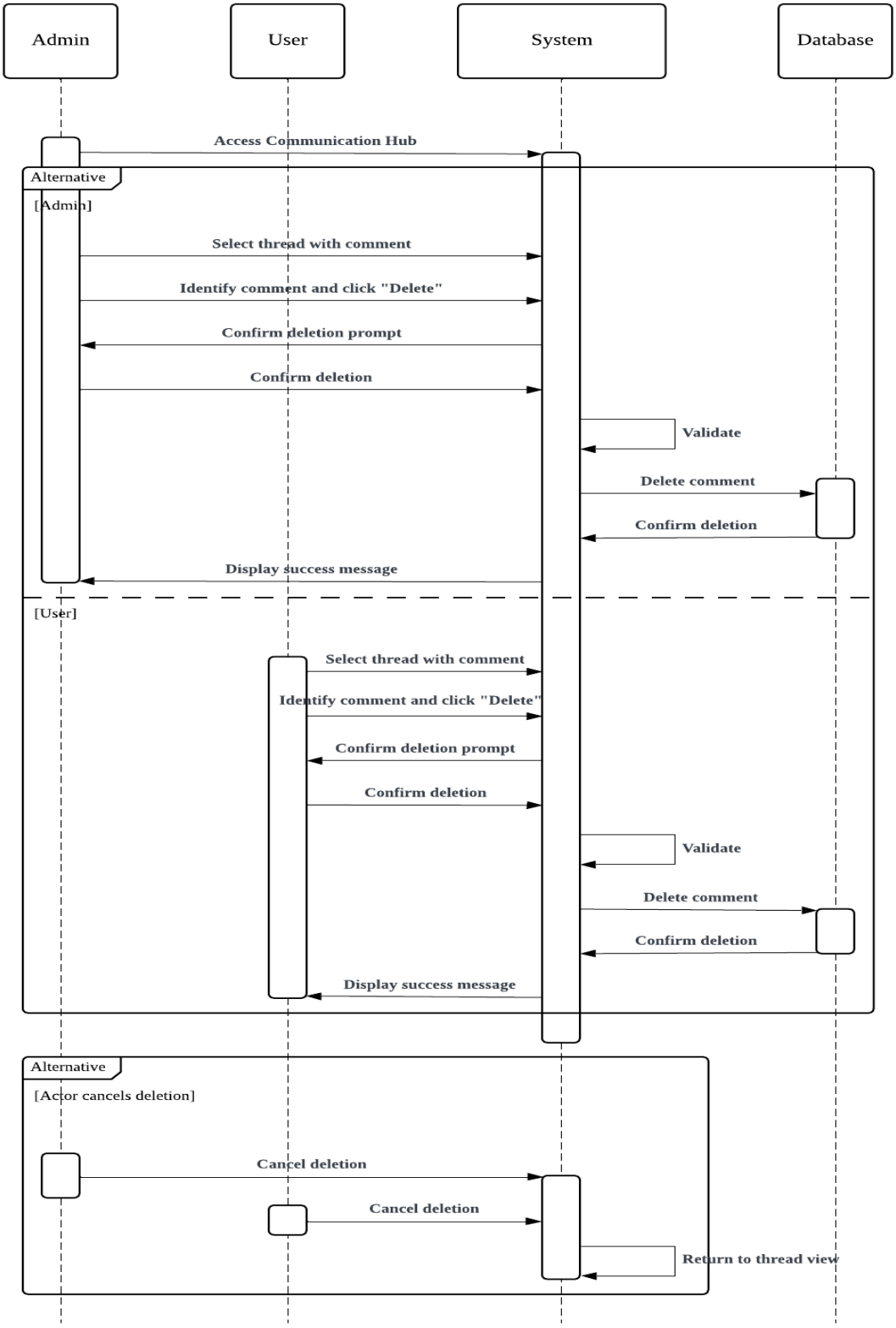


Figure 3.32: Sequence Diagram for Delete Own Comment

Table 3.21: Use Case Description for Delete Own Comment

Use Case	Delete Own Comment
Short Description	Admin or User wants to delete a comment they previously posted in a thread within the Communication Hub.
Actors	Admin, User
Preconditions	• The comment to be deleted must have been posted by the same actor. • The actor must already be logged into the system.
Postconditions	• The comment is removed from the Communication Hub.
Main Flow	1. The actor navigates to the Communication Hub section/page. 2. The actor selects the thread containing their comment. 3. The actor identifies their own comment and clicks the "Delete" button associated with it. 4. The system prompts the actor to confirm the deletion. 5. The actor confirms the action. 6. The system validates that the actor is the owner of the comment. 7. The system processes the deletion and removes the comment from the thread. 8. The system displays a confirmation message to the actor that the comment has been deleted successfully.
Exception Flow	5a. If the actor chooses not to confirm the deletion by clicking ‘cancel’, the system cancels the deletion process and returns to the thread view without making any changes.

3.2.2.6.21 Delete Own Post

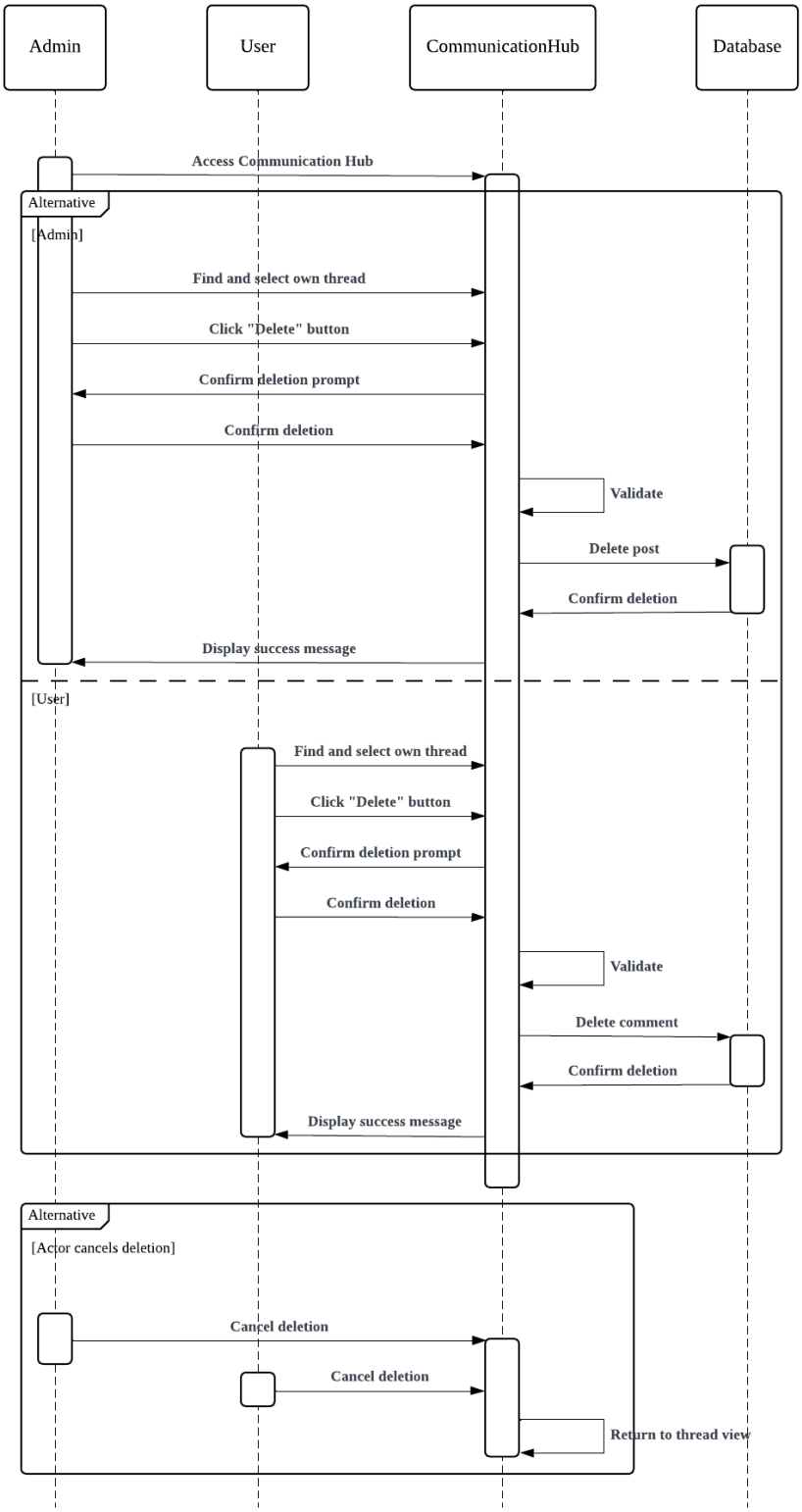


Figure 3.33: Sequence Diagram for Delete own Thread

Table 3.22: Use Case Description for Delete Own Post

Use Case	Delete Own Post
Short Description	Admin or User wants to delete a thread (post) they previously created within the Communication Hub.
Actors	Admin, User
Preconditions	• The post to be deleted must have been created by the same actor. • The actor must already be logged into the system.
Postconditions	• The comment is removed from the Communication Hub.
Main Flow	1. The actor navigates to the Communication Hub section/page. 2. The actor selects the thread (post) they want to delete from the list of threads. 3. The actor clicks the "Delete" button associated with the selected post. 4. The system prompts the actor to confirm the deletion of the post and any associated comments. 5. The actor confirms the deletion. 6. The system validates that the actor is the owner of the post. 7. The system removes the post and all associated comments from the database. 8. The system displays a confirmation message indicating that the post has been successfully deleted.
Exception Flow	5a. If the actor chooses not to confirm the deletion by clicking ‘cancel’, the system cancels the deletion process and returns to the thread view without making any changes.

3.2.2.6.22 View Notifications

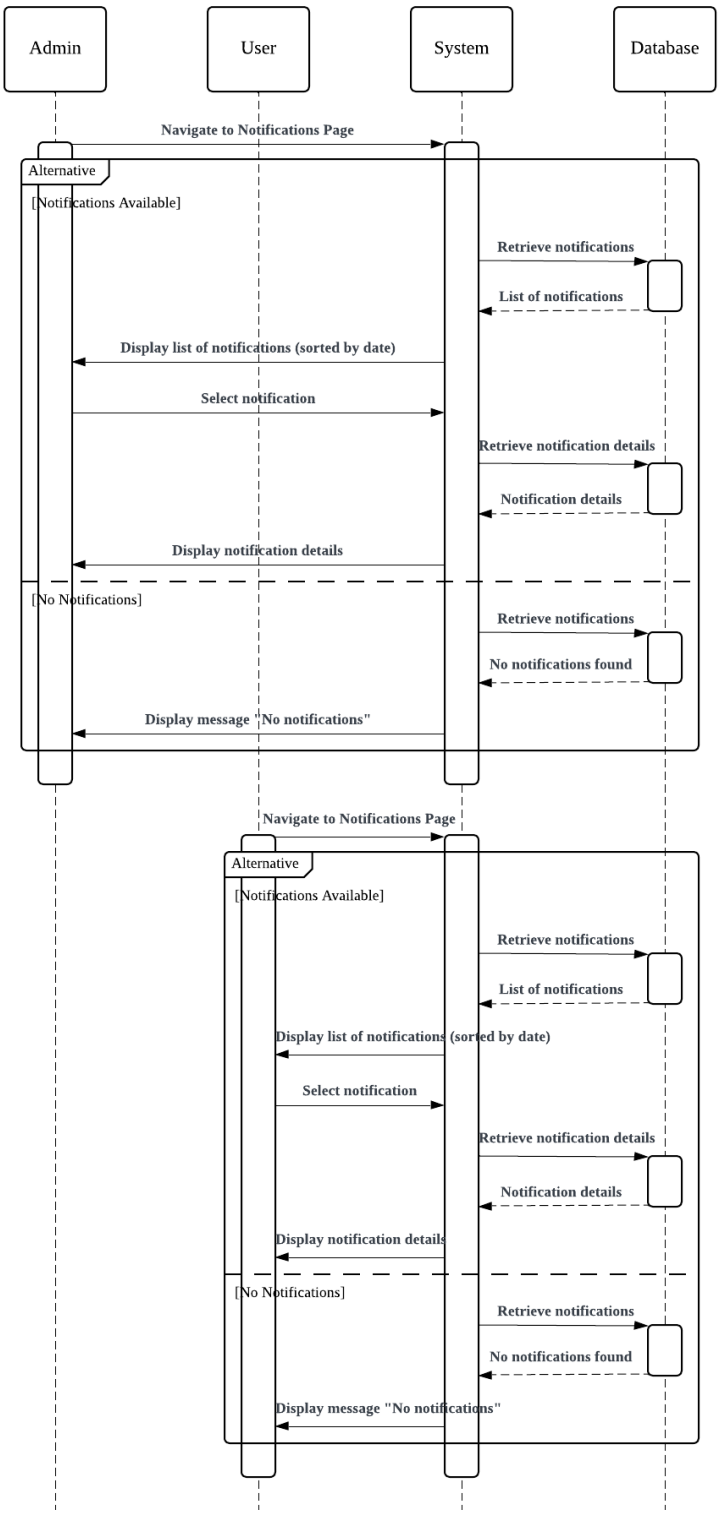


Figure 3.34: Sequence Diagram for View Notifications

Table 3.23: Use Case Description for View Notifications

Use Case	View Notifications
Short Description	The actors view notifications within the system. Notifications include updates about events, booking statuses, voting sessions, announcements, and updates from the Communication Hub.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system.
Postconditions	• The actor successfully views the list of notifications or the details of a specific notification.
Main Flow	1. The actor navigates to the Notifications Page. 2. The system retrieves the list of notifications from the database. 3. The system displays the list of notifications to the actor, sorted by date. 4. The actor selects a notification to view its details. 5. The system displays the content of the notification.
Exception Flow	3a. If no notifications exist in the system, then the system will display a message informing the actor as such.

3.2.2.6.23 View Organization Information

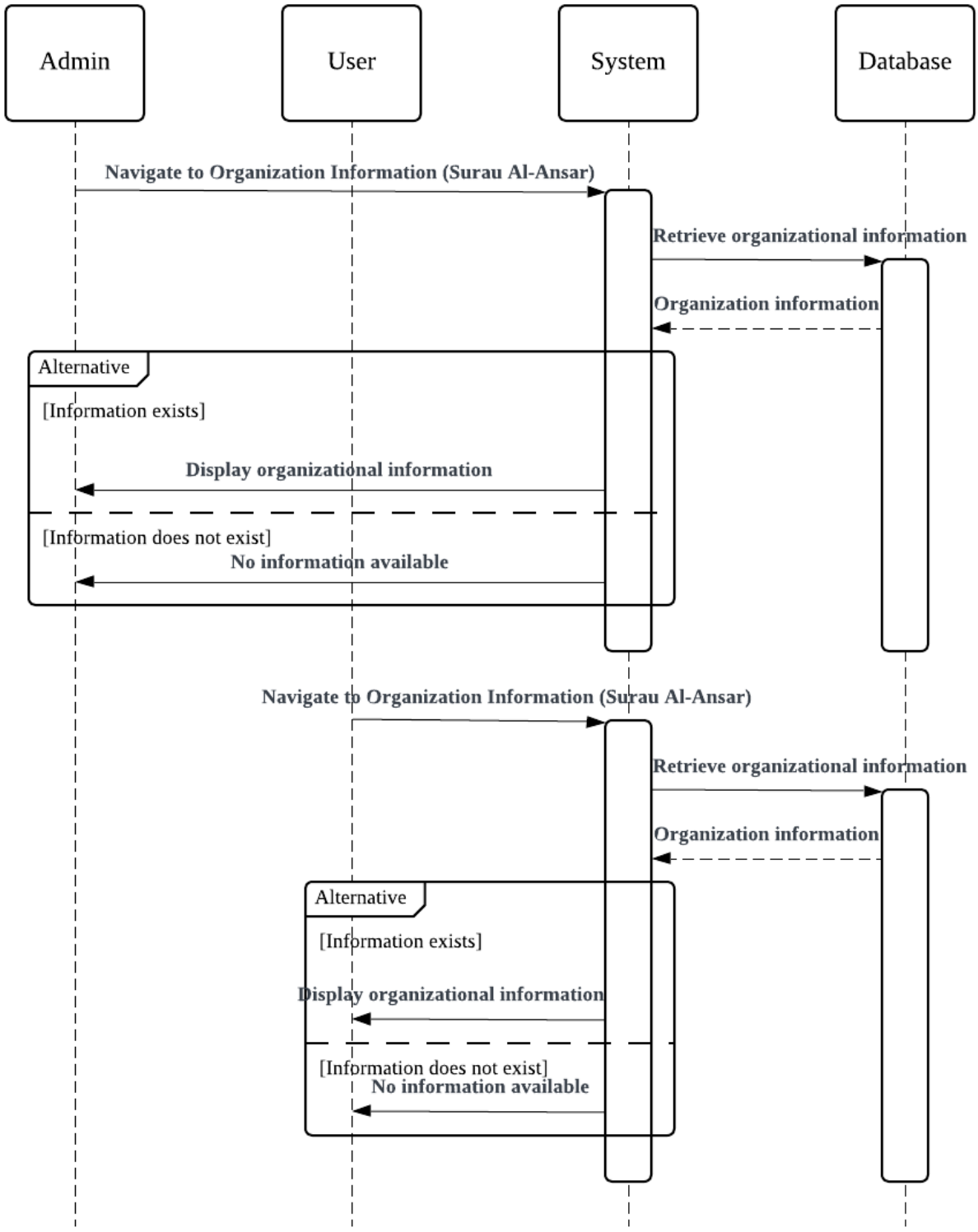


Figure 3.35: Sequence Diagram for View Organization Information

Table 3.24: Use Case Description for View Organization Information

Use Case	View Organization Information
Short Description	The actor gets to view detailed information on the surau's organizational structure, including the list of committee members, and other relevant details.
Actors	Admin, User
Preconditions	• The actor must already be logged into the system.
Postconditions	• The actor can view the organizational information.
Main Flow	1. The actor navigates to the Organization Information section/page (Surau Al-Ansar). 2. The system retrieves the list of organizational information from the database. 3. The system displays the relevant information.
Exception Flow	3a. If no notifications exist organizational information exists in the database, then the actor will not see what they intended to see.

3.2.2.6.24 Class Diagram of the Proposed System

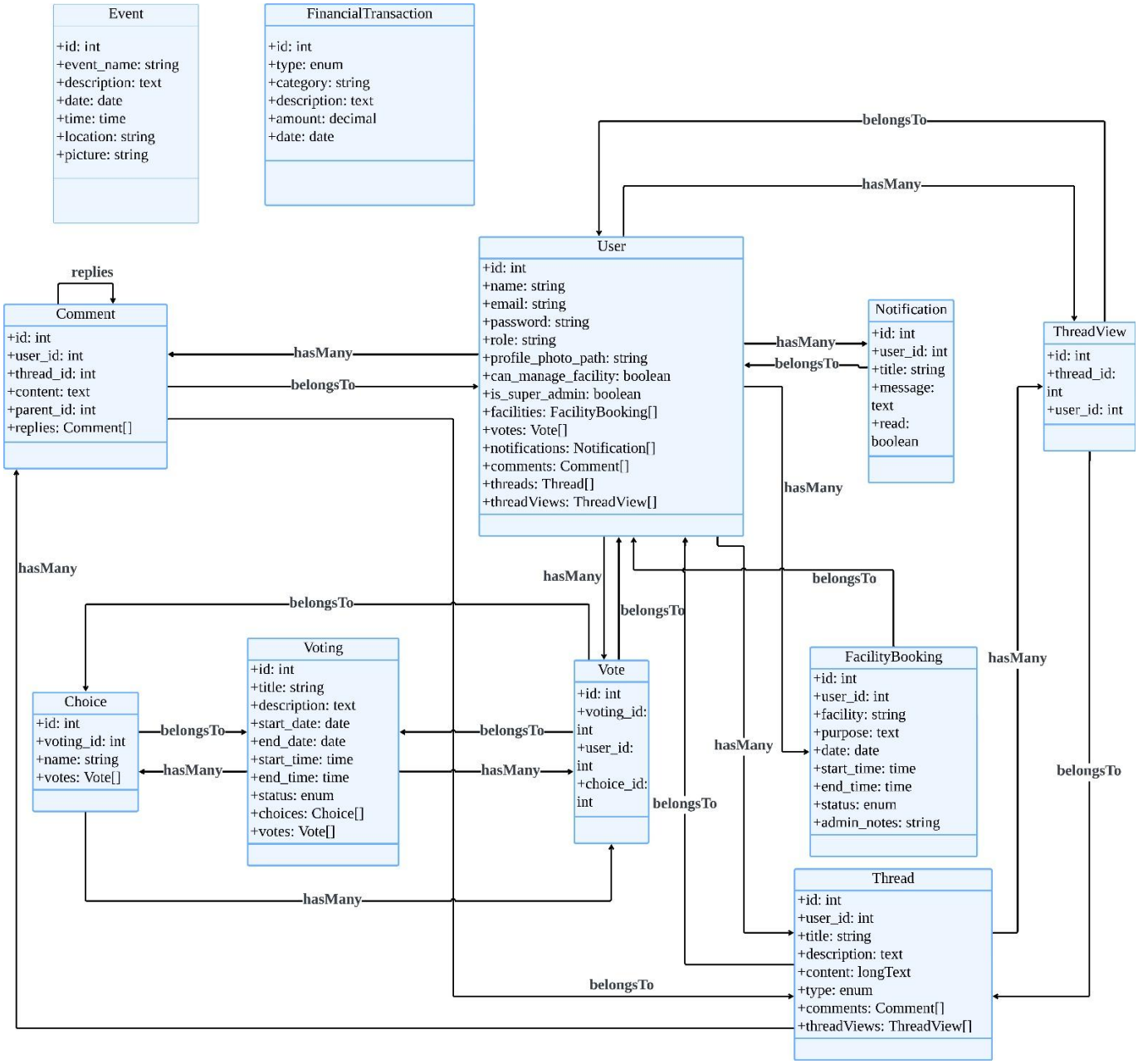


Figure 3.36: Class Diagram of the Proposed System

3.2.2.7 **Wireframes**

Wireframes play a pivotal role in enhancing the understanding and development process of the system through apt visualization. By possessing a good idea of the system's layout and flow, wireframes assist in ensuring that the practical implementation later on will go without many issues. Moreover, it sures that everyone pertinent to the development of the system will have the necessary level of understanding of the system's structure.

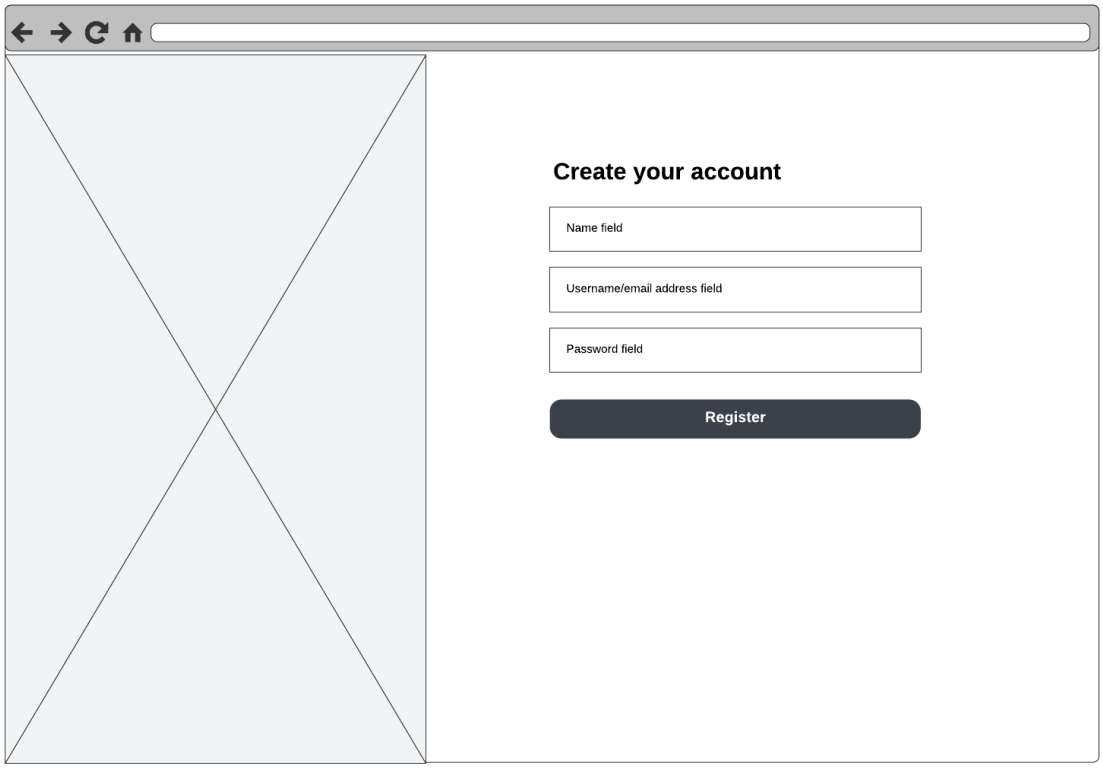


Figure 3.37: Registration Page

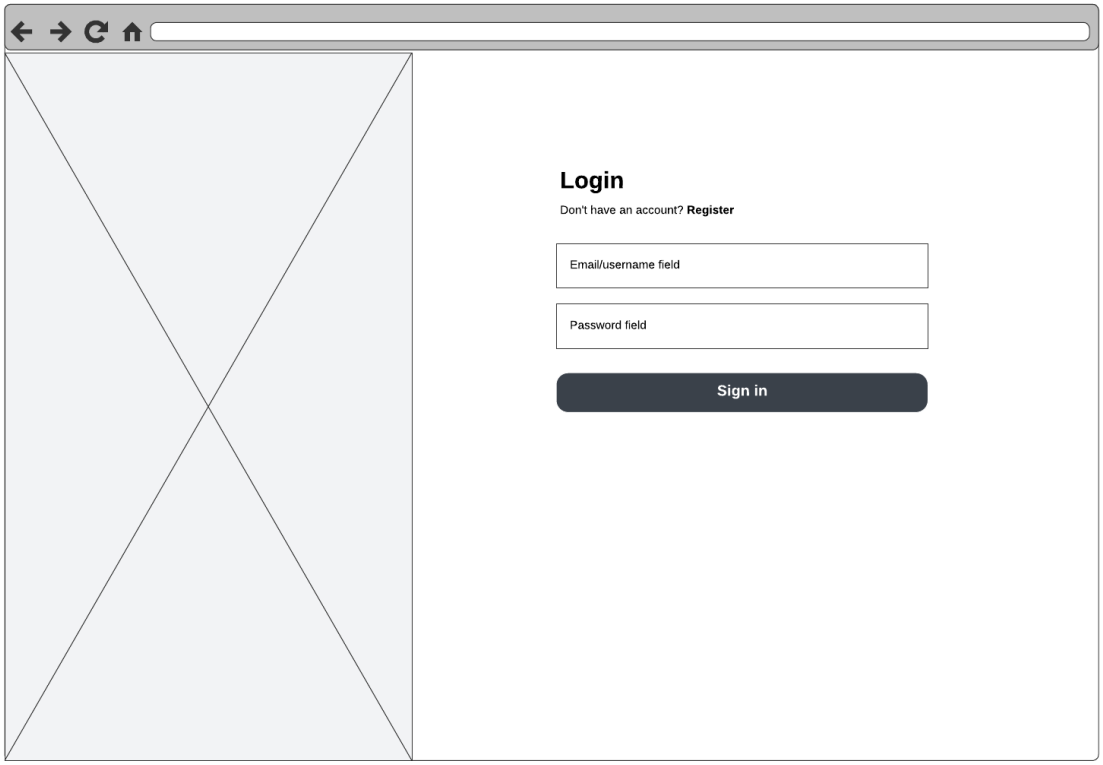


Figure 3.38: Login Page

As can be seen from Figure 3.37 and Figure 3.38, they are the Registration Page and Login Page, respectively. The wireframes depicting them showcases a simplistic yet easy to understand design. Admins and normal Users only need to input the required credentials to gain access into the system, be it through direct login for the Admin from pre-registration provided by the system administrator or through the creation of an account through registration before logging in for the case of normal Users. Only name, username, email address, and password are needed.

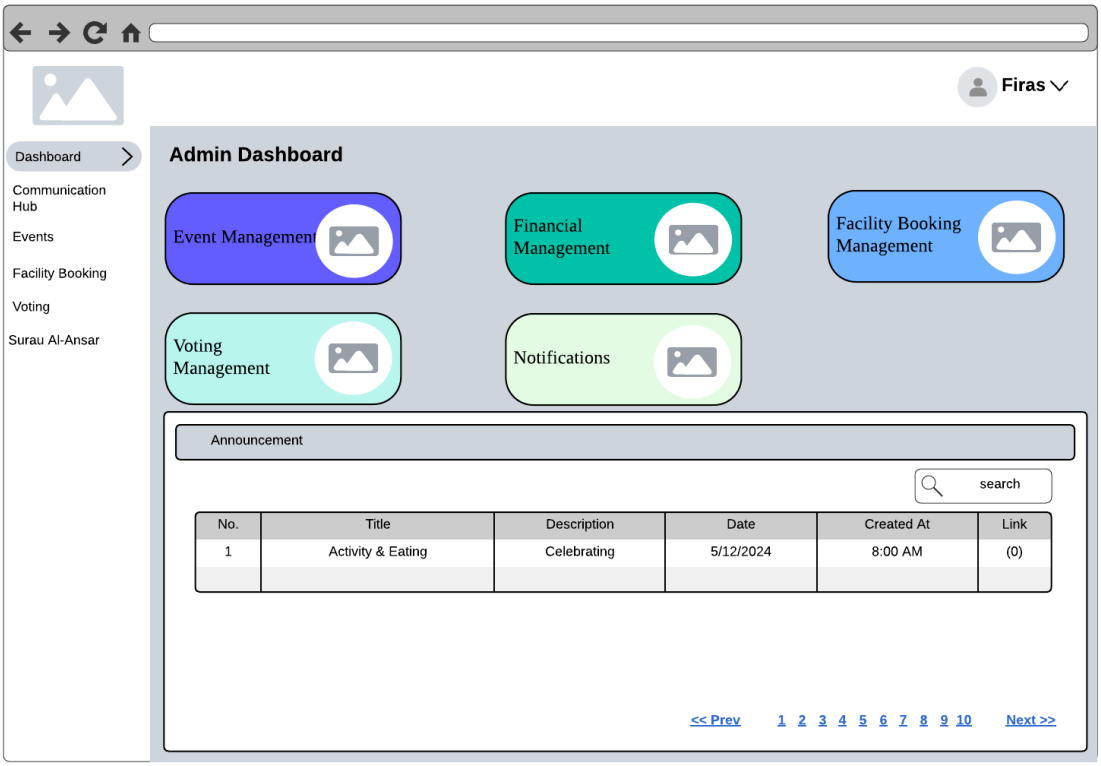


Figure 3.39: Admin Dashboard

The Figure 3.39 showcases the Admin Dashboard that the Admin will redirected to after logging into the system. Present on their user interface are the management modules that they can gain access to for specific operations. Besides that, there is the list of announcements, the sidebar for other pages that normal Users can also gain access to, the name and picture icon on the top-right corner, and the placeholder for logo of the system on the top-left that can be clicked in order to redirect users back to their respective dashboards.

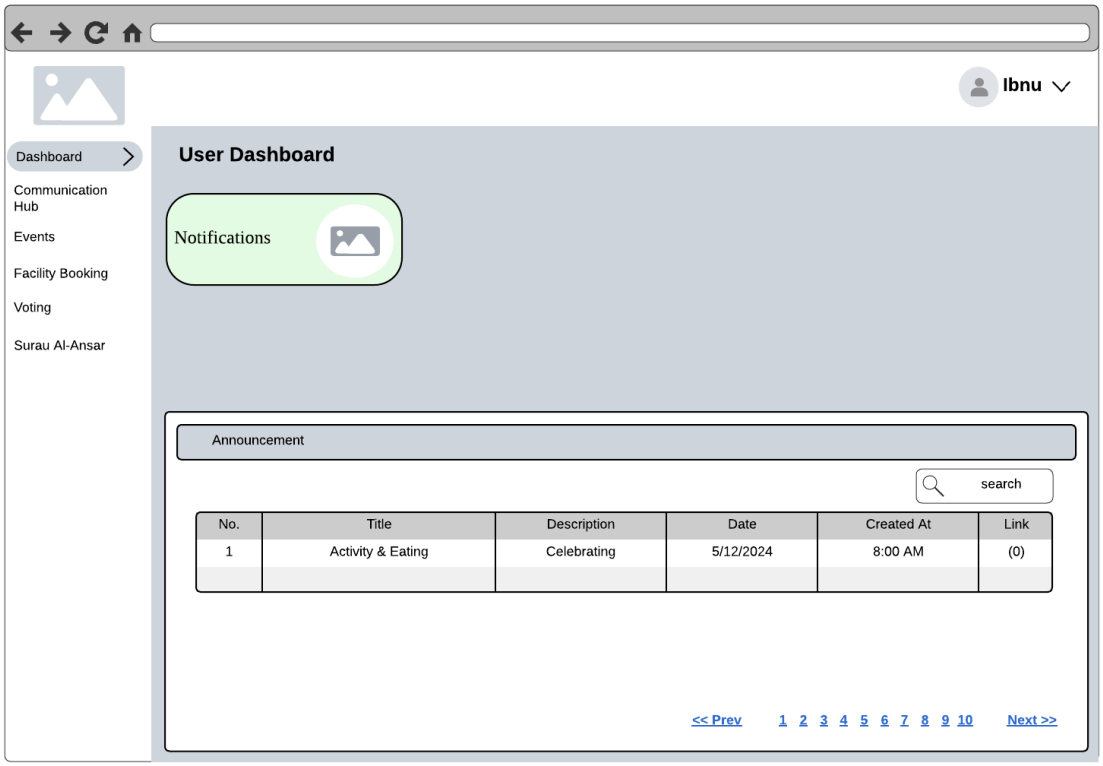


Figure 3.40: User Dashboard

The Figure 3.40 here, the User Dashboard, displays a user interface nearly similar to that of an Admin's, only with the lack of access to the other pages that are for operations that should only be handled by people with the role of Admin. What can be highlighted here is the "Notification" access that will redirect the user to the notifications page to read their notifications.

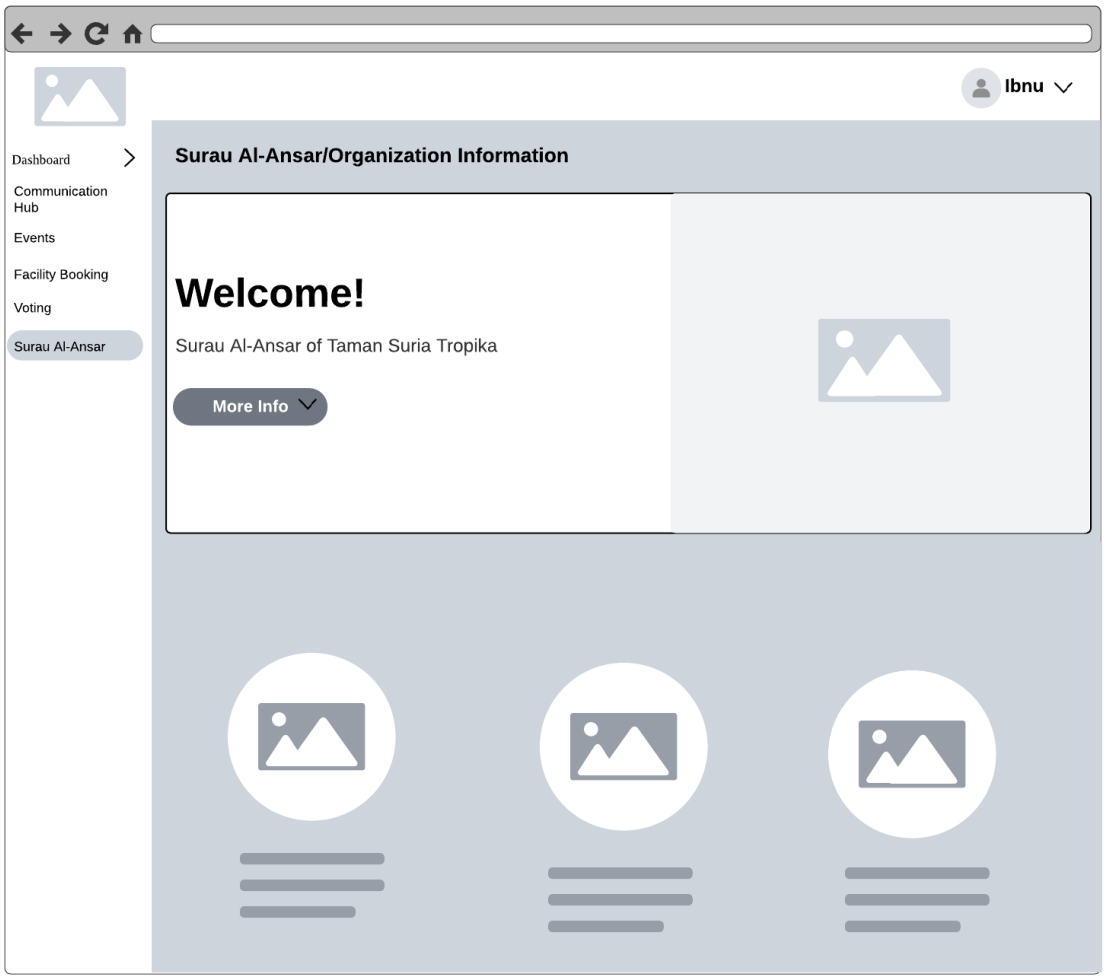


Figure 3.41: Organization Information Page

Here, shown in Figure 3.41, is the Organization Information Page. Users, both normal and Admin, can gain access to this page by clicking on the word “Surau Al-Ansar” that’s located on the sidebar. The user will be redirected to the page and the word “Surau Al-Ansar” will be highlighted as a consequence, as well. By visiting this page, users get to view detailed information on the surau’s organizational structure, including the list of committee members, and other relevant details.

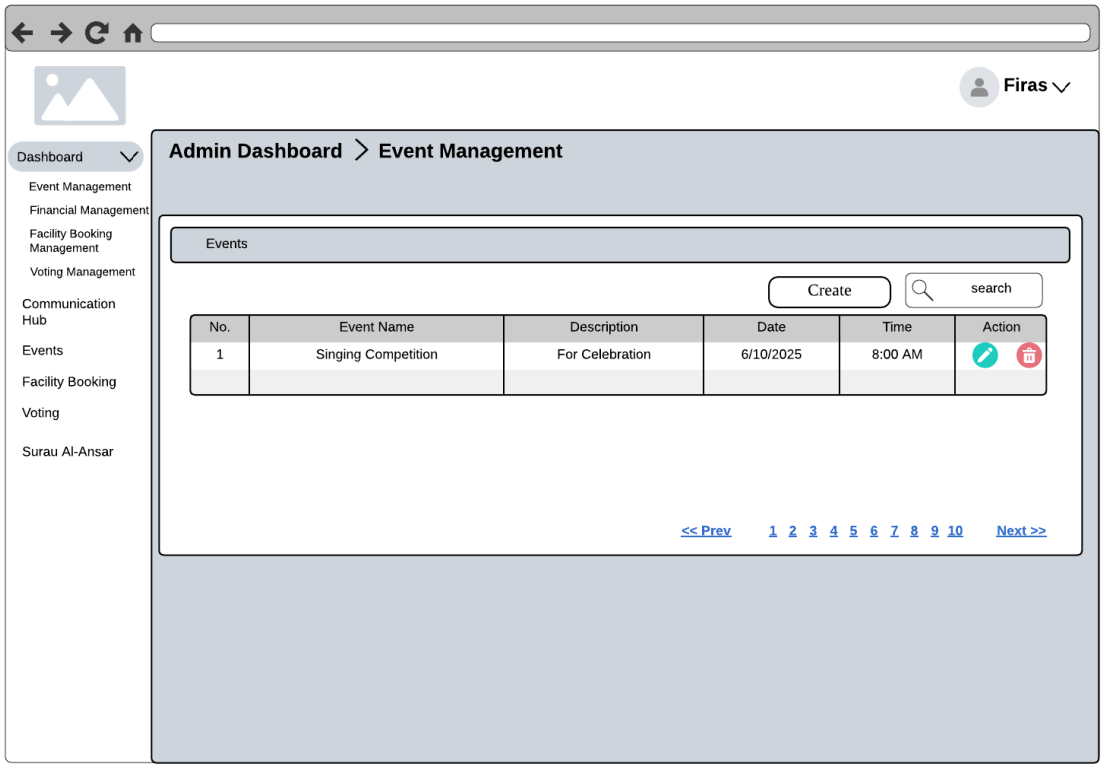


Figure 3.42: Event Management Page (Admin Access)

Figure 3.42 shows the Event Management Page that can only be accessed by people with the role of Admin from the Admin Dashboard. With this page, they will be presented with a list of created events with details attached to each and can even proceed to create more.

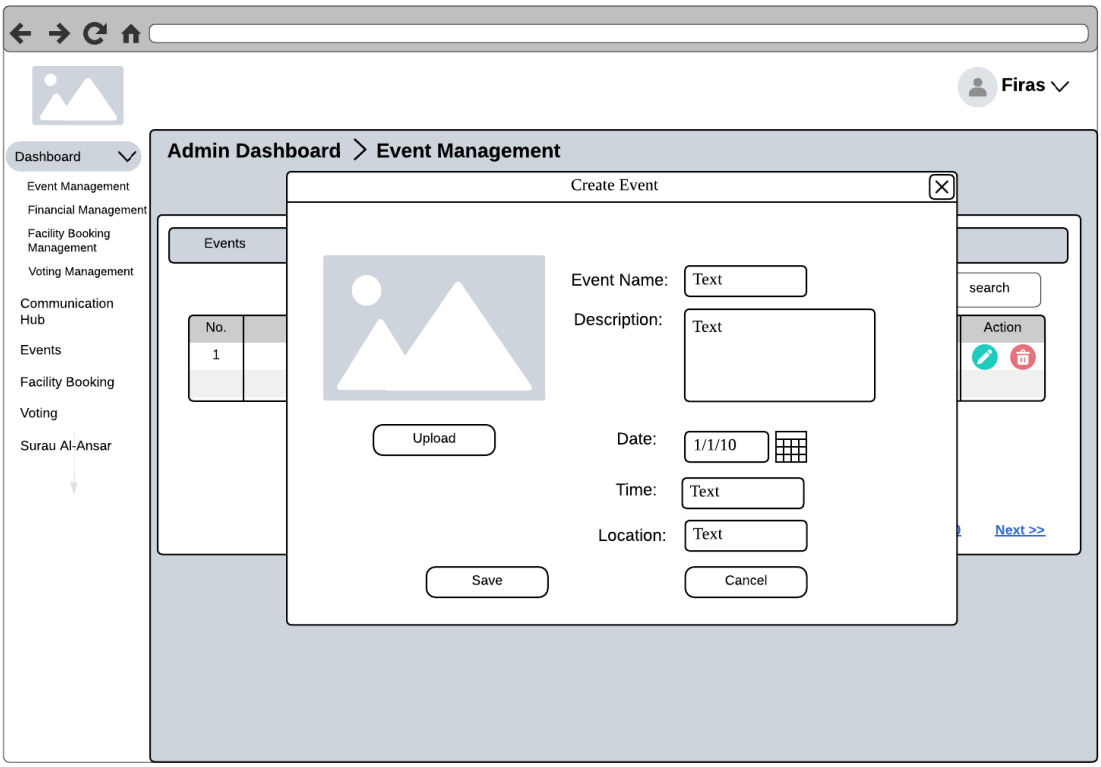


Figure 3.43: Event Management Page (Admin Access) - Create Event Form

Figure 3.43 here is the continuation from Figure 3.42. After clicking the "Create" button on the Event Management page shown in Figure 3.42, the Admin will be presented with a "Create Event" form, where they are required to fill in detail like the event name, description, date, time, location, and, optionally, upload a picture. After all that, they can either click "Cancel" or click "Save" to submit and finalize the creation.

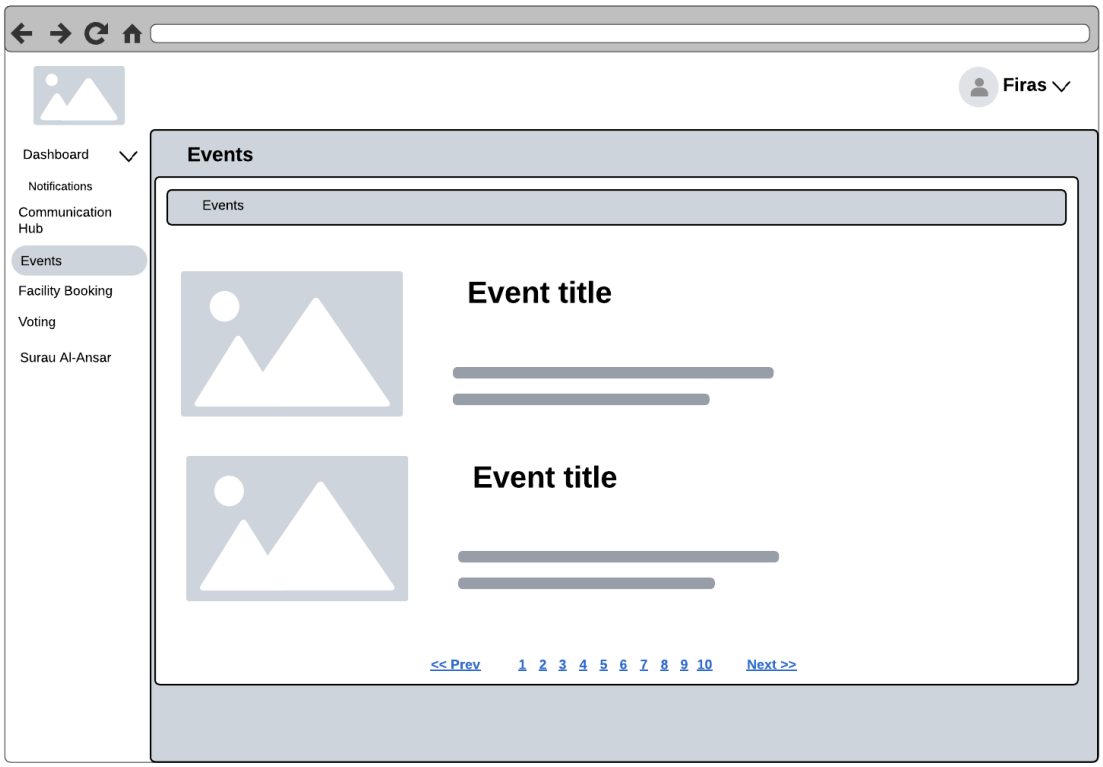


Figure 3.44: Events Page

Following up from the last figure, Figure 3.43, here is Figure 3.44, the “Events Page” that can be accessed by any user. They can simply select and click the word “Events” that is on the sidebar.

On this page, they can view the list of available and upcoming events that will be presented in a more desirable manner. Its details will be placed out there for them to read and be notified.

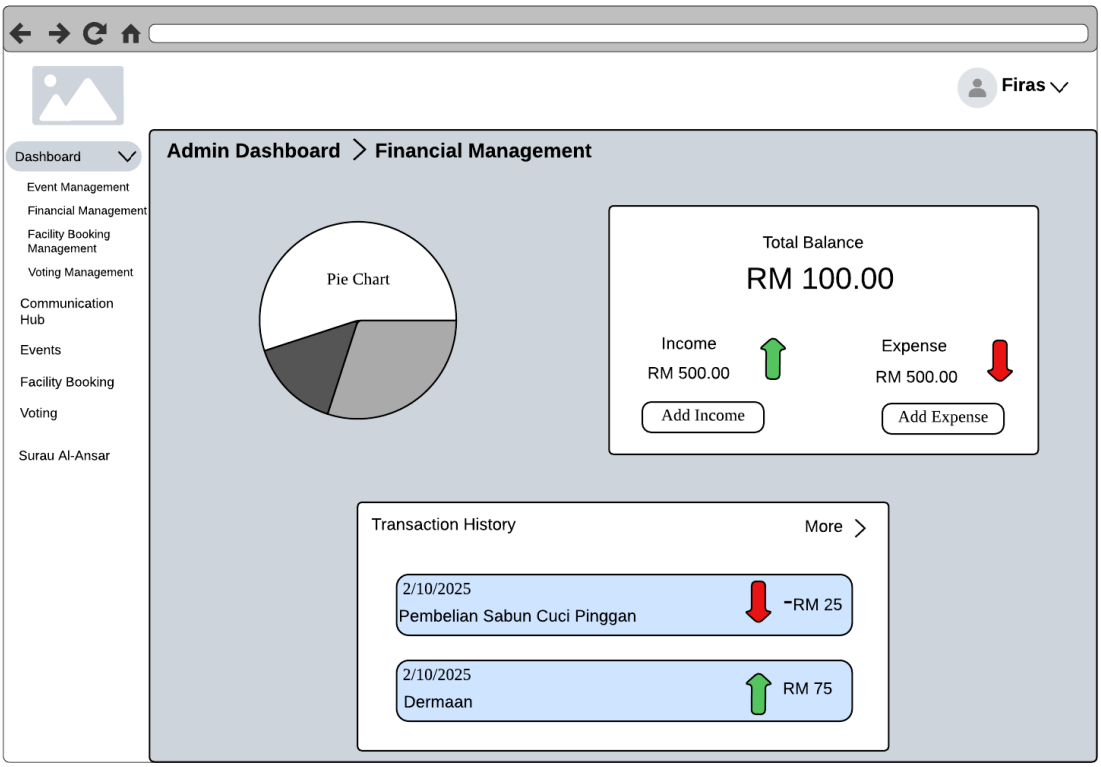


Figure 3.45: Financial Management Page (Admin Access)

Figure 3.45 displays the Financial Management Page that can only be accessed by users with the role of Admin. Here, they can obverse the list and summary of their finances. There is the total balance, the income, the expense, the transaction history, and the pie chart that is meant to depict the state of finance. Furthermore, users with the role of Admin can proceed to conduct actions like adding finance record either in terms of income or expense by clicking the available buttons named “Add Income: or “Add Expense”. They can also click the word “More” in the Transaction History section in order to see more financial information.

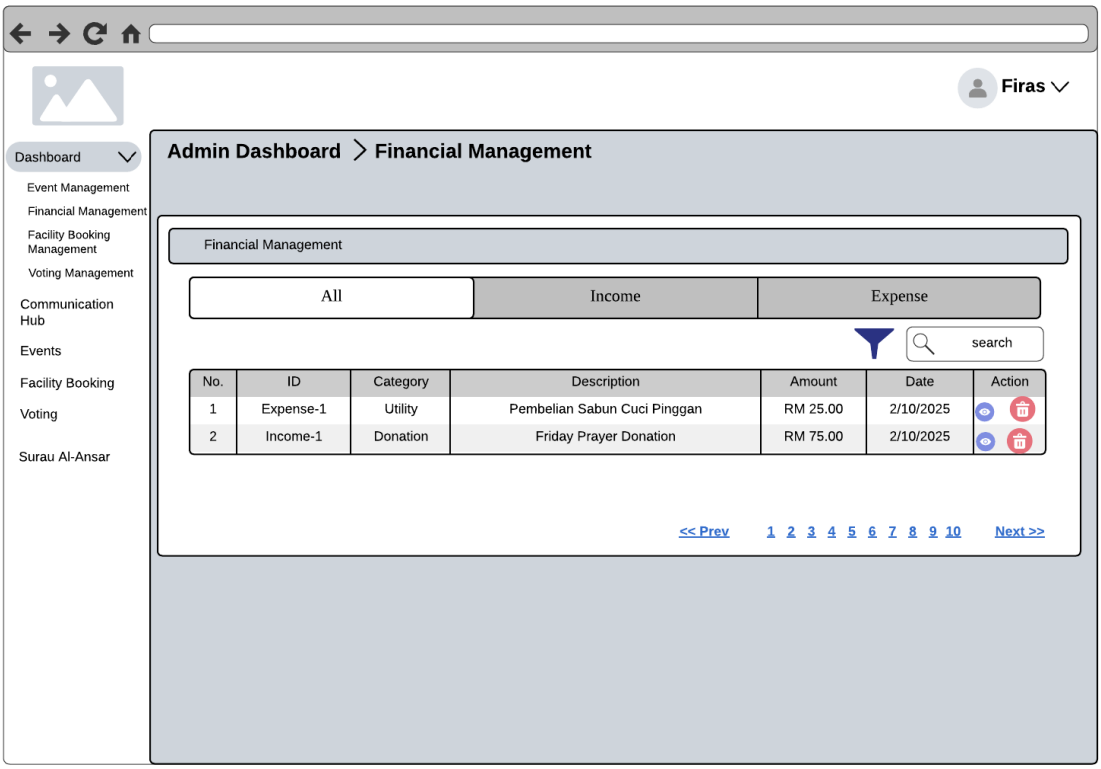


Figure 3.46: Transaction History Page

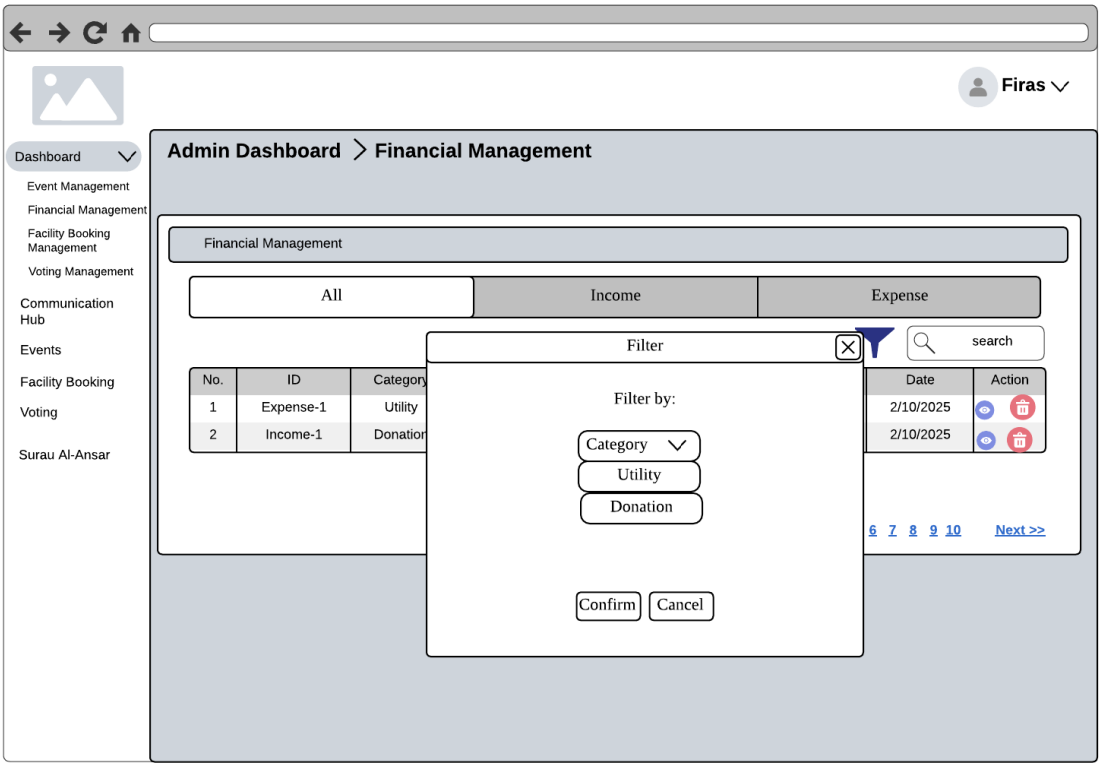


Figure 3.47: Filter Function

Figure 3.46 shows the Transaction History Page that can be accessed by clicking on the word "More" that was shown in the Transaction History section of the last figure, Figure 3.45. The Admin can use this page to see the full list of transactions that has been made and are even given the means to make their search for specific information easier through the use of the search bar on top of the table list. They can also use the tabs named "Income" and "Expense" to focus their search for information and, therefore, make it easier. The filter function next to the search bar also serves the same function, as can be seen from Figure 3.47, where the table can be filtered based on the category of the kept records.

Other than that, the Admin can also click on the 'eye' icon to see the details of a kept record and even edit it for updating. The 'trash' icon right next to it, on the other hand, it for deleting a record.

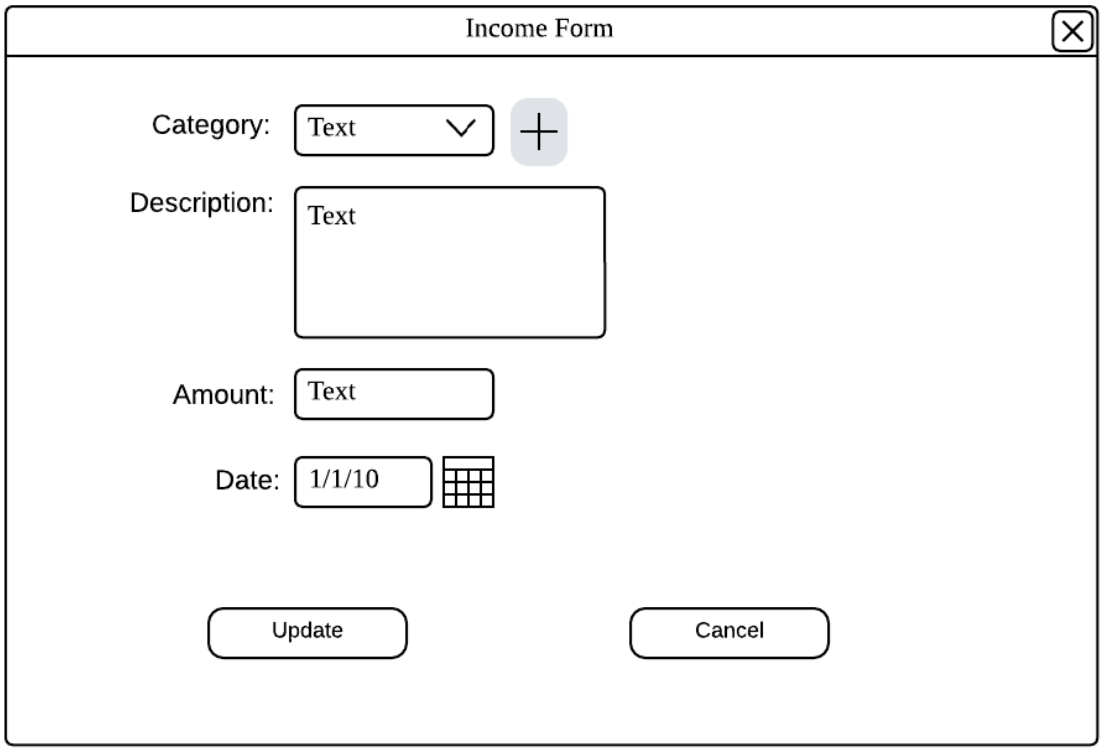


Figure 3.48: Income Form

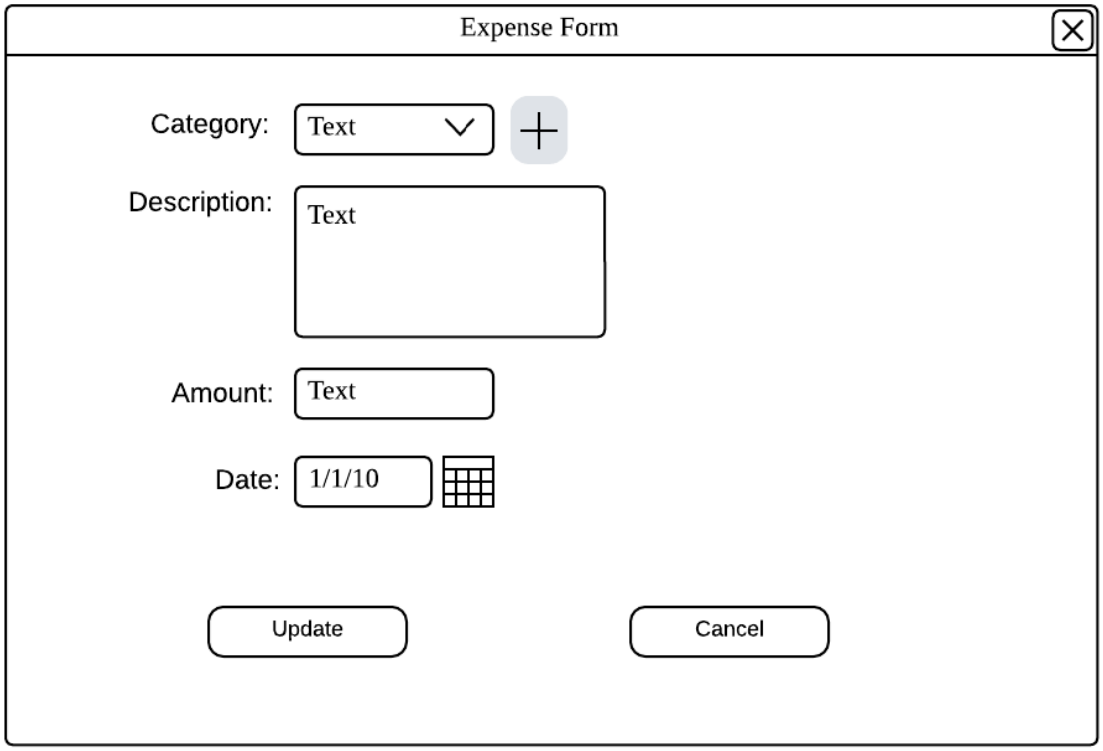


Figure 3.49: Expense Form

Shown in Figure 3.48 and Figure 3.49 are the Income Form and the Expense Form, respectively. They can be accessed by clicking the “Add Income” button and the “Add Expense” button shown in Figure 3.45, the first Financial Management Page.

Both forms require the admin to input details for category, description, amount, and date. The drop-down bar for the category section will display a selection of categories, just as shown in Figure 3.47 where the list of transaction history can be filtered by the categories ‘Utility’ and ‘Donation’. Besides that, the plus symbol right next to it is what will allow the Admin to add more categories based on the specification of the transaction.

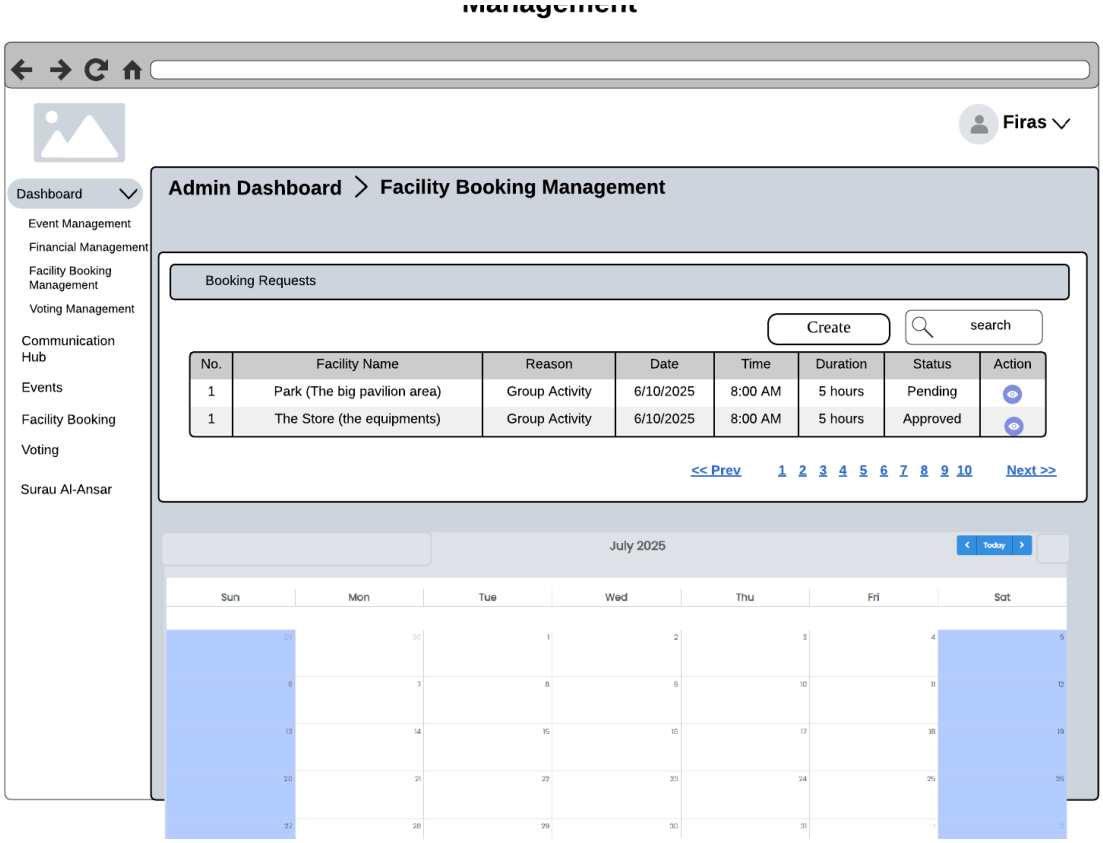


Figure 3.50: Facility Booking Management Page (Admin Access)

The Figure 3.50 here shows the Facility Booking Management Page that can be accessed by the Admin through their dashboard. This page is where the Admin can handle the matters pertaining to facility booking in the community. Here, they can see the list of pending requests and approved requests, alongside a calendar for keeping track of available time slots and existing bookings.

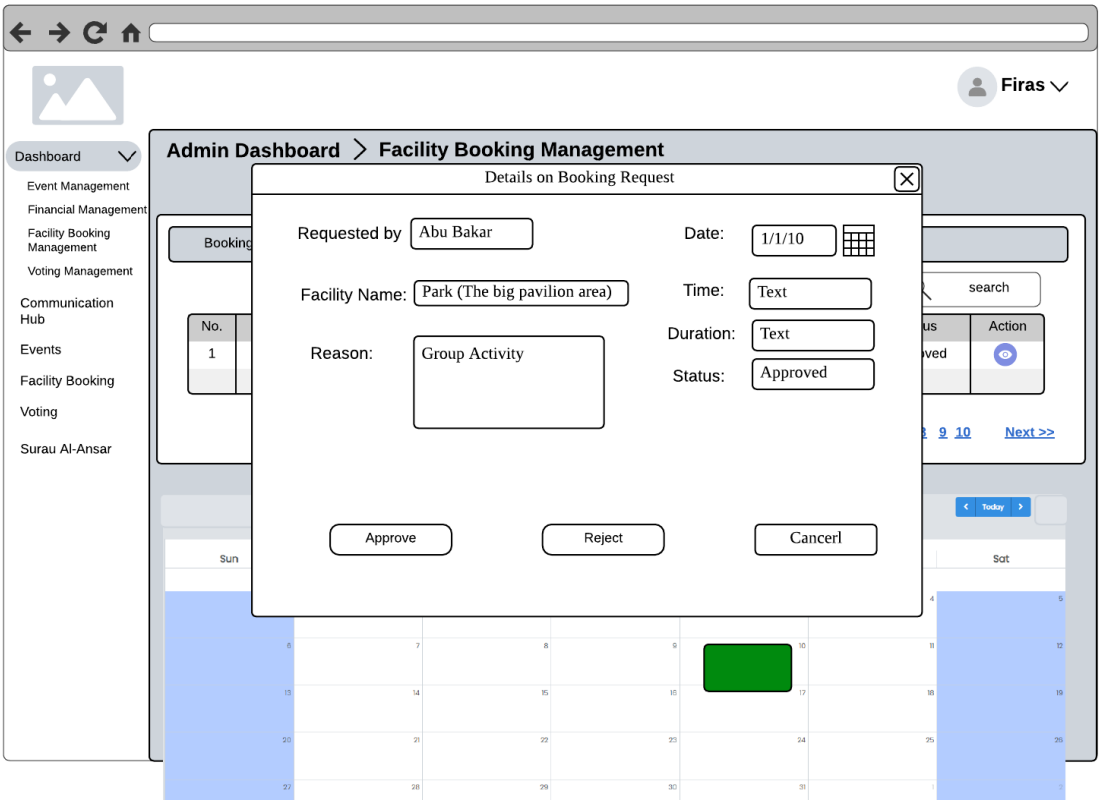


Figure 3.51: Details om Booking Request Form

What is current being shown in Figure 3.51 is what will happen after clicking the 'eye' icon under the 'Action' section in the list. The Admin will be presented with a better look into the details of the request that had been submitted to them. They can see the name of the person who requested the booking, the facility they intend to book, the reason behind the booking, the date of booking, the time of booking, and the expected duration in which the facility will be occupied. It is here that they can make the decision of either approving or rejecting the request.

Should they approve the request, the green highlight that can be seen in the calendar in Figure 3.51 will appear over the time which had been booked. Thus, helping keep track of things better.

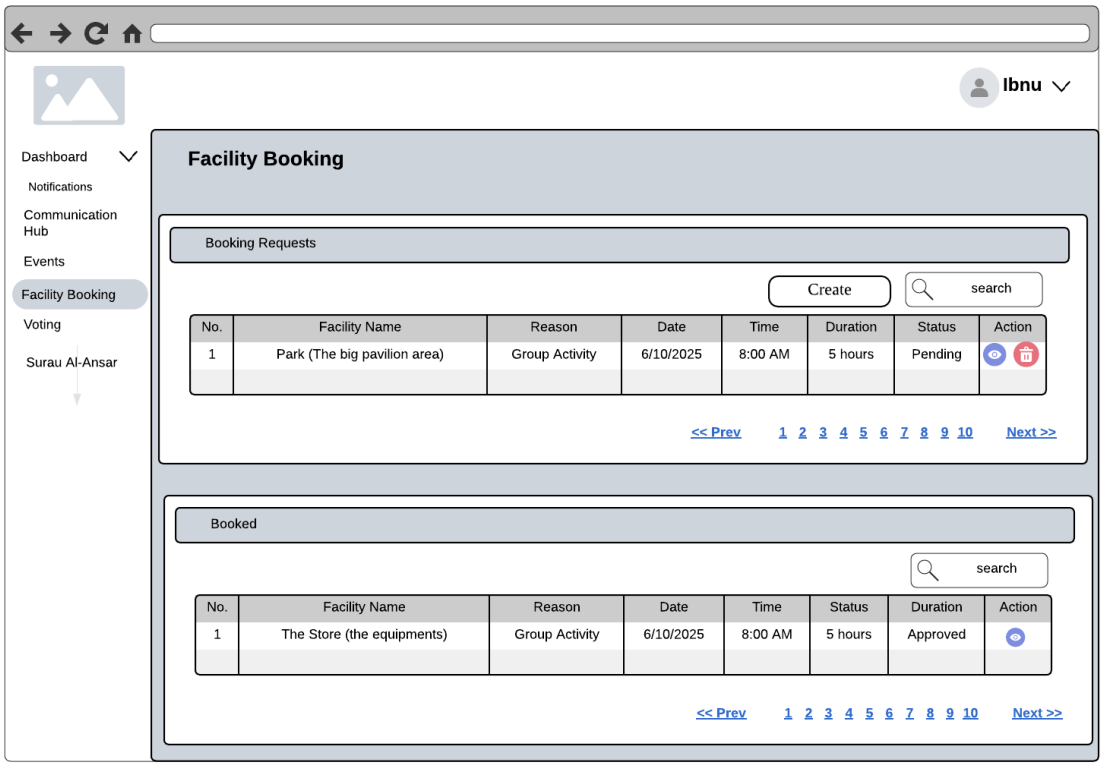


Figure 3.52: Facility Booking Page

The Figure 3.52 shows the Facility Booking Page that can be accessed by normal user to submit their booking requests to the Admin. As can be seen here, the user can keep an eye on their pending requests and their former requests which has already been approved before. They even have a choice of either creating more requests by clicking on the 'Create' button in the booking request section or deleting their still pending requests by clicking on the 'trash' icon in the 'Action' section.

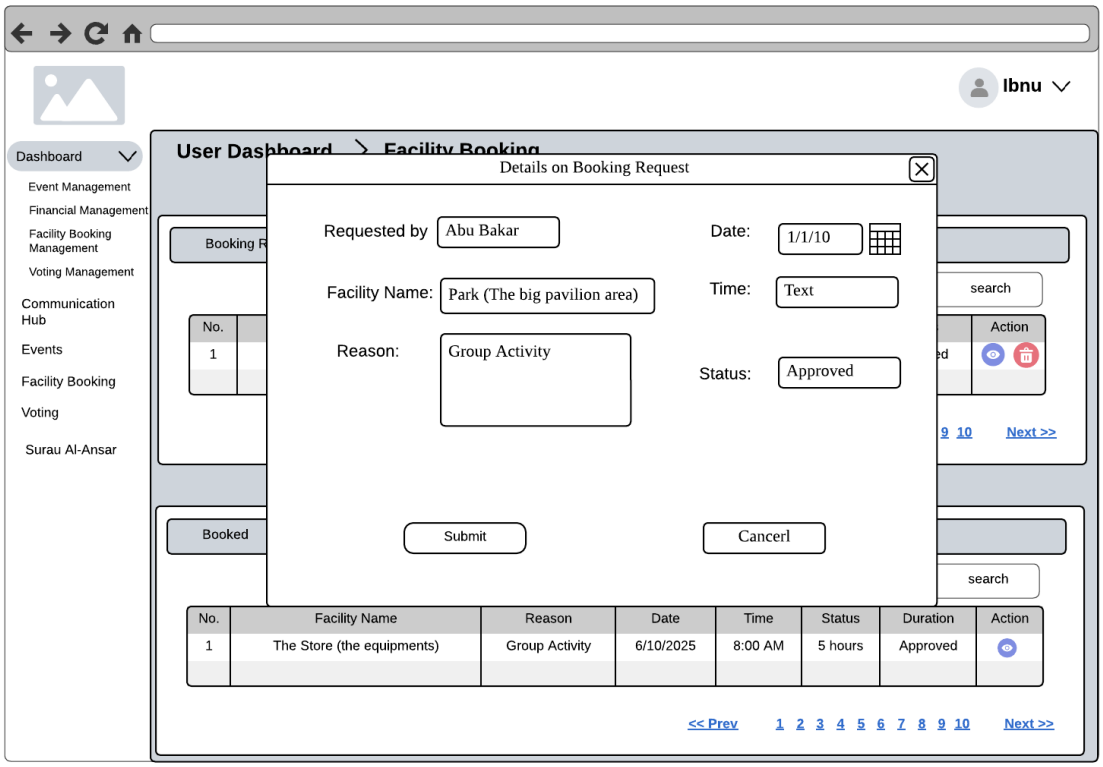


Figure 3.53: Details on Booking Request Form (normal User Side)

Shown here in Figure 3.53 is the Detail on Booking Request Form that will appear once the user clicks on the ‘eye’ icon of one of the records. It is similar to what has been shown on the Admin’s side in Figure 3.51.

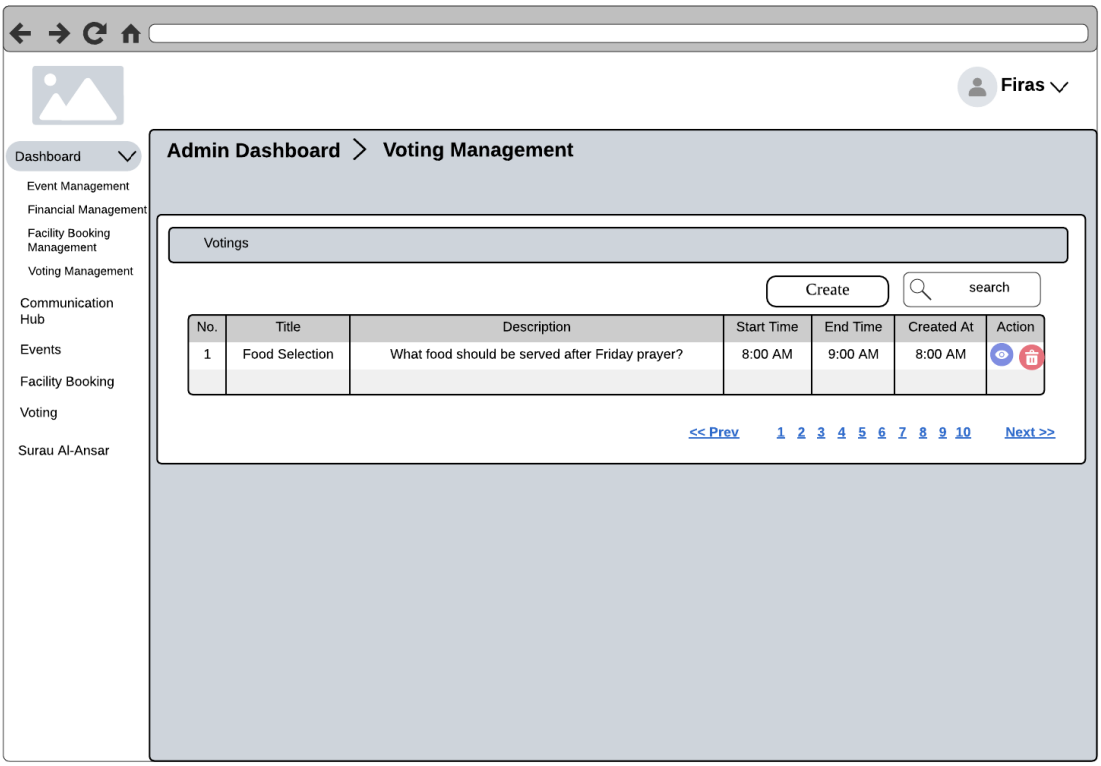


Figure 3.54: Voting Management Page (Admin Access)

In Figure 3.54, we can see the Voting Management Page that can only be accessed by users with the role of Admin through the use of their dashboard. With this page, the Admin can see the list of voting sessions that have been created and carry out action likes ‘view more’ by clicking on the ‘eye’ icon or deleting the session by clicking on the ‘trash’ icon. They can also click on the ‘Create’ button to directly create a voting session.

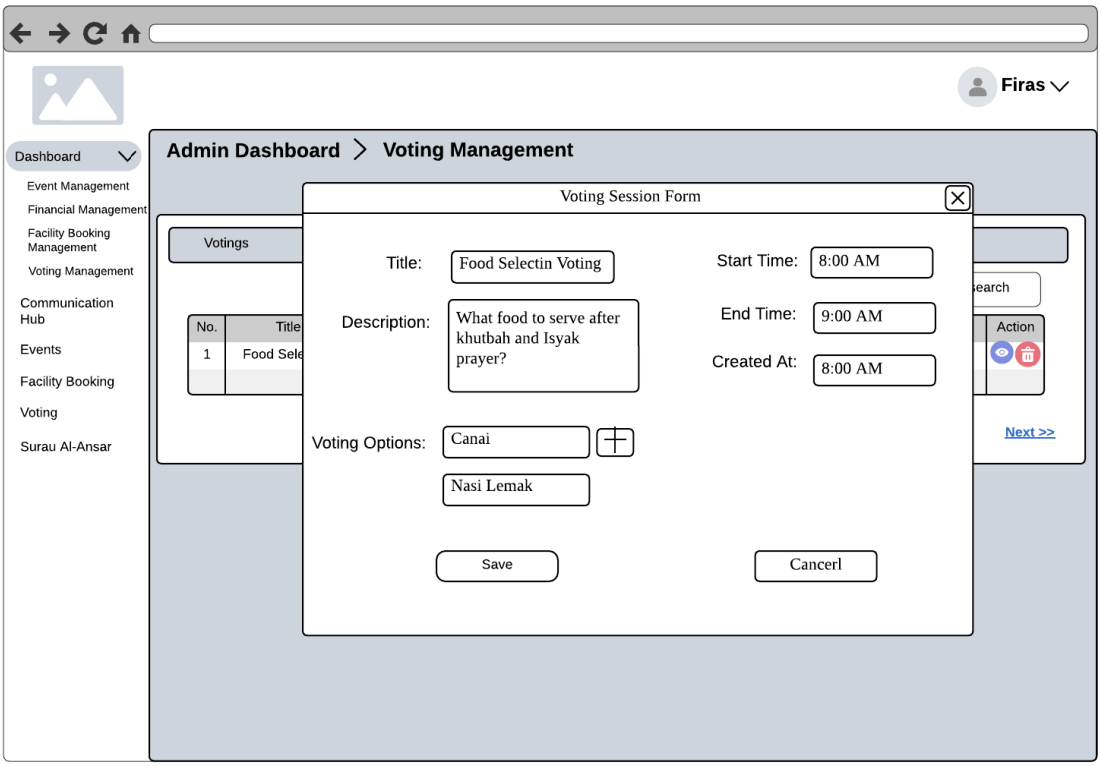


Figure 3.55: Voting Session Form

Figure 3.55 displays the Voting Session Form that appears after the Admin clicks on the button labelled ‘Create’. To create a voting session, they have to input the title of the voting session, the description of the voting session, the voting options available, the start time of the session, and the end time of the session. The ‘Created at’ section will display the time based on when the session had been created. All of these details can be reviewed once more by clicking on the ‘eye’ icon of the record.

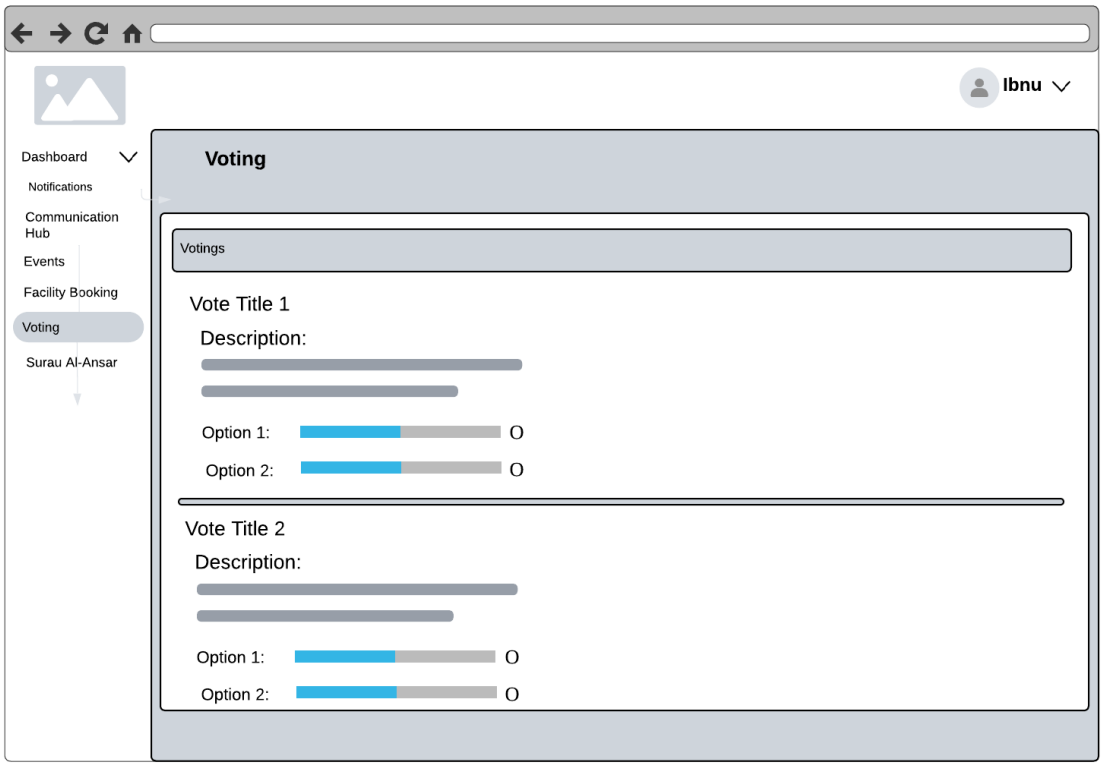


Figure 3.56: Voting Page

Figure 3.56 shows the Voting Page which can be accessed by both normal users and people with the role of Admin just by clicking on the word 'Voting' that is on the sidebar. In this page, they can see all the votes that had been created and currently in session. The users read what each voting session entails and place their votes.

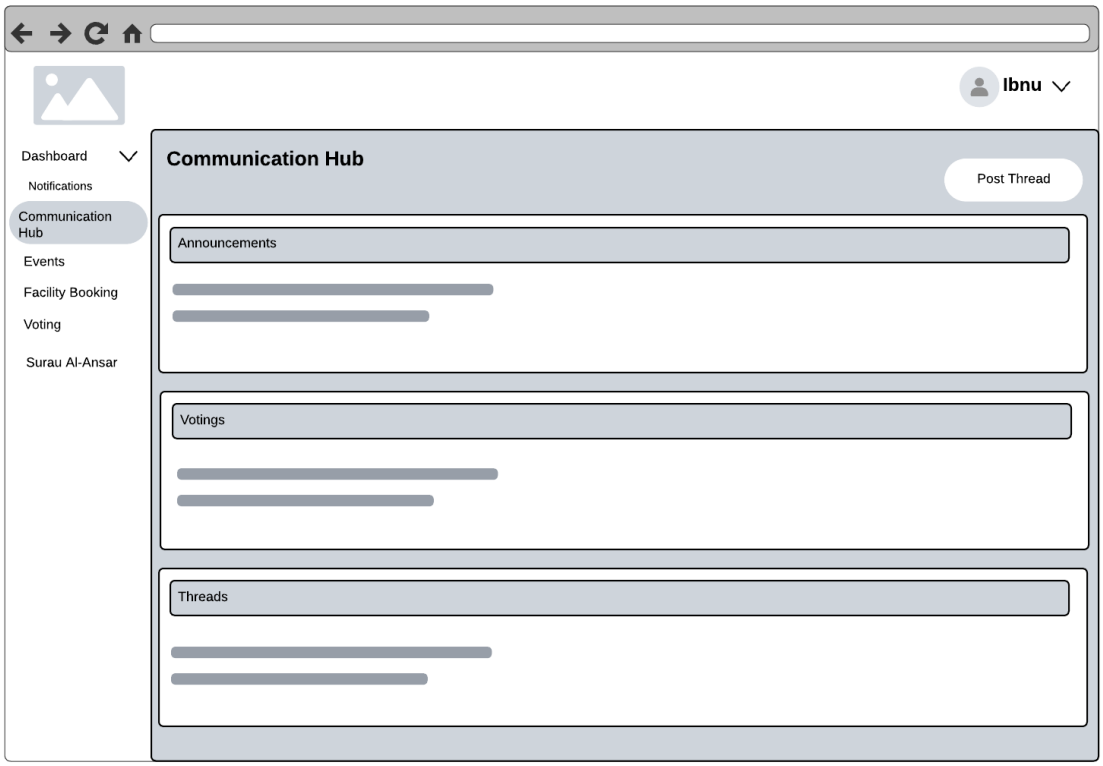


Figure 3.57: Communication Hub Page

Figure 3.57 shows the Communication Hub Page that can be accessed by clicking on the word 'Communication Hub' on the sidebar. In this page, users can initially see announcements, available voting, and threads arranged into their own proper sections in the page. Rather than the full view of each announcement, voting session, or thread, this page will just showcase the simplest information of some from among each section, like the title and time they were added. Only when a user directly clicks one of them will they be directed to see more.

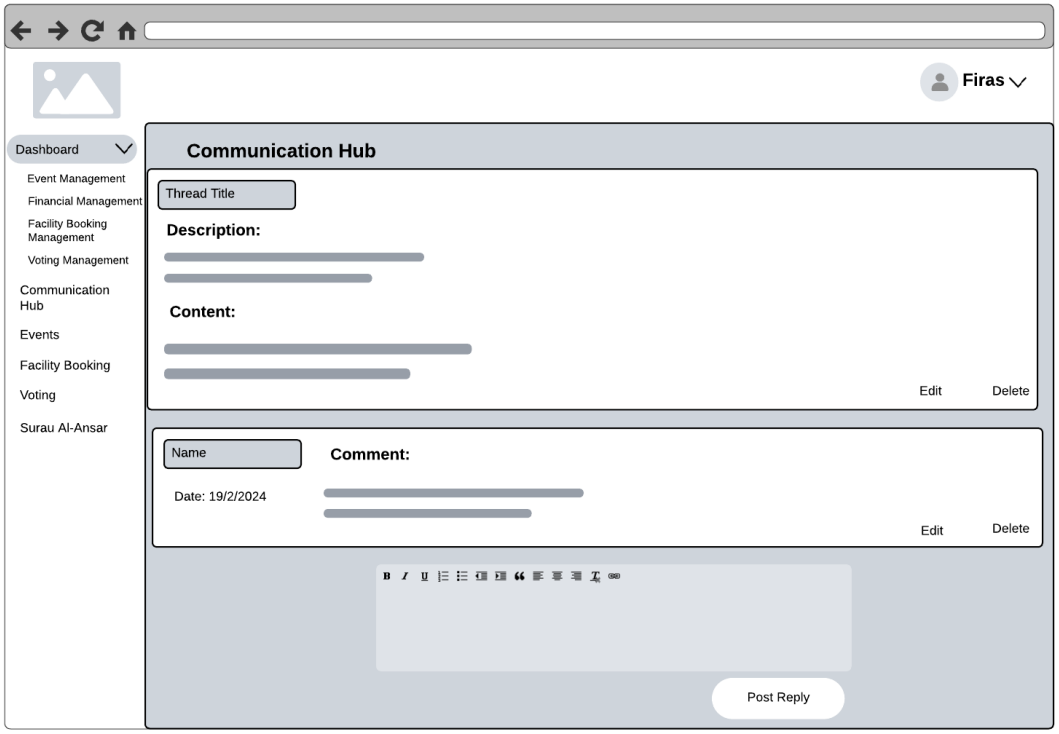


Figure 3.58: Further Into the Communication Hub (Thread Example)

Figure 3.58 shows what happened after clicking one of the threads from the thread section shown on Figure 3.57. The user will be presented with the full content of the thread, where they can see the original post and the comments associated with it. The user will also be able to post their own comment by typing in their message through the message box at the bottom of the thread. Other than that, the user may also edit or delete their comment or thread if they were the one who originally made it.

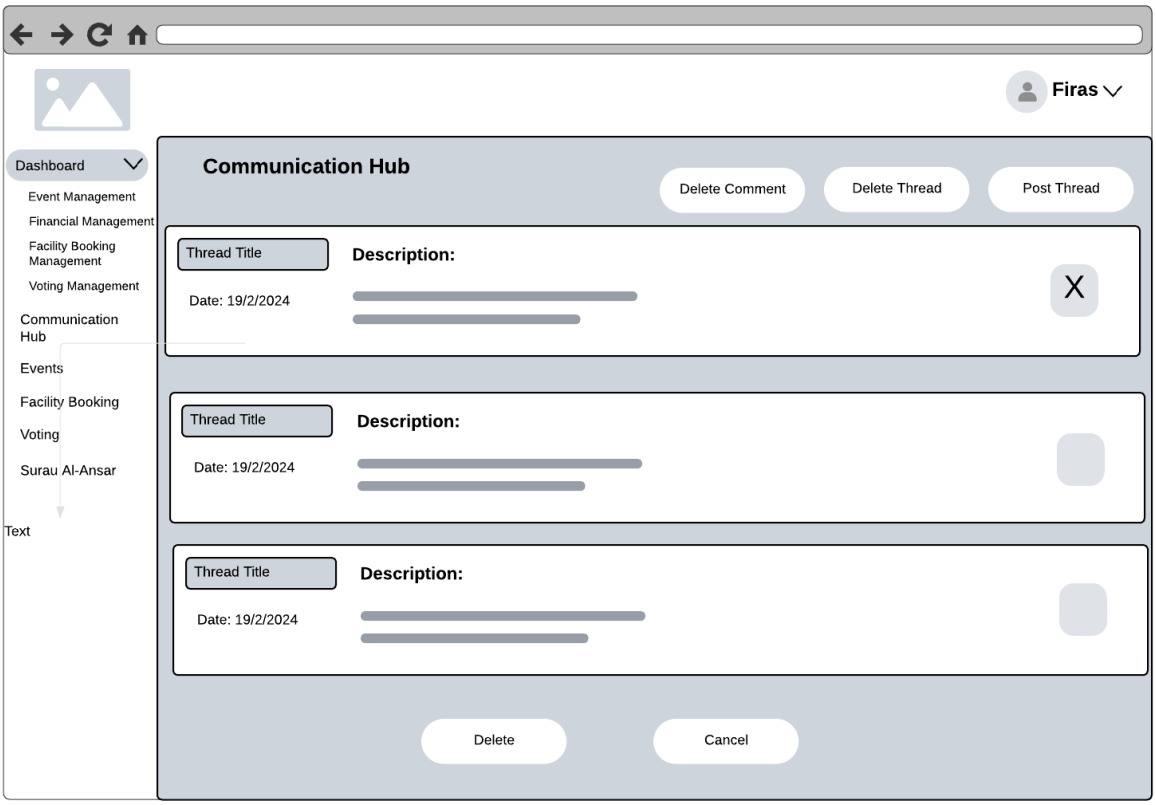


Figure 3.59: Communication Hub (Admin Access/Interface)

If the user has the role of Admin, their interface will be different compared to a normal user. An Admin will have the responsibility and option to moderate threads and comments. Thus, as shown in Figure 3.59, they either delete a thread and all the comments associated with it or they can delete an individual comment. Clicking the 'Delete Thread' button located at the top-right of the page will instantly remove the thread, while clicking the 'Delete Comment' button right next to it will open up a display as shown in Figure 3.59, where the Admin can select and mark the comments that are to be deleted before clicking the 'Delete' button at the bottom of the page or just press the 'Cancel' button to cancel the process.

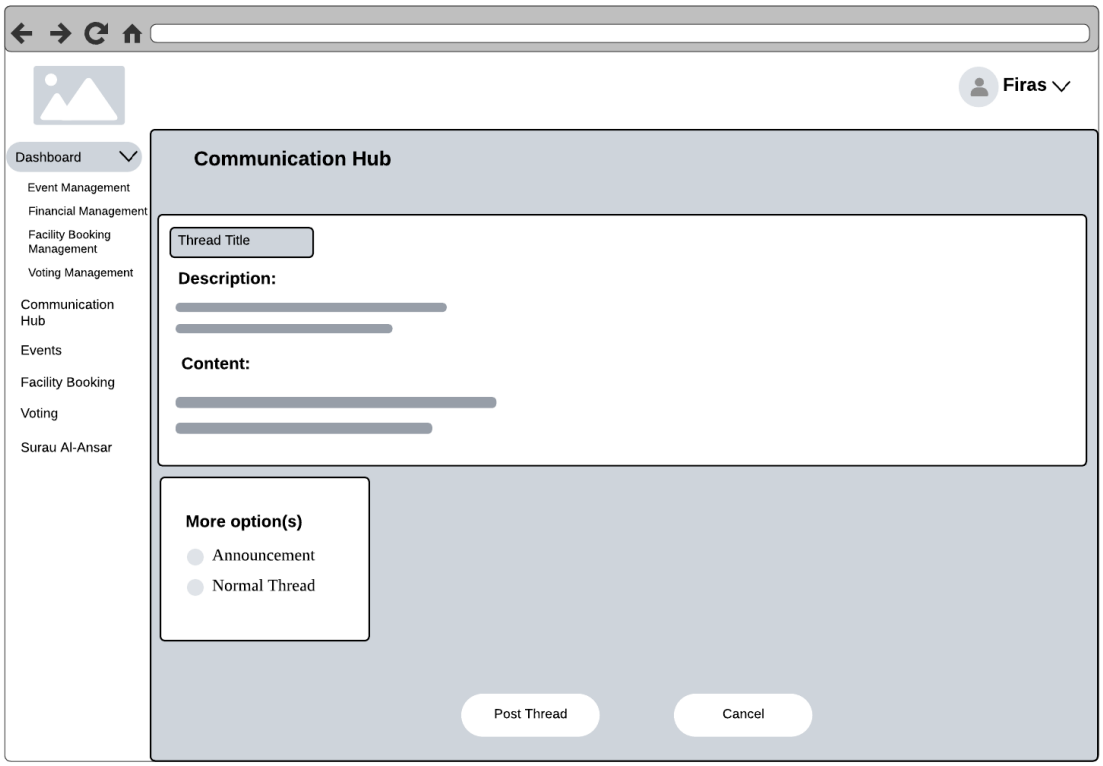


Figure 3.60: Announcement Option (Admin Access/Interface)

Figure 3.60 shows that an Admin, unlike normal users, can choose to make their thread into an announcement just by choosing it as an option. Once marked as an 'Announcement', the thread will be displayed as such, and can be seen within sections where announcements are supposed to be present, like in Figure 3.57, the first page of the Communication Hub, and Figure 3.39 and Figure 3.40, that are the dashboards for Admin and User, respectively.

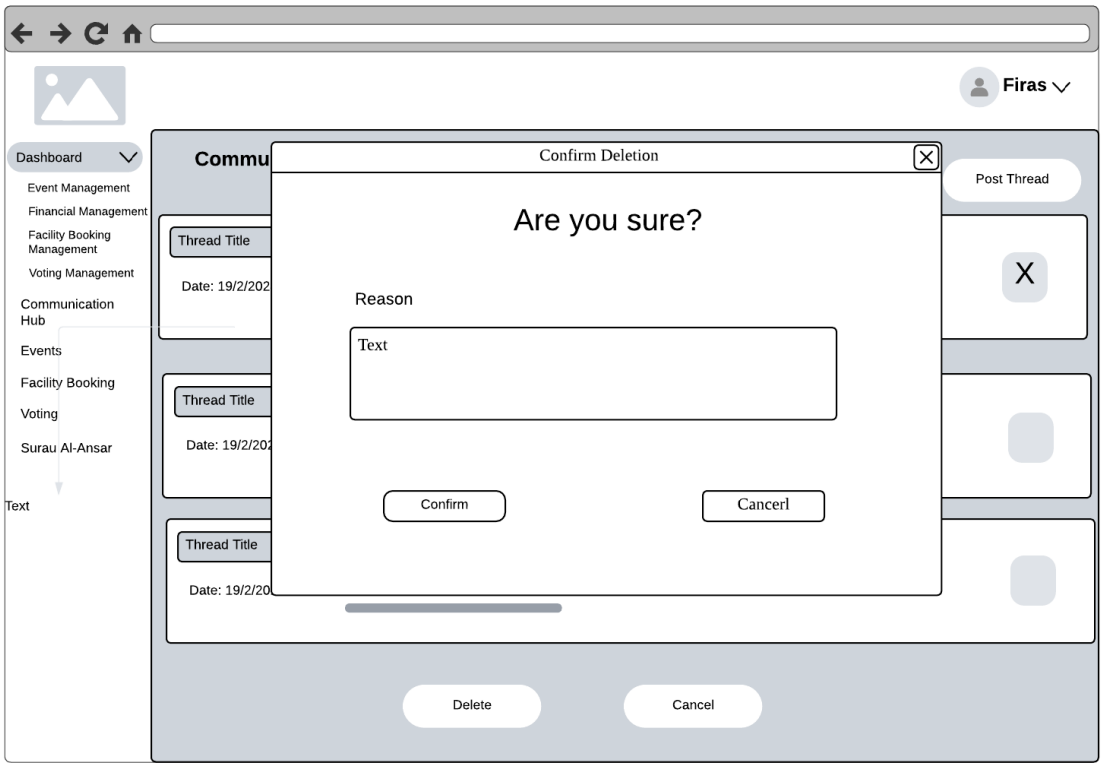


Figure 3.61: Confirm Deletion Form

But, before the deletion process in Figure 3.59 can truly complete, the Admin must first submit a reason as to why the action was taken and have that reason sent to whoever had their thread or comment deleted. Should the Admin go through with it, they will enter the reason as shown in Figure 3.61 and confirm the deletion process.

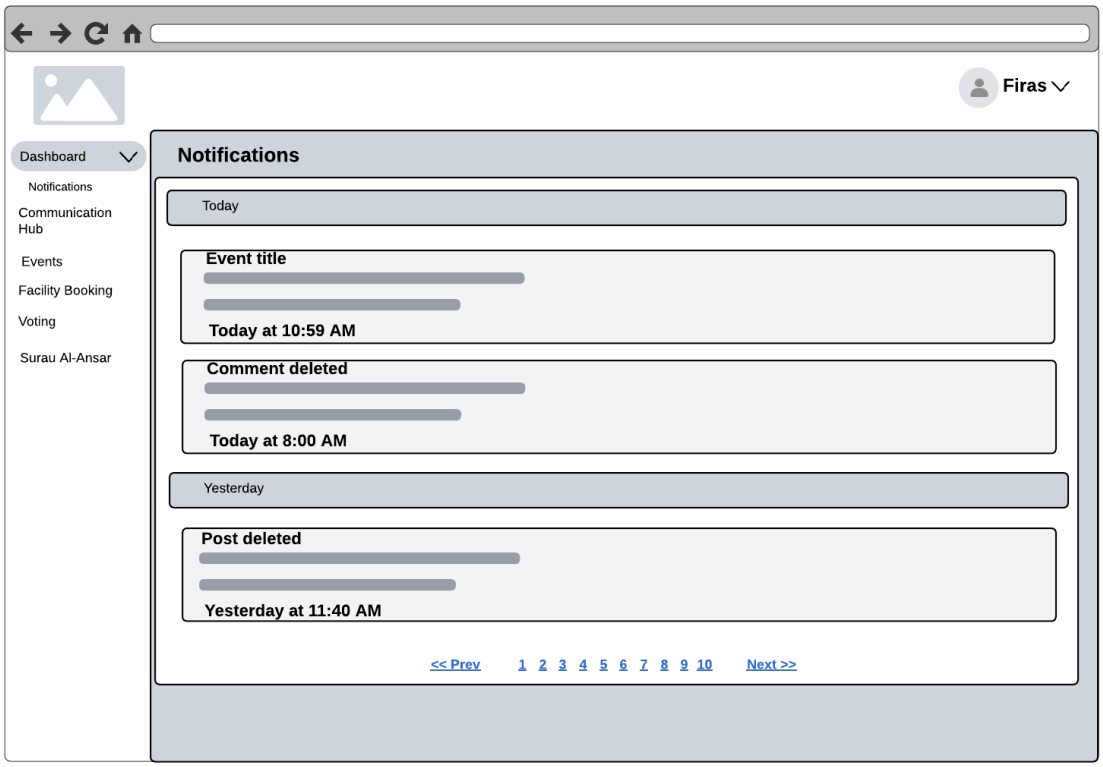


Figure 3.62: Notification Page

Figure 3.62 shows the Notification page which can be accessed by both normal users and users with the role of Admin through the use of their respective dashboards. In this page, they will receive notifications regarding matters like their comment or post being deleted, an up-and-coming-event, or their facility booking request being either approved or denied. The notifications will be sorted by the date in which they had been received. As can be seen from Figure 3.62, the notifications are divided based on what had been sent 'Today' and 'Yesterday'. Other than that, users can also see the specific time of when the notifications came in.

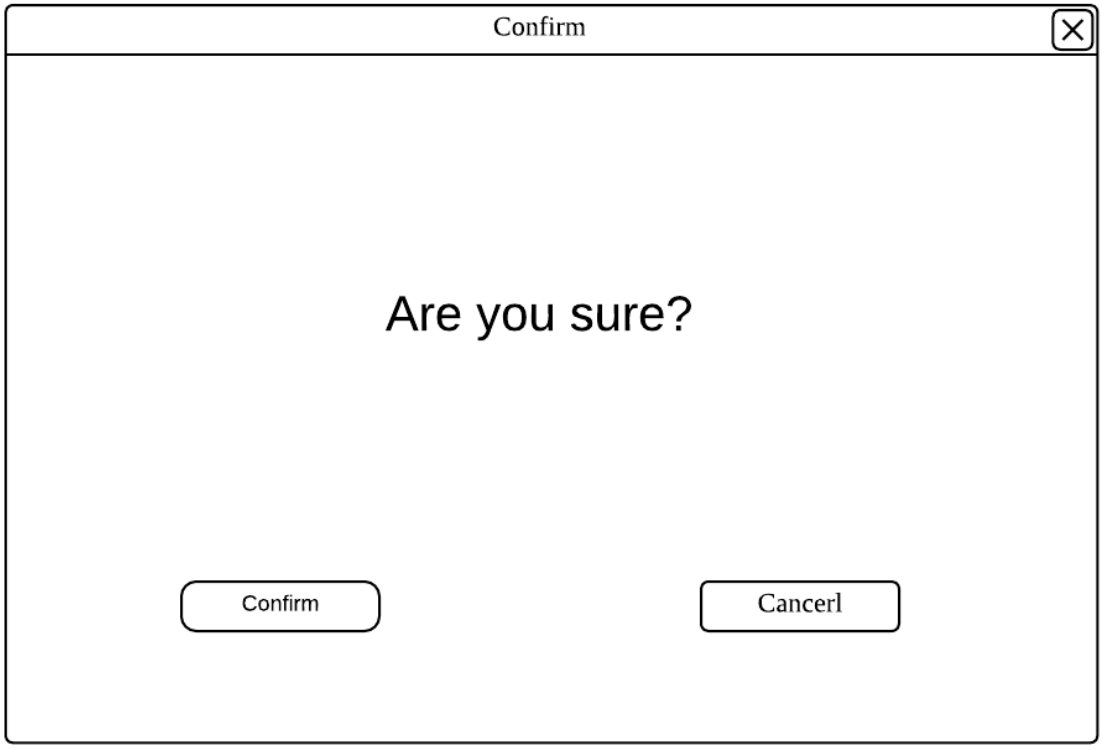


Figure 3.63: Confirmation Display

The Figure 3.63 displays a confirmation display that is expected to be seen every time the system prompts the user for confirmation after a specific action. It will be used and can be seen for most of the modules in the system.

3.2.3 Prototype Cycle

In this particular phase of the methodology, there is a need to adhere to an iterative process of developing, demonstrating, and refining the prototype of the proposed system, Smart Community Platform for Taman Suria Tropika. This cycle is repeated until all user requirements are thoroughly addressed and the prototype aligns with the stakeholders' expectations and needs.

3.2.3.1 **Develop**

Along the duration of this phase of the methodology, the functional prototype of the Smart Community Platform for Taman Suria Tropika is created. Learning from the feedback given during the designing phase, this phase puts a focus on using what has been learned from the refined design to create a functional system. Key features are developed in an iterative manner, with frequent communication and collaboration between developers and stakeholders to ensure the system meets user needs. Early testing is conducted to detect and resolve any issues promptly, ensuring each iteration improves functionality and user experience.

3.2.3.2 **Demonstrate**

In this stage of the methodology, the functional prototype which has been developed so far will be shown to the stakeholders in order to be evaluated and get some valuable feedback from them in turn. The demonstration of what has been developed will need to be something that rightfully meets the defined user requirements while figuring out areas that might need more fixing or improvement.

3.2.3.3 **Refine**

This phase involves analysing feedback and addressing any mistakes by making the necessary adjustments and enhancements to the system. Key aspects of the system, such as its design and functionality, will undergo modifications to resolve previously identified issues and improve overall performance, all while adhering to user requirements. Iterative updates are applied, followed by additional testing to ensure that the changes enhance the system’s usability, functionality, and effectiveness. This process continues until the prototype meets all expectations and agreement from stakeholders and is ready for the next phase.

3.2.4 Testing

In the Testing phase, the system will be examined comprehensively through multiple testing methods to confirm that it is not only fully functional but performs well within the expectation of it. The key types of testing that the system will undergo are system testing, unit testing, and usability testing. Moreover, user feedback will be collected to identify and resolve any issues, as well as to further enhance the system based on real-world usage.

3.2.5 Implementing

In this phase, the Implementation phase, the system is ready to be put into the deployment stage. It is at its final version where it will proceed to be set up for live use with real-world interaction and data in the production environment. Furthermore, this phase involves deploying the system on the chosen web hosting platform, ensuring that essential configurations, such as database integration and server setup, are properly implemented to ensure efficient operation. Then, the system will need to be monitored for any issues or necessary adjustments. Other than that, users will be provided with resources such as training sessions or user manuals to help them become familiar with the system's features and functionality.

3.3 Summary

Chapter 3 delves into the requirement analysis and design process for developing the Smart Community Platform for Taman Suria Tropika. The Rapid Application Development (RAD) methodology was selected for its iterative and user-centric nature, ensuring a structured yet flexible approach to system development. This chapter provides insights into the planning, analysis, and design phases, which form the foundation for the platform's development.

Initial requirements were gathered through interviews with key stakeholders and personal observations. These revealed inefficiencies in certain areas communication, financial management, and facility booking processes – to a degree. The proposed platform addresses these challenges by integrating key modules, including Event Management, Financial Management, Facility Booking, Voting, Notifications, and a Communication Hub. These features aim to ease operations and enhance community engagement.

The technical foundation combines Laravel, PHP, and MySQL for robust back-end functionality, complemented by front-end technologies like HTML, CSS, and JavaScript for an interactive user experience. Hardware requirements, such as a 16GB RAM system with a 64-bit OS, ensure smooth development and operation. Wireframes and UML diagrams were also utilized to visualize the system's structure and functionality, and it will be used as the foundation in which the system will be developed later on.

CHAPTER 4: IMPLEMENTATION

4.1 Introduction

This chapter delves into the processes involved in the development of the Smart Community Platform for Taman Suria Tropika. Building upon the insights and requirements gathered in Chapter 3, along with additional information obtained during development, this section highlights how the system was implemented. It covers the setup of the development environment, the tools and frameworks used, and the implementation of each module that supports the platform's functionality.

4.2 Installation and Configuration of System's Components

To begin the development of the proposed application, several software tools were required to be installed and configured in advance.

4.2.1 Laragon and PHP

Laragon was used as the local development environment due to both its familiarity and its ability to manage PHP, MySQL, and Apache. It provides an isolated and lightweight development environment, allowing for easier understanding and seamless execution of Laravel-based applications. During development, the system was run using PHP's built-in development server, which proved sufficient for testing, debugging, and verifying system functionality in a local environment.

To set up the environment, Laragon was downloaded from its official website (https://laragon.org/download). Figure 4.1 shows the download page for Laragon.

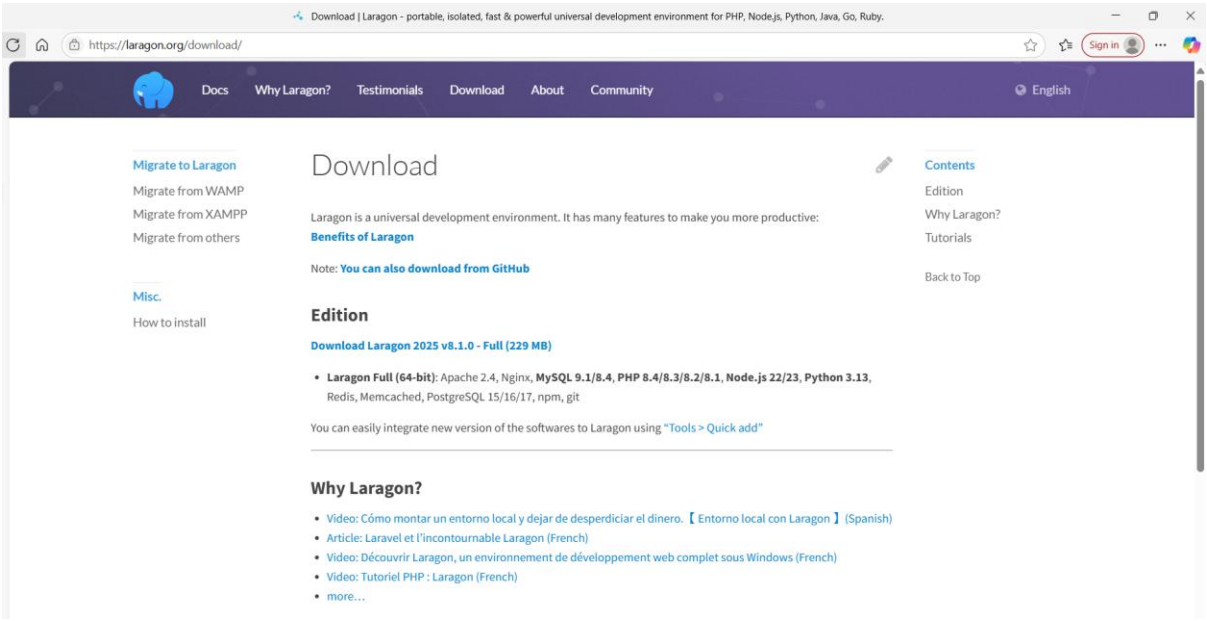


Figure 4.1: Laragon website

Once installation was complete, the Laragon control panel became accessible, as illustrated in Figure 4.2.

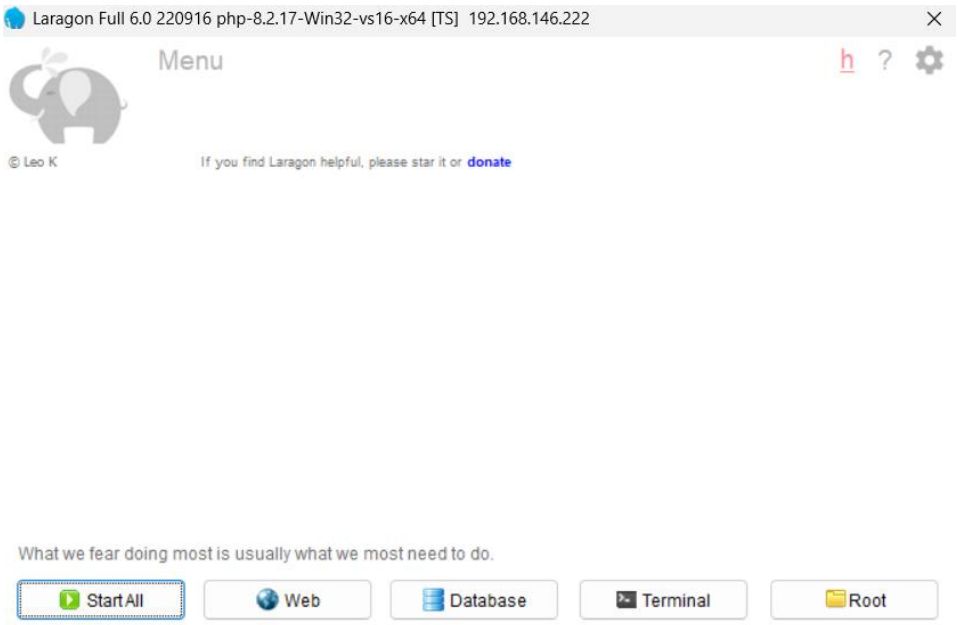


Figure 4.2: Laragon control panel

Afterwards, one of the latest PHP versions compatible with Laravel version 12 (the version used in this project) was downloaded from https://www.php.net/downloads.php, as shown in Figure 4.3.

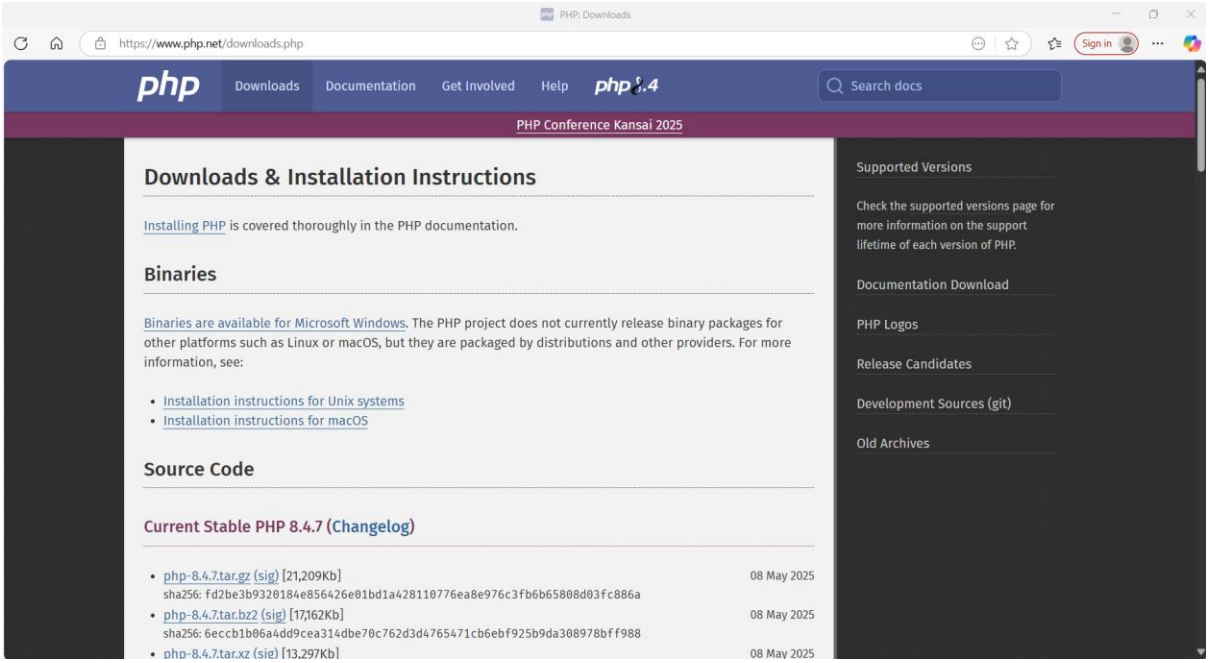


Figure 4.3: PHP website

The downloaded PHP directory was then added to C:\laragon\bin\php to allow Laragon to detect and utilize the correct PHP version, as displayed in Figure 4.4.

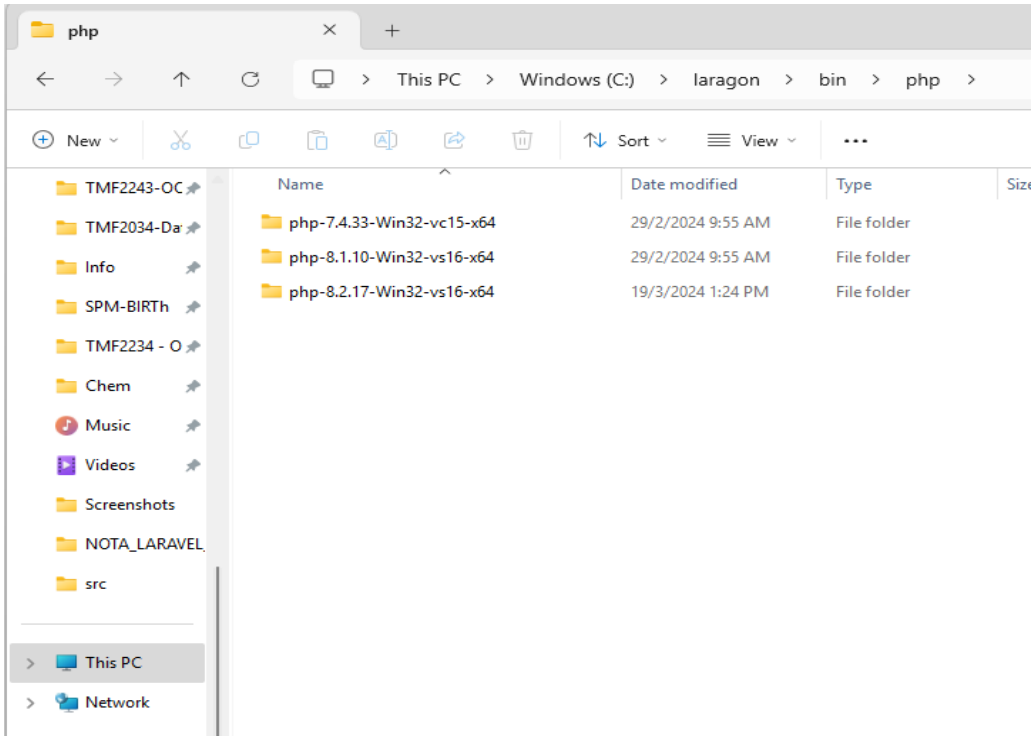


Figure 4.4: PHP version added to C:\laragon\bin\php directory

This configuration ensured that the PHP version used was compatible with Laravel 12 and could be properly selected and accessed within the Laragon environment, as shown in Figure 4.5.

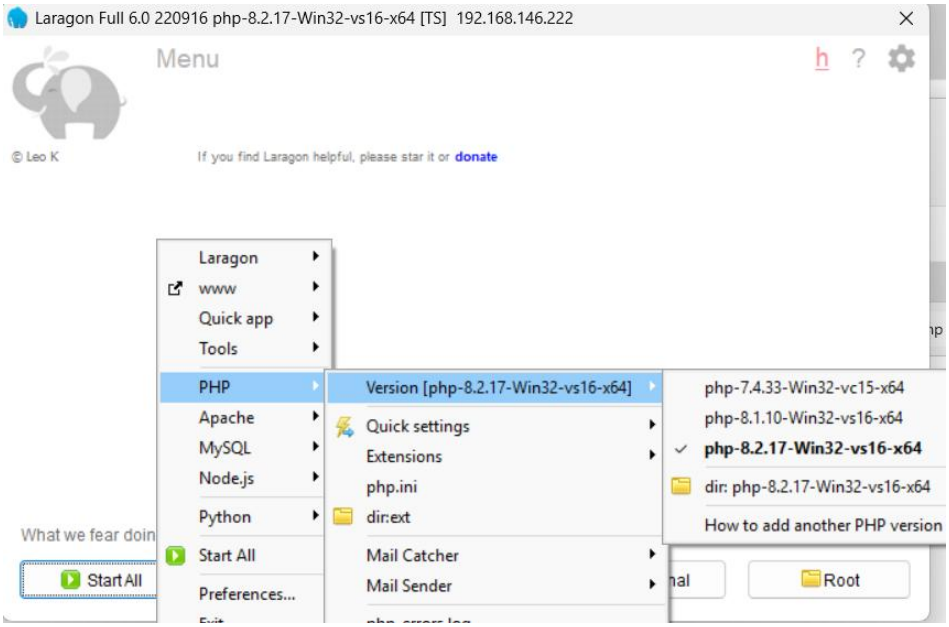


Figure 4.5: Selecting the desired PHP version from the Laragon PHP version menu.

4.2.2 Laravel PHP Framework

4.2.2.1 Installation and Configuration of Composer

Before Laravel could be installed, it is imperative that the local machine has Composer installed, which is a dependency manager utilized by Laravel. This is a simple and straightforward matter which even those new to the framework can be guided into doing, as it has already been instructed in Laravel's readily available documentation that is accessible online. The official Composer website, where the installer and documentation can be accessed, is shown in Figure 4.6.

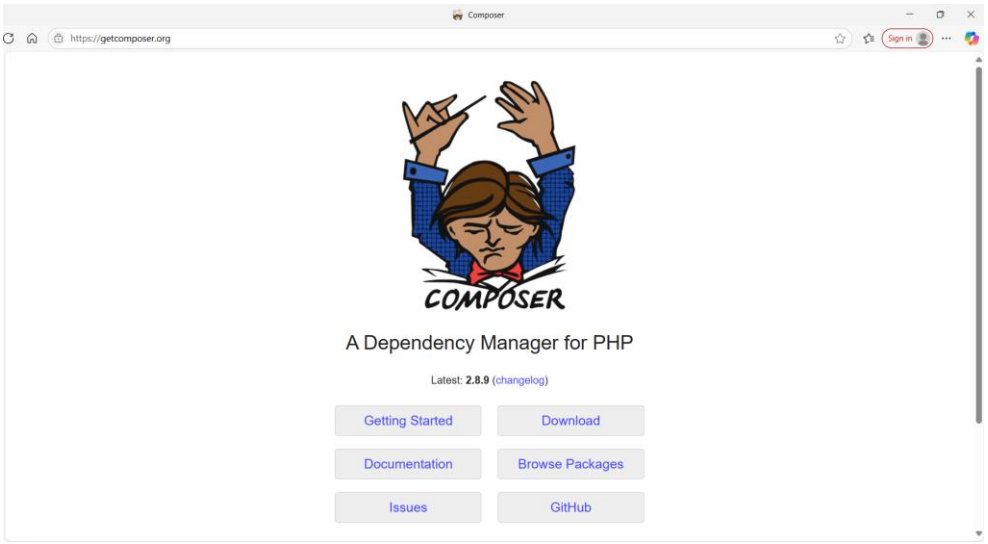


Figure 4.6: Composer official website used to download and install the dependency manager for PHP.

After downloading and installing Composer, its successful installation was verified by executing the “composer” command in the system's command prompt, as shown in Figure 4.7.

Figure 4.7: Command prompt displaying Composer version and available commands after successful installation.

With Composer properly installed and with a compatible PHP version configured, the Laravel framework could be downloaded and executed without issue.

4.2.2.2 **Installation of Laravel and Project Creation**

Laravel was installed using Laragon's built-in "Quick App" feature, which simplifies the process of setting up Laravel projects. After selecting "Start All" to initialize required services, the Laravel installer was accessed via Menu > Quick app > Laravel on the Laragon control panel. From there, a new Laravel project named "SCTropikaNew" was generated. This process automatically downloaded the latest compatible version of Laravel (version 12 at the time of development) and created the project directory within Laragon's default web root (C:\laragon\www\SCTropikaNew). The Figures 4.8 and 4.9 illustrates the aforementioned process.

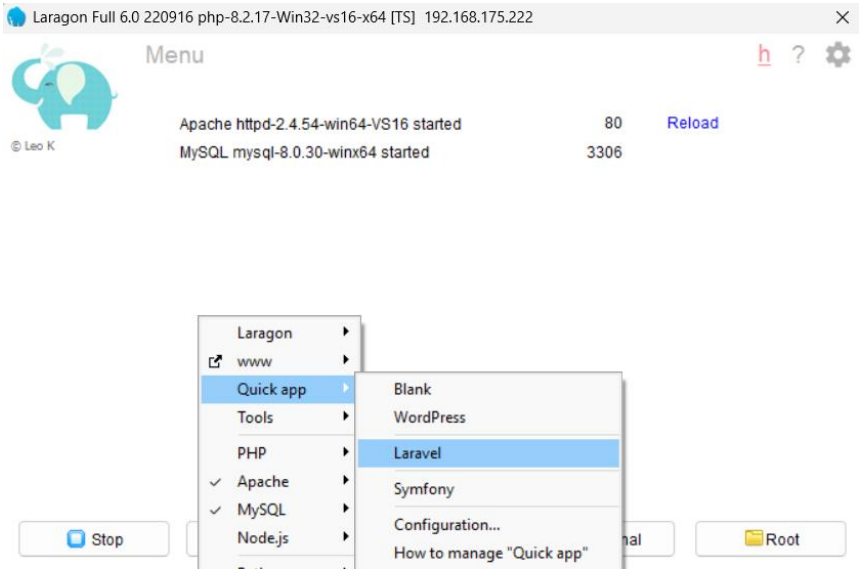


Figure 4.8: Accessing the Laravel installer via Laragon's Quick App menu.

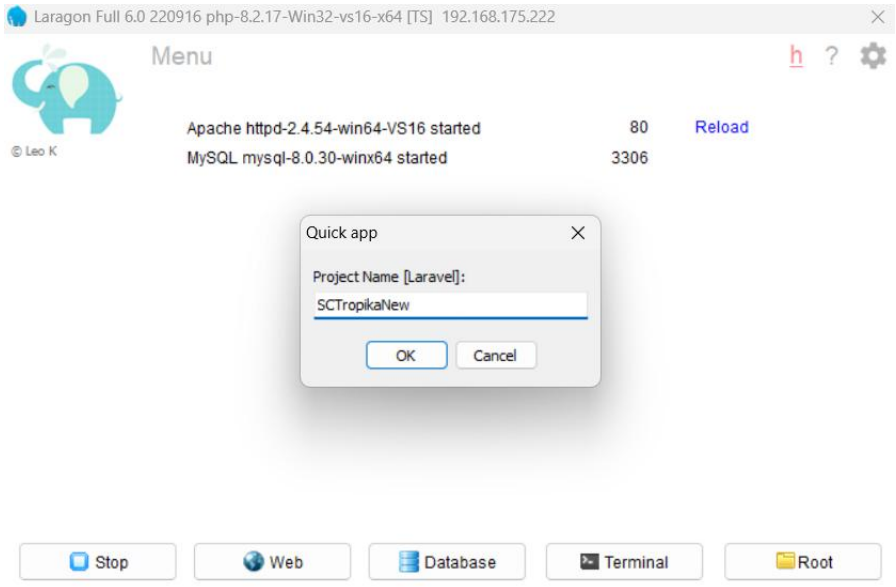


Figure 4.9: Creating a new Laravel project named "SCTropikaNew" using Laragon.

Once the Laravel application was successfully created, it was then prepared for frontend asset compilation. This step was necessary to enable proper styling and interactivity, as the system relied on Laravel Breeze for its authentication scaffolding, along with Tailwind CSS and Alpine.js for frontend styling and interactivity. All three were utilized throughout the system's local development.

To achieve this, Node.js was installed on the local machine to support the use of npm (Node Package Manager), which handles frontend dependencies. Inside the Laragon's terminal, the command "npm install" was executed, as can be seen in Figure 4.10.

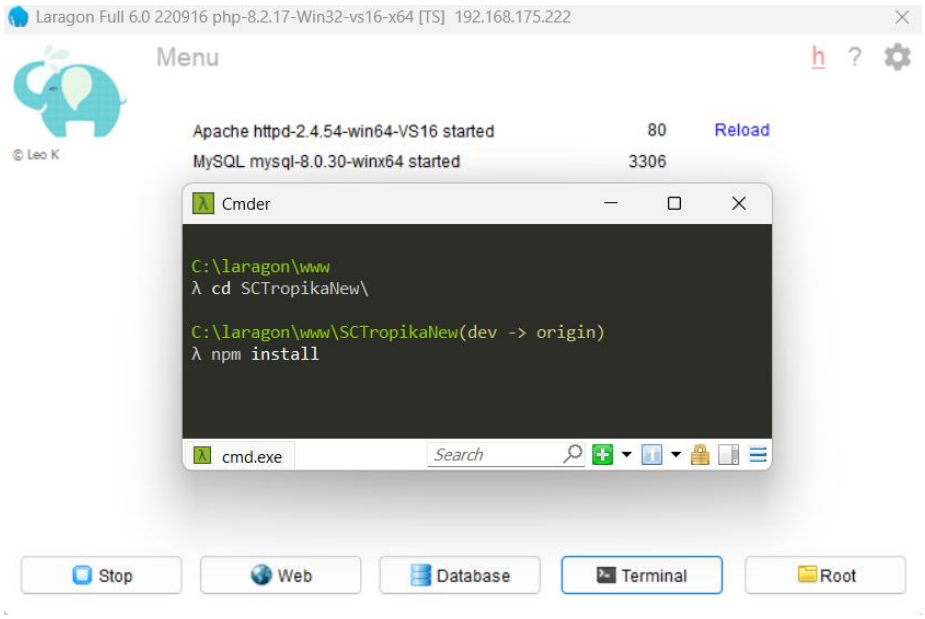


Figure 4.10: Executing npm install within the project directory to install frontend dependencies.

This process installed all the necessary frontend dependencies as defined in the package.json file. Once the installation was complete, the command npm run build was executed to compile the frontend assets for use in the local development environment. This step ensured that Tailwind CSS, Alpine.js, FullCalendar, and other JavaScript files, including custom scripts, were compiled and bundled using Laravel's Vite build tool. These assets enabled the system's dynamic interface and interactivity during development and testing. The process is shown in Figure 4.11.

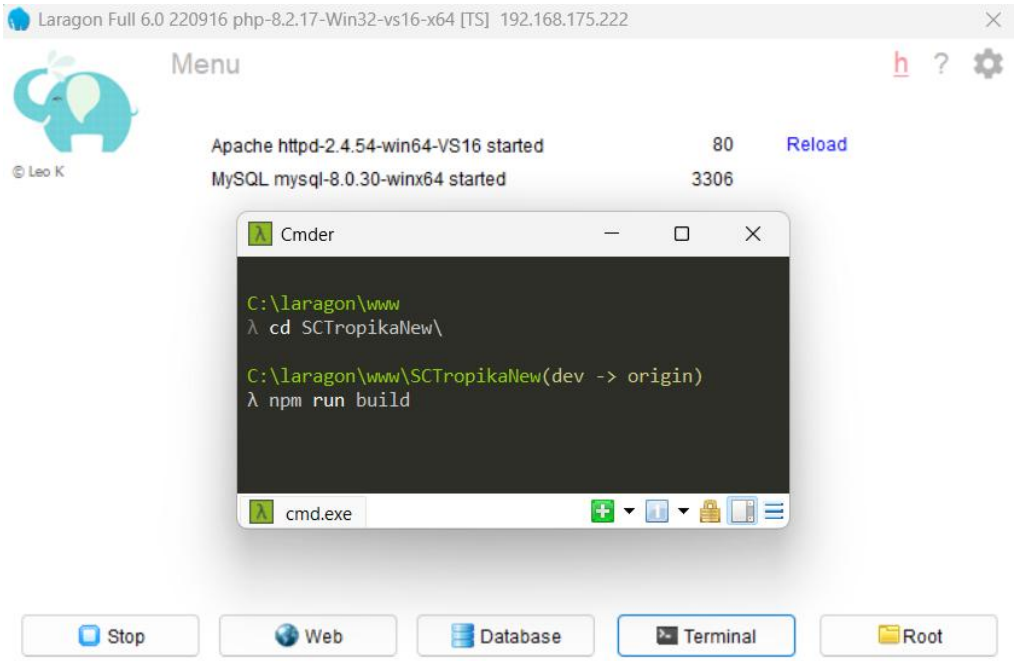


Figure 4.11: Executing npm run build to compile frontend assets within the Laravel project directory.

Alternatively, during development, the command npm run dev could be used instead of npm run build to enable hot module reloading, allowing for faster iteration of frontend changes. This approach automatically reflects updates in the browser without requiring a full rebuild, improving development efficiency. The output of this command is shown in Figure 4.12.

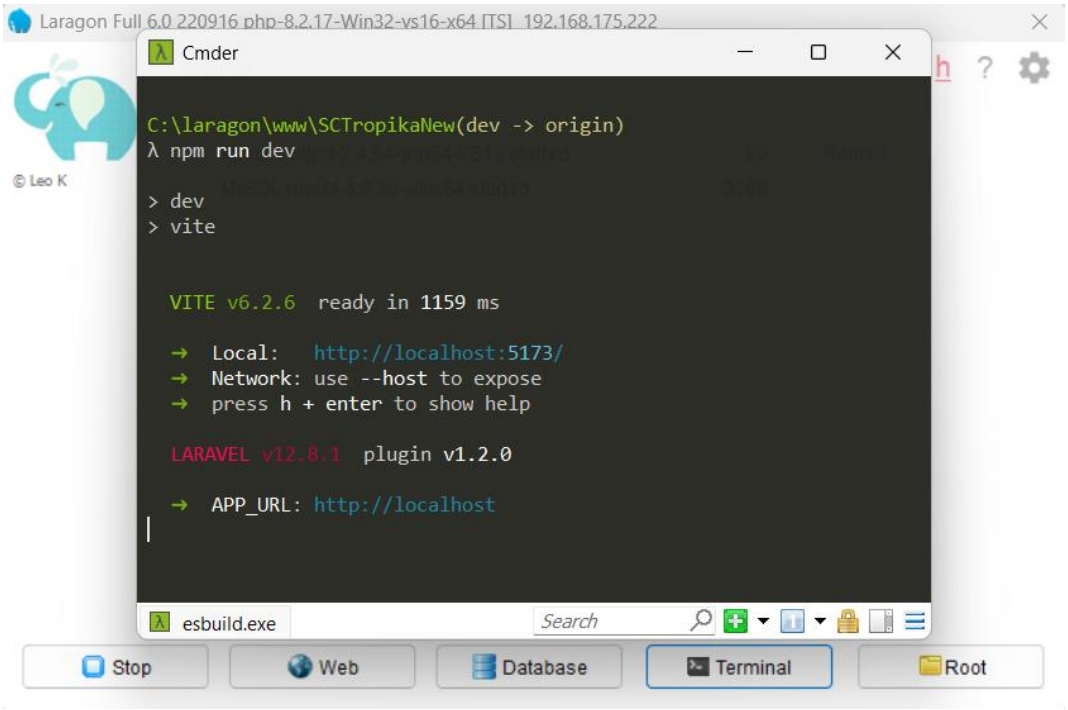


Figure 4.12: Running npm run dev to enable hot-reloading and live updates during frontend development.

With the application and its assets compiled, the Laravel project was ready to be served locally. This was done by running the command php artisan serve in the terminal, which starts Laravel's built-in development server. As shown in Figure 4.13, the server runs on http://127.0.0.1:8000, making the project accessible via a web browser for testing and further development.

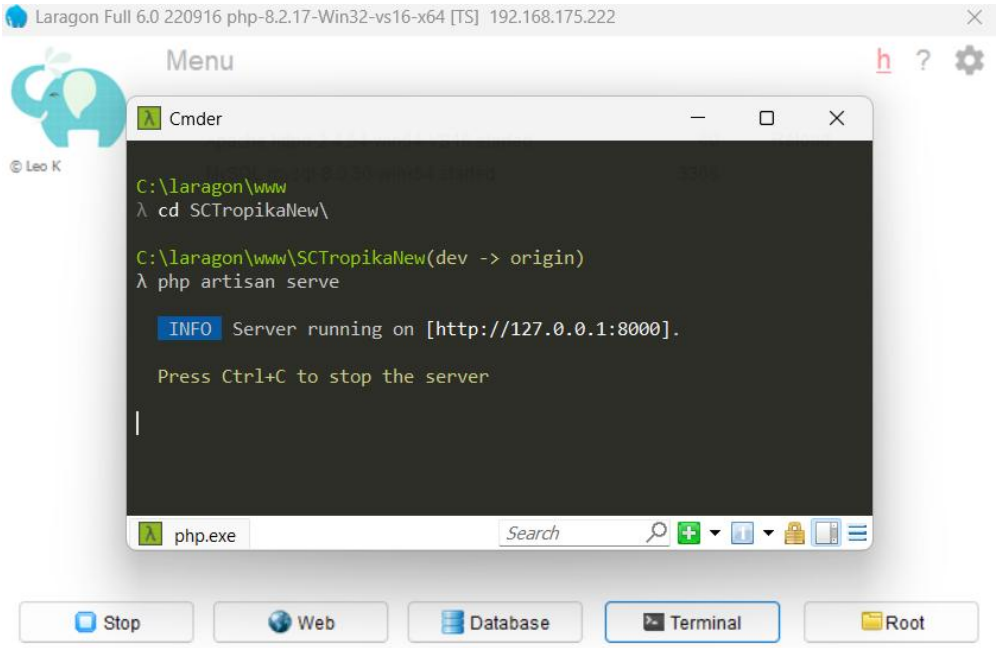


Figure 4.13: Running php artisan serve to launch the Laravel development server locally.

4.2.2.3 **Configuring Database Connection between Laravel and MySQL**

Following the creation of the Laravel project, it was essential to configure the database connection to enable data storage, retrieval, and interaction with the system. For this project, MySQL was selected as the database management system due to its compatibility with Laravel, ease of integration, and support through Laragon.

Laravel manages its database settings through a dedicated environment file named .env, which is located in the root directory of the project. This file stores sensitive configuration variables such as database credentials, application keys, and other environment-specific settings.

To establish the database connection, the .env file was modified with the following settings shown in Figure 4.14.

Figure 4.14: Database configuration for MySQL defined in the .env environment file.

The database name “newsmart” was defined and created through phpMyAdmin, which is included by default in the Laragon environment. By accessing http://localhost/phpmyadmin, a new database was created to match the name specified in the .env file. The default Laragon MySQL credentials were used, where the username was set as root and no password was required.

Once the .env file was configured and the corresponding database was created, Laravel’s built-in migration feature was utilized to generate the default tables required for user authentication and session handling. This was done by executing the command php artisan migrate in the terminal, as shown in Figure 4.15.

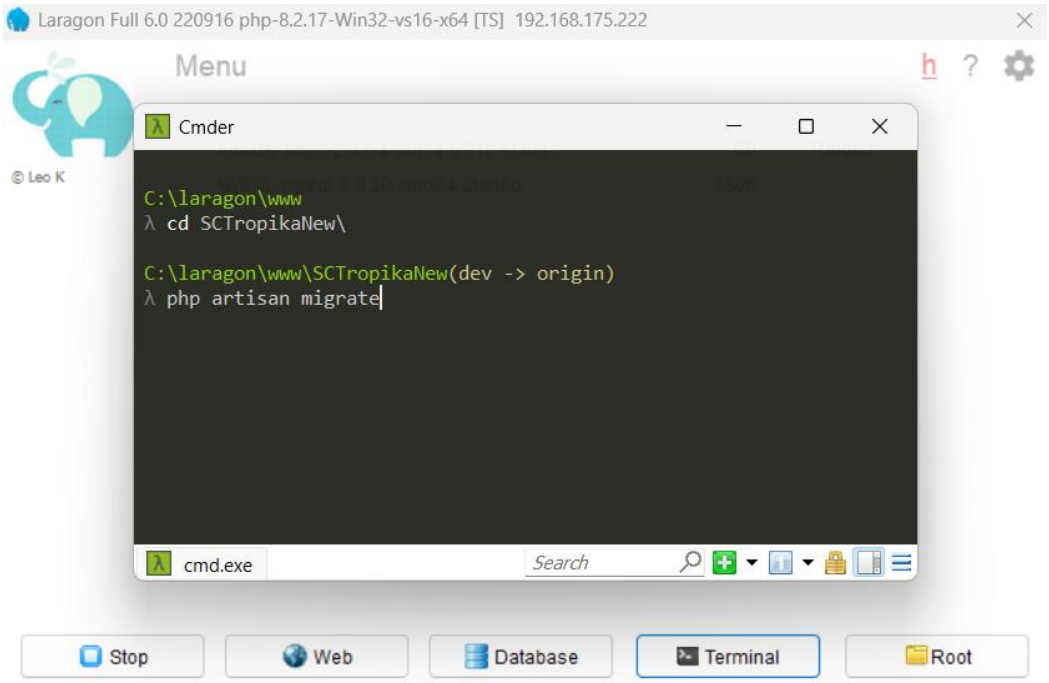


Figure 4.15: Running php artisan migrate to create default Laravel tables in the connected MySQL database.

This command executed the migration files located in the database/migrations directory and created the relevant tables in the MySQL database. Figure 4.16 displays the successful execution of the migration process and the resulting database tables.

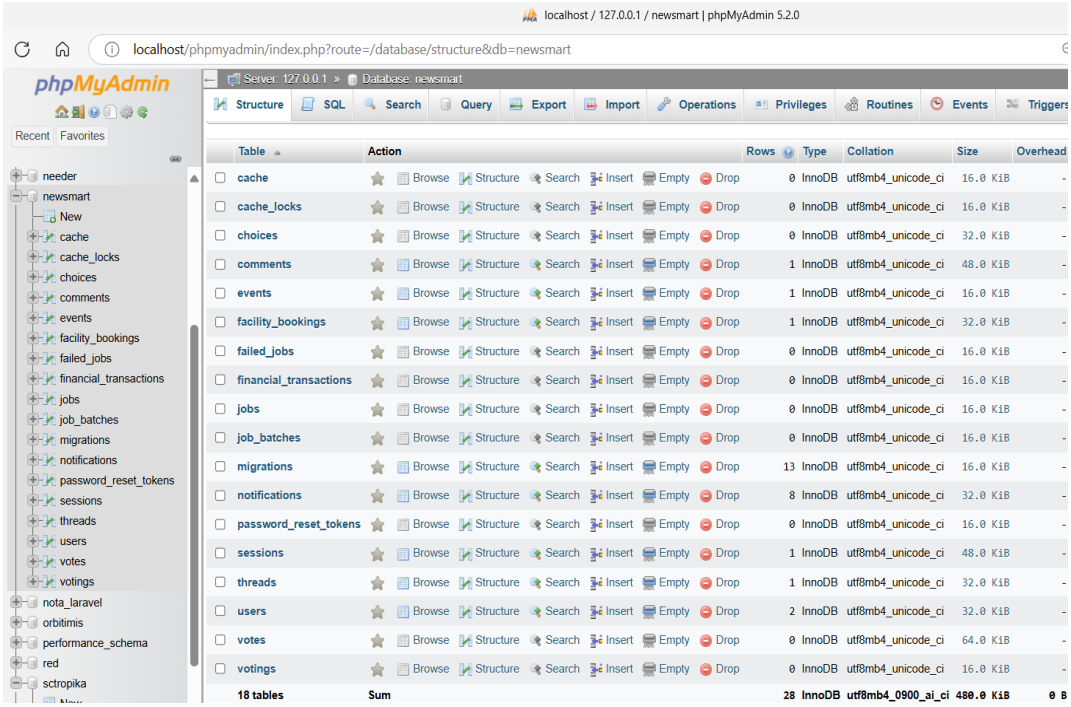


Figure 4.16: Database tables created in MySQL after executing Laravel migration files.

With the database now connected and initialized, Laravel's Eloquent ORM was able to interact seamlessly with the database. This enabled efficient execution of Create, Read, Update, and Delete (CRUD) operations across the application. Most of the logic defined in the system's controllers follows this CRUD structure, facilitating the handling of data-related operations across all implemented modules.

4.3 Introduction to the Role Based Access

The Smart Community Platform for Taman Suria Tropika was designed to support role-based access control to ensure that users have access only to the features and resources relevant to their assigned roles. The system distinguishes between two primary user roles: Admin and User.

Each role has specific permissions and responsibilities. Admin users have access to management-related functionalities such as managing events, managing facility booking requests, handling financial records, setting up voting sessions, moderating communication threads, and overseeing overall platform administration. Standard users, on the other hand, are limited to community-level interactions such as viewing events, submitting facility booking

requests, participating in voting sessions, and posting or commenting in the communication hub.

Role-based access was implemented through the following mechanisms:

- **Database Role Assignment**: Upon registration, each user is assigned a role which is stored in the users table under a role column. Admin accounts were seeded manually during the initial system setup to prevent unauthorized access to administrative features.
- **Middleware Protection**: Laravel's middleware feature was used to restrict route access based on user roles. Custom middleware was created to check the role of the authenticated user and ensure that only authorized users can access certain routes. For example, routes prefixed with /admin are only accessible to users with the Admin role.
- **Blade Directives and View-Level Restriction**: In addition to route protection, Laravel Blade directives such as @can, @if, and @auth were used to control the visibility of UI elements based on the authenticated user's role. This ensures that users do not see interface options that they do not have permission to use.

This role-based system enhances the platform's security and usability by providing a clear separation of access rights, preventing unauthorized actions, and streamlining the user experience based on role relevance.

4.4 Workflow for Smart Community Platform for Taman Suria Tropika system

The Smart Community Platform for Taman Suria Tropika was developed with a structured workflow to support ease of use and efficient access to the system's main features. The workflow describes how both Admin and User roles interact with the system from login to the use of its various modules. Role-based redirection ensures that users are presented with an interface that matches their access level and responsibilities.

Upon visiting the platform, users are greeted with the login interface, where authentication is required to access any features. Depending on the user's assigned role, either it be Admin or User, they are redirected to their respective dashboards upon successful login.

Admins are presented with a dashboard that displays shortcuts to various management modules. The typical workflow includes:

- **Dashboard Overview**: View summary statistics or announcements.

- **Event Management**: Create, edit, or delete community events, which will be visible to all users.
- **Facility Booking Management**: View and manage booking requests submitted by users, including the ability to approve or reject them.
- **Financial Management**: Add, edit, or remove financial records, and view a categorized transaction history.
- **Voting Management**: Create new voting sessions, define options, and monitor participation results.
- **Communication Hub Moderation**: Monitor threads and comments, and take action when necessary, through editing or deletion, including issuing moderation notifications.
- **Notification**: Receive alerts regarding new booking requests, voting activity, and system interactions that require attention.

Standard users are directed to a user-specific dashboard with access to community-level functionalities:

- **View Events**: Browse a list of current and upcoming events created by Admins.
- **Submit Facility Booking**: Fill out booking request forms for community facilities and monitor request status.
- **Participate in Voting**: Engage in active voting sessions and view voting results (if enabled).
- **Communication Hub**: Create and participate in forum-style threads and discussions.
- **Notification**: Receive updates regarding events, facility bookings, voting sessions, and moderation outcomes.

The system's workflow is guided by role-checking mechanisms embedded both in route middleware and within the Blade views. Navigation is dynamically adjusted based on the authenticated user's role, ensuring that users are only presented with actions relevant to their permissions.

All in all, this workflow ensures a logical and secure flow of interaction throughout the system. It supports modular use, encourages user participation, and enforces administrative

control where necessary, aligning with the community's needs and the project's core objectives.

4.5 Modules for Smart Community Platform for Taman Suria Tropika system

The Smart Community Platform was developed with several key modules that work together to serve both administrative and community needs. Each module was designed with specific functions and user roles in mind, ensuring clarity of responsibility and ease of use. The modules implemented in the system include event management, facility booking, financial tracking, voting, communication hub, user management, user profile handling, and a notification system.

4.5.1 Registration, Login, and Home Page

Before accessing the event features, users must first register or log in through the system's authentication interface, provided by Laravel Breeze. Figure 4.17 shows the registration form, which requires users to input their name, email, password, and password confirmation.

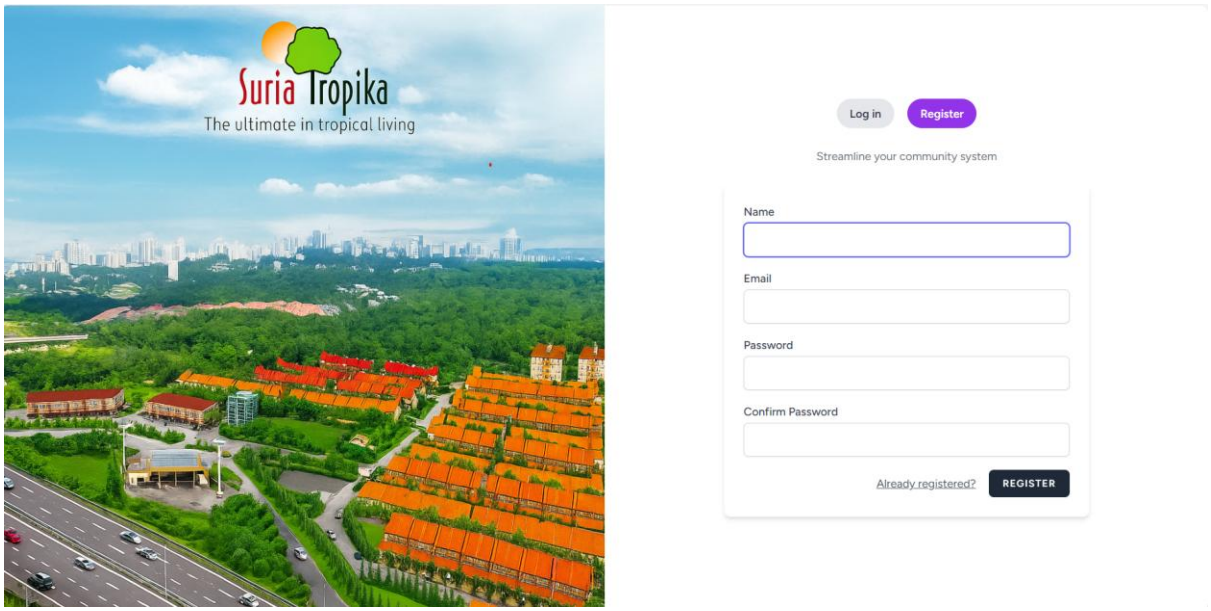


Figure 4.17: User registration form

Once an account has been created, either through the registration form or by being seeded manually in the case of admin users, individuals can proceed to the login interface. Figure 4.18 displays the login form, where users are required to enter their registered email address and password to gain access to the platform.

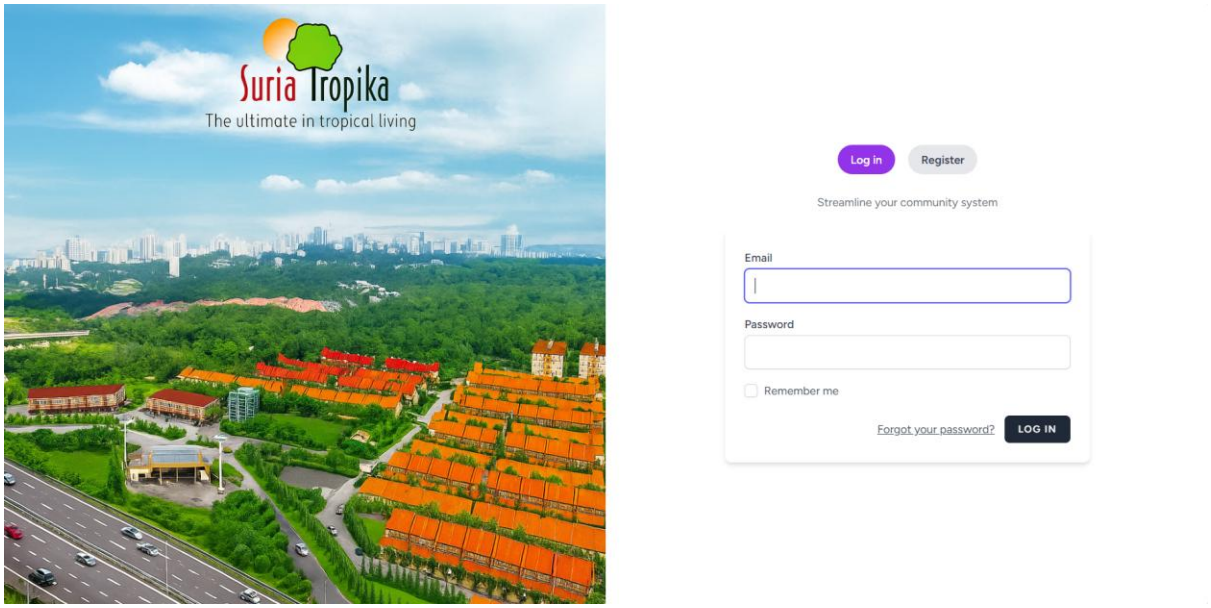


Figure 4.18: User login form

Upon successful login, users are directed to the Home Page, which serves as a central landing area introducing the platform and its purpose. The welcome banner, as shown in Figure 4.19, highlights the system's goal of connecting the residents of Taman Suria Tropika, with visuals referencing Surau Al-Ansar and nearby landmarks.

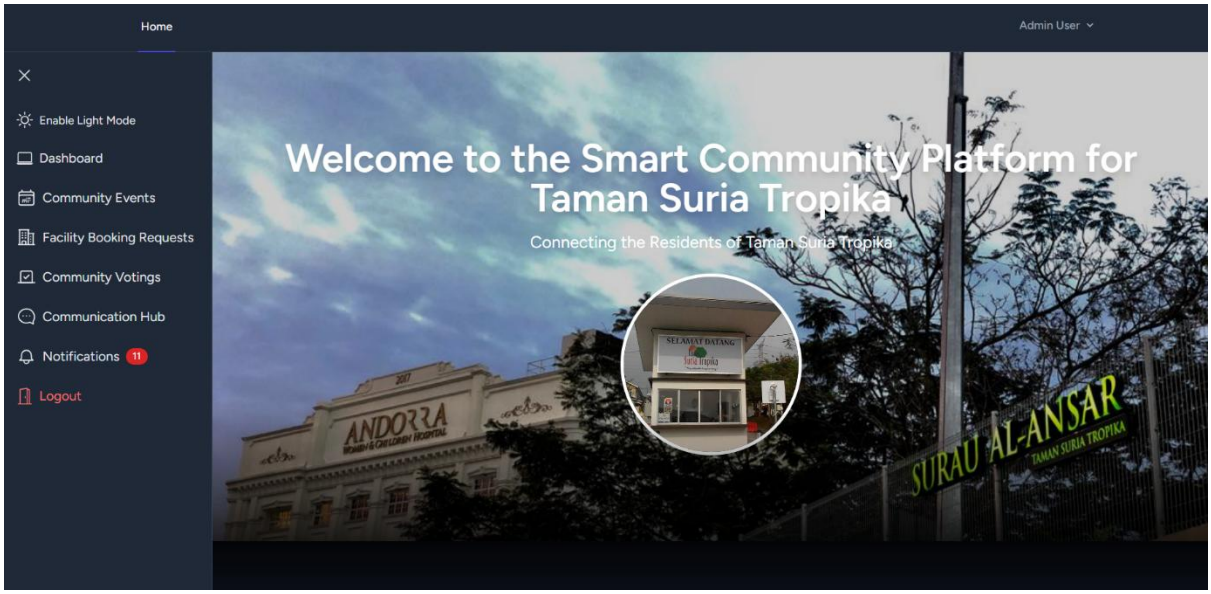


Figure 4.19: Home page

Scrolling down, the system provides an "About Taman Suria Tropika" section, which introduces the community's context. As seen in Figure 4.20, it briefly explains the location, community characteristics, and provides a collage of related photos from past events.

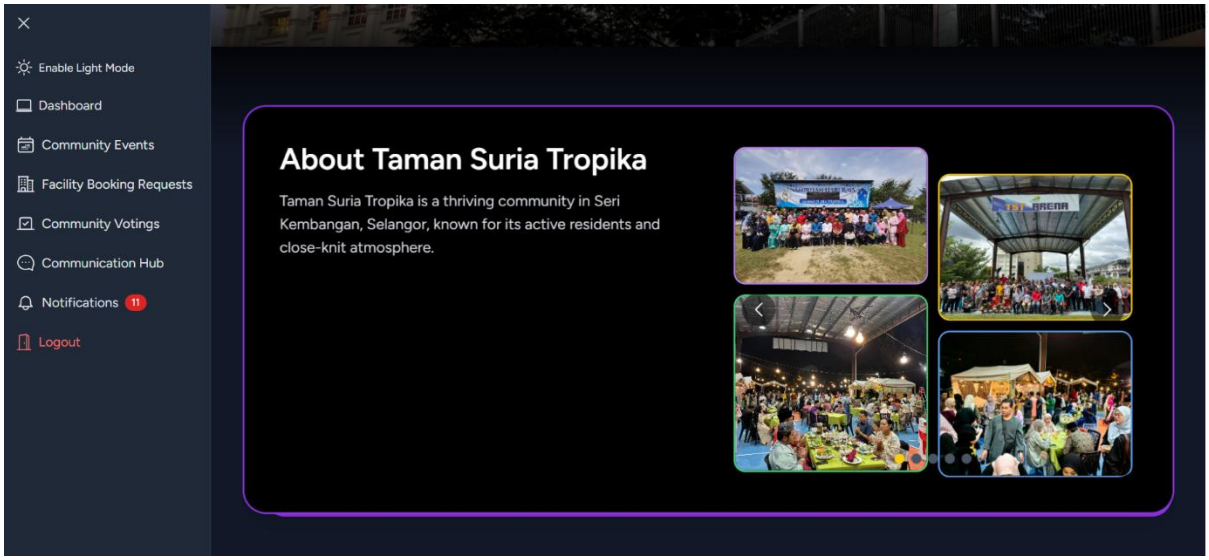


Figure 4.20: About section describing the Taman Suria Tropika community with event highlights.

Following that, the Platform Highlights section summarizes the primary modules of the system. Illustrated with icons and labels, this section introduces users to Event Management, Facility Booking, Financial Records, Voting, and the Communication Hub. This layout is shown in Figure 4.21.

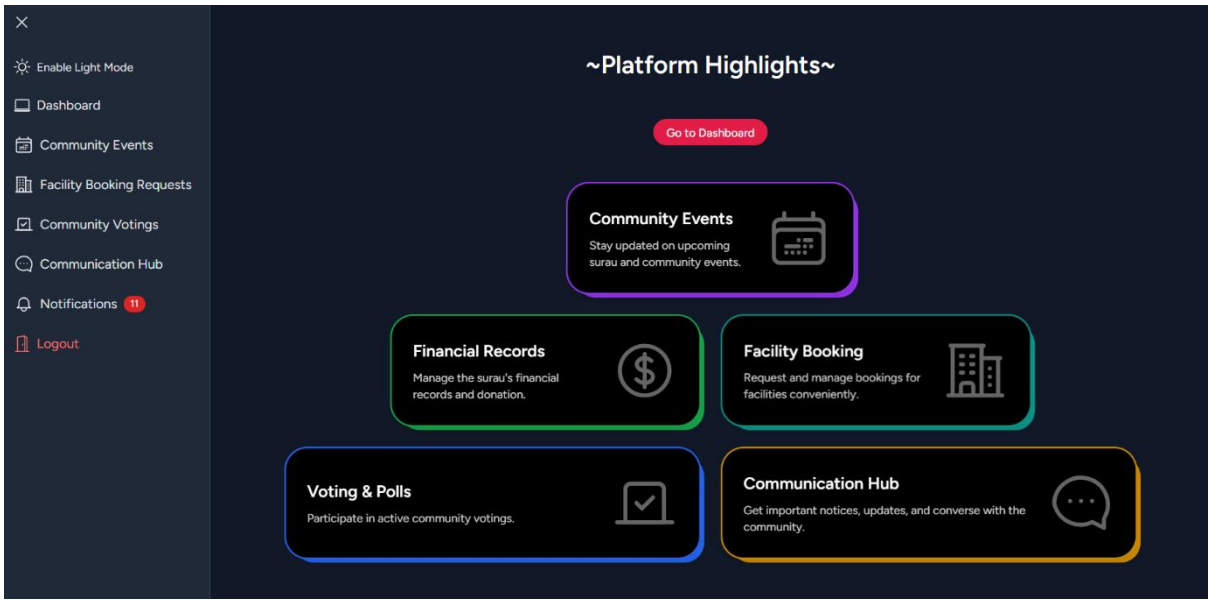


Figure 4.21: Platform Highlights section outlining key community modules and functions.

In addition to community-level information, the Home Page also features details about Surau Al-Ansar, the community prayer hall. Figure 4.22 presents a short description alongside photos of community gatherings and prayer sessions.

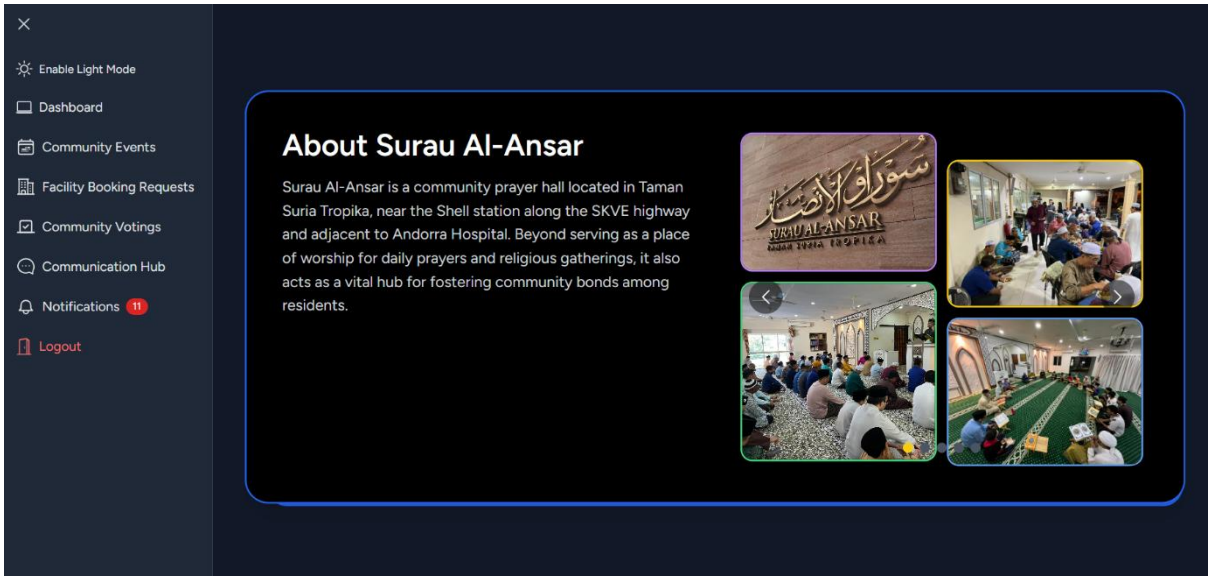


Figure 4.22: Overview of Surau Al-Ansar

Lastly, the system displays a visual organizational structure of the surau's committee for the 2024–2026 term. The hierarchy includes the chairman, vice-chairman, imams, bilals, siaks, treasurer, youth reps, and others. This clear and accessible chart is displayed in Figure 4.23 and Figure 4.24.

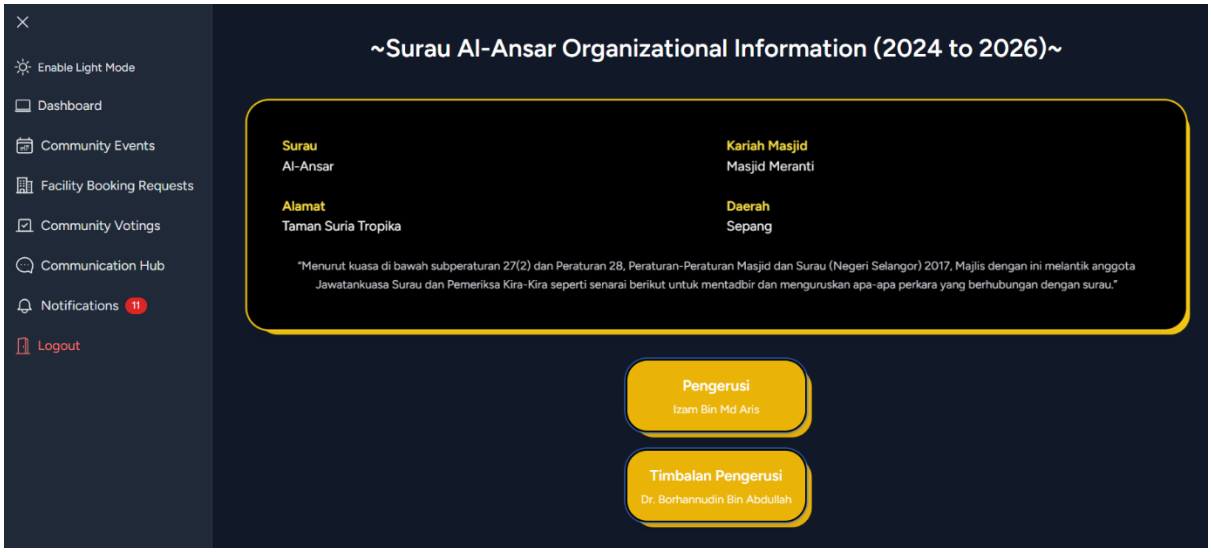


Figure 4.23: Organizational structure of Surau Al-Ansar (Part 1)

Figure 4.24: Organizational structure of Surau Al-Ansar (Part 2)

4.5.2 Event Management Module

The Event Management Module was developed to empower Admin users to efficiently organize, manage, and broadcast events relevant to the Taman Suria Tropika community. Admins can create new events, modify event details, or remove outdated entries. Once published, these events become publicly visible to all logged-in users through the Community Events section.

This module is essential for keeping residents informed about surau activities, community programs, and upcoming initiatives. It includes specific interfaces tailored to the responsibilities of each user role, with admins given full control over content management, while normal users have read-only access to event information.

4.5.2.1 Admin View and Event Access

Upon successful login, Admin users are directed to the Admin Dashboard, as shown in Figure 4.25, which acts as a central hub for system management. The dashboard presents real-time summaries such as the total number of events, facility bookings, and votings for the current month, as well as notification alerts.

It also provides access cards for all primary modules, such as Event Management, Facility Booking, Financial Records, Votings, Communication Hub, User Management, and Notifications. These cards are color-coded for clarity and labeled with role-based access (e.g., Public View, Admin Tools) to help admins quickly navigate to the appropriate sections.

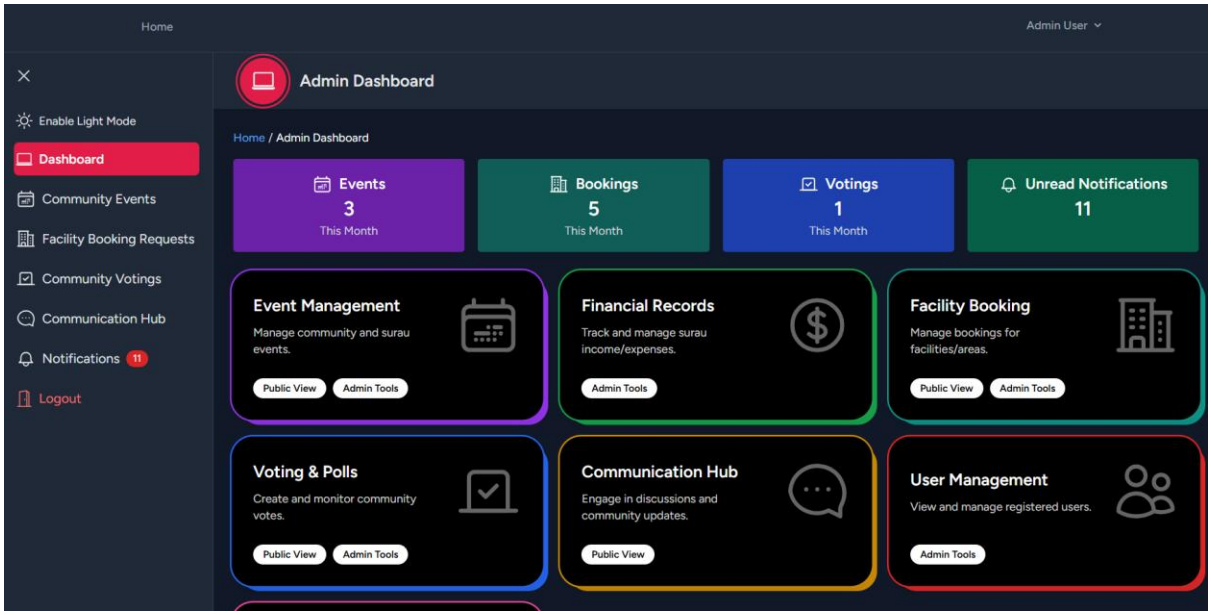


Figure 4.25: Admin Dashboard

From the Admin Dashboard, Admin users can navigate to the Event Management section. This page displays a tabulated list of all existing events, each accompanied by key information such as the event name, brief description, scheduled date and time, and an optional image. The interface also includes action icons for editing or deleting events.

This structured overview allows administrators to efficiently manage upcoming or past events, ensuring accuracy and up-to-date information. The layout and elements of the event list are illustrated in Figure 4.26, which shows the admin view of event records along with their respective controls.

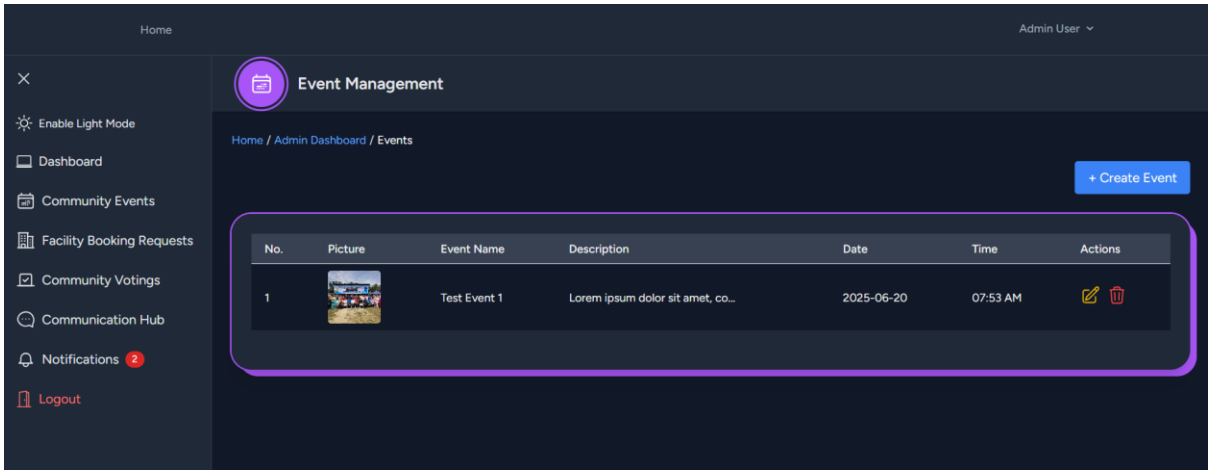


Figure 4.26: Event Management list view showing existing events and admin action controls

4.5.2.2 **Creating a New Event Form**

By clicking the "Create Event" button, Admins are directed to the event creation form. As shown in Figure 4.27, this form includes fields for entering the event name, description, date, time, location, and an optional image upload. Once submitted, the information is stored in the database and later displayed to users through the public dashboard and community events page.

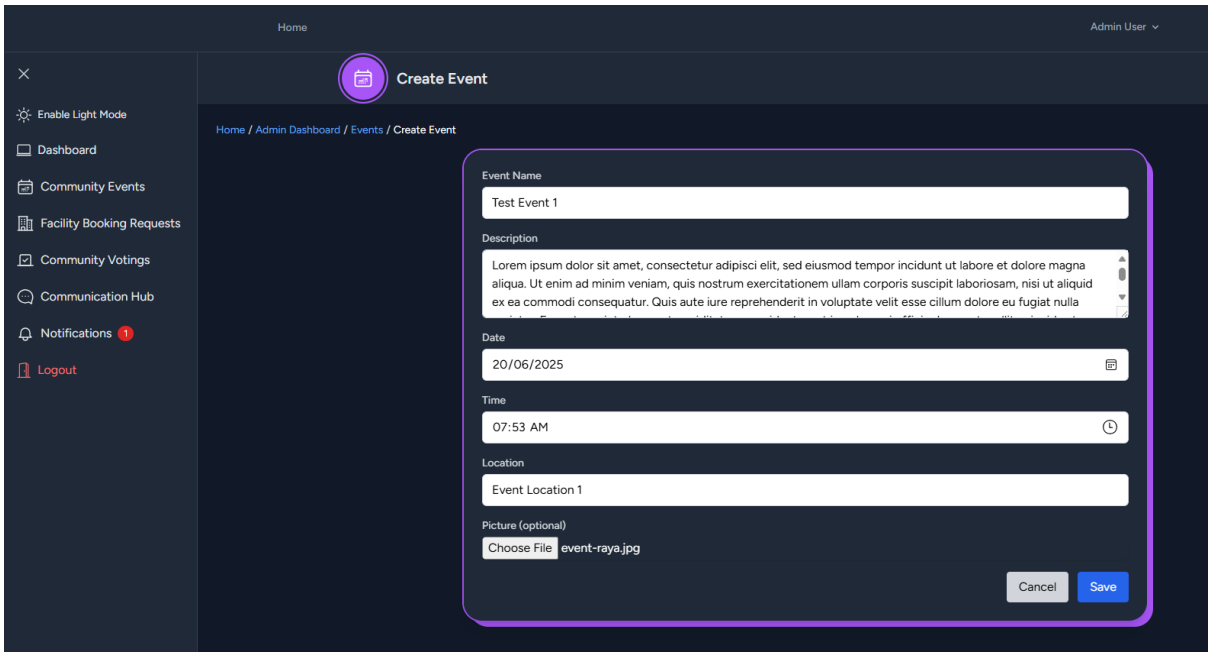


Figure 4.27: Admin event creation form with input fields for event details and image upload

Before the event is finalized and saved to the database, a confirmation modal appears to ensure the admin intends to proceed with the creation. This prompt helps prevent accidental

submissions by asking for explicit confirmation. As seen in Figure 4.28, the modal presents two options: Cancel to abort the action or Confirm to proceed with saving the event.

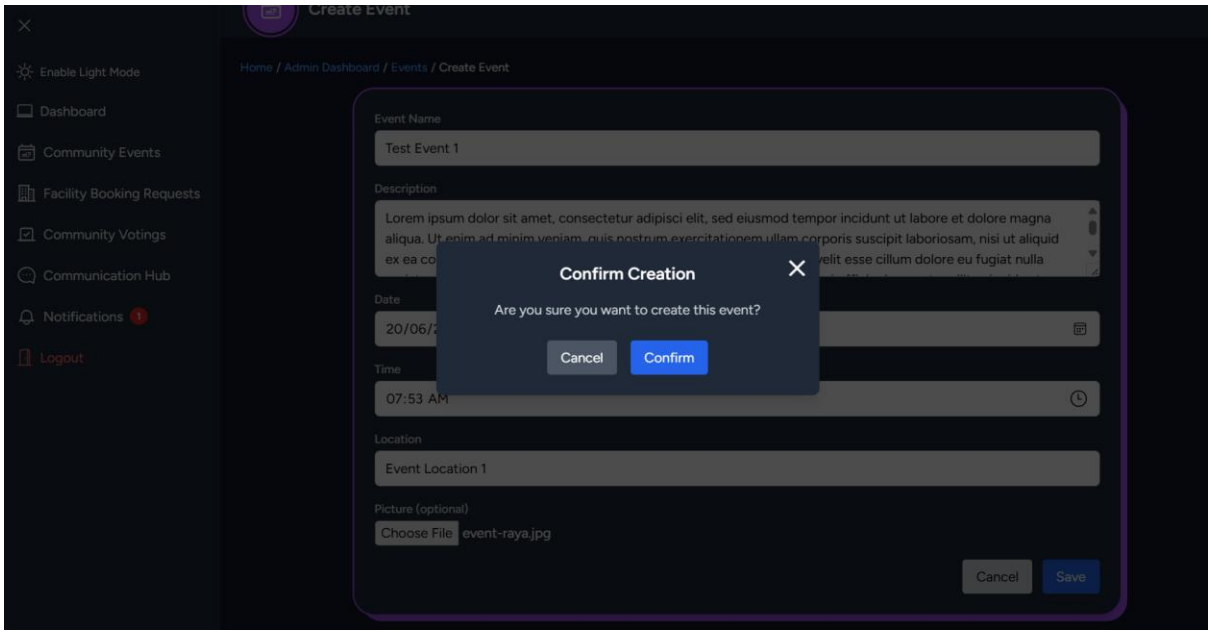


Figure 4.28: Confirmation modal displayed before submitting a new event creation

4.5.2.3 **Editing an Existing Event Form**

Existing events can be updated by clicking the edit icon next to each record. Figure 4.29 shows the event editing form, where all form fields are pre-filled with the existing event details. Admins can modify the name, description, date, time, and location, as well as optionally replace the current event poster with a new image.

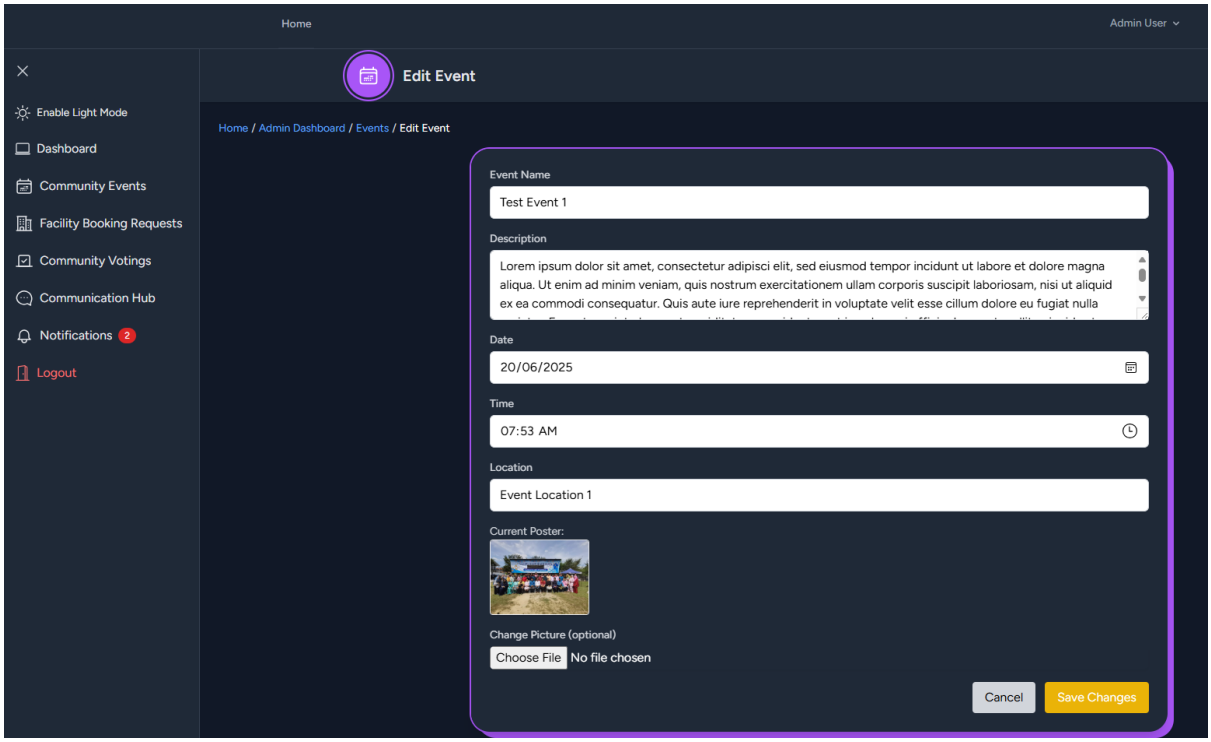


Figure 4.29: Edit Event form displaying current event details and allowing updates

After modifying an event, Admins are prompted with a confirmation modal to verify the update action. As seen in Figure 4.30, the modal asks for confirmation to proceed with saving the changes. This additional step helps prevent accidental modifications and ensures deliberate administrative actions.

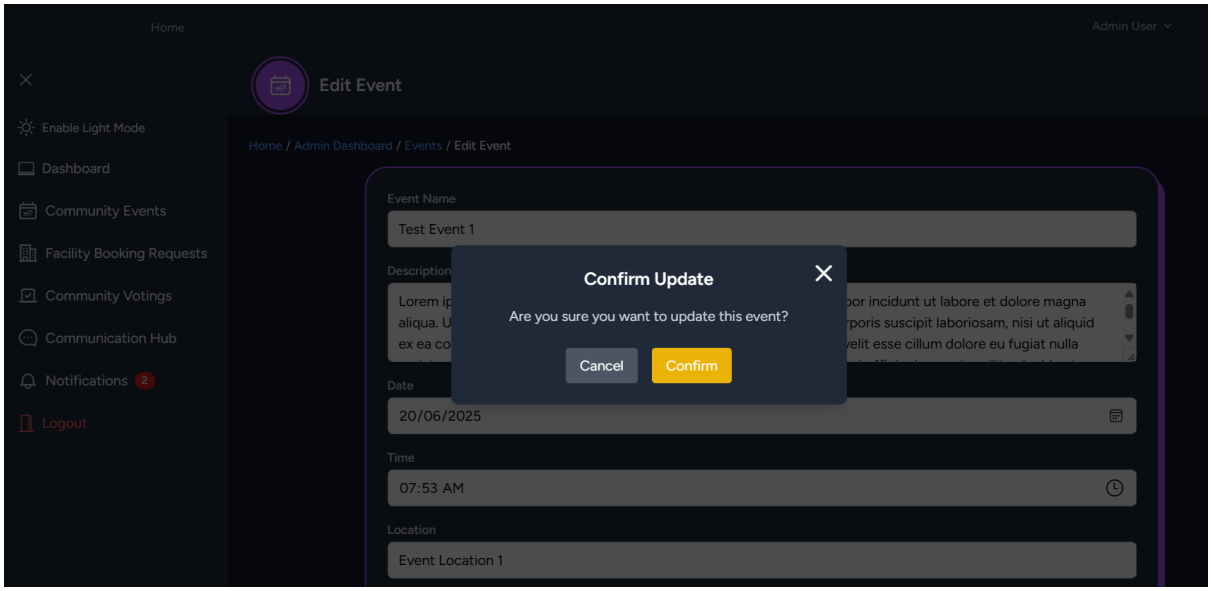


Figure 4.30: Update confirmation modal shown before applying changes to an existing event.

4.5.2.4 **Delete Event Confirmation**

To remove an event, the Admin can click the delete icon corresponding to the event. A confirmation modal will appear, prompting the Admin to confirm the deletion action to prevent accidental removal. As shown in Figure 4.31, this modal displays the event name and provides "Cancel" and "Confirm" buttons for user decision. Once confirmed, the event will be permanently deleted from the database.

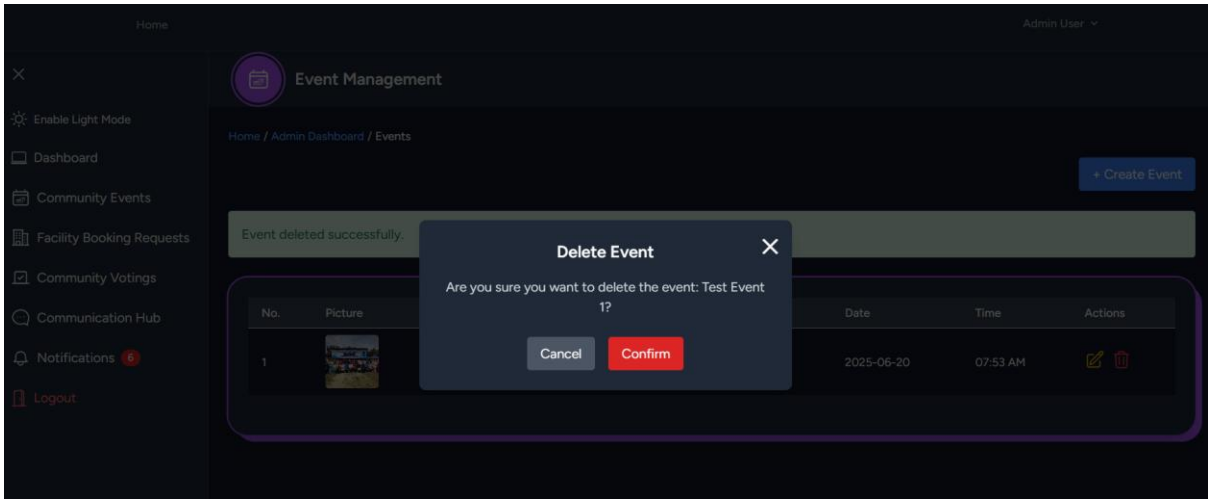


Figure 4.31: Confirmation modal for deleting an event in the Event Management module

4.5.2.5 **Success Confirmation Modal**

After a successful event creation, update, or deletion, a confirmation modal is displayed to inform the Admin that the action was completed. As shown in Figure 4.31, this modal confirms the success of the operation with a clear success message and a timed "OK" button that auto-dismisses. This feedback ensures users are aware that their requested action has been executed correctly and helps reinforce user confidence in the system's responsiveness.

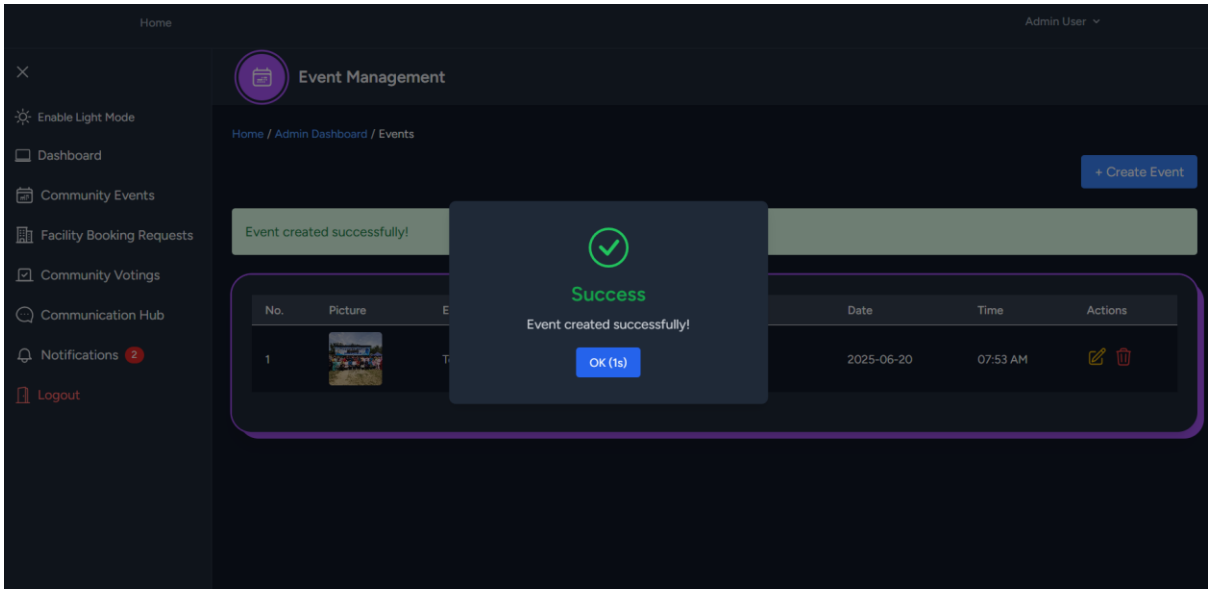


Figure 4.32: Success confirmation modal displayed after successful event operations such as creation, update, or deletion

4.5.2.6 **User View of Events**

On the user side, events are displayed in a card format within the Community Events page. Each card showcases the event's poster (if available), along with the event title, date, time, and location. A carousel highlights featured or ongoing events at the top, followed by categorized listings such as "Upcoming Events" and "Past Events." This visual layout enhances readability and allows users to quickly scan and access event information. Figure 4.33 illustrates the user-facing event listing interface.

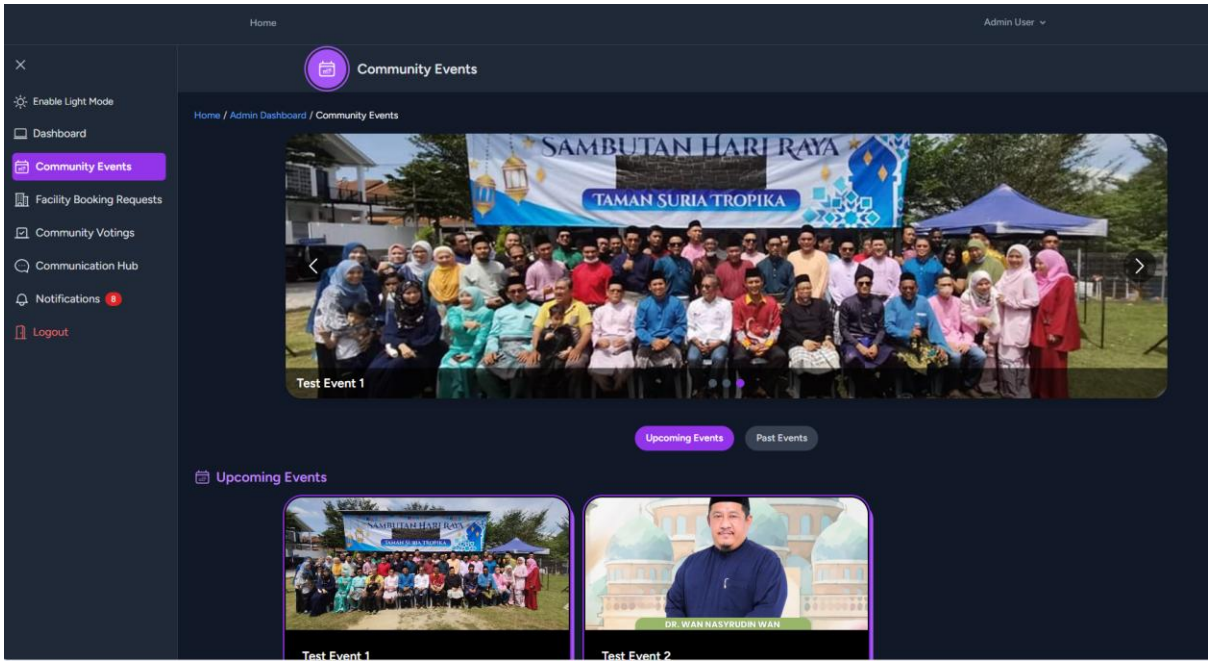


Figure 4.33: User view of the Community Events page with a featured carousel and card-based event listing

Each upcoming event is presented in an individual card layout that includes a poster image (if available), title, short description, date, time, location, and a "Details" button that links to more information. Figure 4.34 illustrates the interface layout as seen by regular users.

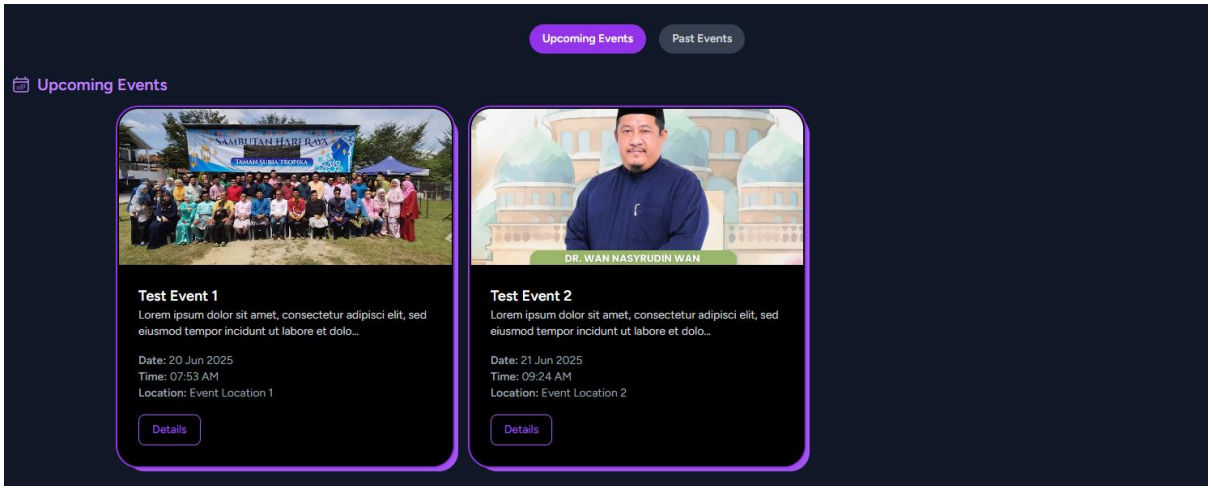


Figure 4.34: User View of Upcoming Events

Users can also toggle to view past events by selecting the "Past Events" filter. Events that have already occurred are displayed in the same card format. This feature allows residents to look back at previous activities and stay informed about the community's engagement

history. Figure 4.35 showcases the past events section with similar styling to the upcoming events view.

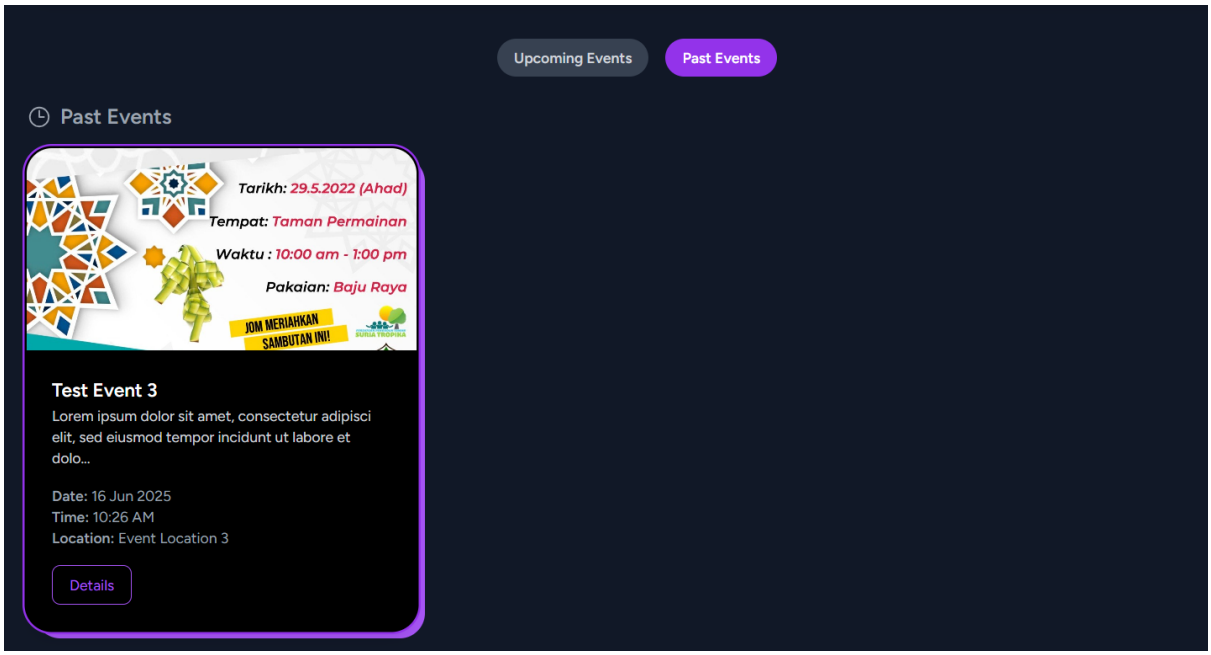


Figure 4.35: User View of Past Events

When a user clicks on any event card shown in the Community Events page, they are directed to a detailed view of that specific event. Figures 4.36 and 4.37 show the Event Details Page, where the full event poster or document is presented in a clean, scrollable format. This interface allows users to view event-specific information such as title, full description, date, time, and location. It is particularly useful for displaying long-form notices, detailed flyers, or community posters that require more space than the card previews.

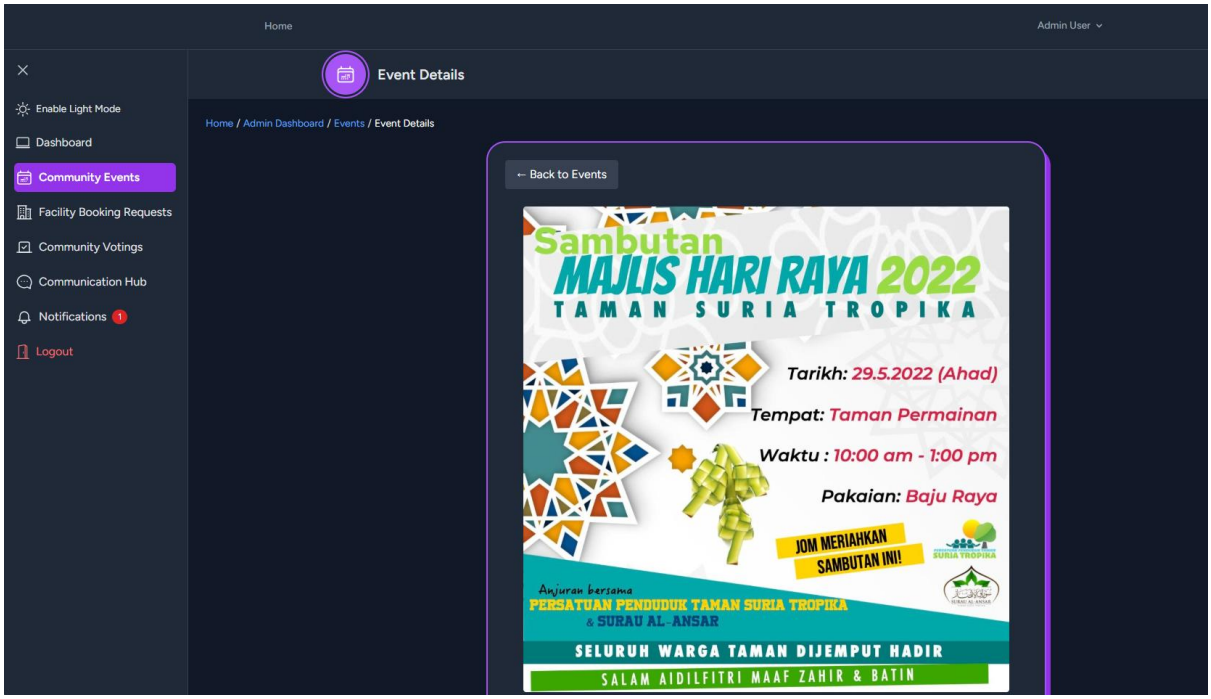


Figure 4.36: Upper section of the Event Details Page showing the featured event poster

Figure 4.37: Lower section of the Event Details Page showing additional event details

To improve the user experience when viewing attached images or posters, a fullscreen modal feature was implemented. When users click on the event image in the detail view, it opens in an overlay modal that expands the image to a larger size without navigating away from the page. As shown in Figure 3.8, the modal includes a clearly visible close button in the top-right corner and supports vertical scrolling for taller images. This design choice improves clarity, especially when displaying posters with detailed or lengthy information.

Figure 4.38: Fullscreen modal displaying an enlarged event poster

4.5.3 Financial Management Module

The Financial Management Module enables Admin users to oversee the surau's income and expenses efficiently. This module offers tools to input transactions, visualize financial data, and maintain an organized record of financial activities. The module features a summary dashboard, categorized transaction forms, and confirmation modals to ensure accuracy.

4.5.3.1 Financial Management Page

The Financial Management Page provides Admins with an overview of key financial statistics, including total income, total expenses, and the current balance. To enhance clarity in fund allocation, the dashboard includes visual breakdowns that can be toggled between pie and bar chart views. Recent transactions are also displayed at the bottom of the dashboard for quick

reference and tracking. Figure 4.39 shows the Financial Management Dashboard, including the pie chart, total amounts, and the latest transaction records.

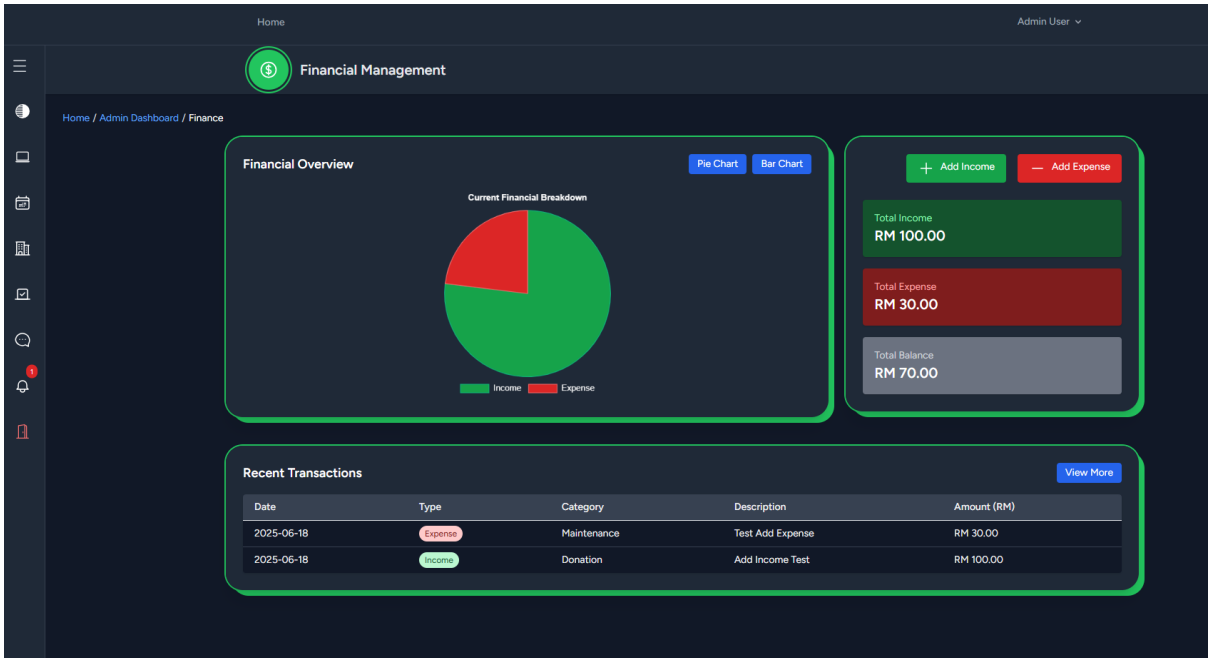


Figure 4.39: Financial Summary Dashboard

4.5.3.2 **Add Income Form**

Admins can add a new income transaction by clicking the Add Income button on the dashboard. This action redirects them to a form where they can input relevant details such as the category, description, amount, and date of the income. Figure 4.40 shows the Add Income form interface used to submit a new income transaction.

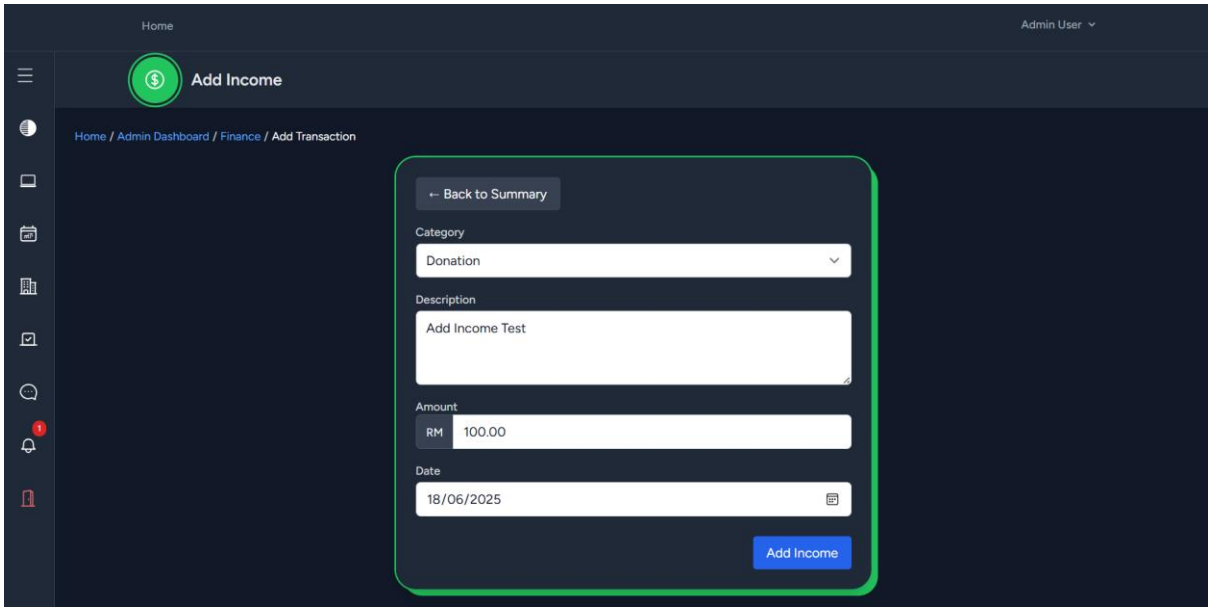


Figure 4.40: Add Income Form

After filling out the income form, a confirmation modal appears to ensure that the Admin verifies the transaction before saving. This helps avoid mistakes and accidental entries. Figure 4.41 shows the confirmation modal that appears when an income transaction is about to be submitted.

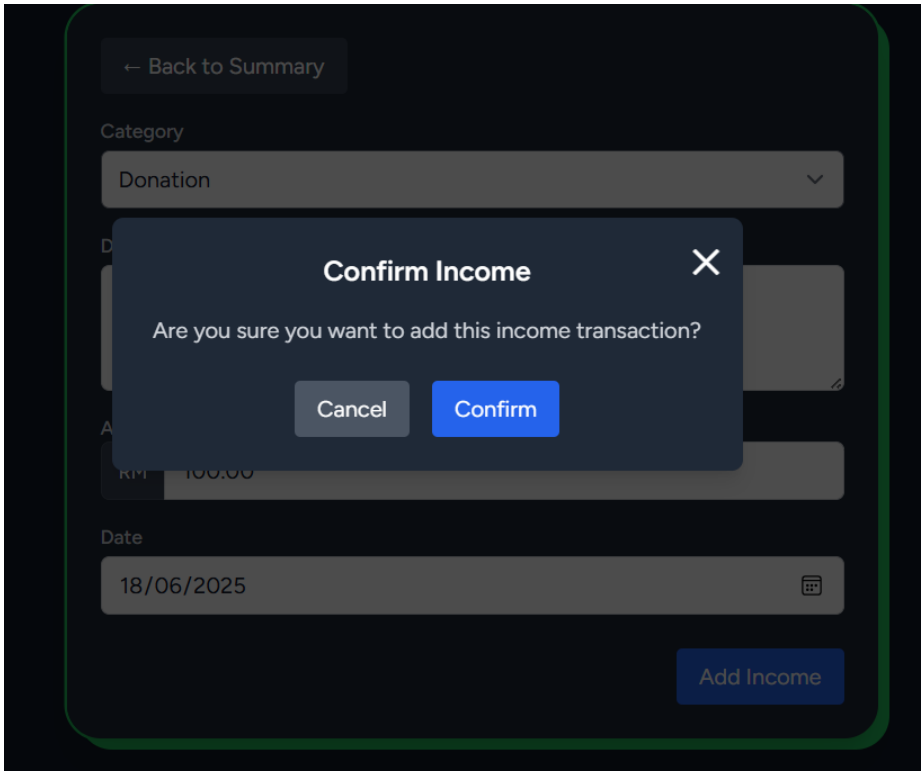


Figure 4.41: Confirmation modal displayed when an income transaction is about to be submitted

4.5.3.3 **Add Expense Form**

Similar to income, Admins can record expenses using the Add Expense form. This form collects details including category, purpose, amount, and transaction date. Figure 4.42 shows the form interface used for submitting an expense transaction.

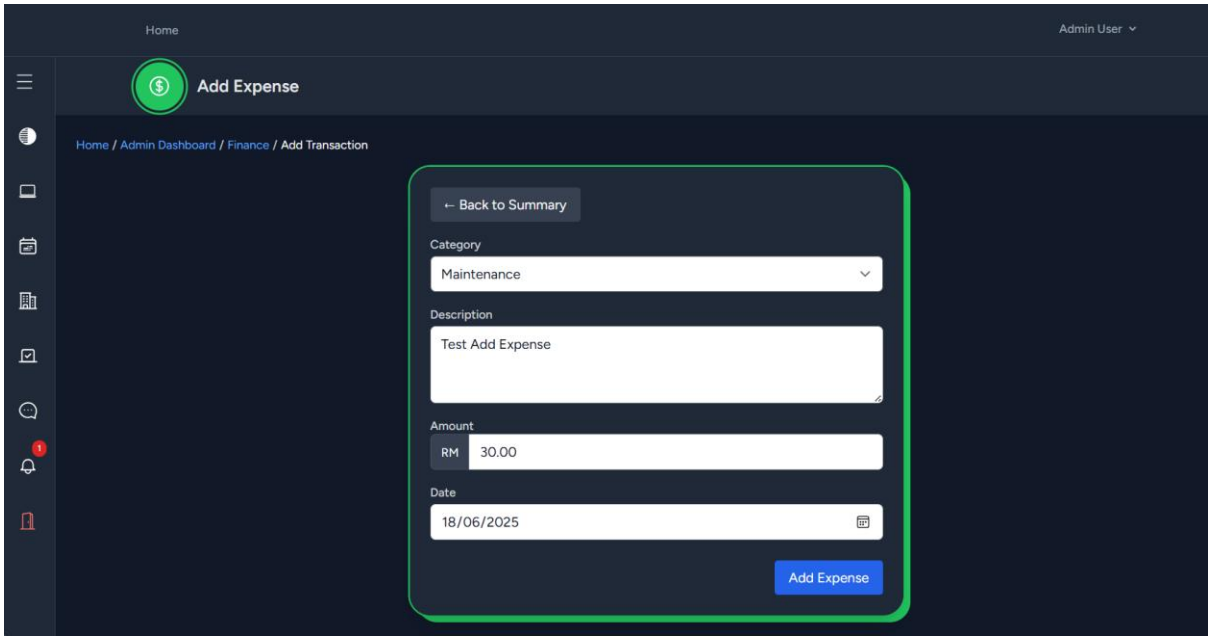


Figure 4.42: Add Expense Form

Upon submitting an expense, a confirmation popup appears to ask the Admin to confirm their input.. Figure 4.39 shows the confirmation modal displayed when an expense entry is about to be recorded.

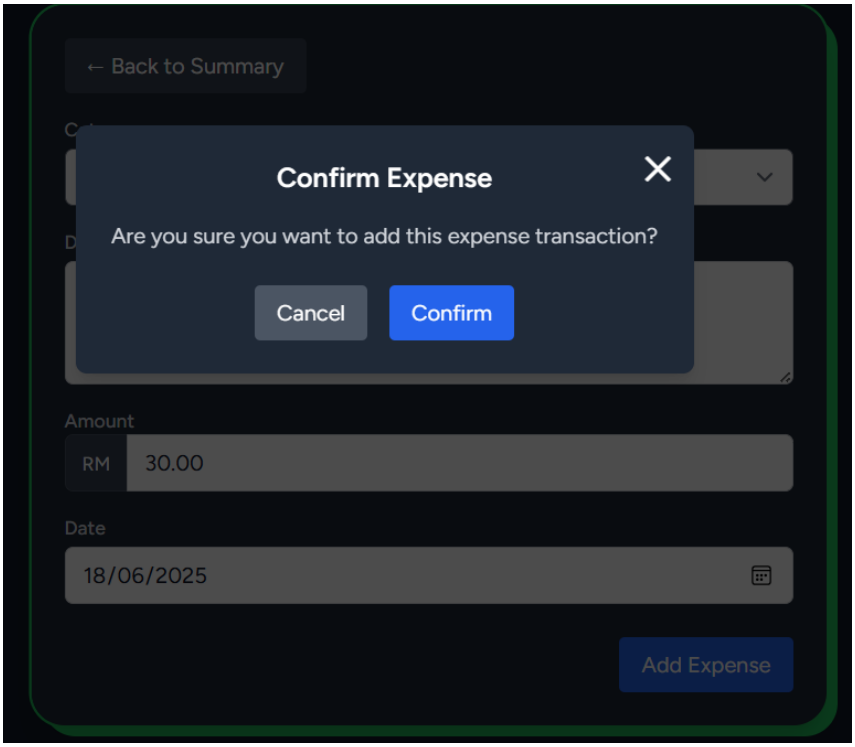


Figure 4.43: Confirmation modal displayed when an expense transaction is about to be submitted

4.5.3.4 **Full Transaction History Page**

Clicking the "View More" button on the Financial Management Page redirects Admin users to the Full Transaction History page. This interface provides a comprehensive list of all recorded income and expense transactions, each categorized by type, date, description, and amount. The table supports real-time filtering by category or search input, as well as data export functionality via the "Download as Excel" button. This feature ensures transparency and traceability of all financial entries for audit or reporting purposes. Figure 4.44 shows the Full Transaction History page with available filtering and action buttons for editing or deleting records.

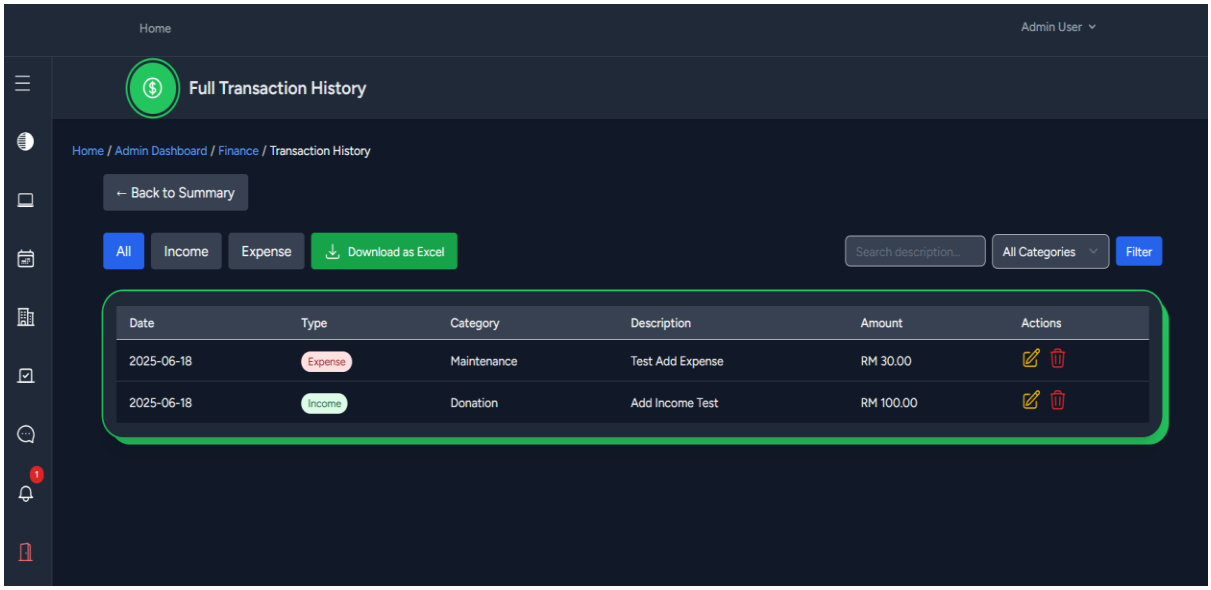


Figure 4.44: Full Transaction History Page with Filter and Export Features

4.5.3.5 **Edit Transaction form**

Admins can update any previously recorded transaction by selecting the edit icon from the full transaction history page. This leads to the Edit Transaction form, where the transaction type, category, description, amount, and date are pre-filled with current values. Figure 4.45 shows the layout of the Edit Transaction form, designed for clarity and ease of correction.

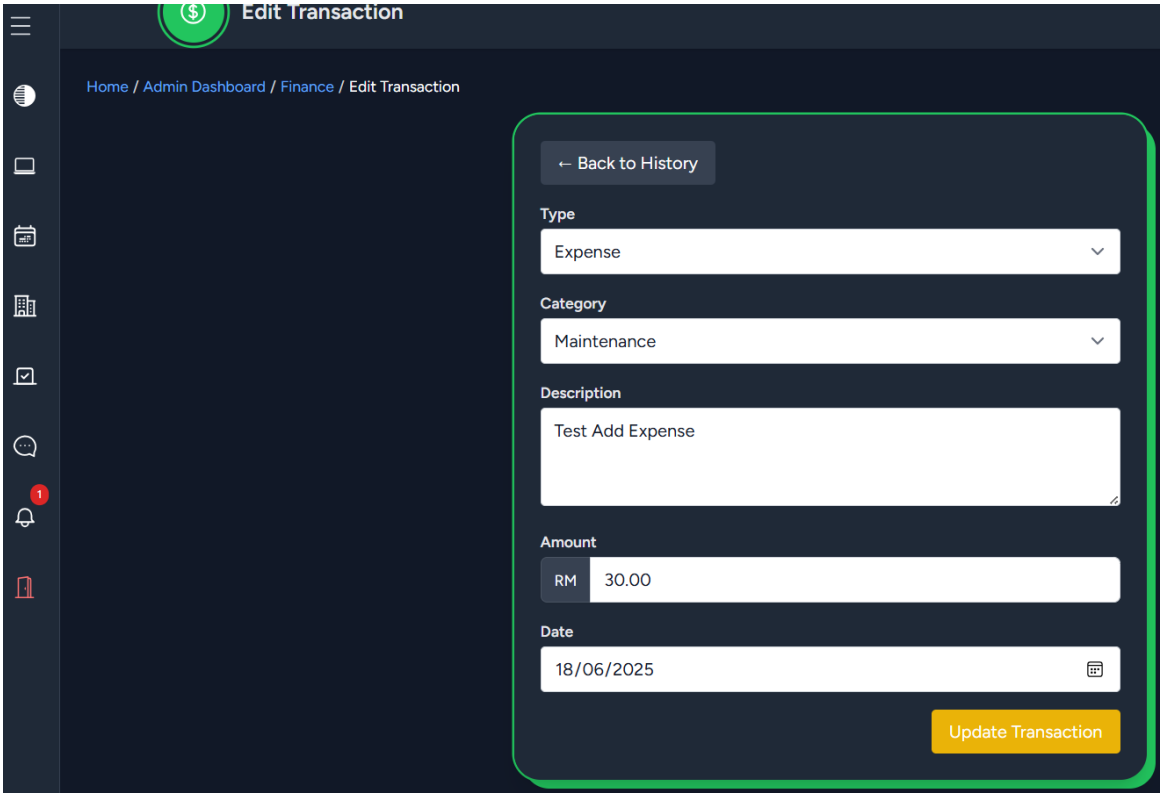


Figure 4.45: Edit Transaction Form

Once an admin clicks the "Update Transaction" button, a confirmation modal is triggered to prevent accidental changes. This modal, shown in Figure 4.46, asks the admin to confirm the update before saving the revised entry.

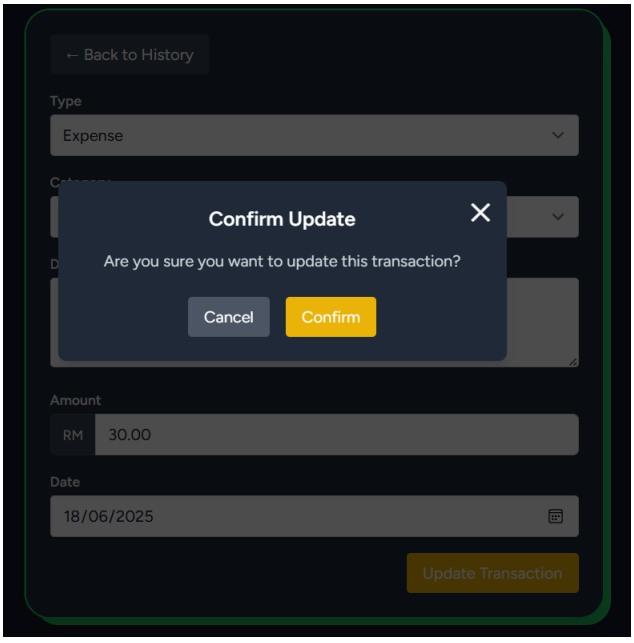


Figure 4.46: Confirmation modal that appears when updating a transaction

4.5.3.6 **Delete Transaction Confirmation**

To ensure deliberate action and prevent accidental deletion, the system prompts the Admin with a confirmation modal when attempting to delete a transaction from the full transaction history. The modal explicitly states the transaction details (e.g., category and amount) for clarity. Figure 4.47 shows the delete confirmation modal, which appears after clicking the trash icon beside the selected transaction.

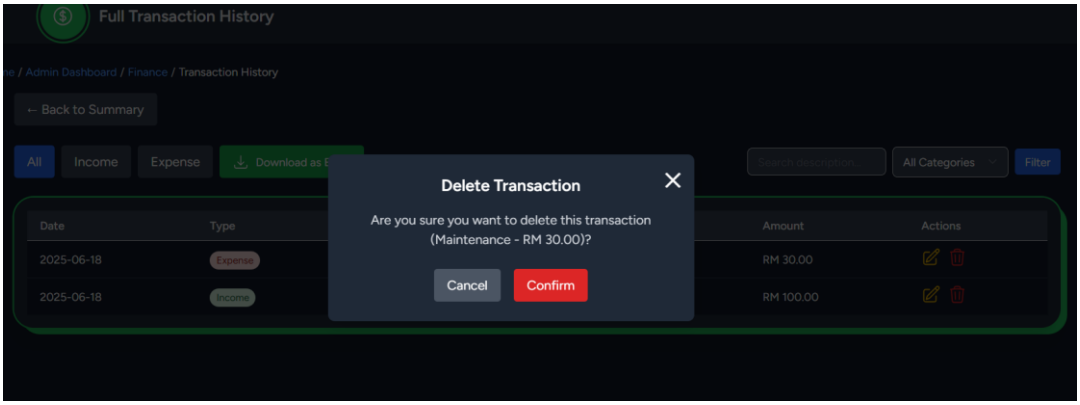


Figure 4.47: Confirmation modal displayed when an Admin initiates deletion of a financial transaction

4.5.3.7 **Success Confirmation Modal**

Once a transaction is successfully added, either income or expense transaction, a modal pops up to confirm that the action was completed. This provides the Admin with immediate feedback and assurance. Figure 4.48 shows the success message that appears after a transaction is successfully saved to the system.

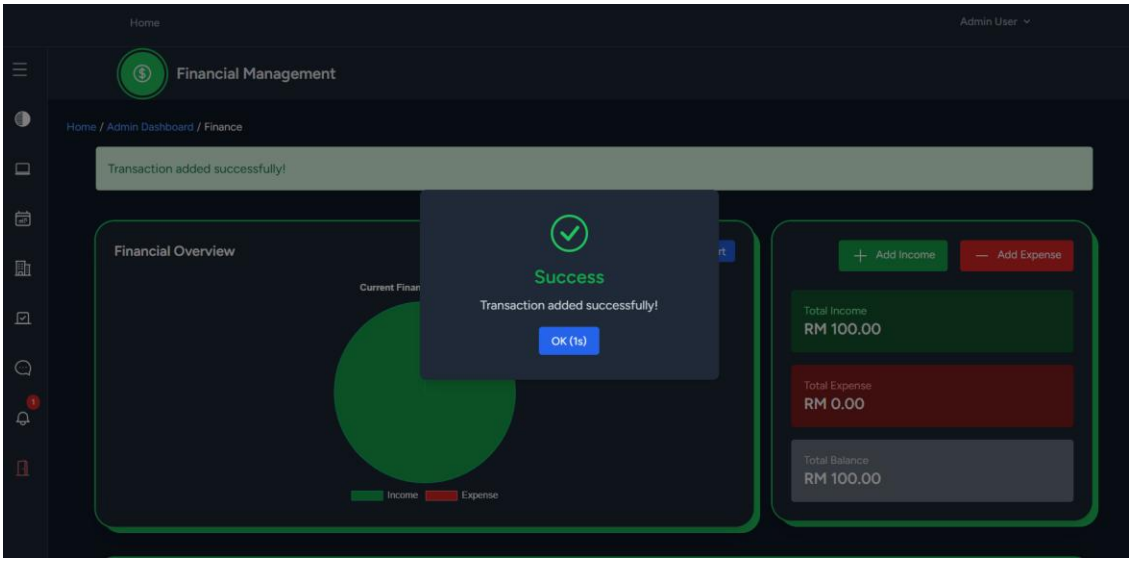


Figure 4.48: Success Confirmation Modal

4.5.4 Facility Booking Request Module

The Facility Booking Request Module is designed to allow users to reserve communal spaces or facilities such as prayer halls, meeting rooms, or outdoor areas within the residential community. This module provides a structured workflow where users can submit booking requests, view their status, and access a calendar-based view of availability.

On the admin side, the admins can manage and review incoming requests, approve or reject bookings, and monitor facility usage to prevent conflicts or overlaps. By incorporating a transparent and interactive system, this module enhances the coordination and fairness of facility usage across the community.

It should be taken note that, similar to other modules, this section also includes confirmation modals, success notifications, and deletion confirmations when users or admins perform booking-related actions. However, to avoid redundancy, these standard modal interactions are not showcased here, as they have already been demonstrated and described in earlier modules such as Event Management and Financial Management.

4.5.4.1 Facility Booking Page (User)

The user-side interface of the Facility Booking Module provides a centralized dashboard for residents to manage their own booking requests. Upon navigating to the "My Facility Booking Requests" page, users are presented with a table view listing all of their previously submitted

bookings along with essential details such as facility name, date, start time, end time, and current status. Action buttons are available to view the full request or delete pending submissions. This allows users to monitor their bookings efficiently and make changes as needed. Figure 4.49 shows the user facility booking dashboard with an overview of recent booking requests.

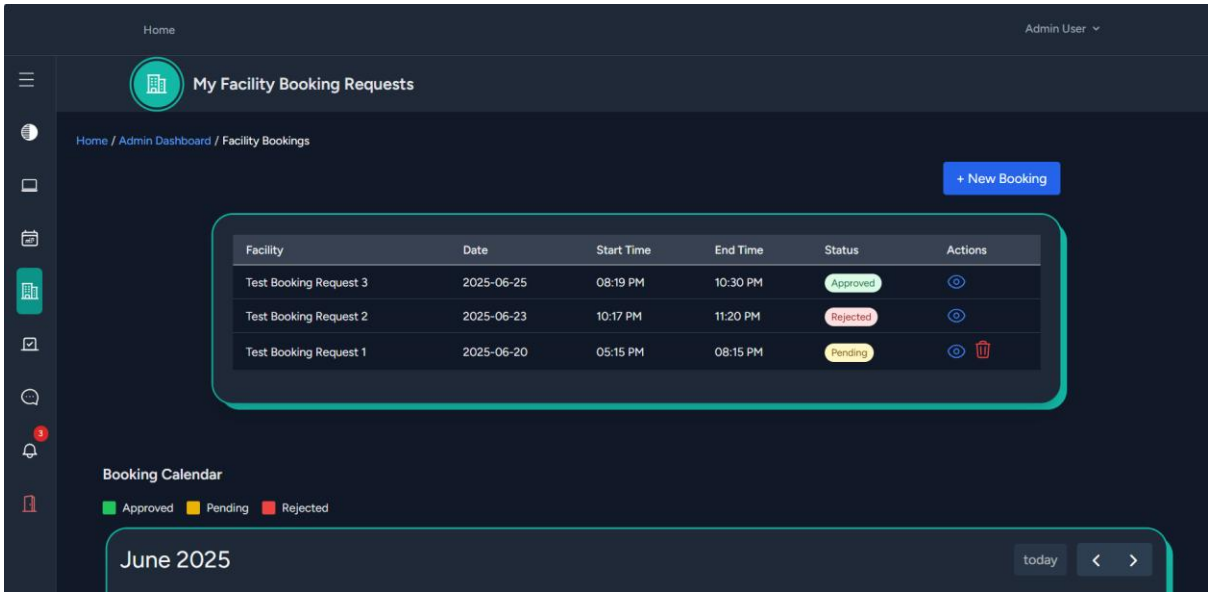


Figure 4.49: User Facility Booking Dashboard

Beneath the table, a built-in calendar provides a visual reference of upcoming facility bookings. It uses color-coded indicators to distinguish between approved (green), pending (yellow), and rejected (red) requests. This enhances user awareness of schedule availability and helps prevent overlapping submissions. Figure 4.50 illustrates the integrated booking calendar displayed on the user dashboard.

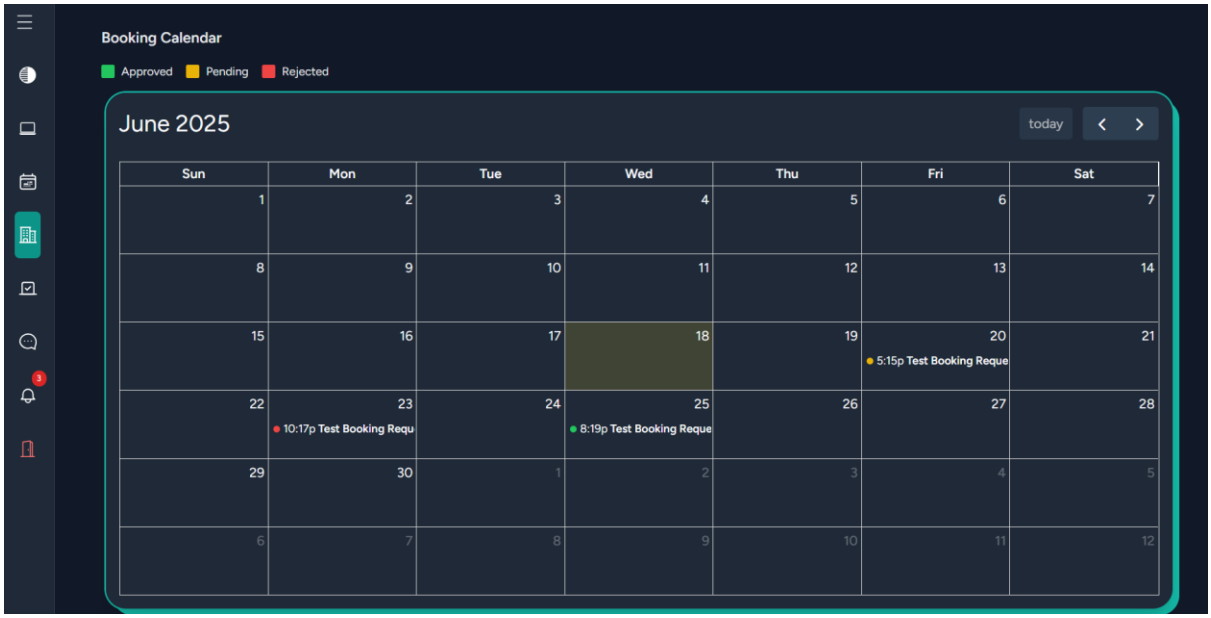


Figure 4.50: Booking Calendar

4.5.4.2 **Create Booking Request Form**

Users can submit a new facility booking by clicking the "New Booking" button, which opens the booking request form. This form collects key details such as the selected facility, booking purpose, date, start time, and end time. Proper validation ensures that end time is after start time to prevent scheduling conflicts. Figure 4.51 shows the booking request form interface that users fill out to submit a new request.

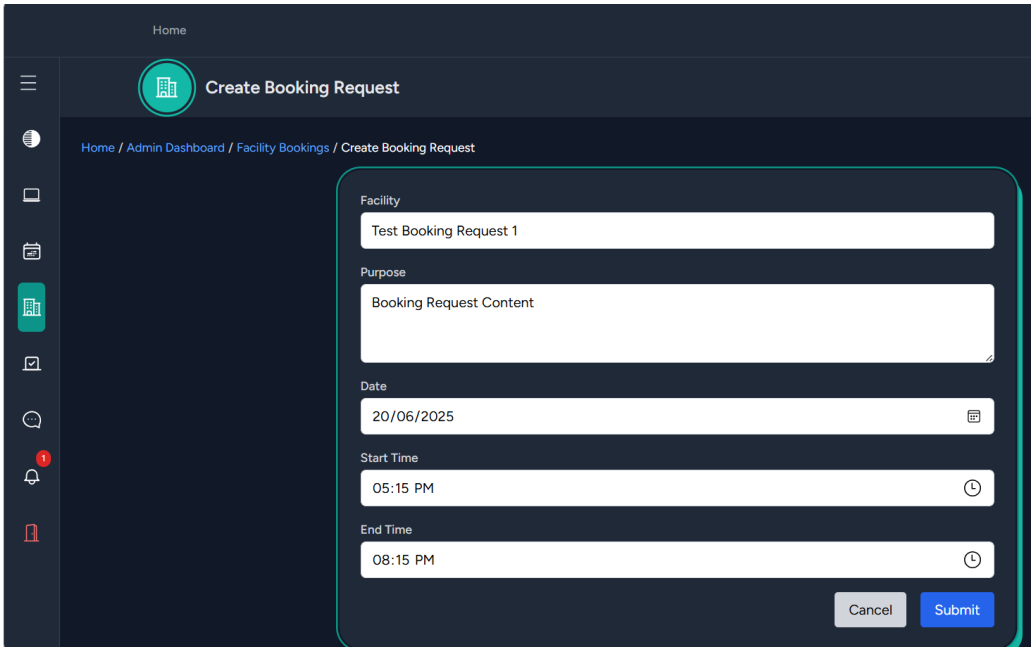


Figure 4.51: Create Booking Request Form

Before submission, a confirmation modal appears to allow users to verify their entries. This dialog prevents accidental submissions and ensures users are aware of the action being taken. Figure 4.52 shows the confirmation modal that appears when submitting a booking request.

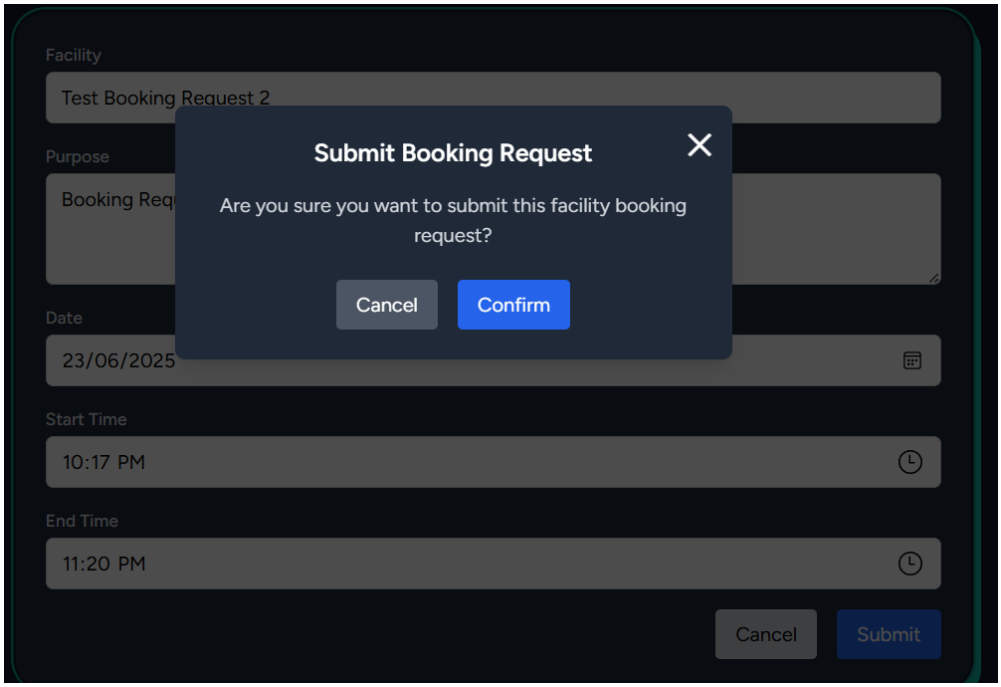


Figure 4.52: Confirmation modal that appears when submitting a Booking Request

4.5.4.3 **User View of Submitted Booking Request**

After submission, users can view the details of their individual booking requests by clicking the “View” button in the actions column of the booking dashboard. This brings up a dedicated view-only page that displays all the information provided during the request, including the facility name, purpose, date, time range, and current approval status. Figure 4.53 shows the detailed view of a submitted facility booking request.

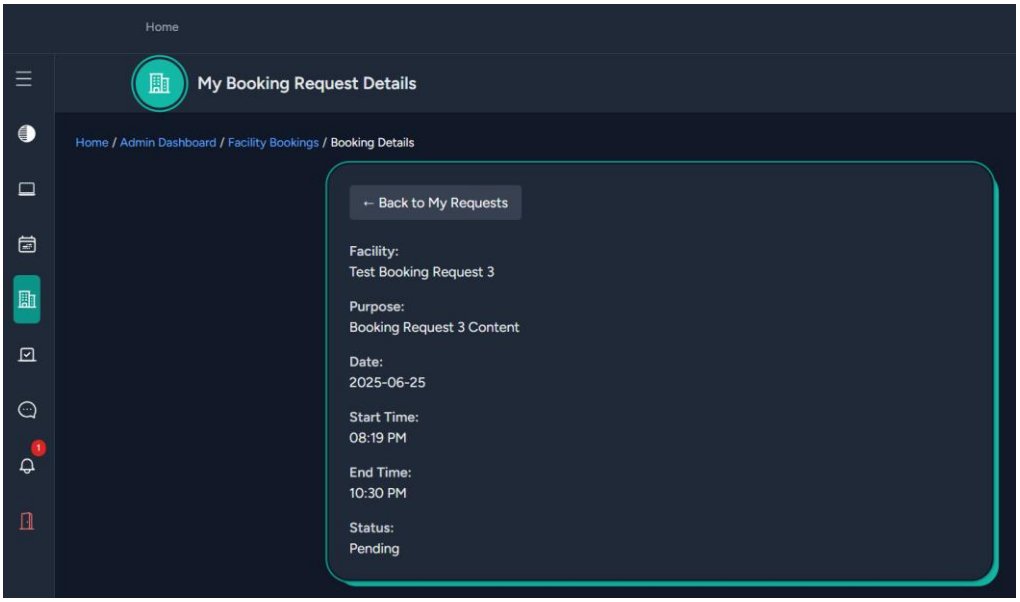


Figure 4.53: Detailed view of a specific booking request after submission

4.5.4.4 **Facility Booking Management Page (Admin)**

Admin users have access to a centralized management interface for handling all incoming facility booking requests. As shown in Figure 4.54, the dashboard presents a tabulated list of all user-submitted booking requests, displaying details such as facility name, requested date and time, status, and the requester's name. A calendar view beneath the table provides a visual overview of upcoming bookings, categorized by status (Approved, Pending, or Rejected) using color-coded labels for easy tracking.

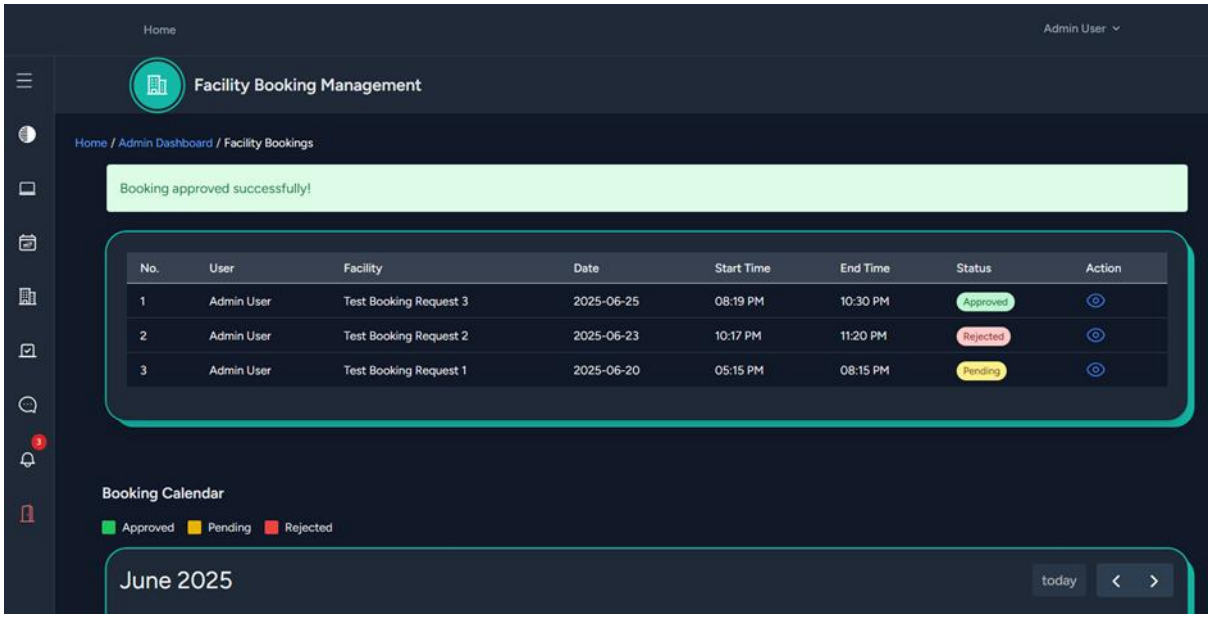


Figure 4.54: Admin Facility Booking Management Dashboard

4.5.4.5 **Admin View of Individual Booking Request**

By clicking the “View” icon in the action column, Admins can open a detailed page for any specific booking. As shown in Figure 4.55, this page displays the full submission, including requester name, facility, purpose, date, time, and current status. Two buttons, which are ‘Approve’ and ‘Reject’, are available for actioning the request.

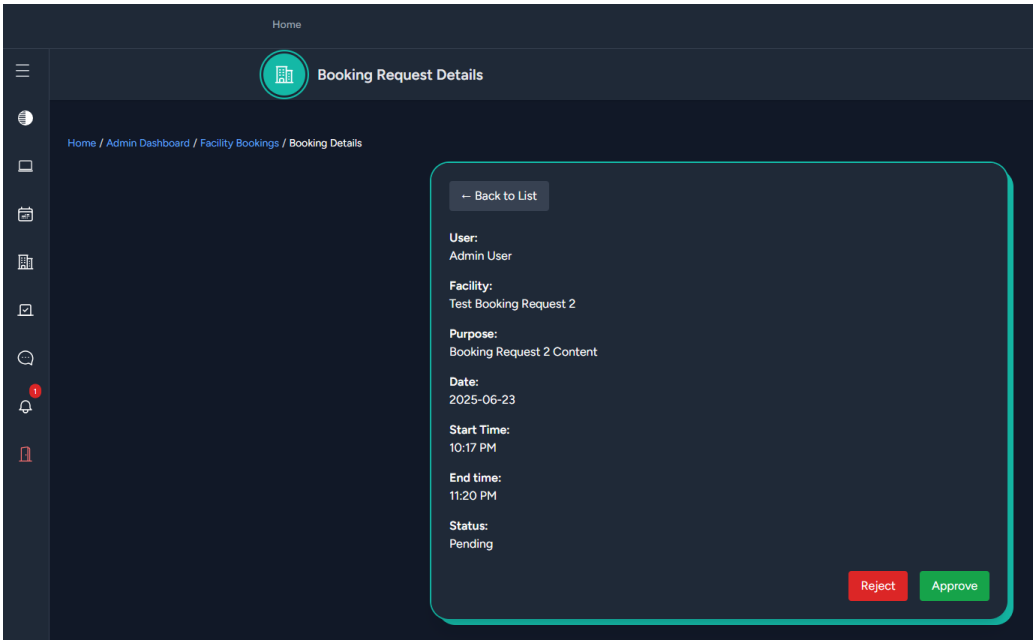


Figure 4.55: Detailed view of a facility booking request with approve/reject options

4.5.4.6 **Approving a Booking Request**

When choosing to approve a booking, the Admin is prompted with a confirmation modal. This step helps prevent accidental approval, and also gives the Admin the option to enter a note to it as a way of explaning the reason for approval. The approval modal is shown in Figure 4.56.

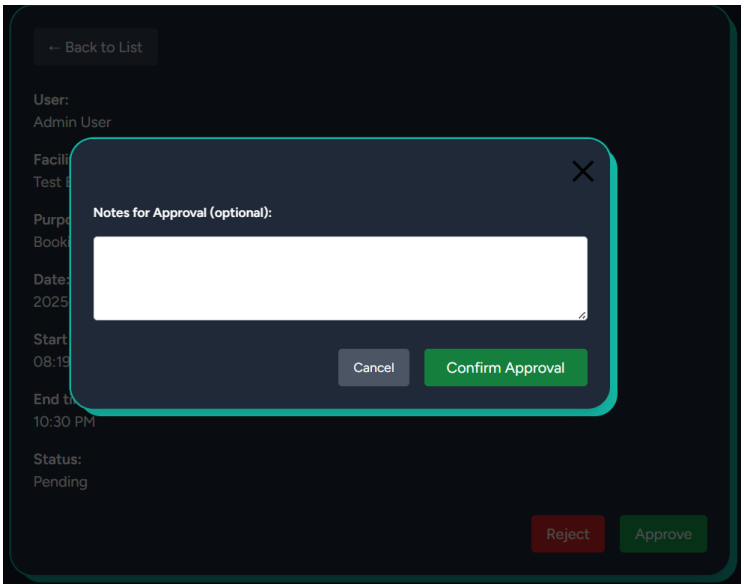


Figure 4.56: Modal confirmation window for approving a facility booking request

4.5.4.7 **Rejecting a Booking Request**

Similar to the approval part, selecting the “Reject” button opens a modal asking the Admin to confirm the rejection. Notes can be optionally provided to explain the reason for rejection. The rejection modal is shown in Figure 4.57.

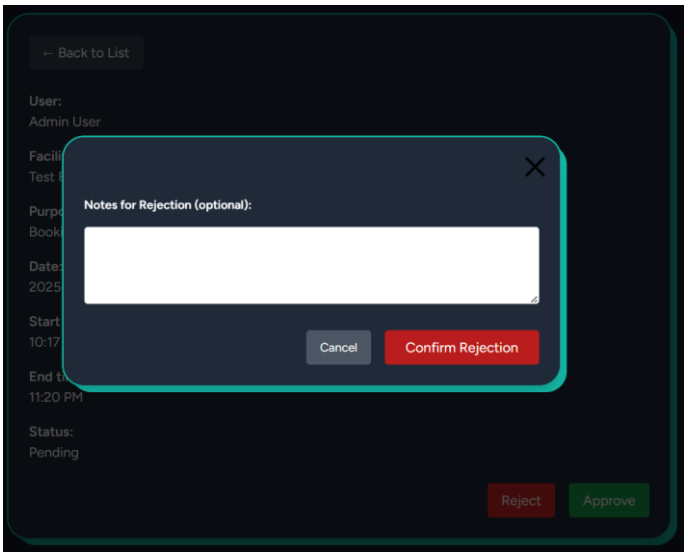


Figure 4.57: Modal confirmation window for rejecting a facility booking request

4.5.4.8 **Admin View of the Booking Calendar**

The calendar integrated into the Admin Dashboard mirrors the one available to users, featuring the same interactive functionality such as color-coded booking indicators and clickable entries that display booking details in a modal popup. While both calendars provide a consistent and user-friendly interface, the admin version grants access to all facility booking requests submitted by any user, whereas the user calendar only displays the bookings relevant to the currently logged-in individual. Figure 4.58 illustrates the interactive facility booking calendar on the admin dashboard.

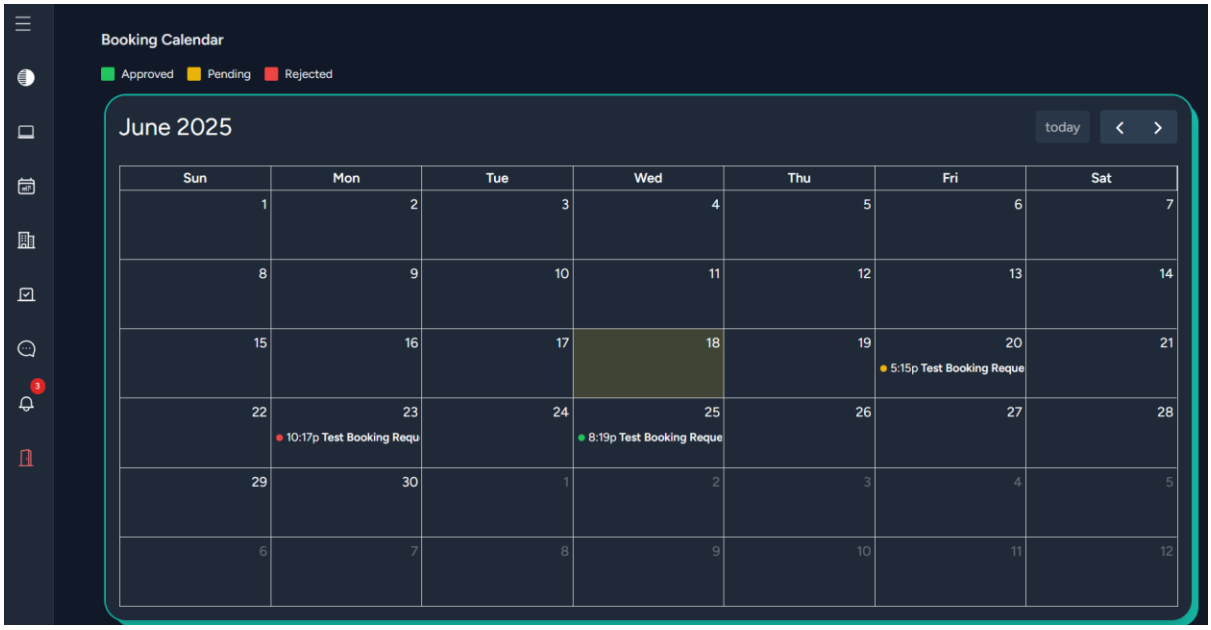


Figure 4.58: Calendar view displaying all facility booking statuses for the month of June 2025 on the admin side

When an Admin clicks on a booking entry in the calendar, a modal appears displaying detailed information about the selected request. This includes the date, time, approval status, purpose, and any notes added by the Admin. This modal enhances clarity and allows quick access to contextual booking information. The same functionality is available on the user side shown in Section 4.5.4.1, where users can also click on calendar entries to view request details in a similar format. Figure 4.59 illustrates the admin-side modal that appears upon selecting a booking entry from the calendar.

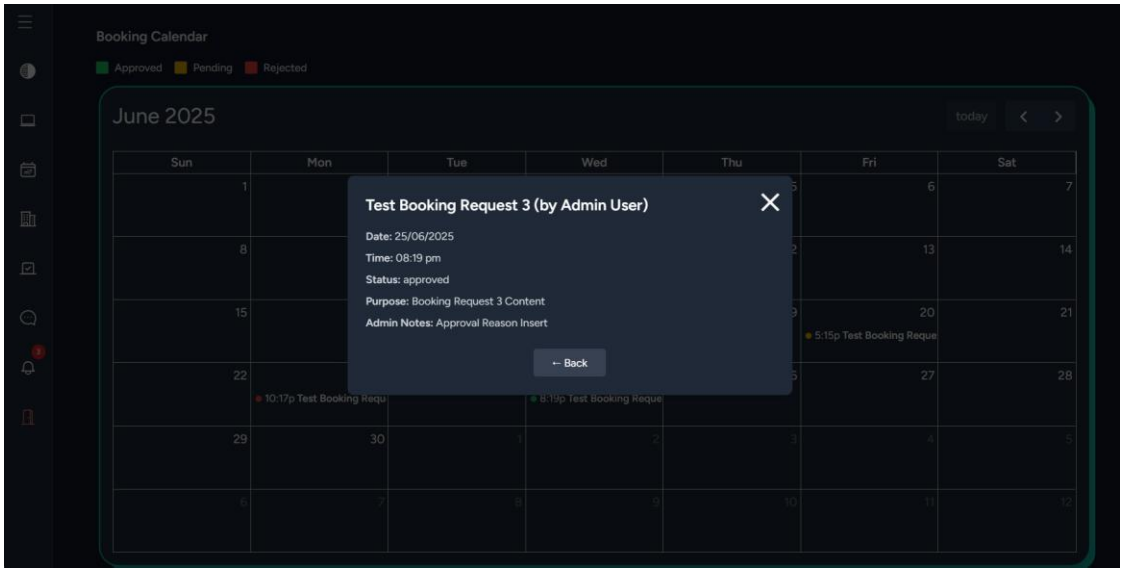


Figure 4.59: Admin Calendar Modal for Booking Details

4.5.5 Community Voting Module

The Community Voting Module enables structured decision-making within the community by allowing admins to create voting sessions and users to participate in them. This feature promotes transparency and inclusiveness by letting residents cast their votes on matters that require community decisions. The module includes interfaces for both administrators and users, with tools to create, manage, and view voting results.

It should be taken note that, similar to other modules, this section also includes confirmation modals, success notifications, and deletion confirmations when users or admins perform voting-related actions. However, to avoid redundancy, these standard modal interactions are not showcased here, as they have already been demonstrated and described in earlier modules such as Event Management and Financial Management.

4.5.5.1 Voting Management Page (Admin)

The Admin interface of the Community Voting Module provides a centralized area to manage all voting sessions. As shown in Figure 4.60, the dashboard displays a list of all created votings with their respective titles, start and end dates, timeframes, and current statuses (e.g., Active or Inactive). Admins can also access action buttons to view, edit, or delete each session.

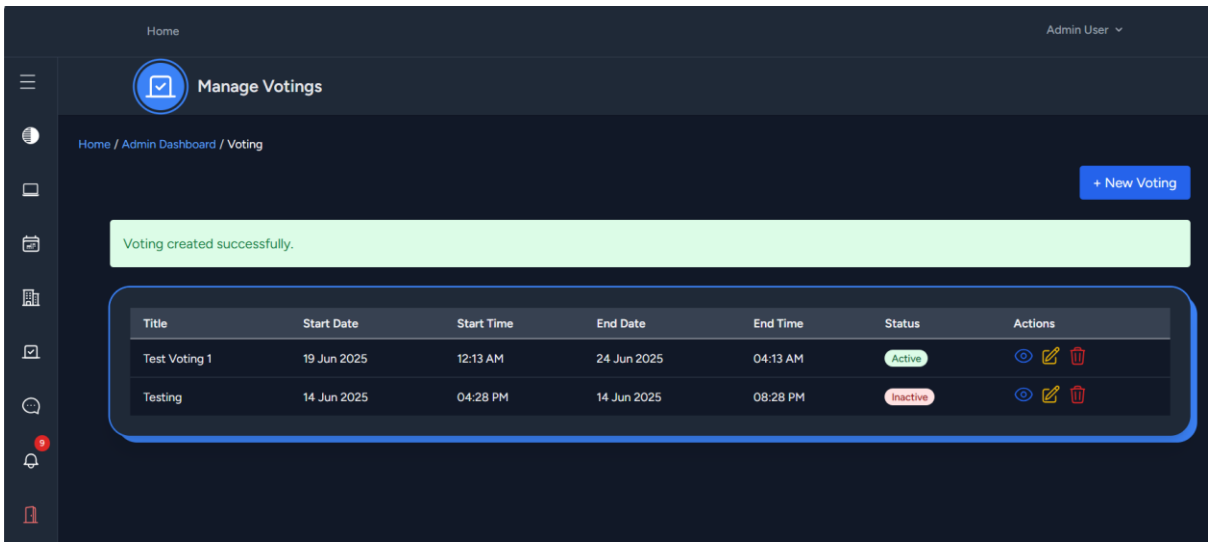


Figure 4.60: Admin view of the Community Voting list

4.5.5.2 **Voting Details Page (Admin)**

When an Admin clicks the view icon in the voting list, they are directed to the Voting Details Page, as illustrated in Figure 4.61 and Figure 4.62. This page displays the voting data, such as the title, schedule, current status, and a short description. Admins are provided with a red "End Voting & Notify Users" button to conclude the session.

Beneath this section, the "Voting Choices" block allows Admins to add, edit, or delete the available options for users to vote on. Below that, a real-time "Voting Results" section shows vote counts and percentages for each choice, offering instant feedback on user engagement.

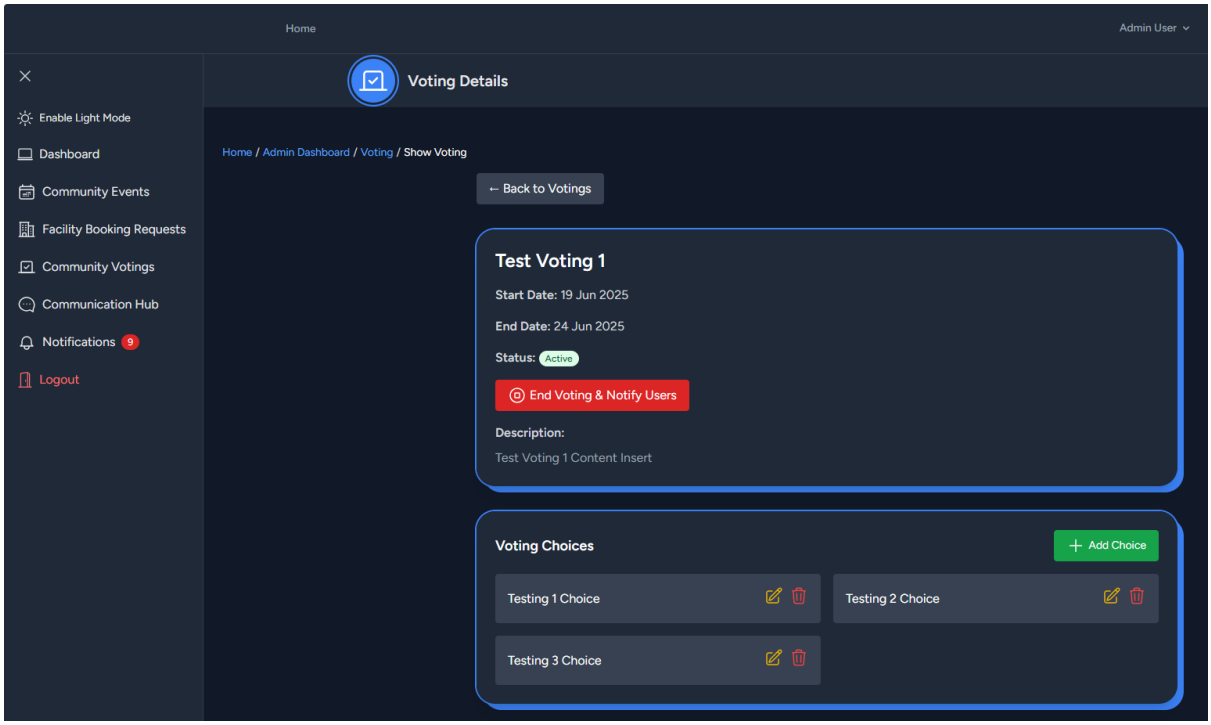


Figure 4.61: Voting Details with editable choices

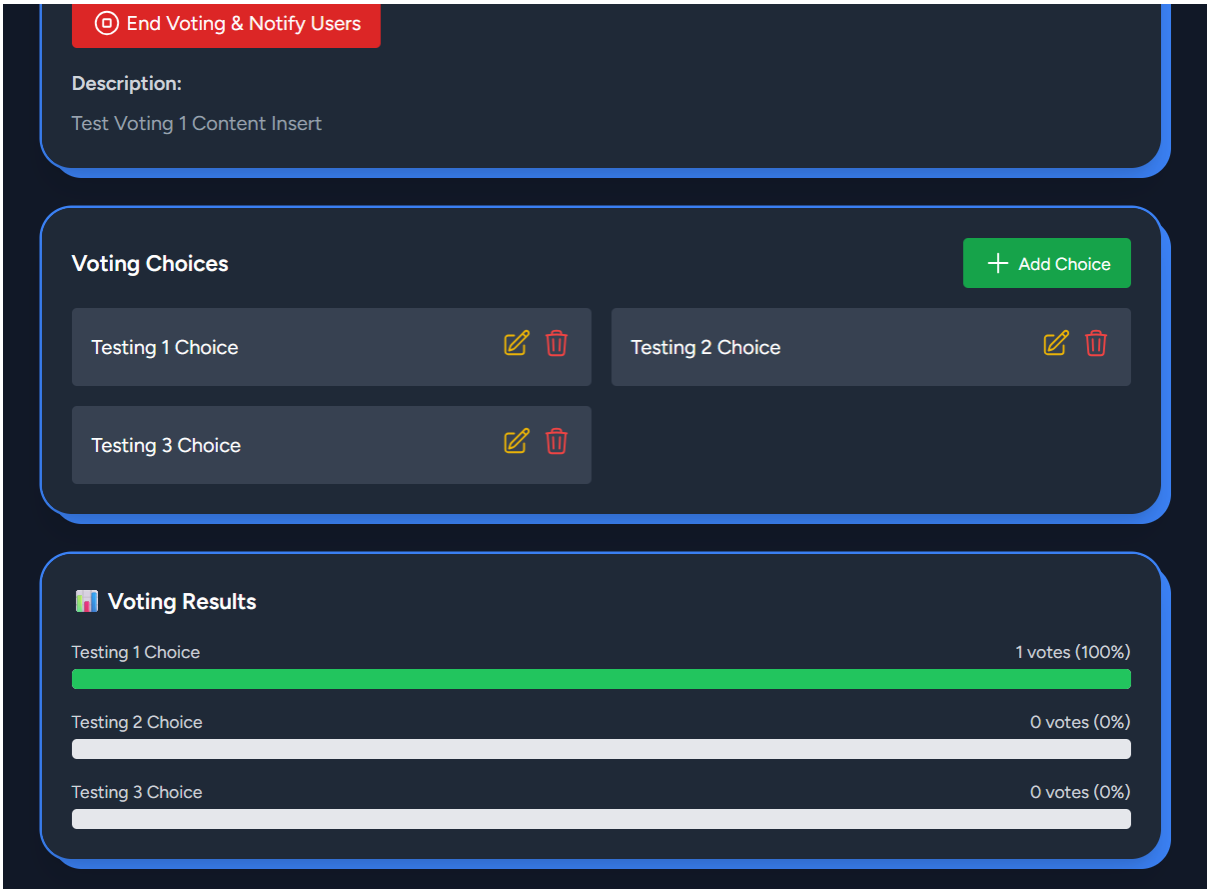


Figure 4.62: Voting Details with voting results

4.5.5.3 **Voting Edit Page (Admin)**

Admins can update an existing voting session by clicking the edit icon located in the actions column of the voting list. This redirects to a form where the existing voting details are pre-filled, allowing admins to make changes conveniently. Editable fields include the title, description, start and end date, and start and end time.

Below these inputs, the existing voting choices are listed, each with a delete button. Admins may also add new choices dynamically using the “Add another choice” option. Once updates are finalized, clicking the “Update Voting” button saves the changes. Figures 4.63 and 4.64 together illustrate the top and bottom portions of the admin voting editing interface.

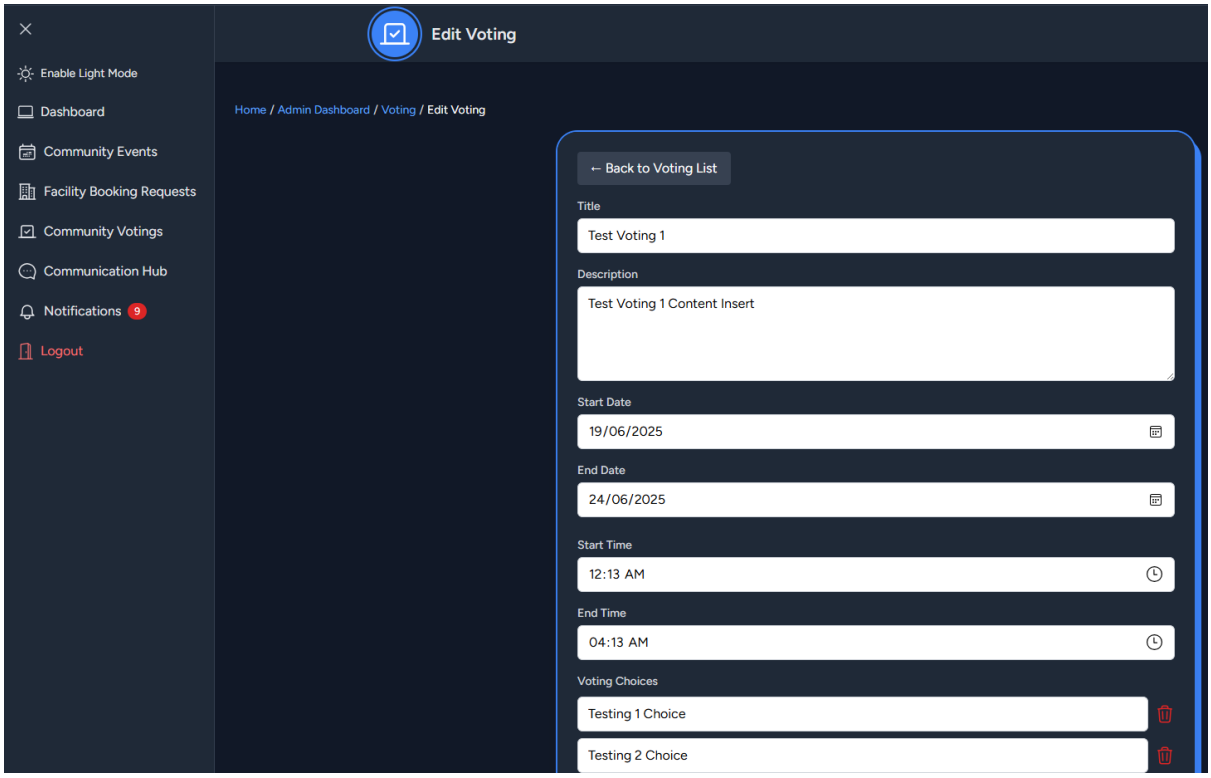


Figure 4.63: Editing page for voting (Part 1)

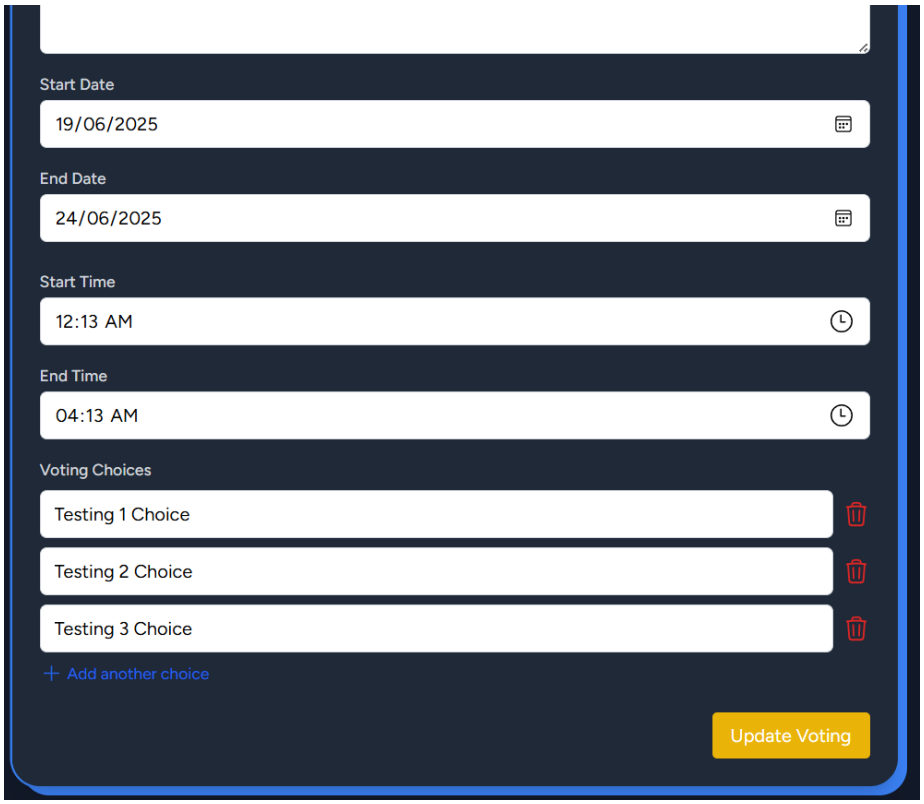


Figure 4.64: Editing page for voting (Part 2)

4.5.5.4 Community Voting Interface for Users

On the user side, community members can access a list of ongoing and past votings through the Community Voting page. Each voting is presented in a card format that displays its title, description, and active duration. If a voting session is active, users will see a “Vote Now” button, allowing them to participate immediately. Figure 4.65 illustrates the interface where available votings are displayed to users.

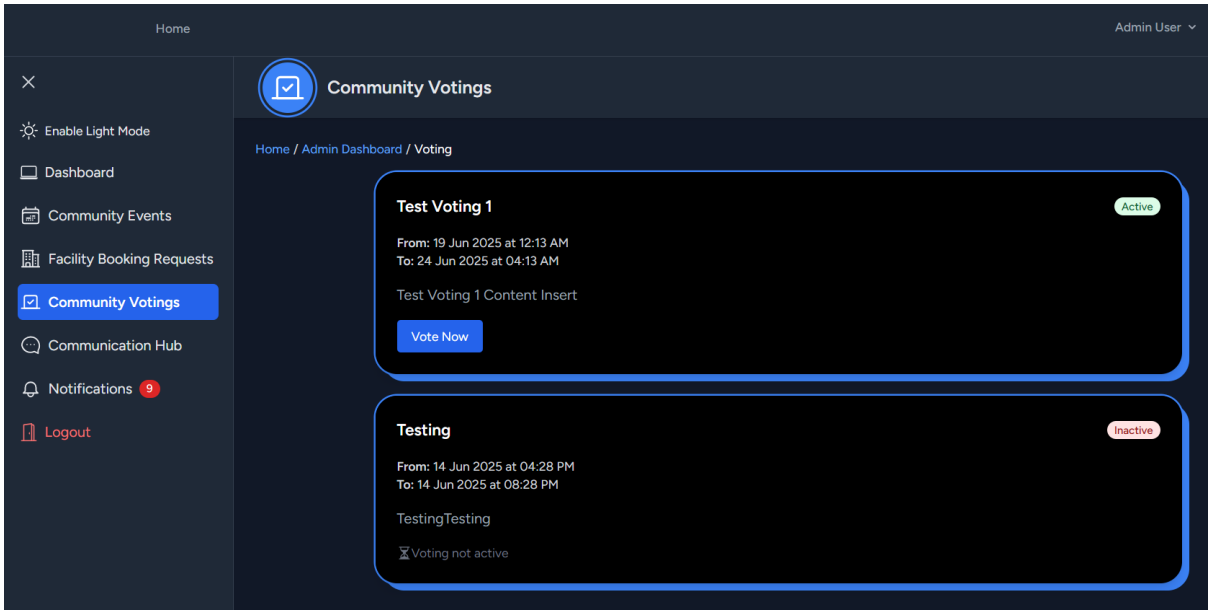


Figure 4.65: Community Voting list page showing available voting sessions for users

Upon selecting a voting session, users are directed to a dedicated voting page that presents the full details of the selected poll with all the available options included. After casting a vote, users receive a visual confirmation that their vote has been submitted. The page also displays a summary of the current vote counts and percentages for transparency. Figure 4.66 shows the detailed voting page, while Figure 4.67 shows the view after voting has been completed, including a “Modify Vote” button for users who wish to change their vote within the active period.

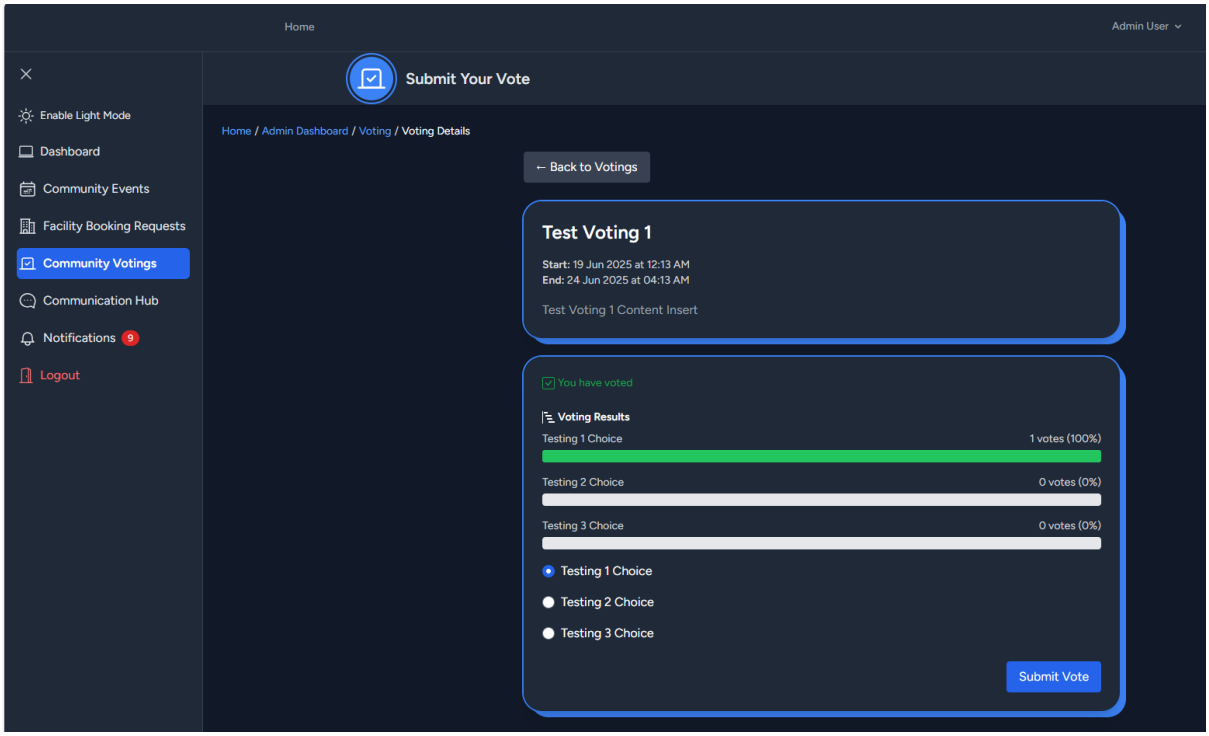


Figure 4.66: Voting submission page

Figure 4.67: User view after vote submission

4.5.6 Communication Hub Module

The Communication Hub Module serves as a centralized discussion space within the Smart Community Platform, enabling users and administrators to engage in topic-driven conversations or community-wide announcements. Threads can be posted, replied to, and moderated, with built-in visibility indicators such as view counts and comment totals. Admins have extended privileges such as posting announcements, editing or deleting threads, and moderating comments with specified reasons.

It should be taken note that, similar to other modules, this section also includes confirmation modals, success notifications, and deletion confirmations when users or admins perform actions. However, to avoid redundancy, these standard modal interactions are not showcased here, as they have already been demonstrated and described in earlier modules such as Event Management and Financial Management.

4.5.6.1 Communication Hub Index Page View

The Communication Hub main page is split into two sections: *Announcements* and *Threads*. Announcements are highlighted messages created by Admins for all users to see, while Threads represent standard discussion topics contributed by either users or Admins. Each card shows essential metadata, such as the author, timestamp, comment count, and view count. Figure 4.68 shows the overall layout of the Communication Hub's landing page.

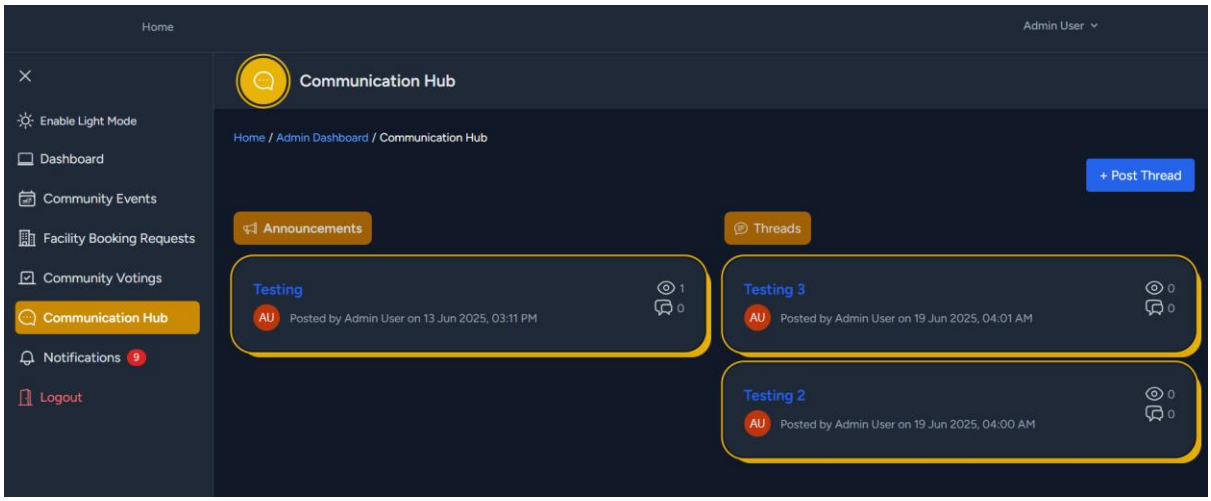


Figure 4.68: Communication Hub index page with Announcements and Threads listed

4.5.6.2 Creating a New Thread

Admins can create new discussion threads or announcements using the “Post Thread” button, while normal Users can only create normal discussion threads. The form includes fields for the thread title, short description, full content, and an optional checkbox to mark the post as an announcement. Figure 4.69 shows the Create Thread interface.

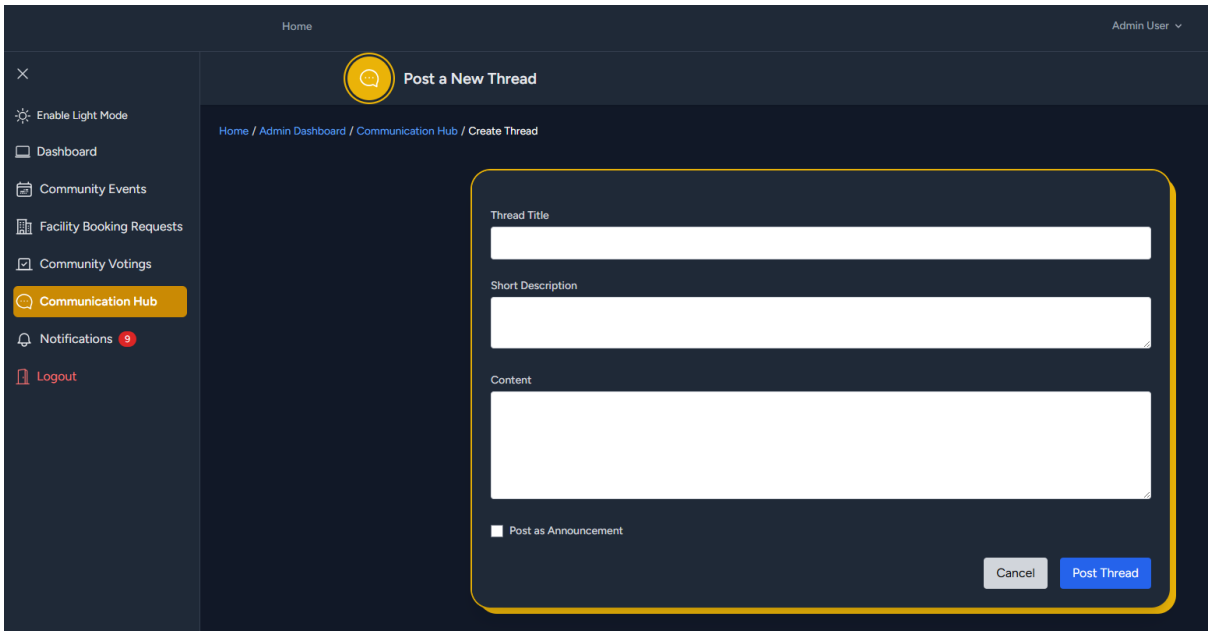


Figure 4.69: Thread creation form

4.5.6.3 Viewing a Thread and Comment Interaction

Clicking on a thread opens the detailed view, which displays the thread content at the top followed by comments and replies. Thread statistics like total views and number of comments are displayed at the top to indicate community engagement. Figure 4.70 illustrates the thread detail view including its engagement statistics and original post content.

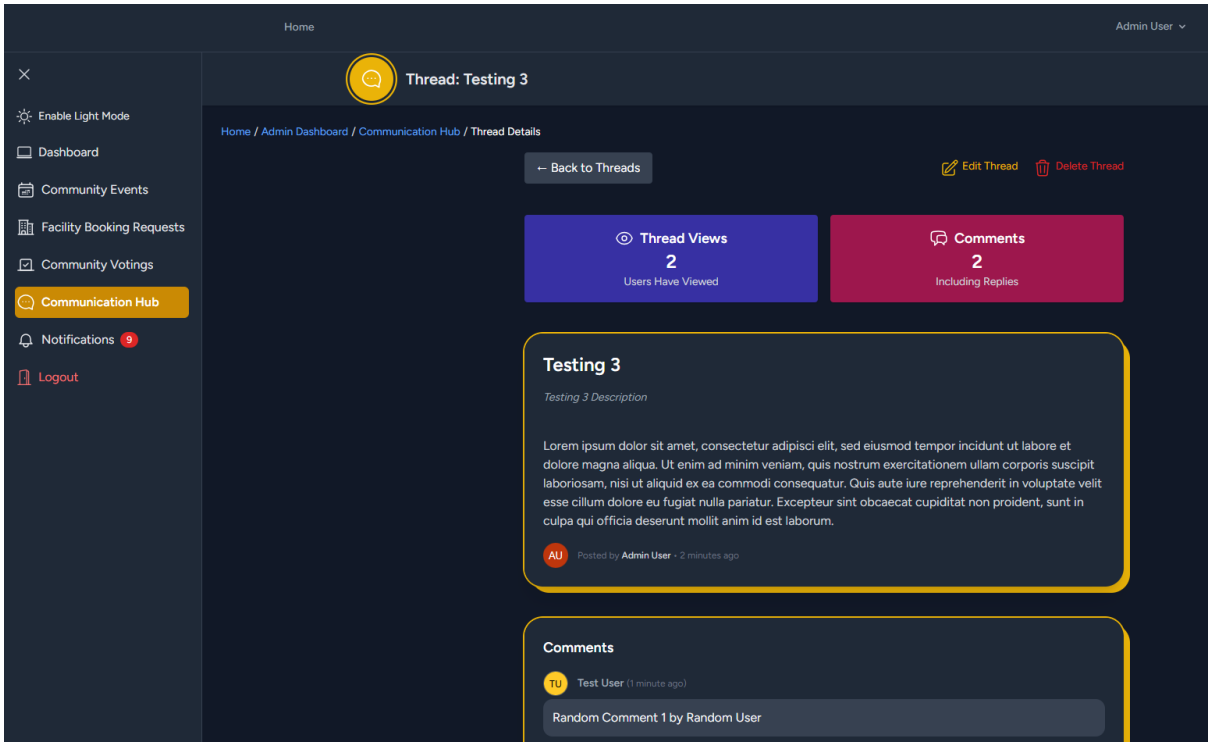


Figure 4.70: Thread detail view

4.5.6.4 **Nested Comments and Moderation**

Users can comment on threads or reply to other comments, supporting nested conversations. Admins are given moderation tools to edit or delete any comment or reply, with a reason input required for moderation actions. Figure 4.71 highlights the comment and reply structure with moderation options.

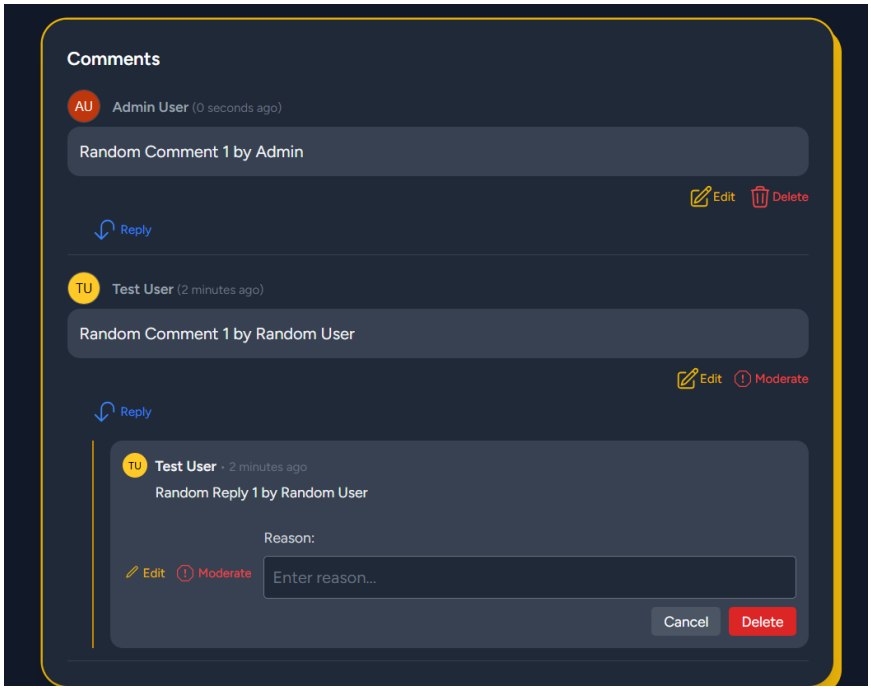


Figure 4.71: Comment and reply section

4.5.6.5 **Leave a Comment Interface**

At the bottom of each thread view is a comment submission box. Users can type in their thoughts and click “Post Comment” to participate in the discussion. Figure 4.70 presents the comment submission interface.

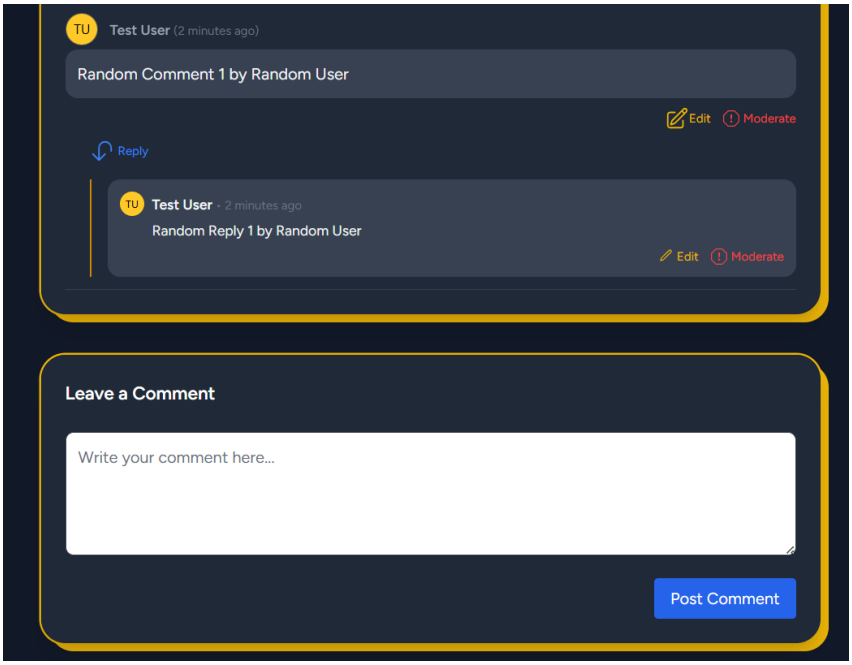


Figure 4.72: Interface for submitting a new comment at the bottom of a thread

4.5.6.6 **Edit Thread Forn**

Admins have the ability to edit any thread they authored. The Edit Thread form mirrors the creation interface, allowing the title, description, and content to be updated. Admins can also change the post's status to an announcement if needed. Figure 4.73 shows the Edit Thread form.

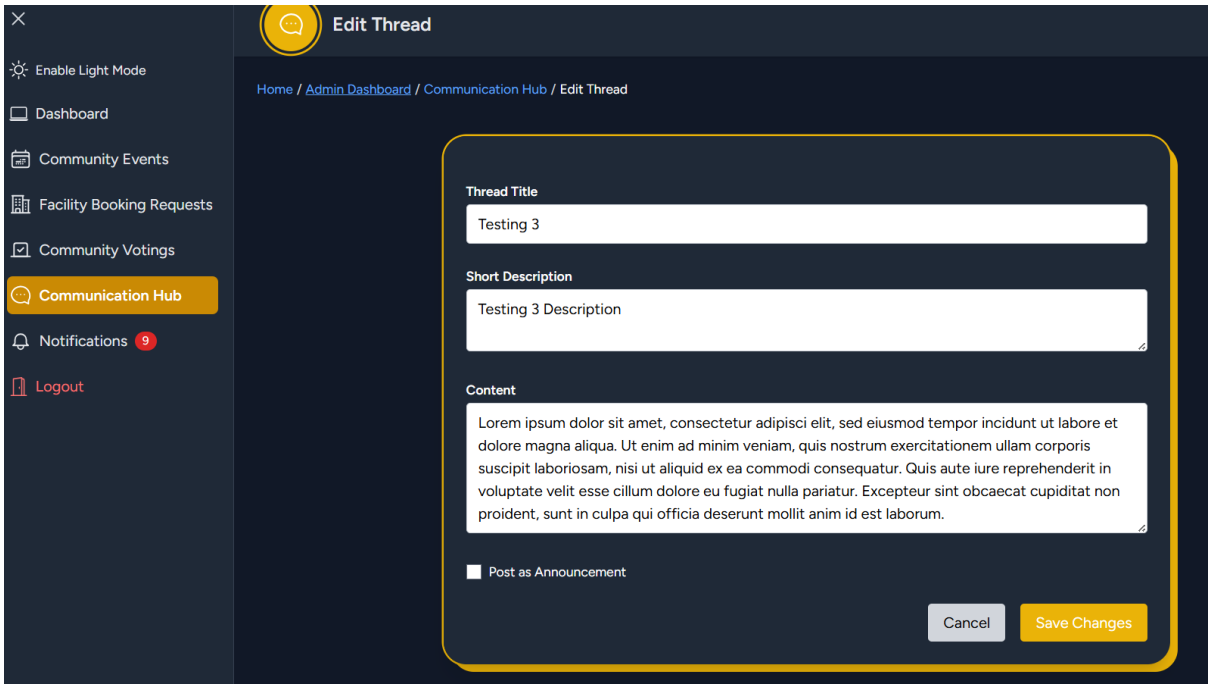


Figure 4.73: Edit Thread Form

4.5.6.7 **Thread Deletion and Moderation Controls**

In addition to standard thread management, the system includes permission-based deletion controls for users and admins. Figure 4.74 illustrates the interface when a user accesses their own thread, where they are provided with the option to delete the post without requiring a reason. This allows users to remove their own content voluntarily and efficiently.

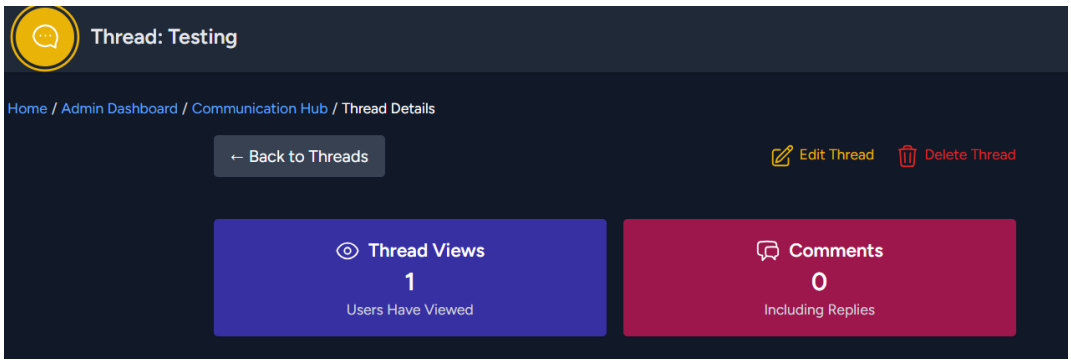


Figure 4.74: Standard thread deletion interface for a user's own thread

For administrative control, the system also supports thread moderation. Figure 4.75 shows the moderation interface available to admins when handling threads created by users. In such cases, a moderation reason must be supplied before deletion can proceed. This reason will later be used to notify the user whose content has been moderated.

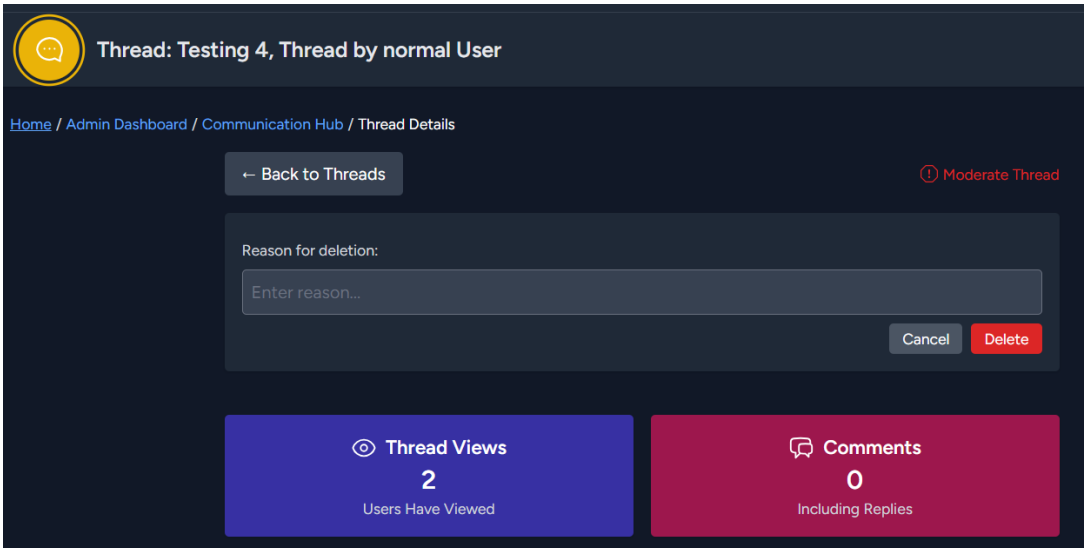


Figure 4.75: Admin moderation interface for deleting threads created by other users with reason input

4.5.7 Notifications Module

The Notifications Module serves as a centralized hub where users are promptly informed of important updates across the Smart Community Platform. It supports automated system notifications triggered by various modules, including voting sessions, event announcements, facility booking approvals or rejections, and communication hub moderation actions. This ensures users remain informed and engaged with ongoing activities in the community.

4.5.7.1 Notifications Page

Notifications are delivered in real-time and can be filtered by status (All, Unread, Read) for better organization. Users also have the option to mark individual messages or all messages as read to manage visibility. The total number of unread notifications is prominently displayed as a red indicator badge on the sidebar, allowing users to quickly recognize pending updates regardless of their current location in the system. Figure 4.76 displays the notification page layout, including filter tabs and status controls, while Figure 4.77 shows the sidebar badge displaying the count of unread notifications.

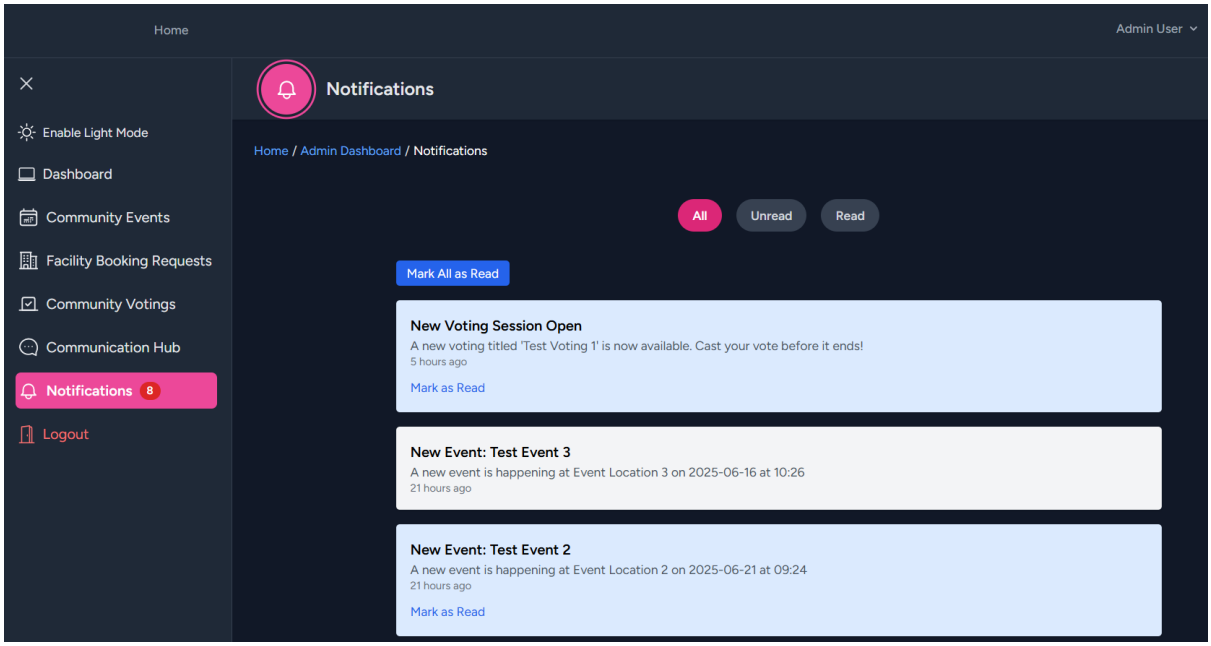


Figure 4.76: Notifications Module displaying system-generated updates with filter and read-status controls

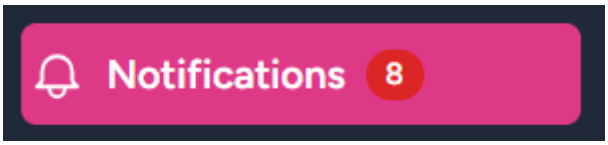


Figure 4.77: Sidebar notification indicator showing the current number of unread notifications

4.5.8 User Management Module

The User Management Module allows administrators to manage user accounts within the system, including assigning roles and managing permissions. Users can either be assigned the Admin or User role, with additional functionality allowing privileged Admins to grant financial management capabilities. This module ensures a clear structure of responsibilities while maintaining the security of high-level accounts through protected status. Only the designated Super Admin can manage other Admin roles.

It should be taken note that, similar to other modules, this section also includes confirmation modals and success messages when user accounts are updated or deleted. However, to avoid redundancy, these interactions are not showcased here as they have already been demonstrated in modules such as Event Management and Financial Management.

4.5.8.1 **User Management Page**

Figure 4.78 presents the main interface of the User Management Module. A searchable and filterable table lists all registered users, displaying profile initials, full name, email address, role, and available actions. Admins are marked with a blue badge and protected from deletion, including the Super Admin with a unique red badge. Users, on the other hand, have deletion and editing privileges enabled. This visual layout ensures Admins can efficiently differentiate roles and manage users accordingly.

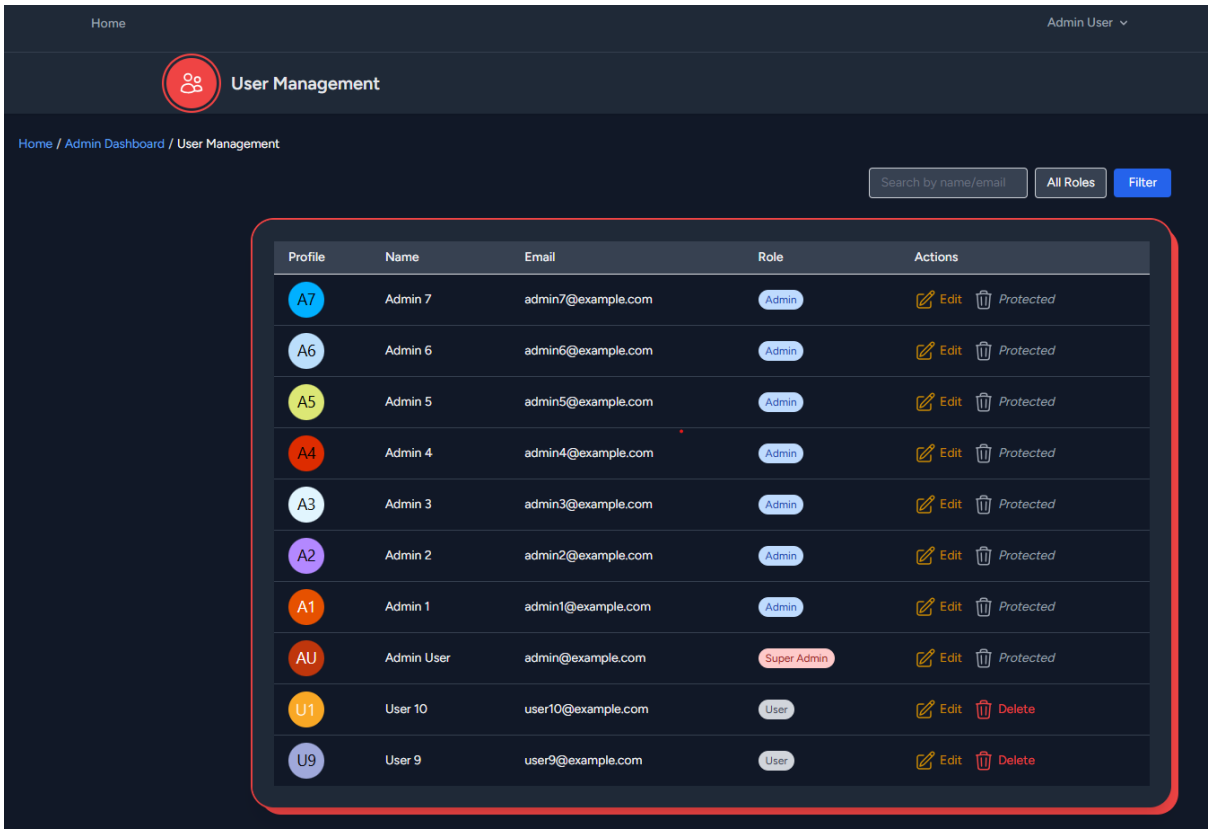


Figure 4.78:User Management Page

4.5.8.2 **Edit User Form**

Figure 4.79 shows the form that appears when an Admin chooses to edit a user's information. It includes fields that cannot be edited and only viewd like the user’s name and email address, as well as a dropdown for selecting their role. This provides a streamlined method for viewing user credentials and permissions.

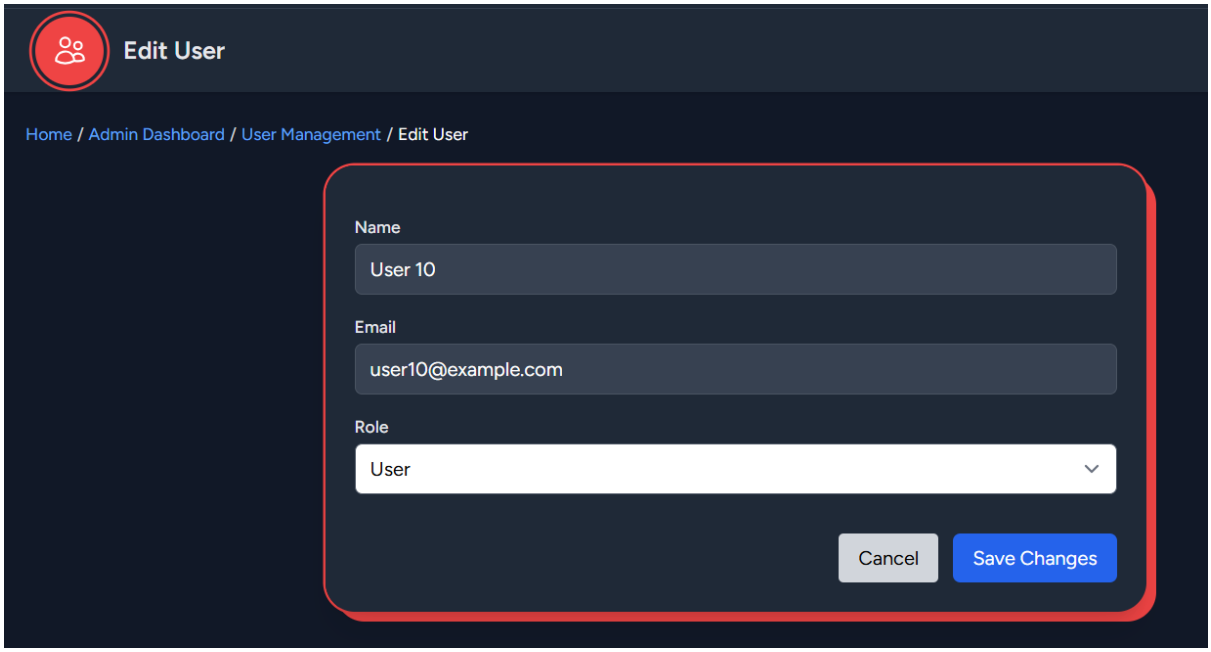


Figure 4.79: Edit User Form

4.5.8.3 **Role Selection Dropdown**

Figure 4.80 focuses on the dropdown field within the Edit User form, showcasing the available role options: "User" and "Admin." This mechanism supports the role-based access system implemented throughout the platform, allowing for flexibility in promoting or demoting users depending on system needs.

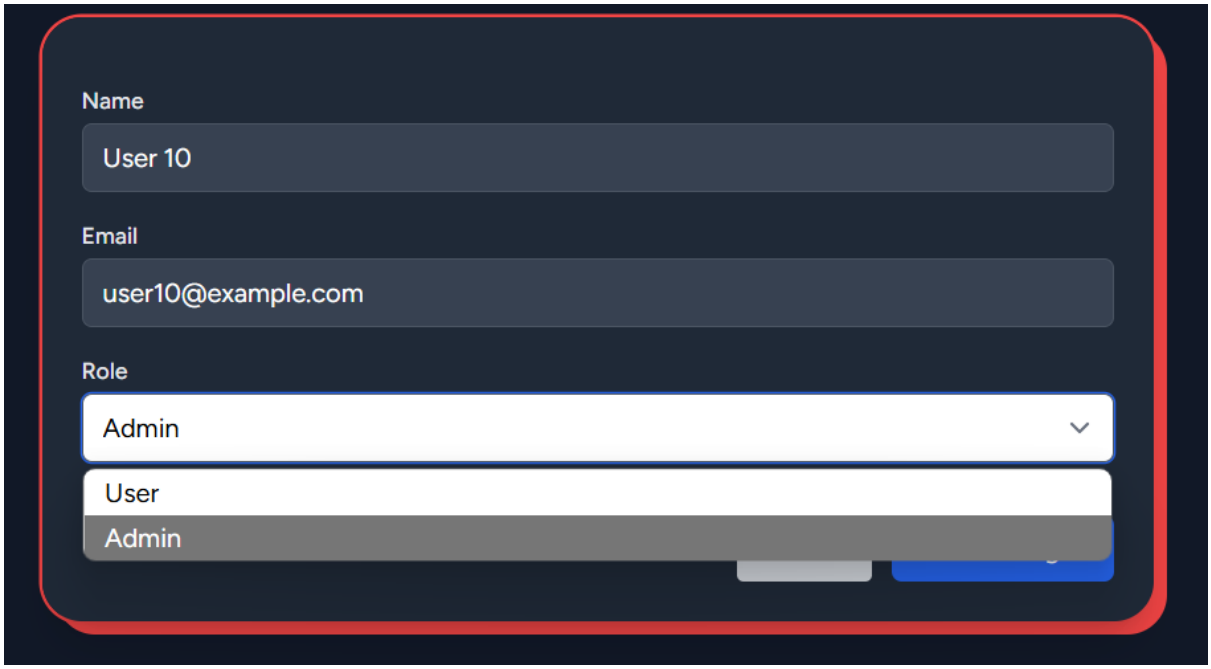


Figure 4.80: Role selection dropdown

4.5.8.4 **Assigning Financial Transaction Privilege**

Figure 4.73 illustrates an extended permission control option that appears only when the "Admin" role is selected. A checkbox labeled "Can Manage Financial Transactions" allows privileged Admins to assign access to financial management functions. This addition sensitive tasks remain under controlled supervision.

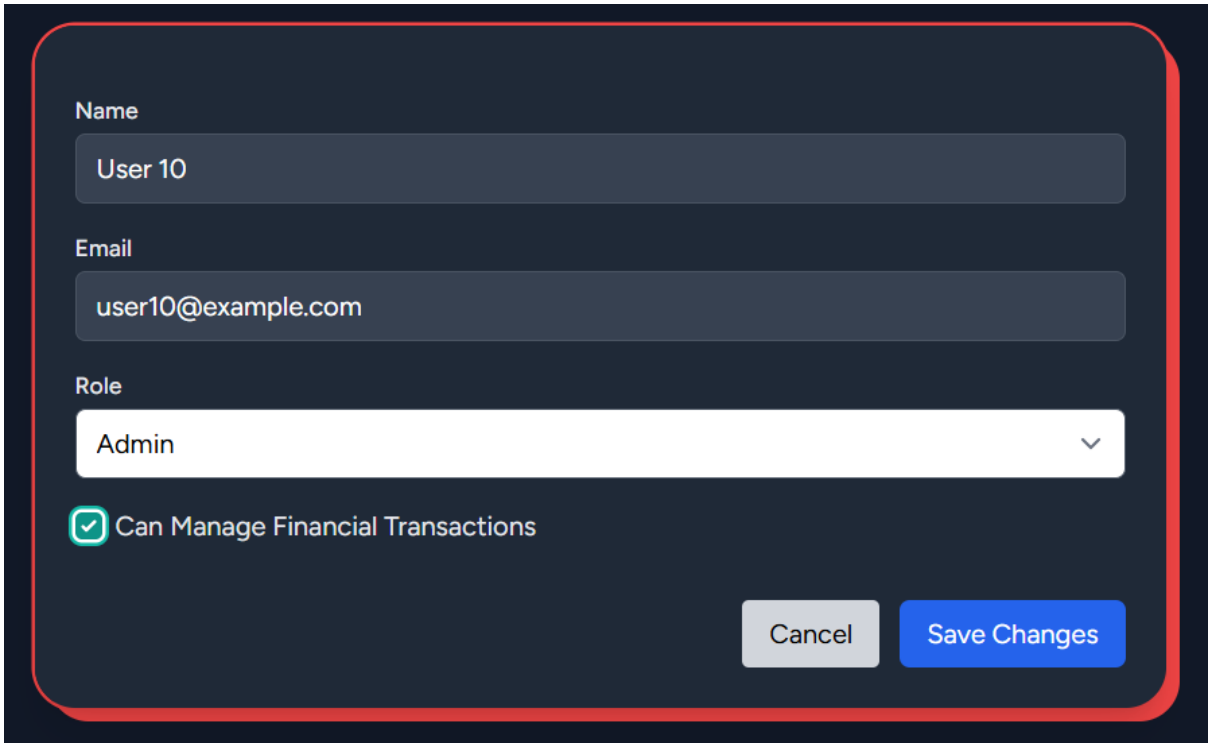


Figure 4.81: Assigning financial transaction privilege

This figure below, Figure 4.82, illustrates the warning modal that appears when an admin attempts to perform restricted actions without having the necessary privileges. While the admin is still able to access and view the Financial Management Module interface, they are prevented from executing specific financial operations. The modal explicitly states, "Access Denied - You do not have permission to access this part of the module," and is accompanied by a countdown-enabled acknowledgment button. This mechanism ensures that only designated admins can manage sensitive financial data.

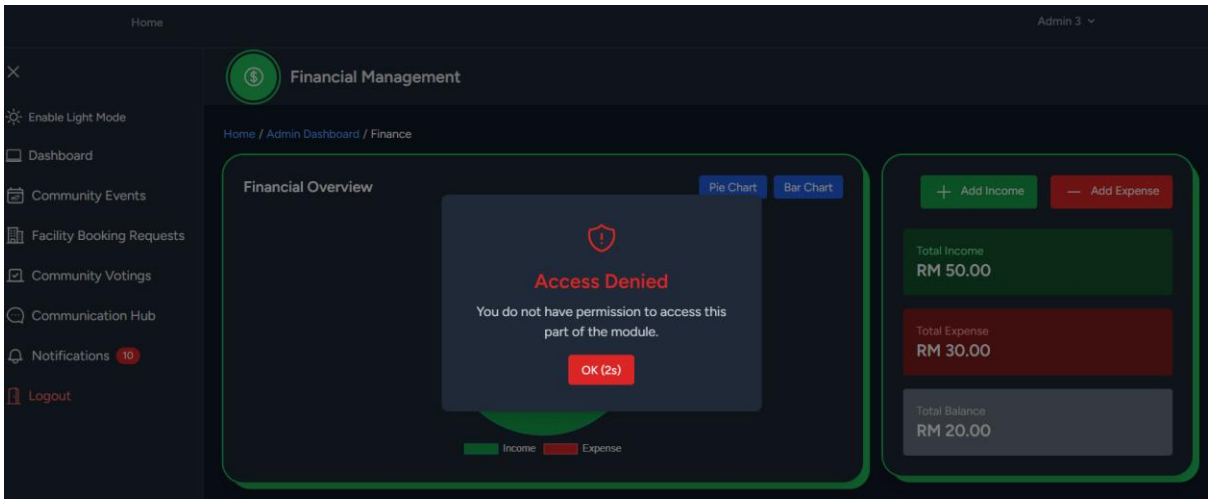


Figure 4.82: Access Denied Modal for Restricted Financial Actions

4.5.9 Profile Management Module

The Profile Management Module allows all users, regardless of role, to manage their personal account settings within the system. Accessible through the dropdown menu at the top right corner, users can update essential profile details, upload a profile picture, and manage their account security. Figure 4.83 illustrates the profile editing interface where users can update their display name, email address, and optionally upload a profile photo.

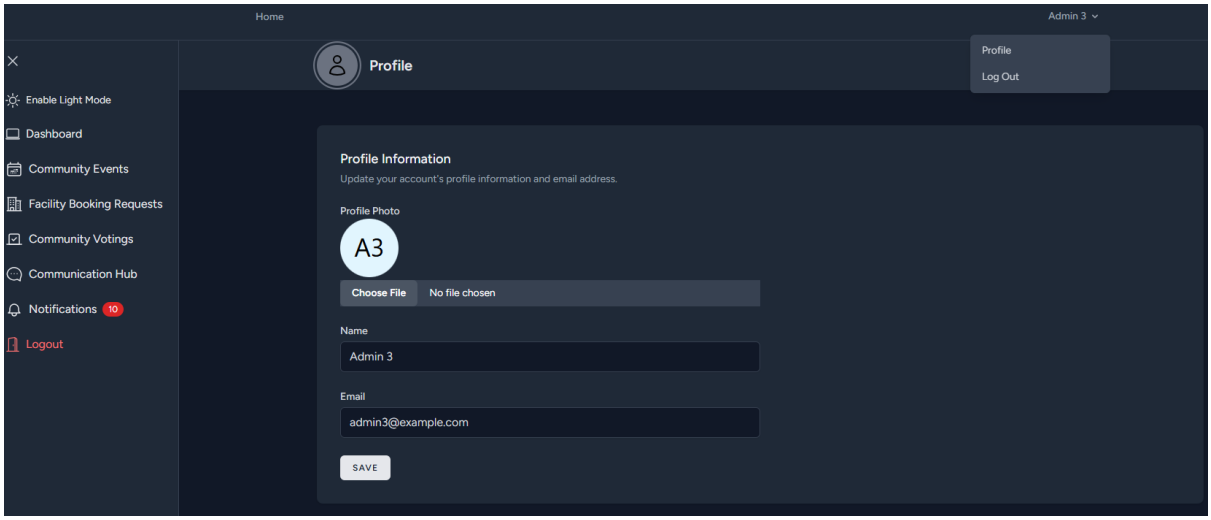


Figure 4.83: Profile information editing interface

Figure 4.84 shows the lower section of the same module, which includes password update fields and the account deletion option. The “Update Password” section requires current password verification before changes can be applied, reinforcing account security. The “Delete Account” section is highlighted with warnings to ensure users are aware that they are aware of the action that they want to take.

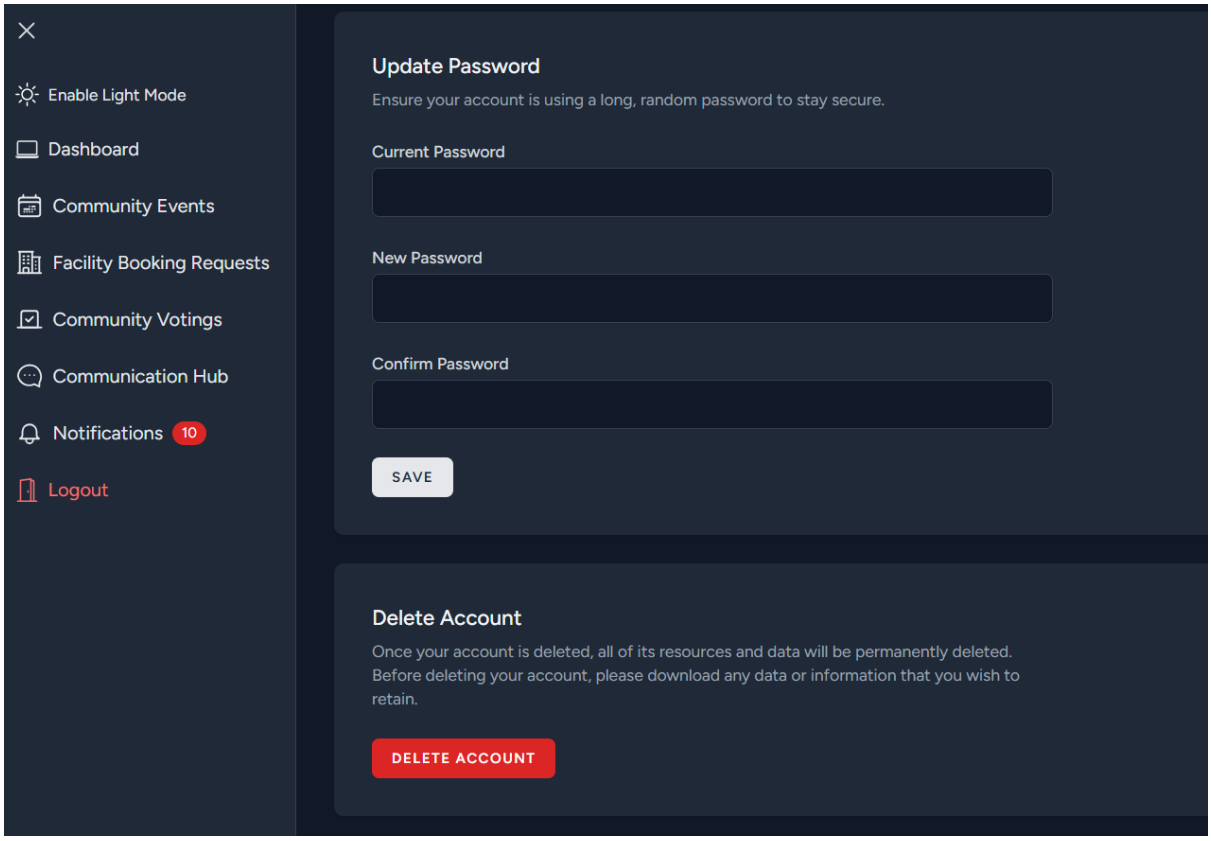


Figure 4.84: Password update and account deletion options

4.6 Summary

This chapter described how the Smart Community Platform for Taman Suria Tropika was developed and how each module works, including features for events, finances, facility booking, voting, communication, notifications, user management, and profile management. The system was built using Laravel (a PHP framework), Tailwind CSS, Alpine.js, and MySQL, with clear role-based access to separate admin and user permissions. Common interface features like confirmation modals and restricted access warnings were used throughout the system but not shown repeatedly to avoid redundancy. Overall, this chapter explained how the system supports easier and more organized community management.

CHAPTER 5: TESTING

5.1 Introduction

This chapter presents the testing activities performed to ensure the Smart Community Platform for Taman Suria Tropika operates correctly and meets user expectations. Testing was conducted based on the system's functional and non-functional requirements, aiming to verify the accuracy, reliability, usability, and stability of the developed modules. Functional testing was used to confirm that each module performs its intended tasks correctly, while non-functional testing assessed performance aspects such as responsiveness and user experience. Unit testing was also conducted using the Pest testing framework integrated into the Laravel environment, which helped validate backend logic and role-based access control. The overall testing process followed the iterative phases of the Rapid Application Development (RAD) methodology adopted in this project.

5.2 Functional Testing

Functional testing ensures that each module performs its required functions as defined in the system specification. Testing was conducted across both admin and user roles to simulate real usage scenarios.

5.2.1 System Testing

System testing was conducted after all modules of the Smart Community Platform were fully integrated. The goal was to verify that the entire system works in accordance with the defined requirements and that interactions between different modules occur seamlessly. Testing scenarios were created for both admin and user roles, covering all major workflows such as creating events, managing facility bookings, tracking finances, participating in votes, and interacting through the communication hub.

5.2.1.1 System Testing: Registration Module

Table 5.1: System Testing: Registration Module

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
R-AUTH-ST01	Registration page is accessible	1. Navigate to registration page	Registration form is displayed	Form loaded correctly	Pass
R-AUTH-ST02	User registers with valid details	1. Fill in name, email, password 2. Submit the form	Account is created and redirected to dashboard	User redirected after registration	Pass
R-AUTH-ST03	Registration fails with invalid input	1. Enter mismatched passwords 2. Submit form	Error message is displayed	Validation error shown	Pass
R-AUTH-ST04	Duplicate email is rejected	1. Register with existing email 2. Submit form	Error message shown, account not created	Duplicate email detected	Pass

5.2.1.2 System Testing: Login Module

Table 5.2: System Testing: Login Module

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
L-AUTH-ST01	Login page is accessible	1. Navigate to login page	Login form is displayed	Login form visible	Pass
L-AUTH-ST02	User logs in with valid credentials	1. Enter valid email and password 2. Click Login	Redirected to user or admin dashboard	Correct dashboard loaded	Pass
L-AUTH-ST03	Login fails with invalid credentials	1. Enter incorrect password 2. Submit form	Error message shown, login blocked	Invalid login prevented	Pass
L-AUTH-ST04	Login redirects based on role	1. Login as admin and get sent to the Admin Dashboard 2. Login as user and get sent to the Home page	Correct redirect based on role	Role-based access confirmed	Pass
L-AUTH-ST05	Logout works correctly	1. Click Logout button	User is logged out and redirected to login	Session ended, login screen shown	Pass

5.2.1.3 System Testing: Event Management Module (Admin)

Table 5.3: System Testing: Event Management Module (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
EM-ST01	Admin creates a new event	1. Login as Admin 2. Navigate to Event Management 3. Click "Create Event" 4. Fill in all required fields and submit	Event is saved and appears in the event list	Event listed with correct info	Pass
EM-ST02	Admin edits an existing event	1. Go to Event list 2. Click "Edit" on a listed event 3. Modify date or title and submit	Changes are updated and reflected in the event list	Event info updates correctly	Pass
EM-ST03	Admin deletes an event	1. From Event list, click "Delete"	Event is removed from the list	Event is no longer visible	Pass

		2. Confirm deletion in modal			
EM-ST04	Confirmation modal appears before saving	1. Fill in event form 2. Click submit	Confirmation modal is triggered	Modal appears with options	Pass
EM-ST05	Success message after event operation	1. Create or edit an event 2. Submit form	Success confirmation modal is shown	Success modal displayed	Pass

5.2.1.4 System Testing: Event Viewing Module (User)

Table 5.4: System Testing: Event Viewing Module (User)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
EV-ST01	User views upcoming events	1. Login as User 2. Navigate to Events page	Upcoming events are listed with images and titles	Events shown in card format	Pass
EV-ST02	User opens event details	1. Click on an event card 2. Scroll through event info	Event image and full description are displayed	Event info correctly shown	Pass

EV-ST03	User opens fullscreen modal for event poster	1. Click event image	Fullscreen modal opens	Poster shown in enlarged view	Pass
EV-ST04	User navigates between Upcoming and Past tabs	1. Click between “Upcoming” and “Past” tabs	Respective events shown based on date	Correct filtering applied	Pass

5.2.1.5 System Testing: Voting Management Module (Admin)

Table 5.5: System Testing: Voting Management Module (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
VM-ST01	Admin creates a new voting session	1. Login as Admin 2. Go to Voting Management 3. Click "Create Voting" 4. Fill in details and submit	Voting session appears in the admin list	Voting listed correctly	Pass
VM-ST02	Admin views voting session details	1. Click on a listed voting session	Voting info and options are displayed	Details shown with correct data	Pass

VM-ST03	Admin adds or updates voting choices	1. On the voting details page, add/edit options 2. Click submit	Choices are updated and saved	New/updated options displayed	Pass
VM-ST04	Admin deletes a voting session	1. From list, click "Delete" 2. Confirm deletion in modal	Voting session is removed	Voting removed from admin list	Pass
VM-ST05	Voting results chart is displayed	1. View a session with recorded votes 2. Scroll to results section	Results displayed as a chart	Chart accurately reflects votes	Pass

5.2.1.6 System Testing: Voting Participation Module (User)

Table 5.6: System Testing: Voting Participation Module (User)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
VP-ST01	User views available votings	1. Login as User 2. Navigate to Voting page	List of active votings displayed	Voting sessions listed	Pass

VP-ST02	User opens voting detail page	1. Click on a voting session	Voting title, description, and options shown	Info and choices displayed	Pass
VP-ST03	User submits a vote	1. Select an option 2. Click submit 3. Confirm in modal	Vote is saved and user redirected to result	Vote recorded and confirmation shown	Pass
VP-ST04	User views vote result after submission	1. After voting, view result page	Result chart is shown with current votes	Chart visible with updated counts	Pass
VP-ST05	Prevent voting again after submission	1. Attempt to vote again on same session	Voting option disabled or replaced with message	Repeat voting prevented	Pass

5.2.1.7 System Testing: Facility Booking Module (User)

Table 5.7: System Testing: Facility Booking Module (User)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
FB-UST01	User views booking dashboard	1. Login as User	Booking dashboard and calendar are displayed	Facilities and dates visible	Pass

		2. Go to Facility Booking page			
FB-UST02	User submits a new booking request	1. Click "New Booking" 2. Fill in form (facility, date, time, purpose) 3. Submit and confirm in modal	Request saved and appears in user's request list	Booking request saved successfully	Pass
FB-UST03	User views details of submitted request	1. Click on a submitted booking entry	Full booking info is displayed	Correct details shown	Pass
FB-UST04	User deletes their pending booking	1. Click "Delete" on a pending booking 2. Confirm deletion	Booking is removed from the list	Booking deleted successfully	Pass
FB-UST05	User views status (approved/rejected)	1. View submitted requests	Status indicators (e.g., Pending, Approved, Rejected) shown	Status updates shown correctly	Pass

5.2.1.8 System Testing: Facility Booking Management Module (Admin)

Table 5.8: System Testing: Facility Booking Management Module (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
FBM-ST01	Admin views all booking requests	1. Login as Admin 2. Navigate to Facility Booking Management	List of all user-submitted bookings shown	All requests listed with status	Pass
FBM-ST02	Admin opens request details	1. Click on a specific booking	Full info including requester name, facility, date, time shown	Correct data displayed	Pass
FBM-ST03	Admin approves a request	1. Click "Approve" on a request and add an optional reason 2. Confirm action in modal	Request status updated to "Approved" and user notified	Booking approved successfully	Pass
FBM-ST04	Admin rejects a request	1. Click "Reject" on a request and add an optional reason 2. Confirm in modal	Request status set to "Rejected" and user notified	Booking rejected successfully	Pass
FBM-ST05	Admin checks calendar view	1. Scroll to calendar section	Approved bookings displayed on calendar by date/time	Visual calendar updates correctly	Pass

5.2.1.9 System Testing: Financial Management Module (Admin)

Table 5.9: System Testing: Financial Management Module (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
FM-ST01	Admin views the financial dashboard	1. Login as Admin 2. Navigate to Financial Management	Summary boxes (Income, Expense, Balance) and pie chart are displayed	Dashboard elements visible	Pass
FM-ST02	Admin accesses add income form	1. Click "Add Income" 2. View income form	Income form appears with fields	Form loaded correctly	Pass
FM-ST03	Admin adds a new income record	1. Fill in income form 2. Confirm in modal 3. Submit	Record is saved and shown in transaction history	Income listed with correct info	Pass
FM-ST04	Admin accesses add expense form	1. Click "Add Expense" 2. View expense form	Expense form appears with fields	Form loaded correctly	Pass
FM-ST05	Admin adds a new expense record	1. Fill in expense form 2. Confirm in modal	Record is saved and reflected in history	Expense listed with correct info	Pass

		3. Submit			
FM-ST06	Admin views full transaction history	1. Click on “Transaction History” 2. Apply filters or search	All transactions are listed and filterable	History shows all relevant records	Pass
FM-ST07	Admin edits an existing transaction	1. Click "Edit" on a transaction 2. Modify and submit	Changes saved and reflected in list	Updated info shown correctly	Pass
FM-ST08	Admin deletes a transaction	1. Click "Delete" 2. Confirm in modal	Record removed from history list	Transaction no longer listed	Pass
FM-ST09	Success modal appears after submission	1. Submit income or expense form	Confirmation modal shown	Modal confirms success	Pass
FM-ST010	Admin uses export function	1. Use Export CSV/Excel button in history view	Download starts with filtered results	File downloaded correctly	Pass

5.2.1.10 System Testing: Communication Hub Module, Thread and Comment Interaction (User)

Table 5.10: System Testing: Communication Hub Module, Thread and Comment Interaction (User)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
CH-UST01	User views the Communication Hub	1. Login as User 2. Navigate to Communication Hub	Announcement and thread list displayed	Page loads with all the items, as expected	Pass
CH-UST02	User creates a new thread	1. Click “New Thread” 2. Fill in title and content 3. Submit	Thread is saved and displayed in thread list	Thread appears correctly	Pass
CH-UST03	User opens a thread	1. Click on a thread title	Thread details and comments are shown	Thread opened with content and replies	Pass
CH-UST04	User posts a comment	1. In thread view, enter comment 2. Click "Post"	Comment saved and shown below thread	Comment visible immediately	Pass

CH-UST05	User replies to a comment	1. Click "Reply" under a comment 2. Enter reply and submit	Reply shown nested under parent comment	Reply formatted correctly	Pass
CH-UST06	User edits own comment	1. Click “Edit” on a comment 2. Modify and save	Comment updated with new content	Changes reflected correctly	Pass
CH-UST07	User deletes own comment	1. Click "Delete" on own comment 2. Confirm deletion	Comment removed from thread	Comment deleted	Pass

5.2.1.11 System Testing: Communication Hub Module, Moderation and Notification (Admin)

Table 5.11: System Testing: Communication Hub Module, Moderation and Notification (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
CH-ST01	Admin moderates a user’s comment	1. Click "Moderate" on a comment 2. Enter reason and confirm	Comment is removed, user notified	As expected	Pass

CH-ST02	Admin moderates a user's thread	1. Click "Moderate" on a thread 2. Enter reason and confirm	Thread removed, user receives notification	As expected	Pass
CH-ST03	Admin views all announcements and threads	1. Navigate to Communication Hub (Admin view)	Full list of announcements and threads shown	As expected	Pass
CH-ST04	Admin posts an announcement	1. Click "Create Announcement" 2. Fill details and submit	Announcement displayed in its section	As expected	Pass
CH-ST05	Admin receives notifications of reports or feedback	1. Trigger moderation action or user post 2. Open notification panel	Notification appears for moderation or new post	Notification displayed and viewable	Pass

5.2.1.12 System Testing: User Management Module (Admin)

Table 5.12: System Testing: User Management Module (Admin)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail

UM-ST01	Admin views all users	1. Login as Admin 2. Go to User Management	List of all registered users is displayed	Users listed with edit/delete buttons	Pass
UM-ST02	Admin edits user information	1. Click “Edit” on a user 2. Modify name/email 3. Submit changes	User info is updated in the system	New details displayed	Pass
UM-ST03	Admin deletes a user	1. Click “Delete” on a user 2. Confirm deletion	User account is removed	User no longer appears in list	Pass
UM-ST04	Admin toggles user role, which are either User or Admin	1. Click role toggle on a user 2. Confirm action	User role updated accordingly	New role shown in list	Pass
UM-ST05	Admin grants/revokes facility access	1. Check/uncheck “Can Manage Facility” box 2. Save	Access permission updated	Checkbox reflects updated access	Pass

5.2.1.13 System Testing: User Profile Management Module (User)

Table 5.13: System Testing: User Profile Management Module (User)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail

UPM-ST01	User edits profile information	1. Login as User 2. Go to Profile 3. Edit name/email and save	Profile updated successfully	New info saved	Pass
UPM-ST02	User changes password	1. Go to "Change Password" 2. Enter old and new passwords 3. Submit	Password is changed and user notified	Password updated	Pass
UPM-ST03	User uploads profile picture	1. Click “Upload Photo” 2. Select image 3. Save changes	Profile photo displayed	New image appears in UI	Pass
UPM-ST04	User deletes own account	1. Go to “Delete Account” section 2. Confirm deletion	Account deleted and redirected to login	User removed from system	Pass

5.2.1.14 System Testing: Notification System

Table 5.14: System Testing: Notification System

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
N-ST01	Notification appears after admin approves a booking	1. User submits a facility booking 2. Admin approves the request	User receives notification about approval	Notification appears in user's notification panel	Pass
N-ST02	Notification appears after admin rejects a booking	1. User submits a facility booking 2. Admin rejects the request	User receives rejection notification	Notification content is accurate	Pass
N-ST03	Notification sent after vote submission	1. User votes in a voting session	User is notified that vote was submitted	Vote confirmation notification shown	Pass
N-ST04	Notification sent after admin moderates a comment	1. Admin deletes a comment with a reason	User receives a moderation notification	Reason is included in notification	Pass

N-ST05	Notification panel shows all recent messages	1. Click notification bell/icon 2. View the dropdown list	All recent, unread notifications listed	Messages are sorted and visible	Pass
N-ST06	Marking a notification as read works correctly	1. Click on a notification 2. Return to notification panel	Notification status changes to read	Status reflected visually	Pass
N-ST07	Admin receives notification after new voting activity	1. A vote is cast 2. Admin views their panel	Notification alert shown on admin dashboard	Admin notified of voting activity	Pass

5.2.2 Unit Testing

Unit testing was carried out continuously during the development process to verify that each individual method and module functioned as intended in isolation. These tests focused on specific functionalities such as data handling, controller logic, and access control without relying on the full system environment. The Pest testing framework, which integrates seamlessly with the Laravel PHP environment, was selected for its simple and expressive syntax, making it easier to write and manage automated test cases efficiently.

5.2.2.1 **Unit Testing: Event Management Module (Admin)**

The following test cases validate the core functionalities of the Event Management module, including creating, updating, deleting, and viewing events, handling images, and notifying users accordingly.

Table 5.15: Unit Testing: Event Management Module (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
EM-UT01	Verify that admin can view the event listing.	Admin logs in and accesses the event management dashboard.	A list of events is displayed.	As expected	Pass
EM-UT02	Verify that admin can access the event creation page.	Admin clicks the 'Create Event' button from the dashboard.	Event creation form is displayed.	As expected	Pass
EM-UT03	Verify storing a valid event and notifying users.	Admin submits valid event details with optional image.	Event is saved, image uploaded, notifications sent.	As expected	Pass
EM-UT04	Ensure validation errors appear for missing required fields.	Admin submits form with empty fields.	Validation error messages are shown.	As expected	Pass
EM-UT05	Verify that admin can access the event edit form.	Admin selects an existing event and chooses to edit it.	Edit form loads with event details.	As expected	Pass

EM-UT06	Verify that admin can update event details and notify users.	Admin updates the event with new information and a new image.	Event updated, old image deleted, new image saved, users notified.	As expected	Pass
EM-UT07	Verify that admin can delete an event and notify users.	Admin deletes an event with an associated image.	Event deleted, image removed, users notified of cancellation.	As expected	Pass
EM-UT08	Ensure unauthorized user cannot update an event.	A normal user attempts to update an event.	Action is forbidden.	As expected	Pass
EM-UT09	Ensure unauthorized user cannot delete an event.	A normal user attempts to delete an event.	Action is forbidden.	As expected	Pass

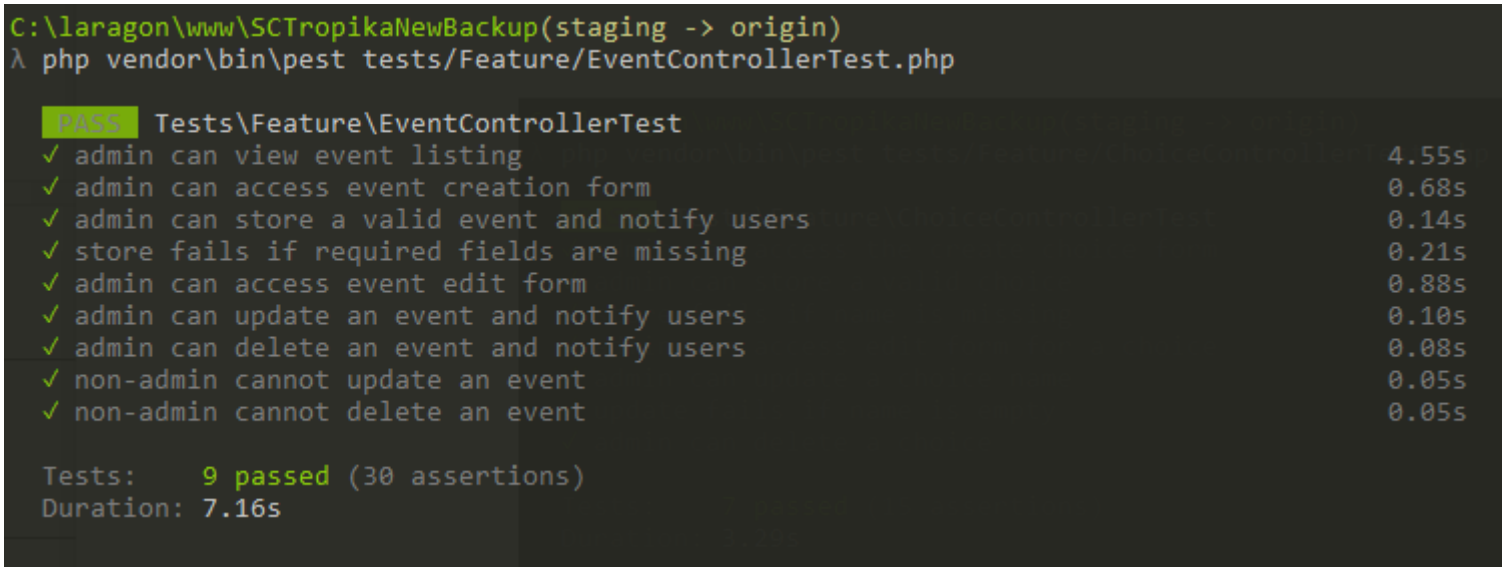


Figure 5.1: Unit Testing using Pest: Event Management Module (Admin)

5.2.2.2 Unit Testing: Facility Booking Management Module (Admin)

The test cases below evaluate the admin's ability to manage facility bookings, including approval, rejection, viewing data, and calendar JSON output.

Table 5.16: Unit Testing: Facility Booking Management Module (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass-Fail

FB-UT01	Verify admin can approve a facility booking	Booking status = 'pending'; Admin logged in; Note submitted	Status changes to 'approved', admin note saved, user notified.	As expected	Pass
FB-UT02	Verify admin can reject a facility booking	Booking status = 'pending'; Admin logged in; Note submitted	Status changes to 'rejected', admin note saved, user notified.	As expected	Pass
FB-UT03	Verify admin can view list of all bookings	Admin logged in	Booking list displayed with pagination.	As expected	Pass
FB-UT04	Verify admin can view detailed booking info	Admin logged in; Booking exists	Correct booking data shown in detail view.	As expected	Pass
FB-UT05	Verify admin can fetch booking calendar data	Admin logged in; Approved bookings exist	Returns valid JSON with booking structure.	As expected	Pass

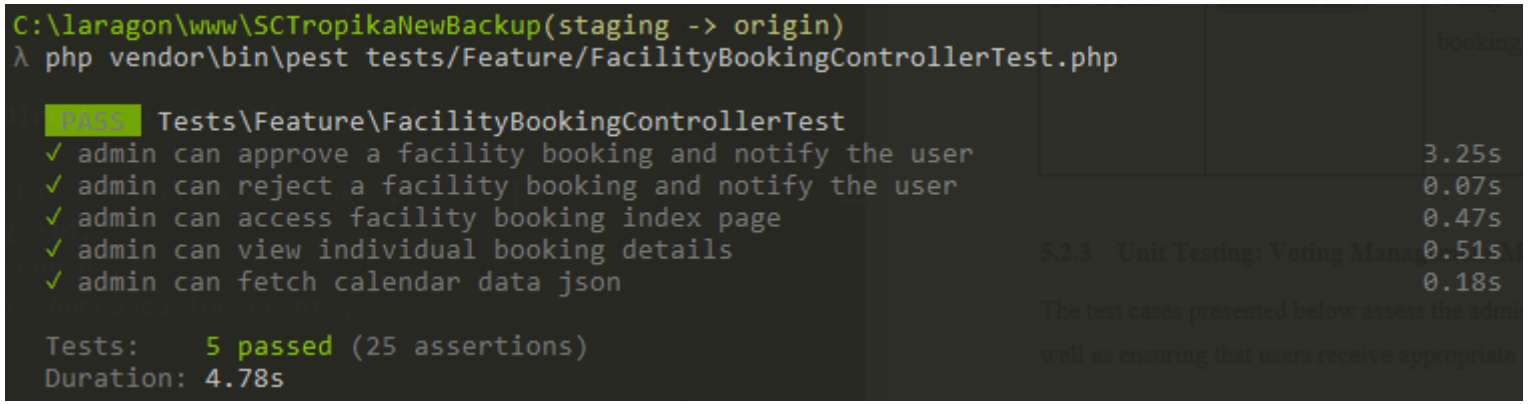


Figure 5.2: Unit Testing using Pest: Facility Booking Management Module (Admin)

5.2.2.3 Unit Testing: Voting Management Module (Admin)

The test cases presented below assess the admin's ability to manage voting sessions, including creating, updating, ending, and deleting votings, as well as ensuring that users receive appropriate notifications and that voting choices are correctly handled.

Table 5.17: Unit Testing: Voting Management Module (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
VM-UT01	Verify that voting status updates based on the current date and time.	Admin views the voting list while voting is ongoing or has ended.	Ongoing voting becomes active; past voting becomes inactive.	As expected	Pass

VM-UT02	Verify that an admin can access the voting creation page.	Admin logs in and opens the voting creation form.	Voting creation form loads successfully.	As expected	Pass
VM-UT03	Verify storing a new voting and sending notifications to users.	Valid title, date, and multiple choices submitted by admin.	Voting saved in DB, choices created, all users notified.	As expected	Pass
VM-UT04	Ensure that invalid input is rejected during voting creation.	Empty or incomplete form fields submitted by admin.	Validation error messages are shown.	As expected	Pass
VM-UT05	Verify that voting details and vote results are displayed.	Admin opens a voting that already has user votes.	Vote counts and percentage per choice are shown.	As expected	Pass
VM-UT06	Verify that an admin can access the voting edit form.	Admin selects a voting and opens its edit page.	Edit form displays current voting data.	As expected	Pass
VM-UT07	Verify updating voting and replacing choices.	Existing voting, updated data, and new choices submitted by admin.	Voting data updated, previous choices deleted, new choices saved.	As expected	Pass
VM-UT08	Ensure validation error when end date is earlier than start date.	Admin provides end date that is before the start date.	System shows date validation error.	As expected	Pass
VM-UT09	Verify that admin can delete a voting.	Admin selects a voting and clicks delete.	Voting is removed from the database.	As expected	Pass

VM-UT10	Verify ending a voting session and notifying users of results.	Admin ends a voting with at least one vote cast.	Voting status becomes inactive, users notified of the winning choice.	As expected	Pass

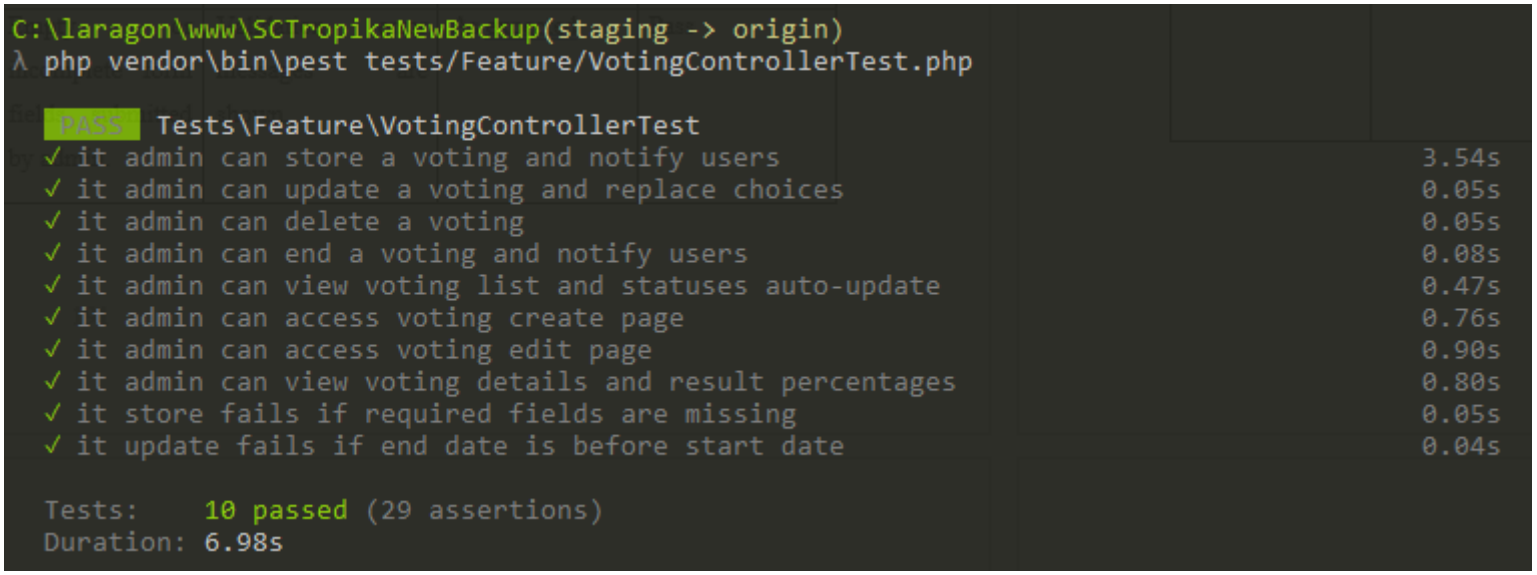


Figure 5.3: Unit Testing using Pest: Voting Management Module (Admin)

5.2.2.4 Unit Testing: Financial Management Module (Admin)

The following test cases validate the core functionalities of the Financial Management module, including storing, updating, deleting transactions, permission-based access, exporting records, and viewing chart summaries.

Table 5.18: Unit Testing: Financial Management Module (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
FM-UT01	Verify that an authorized admin can view the finance dashboard.	Admin logs in and visits the Finanacial Management page.	Dashboard loads with transactions and summaries.	As expected	Pass
FM-UT02	Verify authorized admin can access create page for income.	Admin with permission visits “Add Income” page.	Form loads for income entry.	As expected	Pass
FM-UT03	Verify authorized admin can access create page for expense.	Admin with permission visits “Add Expense” page.	Form loads for expense entry.	As expected	Pass
FM-UT04	Ensure unauthorized admin is denied access.	Admin without permission visits “Add Income” page.	Redirected with error message.	As expected	Pass
FM-UT05	Ensure invalid type returns 404 error.	Admin with permission visits “Add Expense” page.	Redirected with error message.	As expected	Pass

FM-UT06	Verify admin can store a new income transaction.	Valid income form submitted.	Transaction saved, redirected with success.	As expected	Pass
FM-UT07	Verify admin can store a new expense transaction.	Valid expense form submitted.	Transaction saved, redirected with success.	As expected	Pass
FM-UT08	Ensure store fails with invalid data.	Only 'type' field provided.	Validation errors on missing fields.	As expected	Pass
FM-UT09	Verify admin can update an existing transaction.	Valid update form with new category/description.	Record updated in DB.	As expected	Pass
FM-UT10	Verify admin can delete a transaction.	Authorized admin deletes a transaction.	Record removed from DB.	As expected	Pass
FM-UT11	Ensure unauthorized admin cannot delete transactions.	Unauthorized admin attempts deletion.	Redirect with error, record remains.	As expected	Pass
FM-UT12	Verify admin can view financial chart data.	Two transactions created with same category, admin fetches JSON.	JSON includes correct summary.	As expected	Pass
FM-UT13	Verify authorized admin can export data.	Admin with permission triggers export route.	Excel file returned with correct headers.	As expected	Pass
FM-UT14	Ensure unauthorized admin cannot export.	Unauthorized admin accesses export route.	Redirected with error message.	As expected	Pass

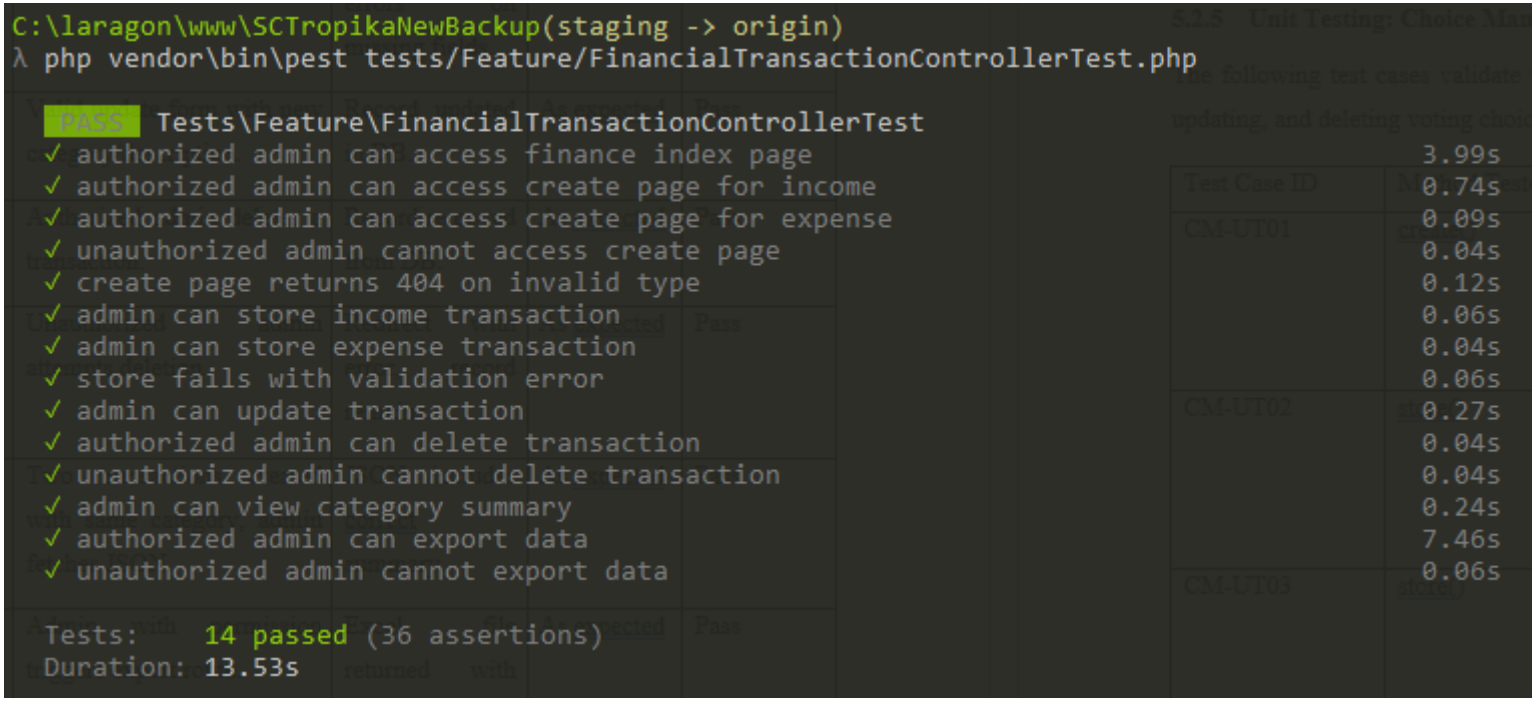


Figure 5.4: Unit Testing using Pest: Financial Management Module (Admin)

5.2.2.5 Unit Testing: Choice Management (Admin)

The following test cases validate the key functionalities of the Choice Management within a voting session. These include creating, editing, updating, and deleting voting choices, as well as enforcing validation rules.

Table 5.19: Unit Testing: Choice Management (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
CM-UT01	Verify that admin can access the create choice form.	Admin selects a voting session and clicks 'Add Choice'.	Create form is displayed with input field.	As expected	Pass
CM-UT02	Verify storing a valid choice linked to a voting session.	Admin submits a new choice with valid name.	Choice is saved and linked to the voting.	As expected	Pass
CM-UT03	Ensure validation fails when choice name is missing.	Admin submits form without entering a name.	Validation error is displayed for name field.	As expected	Pass
CM-UT04	Verify that admin can access the edit form for a choice.	Admin selects a choice and clicks 'Edit'.	Edit form loads with current choice details.	As expected	Pass
CM-UT05	Verify updating a choice with a new name.	Admin submits edited form with new name for the choice.	Choice name is updated in database.	As expected	Pass
CM-UT06	Ensure validation error is triggered for empty name during update.	Admin submits update form with blank name field.	Validation error appears for name.	As expected	Pass
CM-UT07	Verify that admin can delete a choice from a voting session.	Admin clicks 'Delete' on a specific choice.	Choice is removed from the database.	As expected	Pass

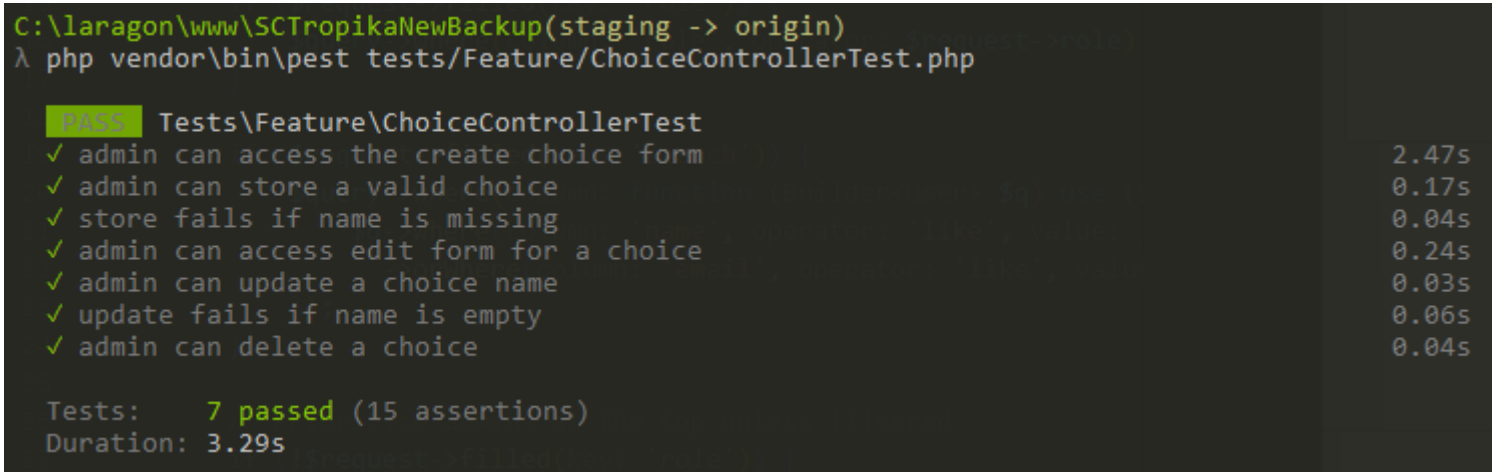


Figure 5.5: Unit Testing using Pest: Choice Management (Admin)

5.2.2.6 Unit Testing: User Management Module (Admin)

The following test cases validate the core functionalities of the User Management module, including listing, filtering, editing, updating, deleting users, and toggling roles and permissions with proper access restrictions.

Table 5.20: Unit Testing: User Management Module (Admin)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail

UM-UT01	Verify that admin can view the user list.	Admin logs in and accesses the user management page.	Paginated list of users is displayed.	As expected	Pass
UM-UT02	Verify that admin can filter users by role.	Admin applies 'admin' filter on user listing page.	Only admin users are displayed.	As expected	Pass
UM-UT03	Verify that admin can search for a user.	Admin searches user by name or email keyword.	Matching user results are shown.	As expected	Pass
UM-UT04	Verify that admin can access the edit form for a user.	Admin selects a normal user to edit.	Edit form loads with user's information.	As expected	Pass
UM-UT05	Ensure admin cannot access edit form for another admin or super admin.	Admin attempts to edit another admin user.	Access is denied and error message shown.	As expected	Pass
UM-UT06	Verify that admin can update user role and facility permission.	Admin updates a normal user's role and permission.	User's role and permission updated in DB.	As expected	Pass
UM-UT07	Ensure that admin cannot update themselves.	Admin attempts to change their own role.	Update is blocked and error is returned.	As expected	Pass
UM-UT08	Verify that super admin can toggle another user's role.	Super admin toggles role of a normal user.	User role updated between admin/user.	As expected	Pass
UM-UT09	Ensure user cannot toggle their own role.	Admin attempts to toggle their own role.	Toggle blocked with error message.	As expected	Pass
UM-UT10	Verify that admin can toggle facility access for another admin.	Admin toggles `can_manage_facility` flag of another admin.	Flag is updated accordingly.	As expected	Pass

UM-UT11	Ensure only admins can be granted facility access.	Admin attempts to toggle facility access for normal user.	Action blocked with error message.	As expected	Pass
UM-UT12	Ensure admin cannot delete super admin or other admins.	Admin tries deleting super admin or another admin.	Action blocked and error returned.	As expected	Pass
UM-UT13	Verify that admin can delete a normal user.	Admin deletes a regular user from the list.	User record is removed from DB.	As expected	Pass

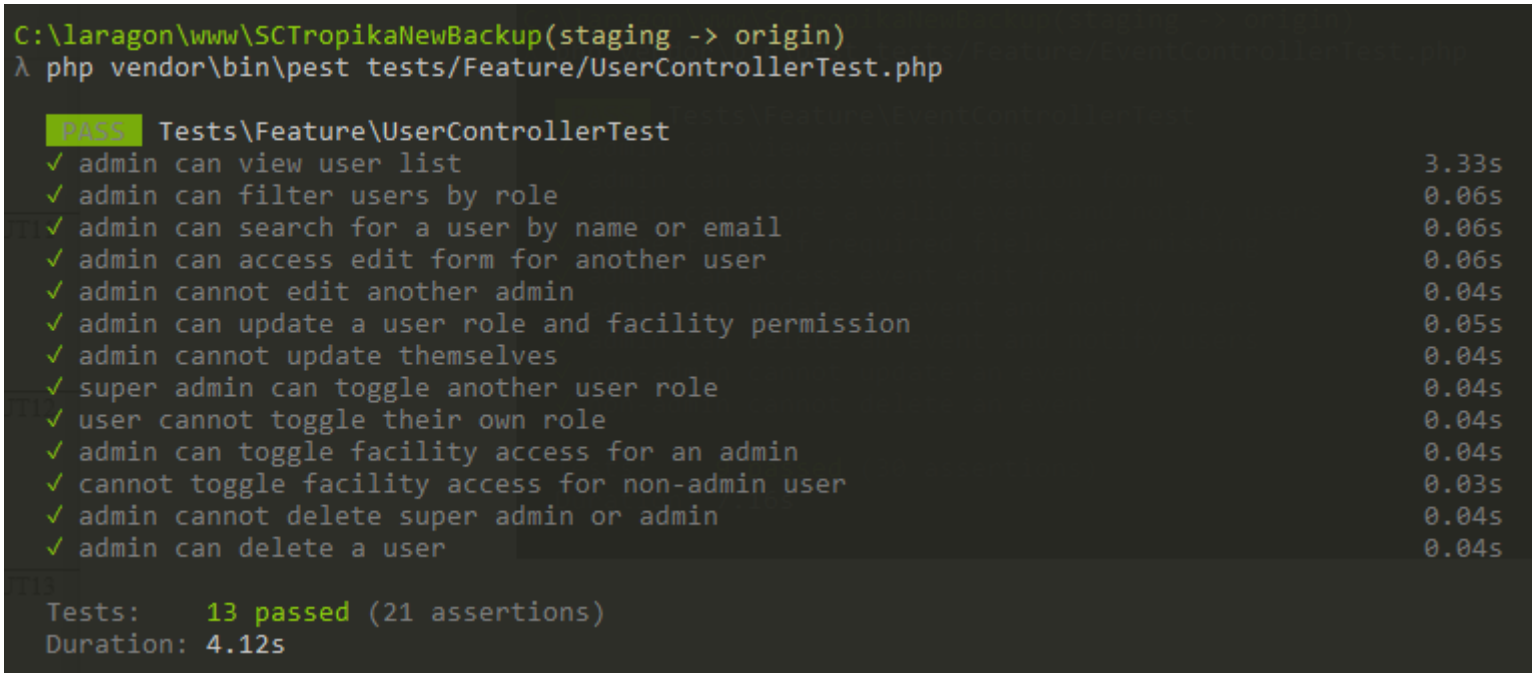


Figure 5.6: Unit Testing using Pest: User Management Module (Admin)

5.2.2.7 Unit Testing: Event Module (User)

The following test cases validate the functionalities available to users for viewing events in the system. This includes accessing a categorized list of upcoming and past events, viewing event highlights, and accessing detailed information for individual events.

Table 5.21: Unit Testing: Event Module (User)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
E-UUT01	Verify that user can access the event listing page.	User navigates to the 'Community Events' section.	Page loads with upcoming and past event sections.	As expected	Pass
E-UUT02	Verify that upcoming and past events are correctly categorized.	Database contains both future-dated and past-dated events.	Events are shown under respective 'Upcoming' and 'Past' sections.	As expected	Pass
E-UUT03	Verify that user can view the details of a selected event.	User clicks on an event card titled 'Community Gathering'.	Full event details are displayed on a dedicated page.	As expected	Pass
E-UUT04	Verify that attempting to view a non-existent event returns a 404 error.	User manually navigates to /events/9999 (non-existent ID).	System returns a 404 not found page.	As expected	Pass

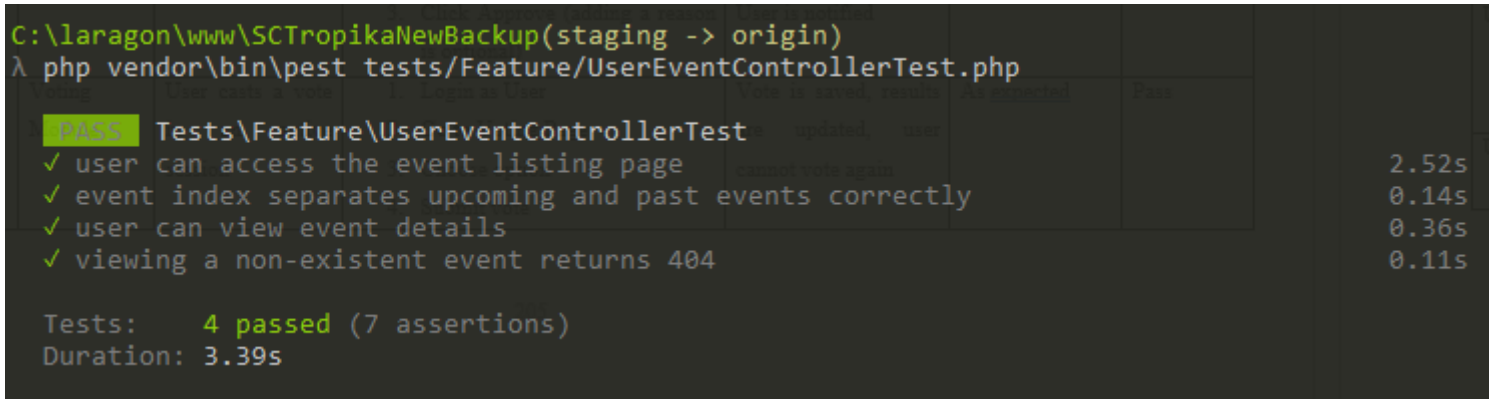


Figure 5.7: Unit Testing using Pest: Event Module (User)

5.2.2.8 Unit Testing: Facility Booking Module (User)

The following test cases validate the core functionalities available to users in the Facility Booking module. These include submitting booking requests, viewing personal booking details, and ensuring access control is enforced to prevent users from accessing or modifying bookings they do not own.

Table 5.22: Unit Testing: Facility Booking Module (User)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
FB-UTU01	Verify that user can view a list of their own booking requests.	User navigates to 'My Bookings' section after login.	List of user's bookings is displayed with status.	As expected	Pass

FB-UTU02	Verify that user can access the booking request form.	User clicks 'Request Booking' button on the booking page.	Form for booking request is displayed.	As expected	Pass
FB-UTU03	Verify that a user can submit a valid booking request.	User fills in valid facility, date, and time fields and submits.	Booking is saved and user is redirected with success message.	As expected	Pass
FB-UTU04	Ensure validation errors are triggered when fields are missing.	User submits empty form.	Validation messages appear for all required fields.	As expected	Pass
FB-UTU05	Verify that user can view details of their own booking.	User clicks on one of their submitted bookings.	Booking details are shown including date, facility, and status.	As expected	Pass
FB-UTU06	Ensure user cannot view booking details of another user.	User tries to access booking URL of another user's booking.	System returns 404 not found error.	As expected	Pass
FB-UTU07	Verify that user can delete their own booking request.	User clicks 'Cancel' on their own pending booking.	Booking is removed from the database.	As expected	Pass
FB-UTU08	Ensure user cannot delete booking requests made by others.	User tries to delete a booking made by a different user.	System blocks the action with 403 Forbidden or similar.	As expected	Pass

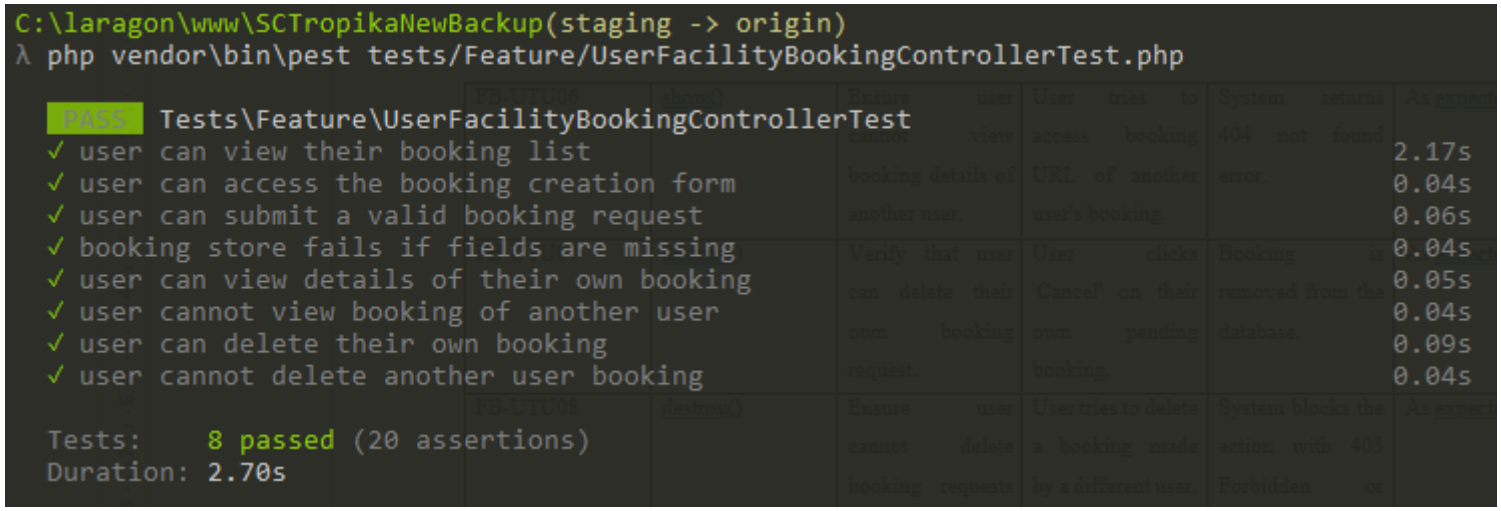


Figure 5.8: Unit Testing using Pest: Facility Booking Module (User)

5.2.2.9 Unit Testing: Voting Module (User)

The following test cases validate the voting features available to users. This includes viewing available voting sessions, accessing voting details, casting votes, and updating existing votes, while ensuring proper validation and session time restrictions are enforced.

Table 5.23: Unit Testing: Voting Module (User)

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail

V-UT01	Verify that users can view the list of voting sessions.	User visits the voting page after logging in.	Voting list is displayed with session status and voting options.	As expected	Pass
V-UT02	Verify that users can access the details of a voting session.	User clicks on a voting title to view its details.	Voting details, choices, and status are displayed.	As expected	Pass
V-UT03	Verify that user can submit a vote for an active session.	User selects a choice from an active voting session and submits.	Vote is recorded and success message is shown.	As expected	Pass
V-UT04	Verify that user can update their existing vote.	User re-submits a different choice for the same voting session.	Existing vote is updated with new choice.	As expected	Pass
V-UT05	Ensure vote submission fails if no choice is selected.	User submits vote form without selecting a choice.	Validation error is returned for missing choice_id.	As expected	Pass
V-UT06	Ensure vote cannot be submitted if the session is inactive or expired.	User tries to vote in a session with status 'inactive' and past end date.	Submission is rejected and user is redirected.	As expected	Pass

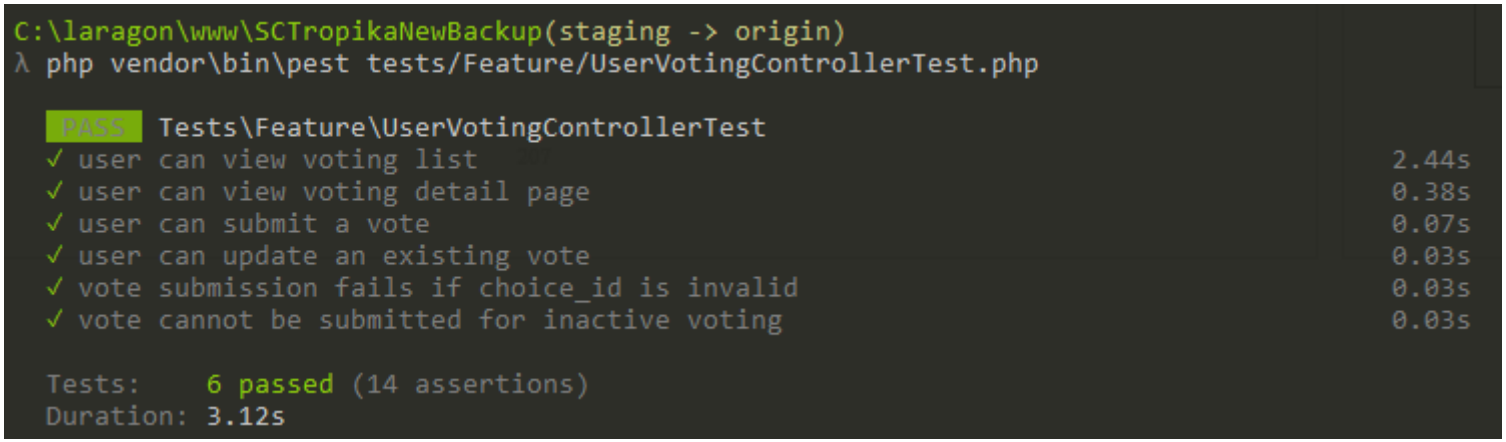


Figure 5.9: Unit Testing using Pest: Voting Module (User)

5.2.2.10 Unit Testing: Comment Management Module

The following test cases validate the core functionalities of the Comment Management feature, which is accessible to both regular users and admins. These include posting new comments, replying to existing ones, editing and deleting one's own comments, as well as admin-specific capabilities such as moderating comments with a stated reason. The tests also ensure proper validation, role-based access control, and the correct triggering of notifications upon administrative actions.

Table 5.24: Unit Testing: Comment Management Module

Test Case ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail

CM-UT01	Verify that user can post a new comment	Valid thread ID, content	Redirect to thread with success	Redirect with success	Pass
CM-UT02	Verify that user can reply to a comment	Valid thread ID, parent_id, content	Redirect to thread with success	Redirect with success	Pass
CM-UT03	Verify that comment store fails if content is missing	Empty content	Validation error	Validation error	Pass
CM-UT04	Verify that user cannot access edit form of another user's comment	Comment owned by another user	403 Forbidden	403 Forbidden	Pass
CM-UT05	Verify that user can update their own comment	Valid comment ID and content	Redirect with success	Redirect with success	Pass
CM-UT06	Verify that user cannot update another user's comment	Comment owned by another user	403 Forbidden	403 Forbidden	Pass
CM-UT07	Verify that user can delete their own comment	Valid comment ID	Redirect with success	Redirect with success	Pass
CM-UT08	Verify that user cannot delete another user's comment	Comment owned by another user	403 Forbidden	403 Forbidden	Pass
CM-UT09	Verify that admin can delete a comment with reason and notify	Comment ID and reason	Redirect with notification stored	Redirect with notification stored	Pass
CM-UT10	Verify that admin deletion with reason fails if reason is missing	Comment ID, empty reason	Validation error	Validation error	Pass

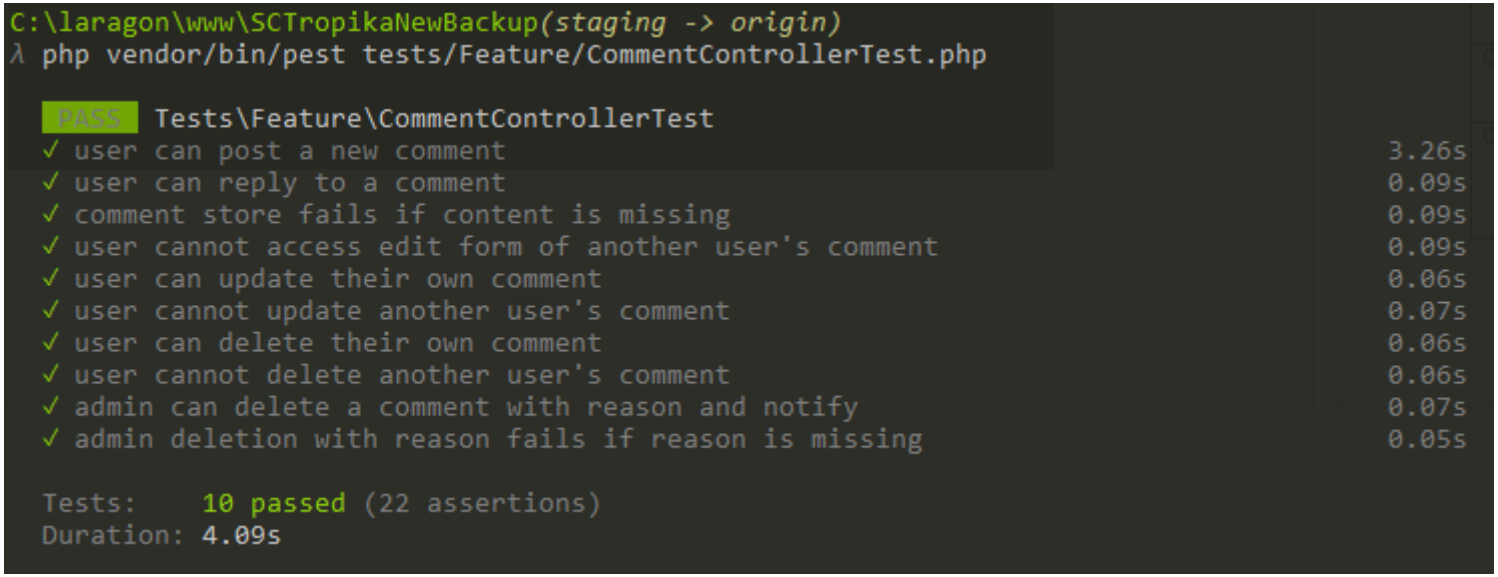


Figure 5.10: Unit Testing using Pest: Comment Management Module

5.2.2.11 Unit Testing: Thread Management Module

The following test cases validate the functionalities provided by the Thread Management Module. This includes listing threads, posting new threads, editing, deleting, and sending notifications on announcements. It also verifies access control, input validation, and admin moderation features.

Table 5.25: Unit Testing: Thread Management Module

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
TM-UT01	Verify that user can view the thread index page.	User navigates to thread listing route.	Thread index page with announcements and threads is displayed.	As expected	Pass
TM-UT02	Verify that user can view details of a specific thread.	Valid thread ID is accessed by user.	Thread detail page is displayed with view count and comments.	As expected	Pass
TM-UT03	Verify that a user can create a new normal thread.	Valid title, content, and optional description without 'type'.	Thread is created with type 'normal' and redirects to index.	As expected	Pass
TM-UT04	Verify that an admin can create an announcement and notify all users.	Valid title, content, and type set as 'announcement' by admin.	Thread is saved as announcement and notifications are sent.	As expected	Pass
TM-UT05	Verify that a user can update their own thread.	User accesses edit form and submits valid title, content, and type.	Thread is updated and redirected to show page.	As expected	Pass
TM-UT06	Ensure that a user cannot access edit form of another user's thread.	User tries to access edit route for another user's thread.	403 Forbidden response is returned.	As expected	Pass

TM-UT07	Verify that a user can delete their own thread.	User sends DELETE request for their thread.	Thread is deleted and user redirected to index.	As expected	Pass
TM-UT08	Verify that an admin can delete a thread with reason and notify the owner.	Admin submits deletion with reason for another user's thread.	Thread is deleted and notification is created.	As expected	Pass
TM-UT09	Ensure validation fails if admin tries to delete with missing reason.	Admin submits form without 'reason' field.	Validation error for missing reason is returned.	As expected	Pass

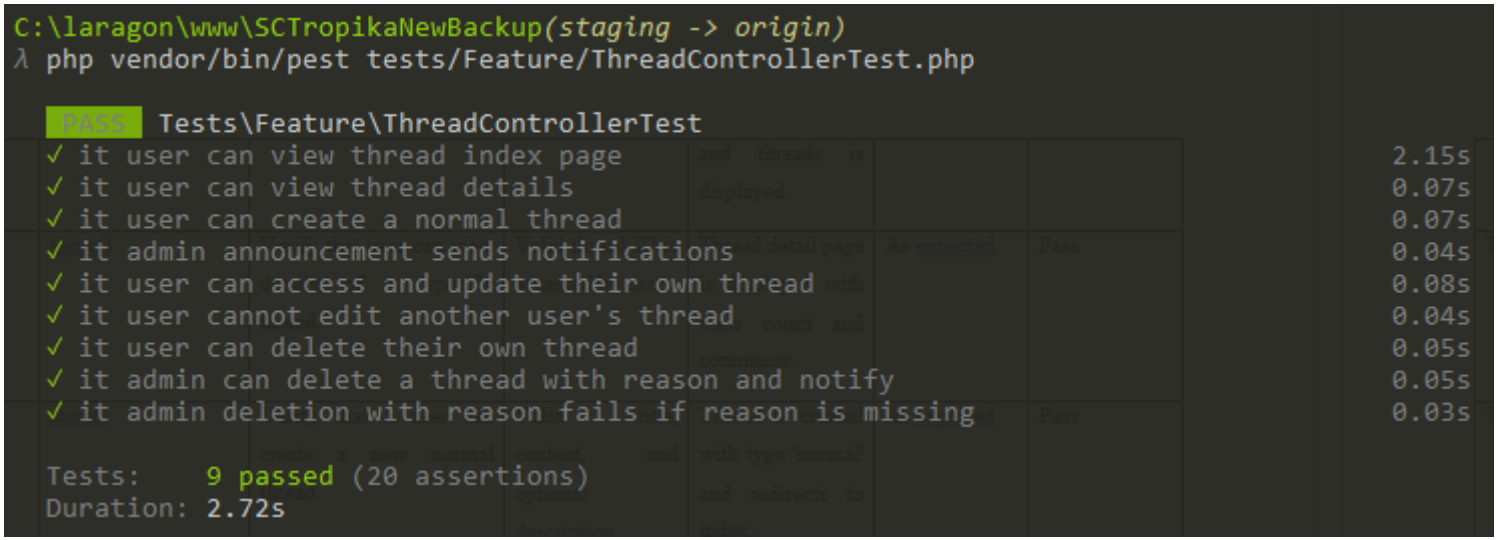


Figure 5.11: Unit Testing using Pest: Thread Management Module

5.2.2.12 Unit Testing: Notification Module

The following test cases validate the Notification Management Module of the system. This module allows users to view, filter, and manage their notifications. The test cases cover functionality for viewing all notifications, filtering read/unread notifications, marking a single notification as read, handling access control for marking another user's notification, and marking all notifications as read in bulk.

Table 5.26: Unit Testing: Notification Module

Test Case ID	Test Objective	Input Data/Steps	Expected Result	Actual Result	Pass/Fail
N-UT01	Verify that user can view all notifications.	User logs in and navigates to the notifications page.	All notifications belonging to the user are listed.	As expected	Pass
N-UT02	Verify that user can filter unread notifications.	User navigates to the notifications page and selects the 'Unread' filter.	Only unread notifications are displayed.	As expected	Pass
N-UT03	Verify that user can filter read notifications.	User navigates to the notifications page and selects the 'Read' filter.	Only read notifications are displayed.	As expected	Pass
N-UT04	Verify that user can mark their own notification as read.	User clicks 'Mark as Read' on one of their unread notifications from the list.	Notification is marked as read.	As expected	Pass

N-UT05	Verify that user cannot mark another user's notification as read.	User attempts to mark as read a notification that belongs to another user.	403 Forbidden or access denied.	Access denied as expected	Pass
N-UT06	Verify that user can mark all their notifications as read.	User clicks the 'Mark All as Read' button while having unread notifications.	All of the user's notifications are marked as read.	All marked as read successfully	Pass

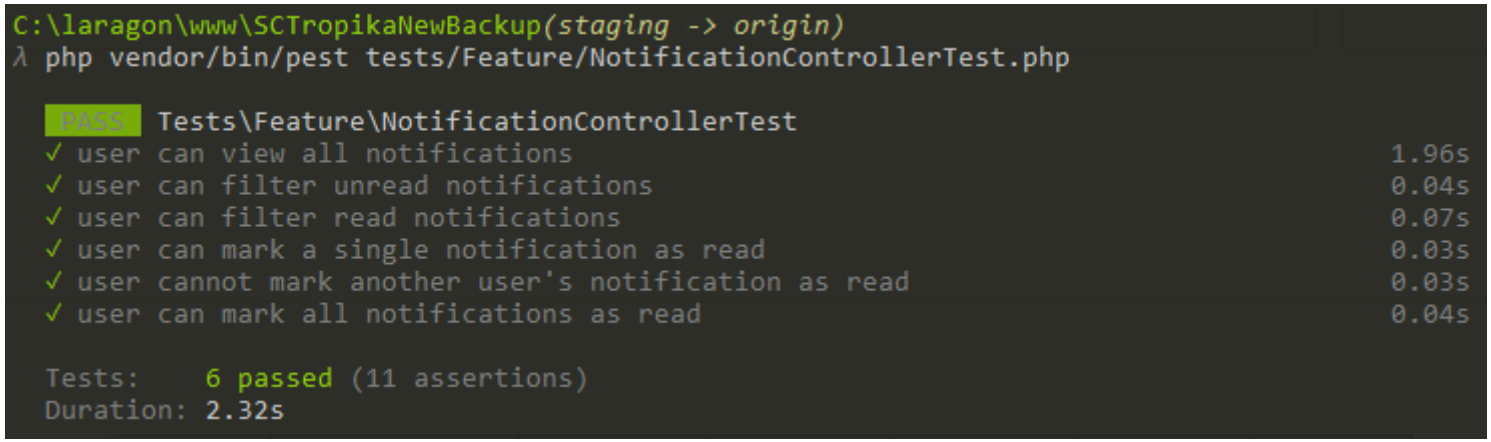


Figure 5.12: Unit Testing using Pest: Notification Module

5.2.2.13 Unit Testing: Profile Management Module

The following test cases validate the profile management functionalities available to authenticated users. This includes viewing the profile page, updating personal information (such as name and email), verifying changes to email status, as well as securely deleting their own account with password validation.

Table 5.27: Unit Testing: Profile Management Module

Test ID	Test Objective	Input Data / Steps	Expected Output	Actual Output	Pass/Fail
PM-UT01	Verify that the profile page can be accessed.	Login as a valid user and visit `/profile`.	Profile page should load successfully with HTTP 200.	Page loads and returns HTTP 200.	Pass
PM-UT02	Verify profile information can be updated.	Login as user, PATCH `/profile` with new name and email.	User's name and email updated, email verification reset.	Data updated and `email_verified_at` set to null.	Pass
PM-UT03	Ensure email remains verified if unchanged.	Login as user, PATCH `/profile` with the same email.	Email remains verified (`email_verified_at` remains not null).	Email stays verified.	Pass
PM-UT04	Verify account deletion with correct password.	Login as user, DELETE `/profile` with correct password.	User is logged out, session invalidated, and account deleted.	User logged out and record removed.	Pass

PM-UT05	Ensure account cannot be deleted with incorrect password.	Login as user, DELETE `/profile` with wrong password.	Validation error, redirected back, and account not deleted.	Redirected with password error, user still exists.	Pass

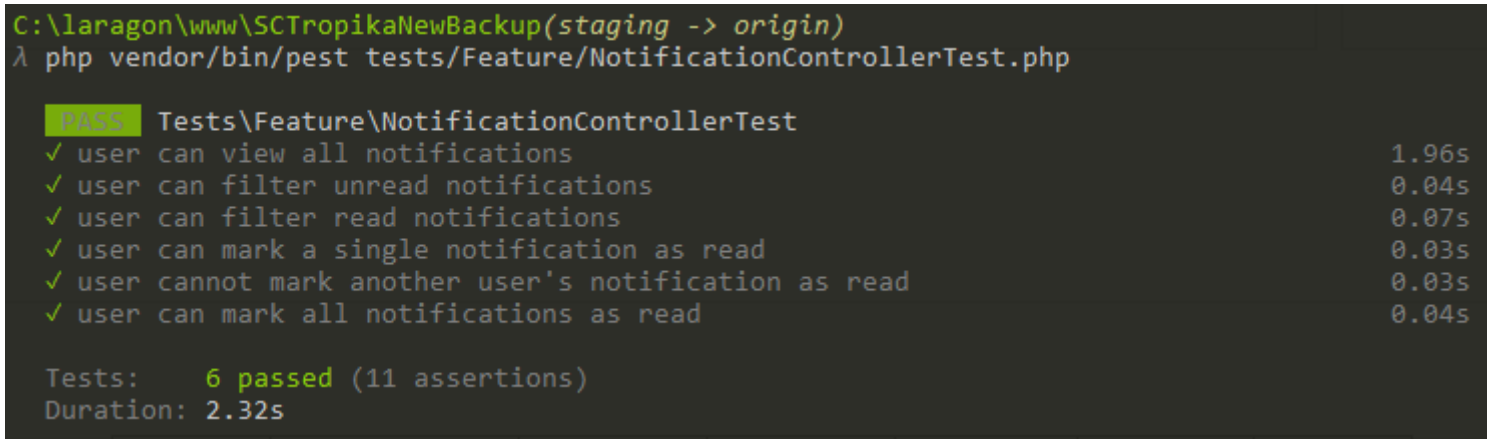


Figure 5.13: Unit Testing using Pest: Profile Management Module

5.2.2.14 Unit Testing: Authentication Module

The following unit test cases validate the core authentication features of the system. These include rendering the login page, successful and failed login attempts, and the logout process. The tests ensure the system correctly handles user sessions, authentication status, and redirection.

Table 5.28: Unit Testing: Authentication Module

Test ID	Test Objective	Input Data / Steps	Expected Output	Actual Output	Pass/Fail
AUTH-UT01	Verify that the login screen can be accessed by unauthenticated users.	Send a GET request to '/login'.	The system returns a 200 OK response with the login form.	200 OK status with login form rendered.	Pass
AUTH-UT02	Ensure that users can authenticate with correct credentials.	POST to '/login' with valid email and password.	User is authenticated and redirected to their dashboard.	Authenticated and redirected to '/user/dashboard'.	Pass
AUTH-UT03	Ensure that users cannot authenticate using an invalid password.	POST to '/login' with a valid email and incorrect password.	Authentication fails and user remains a guest.	User remains unauthenticated.	Pass
AUTH-UT04	Verify that authenticated users can log out successfully.	POST to '/logout' as an authenticated user.	User session ends and is redirected to home page.	User is logged out and redirected to '/'.	Pass

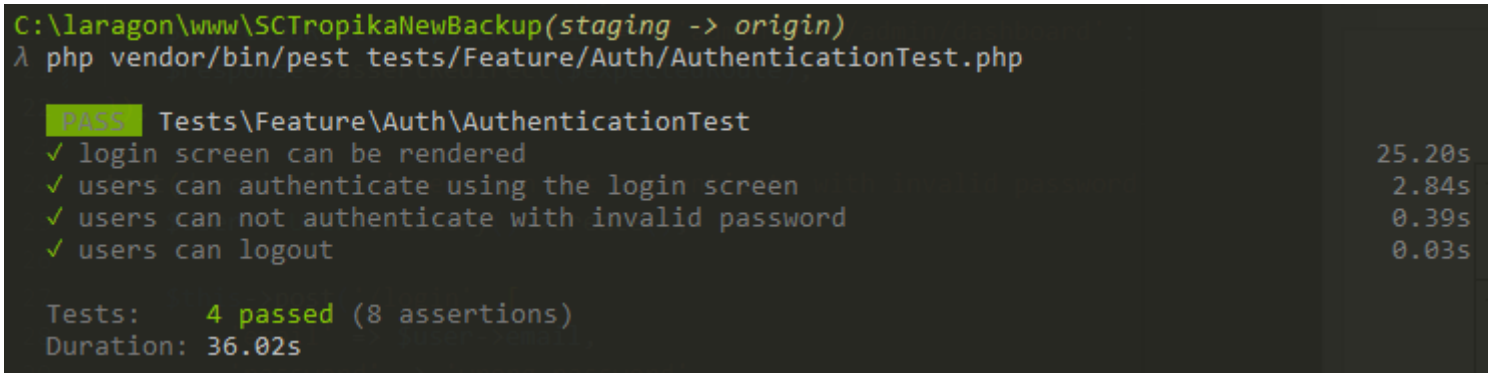


Figure 5.14: Unit Testing using Pest: Authentication Module

5.2.2.15 Unit Testing: Email Verification Module

The following unit tests validate the Email Verification functionality within the authentication system. These tests ensure that unverified users are shown the appropriate verification screen, can verify their emails using a valid signed URL, and that verification fails when the URL hash is invalid.

Table 5.29: Unit Testing: Email Verification Module

Test ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail

EV-UT01	Verify that user can view the email verification screen.	Login as an unverified user and navigate to /verify-email.	The verification screen is displayed with a 200 status.	Verification screen displayed successfully.	Pass
EV-UT02	Verify that user can verify email with valid URL.	Login as unverified user and access a valid signed verification URL.	The email is marked as verified and redirected to dashboard with ?verified=1.	Email verified and user redirected to dashboard.	Pass
EV-UT03	Verify that user cannot verify email with an invalid hash.	Login as unverified user and access an invalid verification URL.	The email remains unverified.	Verification failed, email remains unverified.	Pass

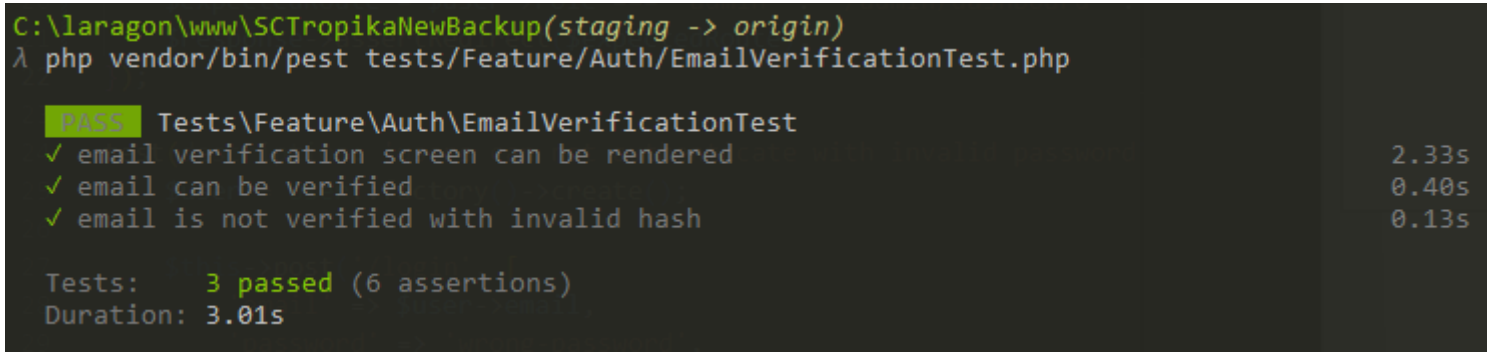


Figure 5.15: Unit Testing using Pest: Email Verification Module

5.2.2.16 Unit Testing: Password Confirmation Module

This section documents the unit testing conducted for the Password Confirmation Module of the system. It is critical for securing sensitive operations that require a recent password confirmation, such as account deletion or accessing protected settings. The test cases verify that the password confirmation screen is accessible to authenticated users, that correct passwords allow access to restricted actions, and that invalid passwords are appropriately rejected.

Table 5.30: Unit Testing: Password Confirmation Module

Test Case ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail
PC-UT01	Verify that the password confirmation screen is accessible when a user is authenticated.	User logs in and navigates to /confirm-password.	HTTP 200 OK response indicating the page is displayed.	HTTP 200 OK response received.	Pass
PC-UT02	Ensure that submitting the correct password confirms the user session successfully.	User submits the correct password to /confirm-password.	User is redirected and no validation errors occur.	Redirect occurs successfully with no session errors.	Pass

PC-UT03	Ensure that an incorrect password fails to confirm the user session.	User submits an invalid password to /confirm-password.	Session contains password validation errors.	Validation errors were triggered as expected.	Pass

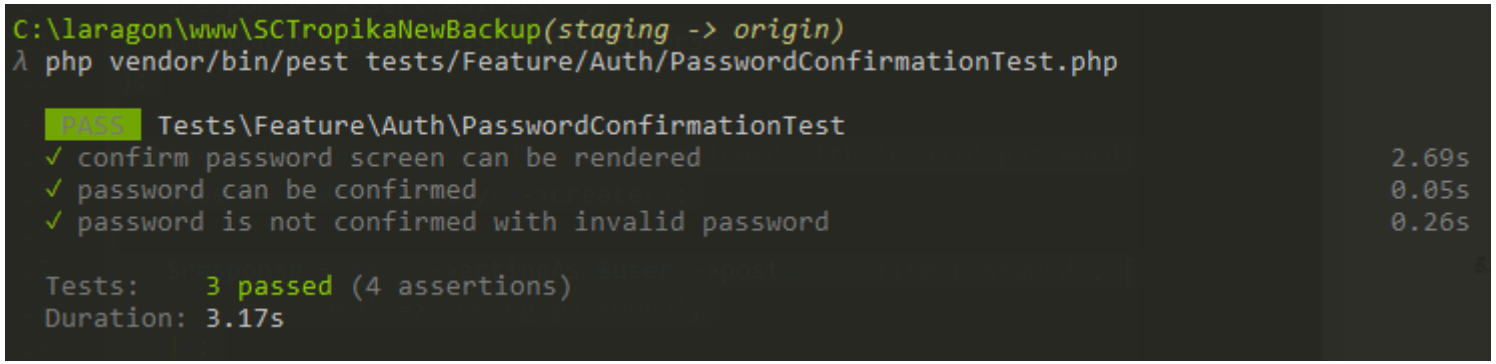


Figure 5.16: Unit Testing using Pest: Password Confirmation Module

5.2.2.17 Unit Testing: Password Reset Module

This section outlines the unit testing conducted for the Password Reset Module of the system. The module facilitates users in securely resetting their passwords in the event they forget their credentials. The test cases verify that the password reset link page is accessible, reset links can be requested, the reset form can be displayed using a valid token, and that passwords can be successfully updated.

Table 5.31: Unit Testing: Password Reset Module

Test ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail
PR-UT01	Verify that the reset password link screen can be rendered.	Navigate to the /forgot-password URL.	Password reset request screen is displayed (HTTP 200).	Screen rendered successfully.	Pass
PR-UT02	Verify that a password reset link can be requested.	Submit the form at /forgot-password with a valid registered email address.	ResetPassword notification is sent to the user's email.	Notification sent successfully.	Pass
PR-UT03	Verify that the reset password screen can be rendered using a valid token.	Submit reset request, retrieve token, and navigate to /reset-password/{token}.	Reset password form is displayed (HTTP 200).	Form rendered successfully.	Pass
PR-UT04	Verify that password can be reset using a valid token.	Submit POST to /reset-password with valid token, email, and new password.	Password is updated and user is redirected to login.	Password reset successful and redirected.	Pass

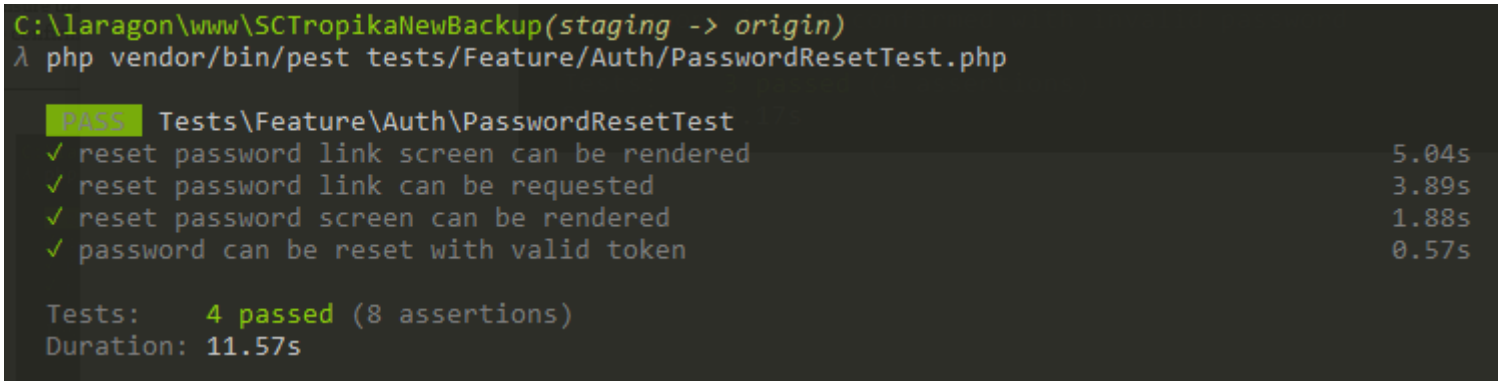


Figure 5.17: Unit Testing using Pest: Password Reset Module

5.2.2.18 Unit Testing: Password Update Module

This section covers unit testing for the Password Update Module, which ensures that users can update their password securely via the profile settings page. The tests validate successful password updates when the correct current password is provided and ensure that errors are returned when the current password is incorrect.

Table 5.32: Unit Testing: Password Update Module

Test ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail

PU-UT01	Verify that the user can successfully update their password using the correct current password.	1. Authenticate as a valid user. 2. Navigate to /profile and submit the form with: • current_password: password • password: new-password • password_confirmation: new-password	User password is updated successfully and redirected to profile page.	Password updated and redirected successfully.	Pass
PU-UT02	Ensure that the system prevents password update if the current password is incorrect.	1. Authenticate as a valid user. 2. Navigate to /profile and submit the form with: • current_password: wrong-password • password: new-password • password_confirmation: new-password	Error message shown; password is not updated.	Validation error shown; redirect to profile.	Pass

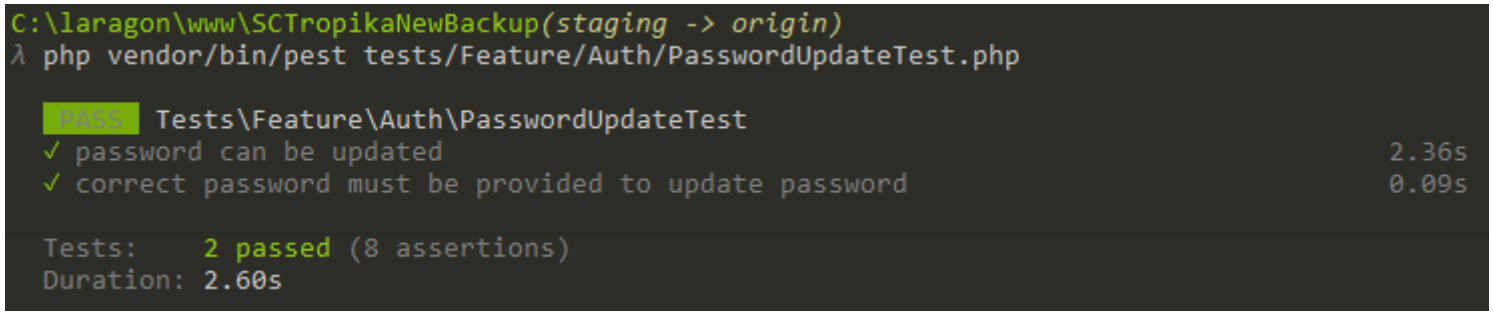


Figure 5.18: Unit Testing using Pest: Password Update Module

5.2.2.19 Unit Testing: Registration Module

This section outlines the unit testing conducted for the Registration Module of the system. The objective is to verify that users can successfully access the registration screen and complete the registration process. Tests include rendering the registration form and verifying that new user accounts are registered and authenticated appropriately.

Table 5.33: Unit Testing: Registration Module

Test ID	Test Objective	Input Data/Steps	Expected Output	Actual Output	Pass/Fail
RM-UT01	Verify that the registration screen can be accessed.	Navigate to /register while not authenticated.	HTTP 200 status code and registration form is displayed.	Registration page rendered successfully.	Pass
RM-UT02	Verify that a new user can register successfully.	Submit the registration form with name, email, password, and confirmation.	User is authenticated and redirected to their dashboard.	User registered and redirected as expected.	Pass

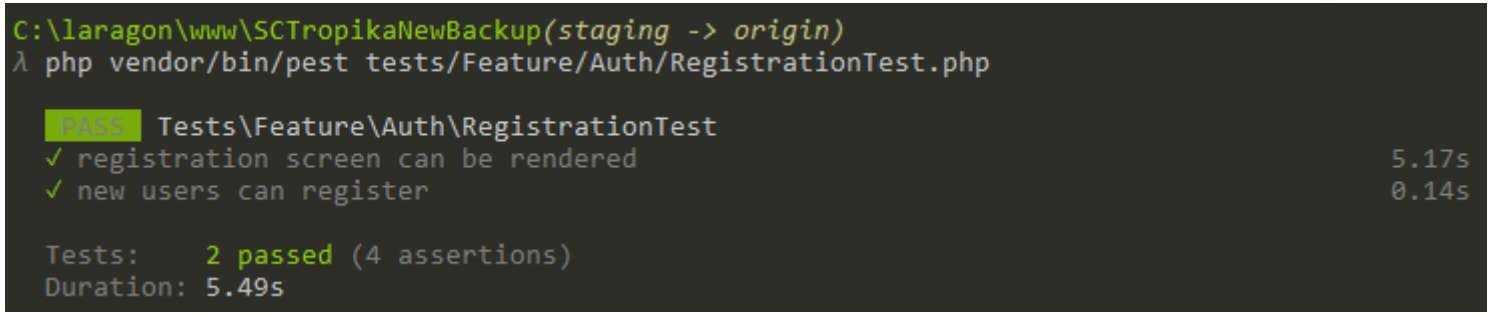


Figure 5.19: Unit Testing using Pest: Registration Module

5.3 Non-Functional Testing

Non-functional testing evaluates how well the system performs in terms of user experience, efficiency, and operational quality, rather than just correctness of functionality. For the Smart Community Platform for Taman Suria Tropika, non-functional testing focused primarily on usability, including interface design, ease of navigation, and user satisfaction across various modules. This type of testing helps ensure that the system not only works as intended but is also intuitive, accessible, and practical for daily use by community members and administrators.

Unlike functional testing, which verifies whether specific features perform their intended tasks, non-functional testing addresses how users interact with the system, how easily tasks can be completed, and how comfortable the interface for utilization. This is particularly important in a community-based system, where users may have varying levels of technical experience.

5.3.1 Usability Testing

Usability testing was conducted to evaluate how effectively users could navigate, interact with, and complete tasks within the Smart Community Platform for Taman Suria Tropika. A number of respondents had been requested to test the system's core modules and provide feedback on their experience through a structured questionnaire.

To collect responses systematically, the questionnaire was created using Google Forms. The form gathered both quantitative data through Likert-scale questions and qualitative feedback through open-ended prompts. The focus areas included ease of use, interface design, responsiveness, and overall satisfaction.

To better understand the background of the respondents, they were first asked about their familiarity with using community or management systems. As shown in Figure 5.20, the results indicated that:

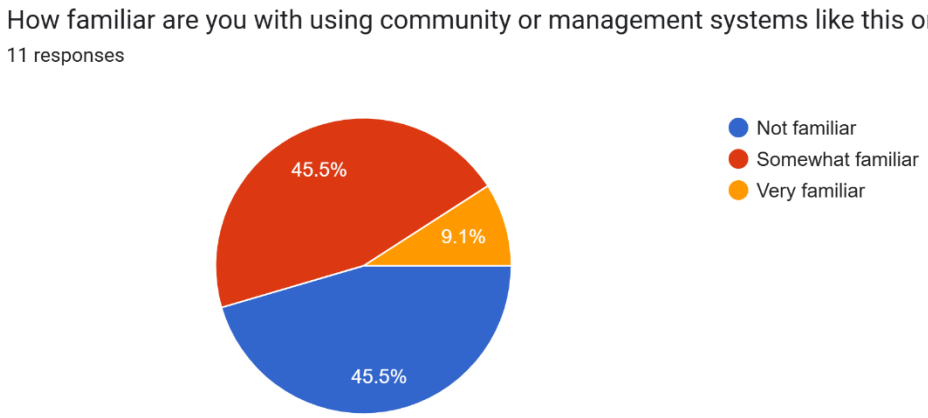


Figure 5.20: User familiarity with community or management systems

- 45.5% of users were not familiar with such systems.
- 45.5% were somewhat familiar.
- 9.1% were very familiar.

Despite the relatively low familiarity among most participants, the majority were still capable of completing tasks and interact with the system without difficulty. The sample questionnaire used for this usability testing is included in Appendix F for reference.

5.3.1.1 System Functionality of Smart Community Platform for Taman Suria Tropika

As shown in Figure 5.21, 10 out of 11 users (90.9%) strongly agreed that they were able to log in without any issues, while one user rated a 2, possibly due to a connection or unfamiliarity issue. This indicates that the login mechanism is generally reliable and accessible.

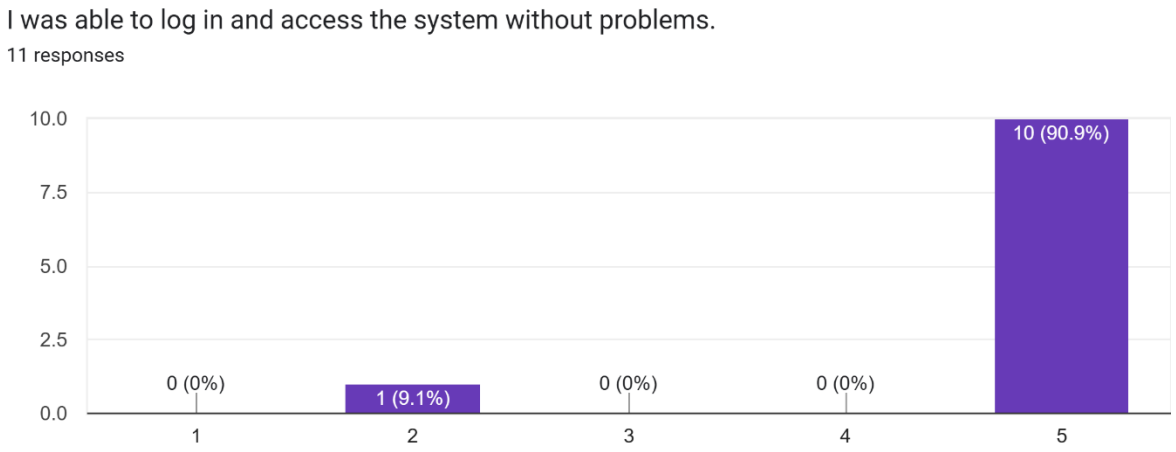


Figure 5.21: User feedback on login and system access experience

According to Figure 5.22, 81.8% of users gave a rating of 5, with two others giving lower scores (3 and 4). This affirms that the dashboard layout is generally effective, though a small refinement to improve icon clarity, text labeling or the overall layout might enhance accessibility further.

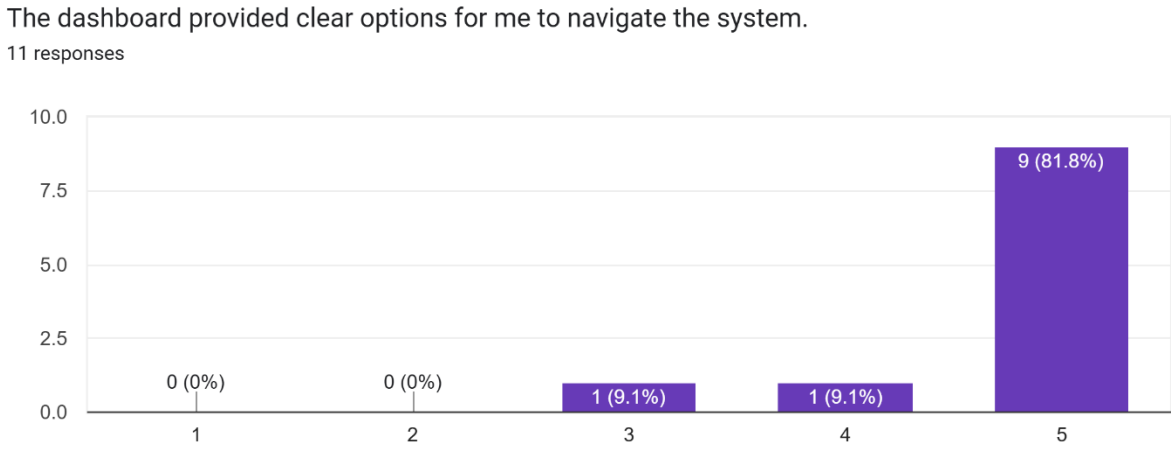


Figure 5.22: User feedback on dashboard clarity and navigation ease

This chart in Figure 5.23 shows a majority of users (63.6%) found the event information easy to interpret, while a few selected 3 and 4. This suggests that while the event cards and modal system are working well, minor improvements could be made for first-time users to improve clarity.

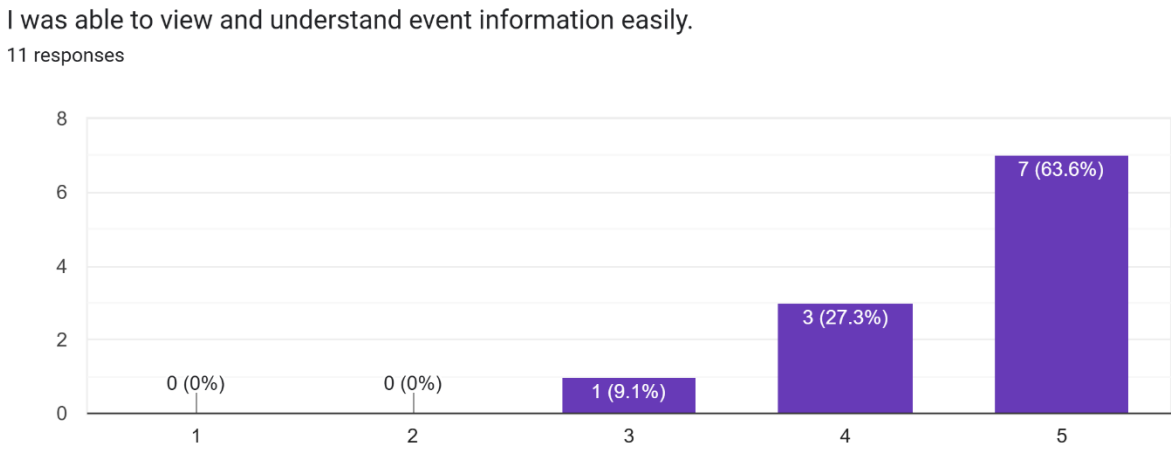


Figure 5.23: User feedback on event information visibility and understanding

As illustrated in Figure 5.24, the majority of users (72.7%) gave the highest rating, strongly agreeing that they could submit a booking without any issues. One user selected a 2 and two selected a 3, which may suggest some minor confusion during form completion or calendar selection. Despite this, overall results reflect that the booking feature was well-received and easy to use for most respondents.

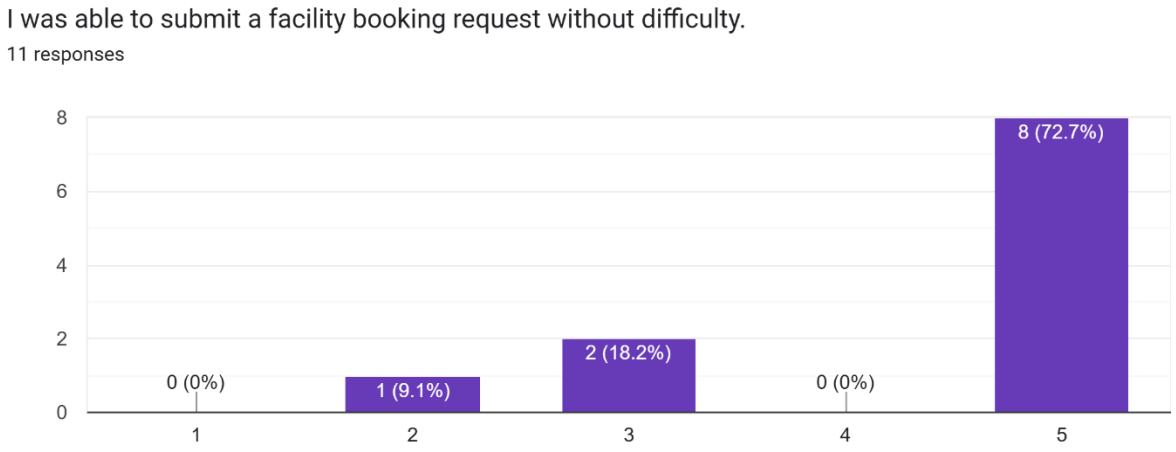


Figure 5.24: User feedback on ease of submitting facility booking requests

Based on Figure 5.25, a strong 81.8% of users found the voting process to be fairly easy, giving it a full score. Two other users selected 3, indicating possible uncertainty or delay

in understanding the vote submission or confirmation process. Still, no users reported severe issues, and the feature received one of the highest overall scores.

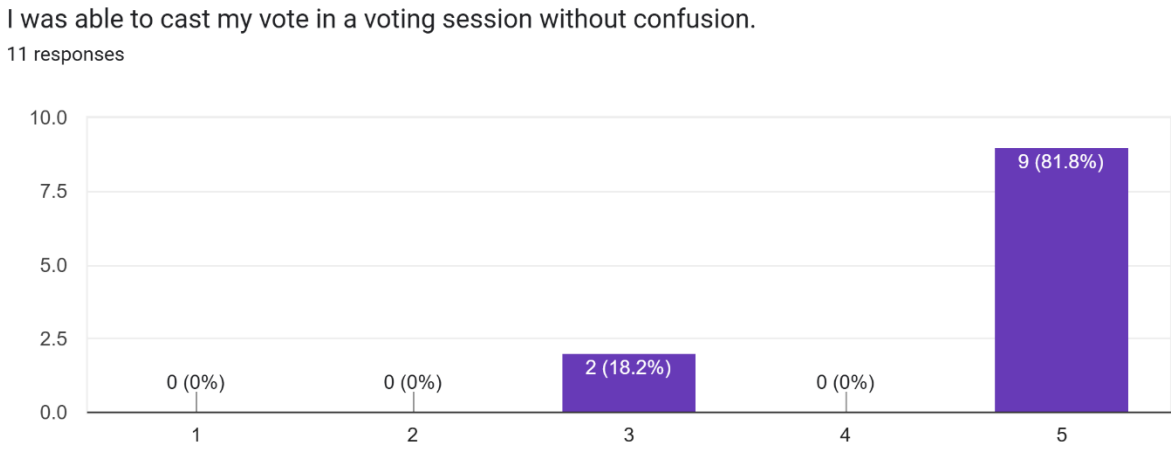


Figure 5.25: User feedback on clarity of the voting process

Looking at Figure 5.26, just over half of respondents (54.5%) rated this with a 5, while 27.3% rated it 4 and 18.2% rated it 3. The lower ratings may reflect initial unfamiliarity with the layout or comment flow in the Communication Hub. Overall, however, users were able to participate in community discussions effectively and with minimal barriers.

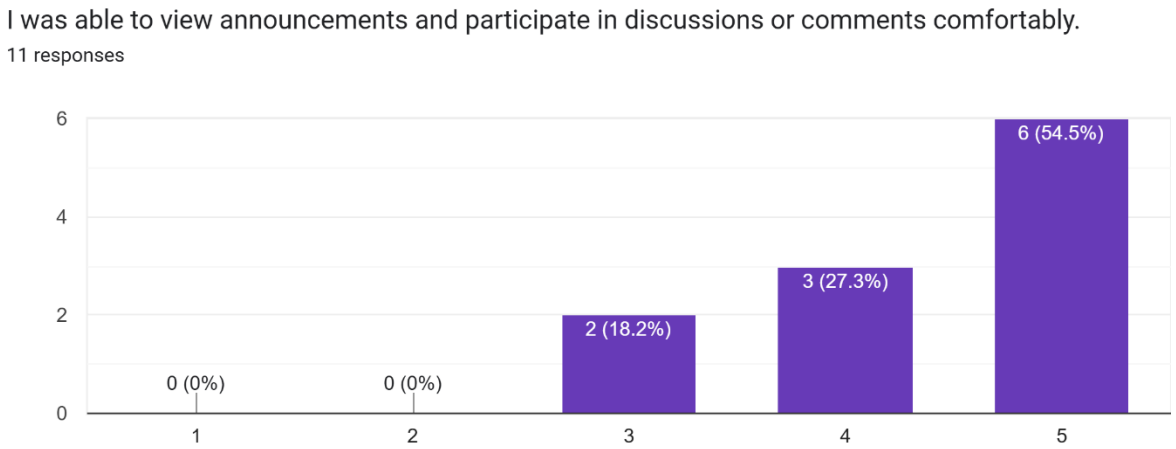


Figure 5.26: User feedback on viewing announcements and joining discussions

As shown in Figure 5.27, 70% of users gave the highest score of 5, indicating that the financial module which includes transaction tables, charts, and categorized summaries was easy to interpret. A small portion (20%) rated it 3 and one user (10%) rated it 4, possibly suggesting that users have difficulty using it and may need brief tooltips or guides. Overall, the feature is accessible and readable for most users.

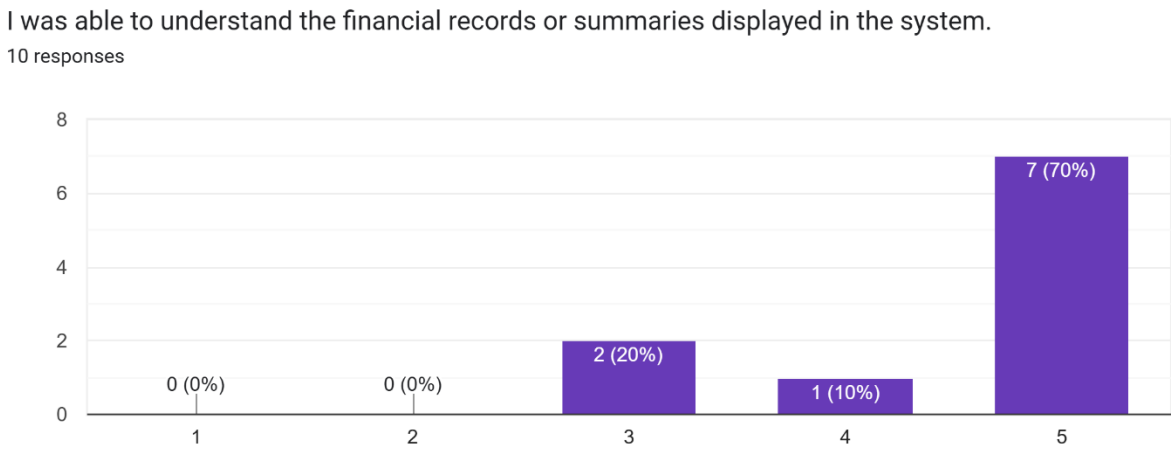


Figure 5.27: User feedback on understanding financial records and summaries

Figure 5.28 demonstrates that 72.7% of users rated this a 5, affirming the usefulness of confirmation modals, confirmation messages, and visual updates after completing tasks. Two users gave a 4 and one gave a 3, suggesting a generally positive experience with some room for improvement. The data confirms that most users were rarely left wondering whether their actions had been successful.

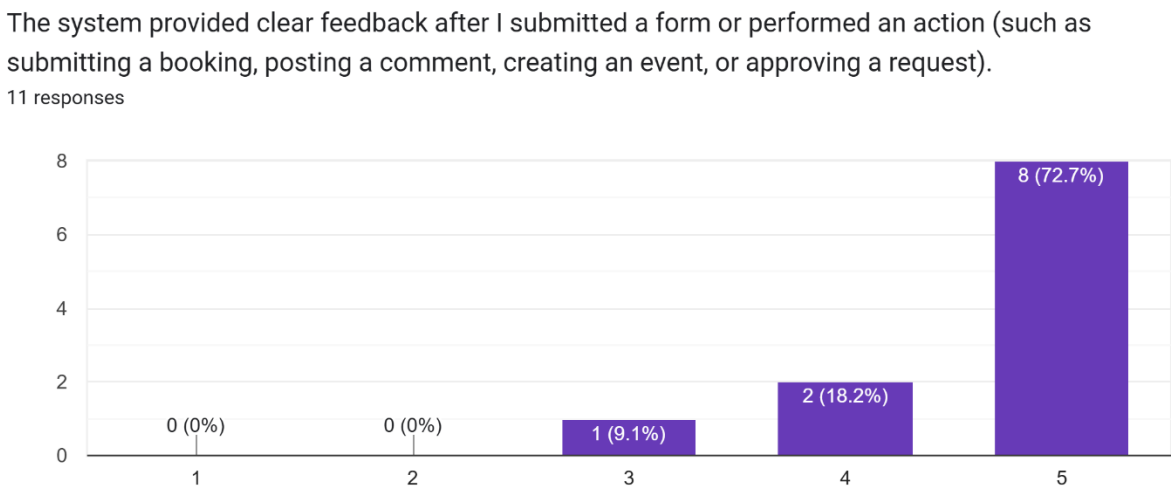


Figure 5.28: User feedback on clarity of system responses and action feedback

According to Figure 5.29, 63.6% of users rated the system's performance at the highest level of 5. Additionally, two users selected a rating of 4, while one user each selected 2 and 3. This indicates that the majority experienced a fast and responsive interface, whereas a small minority may have encountered brief delays or slower response times. Nonetheless, no serious performance issues or system errors were reported, affirming that the platform functions smoothly under normal usage conditions.

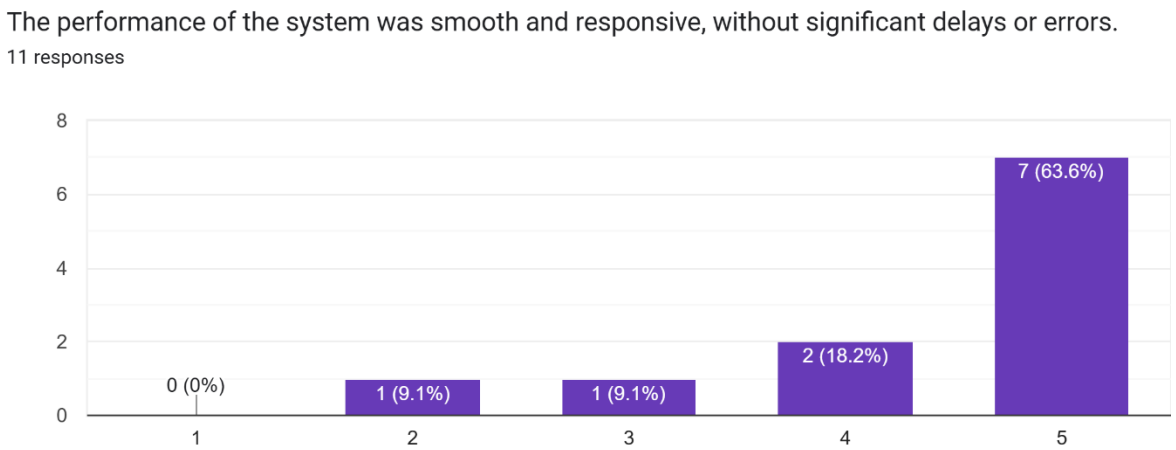


Figure 5.29: User feedback on overall system responsiveness and performance

5.3.1.2 User Interface Design of Smart Community Platform for Taman Suria Tropika

As shown in Figure 5.30, 81.8% of respondents gave a score of 5, indicating strong agreement that the system's overall layout is clear and easy to navigate. Only two users gave scores of 3 and 4, which may reflect slight difficulty with certain module layouts or a learning curve on first use.

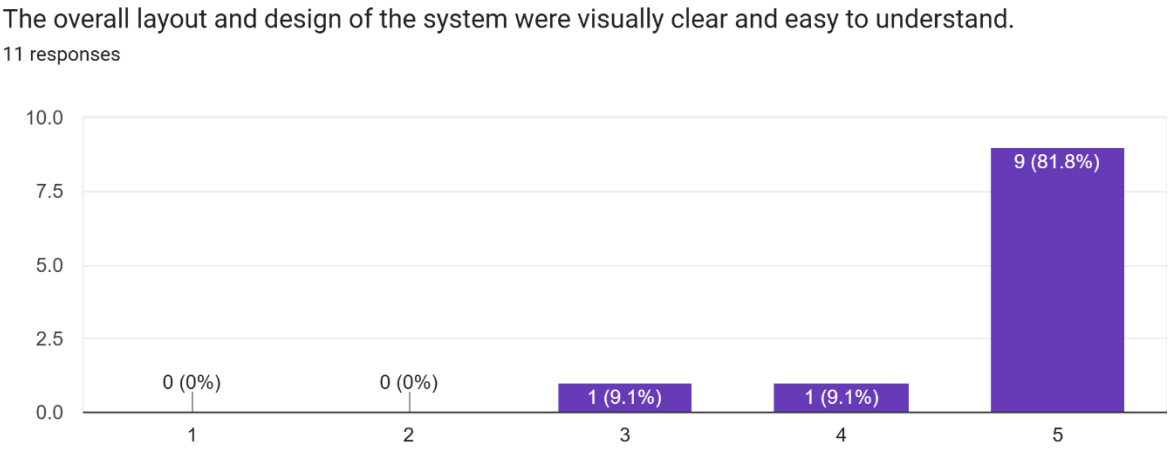


Figure 5.30: User feedback on overall system layout and visual clarity

Figure 5.31 shows that 9 out of 11 users (81.8%) rated this aspect a 5, highlighting that the dashboard layout was intuitive and efficiently structured. Only one user each rated it a 3 and 4, which may be attributed to first-time navigation. Overall, the dashboard effectively supports users in accessing features without confusion or clutter.

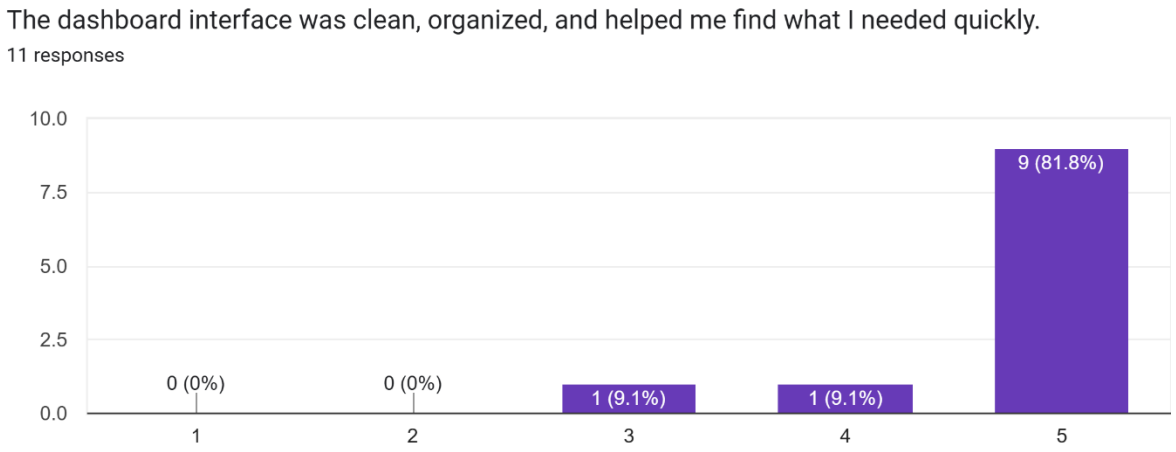


Figure 5.31: User feedback on dashboard organization and usability

According to Figure 5.32, 63.6% of users rated the event module layout with a 5, while the remaining respondents gave scores of 3 or 4. This suggests that the use of image cards, modal popups, and clean lists made event information accessible and attractive to most users, though minor visual refinements could further improve clarity for all user groups.

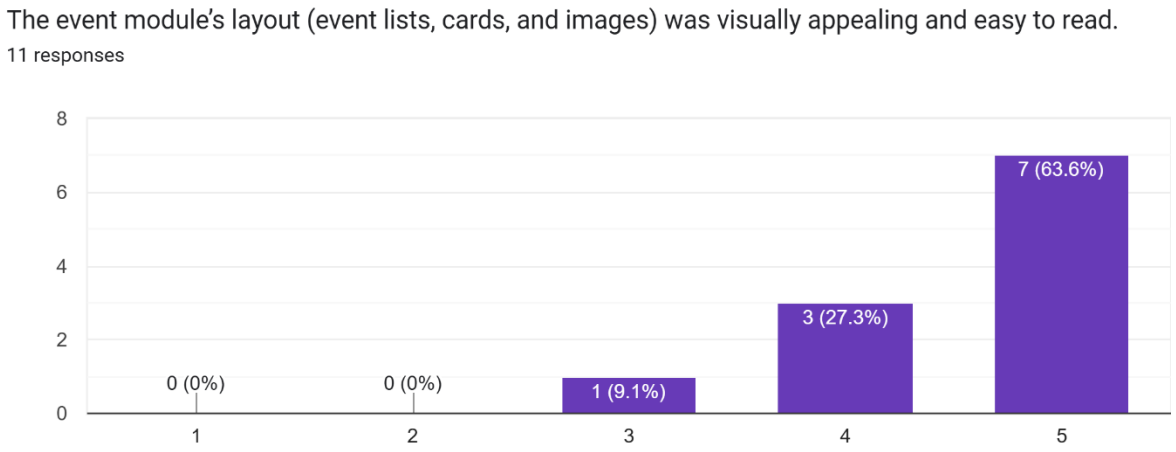


Figure 5.32: User feedback on the visual appeal and readability of the event module

As shown in Figure 5.33, 81.8% of respondents strongly agreed that the facility booking UI which combines a booking table, request form, and integrated calendar was easy to use and well-structured. One user each rated it a 3 and 4, which could reflect uncertainty with time

range inputs or date selections. Nonetheless, the data supports that the booking interface is visually and functionally effective for most users.

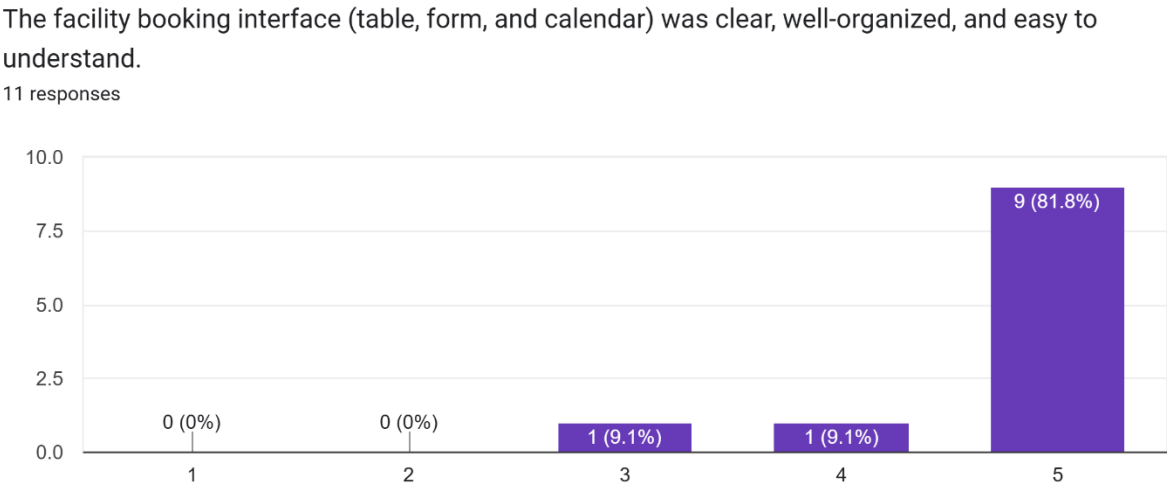


Figure 5.33: User feedback on clarity and organization of the facility booking interface

Figure 5.34 demonstrates that the majority of users (81.8%) rated the voting module a 5. This indicates strong approval of the way vote options and results were structured and displayed. One user each rated it 3 and 4, possibly due to first-time interaction with vote confirmation modals or time-based constraints. Overall, the module's simplicity and immediate results view were well received by most of the users.

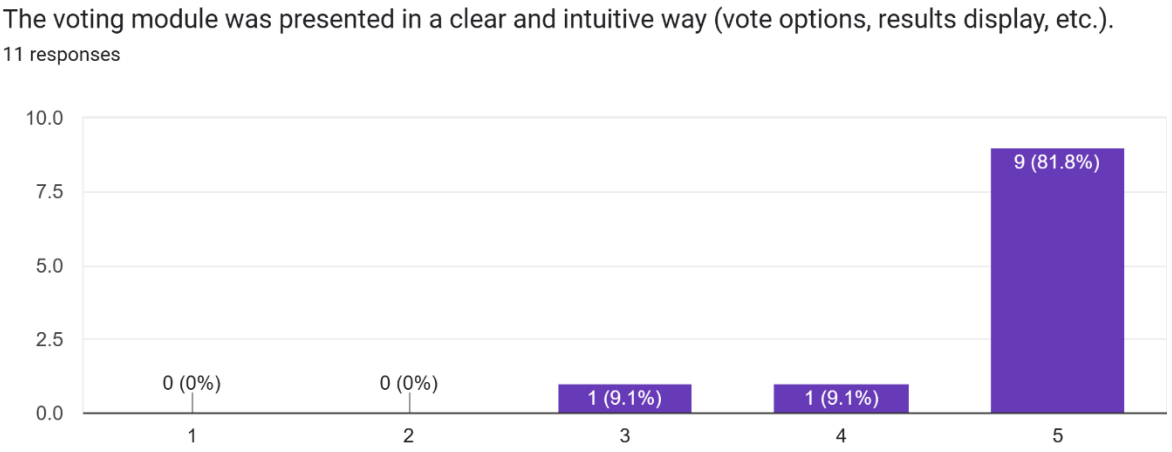


Figure 5.34: User feedback on layout and usability of the voting module

According to Figure 5.35, 9 out of 11 users (81.8%) rated the Communication Hub with the highest score. With only one user each giving a 3 and 4, the feedback suggests that the forum-style layout with modular comment threads and nested replies was mostly intuitive. The visual consistency across posts and announcements was also appreciated, affirming the design choices made for the community discussion space.

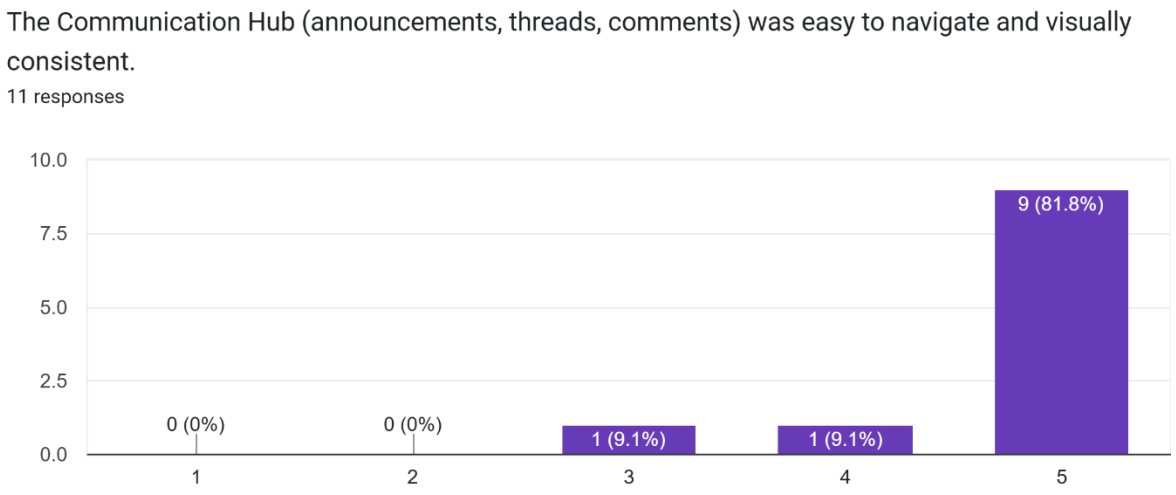


Figure 5.35: User feedback on ease of use and consistency in the Communication Hub

As shown in Figure 5.36, 81.8% of respondents rated this aspect a 5, indicating that the presentation of financial data including pie charts, categorized tables, and summaries was easy to understand. Two users gave a 3, possibly reflecting unfamiliarity with financial terminology or layouts. Overall, the data affirms that the visual formatting in the financial module is accessible and intuitive for most users.

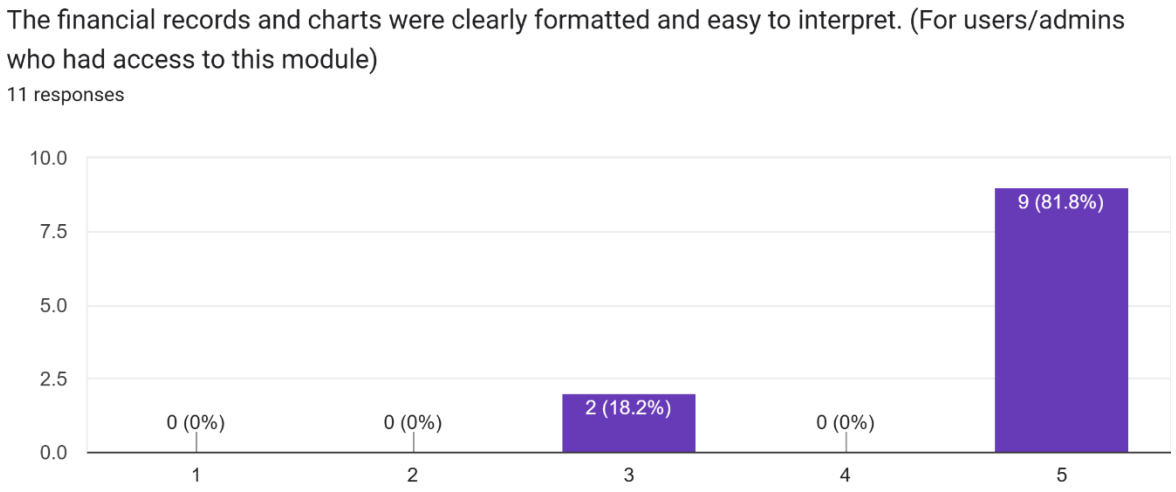


Figure 5.36: User feedback on clarity and formatting of financial records and charts

Figure 5.37 shows that 9 out of 11 respondents (81.8%) agreed strongly that the design system was consistent. This includes visual elements such as color schemes, button behavior, card layouts, and icon styles. One user each gave a 3 and 4, which may suggest that minor inconsistencies were noticed. However, the overall response supports that the platform maintains strong UI cohesion throughout.

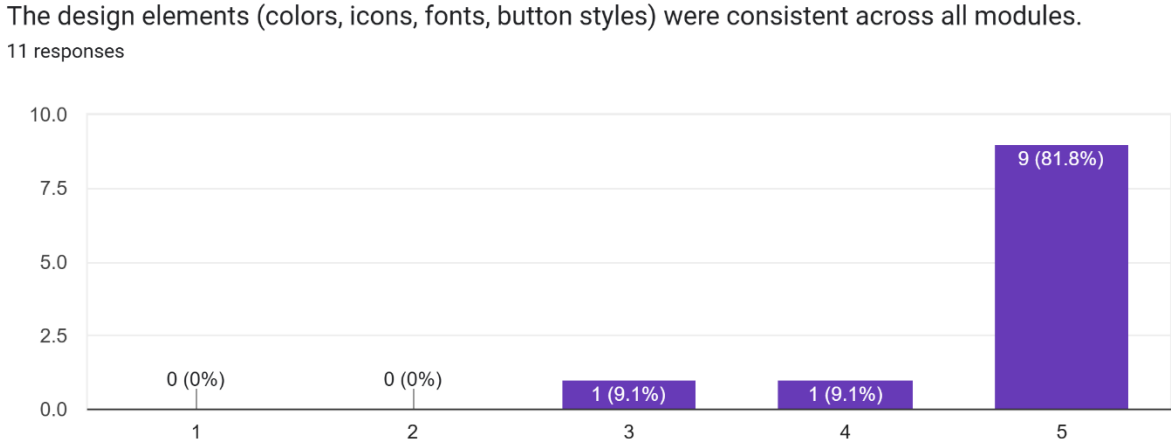


Figure 5.37: User feedback on design consistency across modules

According to Figure 5.38, 90.9% of users gave the highest rating, confirming that the dark mode was aesthetically pleasing and comfortable to use. One user selected 3, potentially

due to personal preference or display calibration issues. Regardless, the positive feedback confirms that the dark mode meets both visual comfort and accessibility standards.

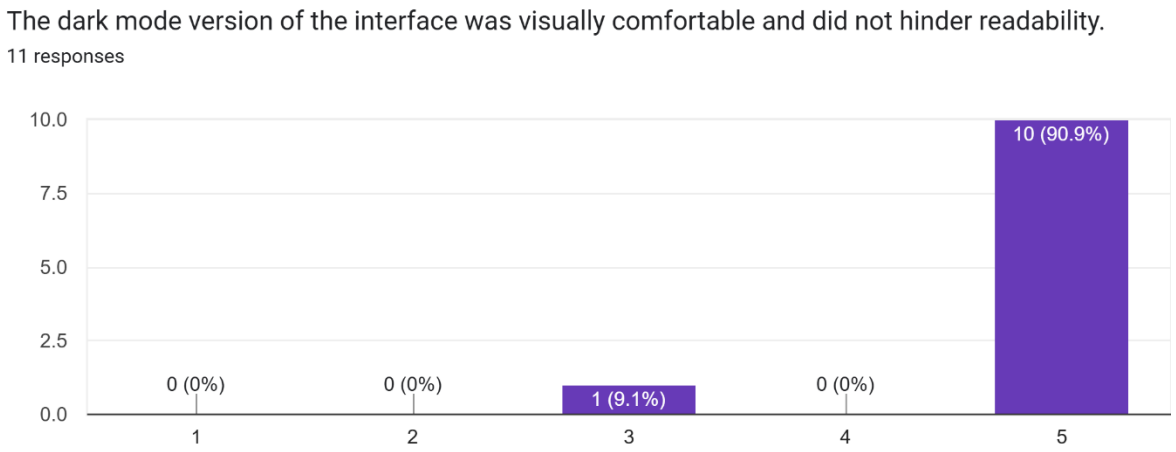


Figure 5.38: User feedback on comfort and readability of dark mode interface

As shown in Figure 5.39, 81.8% of users gave the highest possible score, affirming that the system's visual style felt professional and appropriate for a community environment. One user each gave a rating of 3 and 4, which may reflect individual preference rather than usability concerns. Overall, this feedback reinforces the design's success in balancing modern aesthetics with functional simplicity.

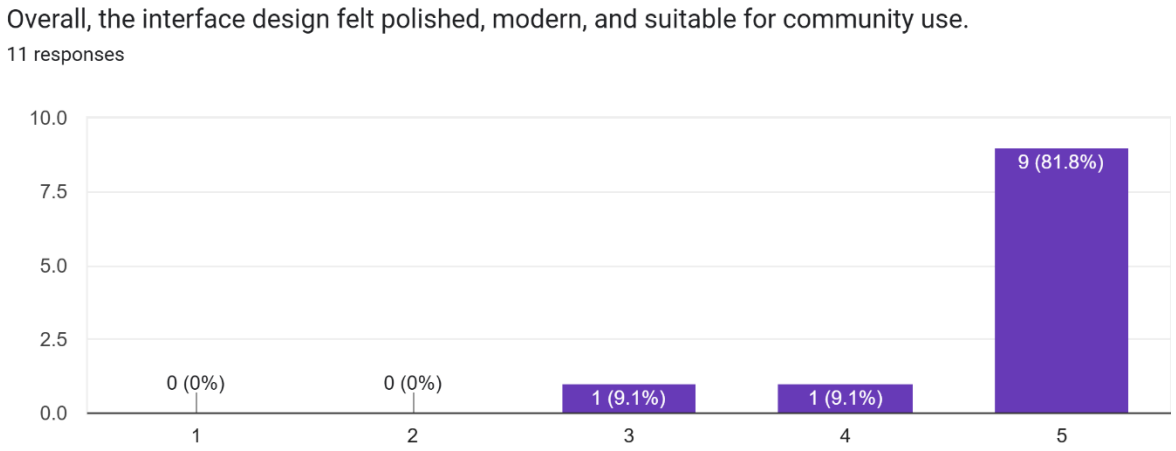


Figure 5.39: User feedback on the overall quality and suitability of the interface design

5.4 Summary

This chapter has presented a comprehensive evaluation of the Smart Community Platform for Taman Suria Tropika, focusing on both functional and non-functional aspects of system quality. Unit testing was carried out using the Pest framework to verify individual controller methods across the modules of the system. System testing followed, evaluating the platform's end-to-end workflows through test scenarios that confirmed the system behaved correctly in real-world usage contexts. In addition, non-functional testing was conducted, with a strong emphasis on usability. A Google Forms-based questionnaire was distributed to users to gather feedback on interface clarity, ease of navigation, performance responsiveness, and overall satisfaction. The results, supported by detailed charts and user opinions, demonstrated that the system was intuitive, user-friendly, and well-suited to be used by a community. These testing activities confirm that the platform meets its functional requirements and delivers a usable and engaging experience for both residents and administrators.

CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Introduction

This chapter provides a summary of the development and outcomes of the Smart Community Platform for Taman Suria Tropika. It revisits the objectives set at the beginning of the project and reflects on the extent to which they have been achieved. The project also discusses existing limitations and outlines suggestions for future enhancement.

6.2 Objective Achievements

The objectives of the Smart Community Platform were established in Chapter 1 and served as the foundation for the project's planning and execution. Table 6.1 below summarizes each objective and how it was accomplished during development.

Table 6.1: Objective Achievement

Objective	**Achievement**
To study and analyse existing systems and methods used for event management, financial tracking, facility booking, voting, and communication, in order to identify key features and areas for improvement.	A review of similar platforms and systems was conducted to gather best practices. The system was planned based on identified gaps and community-specific needs.
To design and implement a web-based management and communication system for Surau Al Ansar and Taman Suria Tropika as a whole, incorporating modules for event announcements, budget management, facility booking, communications, notifications, and voting.	A fully functional Laravel-based web system was developed, consisting of six integrated core modules alongside others, with proper access controls for users and admins.
To test and evaluate the functionality and usability of the developed system through user feedback and usability testing, ensuring that it meets the needs of the administrators and community members.	Comprehensive unit and system testing were conducted for this project. Moreover, usability testing via Google Forms indicated high satisfaction and ease of use, even among less tech-savvy users.

6.3 Project Limitations

While the project achieved its main goals, several limitations were encountered:

- Notification features were limited to internal displays; push notifications via email or SMS were not implemented.
- The system does not yet include interactive onboarding guides or tooltips, which could help elderly or less experienced users become familiar with the platform more easily.
- The current version lacks data analytics or reporting tools for tracking community engagement or financial summaries over time.
- Admin control is centralized and could benefit from more role-based breakdown to better delegate responsibilities.

6.4 Conclusion

The Smart Community Platform for Taman Suria Tropika successfully met its core objectives by providing an integrated web-based solution for community event management, financial tracking, facility booking, communication, notifications, and voting. The system was designed with both administrative and general users in mind, offering tailored access and functionality based on user roles. Through thorough testing and user feedback, the platform was found to be user-friendly, functional, and effective in promoting engagement within the residential community.

The modular nature of the system, combined with a clean and intuitive interface, positions it as a strong foundation for digital community management. While some limitations were encountered, these do not hinder the platform's overall value and impact. The outcomes of this project demonstrate the potential for similar platforms to be implemented in other residential communities aiming to modernize their communication and administrative processes.

6.5 Future Work

The platform holds potential for further development and scalability. Potential future enhancements include:

- Integration of real-time push notifications via email or SMS to improve user responsiveness and communication.
- Addition of interactive onboarding guides and tooltips to assist elderly or less experienced users.

- Real-time reporting and analytics features for event participation, financial performance, or voting turnout.
- Role-specific dashboards and permission refinement for sub-admins (e.g., treasurer-only or event manager-only roles).

REFERENCES

Amini, M., Rahmani, A., Abedi, M., Hosseini, M., Amini, M., & Amini, M. (2021, May 1). *Mahamgostar.com as a case study for adoption of Laravel framework as the best programming tools for PHP-based web development for small and medium enterprises.* SSRN. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3857736

Demchenko, A. (2020, May 26). What is Rapid Application Development? - nCube. *nCube* -. https://ncube.com/what-is-rapid-application-development

Kissflow, T. (2024, September 24). *Rapid Application Development (RAD) | Definition, Steps & Full Guide*. Kissflow, Inc. https://kissflow.com/application-development/rad/rapid-application-development/

Nursari, S. R. C., & Linuwih, P. (2021). The design of management information systems at Jami Kautsar Mosque. *Jurnal TAM (Technology Acceptance Model)*, *12*(1), 1–8.

Sotnik, S., Manakov, V., & Lyashenko, V. (2023). Overview: PHP and MySQL features for creating modern web projects. *International Journal of Academic Information Systems Research (IJAISR)*, 7(1), 11–17.

Setyawan, K. R. V., Rizal, M. F., Widodo, S., & Hikmawan, R. (2023). Design of Continuous Web APP: Guidance and Counseling Management Information System at SMKN 1 Purwakarta using Laravel Framework. *International Journal Software Engineering and Computer Science (IJSECS)*, *3*(3), 410–423. https://doi.org/10.35870/ijsecs.v3i3.1855

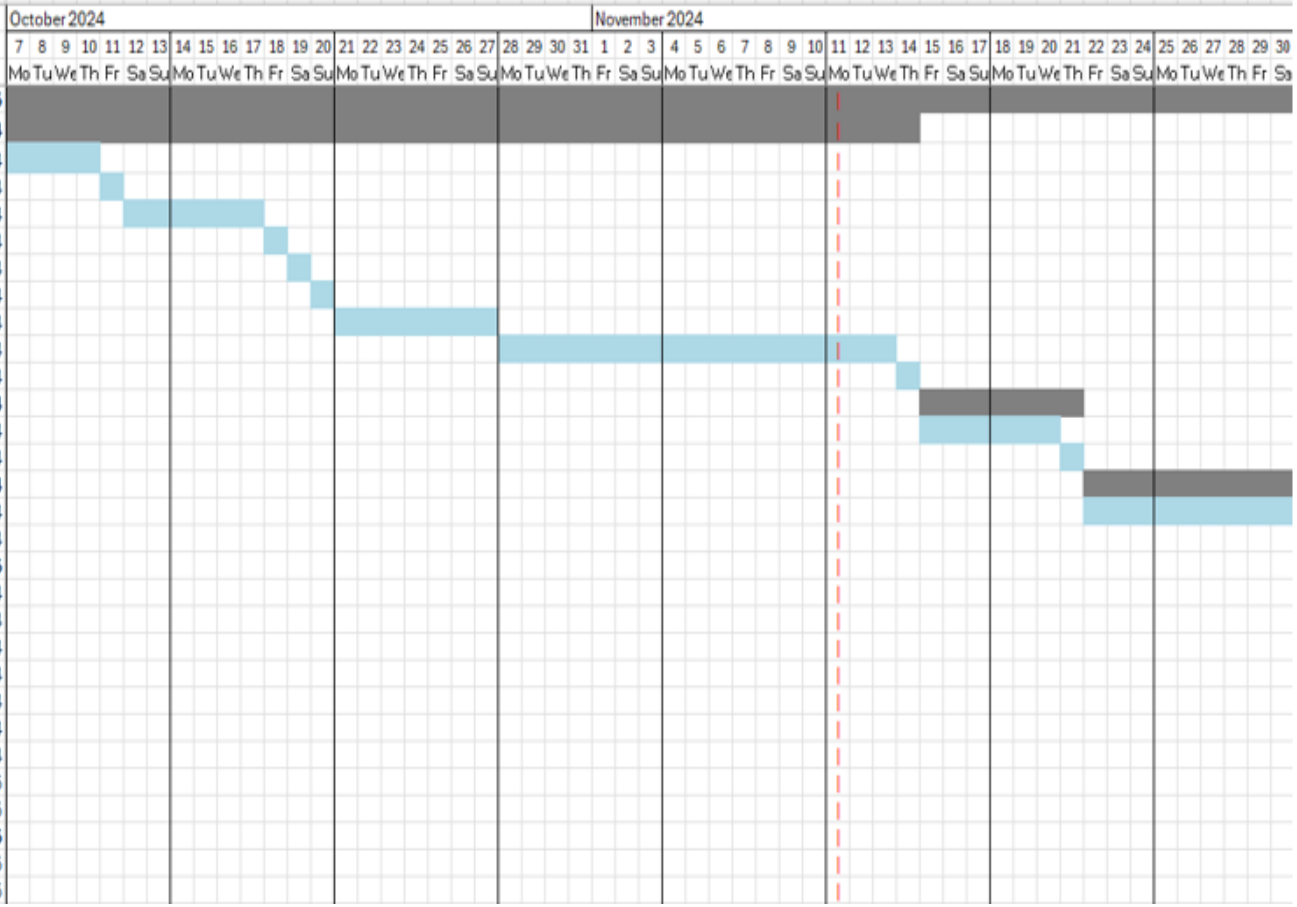
Sukya, F., & Nurfarida, E. (2024). Design and development a web platform portal for mosque financial management. *Journal of Advances in Information and Industrial Technology*, *6*(1), 83–92. https://doi.org/10.52435/jaiit.v6i1.599

Talib, M. N. H., & Mohamed, R. (2023, July 20). *Development of mosque management information System*. Universiti Tun Hussein Onn Malaysia. https://publisher.uthm.edu.my/periodicals/index.php/aitcs/article/view/7472

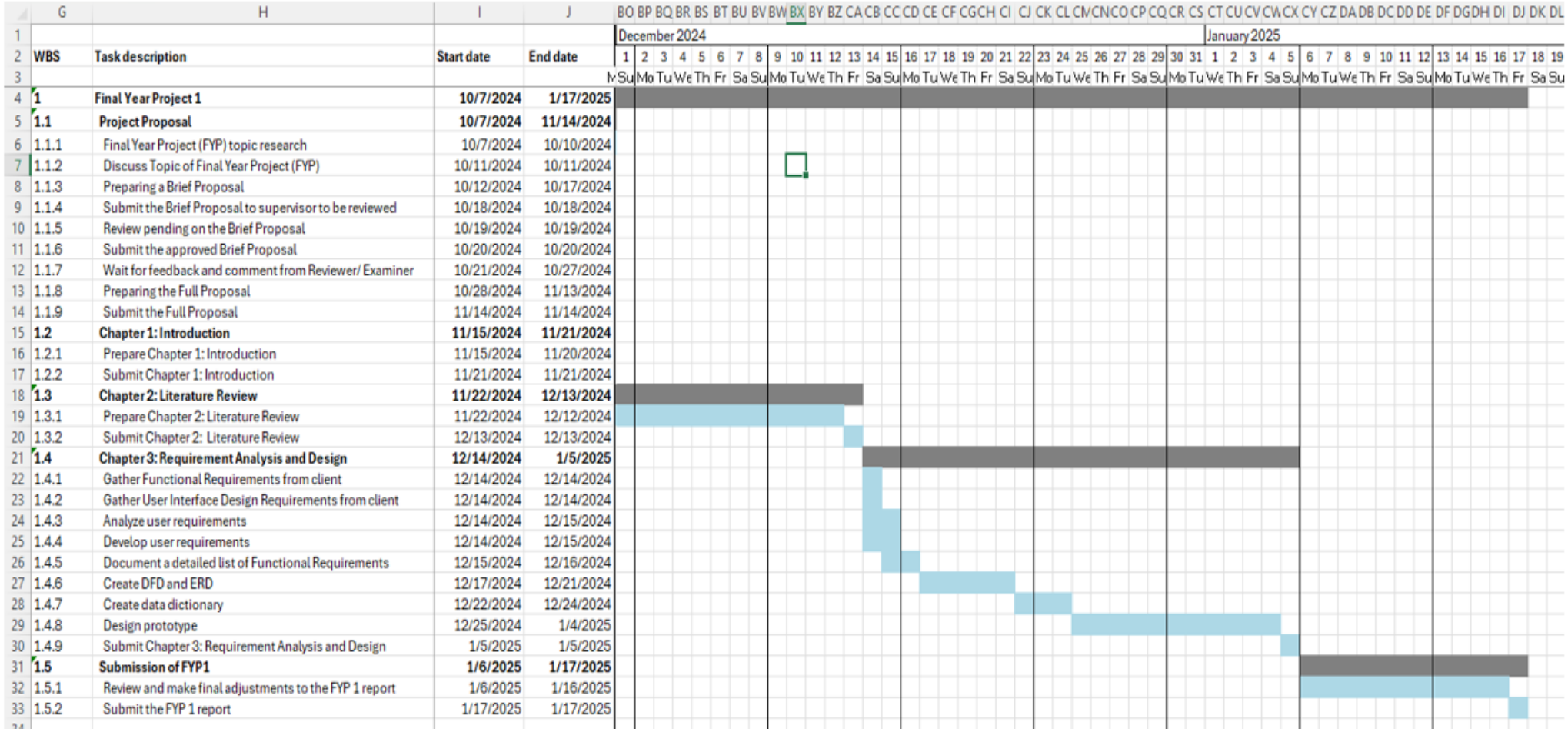
APPENDICES

Appendix A: Project schedule in Gantt chart (FYP 1, Part 1-November to December)

WBS	Task description	Start date	End date
1	**Final Year Project 1**	**10/7/2024**	**1/17/2025**
1.1	**Project Proposal**	**10/7/2024**	**11/14/2024**
1.1.1	Final Year Project (FYP) topic research	10/7/2024	10/10/2024
1.1.2	Discuss Topic of Final Year Project (FYP)	10/11/2024	10/11/2024
1.1.3	Preparing a Brief Proposal	10/12/2024	10/17/2024
1.1.4	Submit the Brief Proposal to supervisor to be reviewed	10/18/2024	10/18/2024
1.1.5	Review pending on the Brief Proposal	10/19/2024	10/19/2024
1.1.6	Submit the approved Brief Proposal	10/20/2024	10/20/2024
1.1.7	Wait for feedback and comment from Reviewer/ Examiner	10/21/2024	10/27/2024
1.1.8	Preparing the Full Proposal	10/28/2024	11/13/2024
1.1.9	Submit the Full Proposal	11/14/2024	11/14/2024
1.2	**Chapter 1: Introduction**	**11/15/2024**	**11/21/2024**
1.2.1	Prepare Chapter 1: Introduction	11/15/2024	11/20/2024
1.2.2	Submit Chapter 1: Introduction	11/21/2024	11/21/2024
1.3	**Chapter 2: Literature Review**	**11/22/2024**	**12/13/2024**
1.3.1	Prepare Chapter 2: Literature Review	11/22/2024	12/12/2024
1.3.2	Submit Chapter 2: Literature Review	12/13/2024	12/13/2024
1.4	**Chapter 3: Requirement Analysis and Design**	**12/14/2024**	**1/5/2025**
1.4.1	Gather Functional Requirements from client	12/14/2024	12/14/2024
1.4.2	Gather User Interface Design Requirements from client	12/14/2024	12/14/2024
1.4.3	Analyze user requirements	12/14/2024	12/15/2024
1.4.4	Develop user requirements	12/14/2024	12/15/2024
1.4.5	Document a detailed list of Functional Requirements	12/15/2024	12/16/2024
1.4.6	Create DFD and ERD	12/17/2024	12/21/2024
1.4.7	Create data dictionary	12/22/2024	12/24/2024
1.4.8	Design prototype	12/25/2024	1/4/2025
1.4.9	Submit Chapter 3: Requirement Analysis and Design	1/5/2025	1/5/2025
1.5	**Submission of FYP1**	**1/6/2025**	**1/17/2025**
1.5.1	Review and make final adjustments to the FYP 1 report	1/6/2025	1/16/2025
1.5.2	Submit the FYP 1 report	1/17/2025	1/17/2025

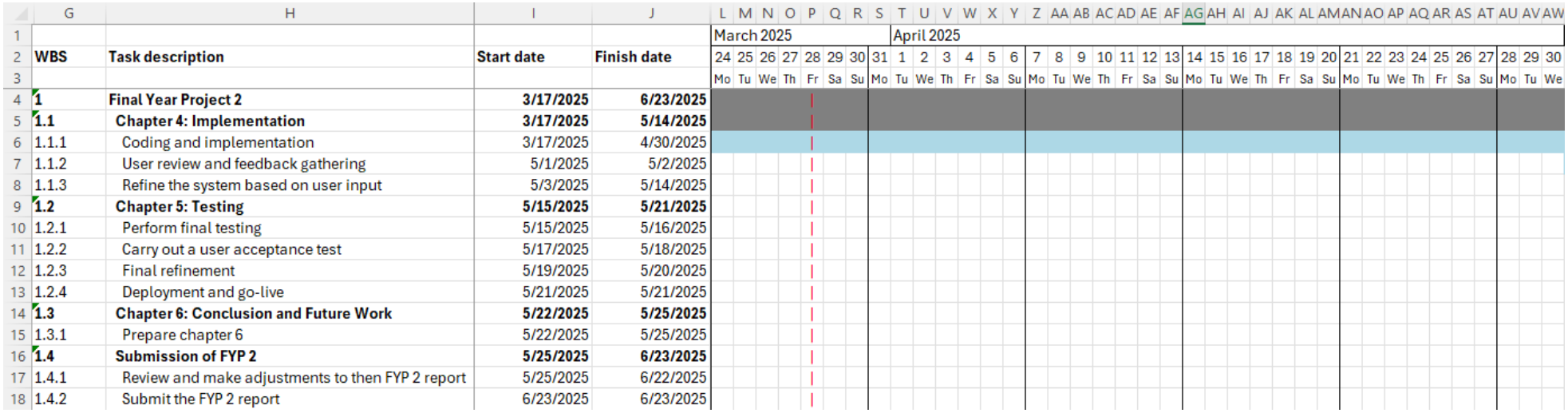


Appendix B: Project schedule in Gantt chart (Part 2-December to January)



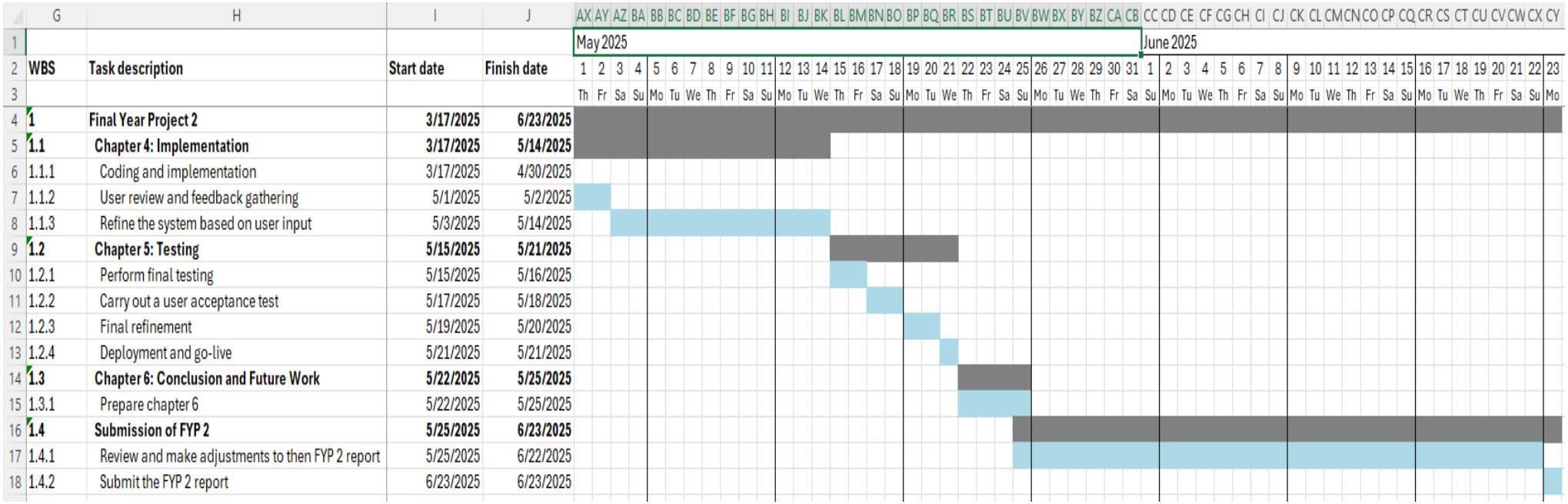
WBS	Task description	Start date	End date
1	**Final Year Project 1**	**10/7/2024**	**1/17/2025**
1.1	**Project Proposal**	**10/7/2024**	**11/14/2024**
1.1.1	Final Year Project (FYP) topic research	10/7/2024	10/10/2024
1.1.2	Discuss Topic of Final Year Project (FYP)	10/11/2024	10/11/2024
1.1.3	Preparing a Brief Proposal	10/12/2024	10/17/2024
1.1.4	Submit the Brief Proposal to supervisor to be reviewed	10/18/2024	10/18/2024
1.1.5	Review pending on the Brief Proposal	10/19/2024	10/19/2024
1.1.6	Submit the approved Brief Proposal	10/20/2024	10/20/2024
1.1.7	Wait for feedback and comment from Reviewer/ Examiner	10/21/2024	10/27/2024
1.1.8	Preparing the Full Proposal	10/28/2024	11/13/2024
1.1.9	Submit the Full Proposal	11/14/2024	11/14/2024
1.2	**Chapter 1: Introduction**	**11/15/2024**	**11/21/2024**
1.2.1	Prepare Chapter 1: Introduction	11/15/2024	11/20/2024
1.2.2	Submit Chapter 1: Introduction	11/21/2024	11/21/2024
1.3	**Chapter 2: Literature Review**	**11/22/2024**	**12/13/2024**
1.3.1	Prepare Chapter 2: Literature Review	11/22/2024	12/12/2024
1.3.2	Submit Chapter 2: Literature Review	12/13/2024	12/13/2024
1.4	**Chapter 3: Requirement Analysis and Design**	**12/14/2024**	**1/5/2025**
1.4.1	Gather Functional Requirements from client	12/14/2024	12/14/2024
1.4.2	Gather User Interface Design Requirements from client	12/14/2024	12/14/2024
1.4.3	Analyze user requirements	12/14/2024	12/15/2024
1.4.4	Develop user requirements	12/14/2024	12/15/2024
1.4.5	Document a detailed list of Functional Requirements	12/15/2024	12/16/2024
1.4.6	Create DFD and ERD	12/17/2024	12/21/2024
1.4.7	Create data dictionary	12/22/2024	12/24/2024
1.4.8	Design prototype	12/25/2024	1/4/2025
1.4.9	Submit Chapter 3: Requirement Analysis and Design	1/5/2025	1/5/2025
1.5	**Submission of FYP1**	**1/6/2025**	**1/17/2025**
1.5.1	Review and make final adjustments to the FYP 1 report	1/6/2025	1/16/2025
1.5.2	Submit the FYP 1 report	1/17/2025	1/17/2025

Appendix C: Project schedule in Gantt chart (FYP 2, Part 1-March to April)



WBS	Task description	Start date	Finish date
1	**Final Year Project 2**	**3/17/2025**	**6/23/2025**
1.1	**Chapter 4: Implementation**	**3/17/2025**	**5/14/2025**
1.1.1	Coding and implementation	3/17/2025	4/30/2025
1.1.2	User review and feedback gathering	5/1/2025	5/2/2025
1.1.3	Refine the system based on user input	5/3/2025	5/14/2025
1.2	**Chapter 5: Testing**	**5/15/2025**	**5/21/2025**
1.2.1	Perform final testing	5/15/2025	5/16/2025
1.2.2	Carry out a user acceptance test	5/17/2025	5/18/2025
1.2.3	Final refinement	5/19/2025	5/20/2025
1.2.4	Deployment and go-live	5/21/2025	5/21/2025
1.3	**Chapter 6: Conclusion and Future Work**	**5/22/2025**	**5/25/2025**
1.3.1	Prepare chapter 6	5/22/2025	5/25/2025
1.4	**Submission of FYP 2**	**5/25/2025**	**6/23/2025**
1.4.1	Review and make adjustments to then FYP 2 report	5/25/2025	6/22/2025
1.4.2	Submit the FYP 2 report	6/23/2025	6/23/2025

Appendix D: Project schedule in Gantt chart (FYP 2, Part 2-May to June)



WBS	Task description	Start date	Finish date
1	**Final Year Project 2**	**3/17/2025**	**6/23/2025**
1.1	**Chapter 4: Implementation**	**3/17/2025**	**5/14/2025**
1.1.1	Coding and implementation	3/17/2025	4/30/2025
1.1.2	User review and feedback gathering	5/1/2025	5/2/2025
1.1.3	Refine the system based on user input	5/3/2025	5/14/2025
1.2	**Chapter 5: Testing**	**5/15/2025**	**5/21/2025**
1.2.1	Perform final testing	5/15/2025	5/16/2025
1.2.2	Carry out a user acceptance test	5/17/2025	5/18/2025
1.2.3	Final refinement	5/19/2025	5/20/2025
1.2.4	Deployment and go-live	5/21/2025	5/21/2025
1.3	**Chapter 6: Conclusion and Future Work**	**5/22/2025**	**5/25/2025**
1.3.1	Prepare chapter 6	5/22/2025	5/25/2025
1.4	**Submission of FYP 2**	**5/25/2025**	**6/23/2025**
1.4.1	Review and make adjustments to then FYP 2 report	5/25/2025	6/22/2025
1.4.2	Submit the FYP 2 report	6/23/2025	6/23/2025

Appendix E: Interview Questions

The following are the questions asked and the collected answers of an interview with the secretary of Surau Al Ansar, Mr Azlan Ithnin. Take note that some editing and translation has been done to provide clarity.

1. **To my knowledge, a management system for mosques or a surau like Surau Al Ansar would usually have features such as Event Announcement, Budget Management, and a Notification page. So, for the system that I intend to create, do you find these features acceptable?**

   Answer: Good suggestions. I believe they will be useful. We mostly just rely on Whatsapp, Facebook, and Microsoft Excel for most matters.

2. **Could I ask you to elaborate on Surau Al Ansar's budgeting practices?**

   Answer: For this question, all payment comes from donation. Also, I keep track of the money manually using Microsoft Excel spreadsheets.

3. **Does Surau Al Ansar have a Funeral and Burial services?**

   Answer: No, we only pay for khairat (Funeral Assistance Fund). Sometimes, you can even see a QR code on the donation boxes that we pass around during prayer for the collection of it. People from Masjid Pulau Meranti are the ones that manages the rest.

4. **How do you feel about a Facility Booking system? Do you think it will be useful?**

   Answer: Well, if it is for the surau, then I do not think it will be that much needed. Almost every activity that uses the surau's space gets conducted and overseen by us, even when guests from outside come to give sermons. I believe sticking to just the Event Announcement is fine.

5. **Based on my observation, the surau usually has several choices in terms of what food gets to be served after khutbah or Friday prayer. While the selection can be determined by votes, not everyone gets to participate with the voting on what gets to be served. Do you believe a voting system will be helpful for this issue and possibly other matters, as well?**

   Answer: Yes, I think it will be helpful. I do not see any problem with adding it into the system.

6. **From what we have talked about, which features, or feature, will be most important to you and should take priority?**

   Answer: I definitely see the one relating to finance being the most important and to be prioritized at this point of time. But I am looking forward to everything else that the system will offer.

7. **Any additional suggestions?**

   Answer: No suggestions.

The following are the questions asked and the collected answers of an interview with a member of the community, Ms. Amirah. Take note that some editing and translation has been done to provide clarity.

1. **Based on my observation, the community coordinates activities together through Whatsapp. What challenges or limitations have you experienced when using WhatsApp for this purpose, and how do you think a dedicated Communication Hub could address these issues?**

   Answer: Well, I am very used to Whatsapp and have found no issues with it. It is just a matter of paying attention to the messages that gets sent into the group, and we (inhabitants of Taman Suria Tropika) would usually already know when it is time to

coordinate big activities together, or at least prepare our own stuff, just like Hari Merdeka last time. But, I do think a Communication Hub might still be helpful for gathering attention and giving out announcements, in a way that we could use it like Facebook, and showcase the food or items that we intend to sell before the events or during other occasions. Maybe even have a better chance at clueing more people in on certain competitions or showcase the winners of those competitions after the event.

2. **Besides announcement and coordination, what other purposes do you think a Communication Hub could serve?**

   Answer: Staying up to date with issues that happens around there community. In fact, there was a break-in case last year in our area, I think. So, the Communication Hub can be used to highlight the event and make people aware of it whenever they use the system.

3. **How do you feel about a Facility Booking system? Do you think it will be useful? For context, perhaps it can be used to book the park/playground location?**

   Answer: Maybe. We do have that one brand-new pavilion area and a store with items for events. Perhaps the booking system will be useful for ensuring that people will not argue over who gets to occupy those stuff by booking beforehand. It can even be used to keep track of usage.

4. **From what we have talked about, which features, or feature, will be most important to you and should take priority?**

   Answer: Definitely the Communication Hub.

5. **Any additional suggestions or more to say?**

   Answer: Nothing.

Appendix F: Usability Testing Questionnaire

Usability Testing – Smart Community Platform for Taman Suria Tropika

Hello and thank you for taking the time to participate in this usability testing session. My name is **Ibnu Firas bin Ahmad Fadzuli**, a final-year Software Engineering student at **Universiti Malaysia Sarawak (UNIMAS)**. This usability test is part of my Final Year Project, which involves the development of a web-based system titled **"Smart Community Platform for Taman Suria Tropika."** The system is designed to support community members and administrators in various tasks such as managing events, booking facilities, voting on community matters, tracking finances, and enhancing overall communication within the residential area.

Your honest feedback through this form will help me evaluate the platform's functionality, interface usability, and overall user experience. I sincerely appreciate your time and support in helping me improve the system. Thank you!

Section 1: Participant Information

1. Name

_____

2. Email (optional)

_____

3. How familiar are you with using community or management systems like this one?

Mark only one oval.

- ◯ Not familiar
- ◯ Somewhat familiar
- ◯ Very familiar

Section 2: System Functionality

Label scale: 1 = Strongly Disagree, 5 = Strongly Agree

4. I was able to log in and access the system without problems.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

5. The dashboard provided clear options for me to navigate the system.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

6. I was able to view and understand event information easily.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

7. I was able to submit a facility booking request without difficulty.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

8. I was able to cast my vote in a voting session without confusion.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

9. I was able to view announcements and participate in discussions or comments comfortably.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

10. I was able to understand the financial records or summaries displayed in the system.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

11. The system provided clear feedback after I submitted a form or performed an action (such as submitting a booking, posting a comment, creating an event, or approving a request).

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

12. The performance of the system was smooth and responsive, without significant delays or errors.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

Section 3: User Interface Design

Use a 1–5 scale: 1 = Strongly Disagree, 5 = Strongly Agree

13. The overall layout and design of the system were visually clear and easy to understand.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

14. The dashboard interface was clean, organized, and helped me find what I needed quickly.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

15. The event module's layout (event lists, cards, and images) was visually appealing and easy to read.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

16. The facility booking interface (table, form, and calendar) was clear, well-organized, and easy to understand.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

17. The voting module was presented in a clear and intuitive way (vote options, results display, etc.).

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

18. The Communication Hub (announcements, threads, comments) was easy to navigate and visually consistent.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

19. The financial records and charts were clearly formatted and easy to interpret. *(For users/admins who had access to this module)*

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

20. The design elements (colors, icons, fonts, button styles) were consistent across all modules.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

21. The dark mode version of the interface was visually comfortable and did not hinder readability.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

22. Overall, the interface design felt polished, modern, and suitable for community use.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

Section 4: Overall Satisfaction

Use a 1–5 scale (1 = Strongly Disagree, 5 = Strongly Agree)

23. I am satisfied with my overall experience using the system.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

24. I would feel comfortable using this system regularly in a real community setting.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

25. The system meets the needs of a residential community like Taman Suria Tropika.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

26. I was able to learn how to use the system without much guidance or help.

Mark only one oval.

1	2	3	4	5
◯	◯	◯	◯	◯

Section 5: Open Feedback

Almost there! Feel free to answer however you like. Thank you for your time.

27. What did you like most about the system?

28. What features or modules were difficult or confusing to use?

29. Do you have any suggestions for improving the system's usability or appearance?

30. Any other comments or feedback?