APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 7 No. 1 (2026) 314-333
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Design and Development of Vegetarian Food Identifier Application

Chow Saw Chun Hwa[1], Norfaradilla Wahid[1]*

[1] *Fakulti Sains Komputer dan Teknologi Maklumat,*
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

*Corresponding Author: faradila@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2026.07.01.017

Article Info

Received: 10 June 2025
Accepted: 18 June 2026
Available online: 30 June 2026

Keywords

Vegetarian, Food Classification, Community Platform, Android Mobile Application, Artificial Intelligence, Optical Character Recognition, Deep Learning

Abstract

Vegetarianism, often practiced for religious or health reasons, is frequently overlooked by businesses, making it difficult for vegetarians to verify and obtain pre-packaged foods and access shared knowledge. This study aims to develop an Android application that classifies pre-packaged food products as vegetarian or non-vegetarian using Optical Character Recognition (OCR) and deep learning, while also supporting community engagement through knowledge sharing. The system follows the Waterfall model with Royce's iterative feedback, utilizing React Native, Laravel, MySQL, and FastAPI in a microservice architecture; OCR extracts text from ingredients list, which are processed by a deep learning AI model. Results show the application successfully simplifies food verification and supports user interaction through posting, commenting, and liking features. While the application achieves its objectives, limitations include Android-only support, UI simplicity, and OCR sensitivity; future work may include iOS or web versions, dynamic label management, and improved AI model accuracy through domain-specific fine-tuning.

1. Introduction

A vegetarian is a person who practices vegetarianism, which is the practice of abstaining from the consumption of meat or all by-products of animal slaughter. There was research into vegetarianism focusing on potential nutritional deficiencies. But in recent years, there have been studies with opposite results to traditional research, which confirm the health benefits of meat-free eating [1]. Vegetarian, when used as an adjective, can be a description of something that aligns with vegetarianism. Even though vegetarianism has health benefits, nowadays, non-vegetarians, which are mainly omnivores, are still significantly more dominant than vegetarians in the world population. Since vegetarians are the minority in the population of most countries in the world, they may face some inconveniences in their daily life, especially in food.

Vegetarians often face challenges in finding and purchasing suitable vegetarian food in the market, especially in pre-packaged products. Vegetarians often lack a dedicated platform to share ideas, recipes, and news, and connect with others in their community. They also face limited access to food options that align with their dietary preferences. The proposed system will eliminate the manual work with a more convenient solution in assisting vegetarians in reviewing ingredient lists. It will also provide a platform for knowledge sharing within the community.

The project has three main objectives. The first objective is to analyse and design a mobile application that allows users to identify and classify food products into vegetarian or non-vegetarian based on ingredients extracted using Optical Character Recognition (OCR) and providing users with a platform for vegetarian knowledge and recipes sharing. Besides that, the project aims to develop an application which has a capability to

categorise foods based on ingredients, and vegetarian knowledge and recipes sharing. The third objective is to perform testing on the functionalities and usability of the system.

At the end of the project, a mobile application for vegetarians is developed. The application simplifies the identification of vegetarian or non-vegetarian food products by analysing ingredient list images. It saves users' time by eliminating the need to manually read and interpret complex or foreign-language ingredient labels. The application provides quick, accurate summaries and helps users overcome language barriers. In addition, users can create groups or communities for sharing knowledge and information relevant to their interests within the vegetarian community. Administrators are provided with an interface to manage the communities created by users.

The paper is organized into five sections. The first section is the introduction of the project background, including the problem statement, objectives and expected results. The next section is Section 2, which will discuss the similar existing systems with the proposed system, and the technology used in the proposed system. Then, Section 3 describes the methodology used and the activities of each phase. In Section 4, the system analysis and design are described, followed by the implementation and testing of the system. Lastly, Section 5 provides a brief conclusion for the paper, with the limitations and future works.

2. Related Work

To propose a mobile application for vegetarian food identifier with the right features, it is necessary to understand a few similar applications as the benchmark to the newly proposed application. This section will discuss three existing applications. An overview of the technology used for the development of the proposed system will also be described in the section. The technology used for the ingredients analysis after text extraction encompasses inferencing using deep learning [2]. During analysis, similar meaning or related words with different variations will be considered rather than relying on static comparison. Table 1 shows the sample of keywords with variations.

Table 1 Sample of Keywords

Keyword	Variant of Keyword
Beef	Steak
Pig	Pork
Fish	Salmon

2.1 Study on Similar Applications

There are three similar existing systems selected, i.e., Yuka, Peanut and Pepper, and compared with the proposed system.

Yuka is a mobile application for Android and iOS that deciphers product labels and analyses the health impact of food products and cosmetics [3]. It aims to improve consumer health by helping to make sense of product labels and promoting healthier choices. The system emphasizes the predefined result on specific products based on results stored in the database. The system requires users to scan the barcode of the product to uniquely identify the selected product, and its rating will be displayed. The system may give ratings like Bad, Poor, Good, and Excellent as guidelines for users for its health level, instead of vegetarian classification.

Besides that, Peanut is a mobile application for Android and iOS, exclusively for women, that has a vision to connect women at every life stage [4]. The platform sets out to reduce feelings of isolation and empower women to connect, all over the world. The application enables users to join support groups tailored to various stages, allowing them to share advice and experiences. Users are given multiple communication options, including chat or video calls to discuss relevant topics privately. Moreover, the application enhances community connection by connecting users with others within their geographical area, creating opportunities for meetups and more social interactions or activities.

In addition, Pepper is a mobile application for Android and iOS, with mission to improve the cooking experience [5]. The application allows users to share their recipes with the community. The application allows users to share images of foods and recipes with the community. Other users can engage with the recipes by leaving reactions and comments. Users have the option to share cooking videos within the application, allowing them to showcase their culinary skills and provide helpful cooking tips to the community.

As result of comparison, the proposed system considers the characteristics and advantages of the existing systems with modifications as improvements, and tailor it for solving the problems. The idea of identifying food products using camera as input in Yuka is modified to capturing ingredients text using camera, followed by text extraction using OCR. The ideas of community creation and information sharing by users in Peanut and Pepper are customized for vegetarians.

Instead of depending on a static database, the proposed system utilizes AI-driven classification based on the extracted text to identify food products more intelligently. Relying solely on a static database is often ineffective, as it may not cover less common food items.

The summary of the systems is presented in Table 2 to facilitate comparison.

Table 2 Comparison of Similar Systems with Proposed System

	Yuka	Peanut	Pepper	Proposed System
Area of Focus	Ingredients Analysis	Community	Recipe	Ingredients Analysis, Community, Recipe
Platform	Mobile Application	Mobile Application	Mobile Application	Mobile Application
User Management	✓	✓	✓	✓
Image Scanning	✓	-	-	✓
Automatic Ingredients Checking	✓	-	-	✓
Ingredients Checking using Artificial Intelligence (AI)	✗	-	-	✓
Classification into Vegetarian	✗	-	-	✓
Multi-Languages Checking	-	-	-	✓
Record Keeping	✓	-	-	✓
New Community Creation	-	✗	-	✓
Community Content Moderation	-	✓	-	✓
Information Sharing	-	✓	-	✓
Vegetarian Food Only	-	-	✗	✓
Recipe Sharing	-	-	✓	✓
Video Sharing	-	-	✓	✗

2.2 Technology Used

The proposed system leverages several technologies for efficient and effective development, including React Native, Laravel, FastAPI, MySQL, Optical Character Recognition (OCR), Deep Learning, and Microservice. React Native is primarily used to develop the user interface and its associated logic. On the backend, Laravel handles server-side functionality, while MySQL serves as the relational database. OCR is integrated to process and convert images of text into machine-readable format for text extraction, and then be processed using Deep Learning, which is exposed using FastAPI and designed based on microservice architecture.

React Native is an open-source cross-platform mobile application development framework, developed and maintained by Facebook (Meta) since 2015. Since it is designed and developed based on React.js, it uses the same programming language as React.js, which is JavaScript. React Native is responsible for translating JavaScript into the native language of the specific platform for implementation.

Laravel is an open-source PHP framework, commonly used for web development. It can be considered as among the top PHP frameworks for web development. Laravel follows the Model-View-Controller (MVC) architecture, enabling it as a robust full-stack framework. Furthermore, it is well-suited for API development, particularly RESTful API, which is increasingly popular.

2.2.1 Optical Character Recognition

Optical Character Recognition (OCR) is a technology that converts images into machine-readable text format. Once the text is detected, it is extracted and transformed into computerised text. OCR is highly efficient for digitising text from real-world sources, whether printed or handwritten, offering significant efficiency in

processing and storing data. The OCR is implemented in Laravel server by receiving and processing image from client-side.

2.2.2 Deep Learning

Deep Learning is a branch of Artificial Intelligence (AI) that uses neural networks to solve complex problems across various domains, including Natural Language Processing (NLP) [2]. It involves different approaches such as training new models, fine-tuning pre-trained models, and using pre-trained models for inference. Leveraging a pre-trained deep learning model for inference enables efficient and accurate analysis of data without the need for additional training.

Microservice is an architectural style where an application is composed of small, independent services that are loosely coupled and focused on specific business functions [6]. These services communicate through lightweight APIs and can be developed, deployed, and scaled independently. This approach enhances flexibility and scalability, allowing organizations to innovate more quickly and maintain systems more efficiently than with traditional monolithic architectures.

FastAPI is a modern Python web framework designed specifically for building APIs. It is widely used due to its high performance, ease of use, and clean, modern design. FastAPI also supports asynchronous programming, enabling it to handle multiple requests at the same time, which greatly improves efficiency and overall performance.

The deep learning model used to classify food ingredients as vegetarian or non-vegetarian is a pre-trained model that was deployed using FastAPI [7]. Rather than calling an external API, the systems run the model in-house to perform inference on input data. FastAPI exposes the model as a RESTful API, allowing it to communicate effectively with the main Laravel-based system. This design follows a microservice architecture, where the inference service is loosely coupled, independently deployable, and focused solely on classification tasks. This setup enables modular development, easier maintenance, and better scalability.

3. Methodology

The methodology is a structured system including practices, techniques, procedures, and rules that guide professionals within a particular discipline [8]. The professionals are not limited to technical members. It provides a schedule and roadmap for each member's day-to-day tasks planning.

The methodology adopted for this project is the Waterfall Model with Royce's Iterative Feedback [9]. The model consists of seven phases: evaluation, requirements, analysis, design, development, validation, and deployment. The project flow primarily follows a sequential progression. If there is feedback from stakeholders or other reasons that require modification on the previous phase work, refinement or improvement can be done by temporarily working on the previous phase. Refinement on more than one previous phase is allowed instead of rigid like the traditional waterfall model, promoting greater flexibility and adaptability. The methodology is illustrated as shown in Figure 1.

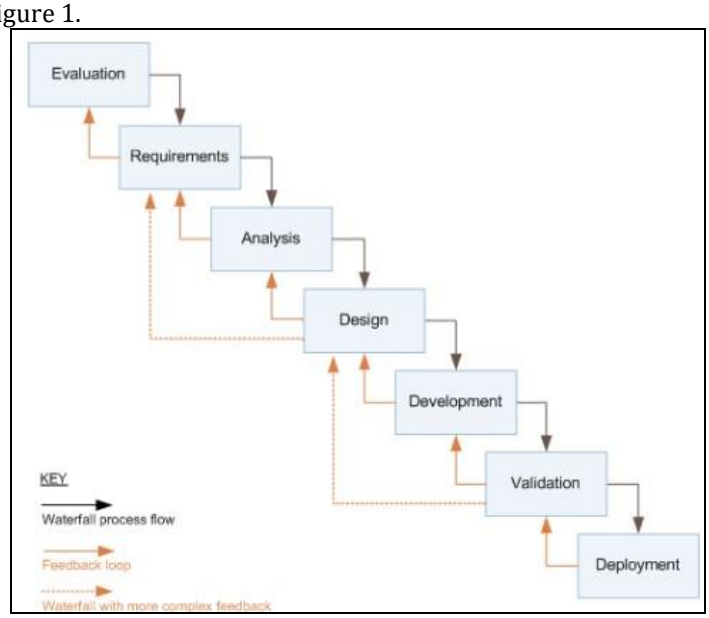


Fig. 1 Waterfall Model with Royce's Iterative Feedback [9]

During the evaluation phase, the project's objectives, feasibility, and constraints were evaluated to ensure the project can solve real-world problems faced by the target users, that is vegetarian community. The key parameters such as time and resources were considered to ensure the efficiency and effectiveness of the project.

The main elements such as problems, with potential solutions, were identified and analyzed. The planning of the project is summarized and illustrated in the Gantt chart. The Gantt chart is divided into several phases, according to the project methodology model.

During the requirements phase, data gathering was conducted using a survey method to gather the problems faced by the target users and their expectations of the proposed system. A questionnaire was designed using Google Forms and distributed to the public on social media to collect the necessary information.

Based on the requirements and information gathered, the detailed functional requirements were defined. The analysis and design approach applied was the Object-Oriented Analysis and Design approach, which emphasizes Unified Modelling Language (UML). Based on the functional requirements, the behaviour of the system was illustrated using the use case diagram, sequence diagram, and activity diagram while the structure of the system was illustrated using the class diagram.

During the design phase, the database schema was initially planned and illustrated using an Entity-Relationship Diagram (ERD), based on the functional requirements. The detailed database schema, including constraints and the data dictionary of the database, was summarized in a table, with each entity represented in separate table. The database schema outlined the data type of each column, while the data dictionary describes the meaning of each column within the tables. Additionally, the user interfaces of the system were designed as skeletal frameworks.

During the development phase, the primary software or applications used are Visual Studio Code (VS Code), MySQL, Android Studio, and Google Chrome. The technology used for the development is React Native, Laravel, and FastAPI, with programming languages of JavaScript, PHP, and Python respectively. React Native is responsible for the user interface (UI) of the mobile application, while Laravel is responsible not only for the UI of the web application for administrator, but also for almost all the backend logic. The FastAPI is responsible for facilitating communication between the AI model and external system.

During the validation phase, the system's functionalities will be tested to ensure there are no errors. Any significant errors identified will prompt a feedback loop for returning to the previous phase, that is the development phase for resolution. This ensures that the system can operate properly and deliver good experience to users. The backend, developed using Laravel, will be tested independently from the React Native frontend to assess its core functionality before integration and testing with the frontend. Usability testing will be conducted to evaluate user-friendliness and identify areas for improvement.

The backend will be deployed on a cloud server to ensure reliability and accessibility. The deployment will only be carried out at the end of the project, after all the development and testing.

The system development workflow is summarized by phases in Table 3.

Table 3 System Development Workflow

Phase	Activity	Deliverable
Evaluation	1. Study problems faced by vegetarians in daily life, and potential solutions, and consider feasibility. 2. Study on related online resources.	1.Project Proposal 2.Gantt Chart 3.Literature Review
Requirements	1. Design survey questions for identifying problems faced by vegetarians and their expectations in the proposed system.	1. Questionnaire
Analysis	1.Identify functional requirements in solving the problems faced by vegetarians. 2.Illustrate the use case diagram, sequence diagram, activity diagram, and class diagram.	1.Use Case Diagram 2.Sequence Diagram 3.Activity Diagram 4.Class Diagram
Design	1.Design database schema using Entity-Relationship Diagram (ERD), with the main entity of User, Community, Post, Comment, Keyword, and AnalysisResult. 2.Design a data dictionary for the database. 3.Sketch wireframes for user interface (UI), categorized into mobile application and web application.	1.Entity-Relationship Diagram 2.Data Dictionary 3.Wireframe
Development	1.Configure the necessary setup. 2.Develop the system.	1.Android Application 2.Web Application
Validation	1.Perform functional testing and usability testing.	1.Verification Report 2.Usability Report
Deployment	1.Deploy the system to the cloud server.	1.Deployed System

4. Result and Discussion

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 314-333

This section describes the system analysis and design, and implementation of the system, followed by functional testing and user acceptance test. System implementation and testing are important to ensure that the application functions as intended, meets user requirements, and performs reliably under different conditions.

4.1 System Analysis and Design

This section details the system requirements, including both functional and non-functional requirements. It also presents system analysis using Object-Oriented approach, incorporating diagrams such as use case diagram, sequence diagram, and class diagram.

4.1.1 Use Case Diagram

A Use Case Diagram in Unified Modelling Language (UML) is a diagram that illustrates the interactions between actors (users) and the system. The system is primarily represented by its functional requirements, showing the functionalities that users can perform in the system. The actors are classified into two categories: user and administrator. The user role can be further divided into community owner and community member, while the administrator role can be extended to include the super administrator. The use case diagram is illustrated in Figure 2.

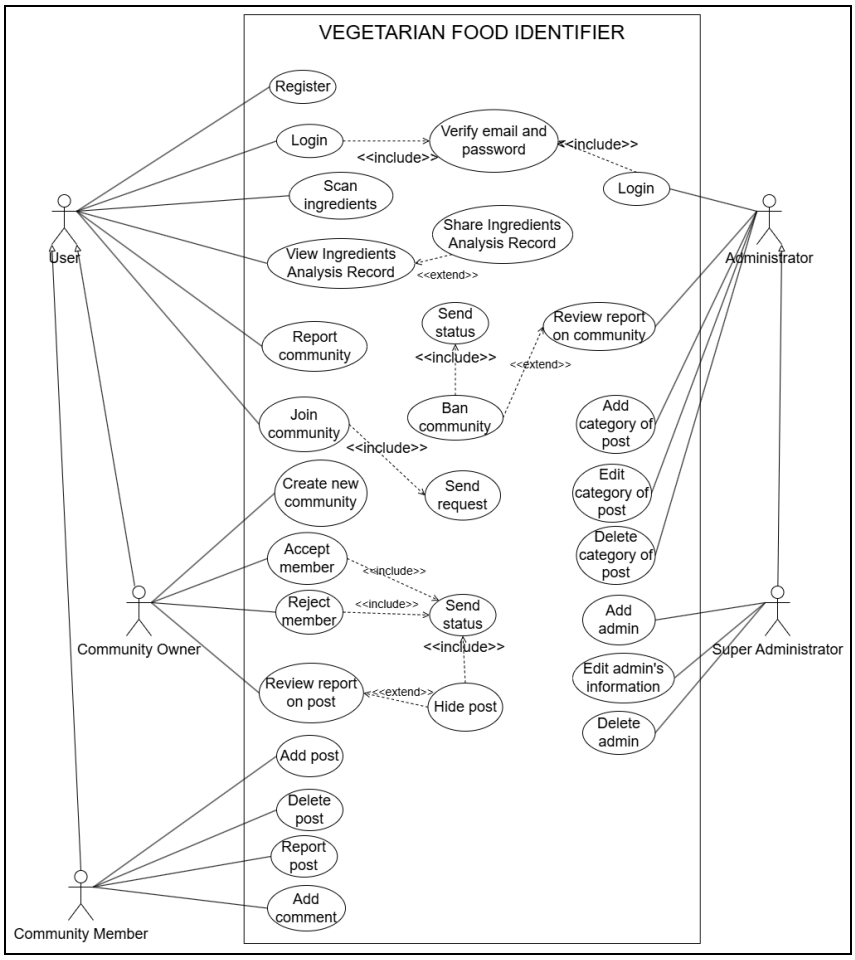


Fig. 2 Use Case Diagram

4.1.2 Sequence Diagram

Sequence Diagram is a component of Unified Modelling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modelling dynamic behaviour in the system [10].

Figure 3 shows the sequence diagram of analyze ingredients. Firstly, the user can upload an image of an ingredients list for analysis to determine if it is vegetarian. The analysis process involves invoking an AI model that exposes the endpoint via FastAPI. The result of the analysis is returned to the Laravel backend. The result of the analysis is then stored in the database by invoking the store() method in the AnalysisResult class. The matched keywords will be stored in the database using the store() method in the AnalysisResultKeyword class.

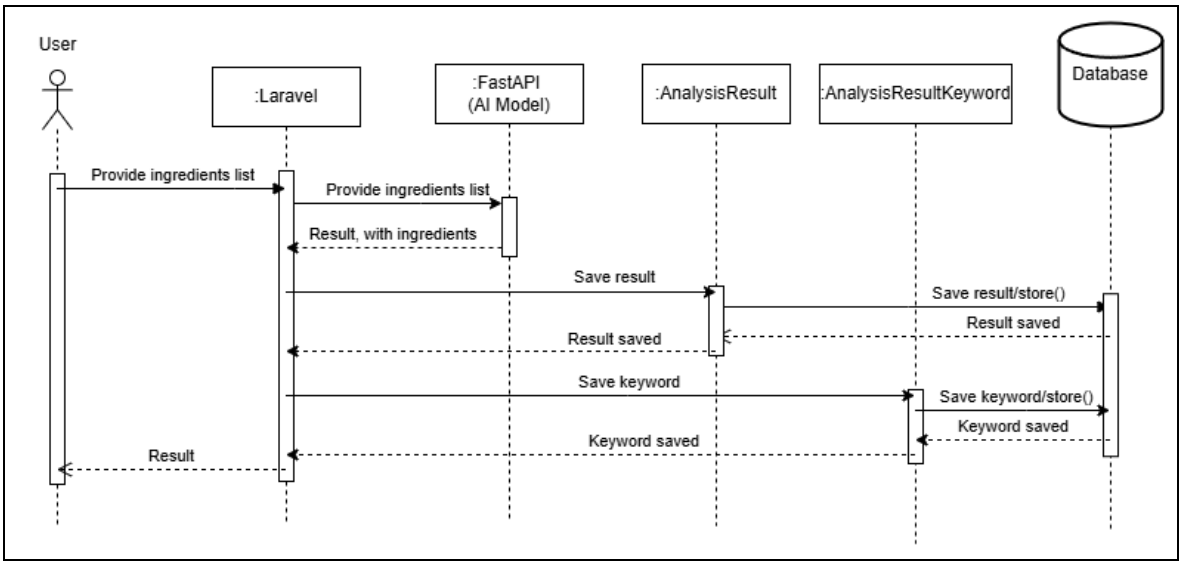


Fig. 3 Sequence Diagram of Analyse Ingredients

Figure 4 shows the sequence diagram of share analysis result. Firstly, a user can view their analysis results using the index() method in the AnalysisResult class and share them by invoking the store() method in the SharedResult class. Other users can view the shared results by using the index() method in the SharedResult class. To like a shared result, a user can invoke the store() method in the SharedResultLike class. If the user wishes to remove their like, they can do so by calling the destroy() method in the SharedResultLike class.

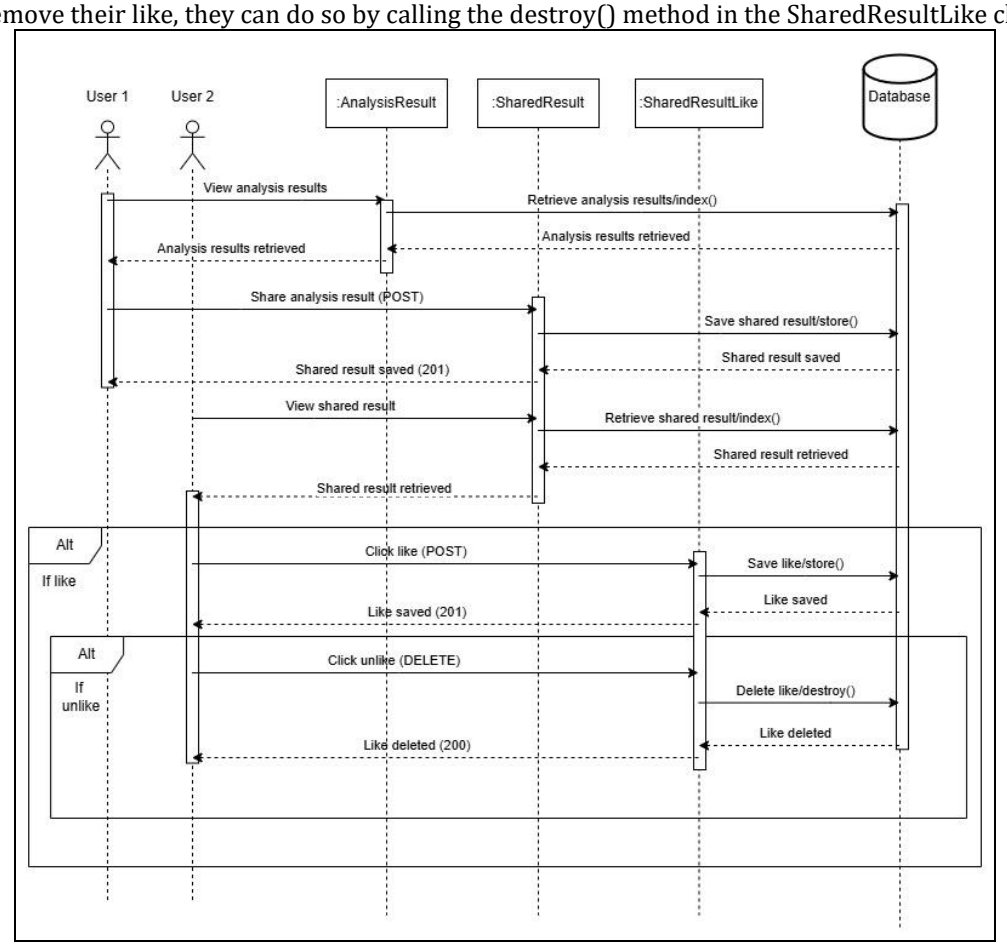


Fig. 4 Sequence Diagram of Share Analysis Result

4.1.3 Database Design

The schema and data dictionary of the database are organised by table. The schema consists of the column names, data types, and constraints, while the data dictionary provides detailed descriptions for each column. There are a total of 16 tables in the system, and only some of them are shown in this paper.

The AnalysisResult table schema defines the structure for storing results of food product analysis. It includes analysis_result_id as the primary key, with user_id as a foreign key referencing the User table. The analysis_result_title field stores the title of the result, serves as reference. The analysis_result_is_vegetarian field indicates whether the food is vegetarian or non-vegetarian, with true for vegetarian and false for non-vegetarian. The analysis_result_cover_image and analysis_result_image fields store images of the food product and its ingredients list, respectively. The created_at field tracks the creation date and time of the record, defaulting to the current timestamp. The summary is as shown in Table 4.

Table 4 Schema and Data Dictionary for AnalysisResult

Column Name	Data Type	Constraints	Description
analysis_result_id	BIGINT	Primary Key	Unique identifier
user_id	BIGINT	Foreign Key	Reference to User table
analysis_result_title	VARCHAR(255)	Null	Title of analysis result
analysis_result_is_vegetarian	BOOLEAN	Not Null	True represents vegetarian, false represents non-vegetarian
analysis_result_image	VARCHAR(255)	Not Null	Image of ingredients list of food product
analysis_result_cover_image	VARCHAR(255)	Not Null	Image of food product
created_at	DATETIME	Not Null, Default(now)	Creation date and time of record

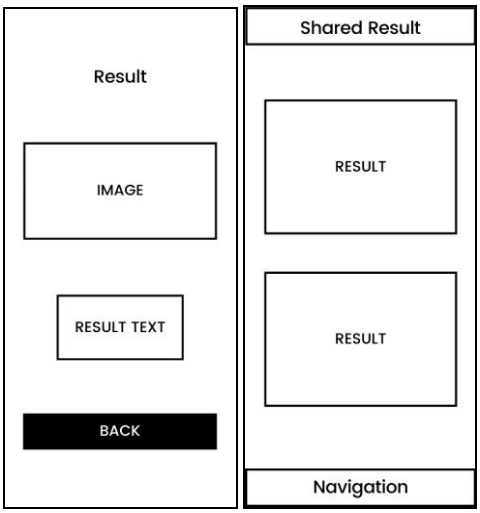
The SharedResult table schema defines the structure for storing shared food analysis results. It includes shared_result_id as the primary key, with analysis_result_id as a foreign key referencing the AnalysisResult table. The created_at field records the creation date and time of the shared result, defaulting to the current timestamp. The summary is as shown in Table 5.

Table 5 Schema and Data Dictionary for SharedResult

Column Name	Data Type	Constraints	Description
shared_result_id	BIGINT	Primary Key	Unique identifier
analysis_result_id	BIGINT	Foreign Key	Reference to AnalysisResult table
shared_result_title	VARCHAR(255)	Not Null	Title of the shared result
created_at	DATETIME	Not Null, Default(now)	Creation date and time of record

4.1.4 User Interface Design

The user interfaces (UI) of the system are illustrated using low-fidelity wireframes, which provide a clear outline of the content and its respective placement within the interface. Some of the wireframes are shown as in Figure 5.



(a) (b)

Fig. 5 User Interface (a) Interface for Result Scanning; (b) Interface for Shared Result

4.1.5 Class Diagram

A Class Diagram is a type of Unified Modelling Language (UML) diagram that visually represents the static structure of a system by illustrating its classes, attributes, methods, and the relationships among objects. It serves as a blueprint for constructing and visualizing object-oriented systems [11]. The class diagram consists of 17 classes, each defined by its attributes and methods, and interconnected through various associations. Each class has its own attributes and necessary methods for business logic and data retrieval. The class diagram is illustrated as shown in Figure 6.

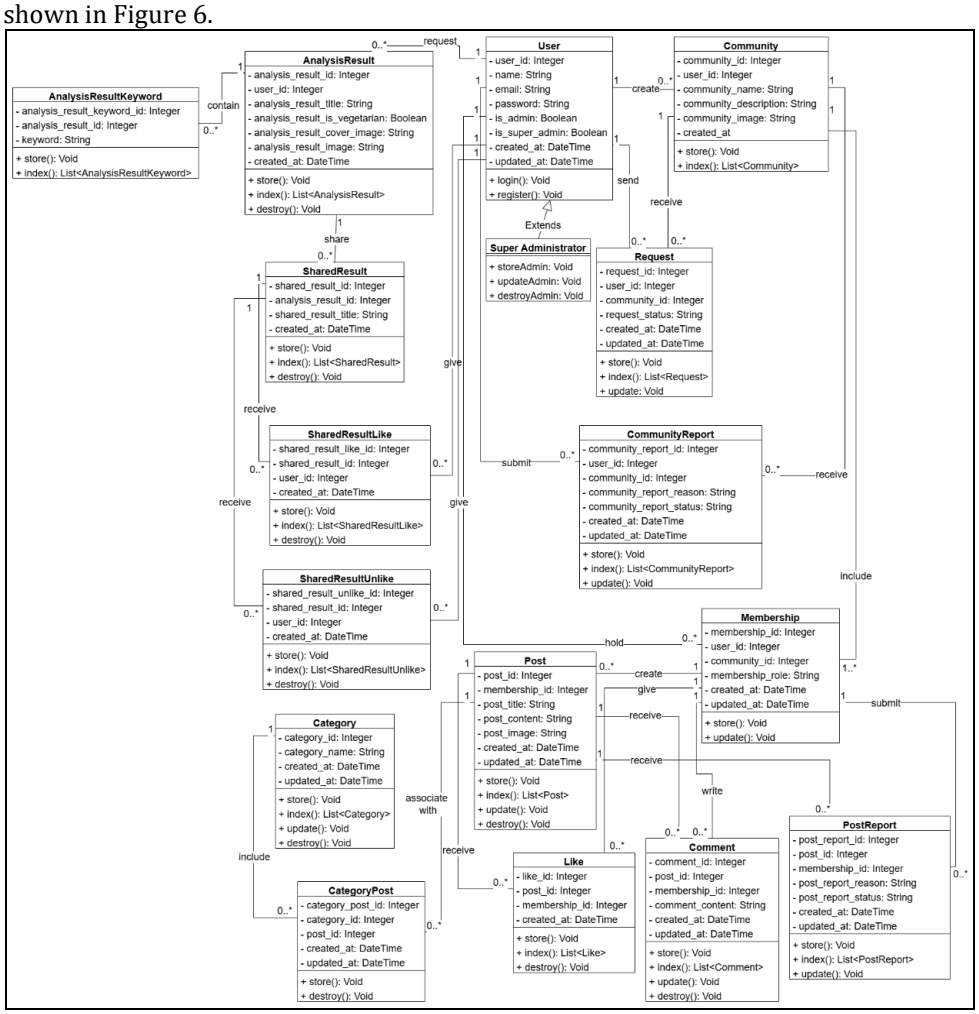


Fig. 6 Class Diagram

4.2 Implementation of System

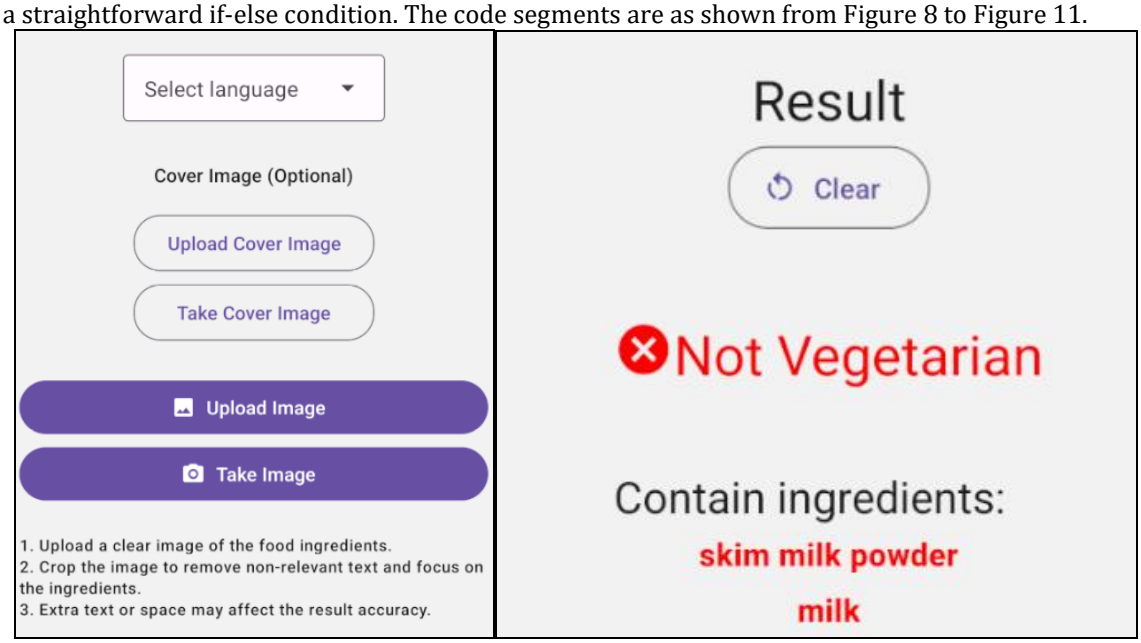
The system is developed using a combination of frontend and backend technologies. The mobile application is built with React Native, utilizing the Material Design components provided by React Native Paper for the user interface. The frontend acts as the client side and communicates with the Laravel backend through API calls. Laravel serves as the primary backend and facilitates communication with a pre-trained AI model via a FastAPI endpoint. FastAPI is implemented as a microservice to handle the AI-related functionality. In addition, Laravel is used to develop a full-stack web application for administrators. The website's frontend is styled using the Bootstrap CSS framework.

4.2.1 Ingredients Checking

Users are required to select the language of the ingredients text, upload the image of the ingredients label on food product, and optionally upload an image of the product itself. After analysis, the results are displayed, highlighting any non-vegetarian ingredients. The user interfaces are illustrated as shown in Figure 7.

On the server side, Optical Character Recognition (OCR) is used to extract text from the uploaded ingredients image. The extracted text is then cleaned and passed to a microservice containing an AI model, which performs inference to determine whether the ingredients are vegetarian or non-vegetarian.

For classification purposes, four candidate labels are predefined, two representing vegetarian categories and two representing non-vegetarian categories. The AI model returns probability scores for each label, and the final decision is made based on these probability values [12]. The probability scores of the two animal-based (non-vegetarian) labels are combined by summing them into a single value. If this combined probability exceeds a threshold of 0.5, then the ingredient is classified as non-vegetarian. The classification logic is implemented using a straightforward if-else condition. The code segments are as shown from Figure 8 to Figure 11.



(a) (b)

Fig. 7 Ingredients Analysis (a) Upload Ingredients Image Interface; (b) Result After Analysing Interface

```
$text = (new TesseractOCR($absolutePath))
        ->lang($language)
        ->run();
```

Fig. 8 Code Segment of Text Extraction using OCR

```
task = "zero-shot-classification"
classifier = pipeline(task, model=model)
```

Fig. 9 Code Segment of AI Model Initialization

```
# labels
label1 = "directly derived from slaughtered animals"
label2 = "indirectly derived from slaughtered animals"
label3 = "plant-based ingredient"
label4 = "not involving animal-based ingredient"

candidate_labels = [label1, label2, label3, label4]

outputs = classifier(normalizedIngredients, candidate_labels, multi_label=False)
```

Fig. 10 Code Segment of Inferencing using AI Model

```
for output in outputs:
    sequence, labels, scores = output.values()

    # index of animal-based labels
    index1 = labels.index(label1)
    index2 = labels.index(label2)

    # score of animal-based labels
    score1 = scores[index1]
    score2 = scores[index2]

    # combine score of both animal-based labels
    probability_not_vegetarian = score1 + score2

    # determine is vegetarian or not
    if probability_not_vegetarian > 0.5:
        result = {"name": sequence, "isVegetarian": False}
        results.append(result)
    else:
        result = {"name": sequence, "isVegetarian": True}
        results.append(result)
```

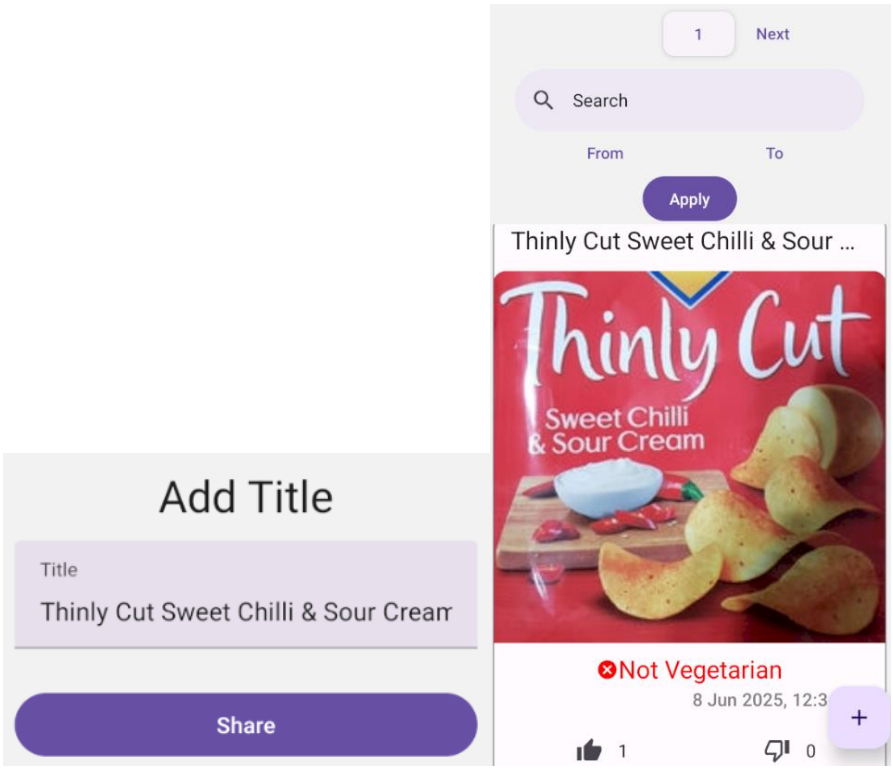
Fig. 11 Code Segment of Deciding Tendency be Vegetarian based on Probability

4.2.2 Result Sharing

Before sharing an ingredients analysis record with others, users are required to provide a title to ensure the shared result is meaningful and understandable to other users. Shared results can be viewed by other users, who may optionally filter them by date or search by title.

Users can express their options on shared results by either liking or unliking them, which indicates a positive and negative view, respectively. However, a user can only either like or unlike a result at any given time, not both. If a user likes a result and later chooses to unlike it, the previous like will be removed automatically. The same logic applies in reverse. The user interfaces are as shown in Figure 12 while the code segment is shown in Figure 13.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 314-333



(a) (b)

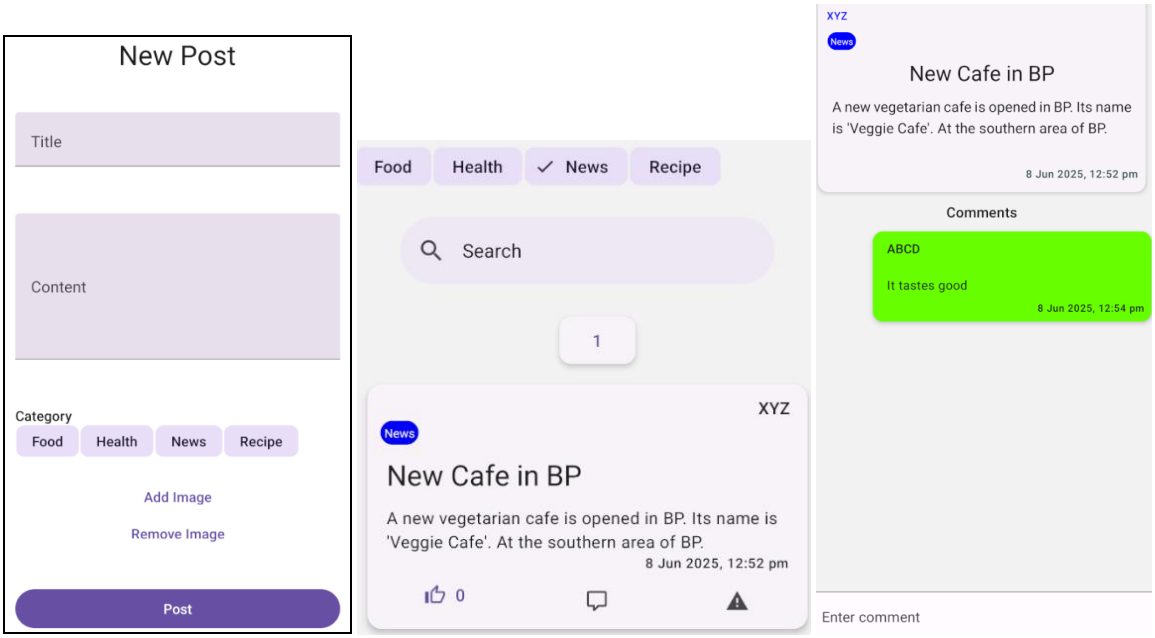
Fig. 12 Result Sharing Interface (a) Share Result Interface; (b) Shared Results Interface

```php
public function showAll($validated){
    $q = $validated['q'] ?? NULL;
    $startDate = $validated['startDate'] ?? NULL;
    $endDate = $validated['endDate'] ?? NULL;

    $query = SharedResult::with(['analysisResult.analysisResultKeywords', 'sharedResultLikes', 'sharedResultUnlikes']);

    if($q){
        $query = $query->where('shared_result_title', 'like', "%{$q}%");
    }

    if($startDate && !$endDate){
        $query->whereDate('created_at', '>=', $startDate);
    }else if(!$startDate && $endDate){
        $query->whereDate('created_at', '<=', $endDate);
    }else if($startDate && $endDate){
        $query->whereBetween('created_at', [$startDate, $endDate]);
    }
    $query = $query->orderBy('created_at', 'desc');
    $sharedResults = $query->paginate(10);
    return $sharedResults;
}
```

Fig. 13 Code Segment of Shared Results

4.2.3 Post in Community

Users can create new posts within a community by providing the required information. Posts are visible to all members of the community. Users can filter the posts by category or search for posts by title within the community. Categories are predefined and managed by the administrator, who can modify them. Additionally, users can comment on and like posts. The user interfaces and code segments are as shown from Figure 14 to Figure 16.



(a) (b) (c)

Fig. 14 Post Interface (a) Create Post Interface; (b) Posts Interface; (c) Comment on Post Interface

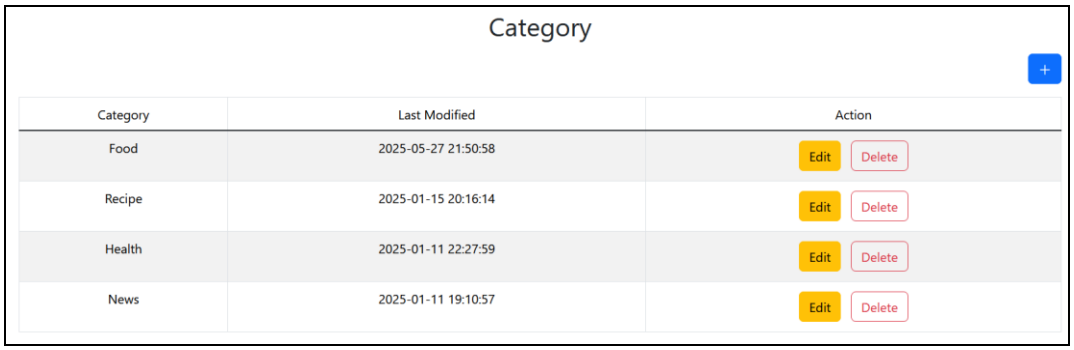


Fig. 15 Manage Categories of Post Interface

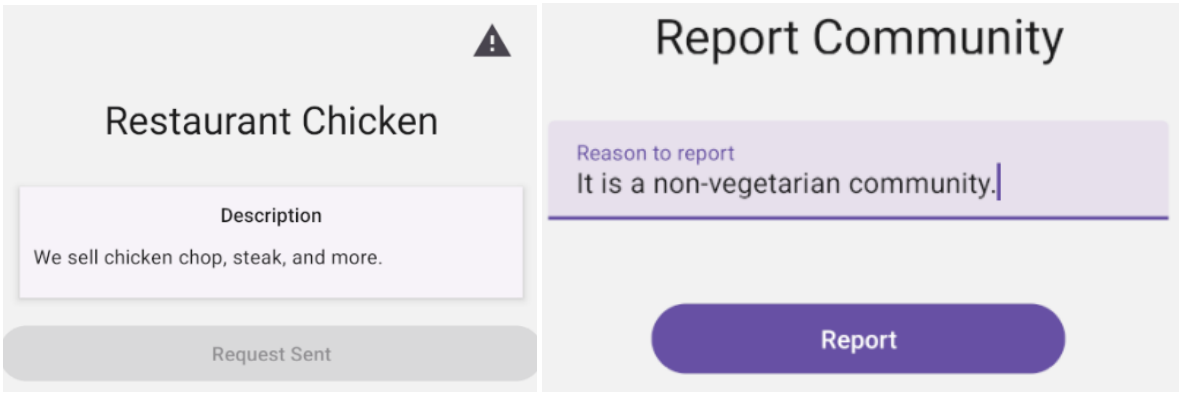
```php
$posts = Post::with(['membership.user', 'likes', 'categoryPosts.category', 'postReports' => function ($query) {
    $query->where('post_report_status', '!=', 'accepted');
}])
->whereHas('membership', function ($query) use ($community_id){
    $query->where('community_id', $community_id);
})
->where(function ($query) {
    $query->doesntHave('postReports')
          ->orWhereHas('postReports', function ($q) {
            $q->where('post_report_status', '!=', 'accepted');
          });
})
->when($category_id, function ($query) use ($category_id) {
    $query->whereHas('categoryPosts', function ($q) use ($category_id) {
        $q->where('category_id', $category_id);
    });
})
->when($q, function ($query) use ($q) {
    $query->where('post_title', 'like', '%' . $q . '%');
})
->orderBy('updated_at', 'desc')
->paginate(10);

return $posts;
```

Fig. 16 Code Segment of Retrieving Posts

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 314-333

4.2.4 Report Community

Users are allowed to report a community even if they have not joined it. When reporting, users are required to provide a reason for the report. The reports will be managed by the administrator, by updating their status to either 'accepted' or 'rejected'. If there are multiple reports submitted for the same community, all related reports are updated simultaneously, preventing the administrator from repeating the process of reviewing on the same community. The user interfaces and code segments are as shown from Figure 17 to Figure 20.



(a) (b)

Fig. 17 Report Community Interface (a) Navigate to Report Interface; (b) Report Community Interface

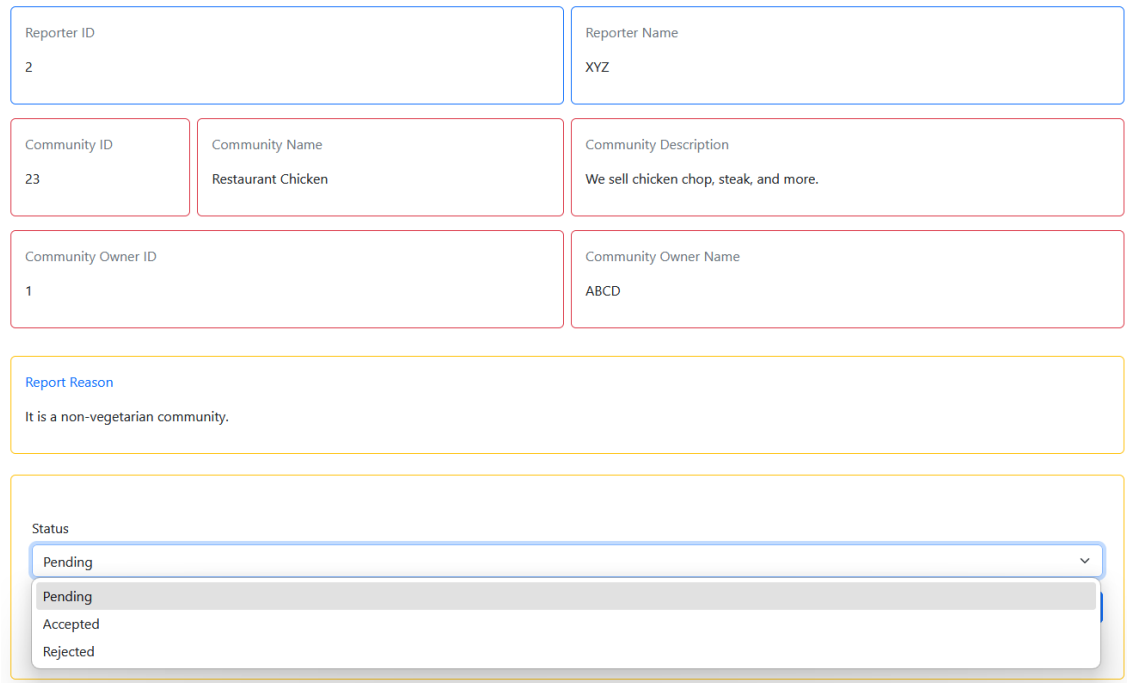


Fig. 18 Manage Community Reports Interface

```php
public function createCommunityReport($auth_id, $community_id, $community_report_reason){
    $existingCommunityReport = CommunityReport::where('user_id', $auth_id)
                                              ->where('community_id', $community_id)
                                              ->where('community_report_status', 'pending')
                                              ->first();

    if($existingCommunityReport){
        return $existingCommunityReport;
    }

    $newCommunityReport = [
        'user_id' => $auth_id,
        'community_id' => $community_id,
        'community_report_reason' => $community_report_reason,
        'community_status' => 'pending',
    ];
    return CommunityReport::create($newCommunityReport);
}
```

Fig. 19 Code Segment of Report Community

```php
public function updateReportStatus($id, $community_report_status){

    $communityReport = CommunityReport::find($id);

    $community_id = $communityReport->community_id;

    // update all reports of the same community
    CommunityReport::where('community_id', $community_id)
                   ->update(['community_report_status' => $community_report_status]);
}
```

Fig. 20 Code Segment of Manage Community Reports

4.3 Functional Testing

Functional testing is carried out module by module, with each module having its own test plan. This testing is essential to ensure that the system behaves as expected. During the process, the actual output is compared against the expected output to determine whether the test passes. Ten randomly selected users participated in the testing. The actual output is observed to check if it matches the expected output. Failure of any actual output to match the expected output for each test case results in fail.

A test plan is a detailed document that defines the scope and approach of testing activities. It outlines the overall testing strategy and identifies the expected results for each test case. The tests were passed, with all outputs behaving as expected. The test plans with their results are shown from Table 6 to Table 8.

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 314-333

Table 6 Testing Results of User Management Module

No	Function	Test Case	Expected Output	Actual Output
1	Register	Incomplete input fields	Error message is displayed if the required field is empty.	Pass
		Unique email address	Error message is displayed if email address was registered.	Pass
		Password confirmation does not match	Error message is displayed if password confirmation does not match the password.	Pass
		Complete input fields	Successfully register an account.	Pass
2	Login	Incomplete input field	Error message is displayed if the required field is empty.	Pass
		Incorrect password	Error message is displayed if the password is incorrect.	Pass
		Complete and correct input fields	Successfully login and redirect to the main screen.	Pass
3	Change Password	Incorrect current password	Error message is displayed if the current password is incorrect.	Pass
		Password confirmation does not match	Error message is displayed if the new password does not match the password confirmation.	Pass
		Complete and correct input fields	Successfully change password, if all the fields correct.	Pass
4	Change Name	Incomplete input data	Error message is displayed if the required field is empty.	Pass
		Complete input field	Successfully change name in database and reflect it immediately in frontend.	Pass

Table 7 Testing Results of Ingredients Module

No	Function	Test Case	Expected Output	Actual Output
1	Ingredients Analysis	Upload image of ingredients	Preview of image and analyzing button are displayed.	Pass
		Perform analysis	Result of analysis displayed to user.	Pass
		Upload cover image	Preview of cover image is displayed.	Pass
		Clear result	Result and images are cleared after clicking clear button.	Pass
2	Historical Record	Filter and search records	Matched historical records of analysis results are displayed.	Pass
		Pagination	Prevent users from reaching pages that do not contain anything.	Pass
		Incomplete title field	Error message is displayed if the title field is empty.	Pass
		Complete title field	Updated title is reflected immediately after modification.	Pass
		Upload cover image	New cover image is reflected immediately after upload.	Pass
3	Result Sharing	Incomplete title field when sharing result	Error message is displayed if the title for shared result is empty.	Pass
		Complete title field when sharing result	Successfully shared result.	Pass
		Filter and search shared results	Matched shared results are displayed.	Pass
		Pagination	Prevent users from reaching pages that do not contain anything.	Pass
		Like or unlike	Successfully like or unlike shared result.	Pass
		Overwrite previous like or unlike	Successfully overwrite previous like, by clicking unlike button, or vice versa.	Pass
		Delete shared result	Successfully deleted shared result.	Pass

Applied Information Technology and Computer Science Vol. 7 No. 1 (2026) p. 314-333

Table 8 Testing Results of Community Module

No	Function	Test Case	Expected Output	Actual Output
1	Community Creation	Incomplete input fields	Error message is displayed if the required fields are empty.	Pass
		Complete input fields	Successfully created a new community.	Pass
2	Join Community	Send request to join	Successfully send requests to join the community.	Pass
		Review requests	Community owner can decide to accept or reject the requests.	Pass
3	Report Community	Incomplete input field	Error message is displayed if the required field of reason is empty.	Pass
		Complete input field	Successfully sent report.	Pass
		Review community reports	Successfully updated status of report, either accepted or rejected.	Pass
		Block community	Blocked community is blocked and hidden.	Pass
4	Post Creation	Incomplete input fields	Error message is displayed if the required fields are empty.	Pass
		Complete input fields	Successfully created post.	Pass
5	Interaction on Post	Like on Post	Successfully like on post.	Pass
		Comment on Post	Successfully commented on post.	Pass
6	Post Filtering	Filter by category	Successfully displayed only matched posts based on category predefined.	Pass
		Manage category of post	Successfully add/edit category of post.	Pass
		Searching	Successfully displayed only matched posts based on search keyword.	Pass
7	Report Post	Incomplete input field	Error message is displayed if the required field of reason is empty.	Pass
		Complete input filed	Successfully sent report.	Pass
		Review post reports	Successfully updated status of report, either accepted or rejected.	Pass
		Block post	Blocked post is blocked and hidden.	Pass

4.4 User Acceptance Test

User Acceptance Test (UAT) is the final phase of software development, where the software is tested by end users to ensure it meets their requirements and to identify any issues that need to be addressed. Ten randomly selected users participated in the testing and completed survey forms to provide their feedback on the system. A Likert scale was used for responses, where 1 indicates 'Strongly Disagree' and 5 indicates 'Strongly Agree' [13]. The results show that the system meets the specified requirements in terms of usability and functionality, with results typically showing that over 50% of users agreeing or strongly agreeing with the system's performance. The results are visualized in percentage as shown from Figure 21 to Figure 23.

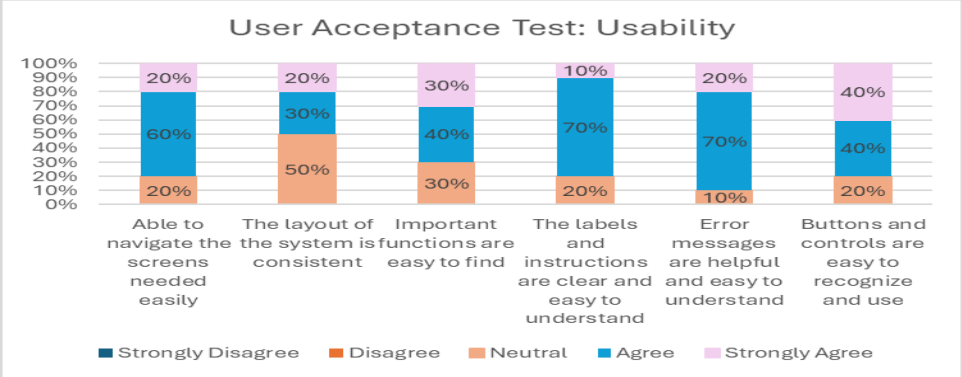


Fig. 21 User Acceptance Test: Usability

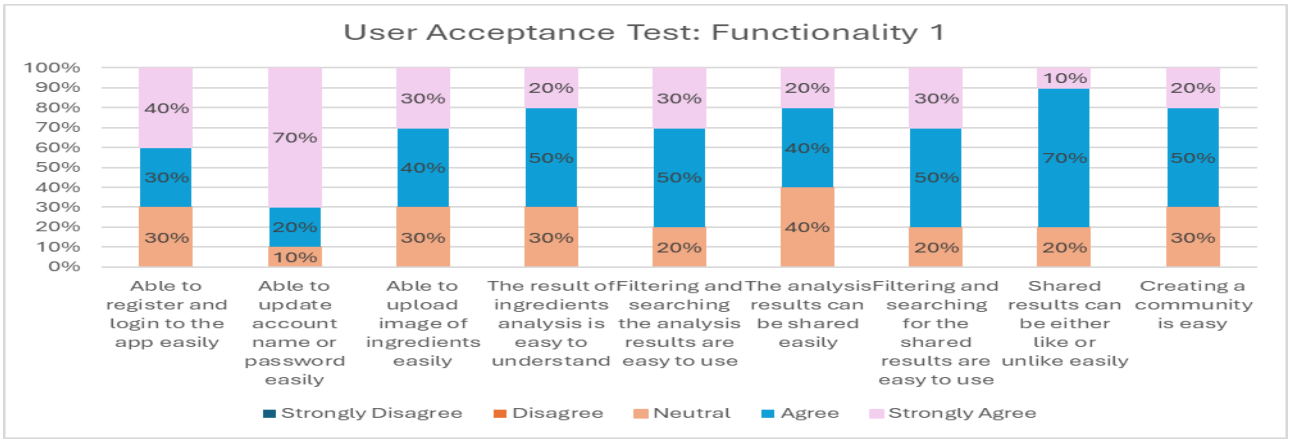


Fig. 22 User Acceptance Test: Functionality Part 1

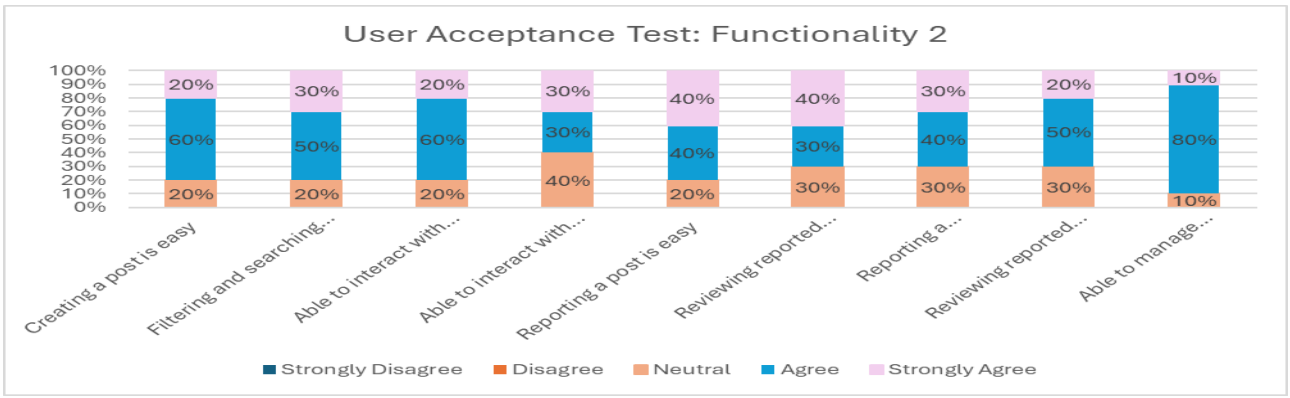


Fig. 23 User Acceptance Test: Functionality Part 2

5. Conclusion

In conclusion, this project has successfully developed a mobile application for Android OS to address key challenges faced by the vegetarian community. These challenges include the manual review of the ingredients list in pre-packaged food products and the lack of a dedicated platform for information sharing among the vegetarian community. By leveraging Optical Character Recognition (OCR) and deep learning-based keyword classification, the system eliminates the need for manual ingredients checking. Users can upload an image of an ingredients list to determine whether a pre-packaged food product is vegetarian or non-vegetarian. Community engagement is also supported through features that allow users to post, comment, and like within the app. On the administrative side, a web application enables efficient management, including moderating reports and defining post categories.

Despite achieving its core objectives, the system has limitations. It currently supports only Android, limiting accessibility for iOS users. The ingredients analysis feature is also limited to languages with clear delimiters like

spaces or commas. The user interface lacks visual appeal due to minimal styling, which may affect user experience. Additionally, OCR accuracy depends on image quality, which can impact ingredients recognition. User acceptance test is conducted to evaluate whether the system meets the specified requirements in terms of usability and functionality, with results typically showing over 50% of users agreeing or strongly agreeing with the system's performance.

To improve accessibility, an iOS or web version can be developed using the existing API-based backend. The backend of the system, developed using Laravel and FastAPI, adopts an API-first and loosely coupled architecture, allowing new frontends such as iOS or web to be added without the need to rebuild the backend, thus ensuring scalability and reusability. The AI model currently relies on a predefined set of labels, but future improvements could allow administrators to manage labels dynamically. With dynamic label editing in the future, administrators will be able to update labels based on new findings or research, enabling more accurate and context-appropriate classifications instead of being limited to the current static set. Fine-tuning the pre-trained AI model with domain-specific data may also enhance classification accuracy and relevance. Additionally, to enhance accessibility and improve user experience for individuals with visual impairments, the system should support adjustable font sizes and provide compatibility with screen readers.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Chow Saw Chun Hwa, Norfaradilla Wahid;* ***data collection:*** *Chow Saw Chun Hwa. Norfaradilla Wahid;* ***analysis and interpretation of results:*** *Chow Saw Chun Hwa, Norfaradilla Wahid;* ***draft manuscript preparation:*** *Chow Saw Chun Hwa, Norfaradilla Wahid. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Howard E.LeWine, MD (2024). Becoming a vegetarian, Harvard Health Publishing. https://www.health.harvard.edu/nutrition/becoming-a-vegetarian
[2] Jim Holdsworth (2024). What is deep learning?, IBM. https://www.ibm.com/think/topics/deep-learning
[3] Yuka (2024). Yuka, Yuka. https://yuka.io/en/
[4] Peanut (2024). Peanut – find friends and support, Peanut. https://www.peanut-app.io/
[5] Pepper (2024). The social cooking app to share food with friends, Pepper. https://www.peppertheapp.com/
[6] IBM (2021). What are microservices?, IBM. https://www.ibm.com/think/topics/microservices
[7] MoritzLaurer (2023). MoritzLaurer/deberta-v3-base-zeroshot-v1.1-all-33, Hugging Face Transformers. https://huggingface.co/MoritzLaurer/deberta-v3-base-zeroshot-v1.1-all-33
[8] J. Sheffield and J. Lemetayer, "Critical success factors in project management methodology fit," presented at PMI® Global Congress 2010 – Asia Pacific, Melbourne, Victoria, Australia, 2010. Project Management Institute, Newtown Square, PA.
[9] Ruparelia, Nayan (2010) Software development lifecycle models, *ACM SIGSOFT Software Engineering Notes*, 35, pp. 8-13, https://doi.org/10.1145/1764810.1764814
[10] GeeksforGeeks (2024). Sequence diagrams – Unified modeling language (UML), GeeksforGeeks. https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
[11] GeeksforGeeks (2024). Class diagram | Unified modeling language (UML), GeeksforGeeks. https://www.geeksforgeeks.org/unified-modeling-language-uml-class-diagrams/
[12] GeeksforGeeks (2024). Zero-shot text classification using huggingface model, GeeksforGeeks. https://www.geeksforgeeks.org/nlp/zero-shot-text-classification-using-huggingface-model/
[13] Joshi, Ankur & Kale, Saket & Chandel, Satish & Pal, Dinesh. (2015). Likert scale: Explored and explained, British Journal of Applied Science & Technology. 7. 396-403. 10.9734/BJAST/2015/14975.