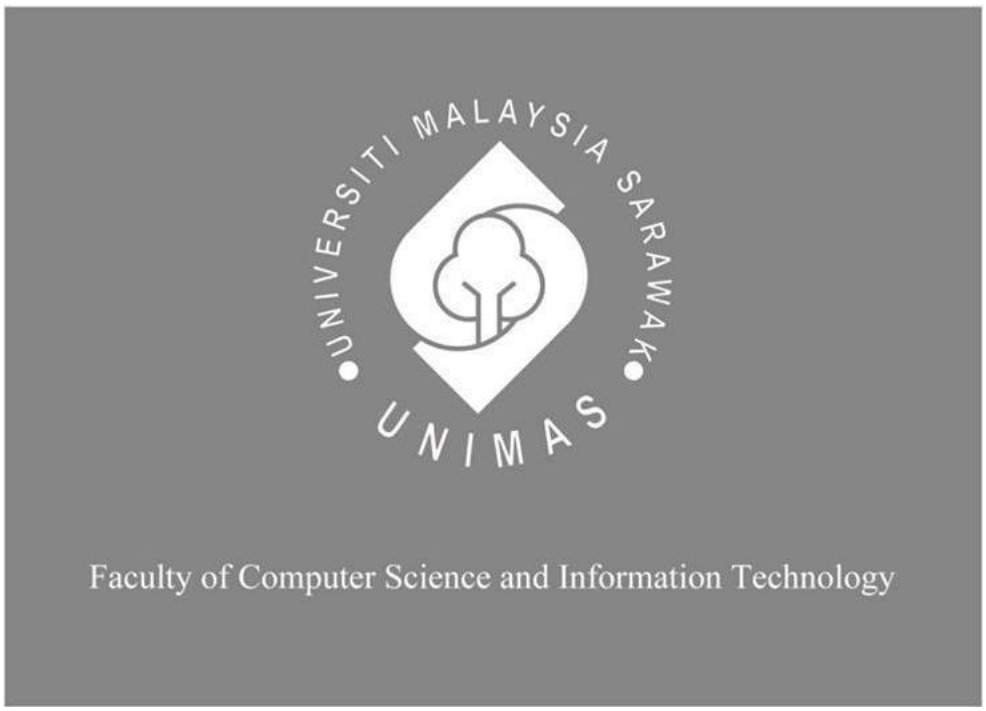

HALALSCAN: A WEB APPLICATION FOR HALAL INGREDIENT VERIFICATION

Misha Athira Binti Zahrul

Bachelor of Computer Science with Honours

(Information Systems)

2025

DECLARATION

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to Allah SWT for granting me strength, resilience and rezeki to complete this final year project. Without His guidance and blessings, none of this would have been possible.

I would like to sincerely thank my supervisor, Madam Hamizan Sharbini for her unwavering support, valuable insights, and guidance throughout the development of this project. Her suggestions and solutions, especially during moments of uncertainty, were instrumental in helping me move forward.

To my university friends, thank you for being more than just peers. Your willingness to teach me, provide feedback on the system's design and flow, participate as testers, and offer motivation and emotional support meant the world to me.

To my family and hometown friends, your belief in me never wavered even when I struggled and doubted myself. Your constant encouragement, especially during moments when I said, "I can't do this", reminded me to keep going and return home proud and safe.

Lastly, to myself, thank you for choosing to find solutions before sleep instead of giving up. Thank you for waking up and showing the next day to fix yesterday's problems. Most of all, thank you for not giving up, and for always believing that there is a way forward, even when it felt like a dead end. I am proud of the knowledge I have gained and the growth I have experienced through this journey. Despite the tears and the challenges, I truly enjoyed this process. Here's to every late night that ended with a breakthrough, every bug that taught a lesson, and every step that brought me closer to the finish line. I did this and I am proud.

ABSTRACT

The increasing concern among Muslim consumers regarding the halal status of imported products has driven the need for a more efficient method of ingredient verification. Thus, this project introduces HalalScan, a web-based system that allows users to determine the hall status of food products by scanning or uploading an image of the ingredient list. The system uses Optical Character Recognition (OCR) powered by Tesseract OCR to extract text from the image and cross-checks it with a curated ingredient database. Users are then presented with color-coded status indicators (green, yellow, red) representing *halal*, doubtful, or *haram* ingredients in the context of Islamic dietary laws. The system also includes a user-friendly interface, ingredient detail popups, a statistical page, and an admin panel for database management. Usability testing involving 13 participants of varied background showed that 92.3% of users rated the system with the highest satisfaction score, and most tasks were completed smoothly without assistance. Only the OCR scanning task posed minor challenges for a user's, due to unfamiliarity with camera handling. However, limitations related to OCR accuracy under certain conditions were identified. Despite this, HalalScan demonstrates a promising approach to simplifying and digitalizing halal ingredient verification.

ABSTRAK

Kebimbangan dalam kalangan pengguna Muslim terhadap status halal produk import telah mendorong keperluan untuk kaedah pengesahan ramuan yang lebih cekap. Projek ini telah memperkenalkan *HalalScan*, satu sistem berasaskan web yang membolehkan pengguna mengenalpasti status halal sesuatu produk makanan pada tahap kandungan nya dengan mengimbas atau memuat naik imej senarai ramuan produk. Sistem ini menggunakan teknologi *Optical Character Recognition (OCR)* yang dipacu oleh *Tesseract OCR* untuk mengestrak teks daripada imej dan menyemaknya dengan pangkalan data ramuan secara dalaman. Pengguna kemudian diberikan penunjuk status berwarna (hijau, kuning, merah) yang mewakili ramuan halal, syubhah atau haram. Sistem ini turut menyediakan antara muka mesra pengguna, paparan maklumat ramuan secara terperinci, halaman statistik, dan panel untuk pengurusan data. Ujian kebolehgunaan yang dijalankan Bersama pelbagai Kumpulan pengguna menunjukkan system ini umumnya mudah digunakan dan berkesan. Namun begitu, beberapa had telah dikenalpasti berkaitan ketepatan OCR dalam keadaan tertentu. Walaupun begitu, *HalalScan* membuktikan pendekatan yang menjanjikan dalam usaha mempermudah dan mendigitalkan proses semakan status halal ramuan produk.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1
1.1 Introduction 1
1.2 Problem Statement 2
1.3 Scope 3
1.4 Aims and Objectives 3
1.5 Methodology 4
1.6 Significance of The Project 5
1.7 Project Schedule 6
1.8 Expected Outcome 6
1.9 Conclusion 7
CHAPTER 2: LITERATURE REVIEW 8
2.1 Introduction 8
2.2 Literature Review 8
2.2.1 JAKIM Website 9
2.2.2 Verify Halal 10
2.2.3 Warees Halal Network 11
2.2.4 Comparative Analysis 12
2.3 Review of Tools and Related Technologies 15
2.4 Proposed System 16
2.5 Summary 17
CHAPTER 3: REQUIREMENT ANALYSIS & SYSTEM DESIGN 18
3.1 Introduction 18
3.2 User Requirement Analysis 18
3.3 System Requirement Specification 19
3.3.1 Functional Requirement 20
3.3.2 Non-Functional Requirement 21
3.4 Software Requirement Specification 23
3.5 Hardware Requirement Specification 23
3.6 System Design 24
3.6.1 Data Flow Diagrams 24

3.6.2 Database Design 28
3.7 User Interface Design 31
3.8 Summary 35
CHAPTER 4: IMPLEMENTATION AND TESTING 36
4.1 Introduction 36
4.2 Installation Instructions 36
4.2.1 Project Dependencies 37
4.3 File Structure 38
4.4 Implementation of HalalScan 39
4.4.1 Homepage 39
4.4.2 Camera Scan Page 40
4.4.3 Upload Image 42
4.4.4 Result Page 43
4.4.5 Ingredient Detail Popup 46
4.4.6 Statistic Section 46
4.5 Database Implementation 48
4.5.1 Database Structure 48
4.5.2 Admin Panel for Data Management 49
4.6 Software Testing 49
4.6.1 Functionality Testing 50
4.6.2 Usability Testing 54
4.7 Summary 56
CHAPTER 5: CONCLUSION AND FUTURE WORKS 58
5.1 Introduction 58
5.2 Achievements 58
5.3 Limitations 58
5.4 Future Work 59
5.5 Summary 60
References 61
Appendices 63

LIST OF FIGURES

Figure 1. Agile methodology diagram 4
Figure 2. Schedule for Final Year Project 6
Figure 3. JAKIM official website Halal Status Checker Interface 9
Figure 4. Verify Halal's product searched result 10
Figure 5. Warees Halal Network ingredient checker 11
Figure 6. Status of ingredient for imported food 12
Figure 7. HalalScan's system architecture 19
Figure 8. HalalScan's context diagram 24
Figure 9. Data Flow Diagram Level 0 25
Figure 10. Data Flow Diagram Level 1 for Login process 25
Figure 11. Data Flow Diagram Level 1 for Choose Report process 26
Figure 12. Data Flow Diagram Level 1 for Scan Ingredient process 27
Figure 13. Data Flow Diagram Level 1 for Manage Ingredient List process 27
Figure 14. HalalScan's entity relationship diagram 28
Figure 15. User interface for HalalScan's homepage 31
Figure 16. Camera interface for scanning ingredients 32
Figure 17. Result page 33
Figure 18. Information page for the ingredient 34
Figure 19. HalalScan's homepage 39
Figure 20. HalalScan's camera scan page 40
Figure 21. HalalScan's upload image section 42
Figure 22. HalalScan's result section 45
Figure 23. Ingredient's detail popup 46
Figure 24. HalalScan's statistic section 46
Figure 25. Admin panel for database 49
Figure 26. OCR result displaying "Malic Acid" extracted from the image but not clickable. 53
Figure 27. Admin panel confirms "Malic Acid" exists in the ingredient database. 53
Figure 28. OCR misrecognition of "wheat", "salt", "corn", "yeast extract" and "fructose" 54
Figure 29. Respondent age ranges 55
Figure 30. Scenario-based task performance on Question 1 56

Figure 31. OCR failure due to shape of the packaging 58
Figure 32. OCR failure due to reflective packaging 58

LIST OF TABLES

Table 1 shows the comparative analysis of existing websites and proposed system 12
Table 2 shows the HalalScan's functional requirements. 20
Table 3 shows the non-functional requirements of HalalScan. 21
Table 4 outlines the software specification 23
Table 5 outlines the hardware requirements for user 23
Table 6 outlines the hardware requirement for admin 23
Table 7 shows the data dictionary for admin 29
Table 8 shows the data dictionary for ingredients 29
Table 9 shows the data dictionary for scans. 30
Table 10 outlines the libraries used in the development 37
Table 11 displays the test case for homepage 50
Table 12 displays the test case for scan function 50
Table 13 shows the test case for upload function 51
Table 14 shows the test case for admin 52

CHAPTER 1: INTRODUCTION

1.1 Introduction

As consumers around the world become more conscious of and exposed to halal standards, there is a growing demand for halal-certified products across the globe according to Aniqoh and Hanastiana (2020). On the other hand, it can be difficult to guarantee that imported products fulfil halal standards, particularly for customers who depend on precise and easily accessible ingredients verification, not only based on the halal logo on the product itself or scanning barcode to know the halal status. Ismail, Hayub, Yahaya., Jamal, Abd Wakil, Rifin, and Nasri (2024) stated that halal logo can be counterfeited, and it is hard to detect as it occurs at the final stages of the halal supply chain. The wide variety of products on the shelf makes it difficult for many consumers to verify the ingredients for each of the products. This often requires consulting several sources, which causes ambiguity and inconvenience for customers who are concerned about halal (Petiwala, Nawazish, Shukla, Sharma, & Nanda, 2021).

Rahmawati, Purnamasari, Sofiana, Setiawan, Alfita, and Nahari (2023) suggest that it is crucial for customers to have quick and easy access to accurate information as people want to know exactly what's in their food. However, Sukarno, Ramadhita, and Rachmawati (2019) highlight the challenge of rapidly distributing food items while maintaining halal standards through automated halal ingredient detection. In Malaysia, Jabatan Kemajuan Islam Malaysia (JAKIM), the only official halal certification authority, offers a halal directory for checking halal products. Nevertheless, the website is not user-friendly, does not provide an adequate platform to report misconduct and requires improvement. Solutions were proposed by Ying and Zaaba (2022) include creating mobile apps to improve search capabilities, incorporating barcode scanning features, and making the JAKIM halal directory more user friendly for both local and global products.

Petiwala et al. (2021) claims that Halal apps empower customers to make informed choices by providing easy access to verified halal products. Hence, HalalScan will provide a simplified method that scans and confirms the halal status of the imported products based on their ingredients. Users will be able to scan the ingredients listed on the product's packaging with the scanner tool, which will instantly compare them to a verified database of permitted, prohibited, uncertain ingredients. HalalScan will be seamlessly integrated into JAKIM's current website to guarantee

dependability and a smooth user experience. The integration will eliminate the need for users to switch between platforms by providing them with access to all halal-related information in one location.

In short, HalalScan greatly increases customer convenience and trust by streamlining halal verification for imported products. By giving users simple access to trustworthy ingredient information within JAKIM's reputable platform, it enables them to make well-informed decisions fast and precisely. In addition to helping halal-conscious customers, this solution upholds industry norms, encourages transparency, and aids in the global growth of the halal sector.

1.2 Problem Statement

In Malaysia, changing lifestyles and a growing demand for food diversity have increased the consumption of imported food products. However, limited academic research has been conducted on this sector, particularly on consumers' purchasing behaviour and halal verification challenges, despite the rising global demand for halal-certified products (Choi & Jeong, 2019). There's still a higher likelihood for dishonest individual to manipulate the halal logo and status for their own gain (Asa, 2019). The complexity of the ingredients too has been one of the factors in increasing difficulty in identifying halal products (Ismail et al., 2024). The challenges faced by consumers in verifying halal products are compounded by several specific issues that hinder their ability to make informed choices.

Consumers will receive incorrect information if halal application is launched with incomplete database or outdated database. This cause consumers to be misled whether the product is halal or not, thus making them lose trust and unable them to make decision (Wan Ismail et al., 2024). Furthermore, outdated database can enable food crime where food fraud and food terrorism can occur, allowing any ingredients to be falsely identified which leads to detrimental to consumers' confidence and health (Ariffin, Riza, Hamid, Awae, & Nasir, 2021)

According to Tieman and Williams (2019), even though there are roughly 500 halal certifying organizations in the world, no single international organization keeps a single database that compiles the information from these organizations. Users are forced to use several platforms in order to obtain trustworthy information, and this restriction makes the verification process difficult and time-consuming (Razak, Lee, Lim, & Tee, 2019). At the moment, the decentralized halal certification makes it extremely difficult to determine whether a product is halal. Without proper context, data related to product information rarely includes anything resembling a list of ingredients and details associated with those products which make extraction and integration into other datasets difficult. A centralized and structured database could solve this problem by providing a single window to view related information from the product on one website which will make things smooth and user friendly (Rakhmawati, Fatawi, Najib, & Firmansyah, 2019)

1.3 Scope

HalalScan focuses exclusively on verifying the halal status of food products based on their ingredients, utilizing a database that classifies ingredients as permitted, prohibited, or dubious and will not extend to non-food items. Additionally, it will focus on verifying ingredients using a predefined database rather than in-depth product certification. For the target users, HalalScan is intended for Muslims for ensuring the products purchased is halal compliance. Furthermore, it will be incorporated into JAKIM's current website to improve functionality and give users easy access to halal verification in addition to current halal certification data, simplifying the verification process and eliminating the need to switch between platforms.

1.4 Aims and Objectives

The goal of HalalScan is to establish a user-friendly tool that enables users to easily verify the halal status of imported products by scanning their ingredients lists. HalalScan aims to improve user confidence in halal compliance by streamlining halal verification procedure by incorporating through integrating this tool into JAKIM’s current platform.

The objectives of HalalScan:

1. To develop HalalScan application that verifies the halal status of imported products based on their ingredients.
2. To integrate the HalalScan application with JAKIM's existing website that streamline with halal verification for users.
3. To generate report based on weekly or monthly based on user search for category of the product.

1.5 Methodology

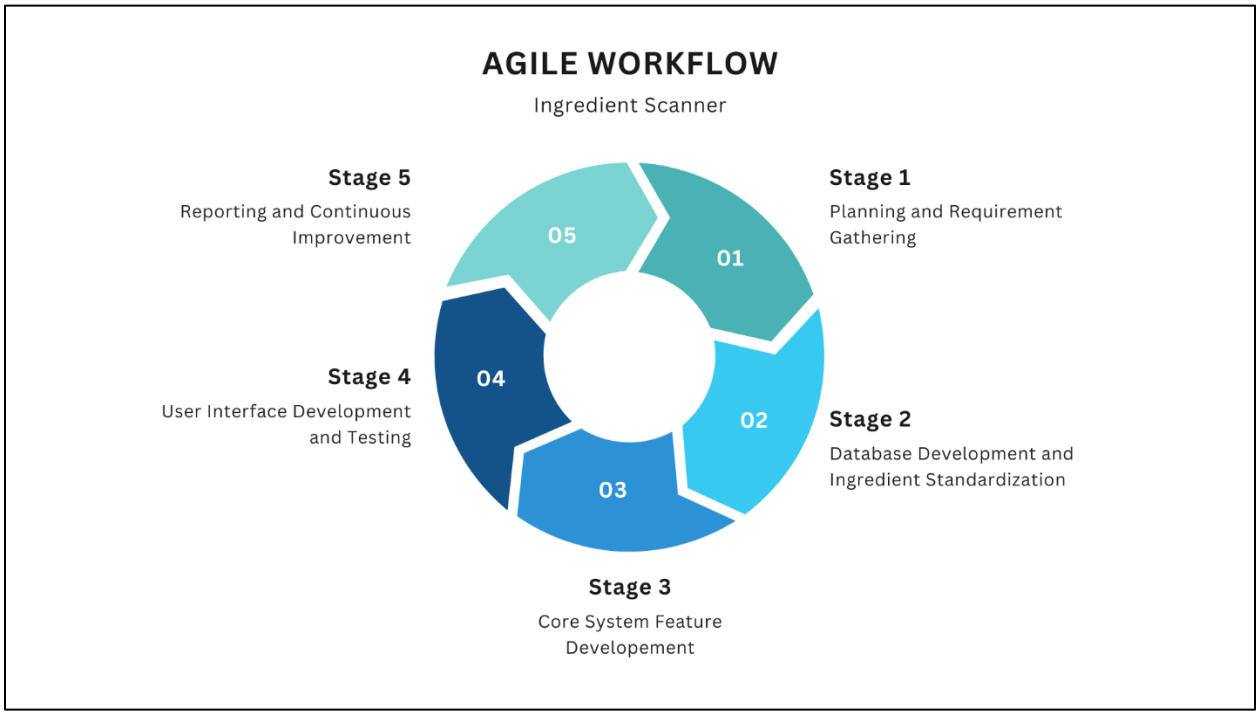

Figure 1. Agile methodology diagram

HalalScan will be developed using an Agile methodology, allowing for iterative development and continuous refinement based on user feedback. Figure 1 shows that the development process will be structured into multiple sprints, each focusing on specific system components to ensure flexibility and adaptability.

Initially, the system will begin with data collection and categorization, where the ingredient data will be classified as permitted, prohibited, or dubious groups to build a standardized database. Core system features, including OCR-based verification and data analytics will be developed incrementally, ensuring that each function is tested and improved throughout the process.

Furthermore, a simple, intuitive user interface will be designed to ensure seamless interaction, with iterative testing to validate system performance, usability, and accuracy. Additionally, a reporting features will be integrated to track user search patterns and generate insights. This Agile approach ensures that HalalScan remains adaptable, improving over time through user feedback and data-driven enhancement.

1.6 Significance of The Project

HalalScan empowers users to make informed purchase decisions by instantly verify ingredients. The system uses OCR technology to extract text from scanned ingredient labels, which is then cross-checked against a structured halal ingredient database. By providing real-time verification, HalalScan ensures that users receive accurate and reliable information about product content in fast and convenient manner.

Furthermore, HalalScan integrates Optical Character Recognition (OCR) to enhance user interaction and accessibility. OCR converts text from ingredient images into a machine-readable format, simplifying the verification process for users. This streamlined approach ensures a user-friendly experience while maintaining accurate and efficient halal verification.

Additionally, HalalScan's analytics feature will improve user experience and awareness of halal ingredients. The system can give users helpful details by identifying commonly scanned ingredients that are flagged as prohibited or dubious. This data-driven approach will make them more conscious of common issues with particular ingredients in addition to assisting users in making better decisions about the product(s).

1.7 Project Schedule

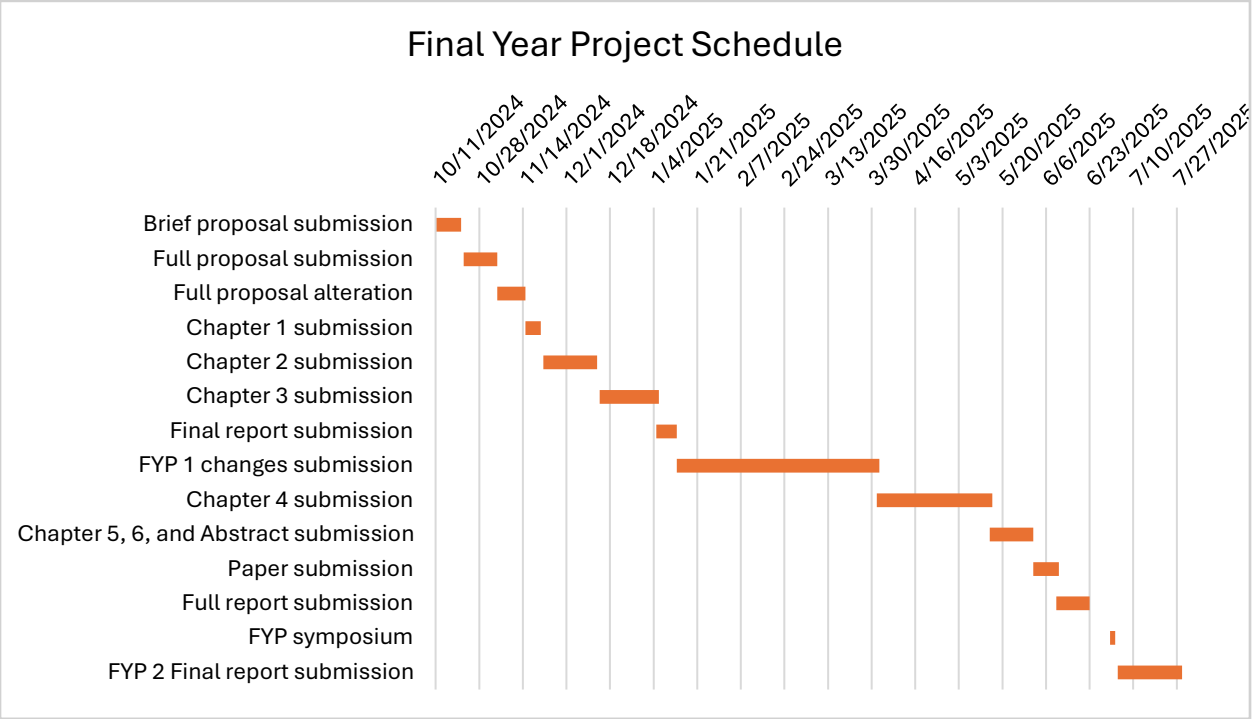

Figure 2. Schedule for Final Year Project

1.8 Expected Outcome

This project aims to implement a fully functional HalalScan scanner tool that will be featured with JAKIM's current website that will be able to scan or upload ingredient image and verify the ingredients according to the halal status in the database, ensuring quick and accurate validation.

Besides that, a reliable ingredient verification system will be developed to process user scans and identify ingredients that are permitted, prohibited, or dubious. The system expected to achieve an accuracy rate of at least 70% by the project completion.

Additionally, data analytics reporting tool will be introduced to monitors and display user search patterns and ingredients that are frequently scanned, giving the user insights into which ingredients are frequently flagged as dubious or prohibited.

1.9 Conclusion

HalalScan aims to simplify the halal verification process for users, particularly for imported food products. This solution made is for cross-check ingredients with the halal status of the product. HalalScan seeks to make the halal process verification easier by incorporating innovative feature like Optical Character Recognition (OCR), content-based recommendations system, and data analytics. A smooth and reliable user experience is ensured by the project's primary focus on integrating with JAKIM's current platform, and ongoing improvement through user feedback is ensured by the agile development methodology. Besides helping users make better buying decisions, HalalScan also supports the global halal industry by promoting transparency and trust. It does this through accurate ingredient classification, a user-friendly experience, and clear, insightful data reporting.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Traditionally, identifying the halal logo on product packaging has been the way to recognize the halal status of the product (Tarannum, Jalal, & Huda, 2024). However, there's a challenge in verifying halal status address by Ismail et al. (2024) related to the growing problem of halal logo counterfeiting, which frequently occurs at the end of the halal supply chain. Verification at the ingredient level, however, offers a more thorough method of guaranteeing conformity, ensuring authenticity and maintaining customer confidence particularly for products without halal markings (Noor, Azmi, Hanifah, & Zamarudin, 2023).

Therefore, this chapter will review similar websites related to halal ingredient verification by identifying their strengths, weaknesses and suitability for modern user requirement. A comparative analysis between a similar website and the proposed system will be conducted with selected features chosen based on the conclusion gathered from research and observation. The rest of this chapter consists of five sections: Section 2.2 reviews similar websites, Section 2.3 reviews the tools and technologies selected for the system, and Section 2.4 briefly explains the proposed system and its improvement compared to the existing websites.

2.2 Literature Review

This section reviews three existing platforms related to halal ingredients verification: JAKIM's official website, Verify Halal, and Warees Halal Network—focusing on their features, strengths and weaknesses. Each system is studied for its core functionality, usability, database coverage and any suitable features that suits with the project's objective. These evaluations seek to identify HalalScan's system advantages, disadvantages, and any potential areas for development. Additionally, gaps in the current offering will be analysed in these solutions, which makes it possible to develop a more user-centric and efficient approach.

2.2.1 JAKIM Website

JAKIM website acts as a centralized hub for halal certification and status. It offers a searchable directory of businesses ad products with halal certification, making verification straight forward. While the database contains information that has been validated by JAKIM, users may search for halal status of the product by entering the product or business name, browse through different categories products and services and scan the halal logo (using the Halal Malaysia app).

The system is occupied with an extensive database of halal-certified products including those from international manufacturers (Ahmad, Widayatmoko, & Taib, 2024). Additionally, the website features a basic yet effective search function, providing users with straightforward search result.

However, the system lacks modern interactive tools for enhanced user convenience, such as ingredient scanning capabilities. Additionally, as seen in Figure 3, the product names are searched manually on the website, which makes navigating difficult without advanced search features. Furthermore, there is no integrated mechanism for halal verification at the ingredient level to confirm specific raw materials (Hashim, Salim, Noah, & Mustapha, 2023)

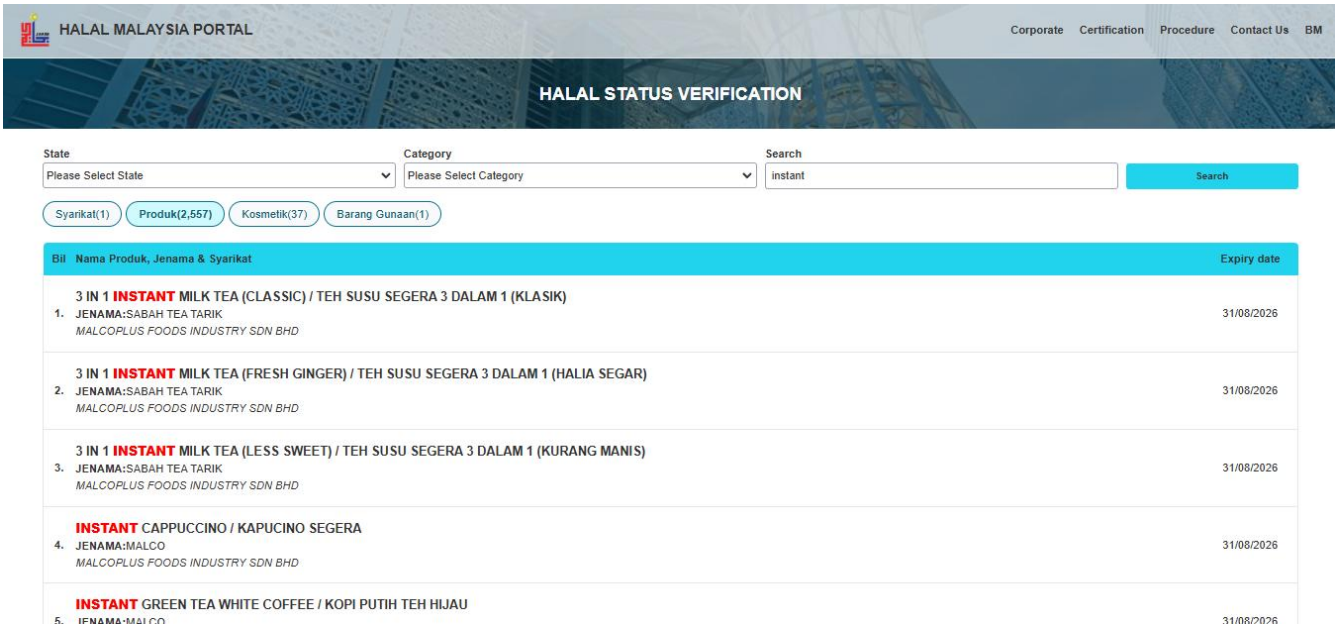

Figure 3. JAKIM official website Halal Status Checker Interface

2.2.2 Verify Halal

Verify Halal is a halal verification app that lets users check the halal status of products, brands, locations, and slaughterhouses across the world. JAKIM and globally renowned halal certification organizations fuel the system, which provides a centralized resource for customers who are concerned about halal. As seen in Figure 4, the app's user-friendly and adaptable capabilities include barcode scanning, search by product, brand, or company, and feedback posting (Serunai Commerce Sdn Bhd, 2023).

The system integrates data from internationally renowned halal certification authorities, including JAKIM, to ensure the credibility and reliability. Additionally, it offers multiple search options, allowing users to look up for product, brand and company for user convenience. Additionally, it contains halal goods, facilities, and slaughterhouses that have been approved by international organizations, making it perfect for usage anywhere in the world.

However, barcode scanning is only available in the mobile application, not on the website limiting accessibility for websites users. The system verification is restricted to certified or registered brands, meaning uncertified or unregistered products cannot be verified. Moreover, neither website nor mobile apps provide detailed product ingredient verification, preventing users from checking the halal status of individual ingredients.

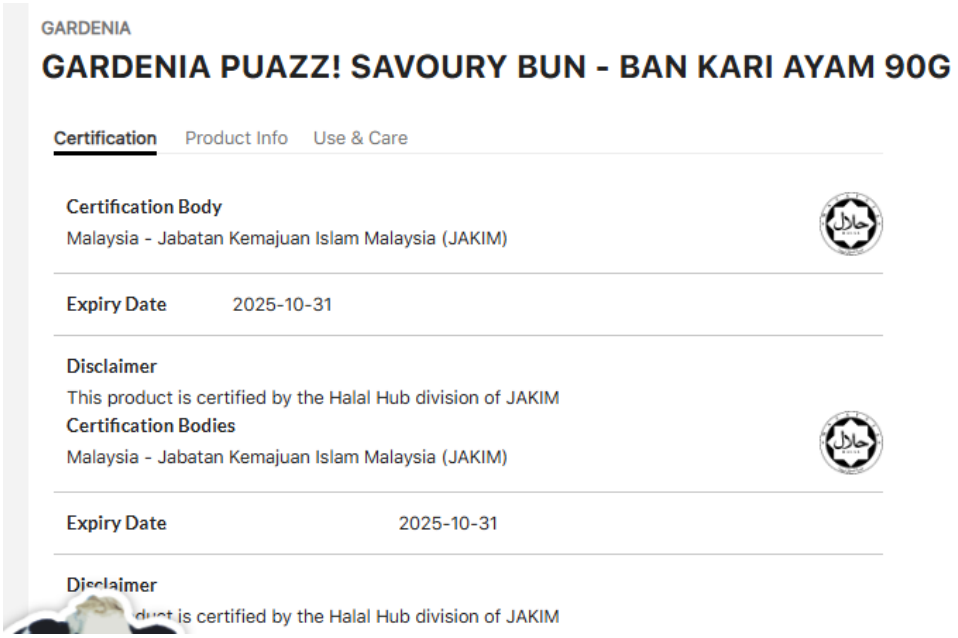

Figure 4. Verify Halal's product searched result

2.2.3 Warees Halal Network

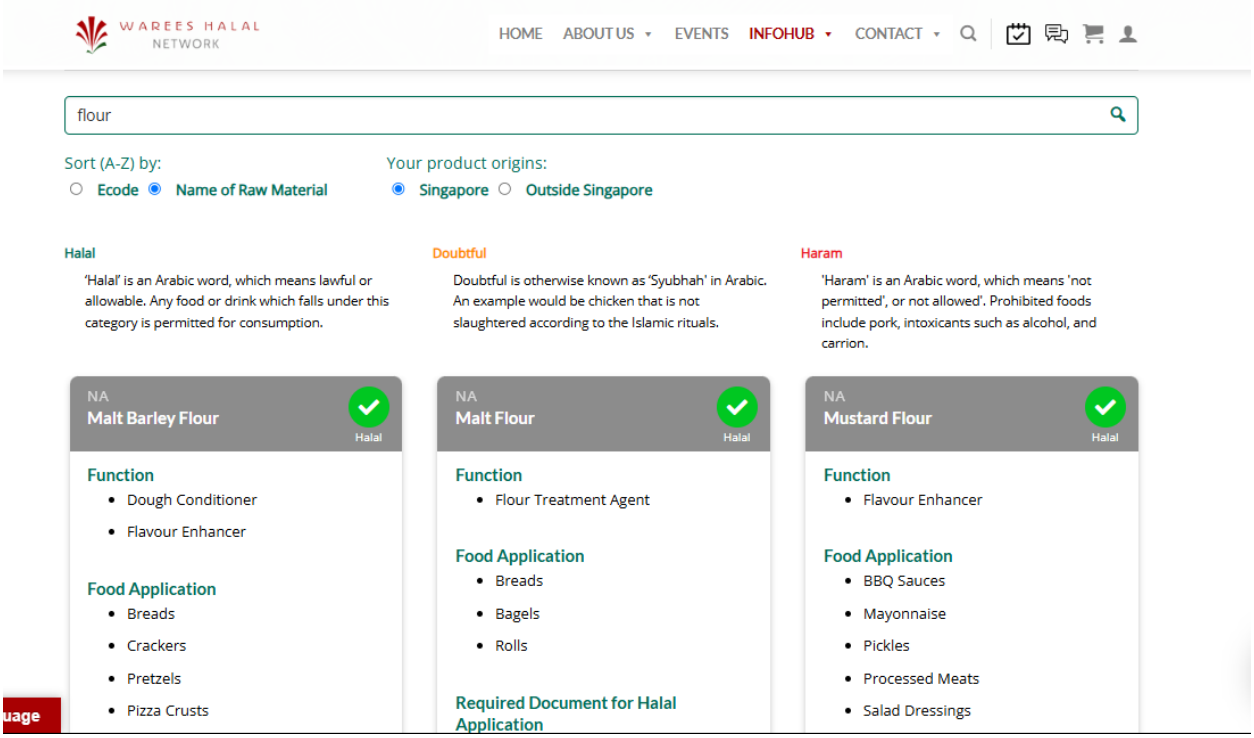

Figure 5. Warees Halal Network ingredient checker

Warees Halal network is a platform based in Singapore was created to help users determine a certain food ingredient is halal or not. It has an ingredient checker that lets users look for ingredients by name or E-code to make sure they're halal. The tools' main goal is to inform users about dubious or illegal ingredients used in food manufacturing (Ingredient Checker - Warees Halal Network, 2021).

One of the key strengths of the system is its unique feature that focuses on food additive and offers precise instructions on their halal status as presented in Figure 5. It offers details on E-codes and how to use them, bringing attention to substances that may be dubious. This specialized approach makes the website particularly useful for users who are worried about certain chemicals because it focusses exclusively on substances rather than goods.

However, this system lacks verification for whole products and as it is solely focused on specific ingredients and E-codes. Users are also unable to scan the packaging for rapid verification, which may limit convenience. Moreover, the database only includes a small number of chemicals, excluding newer additions or foreign versions as demonstrated in Figure 6.

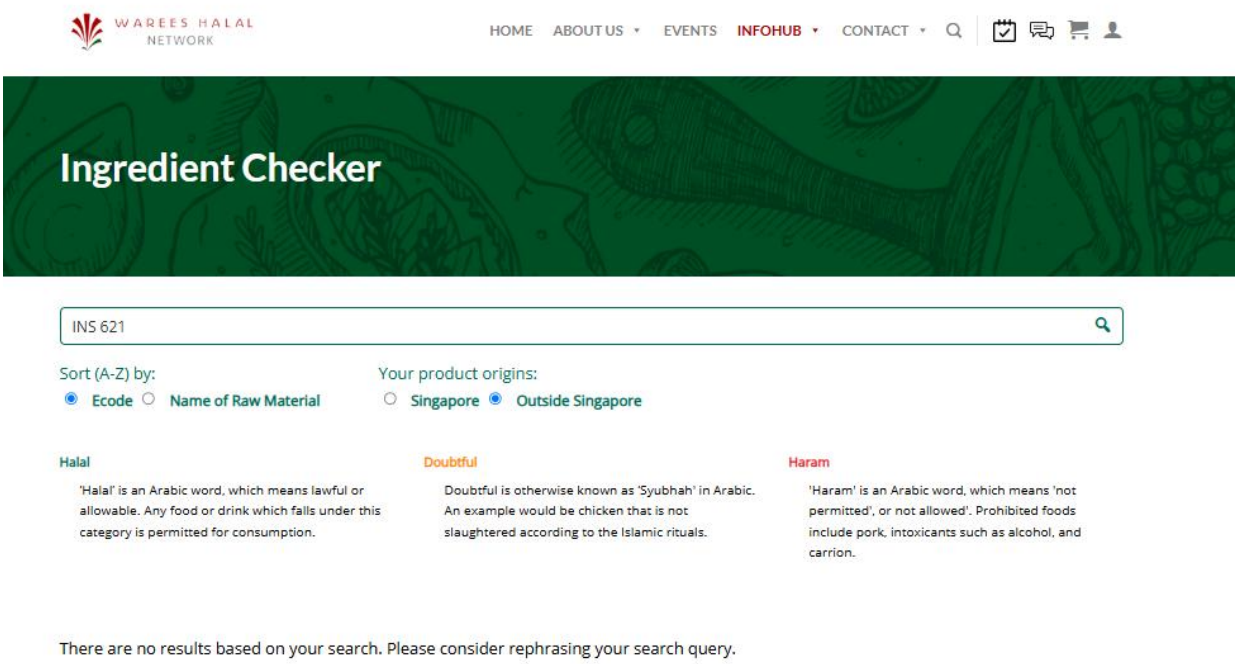

Figure 6. Status of ingredient for imported food

2.2.4 Comparative Analysis

Table 1 shows the comparative analysis of existing websites and proposed system

Application / Feature	**JAKIM**	**Verify Halal**	**Warees Halal Network**	**HalalScan (Proposed System)**
Search functionality	Search product's name.	Website: Search product's name. Mobile apps: Barcode, QR and text scanner	Ingredient Checker with E-code and ingredient name search.	Search products by image-based ingredient list.
Ingredient verification	Not available. Focuses on certified products listed in its directory.	Not available. Focuses on product and certificate verification	Includes an "Ingredient Checker" tool for ingredient status but lacks	Verifies individual status (halal, prohibited, dubious), and

		through database search.	detailed user feedback and interaction.	recommended alternatives.
Database Coverage	Comprehensive for local and international certifications verified by JAKIM	Extensive, covering multiple international certifications.	Limited primarily to their database (based in Singapore)	Comprehensive, regularly updated, and integrates global halal data sources, including JAKIM's certifications and ingredient lists.
User Interface	Basic and straightforward interface	The mobile app is user-friendly, but the website is simple.	Simple and straight forward ingredient checker.	User-friendly interface with advanced features like product recommendations and dynamic explanations.
Reporting Tools	Not available	Not available	Not available	Generates analytics reports

2.2.4.1 Search functionality

JAKIM's website offers a straightforward function where the users just enter its name into the search option to confirm a product's certification. This method is straightforward and appropriate for users who already have previous experience with the system, but it can be cumbersome for user who are not familiar with the specifics of the product or who are exploring big databases. This website lacks modern search technologies that results in limitations in detailed and rapid verification. Next, Verify Halal apps version has significantly enhanced user convenience with the

advanced search options such as barcode scanning, QR code scanning, and text search. However, the same features are not applicable to the website version, as it only provides basic product name search. On the other hand, Warees Halal Network, prioritizes ingredient verification over product-level checks. This site works with specific ingredients as it uses ingredients name or E-code to check the halal status. However, this site does not provide the ingredients verification for a whole product. Although it meets a need for ingredient verification, Warees Halal Network does not offer as many alternatives for product-based searches.

2.2.4.2 Ingredient verification

JAKIM's website and Verify Halal both do not verify individual ingredients as this website only focuses solely on products certified in its directory, by searching for the products name or brand name. The result of the searched products did not disclose the full ingredients, and this limitation creates a barrier for more detailed halal verification. The Wares Halal Network on the other hand has provides users with specific ingredient checker and verifies that the ingredients are halal or not. However, this website only checked specific ingredients searched by the users, not ingredients of a product and tagged each ingredient status. While the ingredient checker tool itself is useful, it can be troublesome for the users to search each ingredient in a product instead of searching the product's name and all the ingredient status will be displayed.

2.2.4.3 Database Coverage

JAKIM's website and Verify Halal both possesses local and international certification making it extremely dependable for Malaysian users looking to confirm the halal status of products. Users can rely on the platform to deliver accurate and official information on certified items both inside and outside of Malaysia thanks to its wide scope. Additionally, Verify Halal expands its user base to international users and makes it easier for them to use the system. Nonetheless, while JAKIM and Verify Halal are strong in database coverage, both systems lack depth in ingredient verification. Warees Halal Network, on the other hand, provides a more limited database with a smaller range of ingredients and a primary focus on certifications from Singapore. Users that can benefited from it are mostly Singaporean, but its limited reach makes it less thorough and inclusive than JAKIM and Verify Halal.

2.2.4.4 User Interface

JAKIM's website has a basic and straightforward interface which features an easy-to-use layout, but it is less appealing due to its lack of contemporary usability features and interaction. Conversely, Verify Halal mobile app provides a more user-friendly interface with modern technology such as scanning capabilities and easy navigation, allowing customers to conveniently check products on the move. In contrast, the website of Verify Halal is lacking in comparison to the mobile apps features due to its simpler, less engaging design. Furthermore, Warees Halal Network too has a clean layout that is centred on ingredients searches, which accomplishes its goal well.

2.2.4.5 Reporting Tools

The reviewed halal verification system –JAKIM, Verify Halal, and Warees Halal Network are lacking in reporting capabilities that may offer analytics or user insights. This limitation hinders users from gaining more knowledge about frequent problems or trends related to ingredient status in a product.

2.3 Review of Tools and Related Technologies

2.3.1 HTML5 Media API for Front End Camera Access

HTML5 Media API is used as text scanner from user's device. This API enables users to capture pictures directly from their device and seamlessly integrates the device's camera into the web application. It is an approachable and user-friendly option to integrate this feature due to its compatibility with modern web browsers.

2.3.2 Tesseract OCR

Tesseract OCR's main objective is to identify characters in an image (Adjetey & Adu-Manu, 2021). The text from the picture captured by the user will be extracted in order to analyze the captured ingredient list. This process is essential as the product ingredients need to be correctly identified before classifying them halal, non-halal, or dubious.

2.3.3 Halal Ingredient Database

The system integrates a manually curated Halal Ingredient database to verify the halal status of each ingredient. Once the OCR extracts the text, each ingredient is checked against this database, which is built from reliable halal sources. This approach ensures accurate verification by leveraging an up-to-date database, while allowing for flexibility in data management and updates.

2.3.4 Flask

The system's backend operation will be managed by Flask, a Python framework. Flask will handle the processing of user inputs (ingredient lists), integrate the OCR tool and execute the categorization logic to determine each status of the ingredient lists.

2.3.5 SQLite

The extracted ingredient data and its associate statuses will be stored in SQLite. Storing data in SQLite ensures the safety and effectiveness of data storage for the proposed system, enabling users to review any input and produce detailed reports for analytical purposes.

2.4 Proposed System

HalalScan aims to reinvent the halal verification process, in ingredients-level to cross check with the product's halal status. It works by properly identifies ingredients lists extracted from images through Optical Character Recognition (OCR) technology. Unlike existing solutions, this system will use both real time camera scanning and upload image functionality, where the users can search for ingredient status of a product by scanning image-based ingredients lists instead of manually searching by the product's name or by the ingredient. A Halal Ingredient database will be utilised to verify ingredient status against a reliable halal database, ensuring up-to-date and accurate classification. Furthermore, this system will be equipped with data analytics tools to produce informative reports that illustrate trends in product searches and ingredient compliance for informed decision-making. Additionally, the interface is designed to be dynamic, user-friendly, and suitable for users of all age groups.

2.5 Summary

The chapter has reviewed the limitations and opportunities addressed by the proposed project by examining current systems, tools, and techniques related to halal ingredients. It emphasized how accuracy and effectiveness of existing system are jeopardized by relying on manual product searches, decentralized database and outdated interface. In order to enhance the user experience and decision making, several technologies will be used in the proposed system to fill in these gaps of the existing system including HTML5 Media API to scan ingredient list of a product, Tesseract OCR for text extraction from image, the halal ingredient database for real-time halal status verification, Flask for backend development and SQLite for data storage. These findings form the foundation for the proposed system, which uses modern tools and technologies to solve the issues detected and underline the need for a more dynamic, centralized, and user-friendly solution.

CHAPTER 3: REQUIREMENT ANALYSIS & SYSTEM DESIGN

3.1 Introduction

The project HalalScan aims to verify the halal status of the product up to ingredient level. This is essential to ensure that the products meet the dietary requirements of Muslim consumers. The system uses a multi-step methodology to achieve real-time halal verification. Initially, users scan the product's ingredient list or upload an ingredient list image, which is processed by Optical Character Recognition (OCR) technology, such as Tesseract OCR, to extract the text. The text is cleaned and formatted for further analysis. Once the ingredients are extracted, their halal status is verified and will then be verified through a halal ingredient database, with each ingredient checked individually. Additionally, the system also provides visual representation of the most searched ingredients and trends from scanned product. Users can view this data weekly or monthly through simple graph, offering insight into common ingredients and trends. Furthermore, the halal status and ingredient list are stored in an SQLite database for future reference. This methodology allows for efficient, real-time verification, providing users with reliable information for informed purchasing decisions.

3.2 User Requirement Analysis

The development of HalalScan necessitates a thorough understanding of user needs and expectations to ensure the system effectively addresses its intended purpose of halal ingredient verification. Microsoft have highlighted the importance of quick and efficient features. For instance, Rosyadi, et al. (2024) found that quick text recognition significantly enhances the overall user experience by simplifying tasks and reducing the time spent on interactions. Additionally, Ying and Zaaba (2022) revealed that users show a low interest in manually searching for items, suggesting an expectation for more convenient and automated search features within applications. This highlights the importance of designing intuitive search functionalities for the HalalScan system.

The usability and interface design of an application play a pivotal role in ensuring user engagement and satisfaction (Priyadarshini, 2024). Kabir, Kabir, and Islam, (2024) indicates that applications with simpler interfaces and fewer, yet well-curated, features are more successful in establishing meaningful connections with users, as supported by the technology as experience

framework. This insight underscores the need for HalalScan to prioritize a streamlined and user-centric design. However, the same article also reveals a gap in current applications, where the absence of analytical features and constructive feedback mechanisms hinders users from deriving meaningful insights. For HalalScan, addressing this gap by integrating feedback features and offering relevant analysis could significantly enhance user engagement and support informed decision-making.

3.3 System Requirement Specification

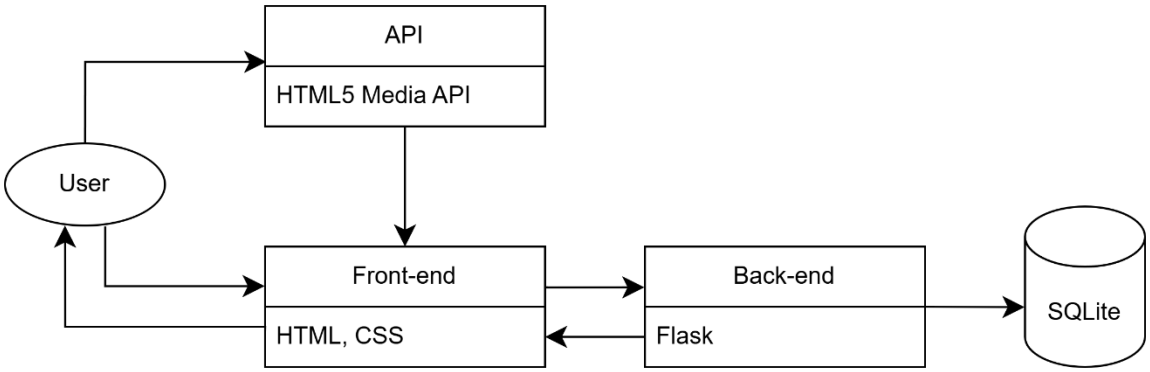

Figure 7. HalalScan's system architecture

For the HalalScan system, the integration begins with users scanning the product's ingredients using their device's camera or uploading an image of the ingredient list. The scanned image will be sent to the frontend (HTML), which will forward it to the backend (Flask) for processing. The backend will then utilize OCR technology to extract the text from the image and clean the data by separating and formatting the ingredients for further processing as presented in Figure 7.

Once the ingredients are extracted, the backend will send each ingredient to the database to verify whether it is halal or non-halal. The database will respond with the halal status for each ingredient, and the backend will aggregate the results to determine the overall halal status of the product. If all ingredients are halal, the product will be marked as halal; if any ingredient is non-halal, the product will be marked as non-halal.

The backend will send the final halal status, along with the list of ingredients, back to the frontend, where the interface will display the results to the user. Additionally, the backend will store the scanned product information, including the ingredients and halal status, in the SQLite

database for future reference, enabling the system to track products and provide historical data for analysis.

3.3.1 Functional Requirement

Table 2 shows the HalalScan's functional requirements.

Requirement	**Explanation**
Ingredient Scanning	The system shall allow users to either scan the product's ingredient list using their device's camera or upload an image from their device. The captured or uploaded text will be processed and extracted for verification.
Ingredient Verification	The system shall verify each of the ingredients and classify their halal status based on the pre-existing database.
Database Management	The system shall maintain an up-to-date ingredient database and their halal status. Admin is allowed to add new ingredient lists and update the halal status of the existing ingredients.
Report Generation	The system will track the scan data over time and store statistic that would allow users to aggregate the information weekly or monthly based on product analytics.

3.3.2 Non-Functional Requirement

Table 3 shows the non-functional requirements of HalalScan.

Requirement	Explanation
Operational	The system supports the web platform and is available 24/7 for ingredient verification.
Performance	The system must be able to process ingredient verification within 10 seconds.
Scalability	The system must be able to handle the expanding ingredient database that will be updated constantly.
Reliability	The system must be able to perform its function as expected.
Security	The system has restricted database access to authorized personnel.
Usability	The system interface should be intuitive, focusing on ease of use for non-technical users.
Maintainability	The system must be able to facilitate updates and maintenance without disrupting the service.
Data backup	The system is equipped with backups of ingredient database.
Cultural and Political	The system ensures that the data align with halal standards recognized internationally and locally.

3.3.2.1 Integration

The integration will begin with users scanning a product's ingredients list using a scanner. The image of the product's ingredients will then be processed through Optical Character Recognition (OCR) technology, such as Tesseract OCR, to extract the text from the image. This text will be cleaned to remove unwanted characters and formatted to separate the individual ingredients for further processing. Once the ingredients are extracted, the system will check their halal status.

The halal status of each ingredient will be verified in the halal ingredient database. The system will send each ingredient as a query to the database and will receive a response indicating whether the ingredient is halal or non-halal. Based on the results, the system will aggregate the halal statuses of the ingredients. If all ingredients are halal, the product will be marked as halal. If any ingredient is non-halal, the product will be marked as non-halal. In cases where the halal status is uncertain or ambiguous, the system will flag the product accordingly.

The final halal status, along with the list of ingredients, will be displayed to users through users' interface. This information will be stored in an SQLite database for future reference, allowing the system to track scanned products and their halal status. This integration will enable the HalalScan system to provide users with real-time halal verification based on scanned ingredient lists.

3.4 Software Requirement Specification

Table 4 outlines the software specification

Component	Details
HTML5 Media API	A text scanner to enable text scanning from users' devices.
Tesseract OCR	Extract text from images (captured ingredient lists).
Flask	Python-based framework for backend operations, mainly on processing inputs and managing the OCR tool.
Halal Status database	Verifies the halal status of ingredients and indicate whether an ingredient is halal, non-halal, or dubious.
SQLite	Store ingredient data and associated statuses.

3.5 Hardware Requirement Specification

User's side:

Table 5 outlines the hardware requirements for user

Component	Details
User device	Scanning ingredient and interact with the application.
Camera on user's device	For ingredient scanning functionality.

Admin's side:

Table 6 outlines the hardware requirement for admin

Component	Details
Processor	Intel Core i5 or AMD Ryzen 5
RAM and SSD	Minimum 8GB with 512 GB SSD
Memory space	Minimum 10GB of free space

3.6 System Design

The system design outlines the architectural components of the system, including the context diagram and data flow diagrams, followed by the entity-relationship diagram (ERD), UML Diagram and data dictionary to detail the system's structure and data management.

3.6.1 Data Flow Diagrams

Context diagram

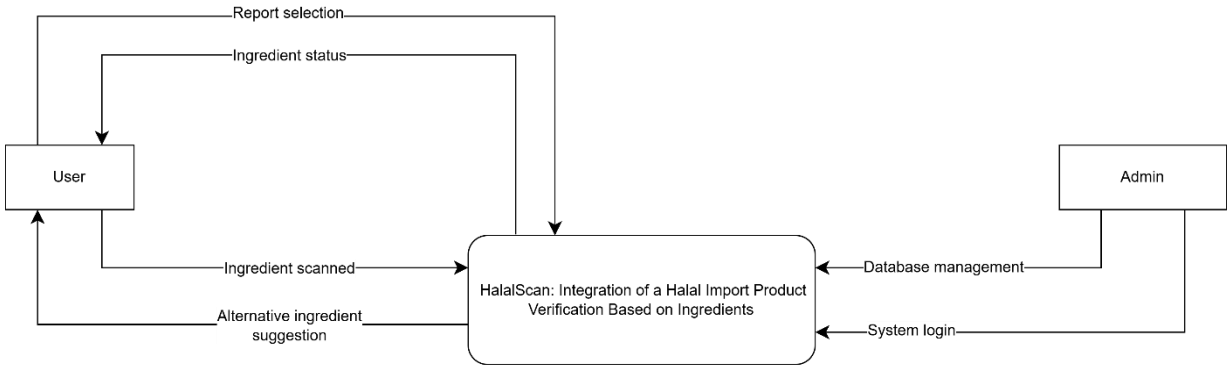

Figure 8. HalalScan's context diagram

According to Figure 8, the system interacts with two primary external entities: User and Admin. Users scan or upload product ingredients through the system, which then provides the ingredient status and suggests alternative ingredients if any issues are detected. Additionally, users are able to select reports (weekly or monthly) from the system, prompting the system to retrieve and display the most searched ingredients or other relevant data. On the other hand, admins interact with the system to manage the database, ensuring that ingredient data and other relevant information are up to date. Furthermore, admins access the system through a login process, allowing them to authenticate and perform administrative tasks.

a) DFD Level 0

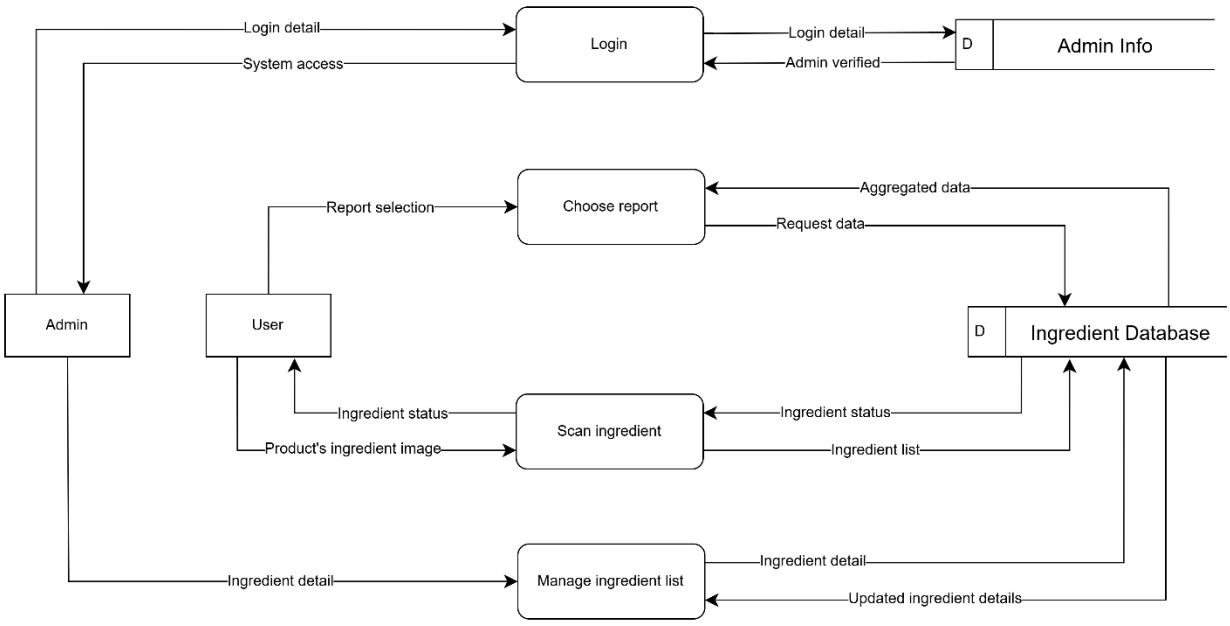

Figure 9. Data Flow Diagram Level 0

Figure 9 illustrates the overall data flow between external entities and the HalalScan system. It highlights the key processes, including login and manage ingredients list for the admin, as well as choosing report and scan ingredient for the user. These processes represent the main interactions with the system, allowing admins to manage data and users to check ingredient statuses and search for alternatives.

b) DFD Level 1—Login

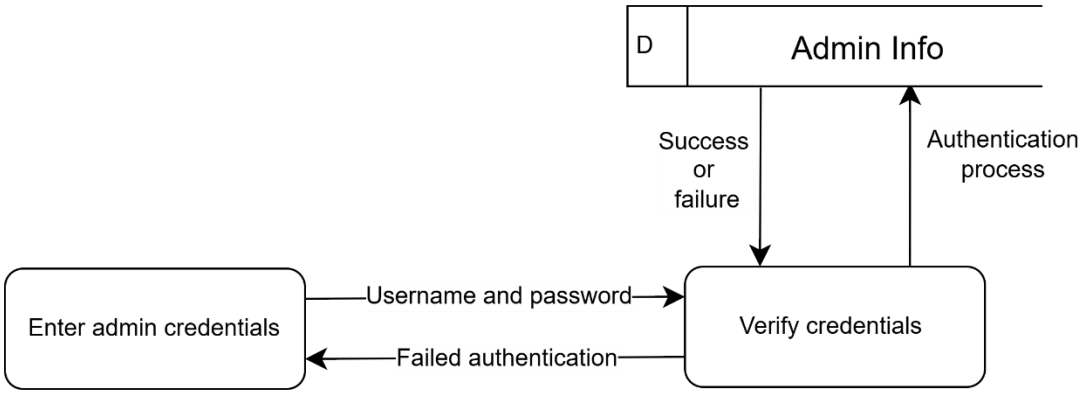

Figure 10. Data Flow Diagram Level 1 for Login process

The login process as depicted in Figure 10 for the admin involves the entry of admin credentials (username and password). These credentials are then verified against the data stored in the

database. If the credentials are correct, the system grants access to the admin interface, allowing the admin to perform further tasks. If the verification fails, an error message is returned, prompting the admin to re-enter their credentials.

c) DFD Level 1—Choose report

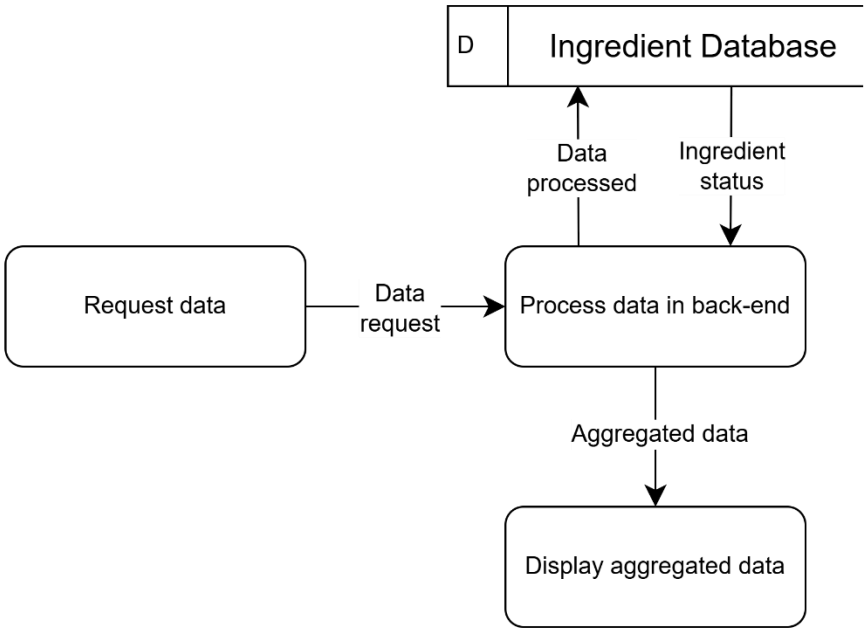

Figure 11. Data Flow Diagram Level 1 for Choose Report process

Users select a report (weekly or monthly) to view. The system then processes the data in the backend, retrieving relevant information from the ingredient database and aggregating it accordingly. Once processed, the system displays the aggregated data to users, showcasing the most searched ingredients or trends as illustrated in Figure 11.

d) DFD Level 1—Scan ingredient

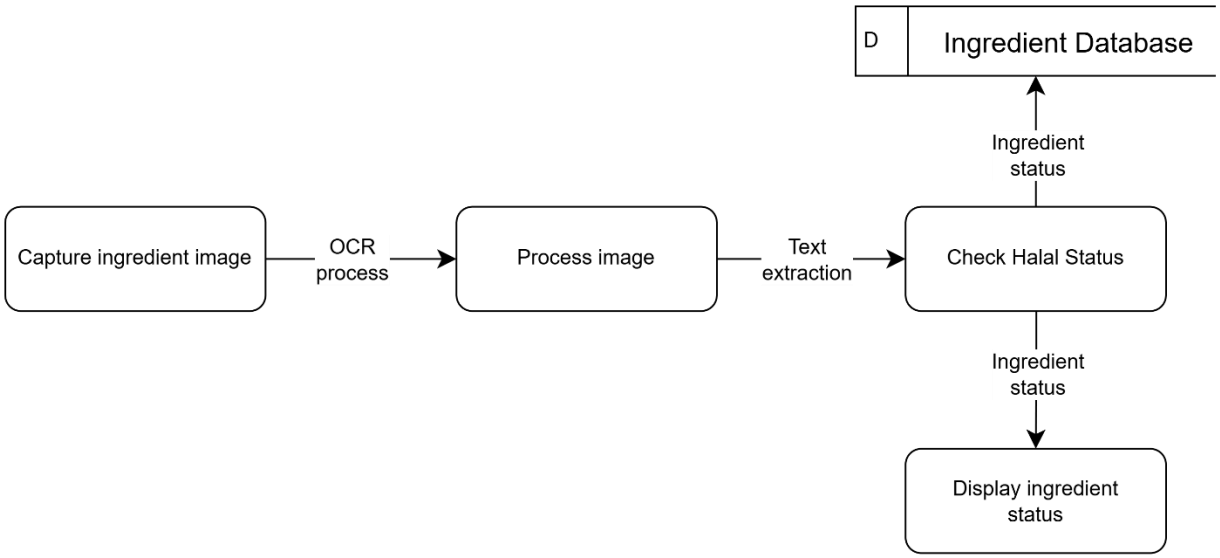

Figure 12. Data Flow Diagram Level 1 for Scan Ingredient process

Users capture an image of the ingredient using the device's camera as demonstrated in Figure 12. The image is processed using OCR (Optical Character Recognition) to extract text from the image. The extracted ingredient data will be sent to the Halal Status database for status checking. Once the database done with status checking, it will return the halal status for each ingredient (halal, non-halal, or dubious). The halal status will be stored into the database for report purpose. Furthermore, the final halal status of the product based on the status of its ingredients will be displayed.

e) DFD Level 1—Manage ingredient list

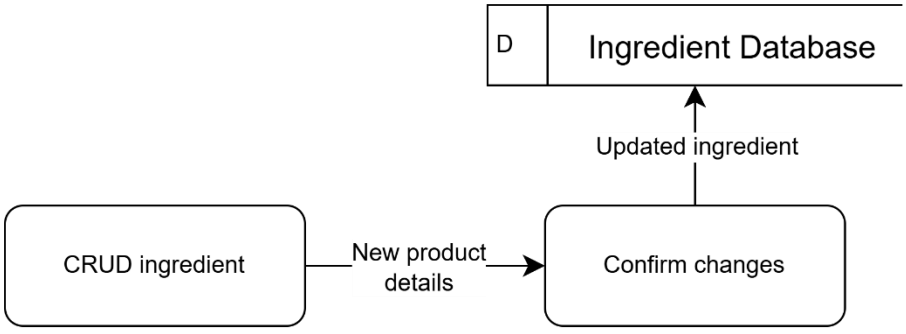

Figure 13. Data Flow Diagram Level 1 for Manage Ingredient List process

The admin performs CRUD (Create, Read, Update, Delete) operations on the ingredient list as seen in Figure 13. After making any changes, the admin confirms these updates. The system then updates the ingredient database to reflect the changes made by the admin.

3.6.2 Database Design

a) ERD

Figure 14 shows a conceptual overview of HalalScan's database structure, showcasing the entities, attributes and relationships between them. This diagram is essential for visualizing how data is interconnected and supports the system's operations.

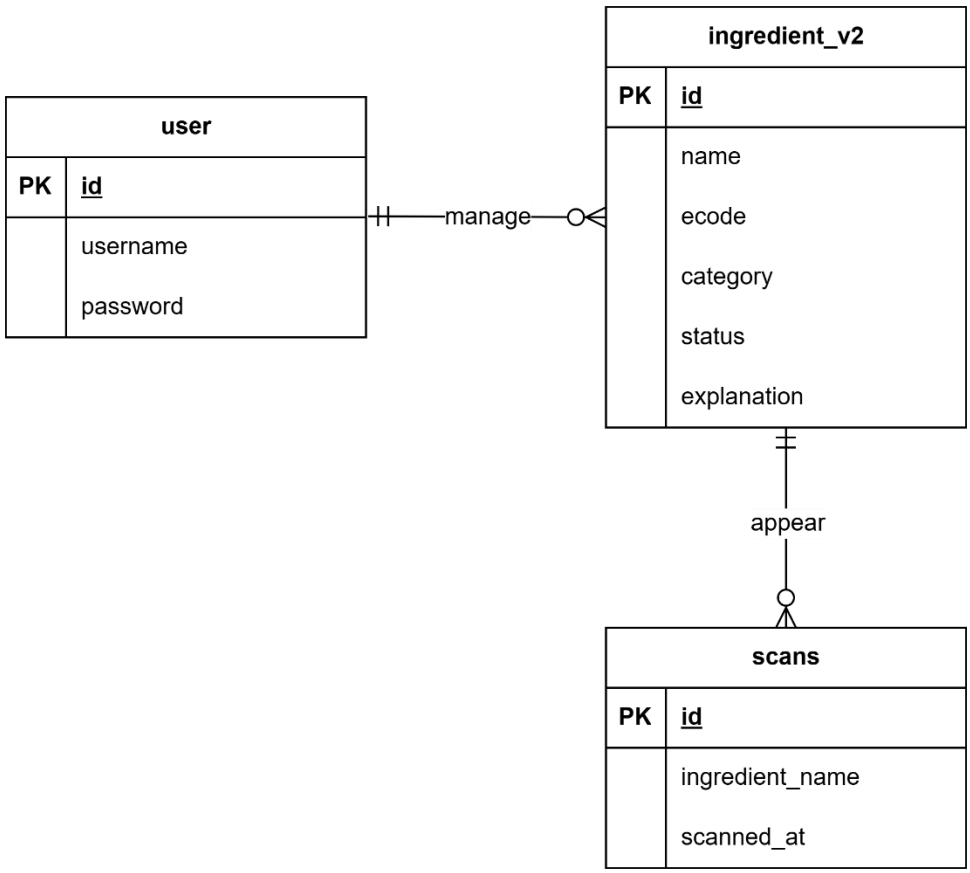

Figure 14. HalalScan's entity relationship diagram

i. A one-to-many relationship exists between user (admin) and ingredient_v2, where one admin can manage (create, update, edit and delete) many ingredients. However, each ingredient is managed by one admin only.

ii. A one-to-many relationship exists between ingredient_v2 and scans, which one ingredient can appear in multiple scan histories, but each scan entry refers to a single ingredient.

b) Data dictionary

The data provides a detailed overview of the database structure, documenting each table's fields, data types, keys, and constraints to ensure clarity and consistency in data management.

Table 7 shows the data dictionary for admin

Field Name	Data type	Description	Key	Constraint
Id	INT	Unique identifier for admin	PK	AUTO_INCREMENT
Username	VARCHAR (80)	Admin's username		NOT NULL
Password	VARCHAR (120)	Admin's password (hashed)		NOT NULL

Table 8 shows the data dictionary for ingredients

Field Name	Data type	Description	Key	Constraint
id	INT	Unique identifier for the ingredient	PK	AUTO_INCREMENT
Name	VARCHAR (255)	Ingredient's name		NOT NULL
ecode	VARCHAR (10)	Ingredient's E-code		
Category	VARCHAR (255)	Ingredient's category		
status	VARCHAR (50)	Ingredient's status		
explanation	TEXT	Details about the ingredient		

Table 9 shows the data dictionary for scans

Field Name	Data type	Description	Key	Constraint
Id	INT	Unique identifier for each scan	PK	AUTO_INCREMENT
Ingredient_name	INT	Ingredient's name		NOT NULL
Scanned_at	DATE	Date when the scan was performed		NOT NULL

3.7 User Interface Design

This user interface design focuses on creating a simple and user-friendly system to verify the halal status of a product's ingredients. At this stage, low-fidelity wireframes were created to map out the structure and flow of key interfaces.

Figure 15. User interface for HalalScan's homepage

Figure 15 illustrates that users are enable to either scan a product's ingredient. The general information box provides details regarding halal status of a product, and features articles related to the halal industry. Furthermore, the homepage includes an aggregate report, allowing users to view the most frequently scanned ingredients by other users, selectable by weekly or monthly intervals. The homepage also displays popular search trends, highlighting ingredients that are frequently queried.

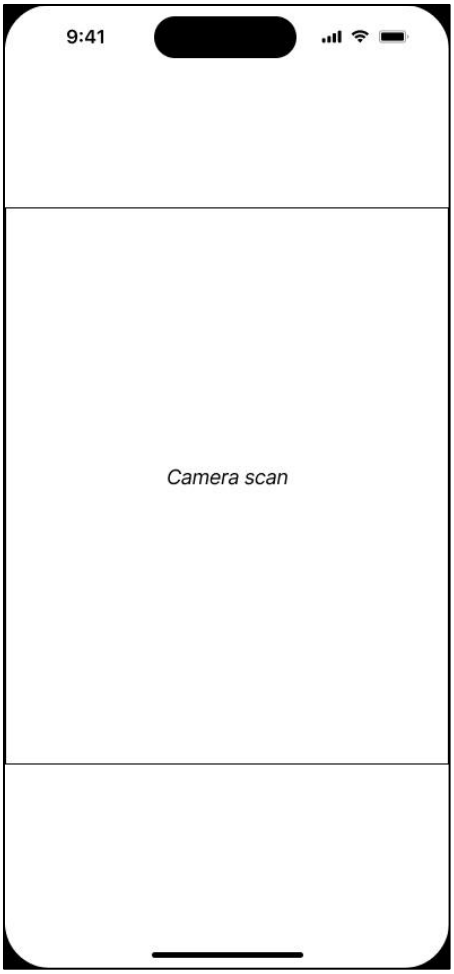

Figure 16. Camera interface for scanning ingredients

Figure 16 shows the placeholder for camera scanning upon clicking the Scan button on the homepage. The system activates the user's device camera to allow real-time image capture. The user position the product's ingredient list within the designated scanning frame to ensure proper alignment and clarity. The system will continuously scans the visible text within the frame.

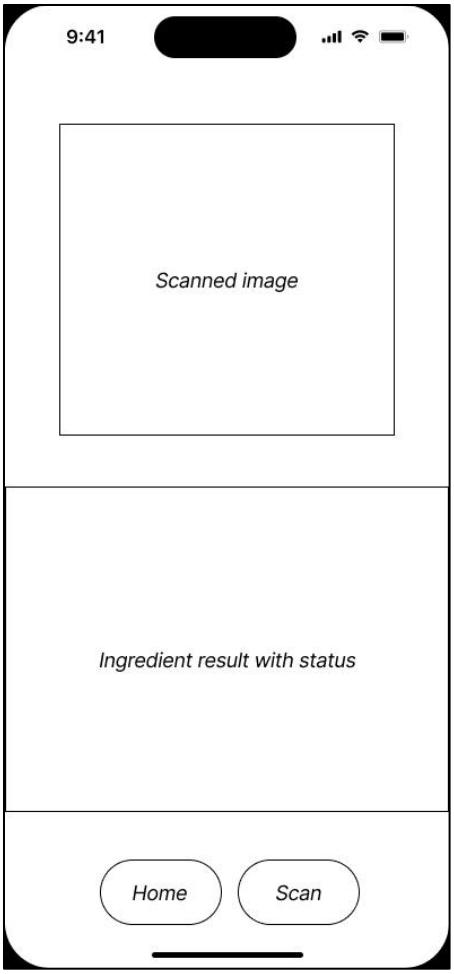

Figure 17. Result page

Figure 17 presents the scanned image of the ingredients along with a list of each ingredient's halal status, categorized as Halal, Dubious or Non-Halal once the text is recognized. The text will undergo a cleaning process to remove noise, unwanted characters, and any formatting issues and the cleaned text is then split into individual ingredients. Furthermore, users can click on the result status for each ingredient to know more about the ingredient. This page also includes a navigation bar that allows users to return to the homepage and initiate a new scan for another product.

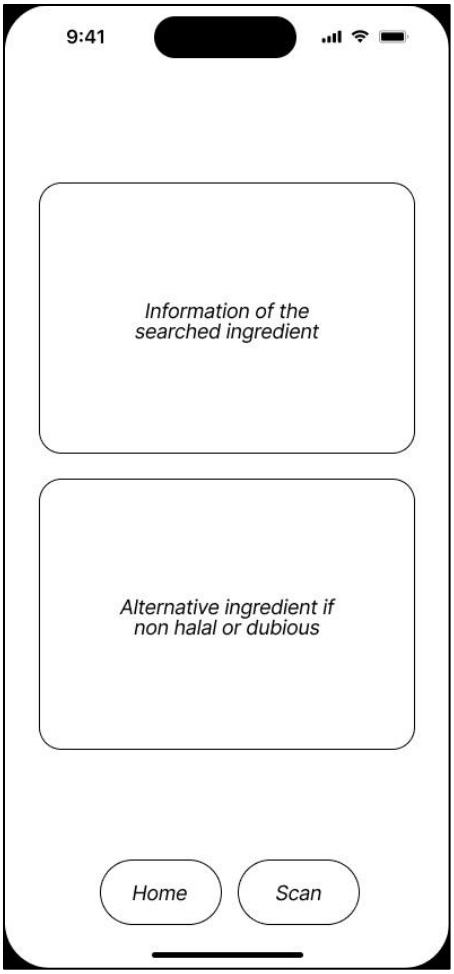

Figure 18. Information page for the ingredient

Figure 18 displays the ingredient name, its halal status and a brief summary once the ingredient is clicked from ingredient results from Figure 17. Additionally, if the ingredient is identified as Non-Halal or Dubious, the system also provides for alternative suggestion on this page.

3.8 Summary

The HalalScan system allows users to verify the halal status of the product up to ingredient level by scanning the ingredients list. Using OCR technology, the system extracts and processes ingredient data, then verifies their halal status through halal ingredient database. The product is marked as halal, non-halal, or flagged for uncertainty based on the results. Additionally, the system aggregates and visualizes trends from scanned ingredients, displaying them in user-friendly charts on a weekly or monthly bases. All data, including halal statuses and trends, is stored in an SQLite database for easy tracking and future reference, ultimately providing users with valuable insights and informed purchasing decisions.

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Introduction

This chapter marks a pivotal stage in the realization of the proposed solution to verify imported product based on its ingredient list. It details the design and implementation process of HalalScan. The design phase focused on creating a user-friendly interface, defining clear system architecture, and structuring database to effectively meet the project's objectives.

The subsequent implementation involved translating these design specifications into a working model. This chapter will outline the key technologies and methodologies employed, such as the application of Tesseract OCR technique to verify the status for each ingredient from an imported product. Each section in this chapter provides insights into the components developed, ow they interact, and how they contribute to the overall system. By detailing the practical steps taken, this chapter demonstrates how the proposed solution was brought to life.

4.2 Installation Instructions

The HalalScan system was developed using the Python programming language and relies on a range of supporting packages for functionality such as image processing, OCR (Optical Character Recognition), database interaction, and web framework features. The system is platform independent but was primarily tested on Windows 11.

1. Python

   Python is downloaded from: https://www.python.org/downloads/. Ensure it is added to the system PATH during the installation.

2. Required Python Libraries

   Install this following Python packages using pip:

   ```
   pip install flask
   pip install pytesseract
   pip install pillow
   pip install flask-admin
   ```

3. Tesseract OCR

   - Install the Tesseract OCR engine from https://github.com/UB-Mannheim/tesseract/wiki. After installing, set the path in the code. Example,

```
pytesseract.pytesseract.tesseract_cmd =
r'C:\Users\misha\AppData\Local\Programs\Tesseract-OCR\tesseract.exe'
```

- Install the Python wrapper (pytesseract). Install it using `pip install pytesseract`. Install `pip install pillow` and `pip install opencv-python` as these libraries are used to handle and preprocess the images before passing them to Tesseract.

4. Python Environment Setup

   Open terminal or command prompt in the project folder:

```
python -m venv venv
./venv/Scripts/Activate
pip install -r requirements.txt
```

   Once the dependencies are installed, the system can be launched with:

```
python app.py
```

4.2.1 Project Dependencies

HalalScan system relies on several Python libraries to support its core functionalities.

Table 10 outlines the libraries used in the development

Libraries	Function
Flask	Building the web application and routing
Flask-Admin	Enabling a secure admin panel to manage the ingredient database
Flask-login	Authentication and restricting access to admin-only features
Flask-SQLAlchemy	Object-relational mapping and database operations with SQLite
pytesseract	Perform optical character recognition (OCR) on ingredient images
Pillow (PIL)	Enhance and preprocess images before OCR
RapidFuzz	Perform fuzzy matching between extracted text and database ingredient names
datetime	Used for logging timestamps of user scans.
os / re / io / base64	Built-in Python modules used for file operations, regular expressions, encoding, and memory handling.

4.3 File Structure

```
HalalScanV2/
│
├── admin.py                          # Admin panel setup and configuration
├── app.py                            # Main Flask application file
├── init.py                           # Initialization file for the project
├── requirements.txt                  # List of all required Python packages
├── README.md                         # Project overview and usage instructions
│
├── backend/                          # Backend-related files and data
│   ├── halal_scan.db                 # SQLite database
│   ├── database.py                   # SQLAlchemy models and database setup
│   ├── migrate_data.py               # Script to migrate or seed database content
│   ├── scans.csv                     # CSV file used during testing or migration
│   ├── common ingredients.pdf        # Reference material for ingredients
│   └── FOOD ADDITIVE LISTING 5.pdf   # Reference file for food additives
│
├── static/                           # Static frontend assets (CSS and images)
│   ├── scan.css                      # Specific styling for scan page
│   ├── styles.css                    # General styling
│   ├── icons/                        # Custom icons and images used in the UI
│   └── uploads/                      # Temporary folder for uploaded images
│
├── templates/                        # HTML templates for the web pages
│   ├── index.html                    # Homepage
│   ├── scan.html                     # Scan results page
│   ├── camera_scan.html              # Camera capture interface
│   └── login.html                    # Admin login page
│
└── venv/                             # Python virtual environment folder
```

Full implementation code can be found in Appendix E: HalalScan complete source code

4.4 Implementation of HalalScan

This section describes the implementation details of HalalScan system, focusing on the key user interface components and their respective functionalities. Each part of the system was designed to provide a seamless and intuitive experience for users when checking the halal status of product ingredients. The interface covers all essential user interactions, starting from the homepage and continuing through the scanning process, result display, ingredient detail popups, and viewing statistics. The following subsections detail each page and feature within the system.

4.4.1 Homepage

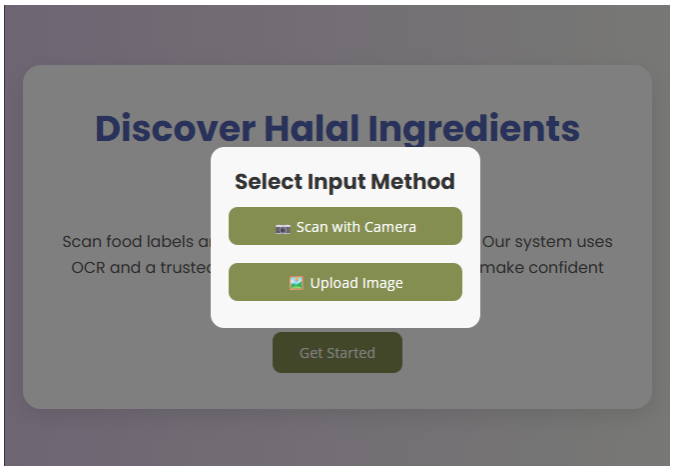

Figure 19. HalalScan's homepage

Figure 19 displays the main landing page of the HalalScan system. From here, users can either scan ingredient list with camera directly using camera or upload an image. The design focuses on simplicity and accessibility. The frontend is built using HTML, styled with CSS. Two buttons are presented to the user for navigation to other parts of the system.

```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/scan-camera')
def scan_camera():
    return render_template('camera_scan.html')

@app.route('/scan')
def scan():
    return render_template('scan.html', trigger_upload=True)
```

The code above defines a route for index page (the whole homepage), `/scan-camera` that will direct user to the camera scan pages and `/scan` that accepts an image user uploaded, processes it with OCR, and renders the result.

4.4.2 Camera Scan Page

Figure 20. HalalScan's camera scan page

This page allows users to scan the product's ingredients list using the device's camera. The interface as illustrates in Figure 20, includes a guide box where the user should align the ingredient list image. Proper alignment that the image is captured with clarity and accuracy. Once the image is captured using the Scan button, it is sent to the backend where Tesseract OCR (Optical Character Recognition) is employed to the extracted ingredients with data in the database to determine their halal status.

```
const video = document.getElementById('camera');
navigator.mediaDevices.getUserMedia({ video: { facingMode: { exact:
"environment" } } })
      .then(stream => video.srcObject = stream)
      .catch(() => {
        navigator.mediaDevices.getUserMedia({ video: true })
```

Once the user initiates a scan via the Scan button, a canvas element is dynamically created to capture only the targeted area of the video feed. This cropped region corresponds to the center guide box, improving OCR accuracy by ignoring unnecessary visual noise.

```
const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth * 0.8;
      canvas.height = video.videoHeight * 0.3;

      const ctx = canvas.getContext('2d');
      const cropX = video.videoWidth * 0.1;
      const cropY = video.videoHeight * 0.35;
      ctx.drawImage(
        video, cropX, cropY, canvas.width, canvas.height, 0, 0, canvas.width,
canvas.height
      );
```

The cropped image is then converted to a JPEG Blob and sent to the backend via a POST request using the `fetch()` API. At the backend, Tesseract OCR is employed to extract text from the image and match the identified ingredient against the database to determine the status. The system handles both successful and failed capture scenarios, storing the result in `sessionStorage` for retrieval on the result page.

```
canvas.toBlob(function(blob) {
  if (!blob) {
     alert("Failed to capture image. Please try again.");
     return;
  }
const formData = new FormData();
formData.append('file', blob, 'scan.jpg');
fetch('{{ url_for("ocr") }}', {
  method: 'POST',
  body: formData
})
```

4.4.3 Upload Image

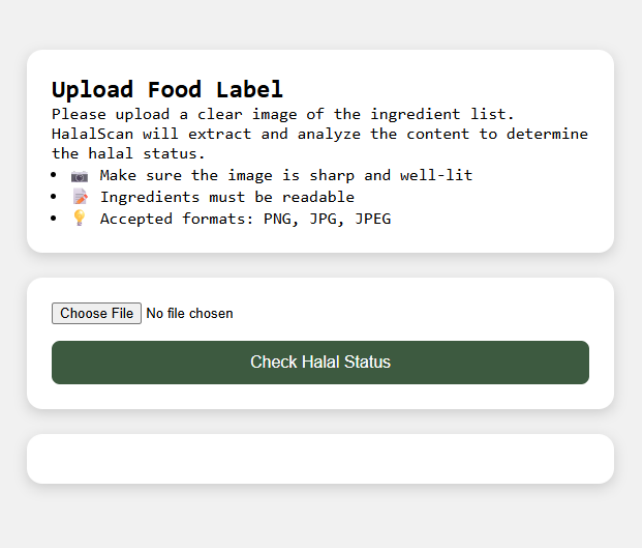

Figure 21. HalalScan's upload image section

In addition to real-time scanning, Figure 21 illustrates that the system provides users with an alternative method to verify product ingredients by uploading an existing image file. This flexibility accommodates users who may already possess a saved image of an ingredient list or prefer manual file selection over live camera capture. The interface enables users to select an image from their device storage. The *accept*="image/*" attribute ensures that only image files are selectable, maintaining data type consistency and preventing unsupported uploads. A dedicated submission button labelled “Check Halal Status”, triggers the OCR processing pipeline upon user interaction.

```
<form id="upload-form" enctype="multipart/form-data" class="upload-box">
  <input type="file" id="upload-image" accept="image/*" />
  <button type="submit" class="check-button">Check Halal Status</button>
</form>
```

Once a user selects a file, the image is rendered as a preview within the interface, enhancing transparency and allowing users to confirm that the correct image has been chosen before submission. On form submission, the selected image is sent to the backend where the same Tesseract OCR engine is used, consistent with the camera capture workflow.

4.4.4 Result Page

When user uploads or captures an image, the system uses Optical Character Recognition (OCR) to extract text from the image. This is handled using the `pytesseract` library in Python, which is an OCR engine that can read text from images. After the user submits an image (either scanned or uploaded), the backend processes the extracted text through several stages:

1. Cleaning the raw OCR text

   The raw text obtained from OCR is cleaned by removing unwanted characters such as brackets, punctuation, and special symbols. All text is converted to lowercase to ensure case-insensitive matching with the ingredient database. This normalization step helps improve matching accuracy.

```
clean_text = text.lower()
clean_text = re.sub(r'[(){}\[\];:_\-]', ',', clean_text)
clean_text = re.sub(r'[^a-zA-Z0-9\s,]', '', clean_text)
clean_text = re.sub(r'[^\w\s,]', '', clean_text)
clean_text = re.sub(r'\(\s+', '(', clean_text)
clean_text = re.sub(r'\s+\)', ')', clean_text)
clean_text = clean_text.replace('(', ',').replace(')', ',')
```

2. Tokenizing the text

   The cleaned text is split into manageable units like phrases and individual words by delimiters such as commas, new lines, whitespace, and dashes. This tokenization allows the system to identify possible ingredient names or E-codes more effectively.

```
tokens = [token.strip() for token in re.split(r'[,|\n]', clean_text) if
token.strip()]
words = re.split(r'[\s\-]+', clean_text)
```

3. Matching ingredients

   Each token and word are matched against known ingredient names in the database using both direct substring matching and fuzzy matching to minimize the OCR errors or spelling variations. E-codes are detected via regex and matched similarly. *Full implementation code can be found in Appendix A.*

```
known_names = [ingredient.name.lower() for ingredient in
Ingredient.query.all() if ingredient.name]
for phrase in tokens:
. . .
fuzzy_result = process.extractOne(phrase, known_names, score_cutoff=85)
```

```
. . .
ecodes = re.findall(r'\b(?:e|ins)?\s*\d{3,4}\b', text.lower())
. . .
```

4. Once matched, the ingredients list is sent to the frontend, where the system highlights each ingredient and E-code within the OCR text dynamically. *Full implementation code can be found in Appendix B.*

```
if (name && !matched.has(`name-${name}`)) {
      const nameRegex = new RegExp(`\\b${name.replace(/[-
   /\\^$*+?.()|[\]{}]/g, '\\$&')}\\b`, 'gi');
. . .
if (/^\d{3,4}$/.test(ecode)) {
      const eVariants = [`e${ecode}`, `E${ecode}`, `ins${ecode}`,
   `INS${ecode}`, `${ecode}`];
. . .
return placeholder;
```

5. Identify whole product status

   After identifying the individual ingredients and its statuses, the frontend aggregates the counts of ingredients by their status categories. Then, based on simple conditional logic, it derives a summary status message describing the overall halal status of the product. *Full implementation code can be found in Appendix C.*

```
const count = { halal: 0, haram: 0, doubtful: 0 };
matchedIngredients.forEach(ing => {
  const status = (ing.status || '').toLowerCase();
  if (status === 'halal') count.halal++;
  else if (status === 'haram') count.haram++;
  else if (status === 'doubtful' || status === 'syubhah') count.doubtful++;
  });
const totalIngredients = count.halal + count.haram + count.doubtful;
```

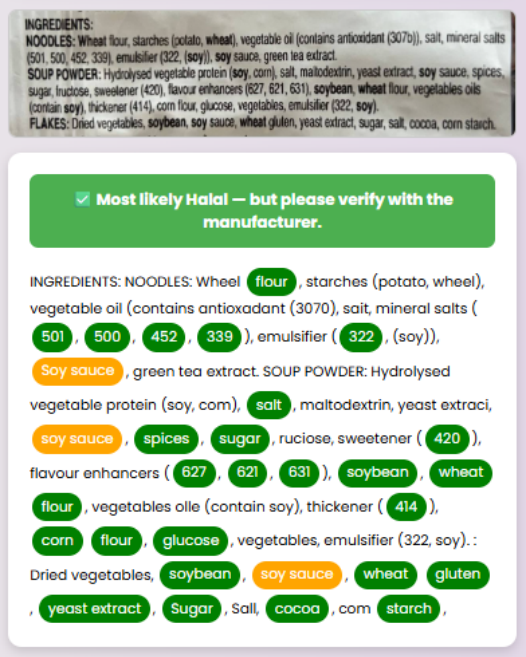

Figure 22. HalalScan's result section

Figure 22 shows the extracted ingredients are displayed along with their corresponding halal status. The results are shown using a color-coded pill-style UI for intuitive understanding: Green for Halal, Yellow for Doubtful and Red for Haram. Ingredients that are found in the database appear these clickable-coloured pill containers. Clicking on an ingredient opens a popup with detailed information, name of the ingredient, status, category, E-code (if applicable) and the explanation. Ingredients that are not found in the database are displayed as plain text, indicating they are unrecognized by the system. Additionally, the summary of the whole product's status is shown to assure the user about the product status.

4.4.5 Ingredient Detail Popup

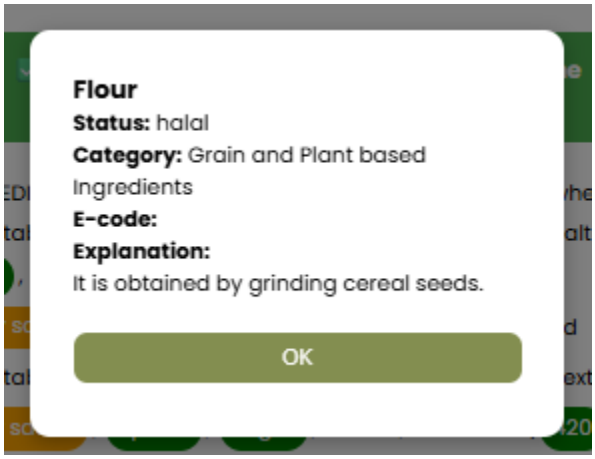

Figure 23. Ingredient's detail popup

When a user clicks on an ingredient extracted from the scanned or uploaded image, a popup window appears to display more detailed information about the selected ingredient as illustrated in Figure 23. This popup provides clarity and transparency by showing the ingredients' name, status, E-code (if any), category, and explanations. Ingredients are displayed in pill-shaped containers with colour coding to indicate status –Green for *Halal*, Yellow for Doubtful and Red for *Haram*. Once the user has reviewed the details, they can click "OK" to close popup and return to result view.

4.4.6 Statistic Section

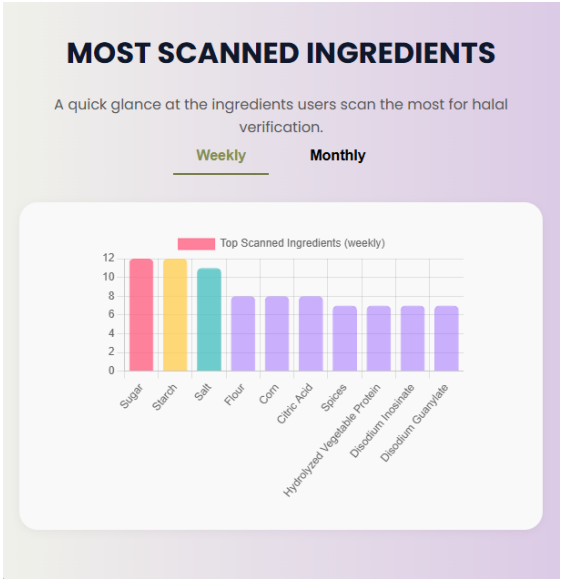

Figure 24. HalalScan's statistic section

The section of the system as shown in Figure 24 provides users with a graphical overview of the most frequently scanned ingredients, helping to identify which ingredients commonly appear in scanned products. The graph supports two time-based views, Weekly and Monthly, which users can toggle to observe ingredient trends over different periods.

The route responsible for fetching this data is defined as:

```
@app.route('/get-scanned-stats')
def get_scanned_stats():
```

Depending on whether the user selects a “weekly” or “monthly”, the system calculates the appropriate start date. It uses Python’s `datetime` and `timedelta` to determine the range.

```
if period == 'monthly':
        start_date = datetime.now() - timedelta(days=30)
    else:
        start_date = datetime.now() - timedelta(weeks=1)
```

Once the time range is established, the system queries the Scan table from the SQLite database. It filters the data to include only scans within the selected time frame, groups the records by ingredient_name, and counts how many times each ingredient was scanned.

```
scanned_stats = db.session.query(Scan.ingredient_name,
db.func.count(Scan.id).label('total')) \
        .filter(Scan.scanned_at >= start_date) \
        .group_by(Scan.ingredient_name) \
        .order_by(db.func.count(Scan.id).desc()) \
        .limit(10) \
        .all()
```

This ensures that only 10 most scanned ingredients are retrieved, sorted in descending order of frequency. The result is then converted into JSON format and return to the frontend. The frontend graph uses this dynamic data to visually display which ingredients appear most often in user scans. Each data point represents the number of scans per ingredient within the selected period. This interactive and data-driven feature enhances user awareness by highlighting commonly occurring ingredients, helping users better understand product compositions over time.

4.5 Database Implementation

The database plays a crucial role in managing ingredient data and storing user scan history to support both system functionality and analytical features. The HalalScan system utilizes SQLite for lightweight, embedded data storage, supporting both real-time verification and usage analytics.

4.5.1 Database Structure

The system comprises two core tables. The database is normalized to avoid redundancy and support efficient queries, particularly for ingredient lookups during OCR processing and result display.

A. Ingredient Table

This table contains structured data on each ingredient data on each ingredient, including the name of the ingredient, it's associated E-code (if available), the ingredient category, it's halal classification status – halal, haram, or doubtful and its explanation. This table is the main reference source for ingredient verification during image analysis and status checking.

```
class Ingredient(db.Model):
    __tablename__ = 'ingredients_v2'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    ecode = db.Column(db.String(10), nullable=True)
    category = db.Column(db.String(50))
    status = db.Column(db.String(20), nullable=False)
    explanation = db.Column(db.Text)
```

B. Scan Table

This table logs the images processed by the system, along with the extracted data and timestamps. The timestamps are used to generate aggregated data visualizations shown on the homepage, such as the most frequently scanned ingredients on a weekly or monthly basis.

```
class Scan(db.Model):
    __tablename__ = 'scans'
    id = db.Column(db.Integer, primary_key=True)
    ingredient_name = db.Column(db.String(100), nullable=False)
    scanned_at = db.Column(db.DateTime, nullable=False)
```

4.5.2 Admin Panel for Data Management

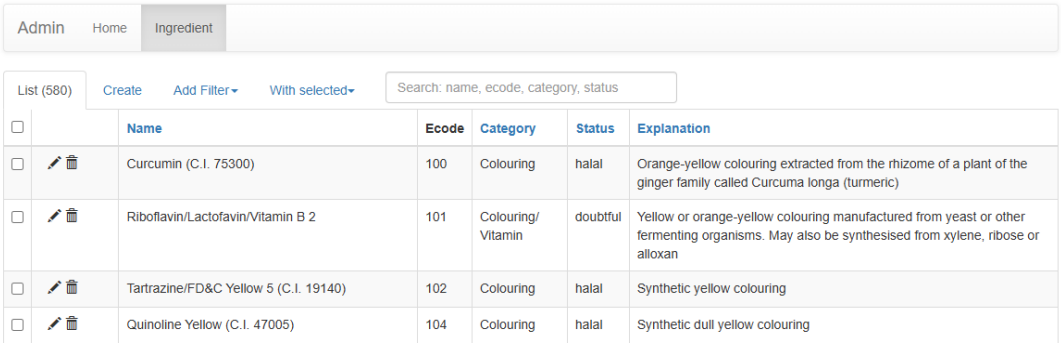

Figure 25. Admin panel for database

This administrative functionality is vital for enabling continuous database updates and corrections without requiring direct modifications to the database file, thus support maintainability and scalability. To maintain and manage the contents of the database, a secure, admin-only panel was developed using Flask-Admin. This page is not publicly accessible and is protected by login authentication to ensure data integrity and security.

The admin panel enables system administrators to perform full CRUD (Create, Read, Update, Delete) operations on the ingredient dataset as shown in Figure 25. Specifically, administrator can add new ingredients to the database, complete with category, E-code, status, and explanation. They can also view and search existing records using filters such as category, presence of E-code, or status classification. Furthermore, administrator can edit or correct entries, ensuring that the information remains accurate and current and remove outdated or duplicate ingredients to maintain clean database.

4.6 Software Testing

To ensure functionality, reliability, and usability of the HalalScan system, a series of software testing activities were conducted throughout the development process. These included both manual functionalities testing to verify that individual features performed as intended, and usability testing involving real users to evaluate the overall user experience. Testing focused on the core modules of the system, such as camera scan, image upload, result display interfaces and navigation across the system. Each feature was carefully tested to identify bugs, assess system behaviour under normal usage, and ensure the application met its intended goals.

4.6.1 Functionality Testing

Functionality testing was conducted to verify that each core component of the HalalScan system performs as intended. This includes testing key user-facing features such as homepage, scanning function, image upload functionality, and result display as well as the backend admin panel used for managing ingredient data. Test cases were designed to validate input, expected outputs, and system behaviour under normal and edge conditions. The results of each were recorded and analysed to ensure the system meets its functional requirements.

4.6.1.1 Test Cases

Table 11 displays the test case for homepage

Test Module: Homepage module			
ID	Functionality	Expected Result	Status
HM01	Click Get Started button	Opens popup with selects input method	Pass
HMP02	Click Scan from input method popup	Navigates to Camera Scan page	Pass
HMP03	Click Upload Image from input method popup	Navigates to Upload Image page	Pass
HMP04	Click Weekly button in Graph	Displays weekly data	Pass
HMP05	Click Monthly button	Displays monthly data	Pass

Table 12 displays the test case for scan function

Test module: Scan module			
Case ID	Functionality	Expected Result	Status
SCN01	Open page with camera permission	Device camera opens.	Pass
SCN02	Open page without camera permissions	Displays an alert message	Pass

SCN03	Click Scan button	Captures image and navigates to Scan page (Upload & Result Page)	Pass
SCN04	Status detection	Detect status from captured image	Fail
SCN05	Click Cancel button	Returns to Homepage	Pass
SCN06	Click Nav button: Home	Navigates to Homepage	Pass
SCN07	Click Nav button: Upload	Navigates to Scan page (Upload & Result page)	Pass

Table 13 shows the test case for upload function

Test Module: Upload Image			
ID	Functionality	Expected Result	Status
UPL01	Click “Choose file”	Opens device gallery or file manager	Pass
UPL02	Upload image from gallery	Image loads in the interface	Pass
UPL03	Upload unsupported file type	Error message is shown	Pass
UPL04	Click “Check Halal Status”	Systems displays scanned image + list of ingredients with status	Pass
UPL05	Click on the result pill	Open popup with ingredient details	Pass
UPL06	Click “OK” on popup	Closes popup	Pass
UPL07	Non-existent ingredient shown as text	Displays as non-clickable plain text	Pass
UPL08	Click on Nav button: Home	Navigates to Homepage	Pass
UPL09	Click on Nav Button: Scan	Navigates to Camera Scan page	Pass

Table 14 shows the test case for admin

Test module: Admin			
ID	Functionality	Expected Result	Status
ADM01	Go to /login and enter credentials.	Admin dashboard loads successfully	Pass
ADM02	Go to /login and wrong credentials.	Display invalid credentials message	Pass
ADM03	Create new ingredient list, fill in the list and save.	New ingredient appears in the list	Pass
ADM04	Click edit on a row, modify fields, and save.	Ingredient info is updated	Pass
ADM05	Click delete on a row and delete list.	Alert message appeared to confirm deletion	Pass
ADM06	Select filter by category/status/E-code presence	Table shows filtered results	Pass
ADM07	Type a keyword in search bar.	Matching ingredients shown	Pass

4.6.1.2 Test Analysis

The testing phase of the HalalScan system revealed several challenges that impacted the overall performance and accuracy of the application. A major issue encountered involved the camera-based scanning, where ingredient lists printed on product packaging were often not detected properly. This was frequently due to environmental factors such as poor lighting, glare caused by glossy packaging surfaces, or shadows affecting image clarity. These visual distortions hindered the OCR engine from extracting text accurately, especially when high resolution is not maintained. As noted by Adjetey and Adu-Manu (2021), higher resolution images tend to yield higher OCR confidence scores and recognizes more meaningful words, while lower resolution images lead to reduced text certainty and accuracy.

Additionally, certain ingredients that were already present in the database were not reflected in the OCR scan result. Upon closer inspection, this was caused by the failing of the Tesseract OCR to correctly recognize some ingredient names, resulting in mismatches between the

scanned text and database entries. This led to approximately 70% accuracy, with the remaining affected by errors in text recognition. For example, as shown in Figure 26, “Malic Acid” appears in the OCR output but it is not clickable. This suggests it was not matched correctly. However, as confirmed in Figure 27, “Malic Acid” is indeed present in the ingredient database, highlighting a limitation in text recognition rather than missing data.

Figure 26. OCR result displaying "Malic Acid" extracted from the image but not clickable.

		Name	Ecode	Category	Status	Explanation
☐	✎ 🗑	Malic Acid	296	Food Acid	halal	Commercially synthesised by means of heating malic with sulphuric acid

Figure 27. Admin panel confirms "Malic Acid" exists in the ingredient database.

In some cases, characters were incorrectly interpreted, resulting in spelling errors in the extracted text. For instance, from Figure 28, the word “wheat”, “corn”, “salt”, “fructose”, and “yeast extract” was recognized as “wheel”, “com”, “sait”, “ruciose” and “yeast extraci” which led to failed matches against the database even though the ingredient existed. This reflects a broader challenge in OCR technology, where character segmentation and alphabet shape variability can lead to recognition errors (Patience, Amaechi, George, and Isaac, 2024).

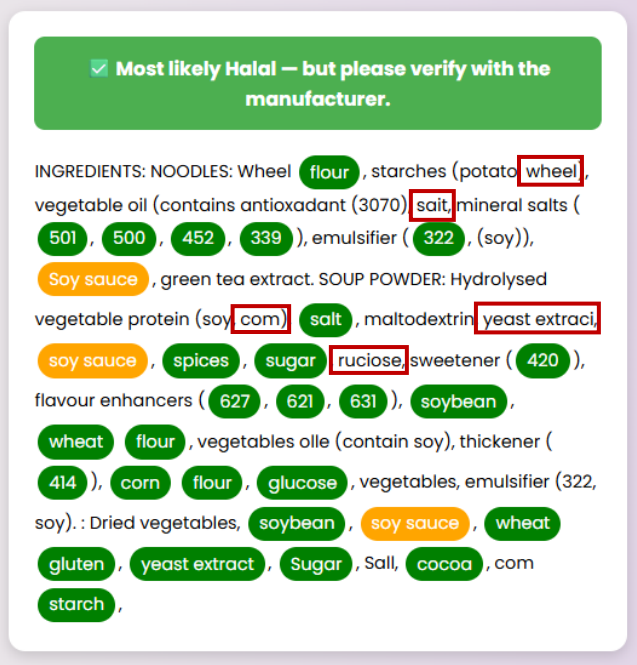

Figure 28. OCR misrecognition of “wheat”, “salt”, “corn”, “yeast extract” and “fructose”

4.6.2 Usability Testing

Usability testing was carried out to evaluate the ease of use and user satisfaction with the system. A task-based approach was used, where users were given instructions to perform specific tasks such as scanning an ingredient list, uploading an image, or navigating the interface. Each user marked whether they completed the task successfully and whether guidance was needed. A post-test questionnaire followed to collect user feedback regarding system usability, navigation confidence, and overall experience.

4.6.2.1 Test Analysis

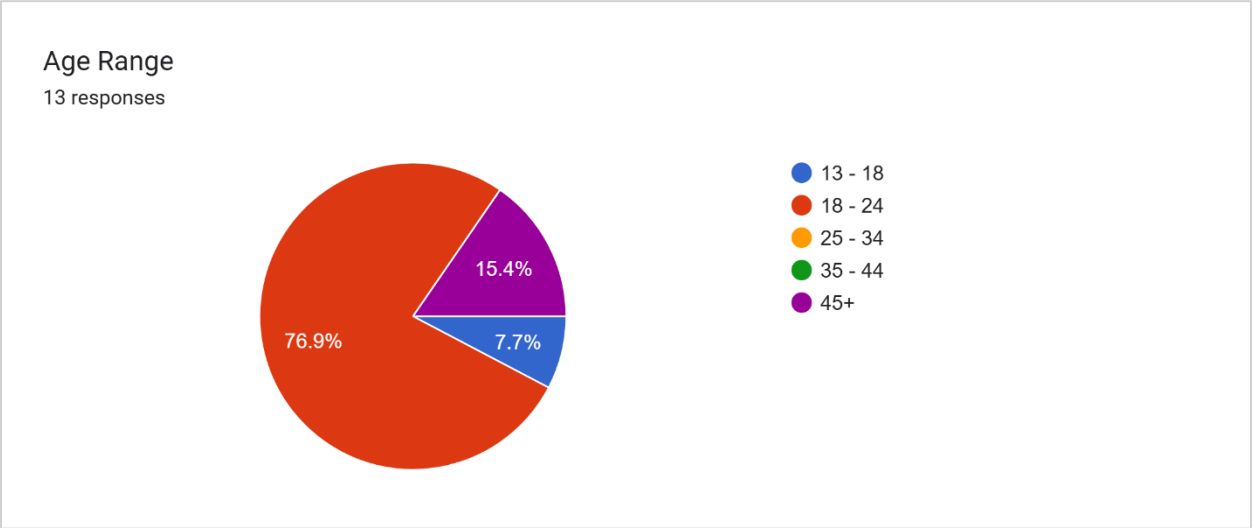

Figure 29. Respondent age ranges

The usability testing was conducted with a total of 13 participants, ranging in age and background to ensure diverse feedback. The age distribution showed that the majority (76.9%) were between 18-24 years of old, followed by 15.4% aged 45 and above, and 7.7% from the 13-18 age group as illustrates in Figure 29. In terms of occupation, 46.2% of respondents were university students, 46.2% were working adults, and the remaining 7.7% were high school students. Notably, none of the participants had previous experience using similar applications. This lack of prior exposure to comparable apps suggest that their responses provide unbiased insights into the system's intuitiveness and learning curve. *Refer to Appendix D for the full usability test questions, including the scenario-based tasks and post-test questionnaire.*

For scenario-based tasks, participants were asked to perform four core actions within the system

1. Scan an ingredient list using their device's camera
2. Upload an ingredient image from their device
3. Click on ingredient image form their device
4. Navigate back to the homepage

Only the first task (Scan an ingredient list using their device's camera) has minor difficulty, with four participants requiring guidance to complete it as shown in Figure 30. This difficulty was attributed to unfamiliarity with camera permissions and positioning the product correctly for OCR capture. The remaining tasks were completed successfully by all users without assistance, suggesting that once the user enters the system, the flow and functions are intuitive.

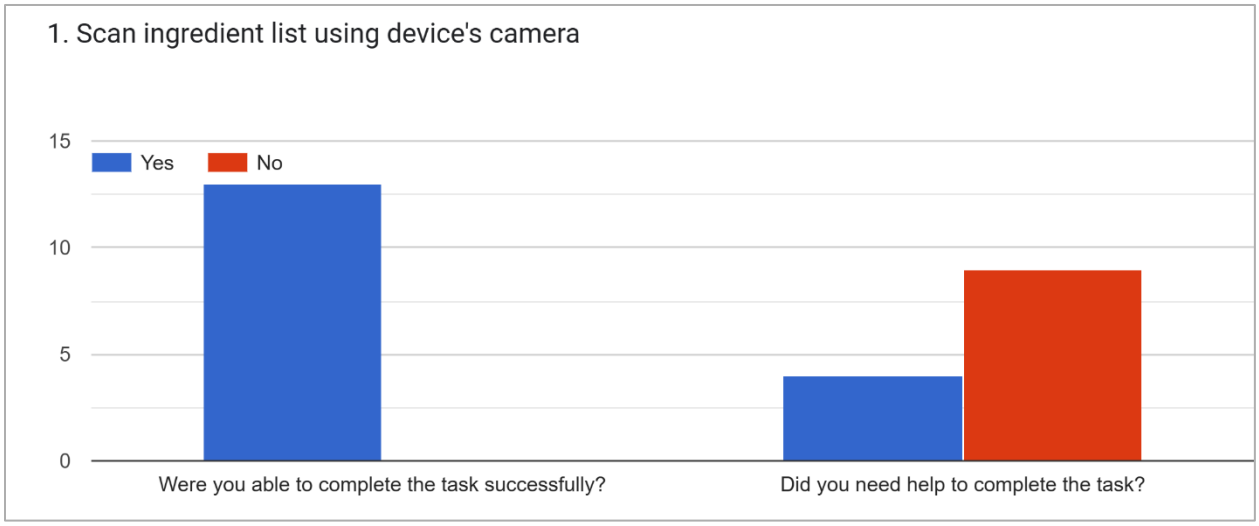

Figure 30. Scenario-based task performance on Question 1

Post-test Questionnaire

For the post-test questionnaire, participants were asked to rate their agreement with five usability statements on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). The results were overwhelmingly positive. For all five statements, 12 out 13 respondents selected the highest rating (5), and one respondent selected 4. These responses indicate a high level of user satisfaction, system usability, and confidence in navigation.

Statements included:

1. The system was easy to use
2. I was able to complete tasks without much help
3. The result display was easy to understand
4. I would use this app again if I needed to check ingredients
5. I felt confident navigating through the system.

From the testing results, it can be concluded that HalalScan is user-friendly and intuitive, even for first-time users with no experience in similar apps. The few usability issues observed, such as needing assistance with the camera scan function, highlight a potential area for improvement, such as adding in-app guidance or onboarding instructions. The consistent high ratings in the post-test questionnaire reinforce that the system has successfully met its usability goals.

4.7 Summary

This chapter detailed the implementation and testing of the HalalScan system. Key features such as the homepage, camera scan, image upload, result display, ingredient detail popup, and statistics page were introduced. The system uses Tesseract OCR and Flask to process images and retrieve ingredient information from an SQLite database. An admin panel supports data management through CRUD operations. The chapter also covered software testing, including functionality, and usability tests, to evaluate system performance and user experience.

CHAPTER 5: CONCLUSION AND FUTURE WORKS

5.1 Introduction

This chapter concludes the development of the HalalScan system by summarizing its accomplishments, analysing its limitations, and proposing future enhancements. The aim is to reflect on the goals achieved throughout the project and identify possible improvements to enhance the system's robustness and usability.

5.2 Achievements

The HalalScan system was successfully developed as a mobile-friendly web application that enables users to scan or upload images of import food ingredient lists to determine their halal status. The system uses Optical Character Recognition (OCR), specifcally the Tesseract engine, to extract text from the scanned or uploaded images. These extracted ingredients are then matched with pre-existing database that categorizes ingredients as Halal, Haram, or Doubtful with status indicators shown using intuitive colour coding.

The interface design prioritizes simplicity and accessibility, featuring a homepage with introductory information, a "Get Started" function that leads users to the camera or upload features, and a graph displaying the most frequently scanned ingredients. Additionally, an admin panel was implemented for managing the ingredient database, including functions for creating, updating, and deleting data. The admin systems also integrate with the scan log database to support the generation of weekly and monthly trend graphs.

5.3 Limitations

Figure 31. OCR failure due to shape of the packaging

Figure 32. OCR failure due to reflective packaging

Despite its core functionality being implemented, several limitations were identified during testing process. The system relies on Tesseract OCR to identify ingredients from scanned or uploaded images, but performance was affected by various environmental and design factors such as lighting

conditions, blur image, reflections from glossy packaging as shown from Figure 32, unclear fonts often reduced text recognition accuracy and the curve shape of cylindrical containers like jars or cans as illustrated in Figure 31. These conditions often distorted text alignment and led to partial or incorrect text extraction.

Furthermore, OCR inaccuracies occasionally caused mismatches with existing ingredient database. Ingredients present in the database were sometimes not detected due to misreading or character substitution errors. Such mismatches hindered the system's ability to display correct status results. Moreover, some ingredients that did exist in the database were correctly detected textually but were not mapped or displayed as clickable items in the result interface. This is likely due to UI rendering mismatches or failure to associate against the text span with a database entry despite correct recognition. The overall estimated matching accuracy between OCR results and the database entries was approximately 70%.

These limitations indicate that while the system functions as intended under ideal conditions, its reliability decreases when faced with less-than-optimal image input quality. Addressing these challenges will be crucial in future enhancements, specially to improve detection robustness and OCR precision.

5.4 Future Work

To further enhance the capabilities and usability of HalalScan, several potential improvements are identified for future development. While the current system utilizes the Tesseract OCR engine for text recognition, its performance can be affected by image quality, lighting, font types, and packaging reflections. Future improvements could include integrating more advanced OCR techniques or preprocessing methods (such as noise reduction, contrast adjustment, and text alignment correction) to improve the extraction accuracy, especially for ingredients with stylized fonts or low contrast.

Furthermore, the system primarily supports English ingredient lists at present. To expand its global applicability, future versions should support multilingual OCR processing. This would allow users from different linguistic backgrounds to scan ingredient lists in their native language, improving accessibility and usability. This aligns with the concept of designing software in a way that makes it easy to adapt to various languages and regions without requiring significant engineering changes.

5.5 Summary

This chapter presented the conclusion of the HalalScan system development. Detailing the accomplishments achieved, limitations encountered, and possible directions for future work. Overall, the system successfully meets its objective of helping users identify the status of a product on ingredient level, contributing to informed consumer decisions. Given future enhancements OCR reliability, security measures, and user experience, HalalScan is well-positioned to become a widely adopted tool among halal-conscious consumers.

References

Adjetey, C., & Adu-Manu, K. S. (2021). Content-based Image Retrieval using Tesseract OCR Engine and Levenshtein Algorithm. International Journal of Advanced Computer Science and Applications, 12(7).

Ahmad, J., Widayatmoko, W., & Taib, F. M. (2024). Enhancing Malaysia's Halal brand identity: Insights and strategies for sustainable Global Halal Hub. Kajian Malaysia, 42(2), 73–94.

Aniqoh, N. a. F. A., & Hanastiana, M. R. (2020). Halal food industry: challenges and opportunities in Europe. *Journal of Digital Marketing and Halal Industry*, *2*(1), 43–54. https://doi.org/10.21580/jdmhi.2020.2.1.5799

Ariffin, M. F. M.., Riza, N. S. M.., Hamid, M. F. A.., Awae, F.., & Nasir, B. M. (2021). Halal Food Crime in Malaysia: An Analysis On Illegal Meat Cartel Issues. *The Journal of Contemporary Issues in Business and Government*, *27*(2), 1407–1412. Retrieved from https://cibgp.com/au/index.php/1323-6903/article/view/1045

Asa, R. S. (2019). ILLEGAL LABELING AND THE ABUSE OF HALAL CERTIFICATE: CASE STUDY OF MALAYSIA. Jurnal Syariah, 27(2), 367–388. https://doi.org/10.22452/js.vol27no2.7

Choi, Y., & Jeong, J. (2019). The determinants of imported food purchase of Muslim consumers in Malaysia. Journal of Islamic Marketing, 11(6), 1539–1556. https://doi.org/10.1108/jima-12-2018-0228

Hashim, S. F. M., Salim, J., Noah, S. a. M. N. M., & Mustapha, W. a. W. (2023). Ontology-Based Traceability System for halal status of flavour. Deleted Journal, 65–77.

Ingredient Checker - Warees Halal Network. (2021). Warees Halal Network.

Ismail, M. A. M., Hayub, S., Yahaya, M. H., Jamal, I. H., Abd Wakil, M. N., Rifin, K. I., & Nasri, M. S. (2024). The Adoption of Halal App Among Young Muslim Consumer: A Study on ACIS Student, UiTM Shah Alam. *al-Qanatir: International Journal of Islamic Studies*, *33*(3), 445-457. https://www.al-qanatir.com/aq/article/view/887/553

JAKIM. (n.d.). Semakan status halal. https://www.islam.gov.my/ms/hukum-undangundang/semakan-status-halal

Kabir, M., Kabir, M. R., & Islam, R. S. (2024). Islamic Lifestyle Applications: Meeting the Spiritual Needs of Modern Muslims. arXiv preprint arXiv:2402.02061.

Microsoft (2024.). Recommendations for optimizing data performance - microsoft azure well-architected framework. Microsoft Azure Well-Architected Framework | Microsoft Learn.

Noor, N. M., Azmi, N. a. N., Hanifah, N. A., & Zamarudin, Z. (2023). Navigating the Halal Food Ingredients Industry: Exploring the present landscape. Halalsphere, 3(2), 32–43.

Patience, O.O., Amaechi, E.M., George, O. and Isaac, O.N (2024) *Enhanced Text Recognition in Images Using Tesseract OCR within the Laravel Framework.* Asian Journal of Research in Computer Science, 17 (9). pp. 58-69. ISSN 2581-8260 https://doi.org/10.9734/ajrcos/2024/v17i9499

Petiwala, F. F., Nawazish, H. N., Shukla, V. K., Sharma, R., & Nanda, I. (2021). Halal Food benefits and challenges: A Mobile Application based technical Review. *2021 International*

Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). https://doi.org/10.1109/icacite51222.2021.9404588

Priyadarshini, A. P. (2024). The impact of user interface design on user engagement. *International Journal of Engineering Research & Technology (IJERT), 13*(3).

Rahmawati, D., Purnamasari, D. N., Sofiana, R., Setiawan, H., Alfita, R., & Nahari, R. V. (2023). Mobile Application For Halal Food Product Information with Geofencing Technology Based On Qr Code. *Deleted Journal*, *2*(2), 258–274. https://doi.org/10.21107/aciel.v2i2.261

Rakhmawati, N. A., Fatawi, J., Najib, A. C., & Firmansyah, A. A. (2019). Linked open data for halal food products. *Journal of King Saud University - Computer and Information Sciences*, *33*(6), 728–739. https://doi.org/10.1016/j.jksuci.2019.04.004

Razak, S. F. A., Lee, C. P., Lim, K. M., & Tee, P. X. (2019). Smart halal recognizer for muslim consumers. *Indonesian Journal of Electrical Engineering and Computer Science*, *14*(1), 193. https://doi.org/10.11591/ijeecs.v14.i1.pp193-200

Rosyadi, A. W., Ma'shumah, N. S., Zaman, N. M. Q., & Fajar, N. M. R. (2024). Ingredients identification through label scanning using PaddleOCR and ChatGPT for information retrieval. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *8*(6), 758–767. https://doi.org/10.29207/resti.v8i6.6119

Serunai Commerce Sdn Bhd. (2023). Verify HalalTM - Serunai Commerce. Serunai Commerce.

Sukarno, I., Ramadhita, S. B., & Rachmawati, N. L. (2024). The identification of halal and nonhalal products by utilizing QR code for disaster logistics: Mobile-friendly website application. *AIP Conference Proceedings*, *2891*, 070024. https://doi.org/10.1063/5.0202092

Tarannum, S., Jalal, M. S., & Huda, M. N. (2024). HALALCheck: a Multi-Faceted approach for intelligent halal packaged food recognition and analysis. IEEE Access, 12, 28462–28474.

Tieman, M., & Williams, G. (2019). Creative destruction of halal certification (Bodies) by blockchain technology? *ICR Journal*, *10*(1), 127–13 https://doi.org/10.52282/icr.v10i1.79

Wan Ismail, W. I., Jaiyeoba H.B., MahamadMaifiah, M. H., & Ndzembanteh, A. N (2024). Factors Contributing to the Acceptance of Verify Halal App Among the Malaysian Muslim Consumers. International Journal of Islamic Business, 9(1), 39-55. https://doi.org/10.32890/ijib2024.9.1.3

Ying, W. X., & Zaaba, Z. F. (2022). MyHalal: an interactive application. *Bulletin of Electrical Engineering and Informatics*, *11*(5), 2406–2417. https://doi.org/10.11591/eei.v11i5.3808

Appendices

Appendix A: Full Code for Ingredient Matching

```
known_names = [ingredient.name.lower() for ingredient in Ingredient.query.all()
if ingredient.name]

    for phrase in tokens:
        if phrase in seen:
            continue
        match = next((k for k in known_names if k in phrase), None)
        if match:
            ingredient = get_ingredient_details(match)
            if ingredient:
                ingredients.append(ingredient)
                insert_scan(ingredient['name'])
                seen.add(match)
            continue
        fuzzy_result = process.extractOne(phrase, known_names, score_cutoff=85)
        if fuzzy_result:
            match_name, score = fuzzy_result[:2]
            if match_name and match_name not in seen:
                ingredient = get_ingredient_details(match_name)
                if ingredient:
                    ingredients.append(ingredient)
                    insert_scan(ingredient['name'])
                    seen.add(match_name)

    for word in words:
        if word not in seen and len(word) > 2:
            ingredient = get_ingredient_details(word)
            if ingredient:
                ingredients.append(ingredient)
                insert_scan(ingredient['name'])
                seen.add(word)

    ecodes = re.findall(r'\b(?:e|ins)?\s*\d{3,4}\b', text.lower())
    for code in ecodes:
        code_num = re.search(r'\d{3,4}', code).group()
        if code_num not in seen:
            ingredient = get_ingredient_details(code_num)
            if ingredient and ingredient['ecode'] == code_num:
                ingredients.append(ingredient)
                insert_scan(ingredient['name'])
                seen.add(code_num)
    return ingredients
```

Appendix B: JavaScript Frontend Full Code for Highlighting Detected Ingredients and E-codes

```
ingredients.forEach(ing => {
    const name = ing.name?.toLowerCase();
    const status = ing.status || 'unknown';
    const explanation = ing.explanation || 'No explanation';
    const ecode = ing.ecode || '';
    const category = ing.category || '';
    const color =
      status === 'halal' ? 'green' :
      status === 'haram' ? 'red' :
      status === 'syubhah' || status === 'doubtful' ? 'orange' : 'gray';

    if (name && !matched.has(`name-${name}`)) {
      const nameRegex = new RegExp(`\\b${name.replace(/[-/\\^$*+?.()|[\]{}]/g,
      '\\$&')}\\b`, 'gi');
      text = text.replace(nameRegex, (match) => {
        const placeholder = `@@MATCH_${counter}@@`;
        const span = `<span class="ingredient-click"
                        style="background-color:${color}; color:white;
                         padding:4px 10px; border-radius:20px; margin:3px;
                         display:inline-block; text-decoration:none;
                         cursor:pointer"
                        data-name="${ing.name}"
                        data-status="${status}"
                        data-ecode="${ecode}"
                        data-category="${category}"
                        data-explanation="${explanation}">${match}</span>`;
        placeholderMap.set(placeholder, span);
        counter++;
        matched.add(`name-${name}`);
        matchedIngredients.add(ing);
        return placeholder;
      });
    }

    if (/^\d{3,4}$/.test(ecode)) {
      const eVariants = [`e${ecode}`, `E${ecode}`, `ins${ecode}`, `INS${ecode}`,
      `${ecode}`];
      eVariants.forEach(variant => {
        const regex = new RegExp(`\\b${variant}\\b`, 'gi');
        text = text.replace(regex, (match) => {
          const key = `ecode-${ecode}-${match}`;
          if (matched.has(key)) return match;
          matched.add(key);
```

```
        const placeholder = `@@MATCH_${counter}@@`;
        const span = `<span class="ingredient-click"
                        style="background-color:${color}; color:white;
                        padding:4px 10px; border-radius:20px; margin:3px;
                        display:inline-block; text-decoration:none;
                        cursor:pointer"
                        data-name="${ing.name}"
                        data-status="${status}"
                        data-ecode="${ecode}"
                        data-category="${category}"
                        data-explanation="${explanation}">${match}</span>`;
        placeholderMap.set(placeholder, span);
        counter++;
        matchedIngredients.add(ing);
        return placeholder;
      });
    });
  }
});
```

Appendix C: Full Code for Halal Status Summary

```
const count = { halal: 0, haram: 0, doubtful: 0 };
  matchedIngredients.forEach(ing => {
    const status = (ing.status || '').toLowerCase();
    if (status === 'halal') count.halal++;
    else if (status === 'haram') count.haram++;
    else if (status === 'doubtful' || status === 'syubhah') count.doubtful++;
  });

  const totalIngredients = count.halal + count.haram + count.doubtful;

  let summaryStatus = '';
  let summaryColor = '';
  if (count.haram > 0) {
    summaryStatus = 'Not Halal (contains haram ingredients)';
    summaryColor = '#f44336';
  } else if (count.halal === totalIngredients) {
    summaryStatus = 'Halal (all ingredients are halal)';
    summaryColor = '#4caf50';
  } else if (count.halal > count.doubtful) {
    summaryStatus = 'Most likely Halal – but please verify with the
      manufacturer.';
    summaryColor = '#4caf50';
  } else if (count.doubtful > count.halal) {
    summaryStatus = 'May not be Halal – check with the manufacturer.';
    summaryColor = '#ff9800';
  } else {
    summaryStatus = 'Halal status uncertain – please verify manually.';
    summaryColor = '#9e9e9e';
  }

  const summaryBox = `
      <div style="background:${summaryColor}; color:white; padding:15px; border-
      radius:10px; font-size:1.1rem; margin-bottom:20px; font-weight:bold; text-
      align:center;">
      ${summaryStatus}
    </div>`;

  placeholderMap.forEach((value, key) => {
    text = text.replace(key, value);
  });
  resultText.innerHTML = text;
  resultSection.innerHTML = summaryBox + resultText.outerHTML;
}
```

Appendix D: HalalScan's Usability Testing Form

1) Age Range
- 13 - 18
- 18 - 24
- 25 - 34
- 35 - 44
- 45+

2) Question 2: Occupation/Role
- High school student
- University student
- Working
- Parent / Homemaker

3) Experience with similar app
- Yes
- No

4) Scenario-bases Tasks

Scenario	Success		I can do without guidance	
	Yes	No	Yes	No
Scan ingredient list using camera				
Upload image from device				
Click the ingredient status				
Navigate back to the homepage				

5) Post-test questionnaire

Statement	Rating (1 – Strongly Disagree, 5 – Strongly Agree)				
	1	2	3	4	5
The system was easy to use.					
I was able to complete tasks without much help.					
The result display was easy to understand.					
I would use this app again if I needed to check ingredients.					
I felt confident navigating through the system.					

Appendix E: HalalScan complete source code

You can access the full source code and documentation at the following GitHub repository: https://github.com/mishazl/HalalScanFYP