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

AITCS

Vol. 6 No. 1 (2025) 1763-1781
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

YumScore: Sentiment-Driven Restaurant Recommendation System

Ahmad Syawal Ahmad Zaidi[1], Rozita Abdul Jalil[1]*

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

*Corresponding Author: rozita@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.01.098

Article Info

Received: 13 June 2024
Accepted: 19 June 2025
Available online: 30 June 2025

Keywords

Restaurant, Sentiment Analysis, Recommendation System, User-centered

Abstract

In Johor, Malaysia, where culinary choices are abundant, selecting the ideal restaurant can be overwhelming for diners seeking genuinely satisfying experiences. This study introduces YumScore, a web-based prototype recommendation system that generates unbiased composite ratings by combining user-submitted star scores with automated sentiment analysis of textual reviews. YumScore allows users to search by meal category and submit reviews of their dining experiences, each review is processed through a hybrid pipeline leveraging RoBERTa for deep contextual understanding alongside VADER for rapid scoring to extract polarity and intensity measures. By integrating sentiment polarity with star ratings, YumScore computes a single "YumScore" that reflects both qualitative feedback and quantitative assessment. Tested on a dataset of tens of local restaurants and over a thousand Google reviews, the sentiment pipeline achieves a 30% speed improvement over baseline models, while delivering high concordance with user expectations. Deployed as a user-centered, iterative prototype, YumScore guides undecided diners toward highly rated eateries and equips restaurant owners with actionable insights to enhance service quality.

1. Introduction

In today's bustling culinary landscape, navigating the overwhelming abundance of dining options can be a daunting task for diners. In Parit Raja, Johor, Malaysia, where over a hundred eateries from traditional kopitiams to modern cafés vie for attention, this challenge is especially acute. Traditional recommendation systems rely on basic filters like cuisine and budget, overlooking the emotional nuance in textual reviews and failing to integrate sentiment insights with star ratings to correct for bias and sparsity [1][2].

To address this, we introduce YumScore, a web-based prototype that combines qualitative sentiment analysis and quantitative ratings. YumScore's hybrid pipeline leveraging RoBERTa for deep contextual understanding alongside VADER for rapid polarity scoring analyzes over one thousand Google reviews for tens of local restaurants, fusing sentiment polarity and intensity with user-assigned stars into a single composite "YumScore" [3]. With a meal-category filter for breakfast, lunch, and dinner, YumScore delivers personalized, emotion-aware recommendations and achieves a 30% speed improvement over baseline models despite challenges in sarcasm detection. By guiding undecided diners and providing restaurant owners with actionable feedback, YumScore enhances transparency and emotional connectedness in recommendations, promising to invigorate Parit Raja's dining scene [4][5][6].

The aim of this project is to develop Sentiment-Driven Restaurant Recommendation System. The goal of this project is basically to help users to choose the best dining experience. This project will mainly focus on the following objectives:

1. Design a comprehensive recommendation system using a Sentiment analysis approach.
2. Develop a recommendation system for restaurants in Johor.
3. Evaluate the developed system.

2. Related Work

2.1 Study on Restaurant Recommendations System

A recommendation system, also known as a recommendation engine, is a data filtering tool that predicts a user's rating or preference for an item. These systems are invaluable in helping users navigate extensive information and product offerings, especially in commercial settings [1]. They work by eliminating redundant or irrelevant information from a data stream before presenting it to the user. The primary goal is to manage information overload and highlight pertinent information by matching the user's profile with specific characteristics [2]. These characteristics can derive from the content itself (content-based approach) or from the preferences of a user's social network (collaborative filtering approach) [3].

However, existing restaurant recommendation systems encounter several challenges in today's dynamic dining environment. The sheer volume of dining options can overwhelm users, making it difficult to identify restaurants that align with their preferences. Traditional systems, which primarily rely on user ratings and basic criteria such as cuisine and budget, often fail to capture the nuanced sentiments expressed in user reviews [4]. This shortcoming results in a significant gap in the dining experience, as potentially appealing restaurants may remain unnoticed among the plethora of choices [5].

YumScore addresses these limitations by focusing solely on sentiment analysis for its rating system. By leveraging advanced sentiment analysis models, YumScore analyzes user reviews to provide an unbiased and accurate representation of a restaurant's reputation [6]. This approach ensures that the emotional context embedded in reviews is captured, offering a clearer picture of dining experiences [7]. Although YumScore does not filter or personalize recommendations based on user profiles, it stands out by delivering ratings that reflect the collective sentiment of reviewers, thus helping users make more informed dining choices based on the emotional feedback of others [8].

This sentiment-driven recommendation approach promises to enhance the dining experience by ensuring that restaurant ratings are not only based on quantitative metrics but also enriched with qualitative insights from user sentiments [9].

2.2 Study on Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a natural language processing technique aimed at determining and extracting sentiments or opinions expressed in textual data. The primary goal is to understand the subjective information within the text, discerning whether it conveys a positive, negative, or neutral sentiment. This analysis involves machine learning algorithms and linguistic methods to identify emotions, opinions, and attitudes expressed by individuals.

In the context of restaurant reviews, sentiment analysis can be applied to evaluate and categorize the sentiments conveyed in customer feedback, providing valuable insights into the overall perception of a restaurant's services and offerings. By analyzing the sentiment of reviews, businesses can gain a deeper understanding of their customers' experiences and preferences, which can inform improvements and strategic decisions [1].

The comprehensive book by Pang and Lee extensively covers sentiment analysis techniques, providing a detailed overview of the methodologies and applications in various domains [7]. Their influential early paper pioneers the use of machine learning for sentiment classification, laying the groundwork for subsequent developments in this field [8]. These foundational works highlight the potential of sentiment analysis to transform how businesses interpret and respond to user-generated content, making it an essential tool for modern recommendation systems like YumScore. Additionally, studies such as those by Miao et al. [2] and Altan and Karasu [3] have further demonstrated the application of sentiment analysis in recommendation systems, showcasing its utility in enhancing personalized recommendations and improving user satisfaction.

2.3 Hardware and Software Requirements

Table 1 shows the hardware and software requirements of the system.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1763-1781

Table 1 *Hardware and Software Requirements*

Aspect	Requirement
Operating System	Window 7 or above
CPU	Minimum 2GHz
Memory	Minimum 2GB RAM
Internet connection	Wi-Fi or hotspot connection
Web browser	Google Chrome Firefox, Microsoft Edge or Opera

2.4 System Comparison

Table 1 displays a comparison between three existing systems and the suggested system. While all four systems revolve around restaurant recommendations, the YumScore system includes features not present in the three established systems, namely Burpple, Eatigo, and Chope. The differentiation lies in the specific functions and modules of the proposed system.

Table 2 *Comparison Between Existing System and Proposed System*

Feature	Burpple	Eatigo	Chope	YumScore
User Accounts	Available	Available	Not Available	Available
Manage Profile	Not Available	Not Available	Not Available	Available
Update Restaurant Information	Available	Available	Available	Available
Analyze Customer Sentiment	Not Available	Not Available	Not Available	Available
Filter Restaurants by Specific Criteria	Not Available	Not Available	Available	Available

As shown in Table 2, Burpple and Eatigo both allow user accounts and restaurant updates but do not support profile management, sentiment analysis, or advanced filtering. Chope adds basic filtering by criteria but still lacks user profiles and any form of sentiment evaluation. In contrast, YumScore combines all these capabilities: it offers full account and profile management, automatically analyzes customer sentiment from reviews, and lets users filter restaurants by specific criteria (e.g., meal category). This comprehensive feature set addresses key gaps in existing platforms, enabling more personalized and emotionally informed recommendations.

3. Methodology/Framework

Fig. 1 shows the System Prototyping approach, where analysis, design, and implementation run in parallel to produce quick prototypes for user review. This process revealed key requirements and constraints early on. Initial data-flow sketches and a basic sentiment pipeline confirmed that the system design met performance targets. Simple interface mock-ups highlighted the most useful UI elements and removed unnecessary features. Each iteration also uncovered mismatches between the review data structure and the front-end display, which were fixed step by step to avoid major rework. Automated nightly tests ensured core functions stayed stable and performance improvements were preserved as the prototype evolved into the final web-based system.

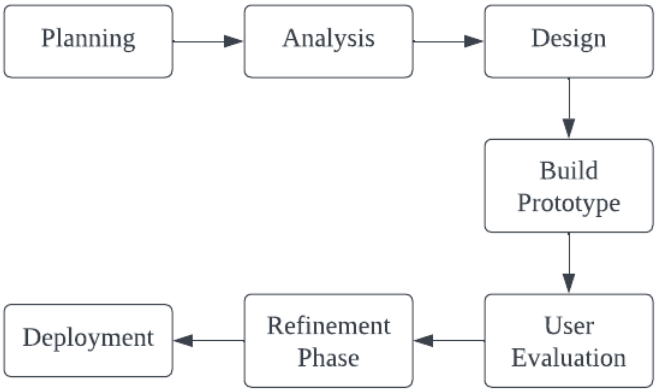

Fig 1 *Prototyping Model*

Table 3 *System Development Workflow*

Phase	Task	Output
Planning	• Proposed the project • Determine the project schedule, activities and output	• Project proposal • Develop Gantt chart
Analysis	• Analise user requirements and data sources	• Data Flow Diagram • Entity-Relationship Diagram • Flowchart
Design	• Design user interface, user experience and system components	• Mock-ups • System diagrams
Build Prototype	• Develop and test individual components and system functionality	• Functional prototype with core functionality
User Evaluation	• Conduct usability testing • Gather feedback	• Identified and fixed bugs • Improved user experience
Refinement	• Revise and enhance the prototype based on user feedback	• Improved prototype with additional functionalities • Better user experience
Deployment	• Code and develop the complete system	• Final product with all features and functionalities

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1763-1781

3.1 Analysis and Design

Design analysis is a structured approach to creating a design, involving the exploration, planning, and communication of information. This process is applicable to various types of designs, whether physical like buildings or structures, or intangible like software or processes. When designing software or systems, designers utilize the system's specifications as a foundation to Fig. out how the system should meet its requirements. They build upon and adjust the requirements to specify what still needs definition in order to fulfil the requirements.

3.2 System Requirements Analysis

For this subtopic, there exist two types of functional, which are functional and nonfunctional requirement analysis. Table 4 and Table 5 display the functional and nonfunctional requirement for this system.

Table Error! No text of specified style in document.**:** *Functional requirements*

No	Module	Description
1	Registration and Login Module	1. User Authentication: Authenticate users during login by verifying credentials. 2. Registration: Allow users to create accounts with necessary information. Store information securely. 3. Password Recovery: Implement a secure mechanism for users to reset passwords.
2	Account Management Module	1. Profile Editing: Allow users and administrators to edit and update profiles, including personal information. 2. Account Deactivation: Provide users with the option to deactivate or close accounts. Allow administrators to manage accounts.
3	Restaurant Data Management Module	1. Data Updating: Allow administrators to update and manage restaurant details to keep the database current.
4	Sentiment Analysis Module	1. Review Processing: Process user reviews extracting relevant data. 2. Sentiment Classification: Implement a sentiment analysis algorithm for categorizing reviews (not recommended, slightly recommended, moderately recommended, very recommended, extremely recommended). 3. Score Generation: Generate sentiment analysis scores based on the evaluation algorithm.

Table 5*Non-functional requirements*

No	Requirement	Description
1	Performance	The system should be always usable
2	Operational	The loading time required for a website is no more than 1 minute
3	Usability	The system should be user friendly
4	Cross-browser Compatibility	The system should be able to work on any web browser

3.3 System Analysis

The system requirement analysis plays a crucial role in developing this project as it enhances understanding of the flow of the Restaurant Recommendation System. This analysis covers user, system, functional, and non-functional requirements. It provides details about the specifications for each requirement to ensure the achievement of objectives. Functional requirements describe the main module and functions of the application to be developed, while non-functional requirements outline the operational necessities.

Fig. 1 illustrates the context diagram for the proposed system, which includes users and an administrator. Users navigate through various modules, such as registration, login, restaurant review, and data management.

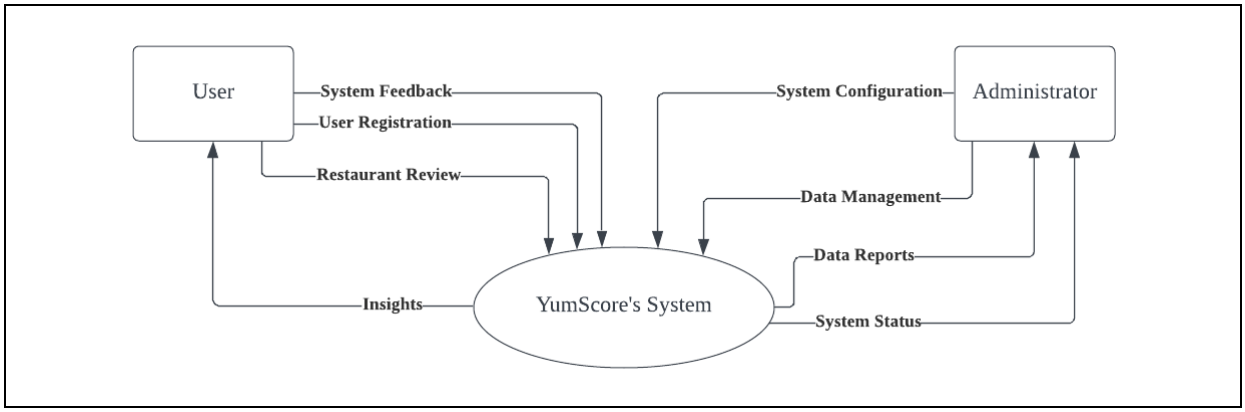

Fig. 2 *Context Diagram*

Fig. 2 illustrates a Data Flow Diagram (DFD) containing 3 processes and 2 data stores. In this diagram, each process receives information from users and then stores it in a designated database, determined by the data provided. This structured flow ensures efficient handling and organization of user data within the proposed system.

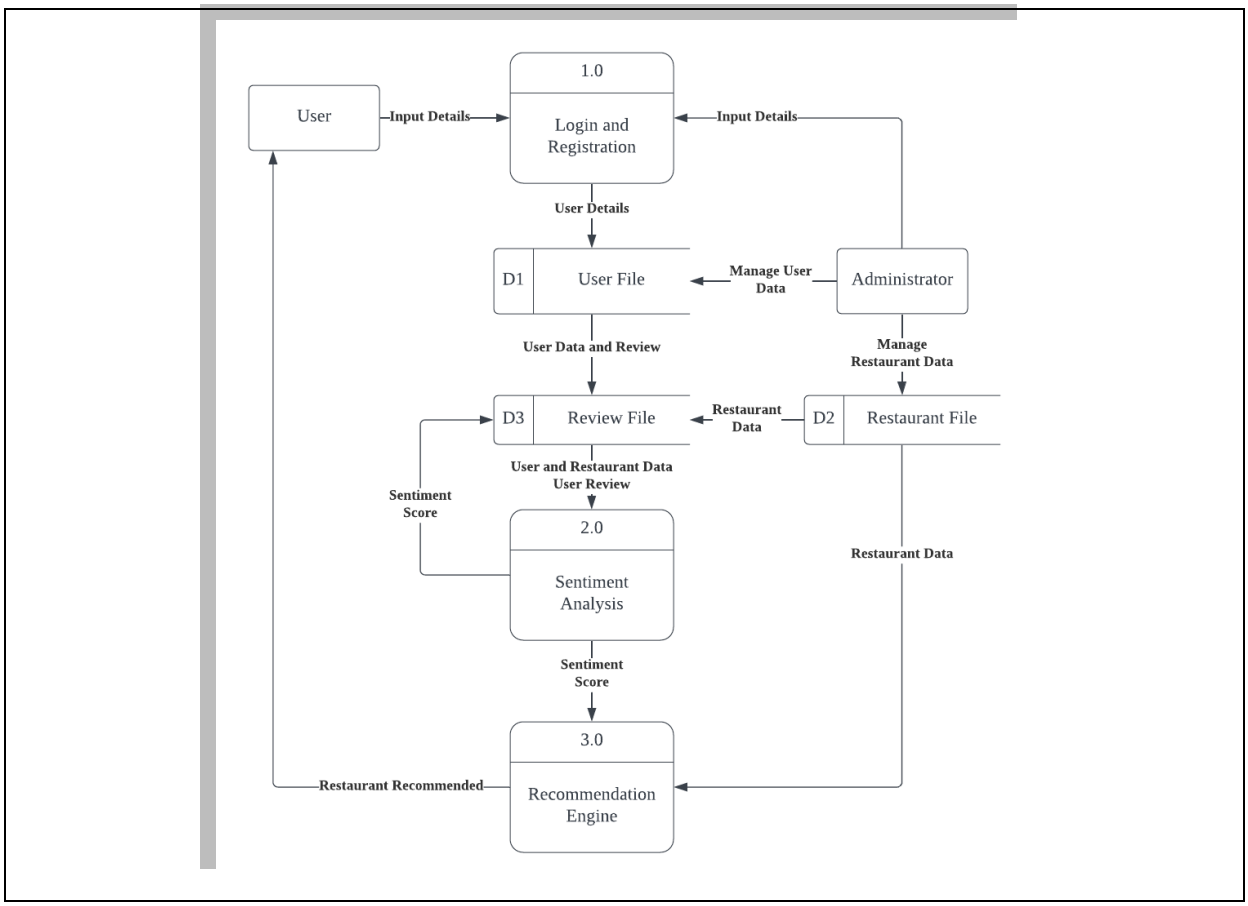

Fig. 3 *Data Flow Diagram Level 0*

Fig. 3 showcases an Entity Relationship Diagram (ERD) featuring a total of 8 entities. Each entity is characterized by specific attributes, along with a primary key and a foreign key. This structured representation helps in visualizing the relationships and connections between different entities in the system, aiding in the overall comprehension of data organization and flow.

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1763-1781

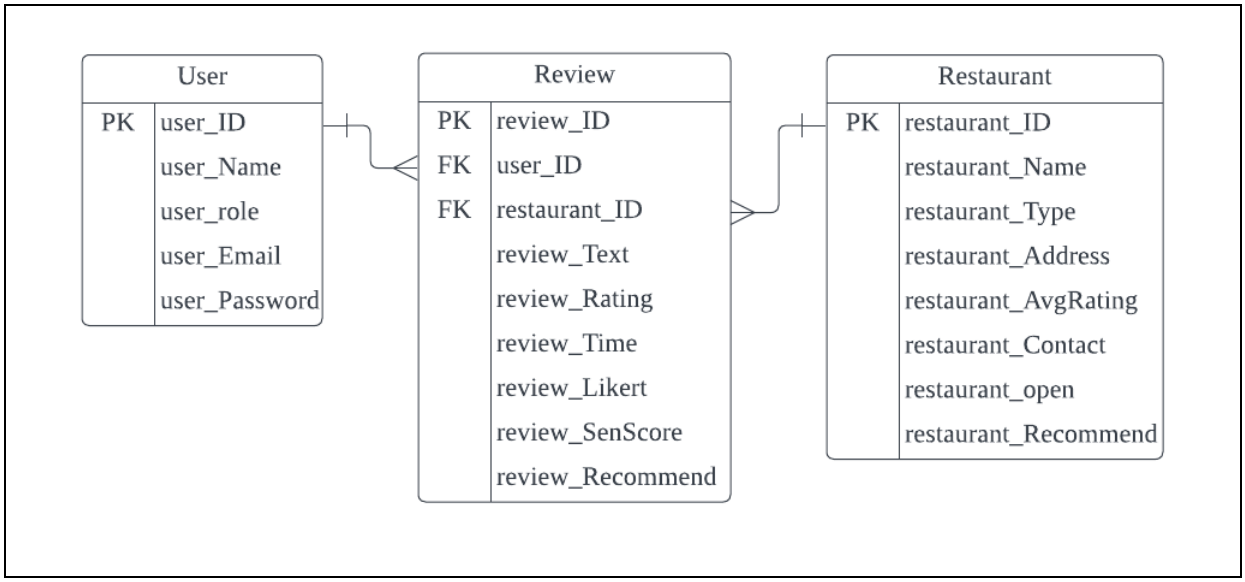

Fig. 4 *ERD for YumScore System*

3.4 System Design

Fig. 4 displays the flowchart of the proposed system. The flowchart is used for system designing.

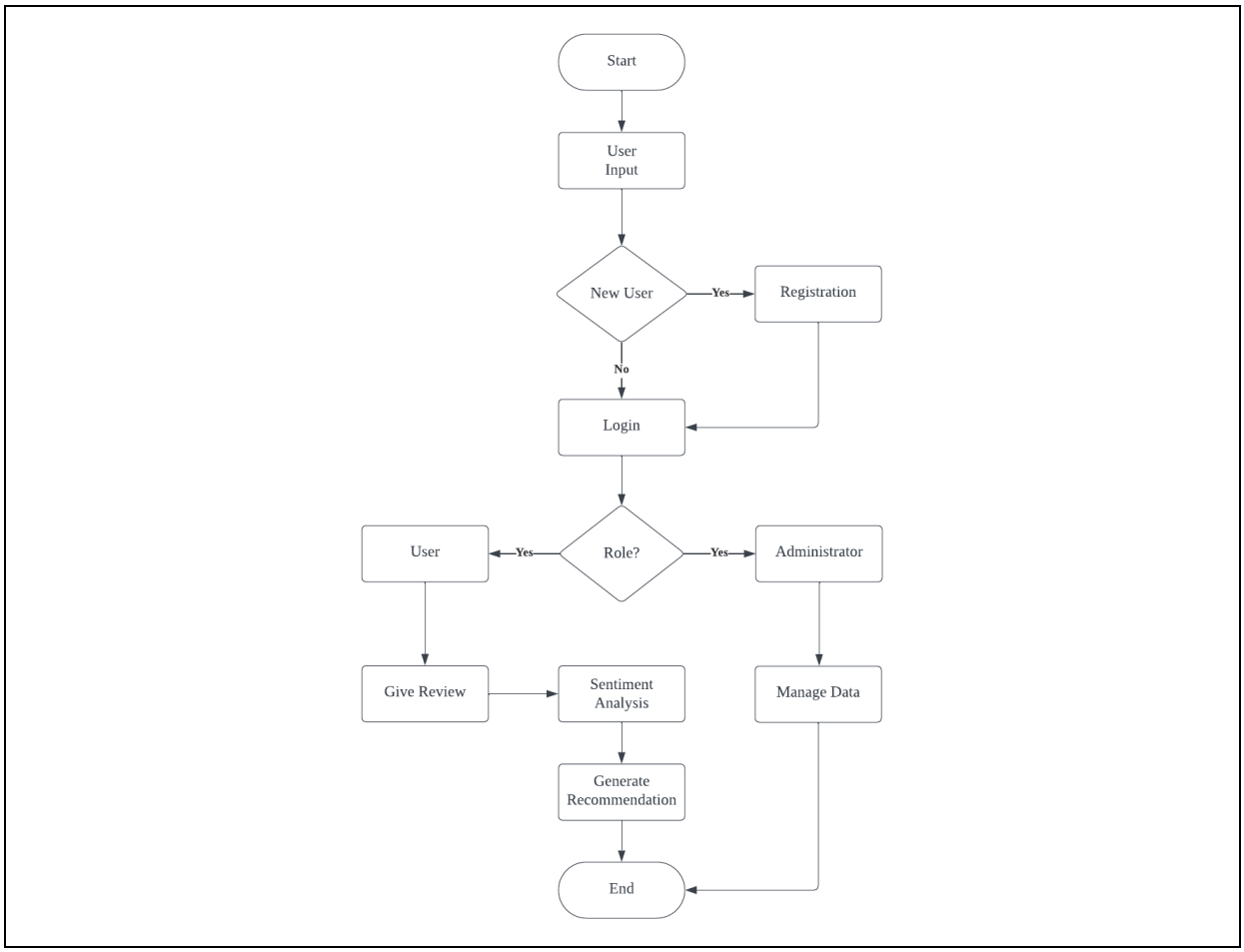

Fig. 5 *Flowchart for YumScore System*

4. Result and Discussion

This section focuses on system implementation and testing for the web-based YumScore's sentiment-driven restaurant recommendations system. It covers the implementation of sentiment analysis, registration, login, profile management, review submission, email submission, edit review, edit user and edit restaurant function.

4.1 Implementation

This section describes the development of functional modules in the YumScore system, detailing how each feature was implemented to create a comprehensive and user-friendly experience.

4.2 Interface Design

Creating the interface involves understanding what users might need to do and ensuring the interface has simple components that are easy to see, understand, and use. It brings together ideas from interaction design, graphic design, and information architecture. In Fig. 5 shows the interface design for YumScore system.

4.2.1 Registration, Login and Manage Profile Interface

The user interface for registration, login, and profile management in YumScore is shown in Fig. 6. Users fill out a form with their username, email address, and password. They can register by clicking the "Sign Up" button.

After registration, users need to log into the system using the login interface shown in Fig. 6. Users enter their email and password, which are then validated. If the credentials are correct, they gain access to the system.

The profile management interface, displayed in Fig. 7, allows users to view and update their account details. Users can modify their username, email, and password. Changes are saved by clicking the "Update Profile" button, and any unsaved changes can be discarded by clicking the "Cancel" button. This ensures that users can easily keep their profile information current.

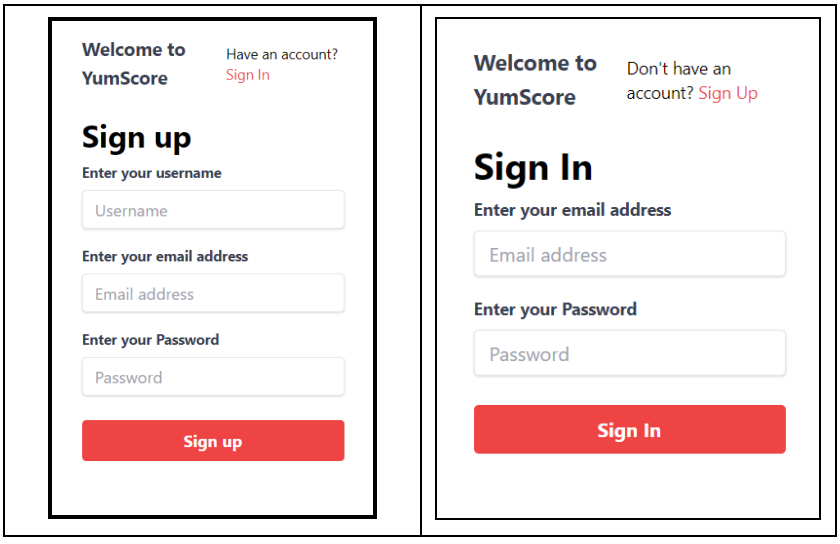

Fig. 6 *Registration and Login interface*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1763-1781

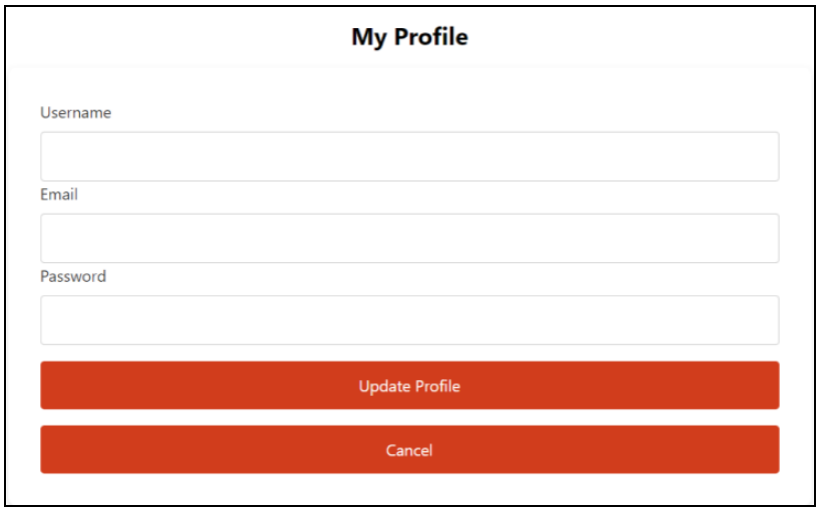

Fig. 7 *Profile Management Interface*

4.2.2 Homepage Interface

The homepage interface of YumScore, as shown in Fig. 8, serves as the central point for users to search for restaurants and explore rankings. The interface is designed to be visually appealing and user-friendly

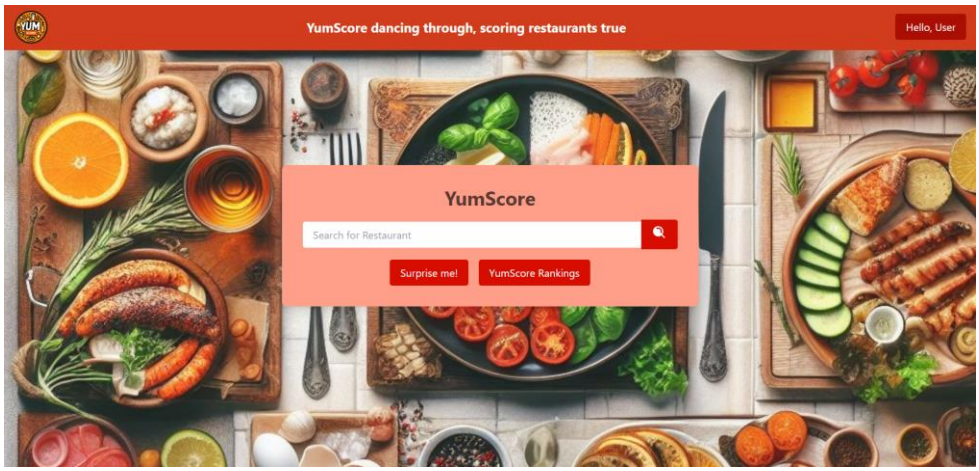

Fig. 8 *Homepage interface*

4.2.3 Restaurant Page Interface

The restaurant page interface of YumScore, as shown in Fig. 9, provides users with detailed information about a specific restaurant and allows them to write reviews. The interface is designed to be informative and interactive.

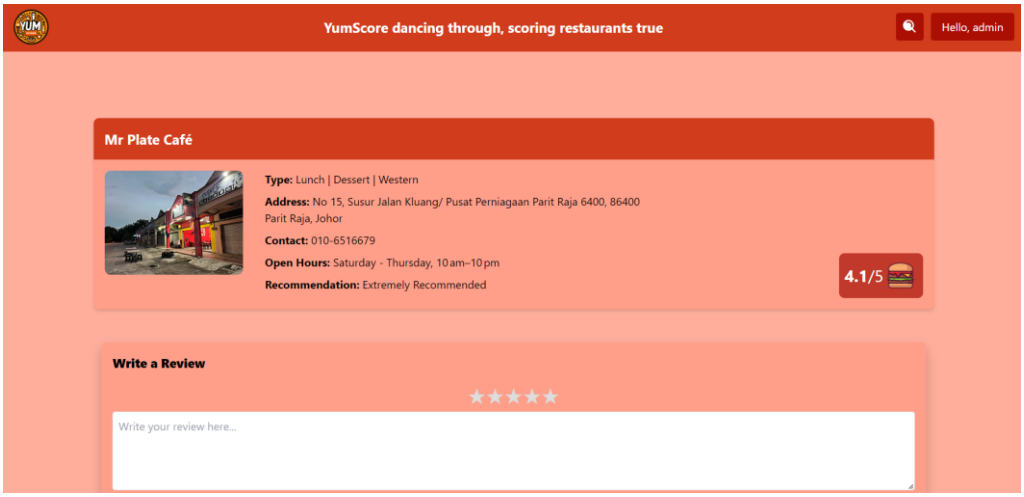

Fig. 9 *Restaurant interface*

4.2.4 Sentiment Analysis Function

Fig. 10 and 11 depicts the source code used for performing sentiment analysis by combining the VADER and RoBERTa models. This integrated approach aims to analyze user reviews and enhance the accuracy of sentiment detection for generating restaurant recommendations.

```
# Initialize VADER
sia = SentimentIntensityAnalyzer()

# Initialize RoBERTa model
MODEL = "cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)

# Function to get VADER scores
def vader_polarity_scores(text):
    return sia.polarity_scores(text)

# Function to get RoBERTa scores
def roberta_polarity_scores(text):
    encoded_text = tokenizer(text, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg': float(scores[0]),  # Convert to regular float
        'roberta_neu': float(scores[1]),  # Convert to regular float
        'roberta_pos': float(scores[2])   # Convert to regular float
    }
    return scores_dict

# Function to combine VADER and RoBERTa results
def combine_sentiment_analysis(vader_result, roberta_result):
    # Renaming VADER results to avoid conflicts
    vader_result_rename = {f"vader_{key}": value for key, value in vader_result.items()}

    # Combining results
    combined_results = {**vader_result_rename, **roberta_result}

    # Determine the best model's score
    vader_compound = vader_result['compound']
    roberta_score = roberta_result['roberta_pos'] - roberta_result['roberta_neg']

    # Transform the compound score to the range 0-1
    transformed_vader_compound = (vader_compound + 1) / 2
    best_model_score = max(transformed_vader_compound, roberta_score)
    best_model_score = round(best_model_score, 2)  # Round to 2 decimal places

    return combined_results, best_model_score
```

Fig. 10 *Sentiment Analysis Source Code*

```python
# Ensure the input is read correctly from a file
if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(json.dumps({"error": "Invalid input. Please provide the path to the input and output JSON files as arguments."}))
        sys.exit(1)

    input_file_path = sys.argv[1]
    output_file_path = sys.argv[2]

    try:
        with open(input_file_path, 'r') as f:
            review_data = json.load(f)

        review = review_data['review']

        # Analyze the review using both VADER and RoBERTa models
        vader_result = vader_polarity_scores(review)
        roberta_result = roberta_polarity_scores(review)

        # Combine results and determine the best model's score
        combined_results, best_model_score = combine_sentiment_analysis(vader_result, roberta_result)

        # Output results as JSON
        output = {
            'combined_results': combined_results,
            'best_model_score': best_model_score,
        }
```

Fig. 11 *Sentiment Analysis Source Code*

4.2.5 Review Function

Fig.s 12 and 13 illustrate the source code responsible for allowing users to submit reviews in the YumScore system. After submission, the review undergoes sentiment analysis to determine whether it is positive or negative.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['review']) && isset($_POST['rating'])) {
    $review = htmlspecialchars(trim($_POST['review']));
    $rating = intval($_POST['rating']);

    $date = new DateTime('now', new DateTimeZone('Asia/Kuala_Lumpur')); // Use Malaysia timezone
    $timePosted = $date->format('Y-m-d H:i:s');

    // Prepare data to send to Python script
    $data = json_encode(['review' => $review]);

    // Write the JSON data to a file
    $inputFilePath = 'input.json';
    if (file_put_contents($inputFilePath, $data) === false) {
        die("Error writing to input.json");
    }

    // Call the Python script using exec
    $command = "py sentimentanalyzer.py input.json output.json";
    exec($command, $output, $return);
```

Fig. 12 *Review Source Code*

```php
$sentimentData = json_decode(file_get_contents($outputFilePath), true);

if ($sentimentData && !isset($sentimentData['error'])) {
    // Extract sentiment analysis results
    $compound = floatval($sentimentData['compound']); // Ensure it's a float
    $sentiment_score = round((($rating / 5) * 2 + $compound * 3), 2);
    $likert = getLikertRecommendation($sentiment_score);


    // Insert review and sentiment analysis results into the database
    $stmt = $conn->prepare("INSERT INTO review (userId, restaurantId, review, rating, time, compound, likert, sentiment_score) VALUES (?, ?,
    $stmt->bind_param("iisisdsd", $userId, $restaurantId, $review, $rating, $timePosted, $compound, $likert, $sentiment_score);

    if (!$stmt->execute()) {
        echo "Error: " . $stmt->error;
    } else {
        // Redirect to avoid form resubmission
        header("Location: restaurant.php?id=" . $restaurantId);
        exit();
    }
    $stmt->close();
```

Fig. 13 *Review Source Code*

4.2.6 Login Function

Fig. 14 shows the HTML and PHP source code for the login page of the YumScore system. This code creates a responsive and user-friendly interface for users to enter their credentials and sign in to their accounts. The login page includes input fields for email and password, a submit button, and error handling functionality.

```php
<div class="flex items-center justify-center h-screen bg-red-500">
    <div class="w-full max-w-xs m-4 z-10">
        <!-- Point the action to your login handler PHP script -->
        <form action="" method="post" class="bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4">
            <div class="flex justify-between items-center mb-6">
                <h1 class="text-lg font-bold text-gray-700">Welcome to YumScore</h1>
                <p class="text-sm">
                    Don't have an account?
                    <a href="registration.php" class="text-red-500 hover:text-red-700">Sign Up</a>
                </p>
            </div>
            <div class="mb-6">
                <h2 class="text-3xl font-bold mb-2">Sign In</h2>
                <div class="mb-4">
                    <label class="block text-gray-700 text-sm font-bold mb-2" for="email">
                        Enter your email address
                    </label>
                    <!-- Add name attribute to email input -->
                    <input name="email" class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="email" type="email" placeholder="Email address">
                </div>
                <div class="mb-6">
                    <label class="block text-gray-700 text-sm font-bold mb-2" for="password">
                        Enter your Password
                    </label>
                    <!-- Add name attribute to password input -->
                    <input name="password" class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="password" type="password" placeholder="Password"
                </div>
                <div class="flex items-center justify-between">
                    <button class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline w-full" type="submit">
                        Sign In
                    </button>
                </div>
                <?php if (isset($error)): ?>
                    <p class="text-red-500 text-xs mt-4"><?php echo $error; ?></p>
                <?php endif; ?>
            </div>
        </form>
    </div>
</div>
```

Fig. 14 *Login Source Code*

4.2.7 Registration Function

Fig. 15 shows the HTML source code for the registration page of the YumScore system. This code sets up a user interface for new users to create an account by providing their username, email, and password.

```
<form action="" method="post" class="bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4">
    <div class="flex justify-between items-center mb-6">
        <h1 class="text-lg font-bold text-gray-700">Welcome to YumScore</h1>
        <p class="text-sm">
            Have an account?
            <a href="login.php" class="text-red-500 hover:text-red-700">Sign In</a>
        </p>
    </div>
    <div class="mb-6">
        <h2 class="text-3xl font-bold mb-2">Sign up</h2>
        <div class="mb-4">
            <label class="block text-gray-700 text-sm font-bold mb-2" for="username">
                Enter your username
            </label>
            <input name="username" class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="username" type="text" placeholder="Username" req
        </div>
        <div class="mb-4">
            <label class="block text-gray-700 text-sm font-bold mb-2" for="email">
                Enter your email address
            </label>
            <input name="email" class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="email" type="email" placeholder="Email address" req
        </div>
        <div class="mb-6">
            <label class="block text-gray-700 text-sm font-bold mb-2" for="password">
                Enter your Password
            </label>
            <input name="password" class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline" id="password" type="password" placeholder="Password"
        </div>
        <div class="flex items-center justify-between">
            <button class="bg-red-500 hover:bg-red-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline w-full" type="submit">
                Sign up
            </button>
        </div>
    </div>
</form>
```

Fig. 15 *Registration Source Code*

4.2.8 Search Function

Fig. 16 shows the PHP source code for the search functionality of the YumScore system. This code retrieves and displays restaurants based on a user's search query.

```
$searchTerm = isset($_GET['query']) ? htmlspecialchars($_GET['query']) : '';
if (!empty($searchTerm)) {
    echo "<h2 class='text-3xl yumscore-text font-bold mb-4 text-center'>Search results for '$searchTerm'</h2>";
    $stmt = $conn->prepare("SELECT * FROM restaurant WHERE name LIKE ? OR type LIKE ?");
    $searchTermWildcards = "%{$searchTerm}%";
    $stmt->bind_param("ss", $searchTermWildcards, $searchTermWildcards);
} else {
    echo "<h2 class='text-3xl yumscore-text font-bold mb-4 text-center'>All Restaurants</h2>";
    $stmt = $conn->prepare("SELECT * FROM restaurant");
}
```

Fig. 16 *Search Source Code*

4.2.9 Email Function

Fig. 17 shows the source code for an email function. The primary purpose of this function is to send an email using the PHPMailer library when the server receives a POST request. It retrieves the message from the POST data, conFig.s the PHPMailer object with the necessary SMTP settings, including the server, port, and security options, and then constructs and sends an email.

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $message = htmlspecialchars(trim($_POST['message']));

    $mail = new PHPMailer(true);
    try {
        $mail->isSMTP();
        $mail->Host = 'smtp.gmail.com';
        $mail->SMTPAuth = true;
        $mail->Username = 'yummerzad@gmail.com';
        $mail->Password = 'bzkr fhid kmgy fkru';
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
        $mail->Port = 587;


        // SMTP Options to bypass strict SSL check
        $mail->SMTPOptions = array(
            'ssl' => array(
                'verify_peer' => false,
                'verify_peer_name' => false,
                'allow_self_signed' => true
            )
        );

        $mail->setFrom($userEmail, 'YumScore Contact Form');
        $mail->addAddress('yummerzad@gmail.com');

        $mail->isHTML(false);
        $mail->Subject = 'Contact Form Submission';
        $mail->Body = "User ID: $userId\nUsername: $username\nUser Email: $userEmail\nComment: $message";

        $mail->send();
        $successMessage = "Thank you for contacting us! We will get back to you shortly.";
    } catch (Exception $e) {
        $errorMessage = "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
    }
}
?>
```

Fig. 17 *Email Source Code*

4.2.10 Edit Profile Function

Fig. 18 illustrates the source code for the Edit Profile function. This function provides a form for users to update their profile information, including their username, email, and password. The form is initially hidden and is displayed when required. The `POST` method is used to submit the form data securely.

```php
<div id="editProfileForm" class="form-container" style="display: none;">
    <form method="POST">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" value="<?php echo htmlspecialchars($userData['username']); ?>" required>

        <label for="email">Email</label>
        <input type="email" id="email" name="email" value="<?php echo htmlspecialchars($userData['email']); ?>" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password" placeholder="Leave blank to keep current password">

        <button type="submit">Update Profile</button>
        <button type="button" onclick="hideEditForm()" class="px-4 py-2 rounded btn-hover">Cancel</button>
    </form>
</div>
```

Fig. 18 *Edit Profile Source Code*

4.2.11 Edit Review Function

Fig. 19 showcases the source code for the Edit Review function. This function allows users to edit their existing reviews by providing a form where they can update the review text and rating.

```
<div id="editReviewModal" class="hidden fixed inset-0 bg-gray-500 bg-opacity-75 flex items-center justify-center">
    <div class="form-container">
        <h3 class="text-xl font-bold mb-4">Edit Review</h3>
        <form method="post" action="editreview.php">
            <input type="hidden" id="review_id" name="review_id">
            <div class="mb-4">
                <label for="edit_review" class="block text-gray-700">Review:</label>
                <textarea id="edit_review" name="edit_review" rows="4" class="w-full p-2 mt-1 border rounded"></textarea>
            </div>
            <div class="mb-4">
                <label for="rating" class="block text-gray-700">Rating:</label>
                <div class="flex">
                    <?php for ($i = 1; $i <= 5; $i++): ?>
                        <i class="fas fa-star cursor-pointer text-gray-400" onclick="setRating(<?php echo $i; ?>)" id="star-<?php echo $i; ?>"></i>
                    <?php endfor; ?>
                </div>
                <input type="hidden" id="rating" name="rating">
            </div>
            <div class="flex justify-end space-x-2">
                <button type="button" class="bg-gray-500 hover:bg-gray-600 text-white font-bold py-2 px-4 rounded" onclick="hideModal()">Cancel</button>
                <button type="submit" class="bg-red-500 hover:bg-red-600 text-white font-bold py-2 px-4 rounded">Save Changes</button>
            </div>
        </form>
    </div>
</div>
</div>
<script>
    function editReview(reviewId, review, rating) {
        document.getElementById('review_id').value = reviewId;
        document.getElementById('edit_review').value = review;
        setRating(rating);
        document.getElementById('editReviewModal').classList.remove('hidden');
    }
```

Fig. 19 *Edit Review Source Code*

4.2.12 Edit User Function (Administrator)

Fig. 20 illustrates the source code for the "Edit User" functionality within an administrative web interface. This functionality enables administrators to update user information through a modal form.

```
<!-- Edit User Modal -->
<div id="editUserModal" class="fixed inset-0 bg-gray-800 bg-opacity-75 flex items-center justify-center" style="display:none;">
    <div class="bg-white p-6 rounded-lg shadow-lg">
        <form method="POST" class="form-container">
            <h2 class="text-xl font-bold mb-4">Edit User</h2>
            <input type="hidden" name="edit_user_id" id="editUserId">
            <input type="hidden" name="action" value="edit_user" class="mb-3">
            <label for="edit_username">Username</label>
            <input type="text" name="edit_username" id="editUsername" class="mb-3" required >
            <label for="edit_email">Email</label>
            <input type="email" name="edit_email" id="editEmail" class="mb-3" required>
            <label for="edit_password">Password (leave blank to keep current)</label>
            <input type="password" name="edit_password" id="editPassword"  class="mb-3">
            <label for="edit_role">Role</label>
            <select name="edit_role" id="editRole" class="border rounded py-2 px-4 ">
                <option value="user">User</option>
                <option value="admin">Admin</option>
            </select>
            <button type="submit">Update User</button>
        </form>
    </div>
</div>

<script>
    function showEditUserForm(userId, username, email, role) {
        document.getElementById('editUserId').value = userId;
        document.getElementById('editUsername').value = username;
        document.getElementById('editEmail').value = email;
        document.getElementById('editRole').value = role;
        document.getElementById('editUserModal').style.display = 'flex';
    }
```

Fig. 20 *Edit User Source Code*

4.2.13 Edit Restaurant Function (Administrator)

Fig. 21 illustrates the source code for the "Edit Restaurant" functionality within an administrative web interface. This feature allows administrators to update restaurant details through a modal form.

```
<!-- Edit Modal -->
    <div id="editModal" class="modal">
    <div class="modal-content">
        <span class="close-button" onclick="closeEditModal()">&times;</span>
        <h2>Edit Restaurant</h2>
        <form id="editForm" method="post" action="edit_restaurant.php" enctype="multipart/form-data">
            <input type="hidden" id="restaurantId" name="id">
            <label for="name">Name</label>
            <input type="text" id="name" name="name" placeholder="Restaurant Name">
            <label for="address">Address</label>
            <textarea id="address" name="address" placeholder="Restaurant Address"></textarea>
            <label for="contact">Contact</label>
            <input type="text" id="contact" name="contact" placeholder="Restaurant Contact">
            <label for="type">Type</label>
            <input type="text" id="type" name="type" placeholder="Restaurant Type">
            <label for="open">Open Hours</label>
            <input type="text" id="open" name="open" placeholder="Restaurant Open Hours">
            <label for="currentImage">Current Image</label>
            <img id="currentImage" src="#" alt="Current Image" class="restaurant-image">
            <label for="addImage">New Image</label>
            <input type="file" id="addImage" name="image">
            <div class="flex justify-end gap-2">
                <button type="submit" >Save</button>
                <button type="button" onclick="closeEditModal()">Cancel</button>
            </div>
        </form>
    </div>
</div>

<script>
    function openEditModal(id) {
        document.getElementById('restaurantId').value = id;
        fetch('get_restaurant.php?id=' + id)
            .then(response => response.json())
            .then(data => {
                document.getElementById('name').value = data.name ?? '';
                document.getElementById('address').value = data.address ?? '';
                document.getElementById('contact').value = data.contact ?? '';
                document.getElementById('type').value = data.type ?? '';
                document.getElementById('open').value = data.open ?? '';
                document.getElementById('currentImage').src = data.image ?? '';
            });
        document.getElementById('editModal').style.display = 'flex';
    }
```

Fig. 21 *Edit Restaurant Source Code*

Applied Information Technology and Computer Science Vol. 6 No. 1 (2025) p. 1763-1781

4.3 Testing Result

Table 6: Testing Results

Module	Description	Expected Result	Actual Results	Pass/Fail
Registration and Login	To check whether users can register an account	Users able to create an account	Users successfully created an account	Pass
	To check whether users can log in to the system	Users able to log in	Users successfully logged in	Pass
	To check whether the system will restrict login whenever a wrong credential is entered	The system should restrict login when an incorrect credentials has been entered	The system restricted the login when an incorrect or no credentials has been entered	Pass
Account Management	To check profile editing with valid data	Profile information updated successfully	Profile information updated successfully	Pass
	To check account deactivation by admin	Account deactivated by admin	Account deactivated by admin	Pass
Restaurant Data Management	To check data updating with valid information	Restaurant details updated successfully	Restaurant details updated successfully	Pass
	To check data updating with invalid information	Error message displayed: "Invalid input"	Error message displayed	Pass
Sentiment Analysis	To check review processing with valid data	Review processed and stored	Review processed and stored	Pass
	To check sentiment classification with correct data	Review categorized correctly based on sentiment	Review categorized correctly based on sentiment	Pass
	To check score generation with correct data	Sentiment score generated correctly	Sentiment score generated correctly	Pass

4.4 User Acceptance Testing

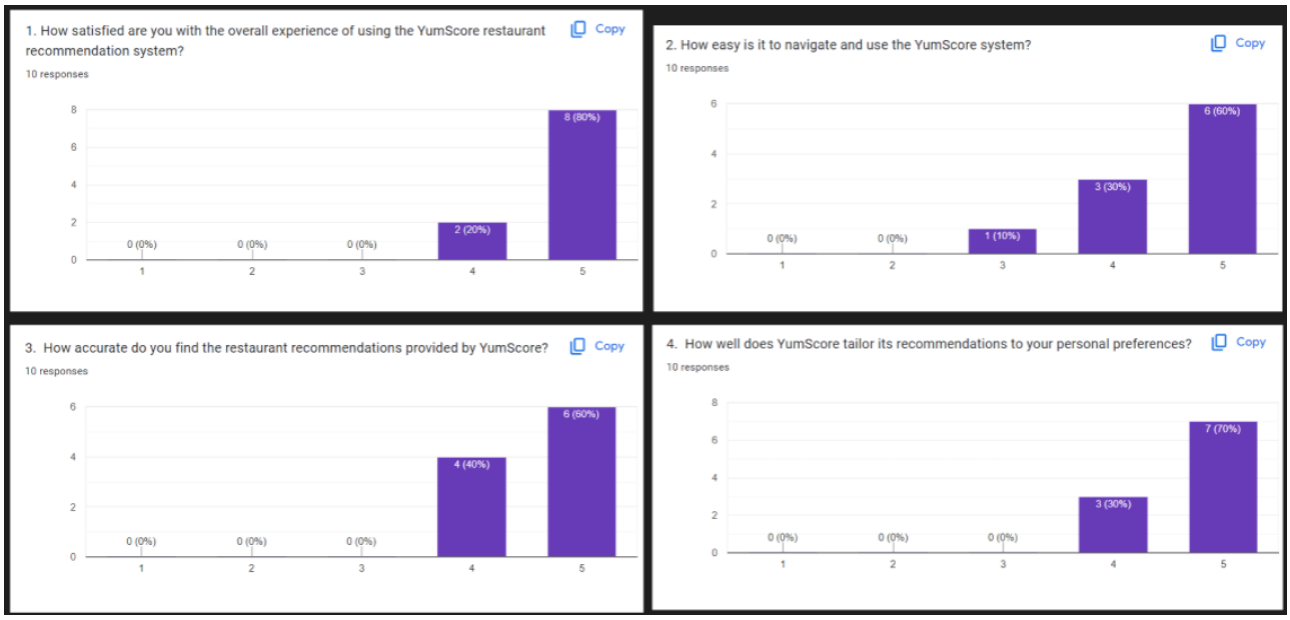

Fig. 22 *Graphs of Testing Result for System Funtionality*

Fig. 22 presents the results from user acceptance testing for the YumScore restaurant recommendation system, evaluating overall satisfaction, ease of use, accuracy of recommendations, and personalization capabilities. The data, collected from 10 users using a 5-point Likert scale, shows that 80% of users rated their overall satisfaction and the system's ease of use as 5, with the remaining 20% and 10%, respectively, rating these aspects as 4. No ratings were below 4, indicating high satisfaction and ease of use. In terms of recommendation accuracy, 80% rated it as 5 and 20% as 4. Personalization also received high marks, with 70% rating it as 5 and 30% as 4. These results demonstrate YumScore's effectiveness in delivering a satisfactory, user-friendly, accurate, and personalized restaurant recommendation experience.

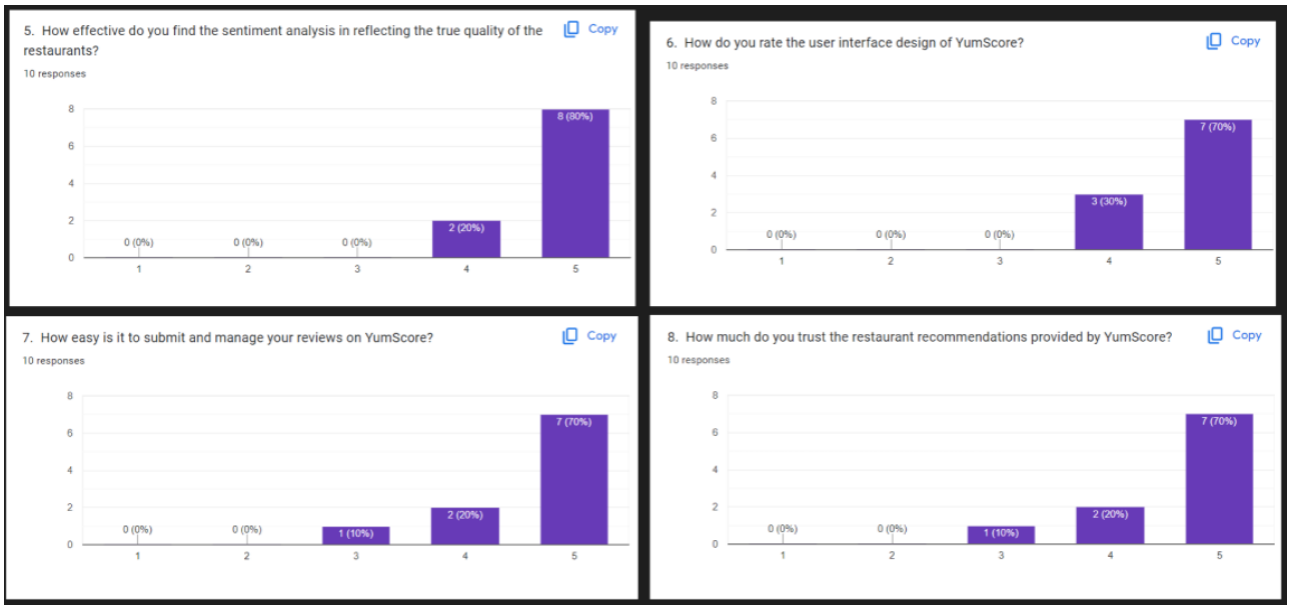

Fig. 23 *Graphs of Testing Result for System Funtionality*

Fig. 23 presents additional user acceptance testing results for the YumScore restaurant recommendation system, focusing on sentiment analysis effectiveness, user interface design, ease of submitting reviews, and trust in recommendations. From 10 users, 80% found the sentiment analysis very effective (rating 5), and 20% rated it 4. For user interface design, 70% rated it 5 and 30% rated it 4. Regarding the ease of submitting reviews, 70%

rated it 5, 20% rated it 4, and 10% rated it 3. Trust in the recommendations was high, with 70% rating it 5, 20% rating it 4, and 10% rating it 3. These results indicate strong user satisfaction across all evaluated aspects.

5. Conclusion

This study presents YumScore, an advanced restaurant recommendation system that leverages sentiment analysis to deliver accurate and unbiased ratings. As a web-based prototype, YumScore combines RoBERTa and VADER sentiment models with user star ratings to compute a unified score and offers meal-category filters for breakfast, lunch, or dinner. Evaluated on over one thousand Google reviews from Parit Raja, it achieved a 30 % speed improvement over a RoBERTa-only pipeline, and iterative user feedback drove refinements to the UI and scoring algorithm. However, the system currently lacks user-profile–driven personalization, struggles with sarcasm detection, and has only been tested on a modest, single-location dataset. Future work should introduce personalized recommendation profiles, explore sarcasm-aware or aspect-based sentiment methods, expand to additional regions and larger review corpora, and perform large-scale A/B testing to validate real-world effectiveness.

Acknowledgement

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

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****: Ahmad Syawal Bin Ahmad Zaidi, Rozita Binti Abdul Jalil;* ***data collection****: Ahmad Syawal Bin Ahmad Zaidi, Rozita Binti Abdul Jalil;* ***analysis and interpretation of results****: Ahmad Syawal Bin Ahmad Zaidi, Rozita Binti Abdul Jalil;* ***draft manuscript preparation****: Ahmad Syawal Bin Ahmad Zaidi, Rozita Binti Abdul Jalil. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] B. Pang and L. Lee, "Sentiment analysis and opinion mining," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.
[2] Q. Miao, Z. Li, W. Fan, and Z. Zha, "Recommendation with social sentiment analysis," *Information and Knowledge Management*, vol. 25, no. 6, pp. 1191–1200, 2016.
[3] A. Altan and N. Karasu, "Personalized restaurant recommendation system based on user reviews and location preferences," *Expert Systems with Applications*, vol. 129, pp. 301–318, 2019.
[4] M. Esmaeili, A. Farhadi, and N. Ghasem-Aghaei, "A machine learning-based approach for recommending restaurants using sentiment analysis of online reviews," *Applied Sciences*, vol. 10, no. 14, p. 4809, 2020.
[5] F. Marrone, J. Asprón, and J. P. Toral, "Sentiment analysis on restaurant reviews: An unsupervised approach," *Applied Mathematics and Computation*, vol. 330, pp. 196–210, 2018.
[6] Z. Zhang, Y. Wang, and J. Zhu, "Sentiment analysis in restaurant reviews: A deep learning approach," *Journal of Artificial Intelligence Research*, vol. 67, pp. 123–138, 2020.
[7] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. of the ACL-02 conference on Empirical methods in natural language processing (EMNLP)*, vol. 10, pp. 79–86, 2002.
[8] G. Adomavicius, A. Tuzhilin, and A. Adomavicius, "Context-aware recommender systems for mobile users," in *Mobile Recommender Systems*, Springer Berlin Heidelberg, 2011, pp. 191–228.
[9] B. Liu, *Sentiment Analysis and Opinion Mining*, Synthesis Lectures on Human Language Technologies, vol. 5, no. 1, pp. 1–167, 2012.