AI MEETS INVENTORY: EMPOWERING STOCK MANAGEMENT WITH REAL-TIME ANALYSIS POWERED BY AI

Gerrard Keneth Sahol

Bachelor of Computer Science with Honours

(Network Computing)

2025

AI MEETS INVENTORY: EMPOWERING STOCK MANAGEMENT WITH REAL-TIME ANALYSIS POWERED BY AI

GERRARD KENETH SAHOL

This project is submitted in partial fulfilment of

the requirements for the degree of Bachelor of Computer Science with Honours

(Network Computing)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

KECERDASAN BUATAN BERTEMU INVENTORI: MEMPERKASA PENGURUSAN STOK DENGAN ANALISIS MASA NYATA DIKUASKAN OLEH KECERDASAN BUATAN

GERRARD KENETH SAHOL

Projek ini merupakan salah satu keperluan untuk

Ijazah Sarjana Muda Sains Komputer dengan Kepujian

(Pengkomputeran Rangkaian)

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2025

DECLARATION

I hereby declare that the thesis of AI meets Inventory: Empowering Stock management with Real-Time Analysis Powered By AI is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that no portion of the work referred in this report has been submitted in support of an application for another degree at University Malaysia Sarawak (UNIMAS).

Signature

Name: Gerrard Keneth Sahol

Matric No.: 78000

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 23/6/2025

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to Dr. Adnan Shahid Khan, my Final Year Project supervisor, for his unwavering support, valuable insights, and constructive feedback throughout the development of this project. His guidance played a vital role in shaping the direction and quality of my work, and I am sincerely thankful for his commitment and encouragement.

I would also like to extend my appreciation to Dr. Chew Kim Mey, who served as the examiner for this project. Her thoughtful evaluation and professional critique contributed significantly to the refinement and improvement of the system.

In addition, I would like to thank the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak, for providing the necessary resources and learning environment that enabled the successful completion of this project.

Lastly, I am grateful to my peers, friends, and family for their moral support and encouragement throughout this academic journey. Their presence and positivity gave me the motivation to stay focused and persistent during the challenging phases of this project.

Abstract

Effective inventory management is vital for small to medium-sized businesses, yet many rely on manual or spreadsheet-based systems, leading to errors and inefficiencies. This project introduces an AI-Enhanced Inventory Management System that automates stock tracking and improves decision-making. The system integrates PHP, MySQL, JavaScript, and Tailwind CSS for development, with the OpenAI API powering an AI-driven chatbot for real-time user interaction. Following an Agile methodology, the project emphasizes iterative development, ensuring the system adapts to user needs. Key functionalities include real-time stock updates, predictive analytics for stock trends, and role-based access control. The system is rigorously tested for performance, reliability, and usability to meet both functional and non-functional requirements. Ultimately, the deployment of this system aims to enhance inventory accuracy, reduce margin of error, and provide businesses with actionable insights, contributing to more efficient and effective inventory management practices.

Abstrak

Pengurusan inventori yang berkesan adalah penting bagi perniagaan kecil dan sederhana, namun ramai yang masih mengguna sistem manual atau berasaskan hamparan, hal ini menyebabkan kesilapan dan tidak cekap. Projek ini memperkenalkan Sistem Pengurusan Inventori berasaskan AI yang mengautomasikan penjejakan stok dan meningkatkan pembuatan keputusan. Sistem ini mengintegrasikan PHP, MySQL, JavaScript, dan Tailwind CSS untuk pembangunan, dengan API OpenAI yang menyokong chatbot berasaskan AI untuk berinteraksi dengan pengguna dalam masa segera. Mengikuti metodologi Agile, projek ini menekankan pembangunan iteratif, memastikan sistem menyesuaikan diri dengan keperluan pengguna. Fungsi utama termasuk mengemas kini stok secara segera, analitik ramalan untuk trend saham, dan kawalan akses berasaskan peranan. Sistem ini diuji secara menyeluruh untuk prestasi, kebolehpercayaan, dan kebolehgunaan untuk memenuhi keperluan berfungsi dan bukan berfungsi. Akhirnya, pelaksanaan sistem ini bertujuan untuk meningkatkan ketepatan inventori, mengurangkan tahap ralat, dan menyediakan perniagaan dengan pandangan yang boleh diambil tindakan, menyumbang kepada amalan pengurusan inventori yang lebih cekap dan berkesan.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Introduction

Efficient stock and equipment tracking is essential for businesses to ensure that items are available when needed, while avoiding overstocking and understocking issues. Many small and medium-sized businesses rely on manual or spreadsheet-based systems for inventory tracking. This leads to frequent errors, lack of real-time visibility, and inefficiencies, which can disrupt supply and impact customer satisfaction.

By integrating AI technology into inventory tracking, this project aims to automate stock monitoring and provide real-time insights into stock levels within storage facilities. The system will help businesses reduce the time spent on manual calculations and make it easier to maintain optimal inventory levels. With the help of Generative AI, users can anticipate low-stock events, optimize stock levels, and simplify the data presentation using an AI-powered chatbot interface.

1.2 Problem Statement

Traditional inventory tracking methods, such as Excel spreadsheets or manual logs, are time-consuming and prone to human error, leading to inaccurate stock management and revenue losses. For instance, according to the National Retail Federation, a total of 1.44% of all sales are actually lost to inventory mismanagement. Furthermore, smaller businesses without dedicated inventory systems find it challenging to implement automated solutions, often due to cost or complexity.

This project plans on addressing these issues by minimizing the time and effort needed for stock management by automating the usual manual inventory tracking procedures. The system will increase accuracy, reducing human error and provide real-time stock level updates with the integration of AI. With predictive analytics providing insights into stock patterns and low-stock warnings prompting swift restocking initiatives, these improvements will enable users to make more informed decisions, ultimately enhancing inventory management and company operations.

1.3 Scope

This project will focus on developing an inventory tracking with chatbot capabilities tailored for internal stock management within small to medium-sized businesses. The solution will provide real-time tracking and alerts for low-stock levels, optimizing stock levels without handling external procurement or order management. Key functionalities include:

- Real-time stock monitoring to reduce manual errors and enhance inventory accuracy
- AI-powered chatbot integration to automate inventory-related queries
- Role-based access control to enhance security by restricting user permissions based on roles.

1.4 Objectives

The primary aim of this project is to enhance inventory tracking through AI integration, allowing businesses to maintain optimal stock levels efficiently. The objectives include:

- Enhance inventory accuracy by implementing a real-time stock tracking system that reduces manual errors
- Improve user interaction through AI-powered chatbot that enables inventory-related queries to be answered automatically, reducing time spent on manual stock searches
- Strengthen system security by implementing role-based access control to prevent unauthorized stock withdrawals, reducing security incidents

This Project is primarily aimed at small to medium enterprises such as grocery stores, apparels shop, electronic retailers and other enterprises needing efficient internal stock management. By focusing on these enterprises, the system aims to address common inventory challenges such as overstocking, understocking, and the lack of real-time visibility, which are prevalent in retail environments.

1.5 Brief Methodology

The project will be executed in the following phases:

a) **Requirements Analysis**: Define specific requirements by gathering data on the challenges faced by small and medium-sized businesses in inventory tracking. Establish user needs for an AI-driven solution.

b) **Design**: Develop the architectural layout of the AI-powered inventory tool, with a focus on modularity and scalability. Define the UI and chatbot interface for usability.

c) **Development**: Build the inventory tracking tool using PHP, MySQL, and JavaScript for the backend and frontend, incorporating AI-based analytics for stock prediction.

d) **Testing and Validation**: Test each component, including real-time tracking, chatbot responses, and alert functionalities, to ensure system reliability and accuracy.

e) **Deployment and Training**: Deploy the system in a simulated environment and provide training for end-users to familiarize them with the chatbot interface and AI-generated insights.

f) **Evaluation**: Gather user feedback and conduct performance evaluations to refine the tool.

1.6 Significance of the Project

The proposed inventory tracking solution addresses a critical need for small and medium-sized businesses to efficiently manage their stock levels without relying on costly, complex inventory systems. By introducing AI-based analytics, this project will provide these small to medium businesses with a powerful tool to automate stock tracking, generate reports, and maintain optimal inventory levels. The inclusion of a chatbot interface will make it accessible even for users with limited technical expertise, enhancing usability and minimizing time spent on inventory-related queries.

1.7 Project Schedule

Task	Start Date	End Date	Duration (Days)
Submission of Approved Brief Proposal	7/10/2024	20/10/2024	14
Feedback and comment from Reviewer/Examiner	21/10/2024	28/10/2024	8
Submission of Full Proposal	29/10/2024	14/11/2024	17
Submission of Chapter 1	15/11/2024	21/11/2024	7
Submission of Chapter 2	22/11/2024	13/12/2024	22
Submission of Chapter 3	14/12/2024	5/1/2025	23
Submission of FYP 1 Final Report and Paper	6/1/2025	17/1/2025	12

Figure 1: Table of project Schedule

1.8 Expected Outcome

The project will deliver a robust AI-enhanced inventory tracking tool tailored for internal stock management. Expected outcomes include:

- Developing a user-friendly interface that offers real-time stock visibility, reducing inventory check times by 20%.
- Implement predictive analytics to forecast stock trends, enabling businesses to reduce stockouts by 10%.
- Create automated low-stock alerts to improve restocking efficiency, cutting restocking delays by 30%.
- Introduce an AI-powered chatbot for quick and interactive stock queries, decreasing the need for manual searches by 60%.
- Establish role-based access control to enhance security, reducing unauthorized item withdrawals by 50%.

Chapter 2: Literature Review

2.1. Introduction

The aim for this literature review is to find and make comparisons to similar research to proposed Final Year Project, “AI Meets Inventory: Empowering Stock Management with Real-Time analysis powered By AI”. In this chapter, similar works that integrate generative AI to assist in completing tasks will be reviewed.

What exactly is generative AI? According to Feuerriegel, S., et al. (2024), generative AI are computational methods that can produce seemingly original, meaningful content—like writing, pictures, or audio from training data. The way that we work and interact with one another is now being revolutionized by the widespread distribution of this technology, demonstrated by instances such as Copilot, GPT-4, and Dall-E 2. Aside from being used as a means for artistic purposes to create new texts or illustrations mimicking certain authors or artists, they can also assist as an intelligent question-answering system. (Feuerriegel, S., et al. 2024), which would be the main quality that will be implemented in my project.

In this literature review, the second section of this chapter is dedicated to reviewing similar works in terms of usage related to the proposed project. And then after the reviewed works will be further summarized in the third part.

2.2 Review of Related Works

This section is dedicated to reviewing existing research employing similar approaches to the proposed project. Each of the reviewed research addresses the contexts, the goals or tasks they wish to accomplish, the technology and methods used to accomplish these goals, and the findings and analysis with limitations when available will all be taken into consideration when analyzing the reviews that follow.

2.2.1 Generative AI in Supply Chain Management

According to Shekhar A., et al. (2023), the foundation of modern commerce hinges on the supply chain management handling aspects such as production, delivery, distribution, and procurement. Due to evolving technologies, the technological shift now focuses on the usage of AI, more specifically, generative AI. Shekhar A., et al. (2023) has further explains that AI enhances systems' capabilities for learning, analysis, and adaptation, leading to improved decision-making.

Based on literature review and case studies done by the authors, they have concluded that the usage of generative AI particularly in Supply Chain Management has enhanced efficiency, where generative AI significantly improves the operational efficiencies by analyzing vast amount of data to quickly extract data, leading to better decision-making and more streamlined processes across various supply chain functions, like procurement and logistics.

Aside from that, the AI ability to analyze historical data and pattern identification has allowed businesses to have a better insight or forecast of future demand with greater accuracy. This predictive capability helps in keeping an optimized inventory level, ensuring the right amount of stock is available at the right time. By reducing excess stock and minimizing

stockouts, businesses can improve operational efficiency and reduce costs, leading to a more streamlined supply chain.

Moreover, The integration of AI enables supply chains to be more flexible and responsive to the ever-changing market conditions. Real-time data analysis allows companies to quickly adjust production schedules, manage inventory more effectively, and optimize routing. This enhanced adaptability ensures that businesses can better meet customer demands and read the market fluctuations, maintaining a competitive edge.

Finally, Generative AI assists in identifying potential risks and disruptions within the supply chain. By simulating various scenarios, organizations can develop strategies to mitigate risks such as supplier insolvency, natural disasters, and other unforeseen events. This proactive approach to risk management enhances the resilience of supply chains, ensuring continuity and reducing the impact of potential disruptions.

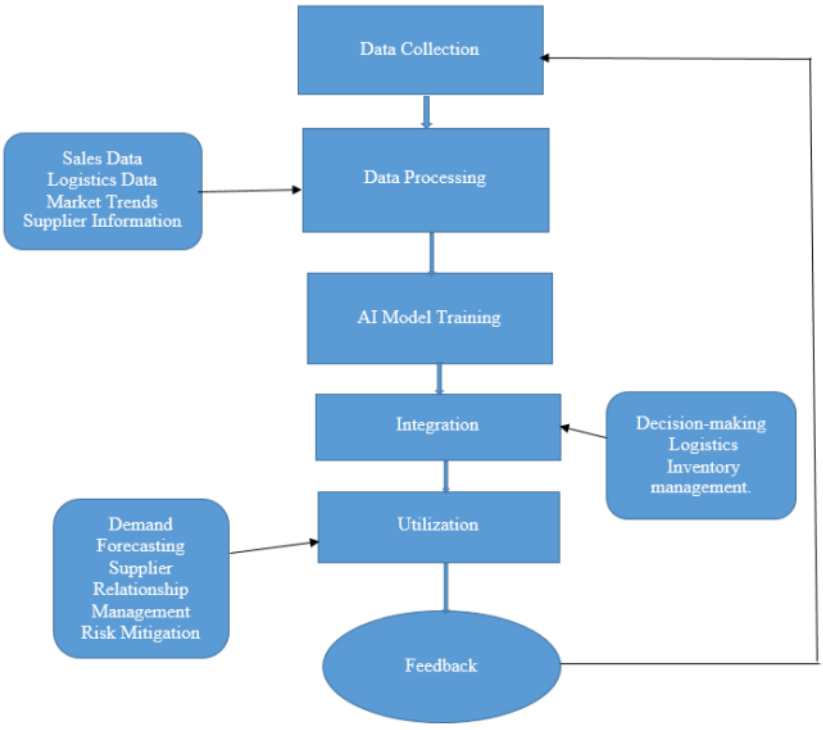

Figure 2: The Effects of AI adoption in Supply Chain Management

While Shekhar A., et al. (2023) has noted that there are a lot of benefits of AI implementation, it also comes with its own set of challenges, mainly in form of integration and quality assurance in supply chains due to the sheer volume of data generated across various nodes, often spread across different systems. Gathering this data proves to be difficult, as there are many sources that it relies on, such as suppliers, logistics providers, and retailers, each using their own formats and technologies. Thus, ensuring the data consistency and accuracy becomes an important task, as any differences in the result can lead to inefficiencies, errors in decision-making, and potential disruptions. Effective data integration and good quality assurance processes are essential to maintain the integrity of supply chain operations, enabling better insights and smoother coordination across the network.

To conclude their research, Shekhar A., et al. (2023) has emphasized that the adoption of Generative Artificial Intelligence (AI) in supply chain management will mark a significant turning point in modern trade. It highlights the revolutionary potential of AI to transform supply chain operations across various areas, including procurement, logistics, risk management, and inventory optimization. Generative AI has been noted to provide valuable insights into supplier behavior, operational complexities, and demand forecasts through predictive analytics and dynamic decision-making. this partnership between supply chain management and Generative AI fosters resilience and adaptability, enabling firms to proactively manage crises and seize opportunities. Ultimately, the integration of AI is expected to enhance productivity, flexibility, and creativity within supply chains, leading to a technological revolution in how companies operate in a constantly changing global economy.

2.2.2 Applications of Artificial Intelligence in Inventory Management: A Systematic Review of the Literature

Based on the study done by Albayrak Ünal, Ö, et al. (2023), they employed a bibliometric analysis to find out the frequency of usage for the interest of application of artificial intelligence (AI), particularly machine learning and deep learning used in inventory management for the past decade. The authors found that these AI techniques are increasingly recognized for their potential to enhance the efficiency and effectiveness of inventory management processes, leading to improved demand forecasting, reduced operational costs, and faster response times to customer needs.

Their research has noted that the most common type of AI implemented in inventory management is on machine learning (ML) and deep learning (DL). These methods are recognized and used for their ability to analyze large datasets and improve demand forecasting accuracy. They have also addressed several inventory-related challenges, including optimization, inventory control, and classification. Researchers have explored how AI can provide solutions to these problems, enhancing decision-making processes.

The research methodology employed by Albayrak Ünal, Ö, et al. (2023) involves a systematic literature review, designed to provide a comprehensive and structured analysis of existing research on AI applications in inventory management. It starts with a research question where they formulate specific research question to guide review processes, then they identify relevant articles published within the last decade, aimed specifically at articles that focus on AI in inventory management. They then performed a bibliometric analysis to quantitatively assess the characteristics of the selected studies. This included examining publication trends, source distribution, geographic distribution, and keyword analysis to visualize the research landscape.

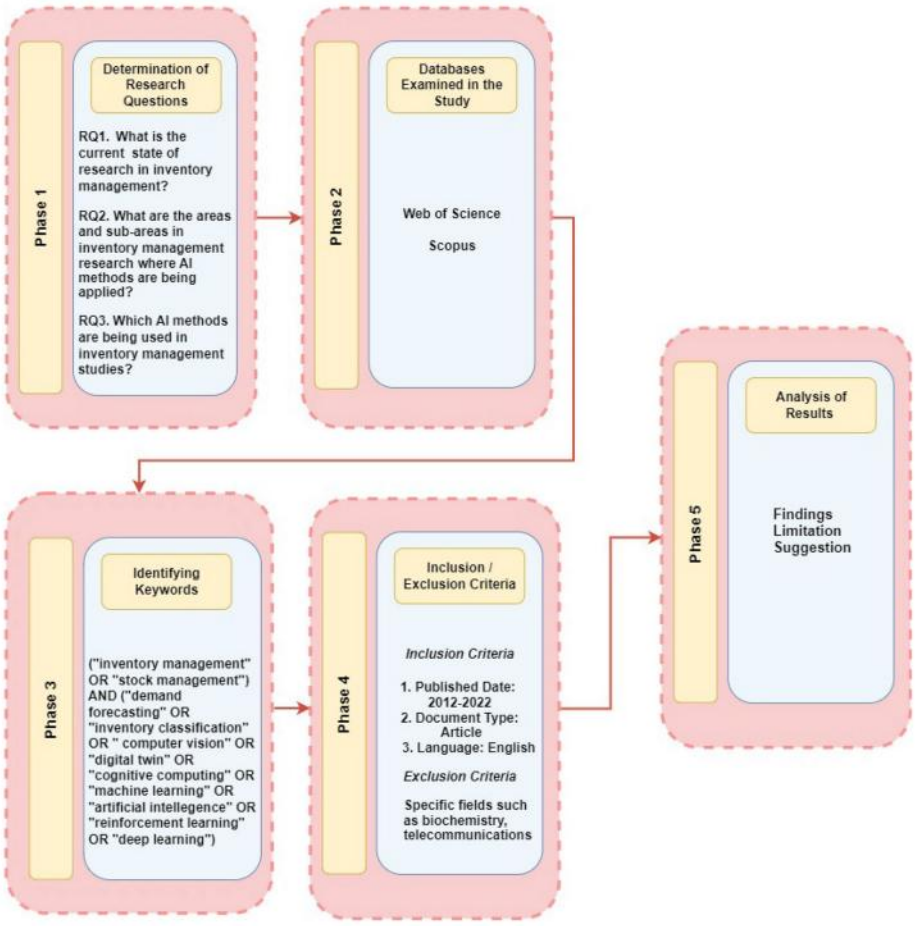

Figure 3: Stages of systematic literature review

Albayrak Ünal, Ö, et al. (2023) Has concluded that there is indeed a growing trend in terms of interest in AI and that the most common types of AI used are machine learning and deep learning, this is due to them being able to provide better and more reliable results compared to traditional AI methods. There was also a substantial portion of the research that is dedicated to demand forecasting, where AI techniques are applied to improve the accuracy of predictions and optimize inventory levels. They also noted that there is still a need for further research as there are still gaps in the research, specifically in exploring the integration of various AI methods and their practical applications in real-world inventory management scenarios. Overall, the authors underscore the critical role of AI in transforming inventory management and meeting the challenges posed by evolving market demands.

2.2.3 Real Time Inventory Management System powered by Generative User Interface

A system that Patil, O., et al. (2024) has explored is the development of an innovative inventory management system (IMS) that leverages generative AI technologies to enhance user interaction and system functionality. Their primary goal is to address the limitations of the traditional inventory management systems, often lacking personalization and flexibility. The system they have proposed aims to provide a more dynamic and user-friendly experience using a generative user interface.

The system architecture they chose is a GPT-4 model, specifically the gpt-4-turbo-2024-04-09 variant, as the core generative AI technology for the proposed Inventory Management System (IMS).

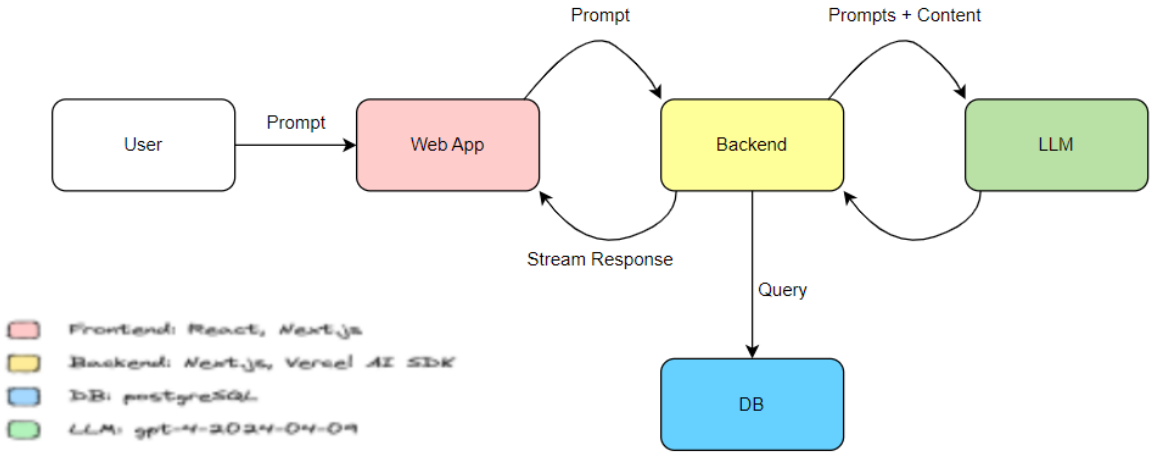

Figure 4: Data Flow Diagram for proposed IMS

The main reason that this model has been chosen is because of the compromise that has been made with generative man-made reasoning models. Meaning that GPT-4 can understand and generate human-like text, enabling natural language processing (NLP) capabilities, allowing users to interact with the IMS in a conversational manner.

Some of the features that Patil, O., et al. (2024) has included in the IMS are the natural language interaction which allows the users to chat to the AI instead of having to navigate menus to perform the task they wish to do. This feature allows for a more intuitive user experience, as users can ask questions or give commands in a way that feels natural. There are also plans to include voice commands, allowing users to interact with the system while remaining hands-free, which can be particularly useful in busy environments.

Patil, O., et al. (2024) also implemented some personalization system, meaning the system will adapt to user preference, learning from past actions to provide tailor-made recommendations and responses. There is also adaptive learning where the system can gradually improve conversational interface from repeated use by the users.

The IMS also offers a Real-Time Data Synchronization from the integration of a PostgreSQL database. This ensures that inventory data is consistently updated in real-time. Users can access the latest inventory levels, stock statuses, and other critical information, which is essential for effective decision-making. This is also followed by instant reports where users can generate reports on demand to get a clear analysis without waiting for batch processing.

The system's modular architecture, built on modern web frameworks like React and Next.js, ensures easy scalability and the addition of new features as business needs evolve, allowing it to grow alongside the organization. Its integration capabilities enable seamless connections with existing enterprise resource planning (ERP) systems and other business

applications, ensuring smooth data exchange and enhancing overall functionality, making it a flexible and scalable solution for dynamic business environments.

In short, Patil, O., et al. (2024) has successfully developed a generative UI powered IMS by integrating advanced generative AI models, conversational interfaces, and modern web frameworks like React and Next.js. This system aims to overcome the limitations of traditional inventory management systems by enabling natural language interaction and dynamic user interface rendering based on AI model responses. However, the author has also emphasized the need for ongoing research and development to explore further enhancements, such as multi-modal interactions, advanced analytics, and integration with IoT devices. These advancements could lead to even more sophisticated inventory management solutions.

2.3 Summary of Literature Review

The literature review explores three key studies on integrating AI into supply chain and inventory management systems. Shekhar A., et al. (2023) examined the role of generative AI in enhancing supply chain management. Their findings highlighted generative AI's ability to improve operational efficiency, forecast demand with greater accuracy, and enhance adaptability to market fluctuations. The study also noted challenges in data integration and quality assurance across different systems.

As for Albayrak Ünal, Ö, et al. (2023), they conducted a systematic review of AI applications in inventory management, focusing on machine learning and deep learning. They found these AI methods significantly enhance demand forecasting, inventory control, and decision-making processes. The study underscored the need for further research to integrate various AI methods and their real-world applications.

Patil, O., et al. (2024) developed a generative AI-powered inventory management system with a user-friendly interface and real-time data synchronization. This system, based on GPT-4, aims to overcome traditional inventory management limitations by allowing natural language interaction and dynamic user interface rendering. The study highlighted the system's benefits and the potential for ongoing enhancements like multi-modal interactions and integration with IoT devices.

Several more research has also been done but will be summarized here instead as the last 3 research is the most relevant to my research. One compelling real-world application of AI in inventory management was described by Simpson, M. D., & Qasim, H. S. (2025), in their narrative review of pharmacy operations across NHS hospital networks. They reported that the deployment of AI and machine learning models reduced stock-outs by 55%, decreased holding

costs by 40%, and increased stock turnover by 70%. This demonstrates the tangible operational improvements that AI systems can bring to real-time inventory control environments.

Another significant case study is found in the work of Subramanian, S., et al. (2025), who utilized reinforcement learning, specifically deep Q-networks in a Coca-Cola inventory setting. Their study showed how AI could dynamically calculate the most efficient restocking schedule, considering variable demand and limited shelf space. This resulted in reduced overstocking and improved sales alignment, further reinforcing the applicability of AI in real-world retail scenarios.

Broader technological trends also show increasing integration between AI and business process automation. Patrício, L., et al. (2024) reviewed how AI and Robotic Process Automation (RPA) are merging to create highly adaptive business workflows. Their findings suggest that the combination of these technologies allows systems to not only execute predefined rules but to learn and optimize those rules over time which is applicable in dynamic inventory systems that must respond to fluctuating supply chain conditions.

Similarly, Machucho, R., & Ortiz, D. (2025) provide a macro-level view of AI's impact on business innovation. Their review highlights AI's growing presence in streamlining operations, improving customer satisfaction, and enhancing predictive analytics. Companies such as Amazon and Alibaba were noted as pioneers in using AI-driven algorithms to predict inventory needs, which supports the overall viability of AI integration into web-based inventory systems like the one developed for this project.

In the context of supply chain resilience, Al-Hourani, S., & Weraikat, D. (2025) explored the role of AI and machine learning in pharmaceutical logistics. Their systematic review identified inventory optimization as a key area where AI can mitigate shortages and minimize wastage, especially in high-risk industries such as healthcare. Their findings align

with this project's goals of improving inventory visibility and reducing human error in item tracking.

Collectively, these studies demonstrate the transformative impact of AI in optimizing supply chain and inventory management. They emphasize the potential of AI to streamline processes, improve decision-making, and adapt to evolving market demands, while also addressing challenges in data integration and system scalability. Future research should continue to explore the integration of advanced AI technologies and their application across various business functions.

Chapter 3: Methodology

3.1 Introduction

In research and project development, the methodology acts as a guide and outlines the systematic approach and procedures used to conduct the study and build the system. It acts like a blueprint, detailing the techniques, tools, and processes that will be used to achieve the objectives. A good methodology ensures the project has a good structure and executed in an efficient manner, enhancing reliability and validity of the outcome. Agile Methodology will be used in this project, as the methodology provides the needed structure to ensure efficient and timely development of the project.

The choice of Agile is supported by Alzeyani, E. M. M., & Szabó, C. (2023), who conducted a comparative study on the effectiveness of Agile practices using real project datasets. Their research found that Agile methodologies—especially those based on iterative sprints—helped teams deliver software on time, minimized backlog issues, and enhanced user satisfaction. These findings validate the decision to apply Agile in developing a system where quick adaptation to feature changes and user testing is essential.

3.2 Agile Methodology

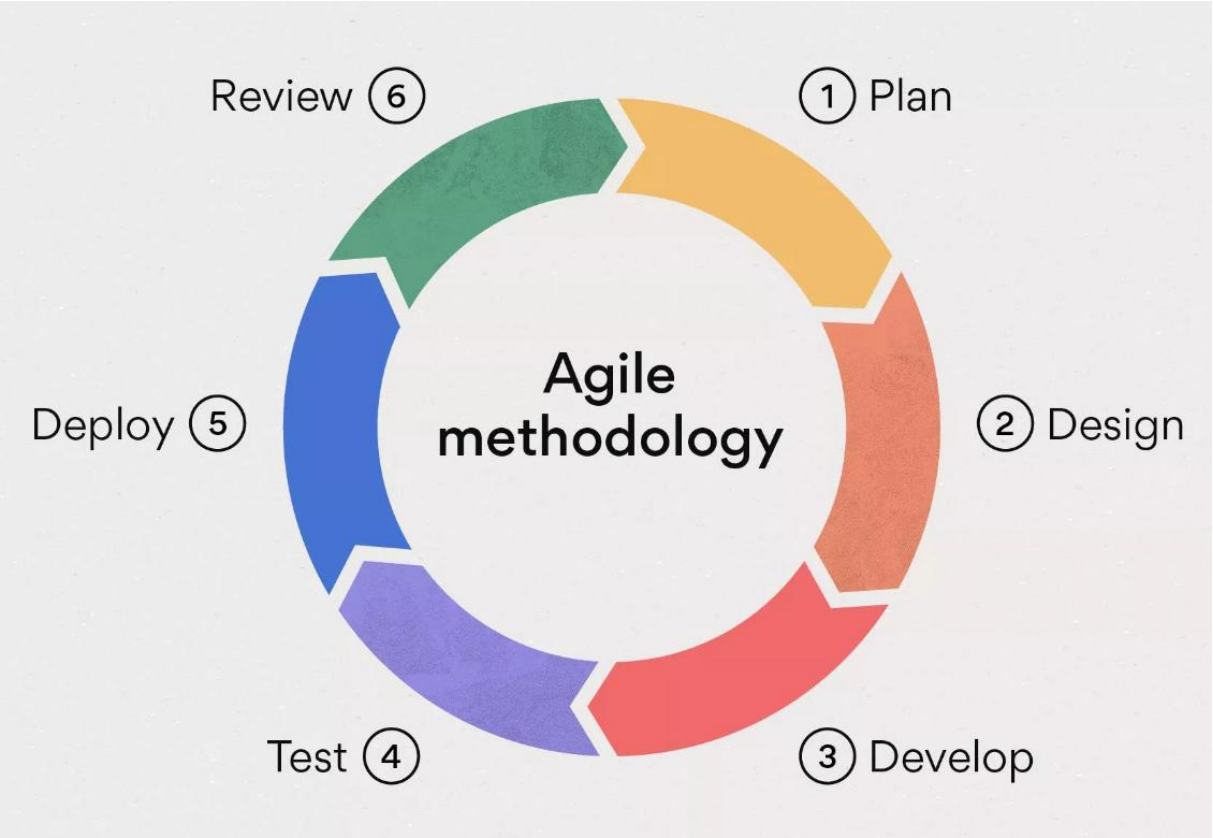

Figure 5: An overview of Agile Methodology

Agile methodology possesses a flexible and iterative approach to software development and project management. It focuses on delivering small improvements over a period of time, making sure a lot of reassessments and adaptations possible for changing requirements. This methodology is crucial for this project because of the frequent testing and feedback being crucial to refining the functionality of the chatbot and improving overall user experience. The Agile Methodology can be divided into 6 main stages listed below:

1. Requirement Analysis
2. System Design
3. Development
4. Testing and Validation
5. Deployment and Training
6. Evaluation and Iteration

Each phase serves their own purpose and is crucial to the development and completion of the project. The phases will be further elaborated in the next part of this chapter.

3.3 Requirement Analysis

The requirement analysis mostly focuses on the gathering of comprehensive insight into the inventory management challenges of small to medium-sized businesses. It involves identifying the pain points, understanding user needs and defining both functional and non-functional requirements. This phase also includes creating the user stories that reflect real-world scenarios to guide the development process and ensure that the system addresses the actual needs of the users.

Since the targeted group was primarily small to medium enterprises, a survey was sent out targeting employees to inquire regarding their experience and expectations from any inventory management systems. The next part will cover the summary of the survey and what the responses were.

3.3.1 Questionnaires

A survey was sent out to contacts who are employed from different companies to ask for their feedback of the current system they have in place to help with stock management. The purpose of this survey is to gather information regarding what system they have currently and what they would want to see from an AI-powered inventory management system. This section will detail the information gathered from the survey response.

3.3.1.1 Section 1: Demographics

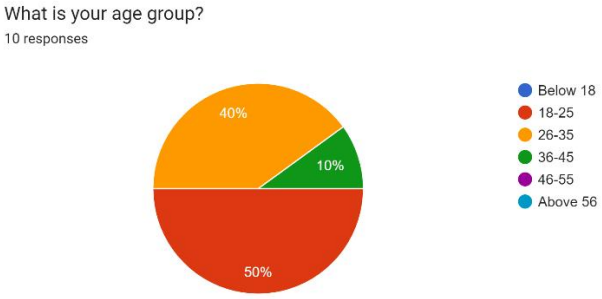

Figure 6: Age group of respondents

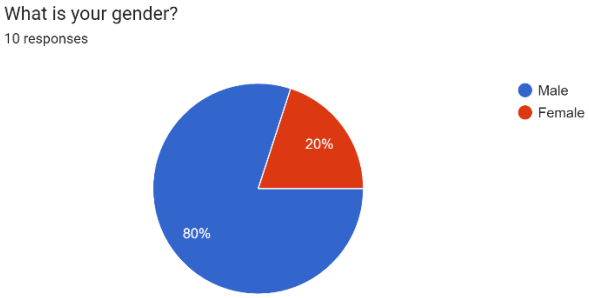

Figure 7: Gender of respondents

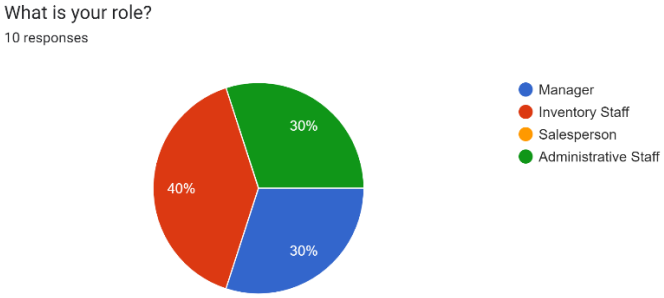

Figure 8: Role of respondents

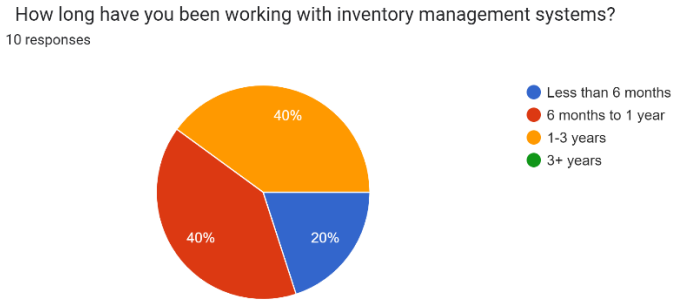

Figure 9: Experience of respondents to Inventory Management Systems

From the demographic groupings, we can see that the primary age grouping is 18-25 is the dominant group, at 50% followed closely by 26-35 at 40% and a very small percentage of

36-45 at 10%. The gender is mostly male at 80% while female is at 20%. As for the role it is very split with the majority being Inventory staff at 40%, followed by 30% of administrative staffs and another 30% being Managers. As for experience both 6 months to 1 year and 1-3 years of experience is at 40% while less than 6 months contributes to the other 20%.

3.3.1.2 Section 2: Pain Points in Current Inventory Management

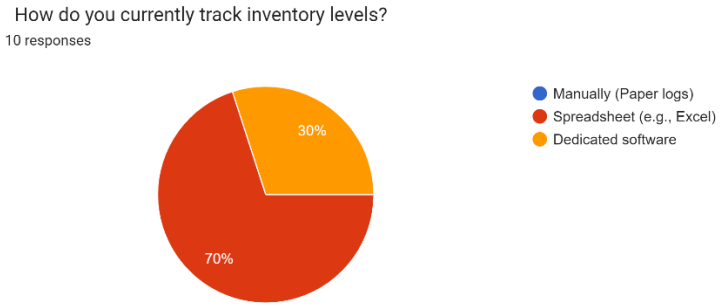

Figure 10: Inventory tracking method of respondents

From this diagram we can see that a majority of the respondents, sitting at 70% still uses spreadsheets as a means to track inventory levels while the other 30% already has dedicated softwares to do so.

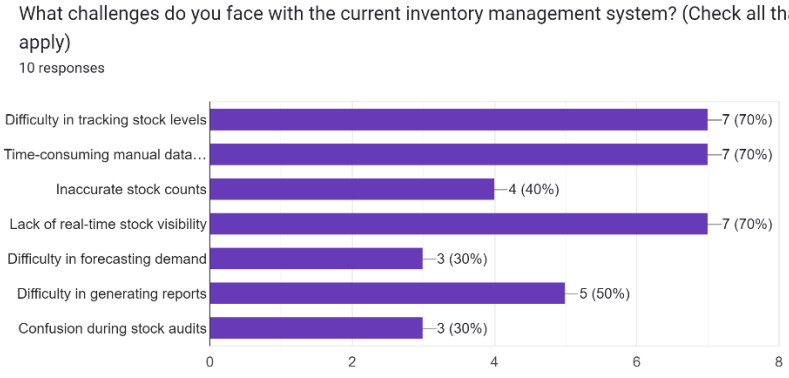

Figure 11: Challenges the respondent faced with current IMS

From the question asked, the three common challenges that a majority of respondents face is difficulty in tracking stock levels, Time consuming manual data entry and lack of real-time stock visibility, each having a high pick rate of 70%. Another noteworthy point to keep in

mind is difficulty in generating reports as 50% of users has this issue. The rest of the selection would encompass inaccurate stock counts at 40%, while both difficulty in forecasting demand and confusion during stock audits contribute by about 30% each.

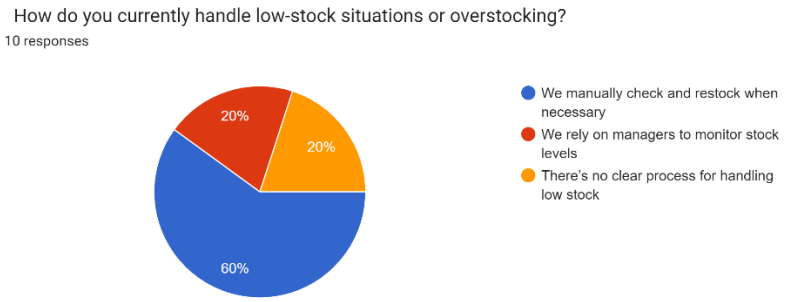

*Figure 12:*How respondents handle low-stocks or overstocking situations

60% of the respondents manually check and restock when necessary while 20% rely on their managers to monitor stock levels and the other 20% don't have a clear process for handling low stock.

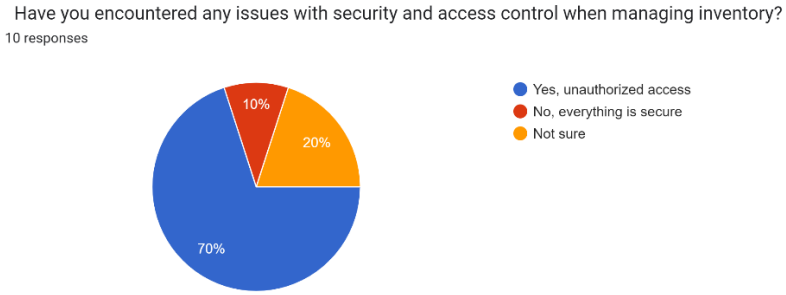

Figure 13: What issues have the respondent encountered with security and access control

About 70% of respondents has had issues in the past with unauthorized access due to the lack of access control from the system, another 20% is unsure of any history they have while 10% of respondents has never had this issue.

3.3.1.3 Section 3: User Needs and Expectations

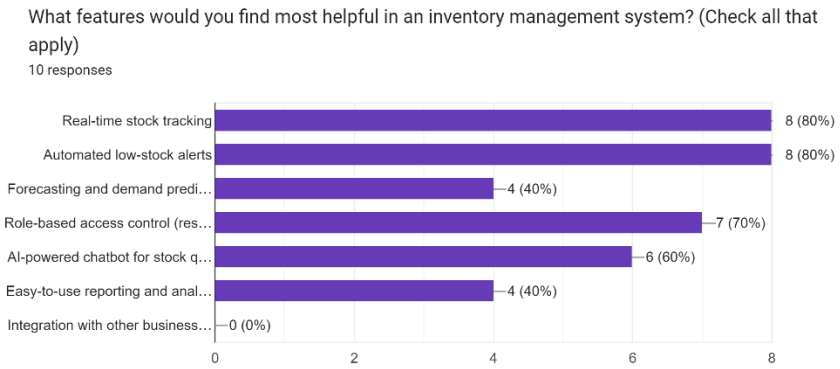

Figure 14: Features that respondents would find helpful

From this part, a majority of the respondents place an emphasis on real-time stock tracking and automated low-stock alerts, both being at 80% while the second highest is having a role-based access control to separate the level of controls users are capable of being at 70%. The third is having an AI-powered chatbot for stock queries at 60% while the rest, including forecasting and demand predictions and easy to use reporting and analytics are placed at a low emphasis of 40%. Surprisingly, no respondent is interested in the system having integration with other business systems and prefer having them separate.

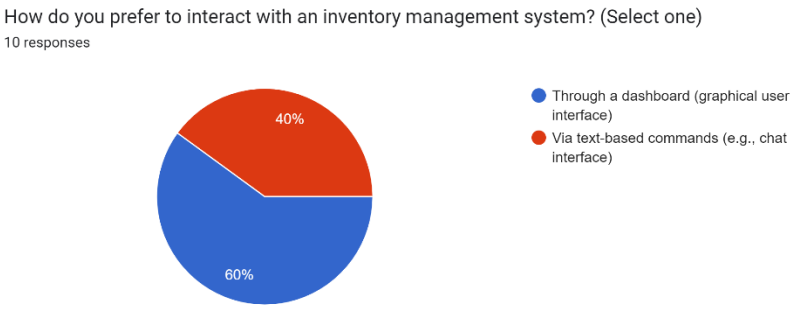

Figure 15: Interaction method the respondents prefer

Most of the respondents prefer having an actual dashboard being at 60% while another 40% would be willing to try a text-based commands interface. This may be due to familiarity or just the reliability of having a dashboard to review from.

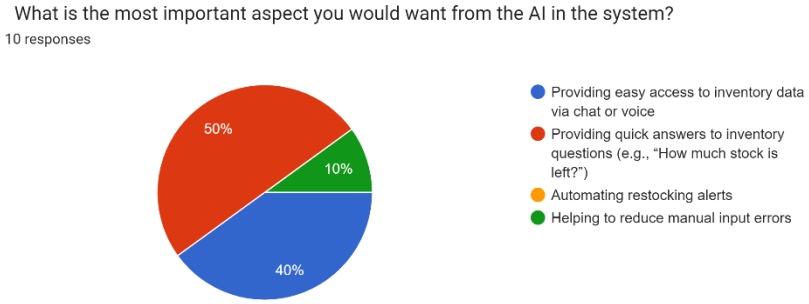

Figure 16: Aspects respondent wish to see from the AI

Around 50% of the respondents place an emphasis on having the AI providing quick answers to inventory questions, followed by 40% of respondents wanting it to provide easy access to inventory data via chat or voice and another 10% just wanting it to help with reducing manual input errors.

3.3.1.4 Section 4: Functional Requirements

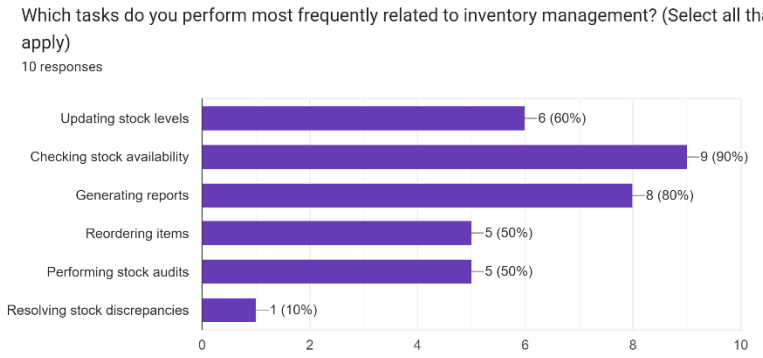

Figure 17: Tasks performed by respondents

Overall, many respondents uses their system to check stock availability at 90%, followed by for generating reports at 80%, 60% of users also use it to update stock levels and both reordering items and performing stock audits are sitting at 50%. 10% of users also use the system for resolving stock discrepancies.

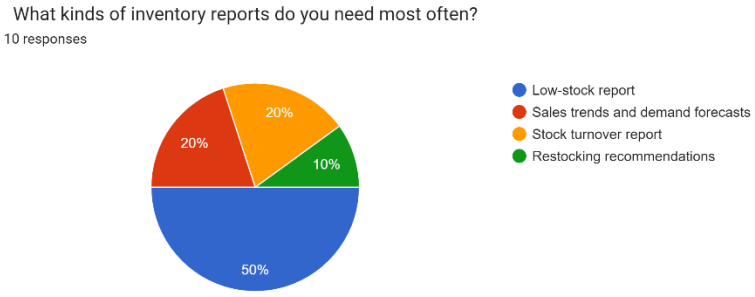

Figure 18: Inventory reports used by respondents

Respondents mostly opt to have the report be for low-stock items, sitting at 50% while the rest is mostly for sales trends and demand forecasts and stock turnover reports at 20% while another 10% is for restocking recommendations.

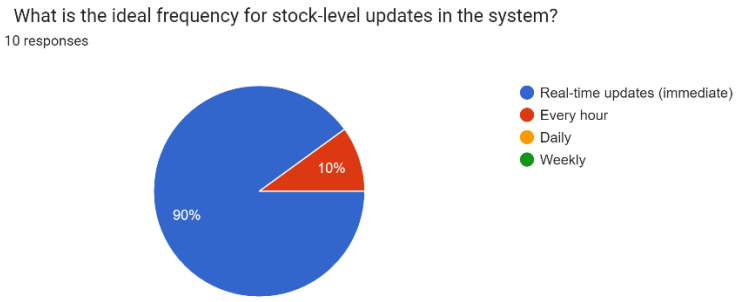

Figure 19: Respondent's Ideal stock update frequency

An overwhelming majority of the respondents would prefer to have real-time updates on the system they're using sitting at 90% while the other 10% thinks every hour is good enough, most of the users may be using systems where time is important and having real-time updates is a huge requirements.

3.3.1.5 Section 5: Non-Functional Requirements

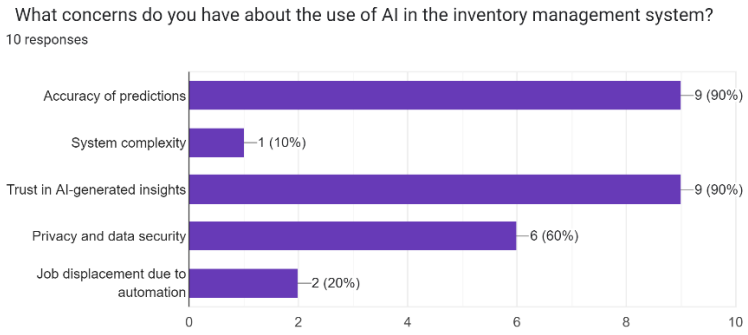

Figure 20: Respondent's concern about AI usage in IMS

Most of the respondents are worried regarding the accuracy of predictions and also have issues with trusting in AI-generated insights, both being at 90%, the second highest concerns is privacy and data security due to the use of AI. While the other 2 concerns are minor, but it is noteworthy as they are worried about job displacement due to automation and also the added AI features may end up bloating the system, making it unnecessarily complex and hard to diagnose were there to be any errors.

3.3.1.6 Section 6: Final Feedback

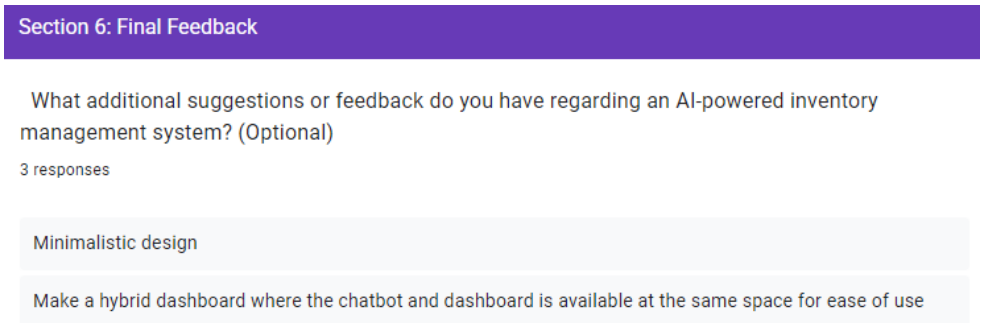

Figure 21: Respondent's additional suggestions and feedback

In this suggestion, 2 respondents have added their own input, placing an emphasis on having a more minimalistic design as to not bloat up the screen and also making a hybrid dashboard combining both the actual dashboard and chatbot. This suggestion is interesting since earlier in the survey, most would still prefer having an actual dashboard as opposed to a text based command, so implementing a hybrid dashboard satisfies both demographic's requirements where they can still have control over what data they are able to view but at the same time can ask the chatbot for clarifications.

3.4 System Design

In the system design phase, this places on the emphasis on developing a scalable and efficient system architecture that can integrate AI functionalities seamlessly. This phase includes diagrams and UI designs where relevant. The goal of this phase is to serve as a guide for the development phase to make sure the development stays on track to meet the project requirements and objectives.

3.4.1 Block Diagram

In this project, the proposed flowchart follows a similar vein to the system reviewed back in chapter 2. Where the user will interact with the chatbot via Dashboard, the chatbot will them send the prompt to the backend where it will be processed by the Large Language Model and then forward the query to the database before the answer is pushed back to the chatbot via the dashboard.

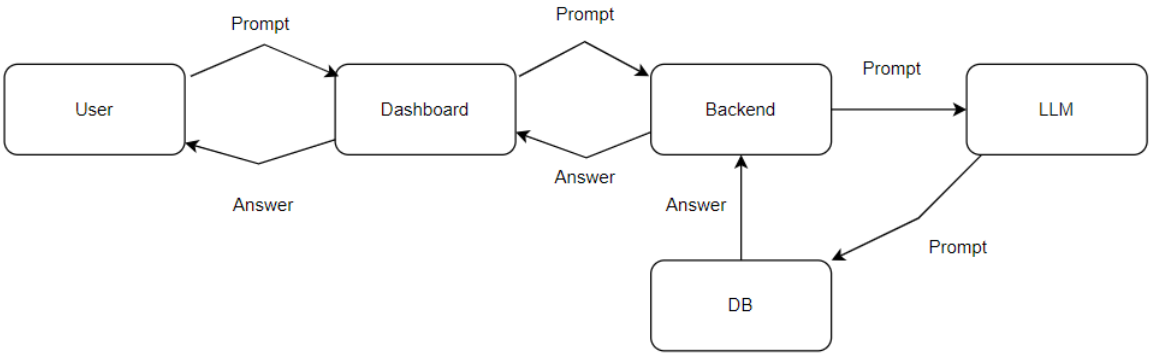

Figure 22: Block Diagram for the project

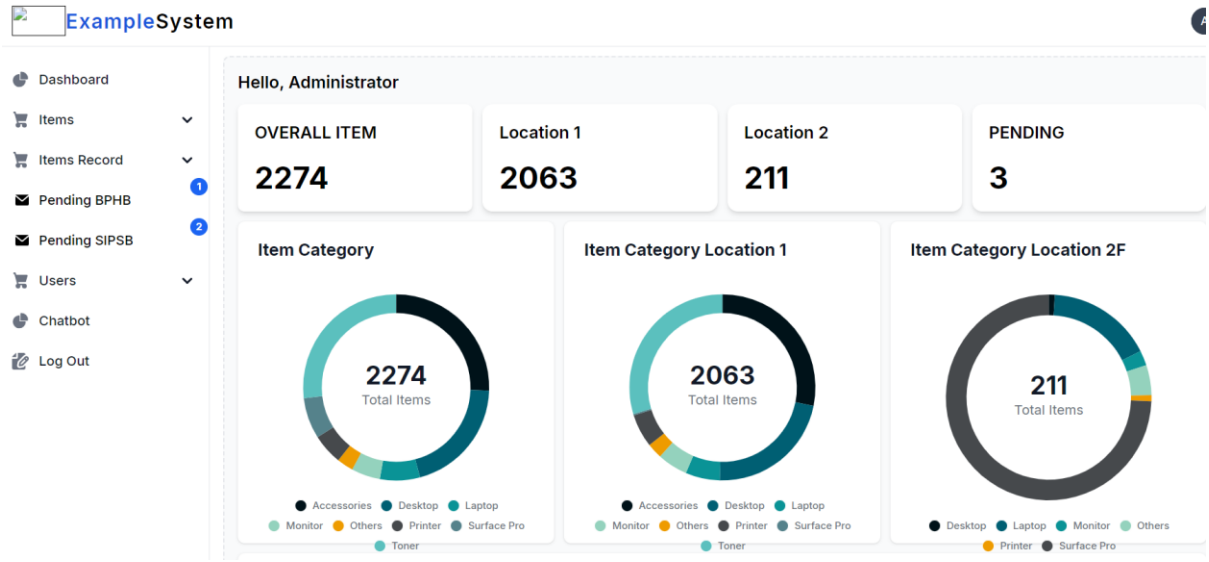

Figure 23: Proposed System UI

3.4.2 Use Case Diagram

In this project, the admin is responsible for a number of operations, mainly for the system design, system development and also the design and implementation of the AI chatbot. There will two primary users of the system, mainly the user and the manager, both with their own roles within the proposed system. The user would use the system, give feedback regarding the system and will be able to request for item withdrawals, whereas the managers are also able to use system and give feedback, their role difference being that they are the ones who approve the normal user's item withdrawals. Any feedback that either the user or manager submit will go through to the admin for the admin to perform updates and maintenances.

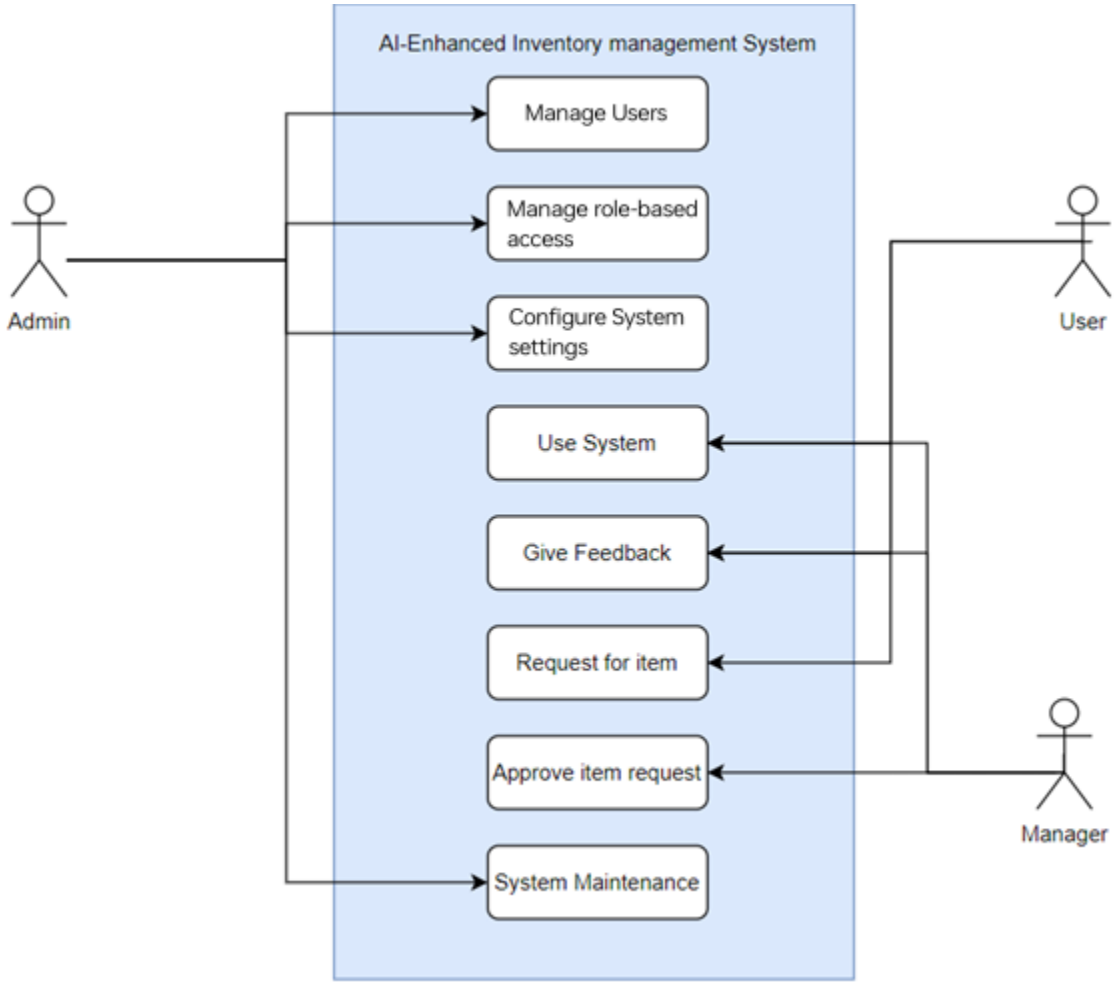

Figure 24: Use Case Diagram of system

3.4.3 Flowchart

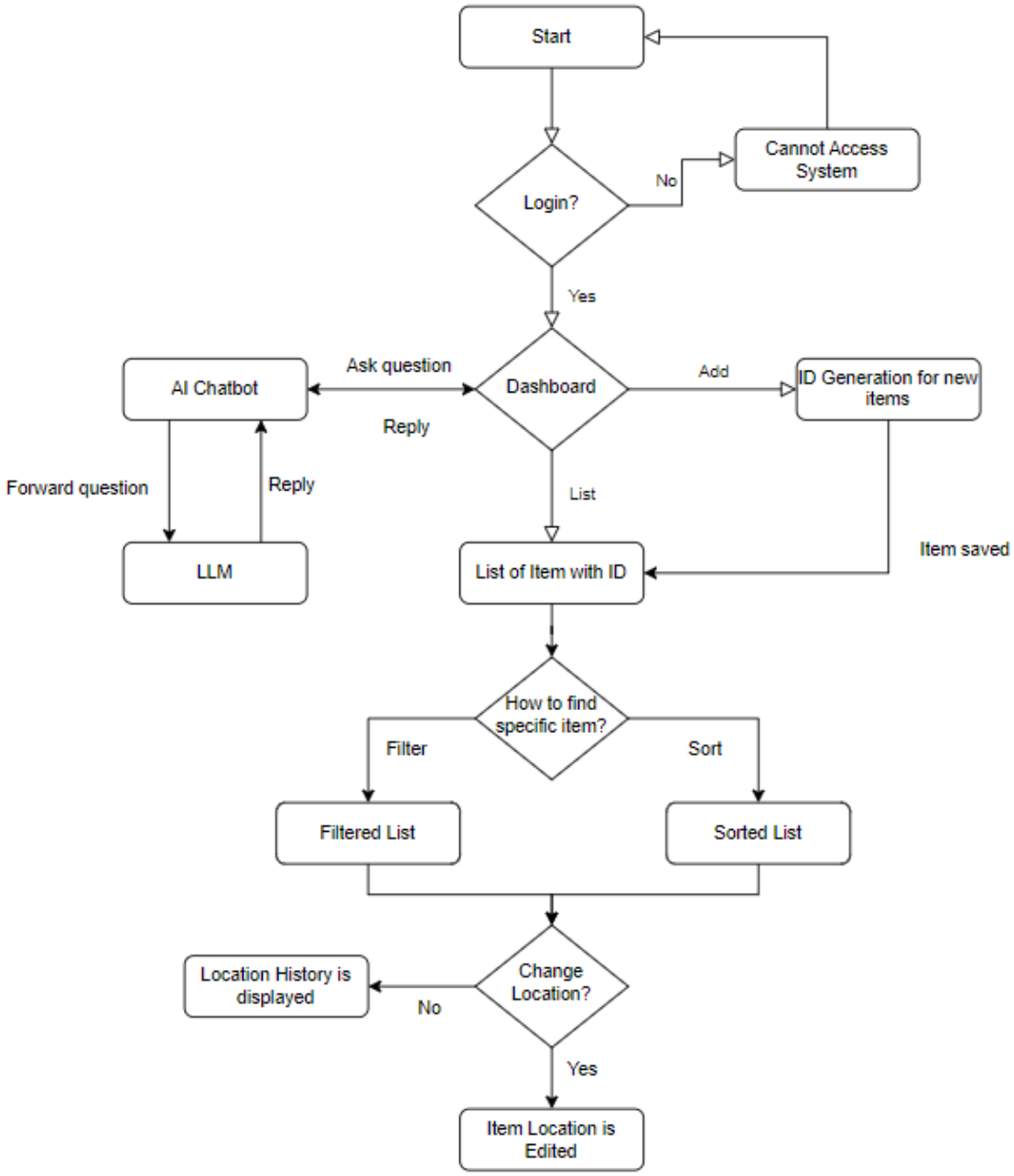

Figure 25: Flowchart of proposed system

The diagram above features the proposed flowchart for the system. It begins with the system prompting the user to login so that the user's role is defined. If the user doesn't log in there the user cannot access the system. Once the user access the system, they have multiple choices to make, adding a new item to the store, list down the item currently in the store or interacting with the AI chatbot, our emphasis would be on the AI chatbot so once the user asks a question to the chatbot, the chatbot would then forward the question to the Large Language Module which connects to the database, it will then retrieve the necessary information and then reply to the user.

3.5 Development Phase

The Development phase of the agile methodology is where the coding of the system takes place. With the design phase It starts with the front end of the system with the usage of development tools like Tailwind CSS. Tailwind CSS can helps streamline the design process by providing utility-first CSS classes, enabling developers to build custom designs efficiently. and implementing development frameworks like Django facilitating the rapid development of secure and maintainable applications by following the model-template-views (MTV) architectural pattern. It also encompasses the development of the back end using PHP and MySQL for to act like the Database, and the integration of AI components with Natural Language Processing (NLP) powered by Large Language Model (LLM) which essentially acts like the backbone for the making and implementation of the AI chatbot.

3.5.1 Requirements Analysis

There is a fundamental need in this project, mainly on all the software requirements it has. The software requirements are the programs and platforms that enable the system to operate efficiently and achieve its intended smart capabilities. Furthermore, the main goal of this project is to develop a system that automates inventory management, reducing error margin and enhancing accuracy and reliability in inventory management.

3.5.2 Software Requirements

There are several software applications that are used in this project. The software applications are:

- PHP
- MySQL
- JavaScript
- Tailwind CSS
- Django (optional for framework use)
- OpenAI API (for AI chatbot integration)

3.5.2.1 PHP

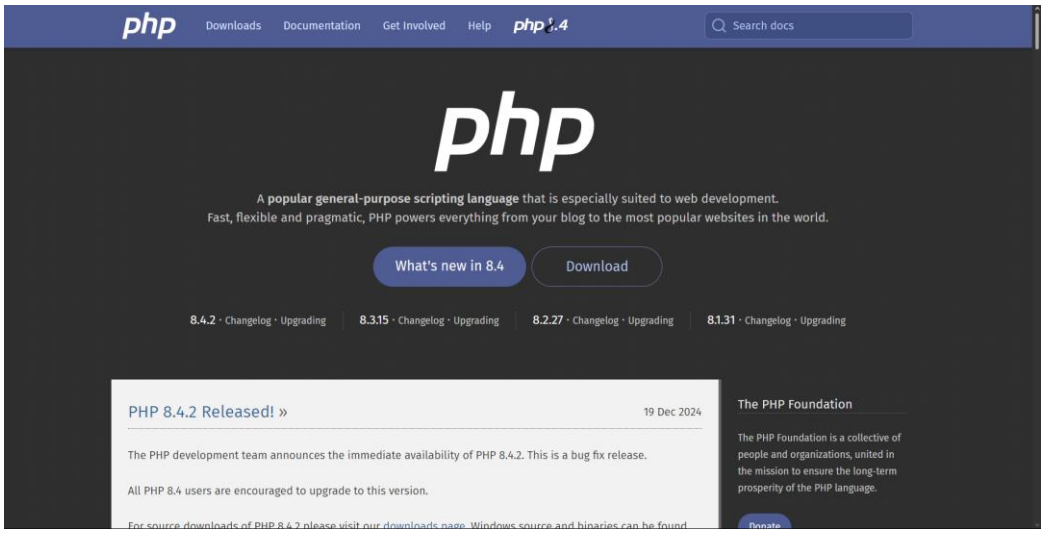

Figure 26: Main page of php website

PHP is a widely used open-source scripting language that is suited for web development. In this project, PHP will be used to handle server-side operations, managing contents, session management, and integrating with the MySQL database to perform CRUD operations on the inventory data.

3.5.2.2 MySQL

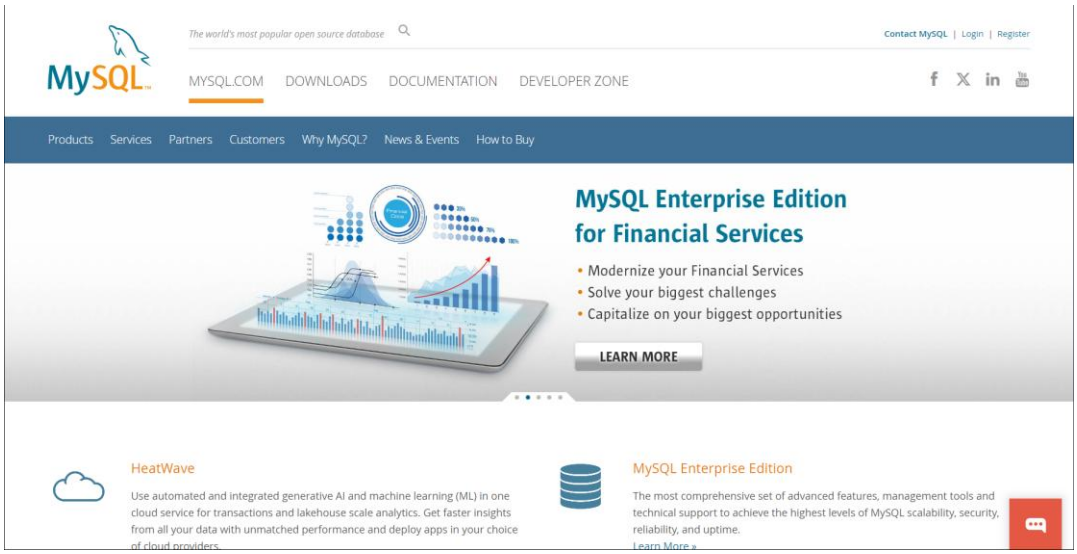

Figure 27: Main page of MySQL website

MySQL is an open-source relational database management system that will serve as the backbone of the data storage for this project. The main use of MySQL will be as the database for the system, including item details, stock levels, and user information, allowing for storing and outputting data efficiently.

3.5.2.3 JavaScript

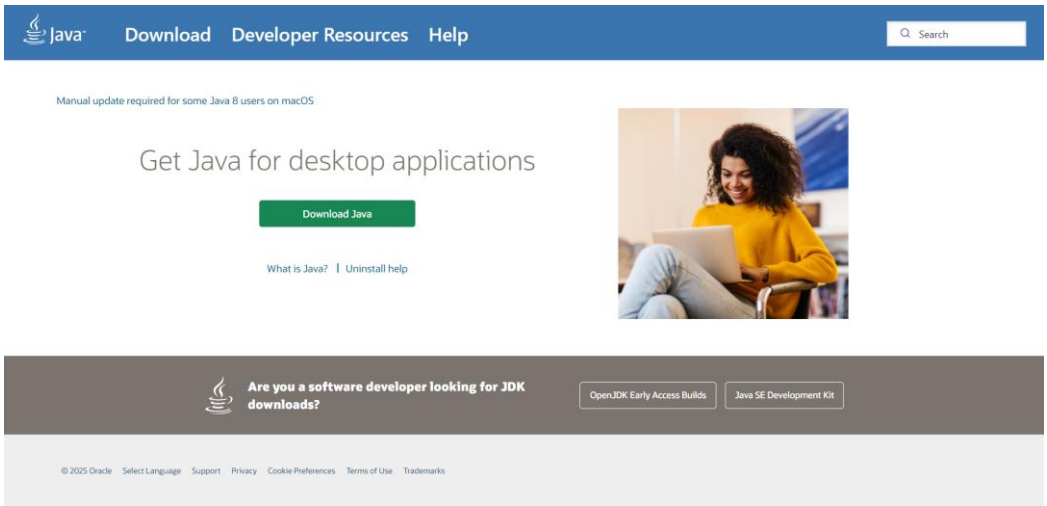

Figure 28: Main page of Java website

JavaScript will be used during the frontend development, creating interactive elements on the user interface. It will enhance the user experience by enabling real-time updates, form validations, and dynamic content manipulation.

3.5.2.4 Tailwind CSS

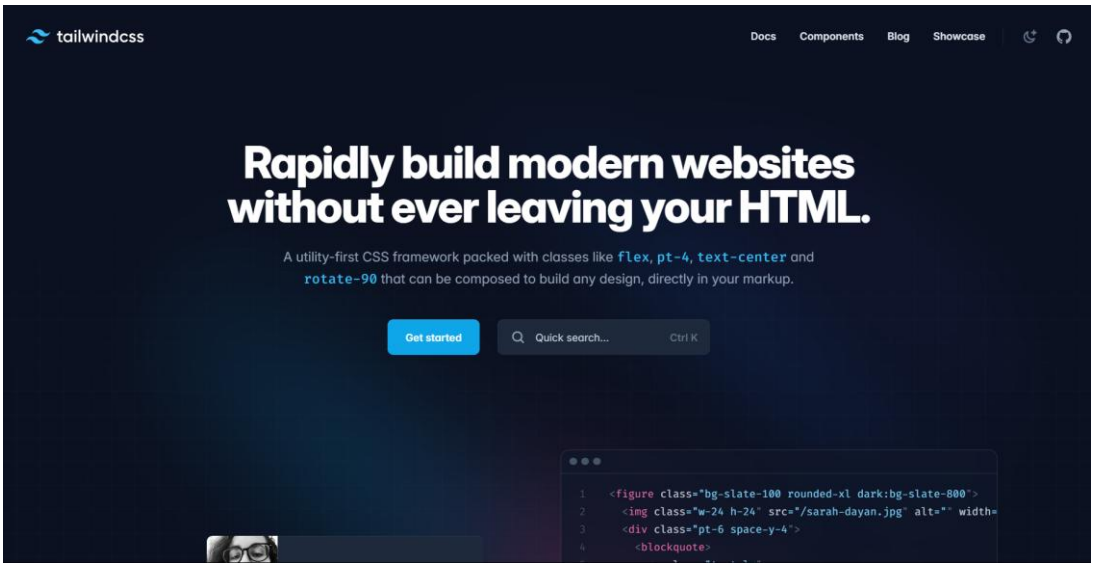

Figure 29: Main page of TailwindCSS website

Tailwind CSS is a type of CSS framework that is used to quickly design a responsive and aesthetically pleasing user interface for the system. It allows for quick customization of the visual elements without writing extensive custom CSS, ensuring that the frontend is both functional and visually appealing.

3.5.2.5 Django

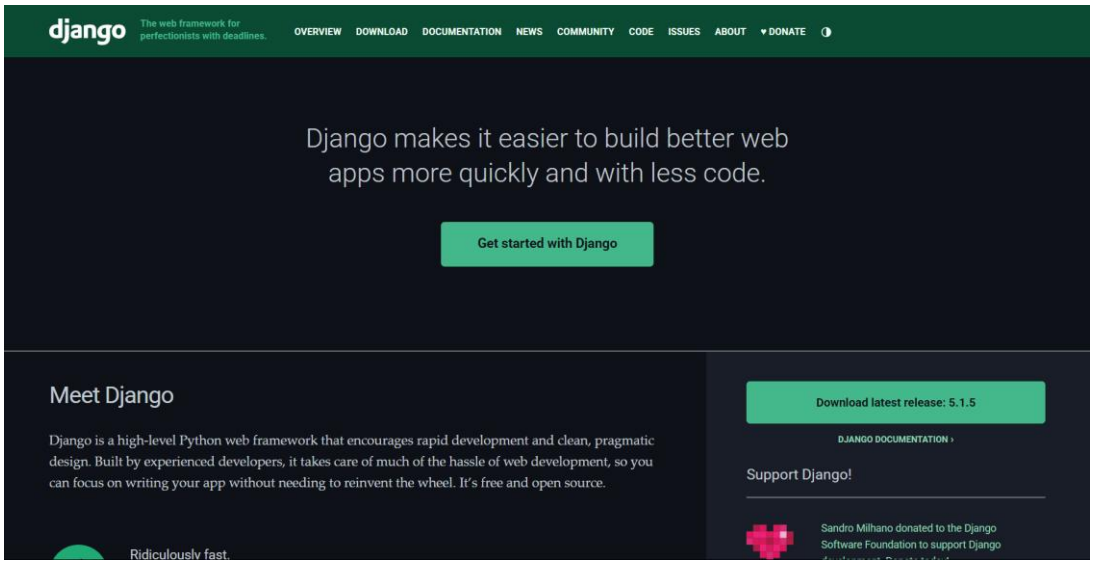

Figure 30: Main page of Django website

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is optional but however, it may be used in this project for its robust features, such as built-in security and scalability, to manage more complex backend operations if required.

3.5.2.6 OpenAI API

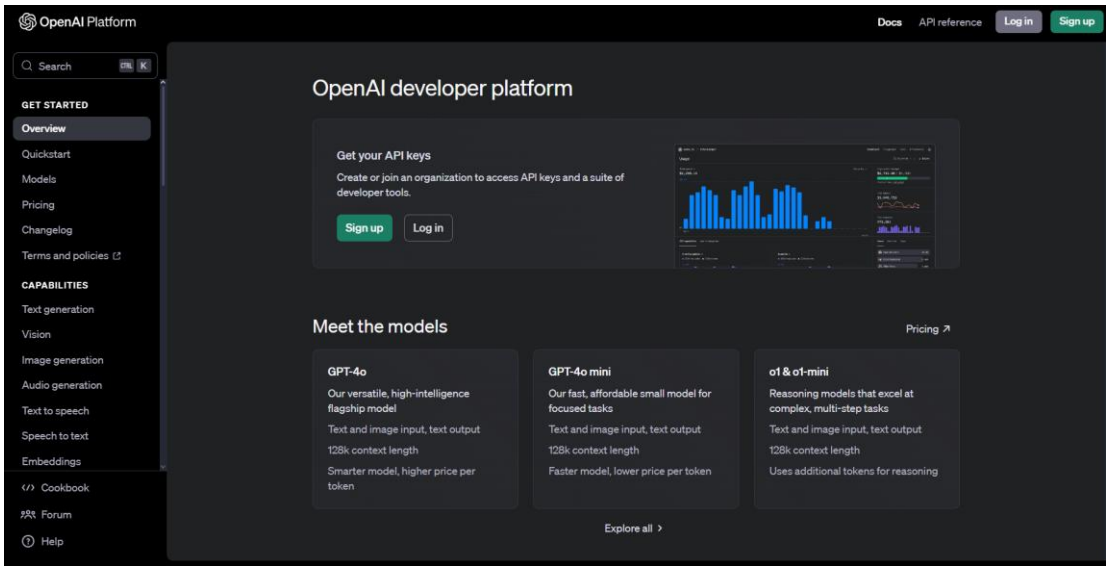

Figure 31: Main page of OpenAI API website

The OpenAI API will be used to integrate a large language model for the AI chatbot. This API will allow the system to process natural language queries from users, providing intelligent responses and assisting in inventory management through conversational interfaces.

3.5.3 Hardware Requirement

To develop this proposed project, several hardware components are important and required to complete this project smoothly. The hardware includes:

- Web Server
- User Devices (PCs, Tablets, Smartphones)
- Internet Connection

3.5.3.1 Web Server

Figure 32: Picture of Several Web Servers on server racks

A web server will act as the host for the AI-Enhanced Inventory Management System, processing requests from user devices and serving web pages. It is essential for ensuring the system is accessible to users across different locations and devices.

3.5.3.2 User Devices

Figure 33: A picture of a work PC

Aside from PC, user devices also means mobile devices such as tablets and smartphones is used to interact with the system. These devices will access the web application to perform tasks such as checking inventory levels, adding new items, and receiving alerts.

3.5.3.3 Internet Connection

Figure 34: A picture of a Modem to allow internet connection

A stable internet connection is required for real-time data synchronization between user devices and the server, as well as for the AI chatbot to interact with the OpenAI API and retrieve responses.

3.6 Testing Phase

The testing phase for the system involves evaluating both the functionality and performance of the system. This stage focuses on ensuring that all components are working correctly and with no errors, from the base functionalities of the system to the AI-driven chatbot responses. Testing will be done throughout the development process to quickly identify and resolve issues immediately.

3.6.1 Testing on Functional Requirements

Functional requirements are the specific actions the system must perform to meet its objectives. For this project, the functional requirements are:

- **Real-Time Stock Tracking**: Verify that the system accurately tracks inventory levels and updates the database in real-time.
- **AI Chatbot Interaction**: Test the chatbot's ability to understand and respond to user queries about stock information.
- **User Role Management**: Ensure that the system properly enforces role-based access control, restricting or allowing functionalities based on user roles.

3.6.2 Testing on Non-Functional Requirements

Non-functional requirements describe how the system performs certain functions rather than what it does. For this project, the non-functional requirements are:

- **Performance**: Test the system's response time and its ability to handle multiple simultaneous user requests without significant delays.
- **Reliability**: Evaluate the system's uptime and its ability to recover from failures, ensuring it is consistently available for users.

- **Usability**: Assess the ease with which users can navigate the system, focusing more on the design and user experience.

3.7 Deploy Phase

In the deploy phase, the features, functions, and user interface of the system is finalized. Stability, usability, and maintainability are tested and approved before the system is ready for real-world use. The deployment will be tested in a controlled environment, showcasing the system's functionality and effectiveness in managing inventory accurately and efficiently.

3.8 Conclusion

In conclusion, this project outlines the development and implementation of an AI-Enhanced Inventory Management System designed to automate and optimize inventory tracking for small to medium-sized businesses. The system integrates many technologies, such as a large language model-based chatbot which is aimed to improve efficiency and decision-making of the inventory management system itself. The successful deployment of this system will enhance inventory management processes as a whole, reducing manual labour and increasing accuracy.

Chapter 4: Implementation

4.1 Introduction

This chapter is made to present a detailed explanation of the implementation process of the AI-Enhanced Inventory Management System, namely AInventory. developed in an attempt solve the problem of inefficient and inaccurate stock management, especially in small to medium-sized businesses. The system leverages a combination of web technologies such as PHP, MySQL, JavaScript, and Tailwind CSS, while integrating advanced AI capabilities through the OpenAI API. The resulting solution is a fully functioning inventory management platform equipped with a natural language chatbot, real-time analytics, and secure user role management.

The purpose of this chapter is to outline the environments, configurations, and tools used in the development process. It also elaborates on all the core modules and functionalities of the system, covering both the front-end and back-end implementations in depth. Special focus is given to critical modules such as the chatbot interface, inventory operations, user access controls, and real-time feedback mechanisms. Each section explains how the technologies were integrated to work seamlessly in a locally hosted environment using XAMPP.

In addition to describing how the various system components were implemented and tested, this chapter also examines important security considerations, deployment strategies, and suggestions for further scalability. It aims to serve not only as technical documentation but also as a foundation for future improvements or deployments of similar systems.

4.2 Development Environment and Tools

4.2.1 Hardware Specifications

Development was conducted on a personal laptop to simulate a practical small-business environment. The device specifications are as follows:

- **Model:** Acer Nitro 5 (2022)
- **Operating System**: Windows 11 Pro (64-bit) 24H2 Release Preview
- **Processor:** Intel Core i5, 12th Generation
- **Graphics Card**: Nvidia GeForce RTX 3050 with 4GB VRAM
- **RAM**: 16 GB DDR4, dual-channel
- **Storage**: 512GB NVMe SSD + 1TB M.2 NVMe SSD
- **Network**: 100Mbps broadband with stable access to the OpenAI API

The selected hardware configuration provides a sufficient processing power and memory to run local server applications, database operations, browser-based frontend rendering, and remote API interactions concurrently without performance degradation.

4.2.2 Software Stack

Backend: PHP 8.1

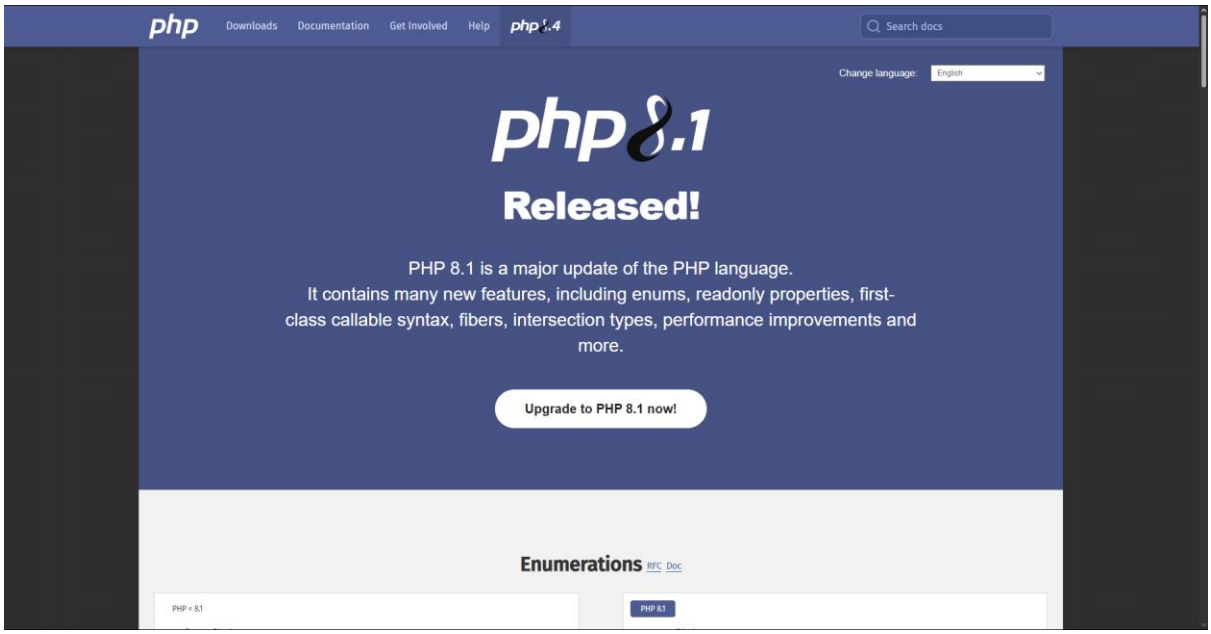

Figure 35: Main page of the PHP 8.1 web page

This version of PHP was chosen for its performance improvements, better error handling, and support for newer language features. It is responsible for handling all backend logic, including CRUD operations, session management, API interaction, and data validation. Its extensive ecosystem and compatibility with MySQL make it ideal for building dynamic web applications such as inventory systems.

Database: MySQL (via phpMyAdmin Version 5.2.1)

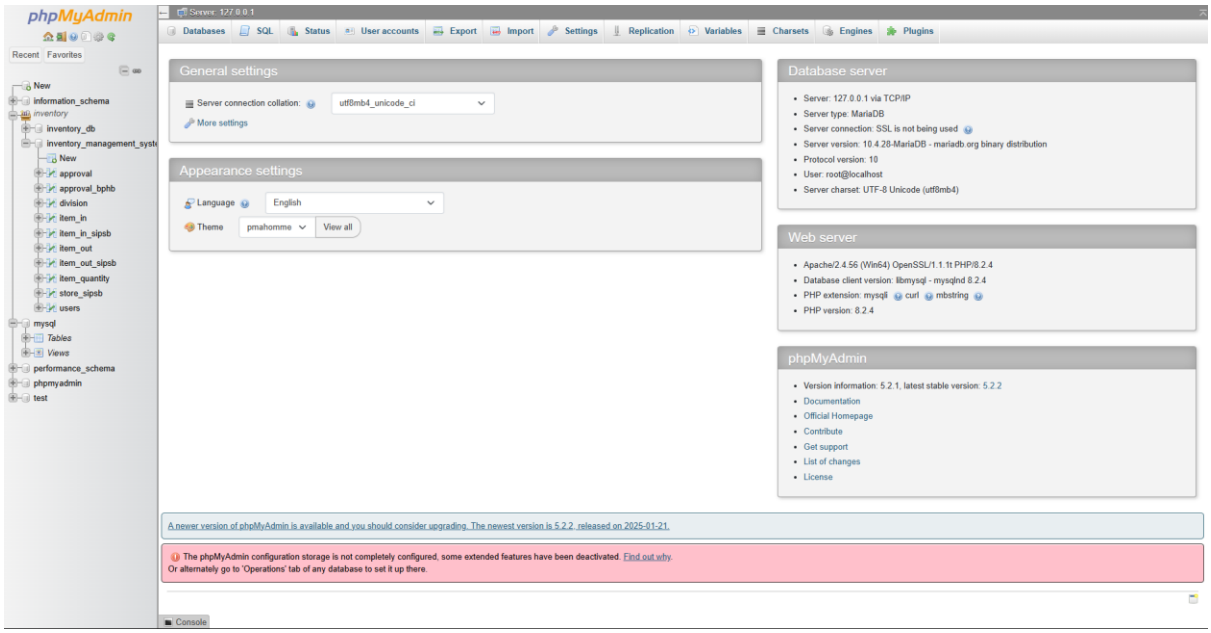

Figure 36: The index for phpMyAdmin version 5.2.1

As a robust relational database management system, MySQL handles all data storage for the system. Its compatibility with PHP, strong indexing features, and support for advanced SQL queries make it suitable for managing structured inventory data, user records, and approval logs. Queries are optimized for performance through indexing and normalization. Paired with this version of phpMyAdmin, it offers improved UI responsiveness and better security features. Providing a user-friendly web interface for administering the MySQL database, executing queries, backing up tables, and monitoring database activities. It was instrumental during the development phase for manually verifying data and debugging SQL issues.

Frontend: Tailwind CSS

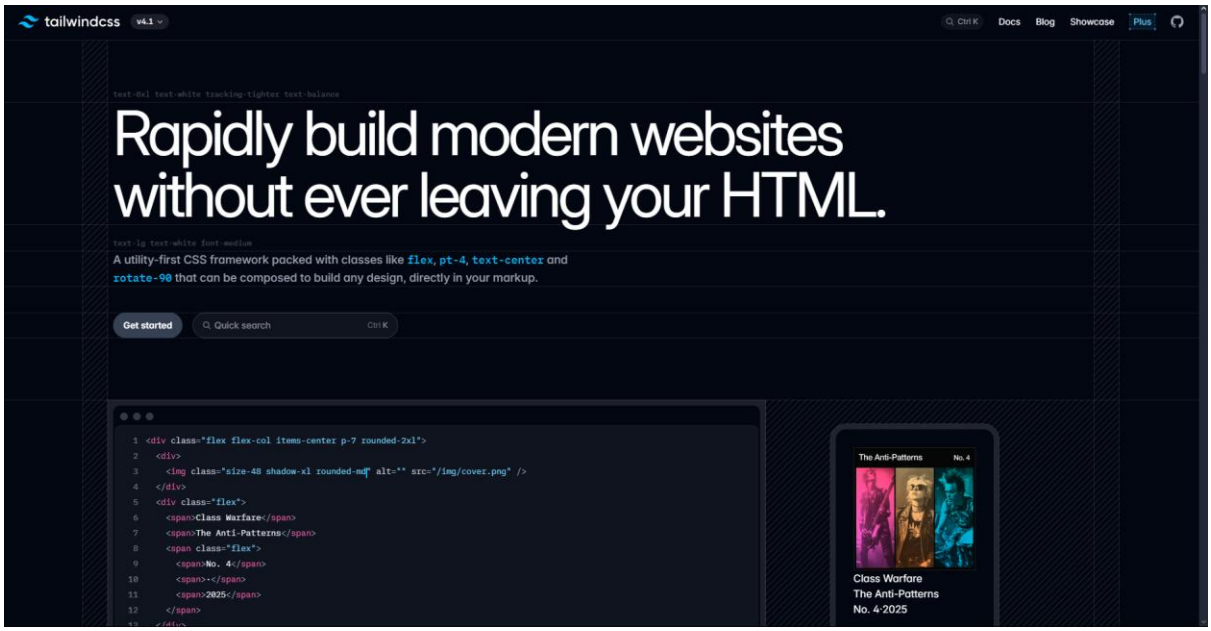

Figure 37: Front page of the Tailwind CSS website

Tailwind version 4.1 was used to speed up UI development with utility-first classes. The design language helps maintain a clean and consistent layout via CSS while allowing fast prototyping. Responsiveness, transitions, and accessibility features are managed through Tailwind utilities.

API: OpenAI GPT API (for chatbot)

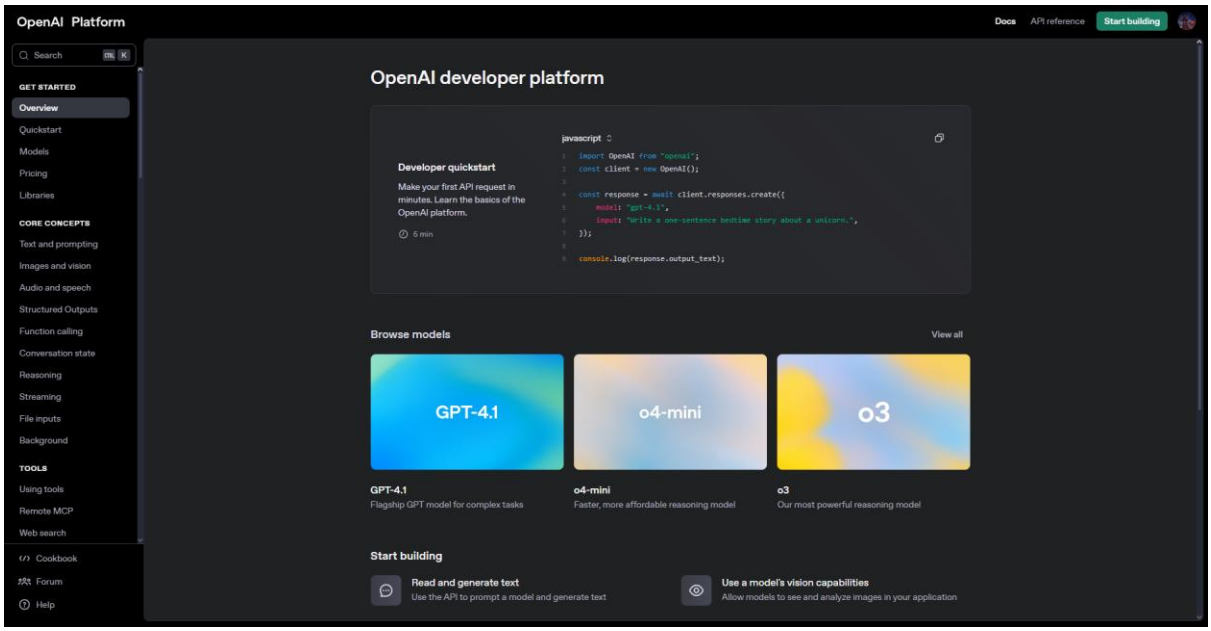

Figure 38: Front page of the OpenAI Platform

Integrated using server-side PHP cURL calls, the OpenAI GPT-3.5 turbo model that is used receives structured SQL result data and generates concise, natural language responses. This transforms static stock data into conversational insight. Error handling and API throttling were also implemented to comply with OpenAI's usage policies

Hosting: Local server via XAMPP v3.3.0

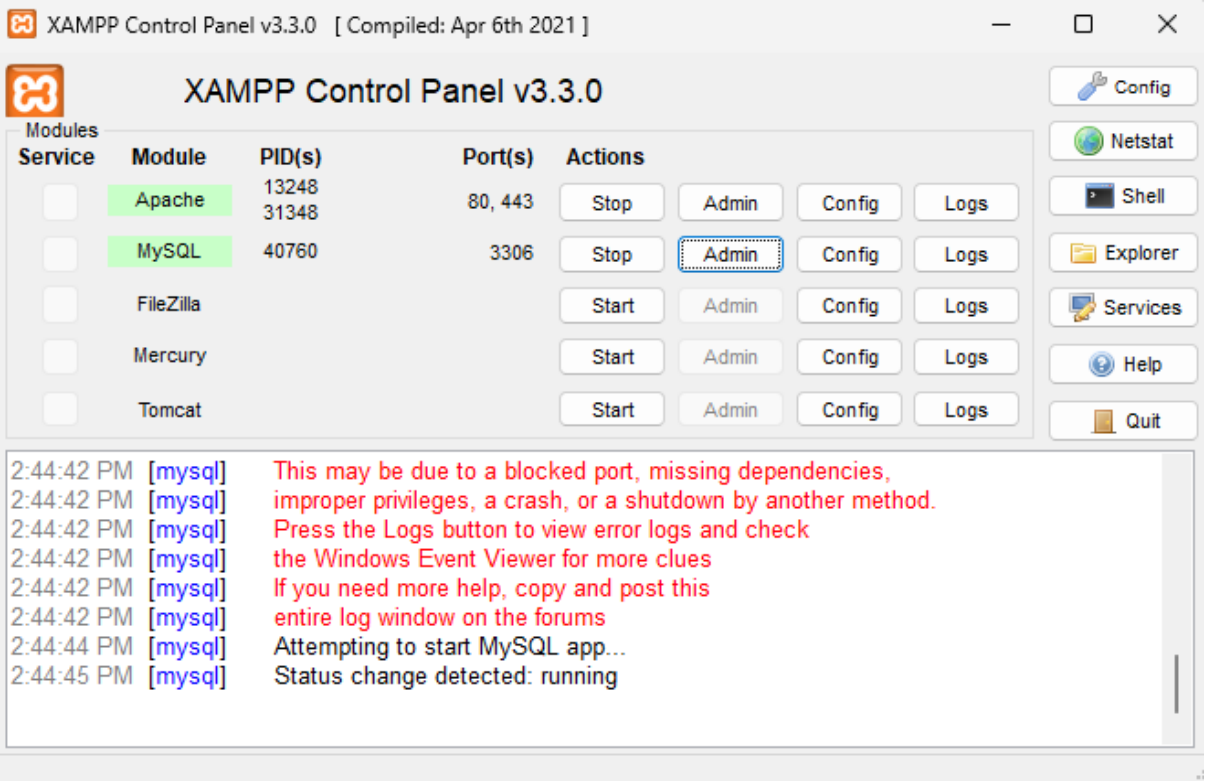

Figure 39: XAMPP Control Panel v 3.3.0

The XAMPP platform offers an all-in-one package that facilitates local testing and deployment. It replicates real hosting environments closely, supporting rapid prototyping, server configuration, and cross-browser testing while allowing network connectivity to functions that needs it.

Visual Studio Code

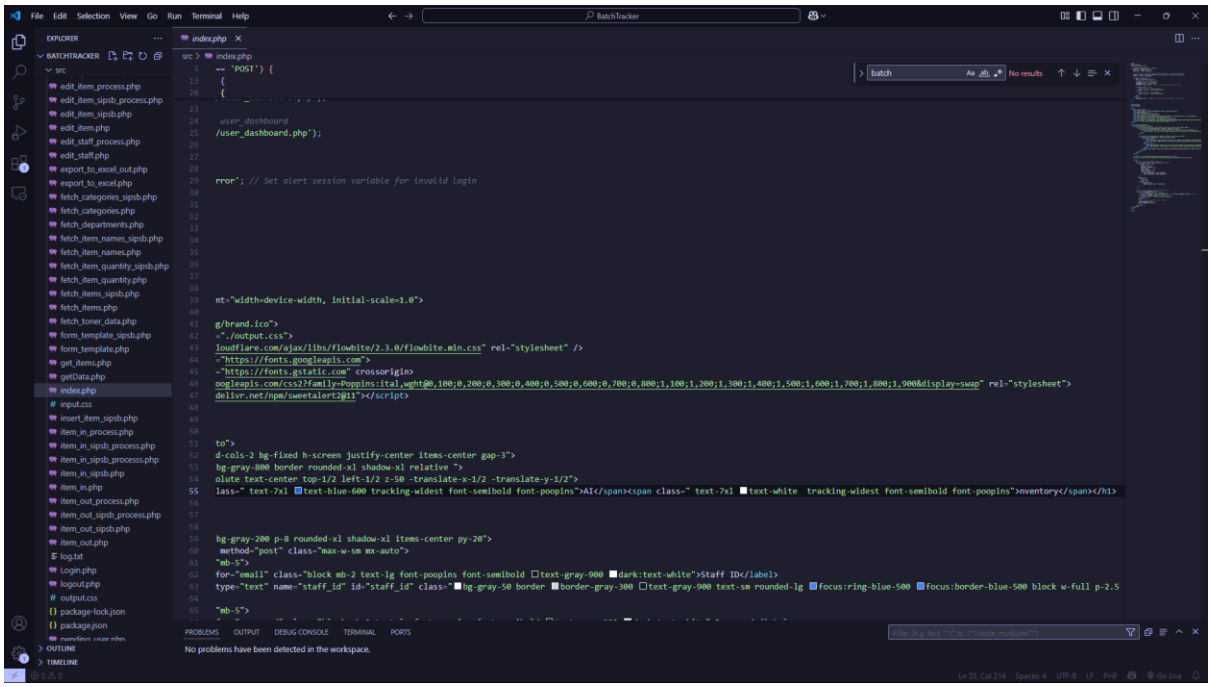

Figure 40: Visual Studio Code with code snippet

Visual Studio Code is a great platform for coding with its wide arrays of compatible extensions to help manage the tasks for its users.

4.3 System Architecture Implementation

The system architecture is designed to balance simplicity, scalability, and maintainability by separating core functionalities into logical layers. It adopts a modular, layered design model that allows developers to work on individual components independently while ensuring the seamless operation of the entire system. This architecture not only streamlines development and testing processes but also makes future upgrades or integrations significantly easier.

At the highest level, the system follows a client-server architecture. Clients access the platform through a standard web browser, interacting with a front-end that is responsible for rendering dynamic content and visual elements. This front-end communicates with a PHP-

based back-end, which acts as the central logic processor. Whenever a request is submitted from the front end, such as viewing inventory levels, adding a new item, or querying the chatbot, the server receives and processes this request before generating a response, either from the internal MySQL database or via interaction with the OpenAI API. The response is then formatted and returned to the client in real-time, typically through AJAX to ensure a seamless and responsive experience.

The architecture is composed of four main layers. The first layer is the presentation layer, which includes the user interface built using HTML, Tailwind CSS, and JavaScript. This layer is responsible for displaying data to users, receiving their inputs, and providing interactive controls such as forms, tables, and charts. Features such as graphs and charts are made possible through JavaScript, making the interface feel modern and intuitive.

The second layer is the application or logic layer, implemented entirely in PHP. This layer serves as the control centre of the system, orchestrating interactions between the UI, the database, and external services. It handles authentication, user role verification, form validation, and logic behind various features such as stock management and request approvals. It also processes chatbot messages by forming appropriate SQL queries and managing responses.

The third layer is the data layer, which relies on MySQL Database to store and manage persistent data. It includes structured tables for storing users, roles, items, stock entries, withdrawal logs, and approval statuses. Relationships between tables are defined using foreign keys to ensure referential integrity, while indexes are applied on frequently queried columns to improve performance. This layer ensures that all historical data is preserved, easily retrievable, and consistently updated.

Finally, the fourth layer introduces a novel AI integration layer. This layer acts as the bridge between the application logic and the OpenAI API. When a chatbot query is submitted, the system dynamically constructs a relevant SQL query, executes it, and formats the resulting data into a prompt suitable for the AI model. This prompt is sent through a secure HTTP request to the OpenAI server. Upon receiving a response, the PHP script interprets and displays the message in a human-readable format within the chatbot window. This integration allows the chatbot to act not only as a query interface but also as a real-time data interpreter.

The interaction between these layers is carefully coordinated. For example, when a user requests to see all items that are low in stock, the request is captured by JavaScript and passed via AJAX to a PHP endpoint. The backend validates the session, constructs a SQL query to identify items below a certain threshold, and retrieves the results from the MySQL database. These results are then either returned directly as JSON to be displayed in a table or formatted and passed to the OpenAI API for a conversational response. This dual-mode capability allows users to switch effortlessly between traditional interface usage and natural language communication.

Several important considerations were kept in mind during the design phase. First, the system adheres to the principle of separation of concerns. Each component is tasked with a distinct responsibility, reducing the chance of cross-functional interference and making the codebase easier to maintain. Second, extensibility is a key advantage of this architecture. New modules, such as reporting tools or external barcode readers, can be added without requiring a full system rewrite. Third, performance was prioritized by reducing server load through asynchronous communication and leveraging lightweight front-end design. AJAX ensures that user interactions are handled in real-time without reloading the page, and Tailwind CSS contributes to rapid UI rendering.

Moreover, the architecture is built with future scalability in mind. Although the current deployment is based on XAMPP for local testing, migrating the application to a cloud-based platform or integrating it with third-party business tools would require minimal adjustments. This flexibility opens the door to eventual commercial deployment or further academic research extensions. Security is also tightly integrated, with role-based access ensuring that users only interact with the components they are authorized to access, while the OpenAI API key is securely stored and never exposed in the client-side code.

To demonstrate how the system works in practice, code snippets from both the front-end and back-end implementations of the chatbot component are provided in the following section. These code samples illustrate the logical flow of data and control from the moment a user submits a query in the chatbot interface, to the processing and SQL query generation handled in the PHP backend, and finally to the point where a response is generated using the OpenAI API and returned to the user.

These code snippets serve as a substitute for a formal system architecture diagram and instead shows the exact path of execution with proof. The screenshots include the HTML responsible for capturing and sending user queries, the schemaDescription that shows the logic in place for the AI to comb through the database and finally the debug log to show what exactly is being called by the AI every time the user asks the question.

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>AI Chatbot</title>
<script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-white text-gray-900">
<div class="flex justify-center items-center min-h-screen">
    <div class="bg-white rounded-xl border border-gray-200 shadow w-full max-w-md flex flex-col overflow-hidden">

    <!-- Header -->
    <div class="bg-blue-600 border-b border-gray-200 text-white py-3 px-4 text-lg font-semibold flex items-center space-x-2">
        <svg xmlns="http://www.w3.org/2000/svg" class="w-6 h-6 text-white fill="currentColor" viewBox="0 0 24 24">
        <path d="M2 5a2 2 0 0 1 2-2h16a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H6l-4 4V5z"/>
        </svg>
        <span>AI Chatbot</span>
    </div>

    <!-- Chat area and input -->
    <div class="flex flex-col p-4 space-y-4 h-[500px]">
        <div id="chat-box" class="flex-1 overflow-y-auto bg-gray-50 rounded-lg p-4 text-sm text-gray-800 space-y-3">
        <!-- Messages will be appended here -->
        </div>

        <form id="chat-form" class="flex space-x-2">
        <input type="text" id="user-input" placeholder="Type a message..." required
            class="flex-grow rounded-full border border-gray-300 px-4 py-2 focus:outline-none focus:ring-2 focus:ring-blue-400">
        <button type="submit"
            class="bg-blue-500 text-white px-4 py-2 rounded-full hover:bg-blue-600 transition">Send</button>
        </form>
    </div>
    </div>
</div>
```

Figure 41: HTML responsible for capturing user query

```php
$schemaDescription = <<<EOD
You are a MySQL assistant. Generate a SELECT query using only these tables:

- users(staff_id, name, email, role)
- item_quantity(item_id, items_name, quantity)
- store_sipsb(item_id, item_name, item_quantity)
- item_in(id, item_name, item_quantity, item_out_date)
- item_in_sipsb(id, item_name, item_quantity, item_in_date)
- item_out(id, item_name, item_quantity, item_out_date)
- item_out_sipsb(id, item_name, item_quantity, item_out_date)
- approval(id, Item_name, item_out_date, location_delivered, request_by)
- approval_bphb(id, Item_name, item_out_date, location_delivered, request_by)

Instructions:
- for users, if role 0 then reply user, if 1 then reply staff
- Use only the tables and columns listed above.
- Use LOWER(column) LIKE LOWER('%value%') for product/item searches.
- Use CASE WHEN role = 1 THEN 'admin' WHEN role = 0 THEN 'user' END AS role if showing user roles.
- Apply LIMIT 10 results unless the user requests more.
- DO NOT invent or assume any columns or tables.
- DO NOT use Markdown formatting or wrap code in backticks.
- DO NOT explain the query. Just return the raw SQL string.
EOD;

$messages = [
    ["role" => "system", "content" => $schemaDescription],
    ["role" => "user", "content" => "Question: $message\nSQL:"]
];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, "https://api.openai.com/v1/chat/completions");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
    "Content-Type: application/json",
    "Authorization: Bearer $apiKey"
]);

$postData = json_encode([
    "model" => "gpt-3.5-turbo",
    "messages" => $messages,
    "temperature" => 0
]);
curl_setopt($ch, CURLOPT_POSTFIELDS, $postData);
$response = curl_exec($ch);
```

Figure 42: $schemaDescription to show AI Logic

Figure 43: debug log to show AI Logic

4.4 Application Setup and Deployment

Setting up the system for development and testing was a critical step to ensure a consistent and realistic environment for building, debugging, and showcasing the inventory management features. Since the goal was to simulate the deployment of the system in a small business environment, XAMPP was selected as the local hosting platform. XAMPP conveniently packages Apache, MySQL, and PHP into a single, easily installable environment that replicates the stack used in most real-world hosting providers.

The setup process began with the installation of XAMPP on a Windows 11 laptop. After confirming that both the Apache and MySQL services were running correctly, the project files were placed within the htdocs directory of the XAMPP installation. This ensured that the PHP scripts could be executed and tested via the built-in local server, accessible through a web browser at http://localhost/AInventory/src/index.php.

The MySQL database was initialized through phpMyAdmin, which is bundled with XAMPP. The system's database schema, including all necessary tables such as users, items, item_in, item_out, and approval, was imported using an .sql file via the phpMyAdmin interface. During this process, data integrity and foreign key constraints were checked to ensure smooth operations when the application performs multiple relational operations.

Environment configuration involved linking the system to the OpenAI API. A configuration file in PHP was created to securely store the API key, ensuring that it is never exposed to client-side scripts. This file was placed outside the web root directory and included in backend scripts when necessary. Proper error handling mechanisms were also introduced to gracefully handle scenarios where the API service is unreachable or when query formatting fails.

Once the backend and database were fully configured, attention turned to testing user roles and permissions. Test accounts were created with different role values in the users table to simulate the perspectives of both administrators and regular users. Admin accounts were granted access to approval panels and user management pages, while user accounts were restricted to adding stock and submitting withdrawal requests. Session management was verified to ensure that users without proper credentials were redirected to the login page, maintaining the integrity of role-based access control.

Static and dynamic assets such as Tailwind CSS files, JavaScript charts, and AJAX modules were also tested under the local server to confirm compatibility. Tailwind's utility classes ensured that all interface elements were responsive and aesthetically clean, while JavaScript and AJAX enabled live updates and smooth transitions without full page reloads. Functional validation included submitting test queries through the chatbot, adding dummy items, processing mock approvals, and generating downloadable reports.

The local deployment also allowed for manual testing of application performance, particularly how the system handles multiple user interactions, the delay in chatbot responses, and the consistency of real-time stock updates. Observations during testing highlighted that local deployment offers low latency, quick feedback loops, and the flexibility to modify backend logic and instantly verify changes.

In summary, the deployment process followed a structured approach, from environment setup, database initialization, API integration, user access validation, to full application testing. While currently hosted locally through XAMPP, the system architecture supports easy migration via a separate connection.php to allow for easy integration into a web server or live hosting service by transferring PHP files and exporting the database. This would require only minor adjustments such as domain configuration, SSL setup, and database connection string updates to fit remote hosting requirements.

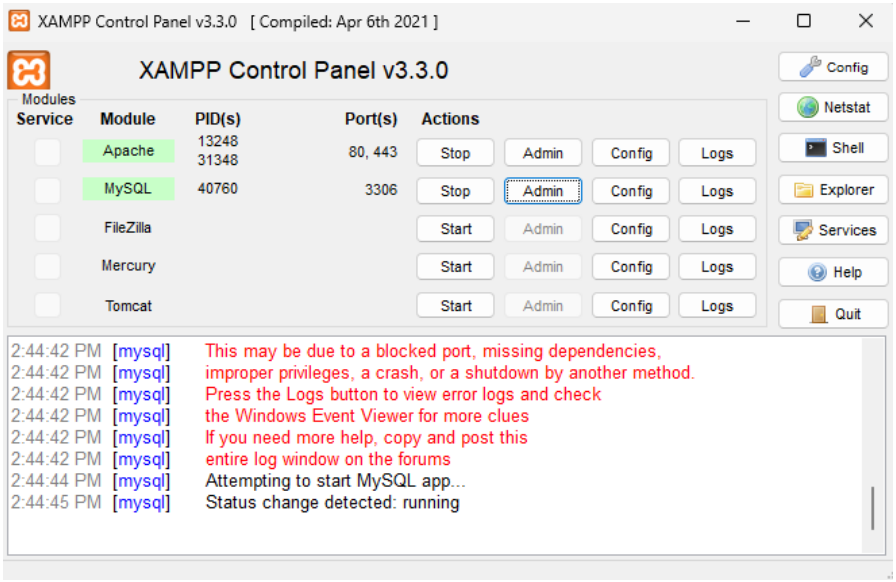

*Figure 44:*Screenshot of XAMPP Control Panel showing Apache and MySQL running

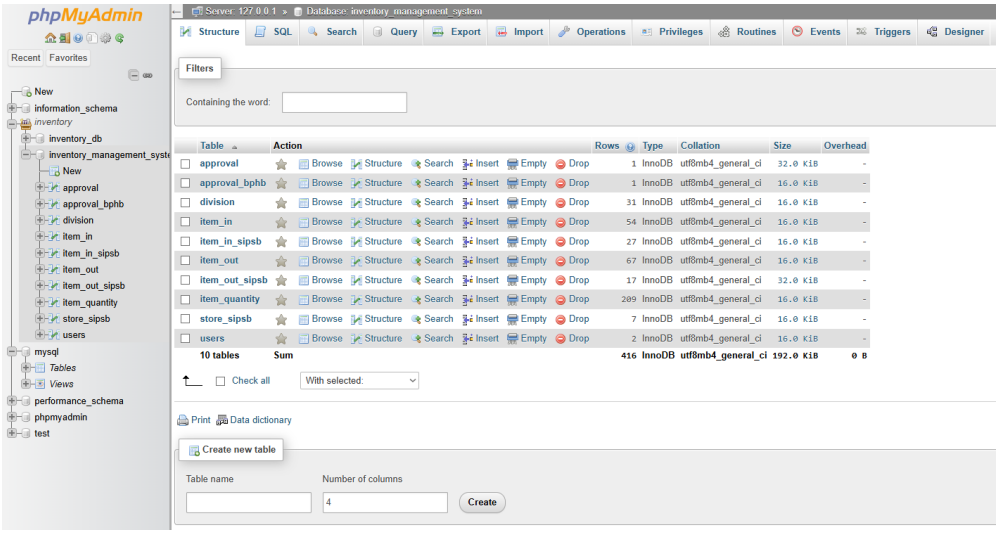

Figure 45: Screenshot of phpMyAdmin interface displaying imported tables

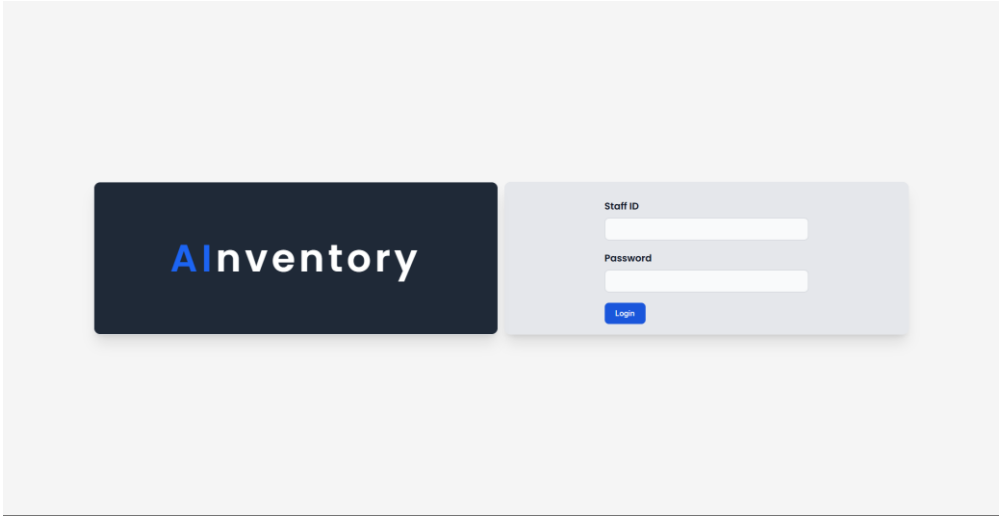

*Figure 46:*Screenshot of web browser accessing system

4.5 Critical Function Implementation

This section is dedicated to elaborating on the critical components of the system that contribute directly to its core functionality. These include the AI-powered chatbot module, role-based access control mechanisms, and the inventory input and withdrawal operations. Each of these subsystems was implemented with a focus on usability, responsiveness, and secure access to ensure smooth and practical everyday use.

One of the most significant features in the system is the integration of the AI-powered chatbot. This module enables users to interact with the inventory system using natural language queries instead of traditional dropdowns or filters. The chatbot interface was implemented using a simple JavaScript frontend that listens for form submissions and sends the user's query to a PHP backend endpoint via an AJAX request. The backend script then takes this query and passes it to a processing module that performs two key tasks: first, it parses the query and constructs a relevant SQL command; second, it forwards the resulting raw data into a prompt to be submitted to OpenAI's GPT API. The API then returns a human-readable explanation or insight, which is displayed in the chat interface.

The two-step logic, which is a query generation followed by natural language interpretation, allows the system to function as a virtual assistant capable of understanding user intent. For example, when a user types "Which items are running low?" the backend identifies this as a stock-level query, fetches all items below a set threshold, and returns a summary like "Keyboard and mouse set | 0" along with 9 others low stock items. This approach bridges the gap between raw database access and easy to understand interface.

Another critical function is the role-based access control (RBAC) system, which governs the level of access available to users based on their assigned role. During authentication, the system checks the role field in the users table after validating login

credentials. A role value of 0 designates a standard user, while 1 designates an admin. Depending on this value, the interface will show a completely different UI. For instance, only admin users can access pages for user account management or approve pending withdrawal requests. This logic is enforced by backend via denying unauthorized access if a user tries to navigate to a restricted page manually. This dual-layer protection ensures both usability and security.

Equally important are the item input and withdrawal modules. Users can register new items into the inventory using a structured form that records the item name, category, quantity, remarks, etc. All data is then written to the item_in table. Similarly, when a user wishes to withdraw items, a request is generated and logged in the approval table. This request is then reviewed by an admin, who can either approve or reject the transaction. Approved requests result in the actual deduction of the item quantity from the items table, and a record is created in the item_out table for historical tracking. This approval mechanism ensures accountability and traceability of every stock movement.

All of these modules are tightly integrated through consistent UI design and logical flow. The chatbot can access the same data as the dashboard forms, meaning users can both interact with the system through structured workflows or simply ask questions and get results instantly. This flexibility caters to different user preference, some may prefer traditional form entries while others might benefit from the conversational assistant, especially when they are unfamiliar with specific stock categories or report filters.

To assist with code traceability, screenshots of key sections are provided. These include the JavaScript that captures chatbot queries, the PHP logic that generates SQL from natural language, and the API request handler that communicates with OpenAI.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>AI Chatbot</title>
<script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-white text-gray-900">
<div class="flex justify-center items-center min-h-screen">
    <div class="bg-white rounded-xl border border-gray-200 shadow w-full max-w-md flex flex-col overflow-hidden">

    <!-- Header -->
    <div class="bg-blue-600 border-b border-gray-200 text-white py-3 px-4 text-lg font-semibold flex items-center space-x-2">
        <svg xmlns="http://www.w3.org/2000/svg" class="w-6 h-6 text-white fill="currentColor" viewBox="0 0 24 24">
        <path d="M2 5a2 2 0 0 1 2-2h16a2 2 0 0 1 2 2v12a2 2 0 0 1-2 2H6l-4 4V5z"/>
        </svg>
        <span>AI Chatbot</span>
    </div>

    <!-- Chat area and input -->
    <div class="flex flex-col p-4 space-y-4 h-[500px]">
        <div id="chat-box" class="flex-1 overflow-y-auto bg-gray-50 rounded-lg p-4 text-sm text-gray-800 space-y-3">
        <!-- Messages will be appended here -->
        </div>

        <form id="chat-form" class="flex space-x-2">
        <input type="text" id="user-input" placeholder="Type a message..." required
            class="flex-grow rounded-full border border-gray-300 px-4 py-2 focus:outline-none focus:ring-2 focus:ring-blue-400">
        <button type="submit"
            class="bg-blue-500 text-white px-4 py-2 rounded-full hover:bg-blue-600 transition">Send</button>
        </form>
    </div>
    </div>
</div>
```

Figure 47:Screenshot of JavaScript function capturing chatbot input and sending AJAX request

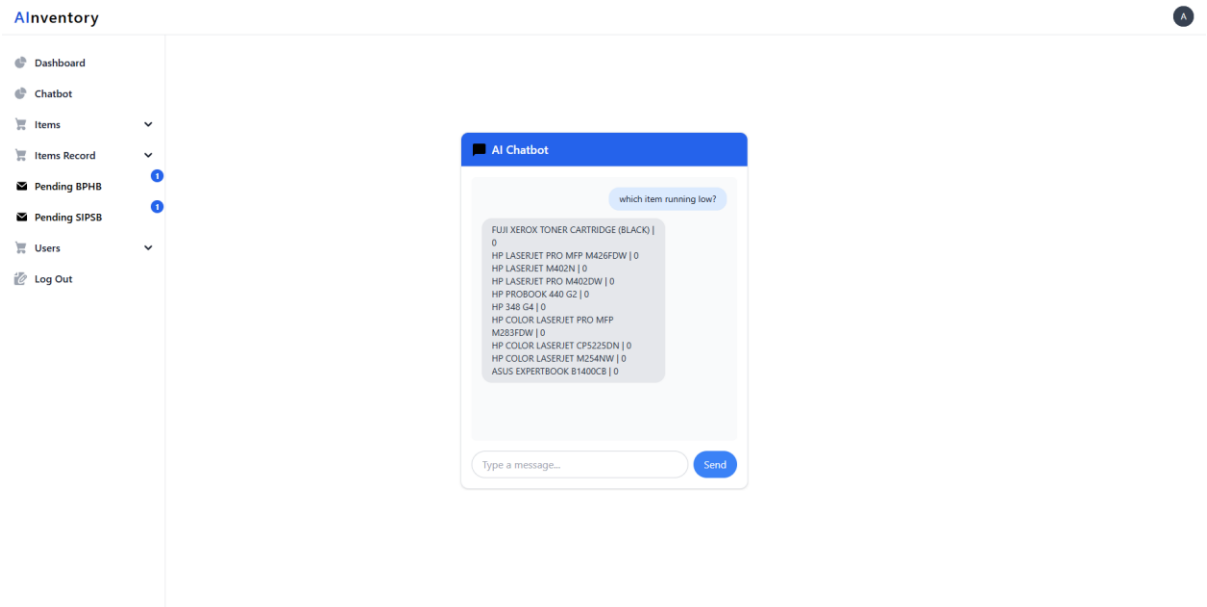

Figure 48: Screenshot of PHP script receiving chatbot input, generating SQL, and formatting OpenAI prompt

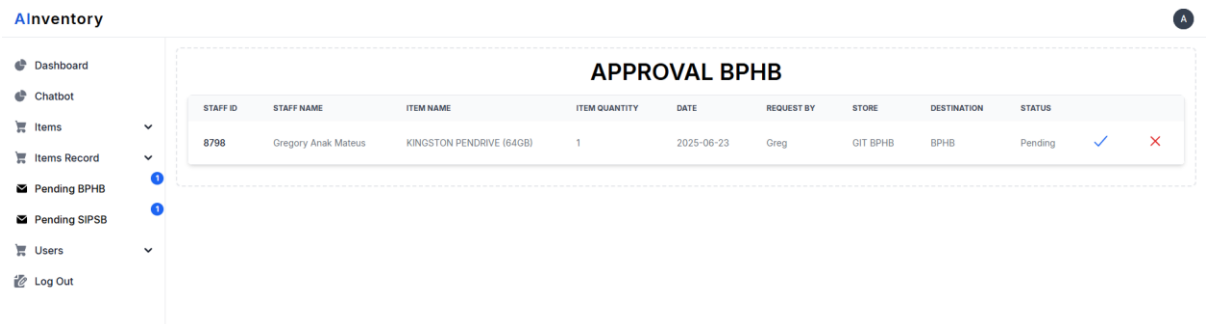

Figure 49: Screenshot of admin interface showing approval panel with approve/reject buttons

4.6 Front-end Implementation

The front-end of the system plays a vital role in shaping the user experience, ensuring that interactions are both intuitive and visually coherent. A consistent and user-friendly interface can significantly improve task efficiency and reduce user fatigue, especially in systems that require regular data input and monitoring. The interface of the AI-Enhanced Inventory Management System was developed using HTML and styled with Tailwind CSS, while JavaScript handled the dynamic behaviours and asynchronous interactions with the backend.

Tailwind CSS, a utility-first framework, was chosen due to its speed of implementation and responsiveness across devices. It allowed the system to maintain a clean, consistent look without the overhead of writing custom CSS from scratch. Tailwind's grid systems and spacing utilities were used extensively to structure dashboard components such as stock tables, input forms, and user controls. This ensured that elements were properly aligned and accessible even on smaller screens.

JavaScript, primarily using ES6 features, was integrated to enhance interactivity. For example, the system dashboard includes charts that visually represent stock trends, item categories, and inventory fluctuations over time. These charts are rendered using libraries such as Chart.js or ApexCharts.js, which integrate seamlessly with real-time data fetched from the backend. The use of such visual aids allows users and admins to quickly interpret system status without combing through rows of text.

One of the most dynamic areas of the front-end is the chatbot interface. This module is implemented using a simple text input box accompanied by a "Send" button. When the user enters a query, JavaScript captures the input, displays the message in the conversation window, and then transmits the query to the backend asynchronously via AJAX. This method avoids

page reloads and ensures that the conversation appears fluid and real-time. Once a response is received, it is parsed and inserted into the chat thread, complete with timestamps and alternating visual bubbles to differentiate between user and system messages.

In terms of user flow, the system's landing page presents a login interface. Upon successful login, users and admins are both redirected to their respective dashboards, where they are greeted by a header, navigation pane, and segmented content areas. Each section was developed to be context-aware; for instance, if the admin has any pending approvals, the pending section will light up with the number of pending requests there is. This helps to make sure the admin is aware of any outgoing requests by the user.

Accessibility was also a consideration during development. Button sizes, contrast ratios, and font choices were adjusted to meet basic web accessibility standards. Hover states and tooltips were added to guide users unfamiliar with specific icons or abbreviations. While screen reader optimization was not fully implemented due to time constraints, the current layout is designed with semantic HTML, allowing future improvements to be added incrementally.

Overall, the front-end implementation is designed to complement the functionality of the system with a layout that feels familiar, responsive, and easy to navigate. It balances the needs of both technical and non-technical users, offering advanced features like the AI chatbot while retaining traditional dashboards and forms for standard operations.

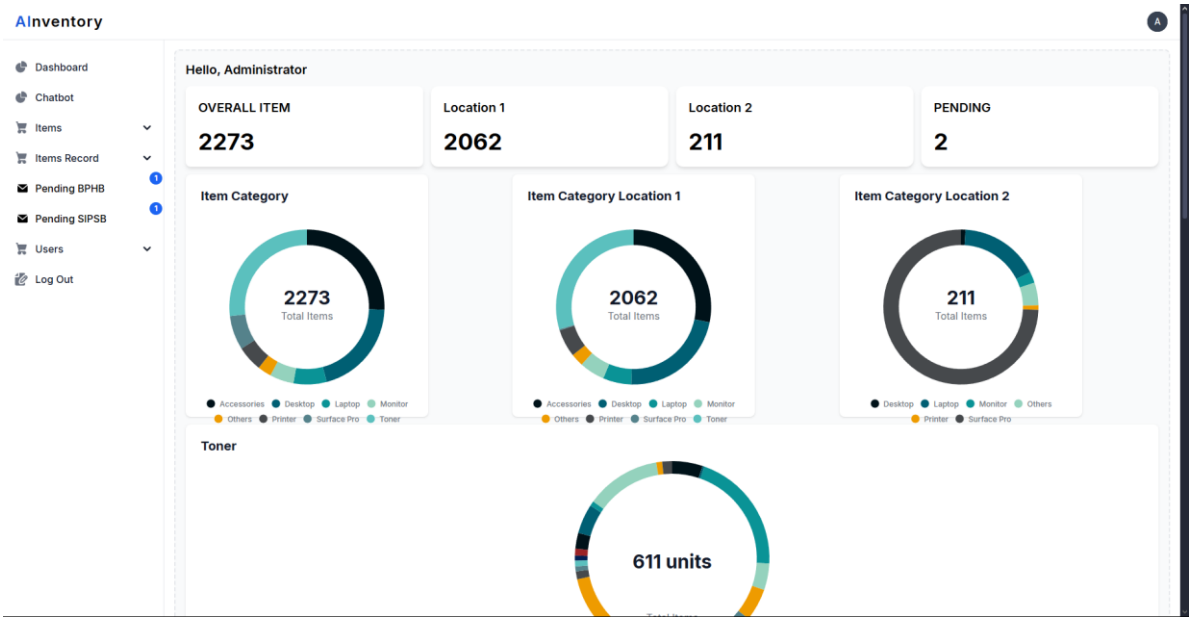

Figure 50: Screenshot of system dashboard with stock summary and navigation bar

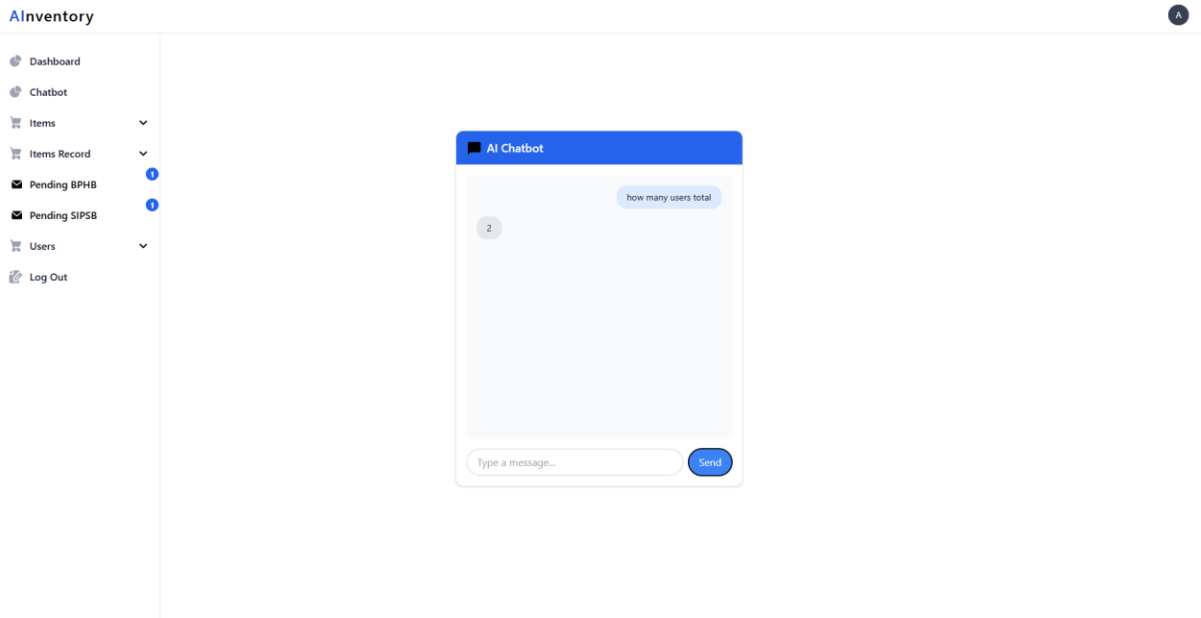

Figure 51: Screenshot of chatbot interface in action showing a user query and response

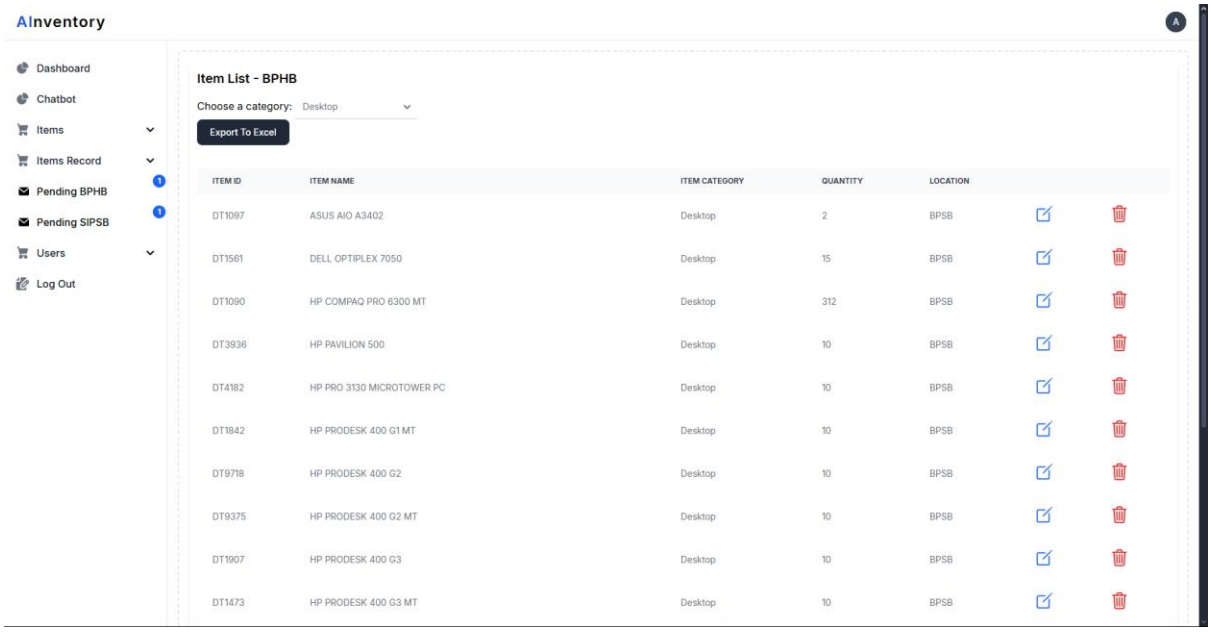

Figure 52: Screenshot of table displaying stock levels by item category

4.7 Back-end Implementation

The back-end of the system serves as the core engine that processes logic, handles database operations, manages user sessions, enforces role-based access control, and interfaces with external APIs such as OpenAI. It is developed entirely in PHP, a widely adopted server-side scripting language that provides flexibility, robustness, and ease of integration with MySQL. The backend operates within the XAMPP environment, with Apache handling HTTP requests and MySQL functioning as the database engine.

The authentication system is one of the first points of interaction between the user and the backend. Upon submitting login credentials, a PHP script verifies the information against stored records in the users table using SQL queries. If the credentials match, a session is created using PHP's built-in session management, and the user is given a role before being redirected to the appropriate dashboard. The session stores key variables such as user_id, name, and role, which are used to personalize the experience and restrict access to certain pages.

Role-based access control is implemented in all key modules. Every secure page includes a session check at the top, ensuring that only authenticated users with the appropriate role can proceed. For instance, if a regular user tries to access the admin approval panel, the PHP logic immediately terminates the process and redirects the user to the login page. This protects sensitive functions from unauthorized access and maintains data integrity.

CRUD operations (Create, Read, Update, Delete) are central to the system's database interactions. Each operation is handled through dedicated PHP scripts that accept user input from HTML forms, sanitize the input using mysqli_real_escape_string() and other validation methods, and execute SQL statements against the database. For example, the item insertion module accepts new stock data and adds a record to the item_in and items tables, while the withdrawal module checks current stock levels before decrementing item counts and logging the transaction in the item_out and approval tables.

The chatbot module represents a unique backend feature due to its use of dynamic SQL generation and external API communication. When a chatbot query is received, the AI will interpret the message and maps it to a specific SQL command using predefined rules or keyword detection. Once the SQL is executed, the result is formatted into a prompt and sent to the OpenAI API using PHP's cURL library. The returned message is parsed and echoed back to the frontend. This logic is encapsulated in functions that handle errors, timeouts, and fallback responses if the API is unreachable.

Backend operations also include data formatting for frontend visualization. Some PHP scripts are dedicated to exporting data in JSON format to support JavaScript-based graphs and tables. These endpoints enable real-time updates to charts reflecting stock trends, item categories, or user activity without requiring full-page reloads. Reports can also be downloaded

in Excel format, generated through third-party PHP libraries that convert database queries into spreadsheet outputs.

Security is a recurring theme in backend development. Input validation, SQL injection prevention, session expiration, and API key protection were all implemented to harden the system against common vulnerabilities. For example, the OpenAI API key is stored in a configuration file located outside the root directory and loaded only on the server side.

Scalability was also considered in the backend design. All critical scripts were modularized to separate business logic from presentation code. This makes it easier to maintain the application, test specific features, and eventually port it to frameworks like Laravel should the system expand. Commenting and naming conventions were followed strictly to ensure clarity and consistency.

```php
<?php
session_start();
include('connection.php'); // Assuming connection.php contains your database connection code
include('authenticate.php'); //to aunthenticate user

// Ensure only admin has access
if (!isAdmin()) {
    // Redirect to user dashboard if not an admin
    header('Location: ./user_dashboard.php');
    exit();
}

// Check if the user is logged in
if (!isset($_SESSION['staff_id'])) {
    // Redirect to the login page
    header("Location: Login.php");
    exit(); // Stop further execution
}

// User is logged in, continue with the page content
$staff_id = $_SESSION['staff_id'];
// Fetch user information from the database
$query = "SELECT * FROM users WHERE staff_id='$staff_id'";
$result = $conn->query($query);

if ($result->num_rows == 1) {
    $user = $result->fetch_assoc();
    $name = $user['name'];
    $email = $user['email'];
} else {
    echo "<h1>User not found</h1>";
}
?>
```

Figure 53:Screenshot of PHP login authentication logic verifying user credentials

```
<script>
    document.addEventListener('DOMContentLoaded', function () {
    const chatBox = document.getElementById('chat-box');
    const chatForm = document.getElementById('chat-form');
    const userInput = document.getElementById('user-input');

    function createMessageBubble(text, isUser = false) {
        const wrapper = document.createElement('div');
        wrapper.className = `flex ${isUser ? 'justify-end' : 'justify-start'}`;

        const bubble = document.createElement('div');
        bubble.className = `max-w-[75%] px-4 py-2 rounded-2xl whitespace-pre-wrap ${isUser ? 'bg-blue-100 text-gray-900' : 'bg-gray-200 text-gray-800'}`;
        bubble.textContent = text;

        wrapper.appendChild(bubble);
        chatBox.appendChild(wrapper);
        chatBox.scrollTop = chatBox.scrollHeight;
    }

    async function fetchReply(message) {
        createMessageBubble(message, true);

        const loadingWrapper = document.createElement('div');
        loadingWrapper.className = 'flex justify-start';
        const loadingBubble = document.createElement('div');
        loadingBubble.className = 'max-w-[75%] px-4 py-2 rounded-2xl bg-gray-200 text-gray-800';
        loadingBubble.textContent = '...';
        loadingWrapper.appendChild(loadingBubble);
        chatBox.appendChild(loadingWrapper);
        chatBox.scrollTop = chatBox.scrollHeight;

        try {
        const response = await fetch('chatback.php', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ message })
        });

        const data = await response.json();
        chatBox.removeChild(loadingWrapper);
        createMessageBubble(data.reply || 'No reply received');
```

Figure 54: Screenshot of chatbot PHP script sending user input directly to OpenAI API

4.8 Security Considerations

Security plays a vital role in the development of any information system, particularly when sensitive inventory data and user accounts are involved. In this project, multiple security mechanisms were embedded at various levels of the stack—from input validation and session management to role-based access control and secure API integration.

The first and most foundational aspect of security is user authentication and session handling. When users log into the system, their credentials are matched against stored entries in the user table. Upon successful authentication, a secure PHP session is created to maintain the user's login state. The session stores key details such as the user's ID and role, which are later used to determine access permissions. If a session is not detected, the system will automatically redirect users to the login page. This ensures that only authenticated users can access protected resources.

To further prevent unauthorized access, role-based access control (RBAC) was implemented across all modules. Admin-only pages and approval functions are protected by conditional checks based on the user's role. These checks are enforced not just on the user interface, where buttons and navigation items are hidden from unauthorized users, but also on the server side, where unauthorized access attempts result in redirection or denial messages. This layered enforcement ensures that access control cannot be bypassed through URL manipulation or browser tools.

Input validation and sanitization are crucial for preventing common vulnerabilities such as SQL injection and cross-site scripting (XSS). All user input fields across login forms, stock entry fields, or chatbot messages are sanitized using PHP's built-in functions like htmlspecialchars() and mysqli_real_escape_string(). This ensures that special characters are escaped and malicious payloads are neutralized before being processed or passed to the database.

With the system integrated with the OpenAI API, it was essential to ensure secure handling of the API key. The key is never exposed in the client-facing code; instead, it is stored in a secure configuration file located outside the public web directory. Backend scripts retrieve this key internally when making API calls, ensuring that even in the event of client-side compromise, the API key remains protected. Additionally, timeouts and error handlers are used to manage API responses and avoid exposing sensitive error messages to the user. Logout functionality is also provided and thoroughly tested to ensure session data is properly cleared.

Database-level security is also considered. The user connected through the PHP application is granted only the minimum necessary privileges. For example, it can only read and write to certain tables and is restricted from executing administrative commands. This principle of least privilege helps contain the damage in case the credentials are ever leaked.

Lastly, security during file uploads, such as Excel report generation, is monitored. The system ensures that only safe file formats are created, and all generated files are temporary, stored with controlled access. No user-uploaded files are accepted, thereby eliminating file upload vulnerabilities entirely.

Together, these considerations form a robust security framework that protects both user data and system integrity. While the system operates on a local host during development, its security architecture is designed to be scalable and resilient enough to be deployed in real-world, internet-facing environments with minimal adjustments.

Chapter 5: Testing and Evaluation

5.1 Introduction

Testing is a crucial phase in any software development project, including academic final year projects (FYPs), as it ensures the system meets its functional, performance, and usability expectations. For the AI-Enhanced Inventory Management System developed in this project, testing plays a significant role in validating the robustness, reliability, and accuracy of features such as user authentication, role-based access, inventory operations, and the integration of the OpenAI-powered chatbot.

The primary goal of the testing process was to simulate real-world scenarios in a controlled environment and assess how the system behaves under different conditions. This includes checking data integrity, enforcing security mechanisms, evaluating the performance of the chatbot, and ensuring that user roles are properly implemented. Given the solo nature of the project, a structured manual testing checklist was employed to carry out these evaluations. This approach, while simple, allows for systematic verification and documentation of key functions across the system.

5.2 Testing Methodology

Manual testing was chosen for this project due to its suitability in small-scale systems where automated tools are either impractical or unnecessary. It involves the tester (the developer in this case) manually executing functions, entering inputs, and observing system behavior to compare it against expected outcomes. This method is particularly useful for usability evaluation and for detecting interface-level issues.

All tests were conducted in a local development environment using XAMPP running on a Windows 11 laptop with 16GB RAM and an intel core i5 12^{th} gen. The browser used for front-end testing was Opera GX, with developer tools enabled to inspect console logs and network requests. Testing was carried out in multiple phases: functional testing, integration testing, and user flow validation.

Test cases were first planned by listing all major system functions. Each function was analyzed to determine critical scenarios, both valid and edge cases. For each scenario, the preconditions, execution steps, and expected results were documented. These test cases were compiled into a checklist that includes status tracking and comments for anomaly logging.

5.3 Manual Testing Checklist

Test Area	Pre-conditions	Test Steps	Expected Result	Status (✓/✗/~)	Remarks
Authentication	User account exists	Login with valid credentials	User is redirected to dashboard	✓	Works as intended
Authentication	User not logged in	Access dashboard URL directly	Redirected to login page	~	Error occurs where localhosts redirect user too much
Role Access	User logged in as admin	Accessing approval page	Pending approval page loads correctly	✓	Admin controls visible
Role Access	User logged in as normal user	Try accessing /users	Redirected or access denied	✓	Error where localhosts redirect too much
Inventory	Logged in as user	Add a new item into inventory	Item is listed under current stock	✓	Item added correctly

Inventory	Logged in as user	Submit withdrawal request	Request appears under approval	✓	Pending status logged
Approval	Admin has pending request	Approve request in panel	Stock level decreases accordingly	✓	Update shown in item_out
Chatbot	System online	Ask chatbot: “How many items in total”	Returns total item amount in store	✓	Query returned correctly
Report Export	Admin/user logged in	Click download Excel	Correct file is downloaded	✓	Excel readable and structured

Figure 55: Manual Testing Checklist

5.4 Test Result Analysis

The test results indicated that the system performs reliably in handling all critical operations. Authentication and session validation were found to be robust, ensuring only verified users could access protected routes. Role-based control effectively limited access to admin-only features and worked as intended even when users attempted to bypass restrictions manually.

Inventory-related operations such as item insertion and withdrawal functioned accurately, with all actions being correctly recorded in the associated database tables. Admin approval logic was correctly triggered, and item count updates were consistently reflected. The chatbot interface successfully handled a variety of queries, especially those related to stock counts and summaries, though some queries showed varied response formats depending on how clearly the question was phrased.

The Report exporting feature is working as intended where the file is converted into an excel format and then saved as item_records.csv. Upon opening all the details of the item in or out is recorded into an excel spreadsheet with no missing data.

5.5 Usability Evaluation

From a usability perspective, the system was straightforward to navigate. The dashboard layout was clean, and the use of Tailwind CSS ensured a responsive interface. Key functions were placed prominently, and icons, labels, and buttons were intuitive.

The chatbot added an extra layer of accessibility, particularly for users who prefer natural language queries over navigating multiple pages. It demonstrated acceptable accuracy in interpreting inventory questions. However, some users unfamiliar with phrasing or technical terms might find it necessary to adjust their queries to achieve the desired outcome plus the accuracy of the AI is subpar as it requires the name of the item to be specific, where it does not tolerate misspelling well.

Overall, the user experience was fluid, and users with minimal technical background would be able to perform inventory tasks comfortably. Improvements could include adding tooltips, command suggestions, or a simple user manual embedded into the system interface.

5.6 User Survey and Quantitative Results

To complement manual testing and qualitative feedback, a structured user survey was conducted to gather quantitative data regarding system usability, interface clarity, chatbot performance, and other key dimensions. The survey used a Likert scale ranging from 1 (Very Poor) to 5 (Excellent), and participants were asked to rate various evaluation criteria after engaging with the system. A total of 10 users participated in the survey, representing different roles including admin users, inventory staff, and first-time users.

The table below summarizes the average scores and standard deviations for each criterion. This statistical evaluation complements the manual checklist by identifying areas of consistent performance and aspects requiring future enhancement.

Evaluation Criteria	Average Score (1–5)	Standard Deviation
Ease of navigation	4.6	0.49
Inventory entry simplicity	4.7	0.45
Clarity of dashboard	4.5	0.51
Responsiveness of system	4.4	0.48
Chatbot accuracy	4.2	0.57
Chatbot usability	3.9	0.66
Access control effectiveness	4.6	0.40
Excel export clarity	4.3	0.52
Overall satisfaction	4.5	0.50

Figure 56: Average User Survey Score

5.6.1 Discussion of Survey Data

The quantitative survey data reveals important patterns in user sentiment. Scores above 4.5 in areas such as navigation, inventory handling, and dashboard clarity indicate that the system provides a smooth and accessible experience for most users. This aligns well with the manual testing checklist, reinforcing the reliability of core functions.

Lower scores for chatbot accuracy and chatbot usability suggest specific improvement areas. Users likely found handling the bot to be a bit of a nuisance if they do not know exactly what to query, which can only be improved with even more rigorous commands to the AI as it still sometimes calls false table name and this needs to be further refined by getting more feedback. Chatbot usability scores, remaining the lowest of the average score, indicate that novice users may require better guidance on phrasing queries as it is still relatively case

sensitive. These insights are crucial for future system iterations, allowing targeted refinements that increase both chatbot accuracy and user confidence.

5.7 Evaluation Summary

This chapter has detailed the full evaluation process of the AI-Enhanced Inventory Management System. Testing involved both systematic manual validation and user-centric survey assessment. The manual checklist confirmed that most of the core modules, from login and inventory functions to chatbot interaction and data export operate reliably and meet intended specifications, except for some redirection errors for when trying to manually head to a page name without the proper authentication, however ultimately it does the job at barring unauthorized access.

The survey results supported these findings, showing strong user satisfaction with the system's navigation, visual design, and functional layout. Notable areas for improvement include the chatbot accuracy and the usability of the chatbot interface for less technical users. These reflections suggest that the system is almost production-ready aside from the chatbot feature, which requires even more refinement to catch queries correctly that will contribute to user satisfaction, with many opportunities for both minor and major refinement in future development cycles.

In conclusion, the testing and evaluation process demonstrated that the system is stable, functional, and user-friendly. With the inclusion of feedback-based improvements, it can offer an efficient and intuitive platform for inventory operations in academic or SME environments.

Chapter 6: Conclusion

6.1 Introduction

This project was proposed and developed to address common challenges faced by small to medium-scale inventory systems, particularly in SME environments. By integrating a web-based inventory management system with a conversational AI component, the system aims to improve both the accessibility and efficiency of inventory operations. The core innovation lies in enabling non-technical users to retrieve inventory data through the AI chatbot, while still keeping the style of a traditional inventory management system for more experienced users. But ultimately this system is meant to ease in users who are used to doing manual spreadsheets to an easy to understand system to help with inventory.

The use of PHP for backend logic, Tailwind CSS for frontend interface, and OpenAI's API for AI integration has resulted in a system that is responsive, practical, and relatively lightweight, with most of the heavy workings done by the API calls. This project showcases how conversational AI can be combined with traditional inventory tracking systems to offer both usability and intelligence in one cohesive platform.

6.2 Objectives and Achievements

The first objective was to develop a role-based inventory management system. This was successfully accomplished through the implementation of user-level authentication, where users are assigned different roles, namely admin or regular user. Admins are tasked to manage

user access, approve inventory withdrawal requests, and generate reports, while users can input item data, make withdrawal requests and generate reports.

The second objective was to implement a conversational AI chatbot that interprets user queries and responds with real-time inventory data. This objective was also met by integrating the OpenAI API into the system, enabling the chatbot to transform user input into SQL commands, query the database, and return human-readable responses. A more thorough integration of the chatbot is definitely necessary to make the query requirement more lax as the query is still search sensitive but ultimately it is functioning.

The third objective focused on system usability and security. The system includes session-based login validation, role restrictions, and frontend dashboards designed for ease of use. User feedback and testing confirmed that these features performed reliably and were well received.

Overall, all core objectives were met, with the system operating as intended under testing conditions and successfully providing both manual and conversational methods for managing inventory.

6.3 Discussion

The technical implementation of the system demonstrates how simple open-source tools can combine and work together to deliver powerful, semi-intelligent systems. The use of PHP 8.1 and MySQL provided a robust and familiar environment for logic and data handling, while Tailwind CSS helped create a user interface that was both clean and responsive.

One of the most notable achievements of this system was the successful integration of the chatbot interface. Unlike traditional interfaces, the chatbot removes the need for specific UI elements for each function, instead allowing users to access data through natural conversation. Although the chatbot's accuracy is dependent on prompt clarity, the overall response was effective in most testing scenarios.

Additionally, the decision to use XAMPP for local hosting ensured that the system remained accessible and testable without the need for live deployment. The system can still work under local condition, but it is strongly recommended to use a live web hosting, preferably a company server. However, the local XAMPP configuration allows for quick debugging and system demonstrations, especially in environments without stable internet connectivity.

The modular design of the backend and the strict separation of user roles further enhanced the system’s maintainability and reliability. Throughout development, best practices for security and database interaction were followed, with attention given to input validation, session control, and safe API usage.

6.4 Constraints and Limitations

Despite its successes, the project encountered several limitations. The most notable constraint was the reliance on OpenAI’s API for chatbot functionality. This means the system requires an active internet connection and a valid API key, which would render the chatbot useless under no active internet connection.

Another limitation is the absence of a session timeout feature, which was initially planned but not implemented. As a result, users who remain idle are not automatically logged out, which could pose a security risk in shared or public access environments. This would

require the workplace to instead have a policy to log out or lock your laptops when you are not there personally, which is what most workplace has already implemented.

Lastly, from a usability standpoint, the chatbot sometimes struggles with ambiguous queries or non-standard phrasing. While this is more a limitation of AI interpretation, it does affect user experience and require significant training on the API logic in future versions.

6.5 Future Works and Recommendations

There are several opportunities for improving and expanding the current system in future iterations.

First, a session timeout and auto-logout feature should be implemented to enhance security and protect unattended sessions. This can be achieved using simple session tracking in PHP and JavaScript-based inactivity timers.

Second, the chatbot could be enhanced by adding a built-in prompt guide or autocomplete suggestions to help users frame more accurate queries. Additional refinement of prompt handling and custom training may also boost the reliability of the AI responses.

Third, an integrated dashboard for chatbot queries, showing previous prompts and suggested follow-ups, would improve interaction flow and usability. As of now, the chatbot does not have a logs feature and lack a suggestion for follow-ups. Additional language support and voice-based queries could also enhance accessibility for users with different needs.

Lastly, implementing a notification system directed to staff email to alert for pending approvals would increase the system's practicality in real-time management scenarios.

Currently the system does not have this feature and when the user wants to have their pending items approved, they would have to directly ask the admins to approve it. Which is not efficient.

6.6 Summary

The AI-Enhanced Inventory Management System successfully met its 3 core objectives by combining traditional stock tracking with AI-driven query handling. Through clear role-based access control, clean UI design, and a working AI chatbot, the system demonstrates how intelligent inventory systems can be developed using open-source tools and.

User testing and feedback validated the effectiveness of the implementation, while also revealing vital areas for improvement such as session timeout control and requiring a more guided AI usage. As inventory systems continue to evolve, the integration of AI will likely become a standard feature, with this project offering a meaningful foundation for such developments.

With improvements in deployment flexibility, security enhancements, and more advanced AI integration, this system has the potential to serve as a reliable and smart inventory solution for SME businesses and if the potential is fully realized, it might even spread to big enterprises or become a stepping stone in the integration of AI in inventory management systems worldwide.

References

1. Feuerriegel, S., Hartmann, J., Janiesch, C. et al. (2024). Generative AI. Bus Inf Syst Eng 66, 111–126. https://doi.org/10.1007/s12599-023-00834-7
2. Shekhar, A., Prabhat, P., Yandrapalli, V., Umar, S., Abdul, F., & Wakjira, W. D. (2023). Generative AI in Supply Chain Management. ISSN: 2321-8169 Volume: 11 Issue: 9. https://doi.org/10.17762/ijritcc.v11i9.9786
3. Albayrak Ünal, Ö., Erkayman, B., & Usanmaz, B. (2023). Applications of artificial intelligence in inventory management: A systematic review of the literature. Archives of Computational Methods in Engineering, 30(4), 2605-2625. https://doi.org/10.1007/s11831-022-09879-5
4. Patil, O., Kulkarni, H., Pottavathini, R., Dhole, I., & Salgaonkar, K. (2024). Real Time Inventory Management System powered by Generative User Interface.
5. Simpson, M. D., & Qasim, H. S. (2025). Clinical and Operational Applications of Artificial Intelligence and Machine Learning in Pharmacy: A Narrative Review of Real-World Applications. Pharmacy, 13(2), 41. https://doi.org/10.3390/pharmacy13020041
6. Subramanian, S., Mahajan, S., & Kolhar, S. (2025). Inventory Management System Using Reinforcement Learning: A Case Study. Ingénierie des Systèmes d'Information, 30(4). https://www.iieta.org/journals/isi/paper/10.18280/isi.300401
7. Patrício, L., Varela, L., & Silveira, Z. (2024). Integration of Artificial Intelligence and Robotic Process Automation: Literature Review and Proposal for a Sustainable Model. Applied Sciences, 14(21), 9648. https://doi.org/10.3390/app14219648
8. Machucho, R., & Ortiz, D. (2025). The Impacts of Artificial Intelligence on Business Innovation: A Comprehensive Review of Applications, Organizational Challenges, and Ethical Considerations. Systems, 13(4), 264. https://doi.org/10.3390/systems13040264

9. Shireen Al-Hourani, & Dua Weraikat. (2025). A Systematic Review of Artificial Intelligence (AI) and Machine Learning (ML) in Pharmaceutical Supply Chain (PSC) Resilience: Current Trends and Future Directions. Sustainability, 17(14), 6591–6591. https://doi.org/10.3390/su17146591
10. Alzeyani, E. M., & Szabó, C. (2023). A Study on the Effectiveness of Agile Methodology Using a Dataset. Acta Electrotechnica et Informatica, 23(1), 3-10. https://doi.org/10.2478/aei-2023-0001

Appendix

Appendix A: Table of Questions for User Survey

No.	Feedback Question	Rating Label / Format
1	How easy was it to navigate the main dashboard and access your assigned functions?	1 (Very Poor) – 5 (Excellent)
2	How clear and understandable were the labels, buttons, and menus throughout the system?	1 (Very Poor) – 5 (Excellent)
3	How would you rate your overall ease of using the system without assistance?	1 (Very Poor) – 5 (Excellent)
4	How simple was it to add or update inventory entries through the provided forms?	1 (Very Poor) – 5 (Excellent)
5	Did you encounter any errors or confusion when inputting item data (e.g., item name, quantity)?	Yes / No
6	How would you rate the system's response speed during inventory-related tasks?	1 (Very Poor) – 5 (Excellent)
7	How visually appealing and organized did you find the dashboard layout?	1 (Very Poor) – 5 (Excellent)
8	Were the graphical elements (charts, cards, tables) helpful in understanding your inventory status?	1 (Very Poor) – 5 (Excellent)
9	How accurate were the chatbot's responses in answering your inventory-related questions?	1 (Very Poor) – 5 (Excellent)

No.	Feedback Question	Rating Label / Format
10	How easy was it to interact with the chatbot using your own words?	1 (Very Poor) – 5 (Excellent)
11	Did the chatbot's replies meet your expectations in terms of usefulness and clarity?	1 (Very Poor) – 5 (Excellent)
12	How comfortable would you be using the chatbot to retrieve data instead of navigating manually?	1 (Very Poor) – 5 (Excellent)
13	How well do you feel the system handled role-based access control (e.g., limiting features based on login role)?	1 (Very Poor) – 5 (Excellent)
14	Did the system restrict you from accessing pages or functions that you were not authorized for? (Yes/No)	Yes / No
15	How clearly formatted was the Excel report export feature?	1 (Very Poor) – 5 (Excellent)
16	Did you successfully export inventory reports without encountering issues?	Yes / No
17	How would you rate the session behaviour (e.g., staying logged in, timing out, responsiveness)?	1 (Very Poor) – 5 (Excellent)
18	How satisfied are you overall with the system's performance and features?	1 (Very Poor) – 5 (Excellent)
19	Would you recommend this system to other staff or businesses with similar inventory needs?	1 (Very Poor) – 5 (Excellent)
20	Do you have any suggestions or comments to improve the system? (Optional)	Open-ended