ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of the FoodFind project, our innovative web application designed to enhance the dining experience.

First and foremost, we extend our sincere appreciation to our project supervisor, Mr. Bhusan Thapa, for his invaluable guidance and support throughout the project. Their expertise and constructive feedback played a pivotal role in shaping the direction and functionality of FoodFind.

Our gratitude also extends to Nepal College of Information Technology for providing us with the platform and resources to undertake this project. The opportunity to work on a real-world application enriched our learning experience and paved the way for our future endeavors.

We would also like to thank our teammates for their dedication, hard work, and collaboration. Each member's unique skills and efforts were crucial in bringing FoodFind to life.

In conclusion, the success of the FoodFind project would not have been possible without the support and contributions of our supervisor, teammates, and institution. We are immensely grateful for this opportunity and look forward to exploring new horizons in the world of technology and innovation.

Sincerely,
Bhok Lagyo Team

ABSTRACT

It is difficult to find delicious food at good restaurants in today's crowded city. Especially for those who are unfamiliar with places they have never been. FoodFind attempts to help those users in finding an appropriate place to eat near them. FoodFind is a webapp that helps users find restaurants and small businesses that match their tastes. FoodFind allows users to simply search for restaurants by name and location, as well as filter them depending on the food they provide. FoodFind allows users to see menus of different restaurants, see prices and see the variety of food items available. User's can also provide the reviews for the restaurants they visit. Which will help other users to know about the place. Additionally, users can also share the location of the restaurants they visited with others. FoodFind aims to help restaurants by showcasing them to other users that their food is best. FoodFind also includes filters to narrow down the user's choice based on food type, location and user reviews. This allows user to find exactly the place where they can enjoy their food. FoodFind is about connecting people with the quality food. A community where user's can share their love for the food and discover new places to try out as well tell others if the place is worth it or not. With FoodFind, every meal is an opportunity to explore and enjoy the city's rich flavours.

Keywords: *Web Application (Webapp), Hybrid Recommendation*
RESTFUL APIs, MVC, RestaurantDiscovery

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
DRF	Django Rest Framework
ER	Entity Relationship
ORM	Object-Relational Mapping
REST	Representational State Transfer
OSS	Open Source Software
S.N.	Serial Number
CSS	Cascading Style Sheets
HTML	Hyper-Text Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language

1. Introduction

Restaurants have become an integral part of the daily life of the average Nepali salaried class. [1]. Food has been considered as an important attraction for travelers, especially those who are traveling internally or abroad. As food tourism grows, food is becoming an important part of the marketing strategy of places of destination. Therefore, many destinations are trying to introduce or develop exotic foods to attract tourists by providing various culinary experiences. [2].

According to The Kathmandu Post, an estimated 15,000 restaurants and cafés are currently operating within the Kathmandu Valley. [3]. Rather than staying at home and cooking, people are spending that extra time (and money) to try new things in a city. [1]. Due to increasing numbers of restaurants and cafés people are confused on what to try and where to try. Due to the bustling number of restaurants available, many people are unaware of the great places with good food near them. FoodFind helps them to find the restaurants or small café based on their tastes and preferences. It is aimed at both internal and external tourists who are unsure where to go for a meal nearby them.

This project is an endeavor taken to contribute as much as we collectively can to help the people and the tourists flowing in the country by offering a platform that not only makes it easier to find high quality food, but also encourages a community of foodies eager to share their tastes with others.

1.1 Project Overview

FoodFind is a web app that helps users to find places nearby to eat. Users can sign up and start looking for restaurants right away. Once the user successfully signed up they can search for a restaurant by its name or where it's located. Users can also choose what kind of food they want to eat. On FoodFind, users can look at menus, see how much things cost, and check out different kinds of food. They can also write reviews about the restaurants they visit to help other people know what's good. Additionally, users can tell others where they've been by sharing the location. Users can also see the restaurants with the highest reviews, so they can decide where to go. For restaurants, FoodFind is a way to show off their food to more people. The app makes it easy for users to find exactly what they're hungry for with filters for food type, place, and what other people say about the food.

1.2 Problem Statement

Kathmandu Valley, the capital of Nepal, is one of the country's busiest cities. Just like the thousands of people, It consists of many places to eat, from small food carts to big fancy restaurants. However, with an increasing number of restaurants, it is difficult for people to find the truly good ones. Both locals and visitors new to the city have difficulty finding the best places to eat and relax. As a result, some great places go unnoticed. There is also no easy way for people to find food that matches their tastes and food preferences. Many people either ask locals or go to random places in search of good food. This makes it difficult for them to discover new favourite restaurants. FoodFind aims to address this by serving as a guide, assisting users in discovering great restaurants that match their preferences. FoodFind is a method of consistently providing them with information about unnoticed and unvisited places to eat, as well as encouraging them to share the locations with others so that others can enjoy their meals as well.

1.3 Objectives

To overcome the existing challenges and for alignment with our goals, the project aims to fulfill the objective listed below

- To develop a user-friendly platform to explore Kathmandu's restaurant scene.
- To enhance online presence for local restaurants and facilitate the sharing of culinary reviews and experiences.

1.4 Significance of the Study

The importance of the FoodFind project lies in its potential to transform the dining landscape of the Kathmandu Valley. By providing a user-friendly platform, it enables locals and internal or external tourists to discover and enjoy a variety of dishes that suit their taste, preferences and location. This will not only make it easier to find quality restaurants, cafés or local eateries, but will also help create a vibrant community of food lovers who can share their valuable reviews and recommendations. In addition, it offers restaurants a valuable opportunity to showcase their offerings and attract new people, helping them grow their business and the local food industry. In short, FoodFind aims to enrich the dining experience for everyone and support restaurants and local restaurants to grow their business.

1.5 Scope and Limitations

In the current scope, FoodFind aims to be a user-friendly web app that connects food lovers with a wide variety of dining options. It's designed to simplify the search for good eats, whether you're a local or a visitor. The platform will offer detailed information about restaurants, including menus, prices, and user reviews, making it easier for everyone to find the perfect meal to match their preferences and budget.

The following are the limitations of the project that are realized:

- Initially, FoodFind will only cover restaurants, cafés and local food place within Kathmandu Valley, which might limit options for users outside the city.
- Users will need a stable internet connection to access the app, which could be a barrier in areas with poor connectivity.
- The success of the app relies heavily on users actively sharing their dining experiences, and less participation could affect the quality and quantity of reviews and recommendations.

2. Literature Review

Restaurant discovery applications have revolutionized the way people find, choose, and experience dining options. They leverage user-centered design, data analytics, and community engagement to provide personalized and practical solutions. The ”FoodFind” web app aims to assist users in finding restaurants that align with their culinary preferences by offering filters, reviews, and menu browsing features, promoting a supportive ecosystem around dining. This literature review explores various aspects of restaurant discovery and community engagement through similar applications to understand how FoodFind fits into the broader context.

2.1 User-Centric Design in Restaurant Discovery Apps

User-centric design is vital in improving user satisfaction and engagement in restaurant discovery apps. According to Ghani and Suleman (2020), personalization significantly impacts user experience, as users prefer recommendations based on their tastes and preferences. They discuss how features like filtering by cuisine type, location, and reviews improve the accuracy of recommendations. [4]. Community-driven reviews provide social validation, aiding in decision-making [5]Similarly, in their study on mobile applications for food services, Alomari et al. (2019) emphasize the importance of customizable filtering options that allow users to refine searches according to specific requirements such as budget or dietary restrictions [6].

2.2 Community Engagement and Social Influence

The role of community engagement is crucial in restaurant discovery apps, as it fosters trust and enhances user engagement. Research by Pantelidis (2019) highlights how user reviews influence consumer perceptions and behavior. Social influence plays a significant role in shaping users' dining preferences and encourages active participation [5]. Another study by Zhang et al. (2021) supports the notion that community engagement through reviews and recommendations helps build a supportive ecosystem where users share experiences and discover quality dining options [7].

2.3 Technical Framework and Data Management

Although FoodFind's abstract does not delve into technical details, understanding the technical framework is crucial for improving restaurant discovery apps. In their work, Singh et al. (2020) describe a system architecture based on machine learning algorithms that predict user preferences using data collected from past behavior. The use of clustering techniques, collaborative filtering, and content-based filtering can significantly enhance recommendation accuracy [8]. Furthermore, ensuring data integrity and regularly updating restaurant information, such as menus and prices, can vastly improve user trust and engagement [9].Expanding international availability and handling scalability are essential challenges in restaurant discovery [10]

2.4 Gaps and Opportunities

Despite the comprehensive features provided by restaurant discovery apps, certain gaps and areas of improvement remain :

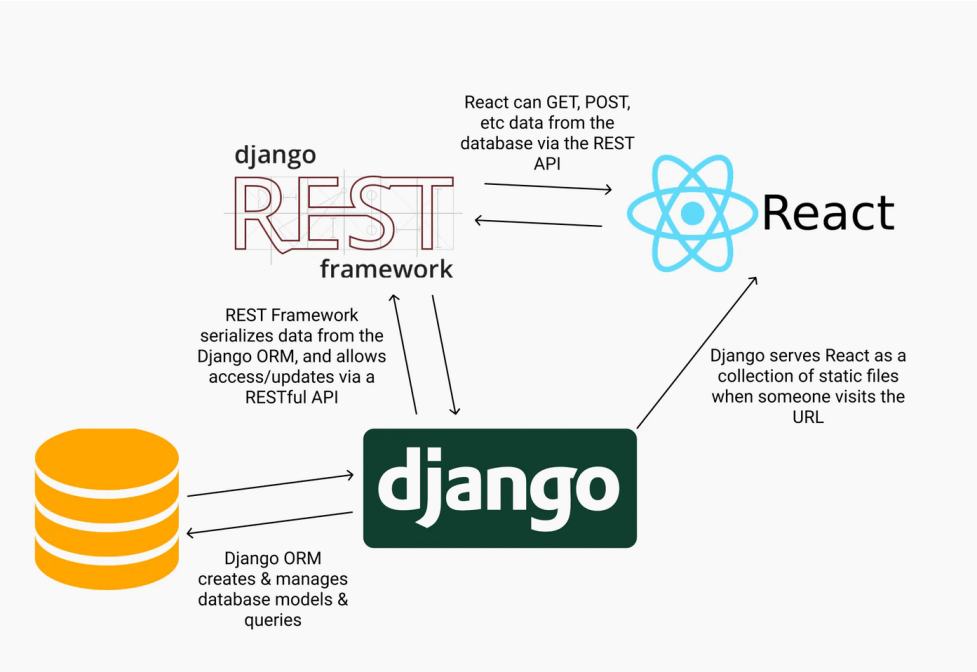
- Diversity in Listings : Expanding listings to include a wider variety of cuisines and dining experiences enhances inclusivity [11].
- Enhanced Personalization : Incorporating machine learning algorithms for more personalized recommendations remains a challenge [8].
- Regular Data Updates : Ensuring menus and prices are accurate and updated regularly requires better partnerships with restaurants [9].

3. Methodology

This section includes the methodology that is being followed during the development of the project.

3.1 Technical Architecture

React.js and Django REST Framework (DRF) interact to form a robust client-server setup. React.js, a popular JavaScript library for building user interfaces, is utilized on the client side to create a dynamic and responsive experience for users. It manages the presentation layer, handling user interactions, data presentation, and sending requests to the server. On the server side, Django REST Framework acts as the backend, handling data management, business logic, and API endpoints. DRF receives HTTP requests from the React.js frontend, processes these requests (such as user authentication, data retrieval, and manipulation), and sends back the appropriate responses in JSON format, which React.js then uses to update the user interface. This separation of concerns ensures that the frontend and backend are loosely coupled, promoting scalability and ease of maintenance while enabling efficient development workflows where frontend and back-end developers can work independently yet collaboratively.



[12]

Figure 1: Technical Architecture

3.2 Software Development Life Cycle

We'll adopt the Incremental Software Development Model, tackling requirements, research design, training, evaluation, and documentation in iterative cycles.

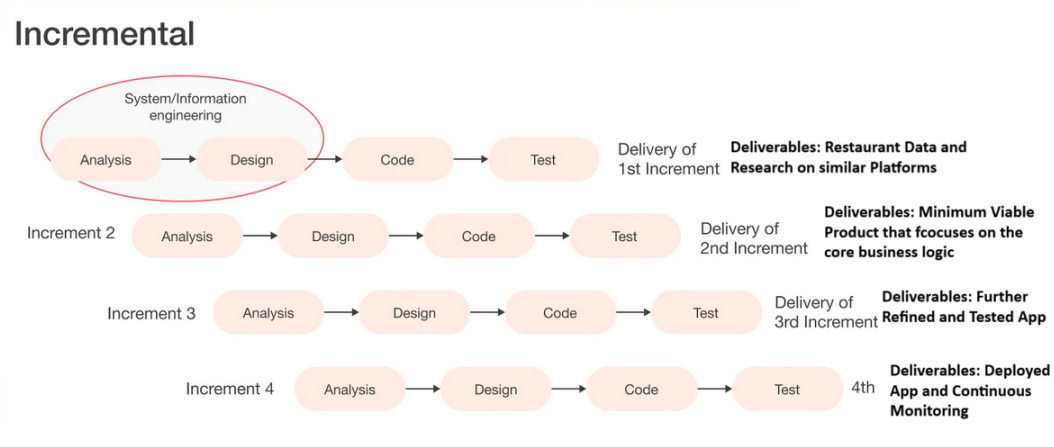


Figure 2: Incremental model

In the first increment, we'll analyze the project's requirements, defining its scope, objectives, and constraints. For FOODFIND, this translates to pinpointing the system's essential characteristics, such as restaurant menu and food data modeling, user profile, user reviews and recommendation. We'll then embark on crucial research, gathering and examining data that aligns with these objectives. This entails investigating existing platforms providing similar services. We'll gather the required data for the application using platforms providing dataset for testing purposes.

The second increment focuses on design, prototyping, and early testing. We'll thoroughly plan the FOODFIND implementation, designing the system architecture, selecting suitable technologies, and developing streamlined prototypes to demonstrate core functionalities. This is followed by coding the core business logic and integrating them with the data sources. Rigorous tests will be conducted to validate initial functionality and identify potential areas for improvement. While initial deployment isn't the primary focus here, preliminary feedback from the supervisor and the test results are valuable for ensuring the system is on track to meet the defined requirements.

The third increment revolves around refinement, integration, and enhanced testing. We'll revisit and reiterate the requirements based on insights from testing, supervisor feedback, and evolving project goals. The research phase might involve acquiring additional data sets and investigating new research advancements. Taking into account the findings from the previous increment, we'll optimize the system architecture, enhance the algorithms, and integrate feedback to improve the user interface. This phase also plays a more critical role in evaluation and improvement, quantifying the system's

effectiveness and identifying opportunities for further refinement.

The fourth increment marks the deployment, monitoring, and continuous improvement stage. The requirement analysis becomes an ongoing process, adapting to evolving business needs and incorporating user feedback. We'll continuously benefit from new research findings allowing for ongoing adaptation. Finally, the evaluation and improvement become long-term endeavors, monitoring the system's performance, gathering user feedback, and continuously improving the system.

3.3 Use Case Model

The Use Case Model for the FoodFind project provides a structured representation of the interactions between users and the system. This section outlines the primary actors, their associated use cases, and the detailed scenarios that describe these interactions. The primary actors identified for the system are Users and Admins.

3.3.1 Actors

- **User**: Represents the end-users of the FoodFind platform who interact with the system to search for, view, review and add restaurants as their favourite.
- **Admin**: Represents the system administrators who manage the restaurant data, user accounts, and ensure the overall system integrity.

3.3.2 Use Cases for Users

a. Login

- **Description**: The user logs into the FoodFind system.
- **Precondition**: The user must have an account with FoodFind.
- **Postcondition**: The user gains access to the system's features.

b. Search Restaurant

- **Description**: Allows users to search for restaurants by name and location.
- **Precondition**: User is logged in or using the app without login (if allowed).
- **Postcondition**: Displays a list of restaurants matching the search criteria.

c. View Restaurant

- **Description**: Users can view details about different restaurants, including menus, prices, and available food items.
- **Precondition**: A restaurant has been selected from search results or another method of selection.

- **Postcondition**: Detailed information about the restaurant is displayed to the user.

d. Read Reviews

- **Description**: Users can read reviews provided by other users for various restaurants.
- **Precondition**: Reviews exist for a given restaurant.
- **Postcondition**: Reviews are displayed to help inform users' choices.

e. Add Reviews

- **Description**: Users can provide reviews for restaurants they visit.
- **Precondition**: The user has visited a restaurant.
- **Postcondition**: The review is added to the restaurant's profile.

f. Add Favorite Restaurant to User Profile

- **Description**: Users have the option to add their favorite restaurants to their profiles for easy access. This feature allows users to bookmark and revisit their preferred dining establishments without the need to search for them again.
- **Precondition**: The user is logged in.
- **Postcondition**: The restaurant is added to the user's profile for future reference.

3.3.3 Use Case Diagram for Users

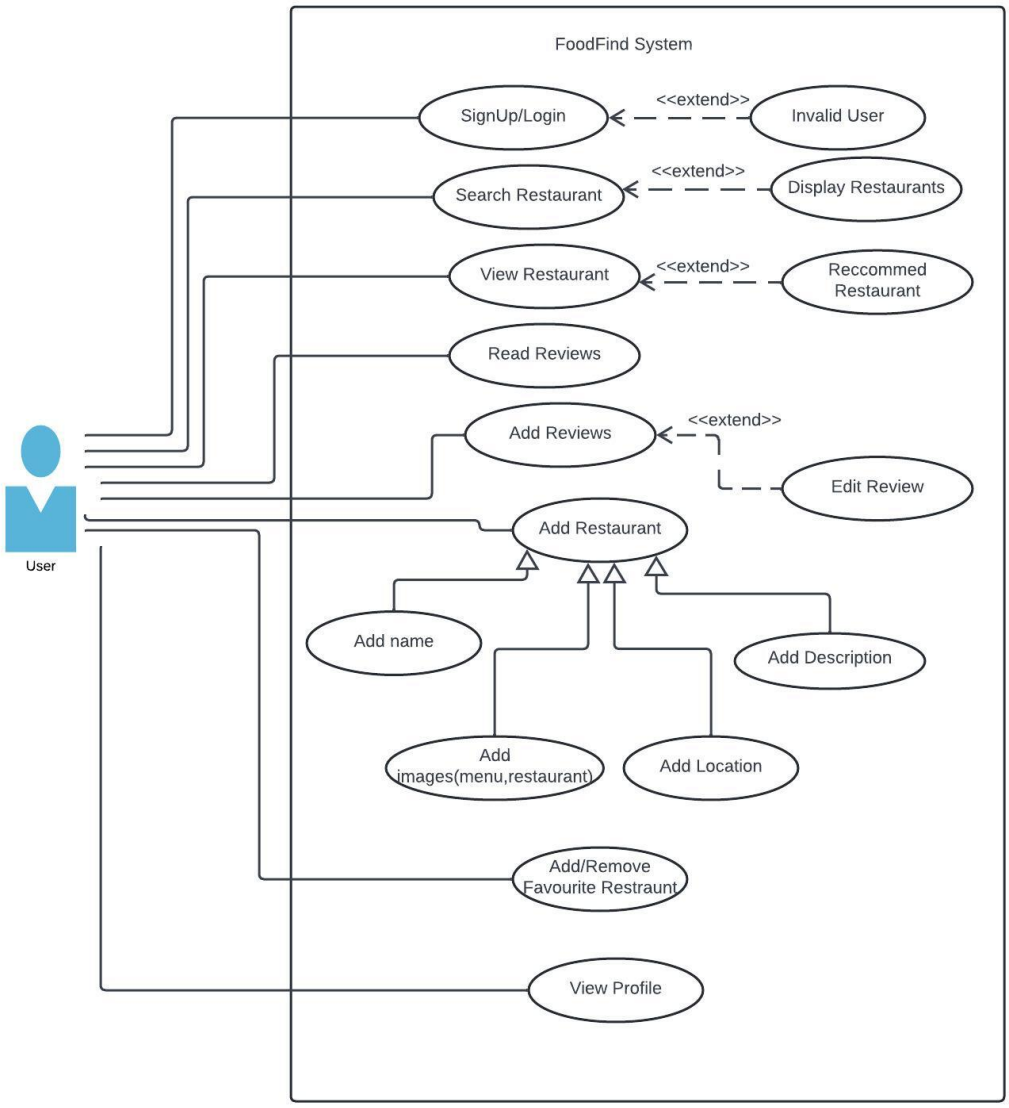


Figure 3: Use Case Diagram for Users

3.3.4 Use Cases for Admin

a. Login

- **Description**: Admin authenticates themselves to gain access to the system.
- **Precondition**: Admin must have valid credentials.
- **Postcondition**: Admin gains access to the system functionalities.

b. Add Restaurants

- **Description**: Admin adds new restaurant details into the system.
- **Precondition**: Admin must be logged in.
- **Postcondition**: New restaurant details are added into the system.

c. Verify Add Restaurant Request

- **Description**: Admin reviews and verifies requests for adding new restaurants.
- **Precondition**: There must be pending requests for adding restaurants.
- **Postcondition**: Requests are either approved or denied, updating the status of restaurant addition requests.

d. Update Restaurants

- **Description**: Admin modifies existing restaurant details in the system.
- **Precondition**: Restaurant details already exist in the system; admin must be logged in.
- **Postcondition**: Updated information of restaurants is saved in the system.

e. Delete Restaurants

- **Description**: Admin removes existing restaurant listings from FoodFind.
- **Precondition**: Restaurant listings exist; admin has authenticated access.
- **Postcondition**: Restaurant listings no longer appear on FoodFind.

f. Manage Users

- **Description**: Admin oversees user accounts, possibly editing or removing them.
- **Precondition**: Users have accounts within FoodFind; admin has authenticated access.
- **Postcondition**: User account data is updated or removed based on admin actions.

3.3.5 Use Case Diagram for Admin

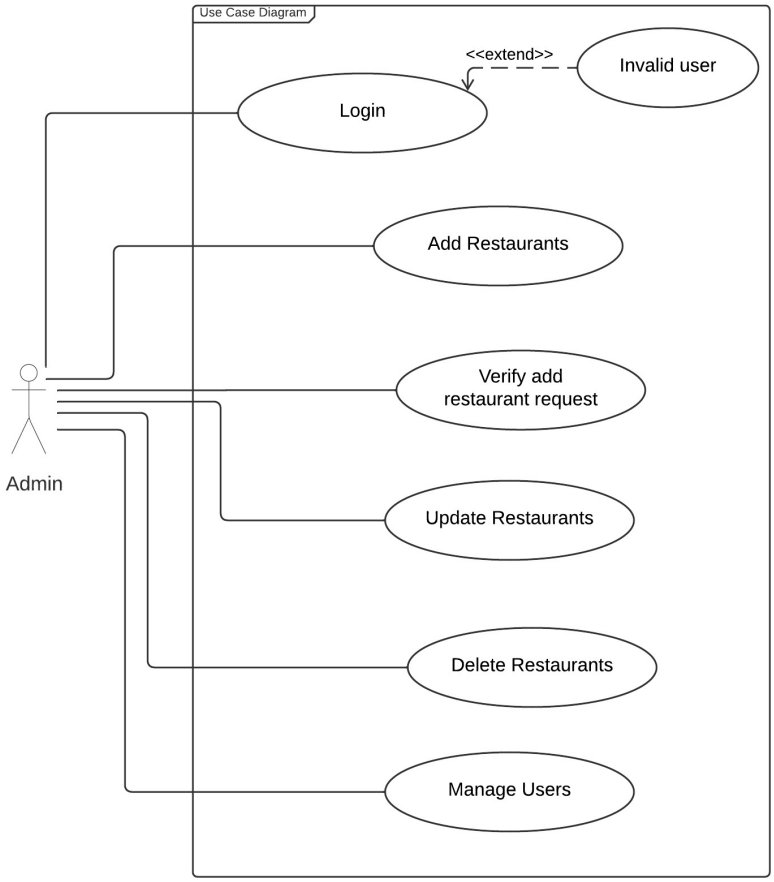


Figure 4: Use Case Diagram for Admin

3.4 Sequence Diagram

A sequence diagram is a graphical representation that illustrates the interactions between objects or components within a system in a chronological order, aiding developers in understanding system behavior, identifying patterns, and validating design decisions.

The user interacts with the FoodFind System to perform various actions, such as logging in, searching for restaurants, viewing menus, adding reviews, and sharing restaurant locations.

The system communicates with the Database to check user details during login, query restaurant data, and update information based on user reviews or newly added restaurants.

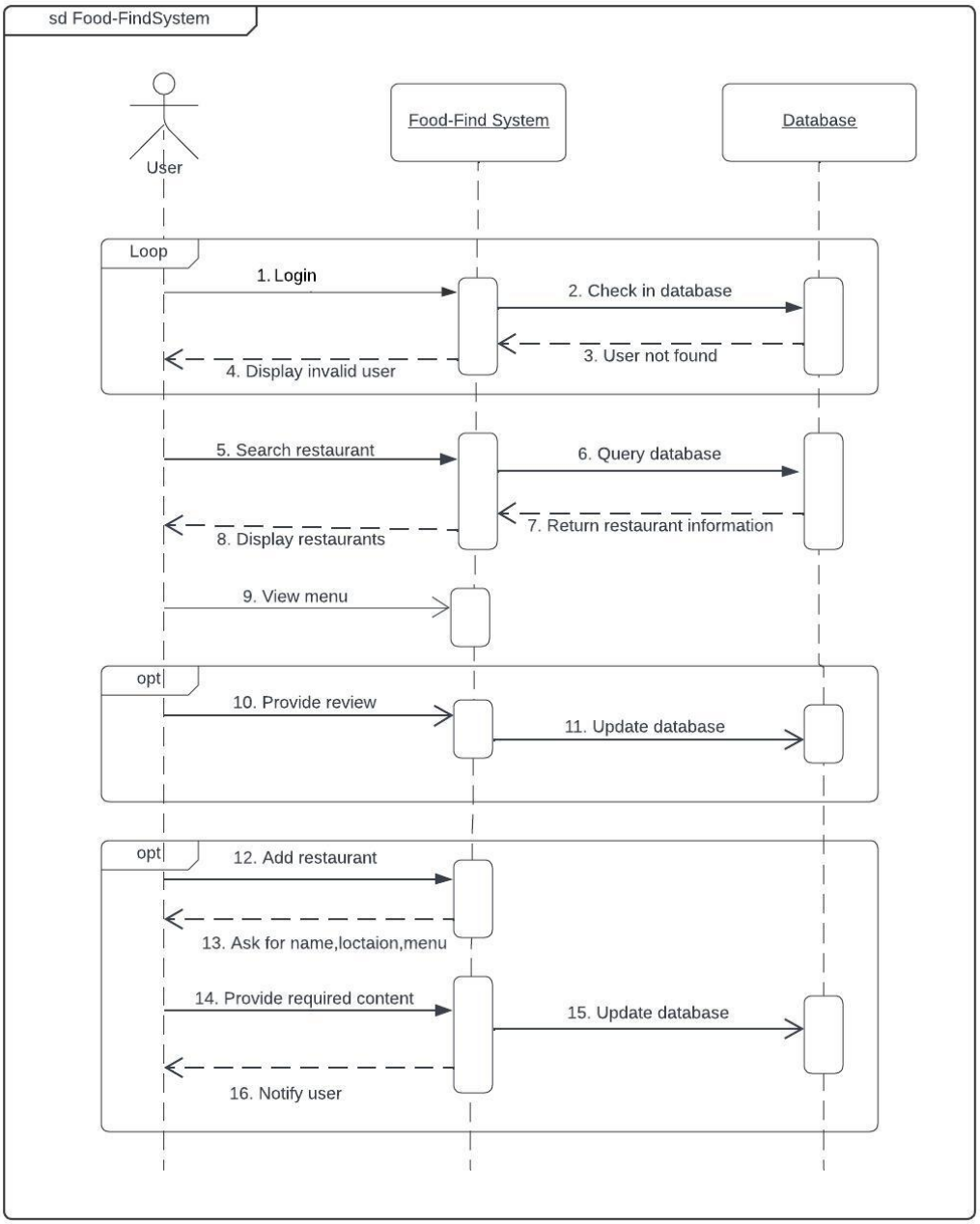


Figure 5: Sequence Diagram

3.5 Database Diagram

The diagram is data model for FoodFind database. It outlines the structure and relationships between different entities in the database. Here's a brief explanation of each entity and its attributes

3.5.1 Entities and Their Attributes

- **Customers**
  - **customer_id**: Primary Key, INT, NOT NULL
  - **username**: VARCHAR(150), NOT NULL
  - **email**: VARCHAR(254), UNIQUE, NOT NULL
  - **google_id**: VARCHAR(255), UNIQUE, NOT NULL
  - **profile_picture**: VARCHAR(200)
  - **favourite_restaurants**: Many-to-Many relationship with **Restaurant**, NOT NULL
- **Review**
  - **id**: Primary Key, INT, NOT NULL
  - **user_id**: Foreign Key referencing **customer_id**, INT, NOT NULL
  - **restaurant_id**: Foreign Key referencing **restaurant_id**, INT, NOT NULL
  - **rating**: INT, NOT NULL
  - **review_text**: TEXT
  - **created_at**: DATETIME, NOT NULL
  - **updated_at**: DATETIME, NOT NULL
- **Restaurant**
  - **restaurant_id**: Primary Key, INT, NOT NULL
  - **name**: VARCHAR(255), NOT NULL
  - **location**: VARCHAR(255), NOT NULL
  - **price**: VARCHAR(100), NOT NULL
  - **opening_hours**: VARCHAR(100), NOT NULL
  - **description**: TEXT, NOT NULL
  - **map_url**: VARCHAR(500), NOT NULL

- **Menu**
  - **id**: Primary Key, INT, NOT NULL
  - **restaurant_id**: Foreign Key referencing **restaurant_id**, INT, NOT NULL
  - **name**: VARCHAR(255), NOT NULL
  - **price**: DECIMAL(10, 2), NOT NULL
  - **category**: VARCHAR(20), NOT NULL
- **Tags**
  - **id**: Primary Key, INT, NOT NULL
  - **name**: VARCHAR(50), UNIQUE, NOT NULL
- **TopRestaurant**
  - **id**: Primary Key, INT, NOT NULL
  - **restaurant_id**: Foreign Key referencing **restaurant_id**, INT
  - **ranking**: INT, UNIQUE, NOT NULL
- **RestaurantImage**
  - **id**: Primary Key, INT, NOT NULL
  - **restaurant_id**: Foreign Key referencing **restaurant_id**, INT, NOT NULL
  - **image**: VARCHAR(100), NOT NULL

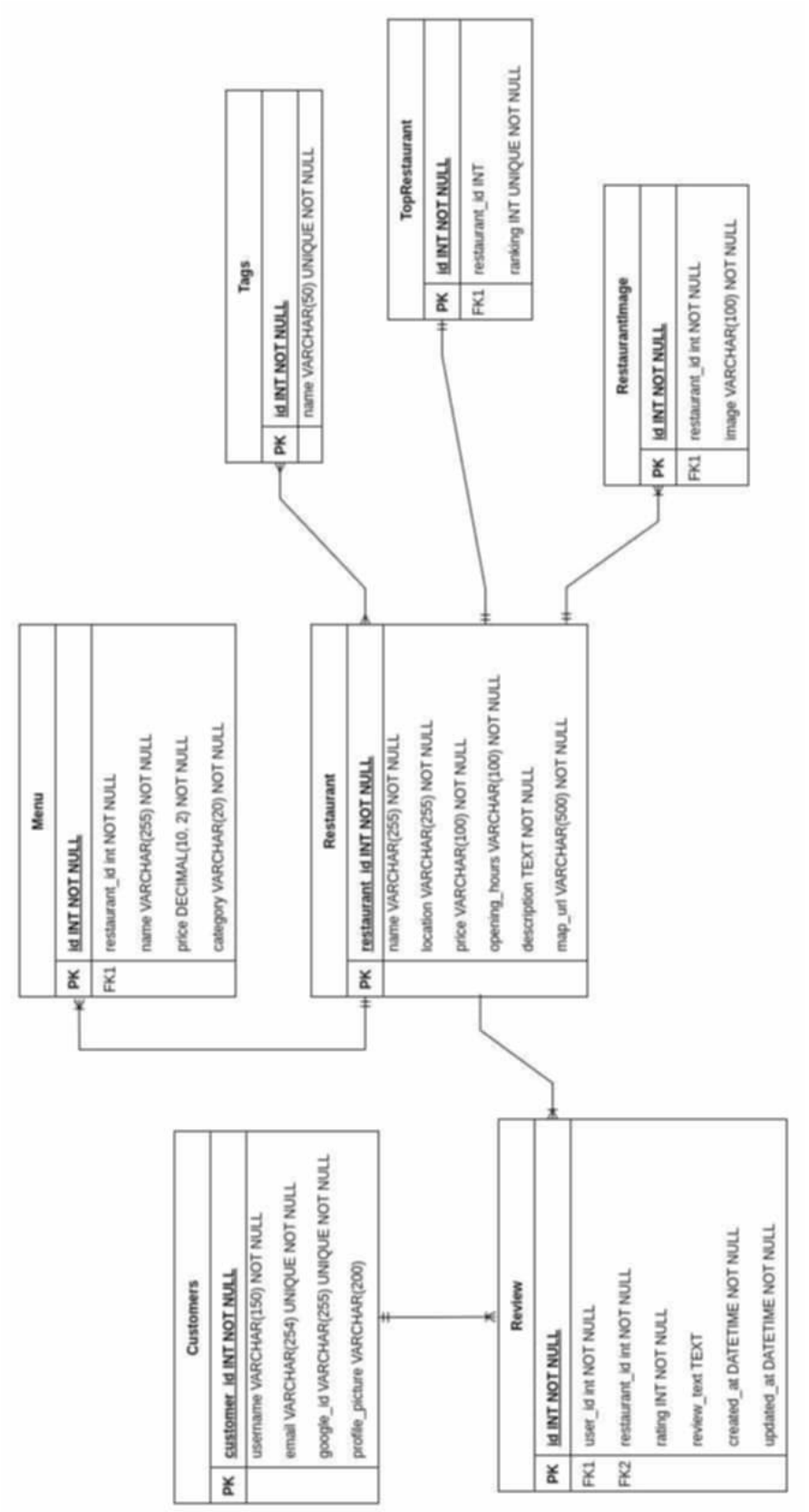


Figure 6: Database Diagram

3.5.2 Relationships

- **Customers to Review**: One customer can write multiple reviews (**user_id** in Review references **customer_id** in Customers).
- **Restaurant to Review**: One restaurant can have multiple reviews (**restaurant_id** in Review references **restaurant_id** in Restaurant).
- **Restaurant to Menu**: One restaurant can have multiple menu items (**restaurant_id** in Menu references **restaurant_id** in Restaurant).
- **Restaurant to Tags**: Many-to-many relationship implied (not explicitly shown but generally implemented with a join table).
- **Restaurant to TopRestaurant**: One restaurant can be listed as a top restaurant with a unique ranking (**restaurant_id** in TopRestaurant references **restaurant_id** in Restaurant).
- **Restaurant to RestaurantImage**: One restaurant can have multiple images (**restaurant_id** in RestaurantImage references **restaurant_id** in Restaurant).

This ERD represents the database schema for managing customers, restaurants, reviews, menus, tags, top restaurant rankings, and restaurant images.

3.6 Activity Diagram

An activity diagram for FoodFind represents the user journey and interactions within the restaurant recommendation application. It shows the sequence of activities, such as signing up, logging in, searching, filtering, and reviewing restaurants, as well as decisions like whether to register or add a new restaurant. This helps in understanding how users navigate through the application and the flow of various functionalities.

3.6.1 Overall User Activity

The following activity diagram illustrates the user flow within the FoodFind System:

1. **SignUp/Login**: Users start by signing up or logging in if already registered.
2. **Registration**: New users enter details to create an account.
3. **Login Validation**: Credentials are validated; successful login leads to viewing restaurants.
4. **View/Search Restaurants**: Users can view all restaurants or search for specific ones.
5. **Filer Restaurants**: Users filter search results by food type or tags.
6. **View Details/Results**: Users view detailed information about selected restaurants.
7. **Provide Review**: Users can leave reviews for restaurants.
8. **Request to Add Restaurant**: If a restaurant is not listed, users can request to add it by providing details.

The diagram effectively maps the decision points and actions a user takes within the FoodFind app.

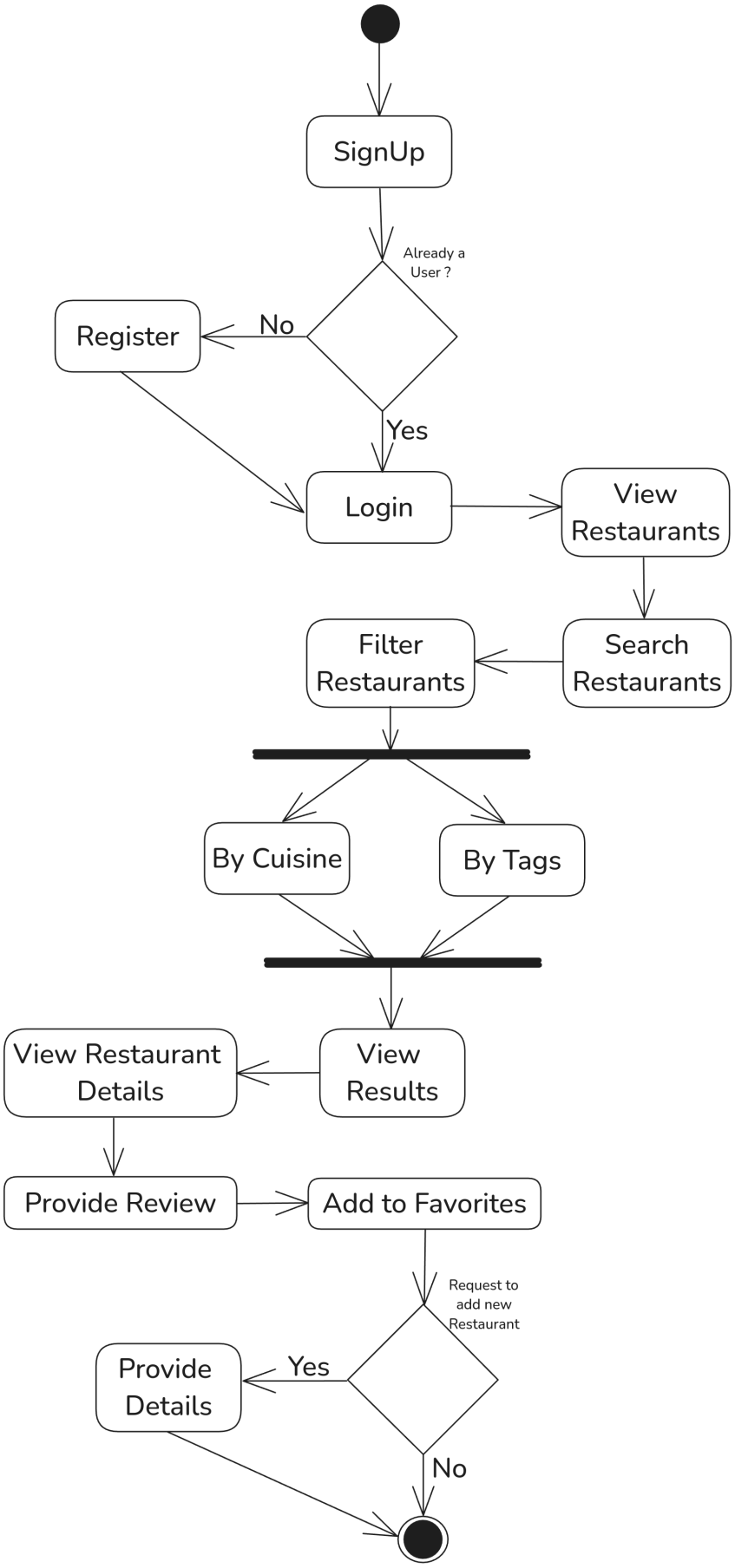


Figure 7: Activity Diagram for User

3.6.2 Providing Review

Figure 8 illustrates the user's process of selecting a restaurant, writing a review, and the subsequent update to the restaurant's profile with the new review.

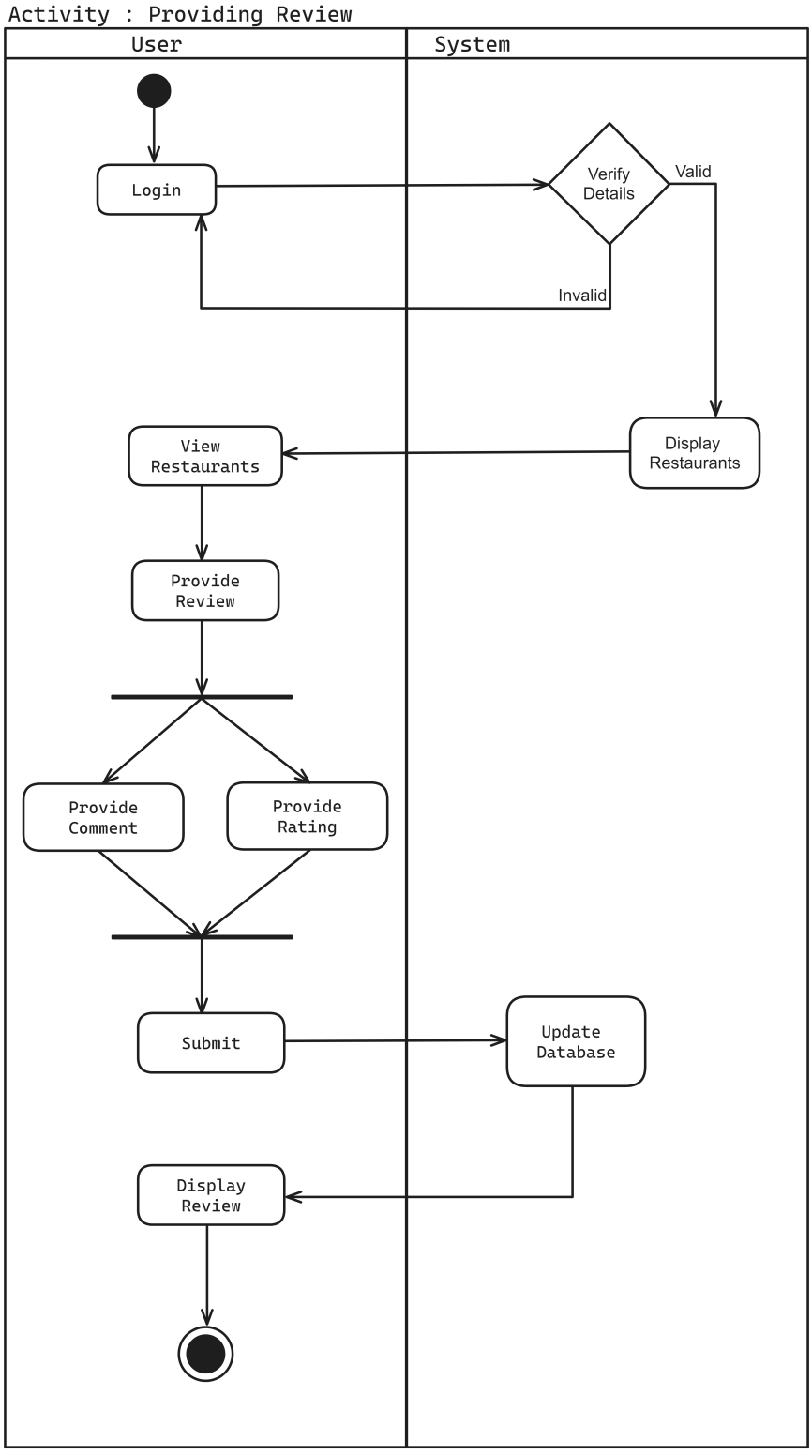


Figure 8: Activity Diagram for Providing ReviEW

3.6.3 Adding a Restaurant

Figure 9 illustrates the user's request to add a new restaurant and the subsequent approval process handled by an admin, ensuring quality control before the listing goes live on FoodFind.

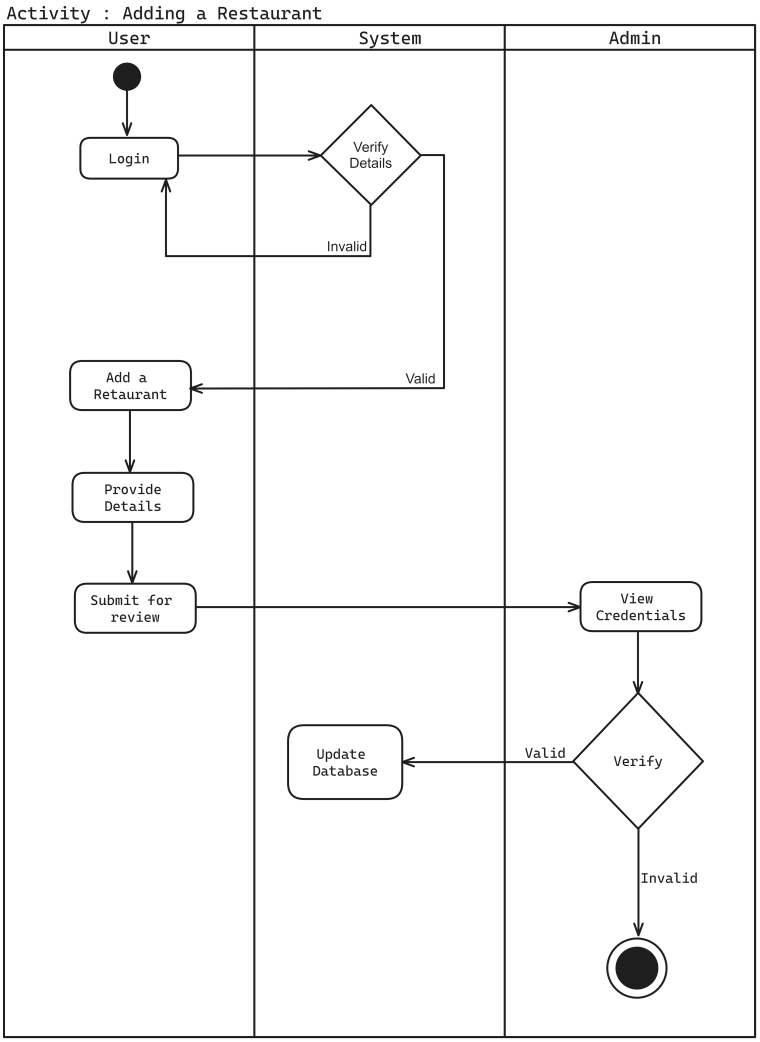


Figure 9: Activity Diagram for Adding Restaurants

3.7 System Flow

In context of FoodFind, below Figure **??** illustrates a streamlined web development architecture where TypeScript is the core language for the frontend, transpiled by Vite and styled with Tailwind CSS and Daisy UI for a responsive design.

The frontend communicates with the backend via a REST API, which is powered by Django Restful API, utilizing Django ORM for object-relational mapping and SQLite for database management. This setup ensures a robust, scalable, and maintainable web application framework.

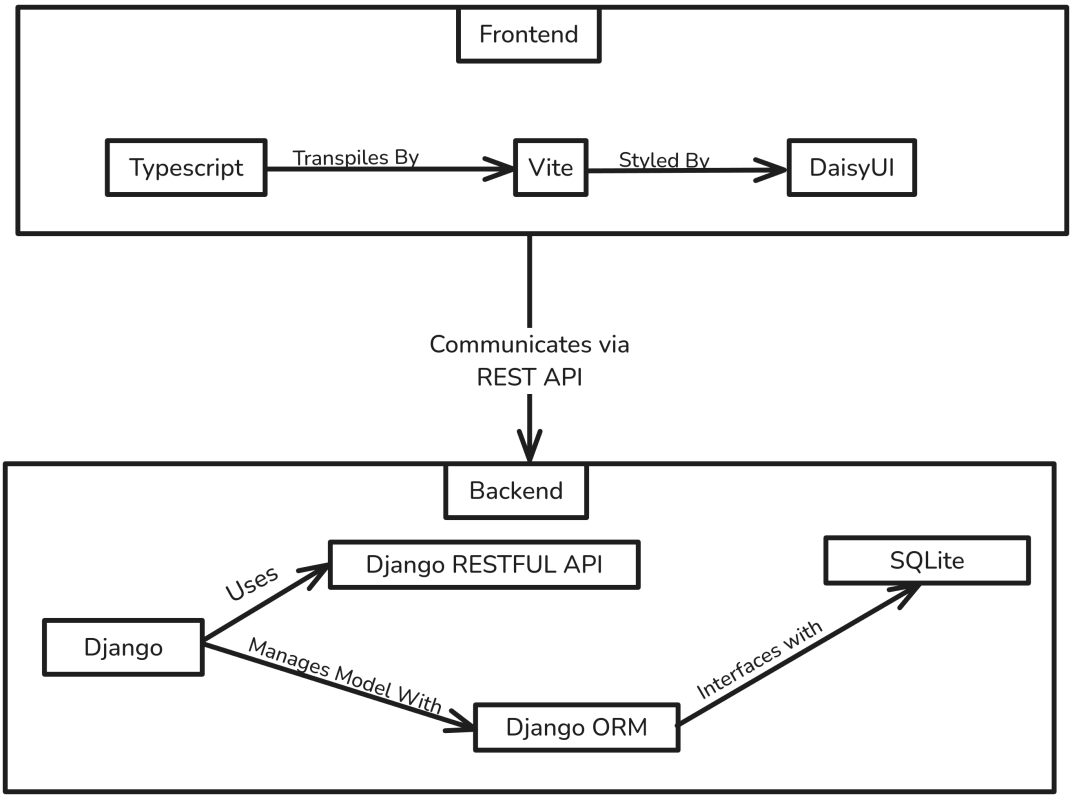


Figure 10: System Flow Diagram

3.8 Component Diagram

The component diagram illustrates the architectural structure of the FoodFind system by showing the components and their relationships. Here's a brief explanation of the key components and their interactions:

Components

- **CuisineModel, RestaurantModel, ReviewsModel, AdminModel**
  - These are the core models representing different entities in the FoodFind system.
  - They handle the data and logic related to cuisines, restaurants, reviews, and admin functionalities, respectively.
- **ORM (Object-Relational Mapping)**
  - This component manages database interactions.
  - It converts data between incompatible type systems in object-oriented programming and relational databases.
- **Views**
  - This component is responsible for rendering the user interface elements.
  - It interacts with the models to fetch data and present it to the users.
- **ApiEndpoints**
  - These endpoints serve as the interface for the frontend to communicate with the backend.
  - They handle requests and responses, ensuring data is correctly transferred between the frontend and backend.
- **FrontendApplication**
  - This component represents the main application that users interact with.
  - It includes various React UI components to create an interactive user interface.
- **ReactUIComponents**
  - These are the individual components that make up the user interface in the frontend application.

  - They ensure a modular and reusable structure for building the UI.

- **BrowserApi**
  - This component allows interaction with the web browser's capabilities.
  - It can handle tasks such as making HTTP requests, storing data locally, etc.

- **RecommendationEngine**
  - This component is responsible for generating recommendations for users.
  - It uses various data points and algorithms to suggest restaurants or cuisines.

- **Serializers**
  - These components handle the conversion of complex data types, such as querysets and model instances, into native Python datatypes that can be easily rendered into JSON or other content types.
  - They ensure data is correctly formatted when sent between the server and the client.

Interactions

- **DataObjects**: These are used to transfer data between the ORM and Views components.
- **RecommendationObject**: This is used by the RecommendationEngine to process and generate recommendations.
- **SerializationMethod**: Utilized by the Serializers component to format data for API responses.

This diagram provides a high-level overview of how different parts of the FoodFind system interact with each other to deliver cohesive functionality, from handling data at the backend to presenting it in the frontend.

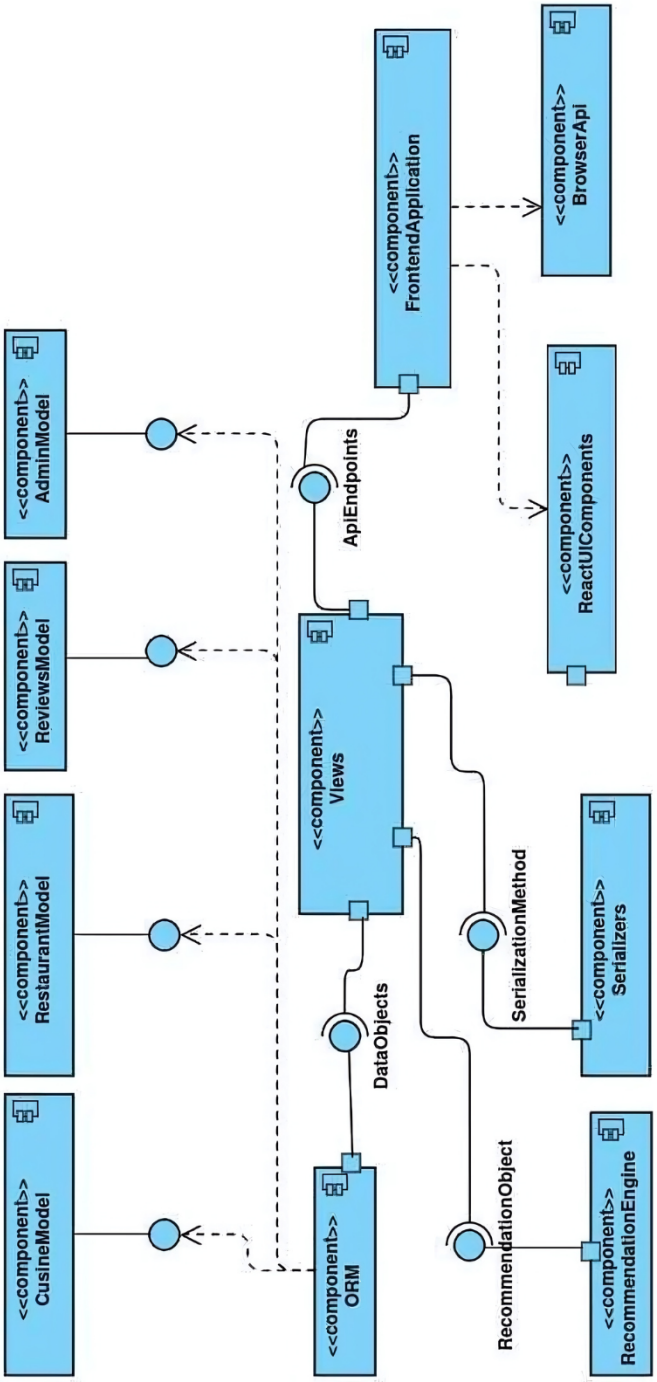


Figure 11: Component Diagram

4. Technologies Used

Table 1 consists of the major technologies that are being used for development and deployment of the application. They are briefly described in the subsections that follow.

Subject	**Tools and Technologies Used**
Backend Database	SQLite
REST API Service	Django REST Framework; Postman
Web Application	Vite; TypeScript
IDE / Code Editor	Sublime Text; Code-OSS
Version Control System	GitHub
Documentation	Overleaf

Table 1: Technologies Being Used

4.1 SQLite

SQLite provides a reliable and efficient database solution for storing user profiles, restaurant details, and reviews. Its serverless architecture is ideal for the FoodFind app, which requires a lightweight database that can handle high traffic and complex queries with ease.

4.2 Django REST Framework

The Django REST Framework will serve as the backbone for the FoodFind API, handling data serialization and request processing. It will enable seamless communication between the app's frontend and the SQLite database, ensuring a smooth user experience.

4.3 Postman

Postman will be utilized during development to test and refine the FoodFind API. It will help ensure that the API endpoints return the correct data and handle errors

properly, which is crucial for the reliability of the app.

4.4 Vite

Vite will accelerate the development process of FoodFind by providing a fast and modern build system. Its hot module replacement feature will allow developers to see changes in real-time, enhancing productivity.

4.5 TypeScript

TypeScript will be used to write scalable and maintainable code for the FoodFind web app. Its static typing system will help catch errors early in development, leading to a more robust and error-free application.

4.6 Sublime Text

Sublime Text will be the code editor of choice for developers working on FoodFind due to its speed and efficiency. Its vast array of plugins will aid in writing clean and optimized code for the project.

4.7 Code-OSS

Code-OSS will provide a comprehensive development environment for the FoodFind project. Its support for debugging, version control, and extensions will contribute to an efficient development workflow.

4.8 GitHub

GitHub will be essential for version control and collaboration among the FoodFind development team. It will track changes, manage code contributions, and host the project's repository, facilitating teamwork and progress tracking.

4.9 Overleaf

Overleaf will be used to create and maintain the FoodFind project's documentation. Its real-time collaboration features will allow multiple contributors to work on the documentation simultaneously, ensuring it stays up-to-date and comprehensive.

5. Recommendation Algorithms

The recommendation system in the FoodFind project is designed to provide personalized restaurant suggestions to users based on their preferences and interactions. The algorithm employs a hybrid approach, combining collaborative filtering and content-based filtering techniques

5.1 Data Retrieval

The algorithm starts by retrieving relevant data based on user interactions and specified parameters:

- **User ID**: Identifies the user for whom recommendations are to be generated.
- **Recommendation Flag**: Determines if recommendations are requested.
- **Tags**: Filters restaurants based on specified tags.

5.2 Collaborative Filtering

Collaborative filtering generates recommendations by identifying similarities between users. This involves:

- **User Favorites and Reviews**: Collecting the list of favorite restaurants and reviews for the specified user.
- **Identifying Similar Users**: Finding other users who share similar tastes by checking for overlapping favorite restaurants and reviewed restaurants.
- **Generating Collaborative Recommendations**: Recommending restaurants that are liked or reviewed by these similar users but not by the current user.

5.3 Content-Based Filtering

Content-based filtering uses the attributes of restaurants to find items similar to those the user already likes:

- **User Tags**: Identifying tags associated with the user's favorite restaurants.

- **Generating Content Recommendations**: Selecting restaurants that match these tags, ensuring they share characteristics with those that the user has already shown interest in.

5.4 Combining Recommendations: Hybrid Filtering

The algorithm combines the results from collaborative filtering and content-based filtering:

- **Union of Recommendations**: Merging the collaborative and content-based recommendations to form a unique set of suggested restaurants.

5.5 Annotation and Sorting

To present the most relevant recommendations, the algorithm:

- **Annotates Recommendations**: Adds metadata such as average ratings and the number of reviews to each restaurant.
- **Sorting**: Orders the restaurants by average rating and review count, ensuring the highest-rated and most-reviewed options are prioritized.

5.6 Handling No Recommendations

If the algorithm does not generate any specific recommendations, it defaults to returning the entire list of available restaurants, ensuring that the user always receives some suggestions.

6. Performance Analysis Methodology

In order to make this project effective and robust, we will be making sure it adheres to top notch quality by testing it with different performance metrics, validation scheme and benchmark tests.

6.1 Data Management

A detailed database of restaurants and cafes will be manually compiled, including information such as menus, special deals, and events. This database will not only facilitate efficient data access but also enable comprehensive analysis of dining trends and customer preferences.

6.2 API Development

A robust API service will be developed using the Django REST Framework, with testing conducted through Postman to ensure seamless data retrieval and display. This API will dynamically generate profiles for each restaurant, pulling real-time data from the backend to provide up-to-date information without the need for manual profile creation.

6.3 Web Application Functionality

The front-end will be crafted using Vite and TypeScript, focusing on a responsive and interactive user experience. Emphasis will be placed on creating an intuitive design that adapts to different devices, ensuring accessibility and ease of use for all users.

7. Risk Analysis

Risk analysis for the FoodFind project involves identifying potential risks, assessing their impact, and devising strategies to mitigate them. This section covers both technical and operational risks.

7.1 Technical Risks

Risk	**Description**	**Mitigation**
Security Vulnerabilities	Unsecured API endpoints may expose the application to data breaches, unauthorized access, and data manipulation.	Implement secure API practices such as HTTPS, token-based authentication, and regular security audits.
Scalability Issues	The initial architecture may not handle increased user load as the app gains popularity.	Design the system with scalability in mind, use load balancers, and implement efficient database indexing and caching strategies.
Data Accuracy and Consistency	Inaccurate or inconsistent data about restaurants could lead to user dissatisfaction.	Implement rigorous data validation processes and regular data audits to ensure accuracy and consistency.
Performance Bottlenecks	High traffic or complex queries could lead to slow response times.	Optimize the code, use performance monitoring tools, and conduct regular load testing to identify and resolve bottlenecks.
Technical Debt	Accumulation of quick fixes and suboptimal code may hamper future development and maintenance.	Follow best coding practices, conduct regular code reviews, and allocate time for refactoring.

Table 2: Technical Risks

7.2 Operational Risks

Risk	**Description**	**Mitigation**
User Engagement and Participation	Low user engagement may affect the quality and quantity of reviews and recommendations.	Implement user engagement strategies such as rewards for active participation, regular updates, and interactive features.
Resource Constraints	Limited resources (time, budget, personnel) may affect the development and operational aspects.	Prioritize tasks, allocate resources efficiently, and seek additional funding or partnerships if necessary.
Regulatory Compliance	Non-compliance with local laws and regulations could result in legal issues.	Stay informed about relevant laws and regulations, and ensure the app complies with all legal requirements.
Technological Changes	Rapid changes in technology may render some components of the app obsolete.	Stay updated with technological trends, be prepared to adapt, and invest in continuous learning and development.

Table 3: Operational Risks

8. Test Cases

This section outlines the test cases for the FoodFind web application. The scope of testing includes functionality related to searching, reviewing, and marking restaurants as favorites.

8.1 Features to be Tested

- Restaurant profiles
- Review and rating
- Add and remove favorite restaurants
- Map section for searching and suggesting nearby restaurants
- Restaurant addition by users
- Filtering restaurants based on tags
- Admin panel functionalities
- User profile section
- Google authentication login

8.2 Test Case Format Description

Each test case is described with the following details:

- **Test Case ID** - Unique identifier for the test case.
- **Description** - Overview of what is being tested.
- **Test Steps** - Steps to execute the test.
- **Expected Result** - Expected outcome of the test.
- **Actual Result** - Actual outcome of the test.

8.3 Test Cases

8.3.1 Restaurant Profiles

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC1	View restaurant details	• Navigate to the search page. • Enter the restaurant name and search. • Click on a restaurant to view details.	Restaurant details are displayed correctly.	Restaurant details displayed as expected, including name, address, and contact information.
TC2	View restaurant menu	• Click on the 'Menu' tab.	The menu for the restaurant is displayed with prices and items.	Menu displayed correctly with all items and prices listed.

Table 4: Test Cases for Restaurant Profiles

8.3.2 Review and Rating

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC3	Add a review to a restaurant	• Click on the 'Write Review' button. • Enter review text and rating. • Submit the review.	Review is saved and displayed under the restaurant profile.	Review was successfully submitted and is visible on the restaurant's profile with correct text and rating.
TC4	View reviews for a restaurant	• Scroll to the reviews section.	User reviews are displayed with ratings.	Reviews are displayed with correct user names, text, and ratings.

Table 5: Test Cases for Review and Rating

8.3.3 Add and Remove Favorite Restaurant

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC5	Add restaurant to favorites	• Click on the 'Add to Favorites' button.	Restaurant is added to the user's favorite list.	Restaurant was successfully added to the favorite list and appears under the 'Favorites' section.
TC6	Remove restaurant from favorites	• Click on the 'Remove from Favorites' button.	Restaurant is removed from the user's favorite list.	Restaurant was successfully removed from the favorite list and no longer appears under 'Favorites'.

Table 6: Test Cases for Adding and Removing Favorite Restaurants

8.3.4 Map Section

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC7	Automatically display nearby restaurants on the map	• Open the map page.	The map automatically shows restaurants within a 3 km radius along with routes and distances from the user's current location.	Nearby restaurants were displayed correctly on the map with accurate routes and distances.
TC8	Search for a specific restaurant on the map	• Enter the restaurant name in the search bar. • Select the desired restaurant from suggestions.	The route and distance to the selected restaurant from the user's current location are displayed on the map.	Route and distance to the selected restaurant were displayed accurately on the map.

Table 7: Test Cases for Map Section

8.3.5 Restaurant Addition by Users

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC9	Add a new restaurant	• Click on the 'Add Restaurant' button. • Fill in the restaurant details (name, location, menu, etc.). • Submit the form.	Restaurant data is submitted to the admin for review.	Restaurant was successfully added and appears in search results with correct details.

Table 8: Test Cases for Restaurant Addition by Users

8.3.6 Filtering Restaurants Based on Tags

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC10	Filter restaurants by tag	• Go to the filter section. • Select a tag from the available options. • Apply the filter.	Restaurants matching the selected tag are displayed.	Restaurants filtered correctly based on the selected tag and displayed as expected.

Table 9: Test Cases for Filtering Restaurants Based on Tags

8.3.7 Admin Panel Functionalities

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC11	Admin access to restaurant and user management	• Log in as an admin. • Navigate to the restaurant or user management section. • View, edit, or delete restaurant or user entries.	Admin can view, edit, and delete restaurant or user details.	Admin access works as intended; restaurant and user entries can be managed correctly.

Table 10: Test Cases for Admin Panel Functionalities

8.3.8 User Profile Section

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC12	View and edit user profile	• Log in as a user. • Go to the user profile section. • View the user info, reviews provided and the favorite restaurants.	User profile details are displayed.	User profile section displayed.

Table 11: Test Cases for User Profile Section

8.3.9 Google Authentication Login

Test Case ID	Description	Test Steps	Expected Result	Actual Result
TC13	Google authentication login	• Go to the login page. • Click on 'Login with Google'. • Authenticate with Google credentials.	User is logged in and redirected to the homepage.	Google authentication worked; user logged in successfully and redirected to the homepage.

Table 12: Test Cases for Google Authentication Login

9. Project Task and Time Schedule

The working time period for the project was four months, spanning the entire duration of the spring semester. The project was structured to ensure all tasks were completed in a timely manner, adhering to the university's requirements. The project timeline was divided into various phases, including planning, development, testing, and deployment. Each phase had specific milestones that the team aimed to achieve within set deadlines. This structured approach ensured efficient progress and timely completion of the project..

9.1 Division of Roles and Responsibilities

The success of the project heavily relied on the effective collaboration of all team members, each assigned specific roles and responsibilities. The roles were divided based on the expertise and skills of each member to ensure optimal contribution to the project. Table 13 outlines the division of roles and responsibilities among the team members.

S.N.	Team Member	Role	Responsibilities
1.	Pradip Dhungana	Backend Developer	• Admin Panel and develop API endpoints to handle requests from and send responses to the web application. • Manage the database and handle documentation.
2.	Vision Rijal	Frontend Developer	• Prepare the User Interface (UI) of the web application. • Manage the project team and guide the team along various phases of project development.
3.	Bishnu Timilsena	Backend Developer	• Develop API endpoints for dynamic pages. Test and ensure validity and security of API endpoints. • Handle data collection tasks.

Table 13: Division of Roles and Responsibilities of Team Members

9.2 Gantt Chart

The time schedule for the development of the project is illustrated in the following figures.

Figure 12: Gantt Chart for increment 1

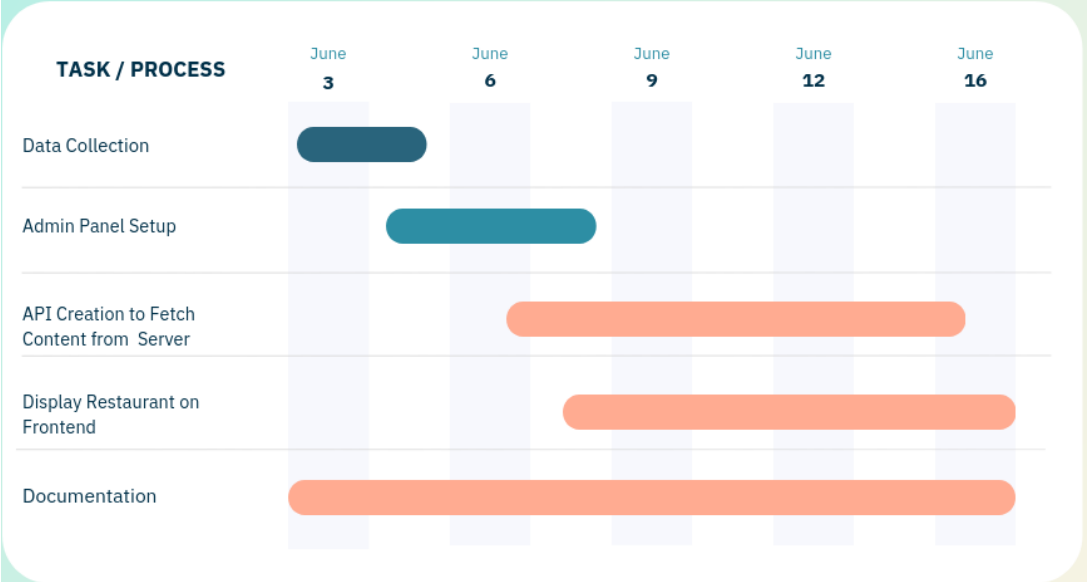


Figure 13: Gantt Chart for increment 2

Figure 14: Gantt Chart for increment 3

Figure 15: Gantt Chart for increment 4

10. Deliverable/Output

As per our objective, we have developed a web-based platform, FoodFind, which helps restaurant owners to showcase their dining establishments and provides a platform for customers to discover and choose the best dining options. This platform allows restaurant owners to publish detailed information about their eateries, while offering customers the convenience of selecting their desired dining venue from the comfort of their homes, thus saving time and effort.

10.1 Search and Discovery

The primary feature of FoodFind is its robust search functionality. Users can effortlessly locate restaurants by entering a name or selecting a location. This search process is designed to be intuitive and accessible to all users, regardless of their technical expertise.

10.2 Filtering Options

The platform includes advanced filtering options, allowing users to refine their searches based on specific criteria such as cuisine type and location. This functionality makes it easier for users to find restaurants that meet their specific tastes and preferences.

10.3 Menu and Pricing Information

FoodFind offers comprehensive information on restaurant menus and pricing. This transparency allows users to make well-informed dining decisions based on their budget and food preferences, enhancing their overall dining experience.

10.4 User Reviews and Sharing

While FoodFind does not host internal community postings, it enables users to share their dining experiences on external platforms. This feature encourages community-driven recommendations and helps users make better dining choices.

10.5 AI-Powered Recommendations

I Provides personalized restaurant suggestions based on user preferences and past behavior, leveraging advanced machine learning algorithms to enhance recommendation accuracy and user satisfaction.

10.6 Personalized Favorite Restaurant Bookmarking

FoodFind offers a feature that allows users to add their favorite restaurants to their profiles for easy access. This functionality enables users to bookmark and revisit their preferred dining establishments without the need to search for them again, enhancing user convenience and engagement.

10.7 User-Friendly Interface

FoodFind boasts a user-friendly interface designed to provide a seamless browsing experience. The platform is optimized for both desktop and mobile use, ensuring accessibility across different devices.

10.8 Support for Local Establishments

FoodFind plays a crucial role in promoting local restaurants. By featuring these establishments, the platform helps increase their visibility, driving growth and fostering customer engagement, which is beneficial for the local economy.

10.9 Integration with Maps

FoodFind integrates with mapping services, allowing users to view restaurant locations and get directions easily. This feature adds a layer of convenience for users planning their dining experiences.

11. Conclusion

The FoodFind project has successfully met its objectives of developing a comprehensive web-based platform that enhances the dining experience in Kathmandu. By leveraging advanced search functionalities, detailed menu and pricing information, user reviews, and a user-friendly interface, FoodFind provides a robust solution for discovering and choosing dining options that best suit users' preferences.

The platform effectively supports local restaurants by increasing their visibility and customer engagement. Features such as search and discovery, filtering options, and integration with mapping services contribute to a seamless user experience, allowing users to make informed dining decisions effortlessly.

The successful completion of the FoodFind project demonstrates its potential to transform the dining landscape in Kathmandu. The platform is poised to become a valuable resource for both users seeking quality dining experiences and local establishments looking to reach a broader audience. We are confident that FoodFind will significantly contribute to a more connected and enjoyable dining community.

12. Future Enhancements

To ensure continuous improvement and adaptability, the following future enhancements are planned for FoodFind:

- **Reservation System:** Introducing an integrated reservation system that allows users to book tables directly through the platform.
- **Expanded Reviews:** Hosting community reviews to provide more comprehensive insights into dining experiences.
- **Real-Time Updates:** Offering real-time updates on restaurant availability and promotions to keep users informed about the latest dining options and special deals.

These enhancements will further solidify FoodFind's position as a leading platform for restaurant discovery and dining experiences, continually evolving to meet the needs of its users and the local restaurant community.

References

[1] K. PRASAIN, "Restaurants : High hopes, big challenges," 2017. [Online]. Available: https://www.newbusinessage.com/MagazineArticles/view/1730

[2] S. Lee, H. Park, and Y. Ahn, "The influence of tourists' experience of quality of street foods on destination's image, life satisfaction, and word of mouth: The moderating impact of food neophobia," *International Journal of Environmental Research and Public Health*, vol. 17, no. 1, 2020.

[3] S. Khadka, "As inflation soars, consumers are putting brakes on restaurant spending," january 29 2024. [Online]. Available: https://kathmandupost.com/money/2024/01/29/as-inflation-soars-consumers-are-putting-brakes-on-restaurant-spending

[4] M. Ghani and H. Suleman, "Restaurant recommendation system based on personalized preferences," *International Journal of Computer Applications*, vol. 175, no. 24, pp. 1–8, 2020.

[5] I. S. Pantelidis, "Social media and hospitality customers: A case study of restaurateurs in the uk," *International Journal of Contemporary Hospitality Management*, vol. 31, no. 2, pp. 892–908, 2019.

[6] E. Alomari, M. S. Aslam, and R. Alizadeh, "Mobile applications in the food industry in the kingdom of saudi arabia," *Journal of Foodservice Business Research*, vol. 22, no. 4, pp. 316–337, 2019.

[7] Y. Zhang, Q. You, C. Xu, and J. Sun, "Food recommendation and social network applications: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 192, p. 103140, 2021.

[8] P. Singh, R. Kapoor, and G. Sikka, "Restaurant recommender system based on food preferences," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 5, pp. 604–611, 2020.

[9] J. Serrano, S. Iglesias, and V. García-Díaz, "Food recommendation system prototype based on collaborative filtering," *Computers and Electronics in Agriculture*, vol. 169, p. 105196, 2020.

[10] D. Bianchini, V. De Antonellis, and M. Melchiori, "A restaurant recommendation system based on collaborative filtering and contextual information," *International Journal of Business Intelligence Research*, vol. 8, no. 3, pp. 1–17, 2017.

[11] I. Tanta, M. Miharja, and Q. Munajat, "Inclusivity of food review applications: User perspectives," *Journal of Contemporary Hospitality Management*, vol. 7, no. 3, pp. 1–7, 2021.

[12] B. Garner, "React on django: Getting started," Dec. 2021, [Online; accessed 17-May-2024]. [Online]. Available: https://bennettgarner.medium.com/react-on-django-getting-started-f30de8d23504

APPENDIX A

Endpoint: /restaurants/

Method: GET
Description: Retrieve a list of restaurants. If `recommend=true` is provided, it retrieves recommended restaurants for the specified `user_id`. If `tags` are provided, it filters restaurants based on tags.
Request Parameters: `recommend=true` and `user_id={user_id}` OR `tags=[tag1, tag2]`
Request Body: -
Response:


```
[ {"id": 1, "name": "Restaurant Name", "location": "Location",
"description": "Description", "images": [{"id": 1, "image":
"image_url"}]} ]
```


OR


```
{"images": [{"id": 1, "image": "image_url"}], "id":
1, "name": "Restaurant Name", "location": "Location",
"price": "Price", "opening_hours": "Opening Hours",
"description": "Description", "average_rating": 4.5,
"reviews": [{"id": 1, "user": {"id": 1, "username":
"User Name", "email": "user@example.com", "profile_picture":
"profile_picture_url"}, "rating": 5, "review_text": "Review
text", "created_at": "2024-07-25T05:33:38.644Z", "updated_at":
"2024-07-25T05:33:38.644Z", "restaurant_name": "Restaurant Name",
"restaurant_id": 1, "map_url": "Map URL", "menu_items": [{"id": 1, "name":
"Menu Item", "price": "Price", "category": "Category"}], "no_of_reviews": 10,
"tags": [{"id": 1, "name": "Tag"}]}
```


Endpoint: /restaurants/{id}/

Method: GET
Description: Retrieve details of a specific restaurant.
Request Parameters: -
Request Body: -
Response:


```
{"images": [{"id": 1, "image": "image_url"}], "id":
1, "name": "Restaurant Name", "location": "Location",
"price": "Price", "opening_hours": "Opening Hours",
"description": "Description", "average_rating": 4.5,
"reviews": [{"id": 1, "user": {"id": 1, "username":
```

"User Name", "email": "user@example.com", "profile_picture": "profile_picture_url"}, "rating": 5, "review_text": "Review text", "created_at": "2024-07-25T05:33:38.644Z", "updated_at": "2024-07-25T05:33:38.644Z", "restaurant_name": "Restaurant Name", "restaurant_id": 1, "map_url": "Map URL", "menu_items": [{"id": 1, "name": "Menu Item", "price": "Price", "category": "Category"}], "no_of_reviews": 10, "tags": [{"id": 1, "name": "Tag"}]}

Endpoint: /toprestaurants/

Method: GET
Description: Retrieve a list of top restaurants.
Request Parameters: -
Request Body: -
Response:
[{"id": 1, "restaurant": {"id": 1, "name": "Restaurant Name", "location": "Location", "description": "Description", "images": [{"id": 1, "image": "image_url"}]}, "ranking": 1}]

Endpoint: /tags/

Method: GET
Description: Retrieve a list of all tags.
Request Parameters: -
Request Body: -
Response:
[{"id": 1, "name": "Tag Name"}]

Endpoint: /google-login/

Method: POST
Description: Authenticate user via Google login.
Request Parameters: -
Request Body:
{"token": "Google OAuth Token"}
Response:
{"id": 1, "username": "User Name", "email": "user@example.com", "profile_picture": "profile_picture_url", "favorite_restaurants": [7, 8, 9]}

Endpoint: /create-review/

Method: POST
Description: Create or update a review for a restaurant.
Request Parameters: -
Request Body:

```
{"user": 1, "rating": 5, "review_text": "Great place!",
"restaurant": 1}
```

Response:

```
{"id": 1, "user": 1, "rating": 5, "review_text": "Great
place!", "created_at": "2024-07-25T05:33:38.644Z", "updated_at":
"2024-07-25T05:33:38.644Z", "restaurant_id": 1}
```

Endpoint: /user-reviews/{user$_id$}/

Method: GET
Description: Retrieve all reviews created by a specific user.
Request Parameters: -
Request Body: -
Response:

```
[ {"id": 1, "user": {"id": 1, "username": "User
Name", "email": "user@example.com", "profile_picture":
"profile_picture_url"}, "rating": 5, "review_text": "Great
place!", "created_at": "2024-07-25T05:33:38.644Z", "updated_at":
"2024-07-25T05:33:38.644Z", "restaurant_name": "Restaurant Name",
"restaurant_id": 1} ]
```

Endpoint: /add-to-favorites/

Method: POST
Description: Add a restaurant to the user's favorites.
Request Parameters: -
Request Body:

```
{"user_id": 1, "restaurant_id": 1}
```

Response:

```
{"message": "Restaurant added to favorites"}
```

Endpoint: /remove-from-favorites/

Method: POST
Description: Remove a restaurant from the user's favorites.
Request Parameters: -

Request Body:

{"user_id": 1, "restaurant_id": 1}

Response:

{"message": "Restaurant removed from favorites"}

Endpoint: /favorite-restaurants/{user$_id$}/

Method: GET

Description: Retrieve all favorite restaurants for a specific user.

Request Parameters: -

Request Body: -

Response:

[{"id": 1, "name": "Restaurant Name", "location": "Location"}
]

Endpoint: /add-restaurant/

Method: POST

Description: Add a new restaurant.

Request Parameters: -

Request Body:

{"user": 1, "name": "New Restaurant", "location": "Location",
"description": "Description", "opening_hours": "Opening Hours",
"price": "Price", "menu_images": [{"image": "image_url"}]}

Response:

{"user": 1, "name": "New Restaurant", "location": "Location",
"description": "Description", "opening_hours": "Opening Hours",
"price": "Price", "menu_images": [{"image": "image_url"}]}

Endpoint: /user-requested-restaurants/{user$_id$}/

Method: GET

Description: Retrieve all restaurants requested by a specific user.

Request Parameters: -

Request Body: -

Response:

[{"user": 1, "name": "Requested Restaurant", "location":
"Location", "description": "Description", "opening_hours":
"Opening Hours", "price": "Price", "menu_images": [{"image":
"image_url"}]}]

APPENDIX B

OUTPUTS

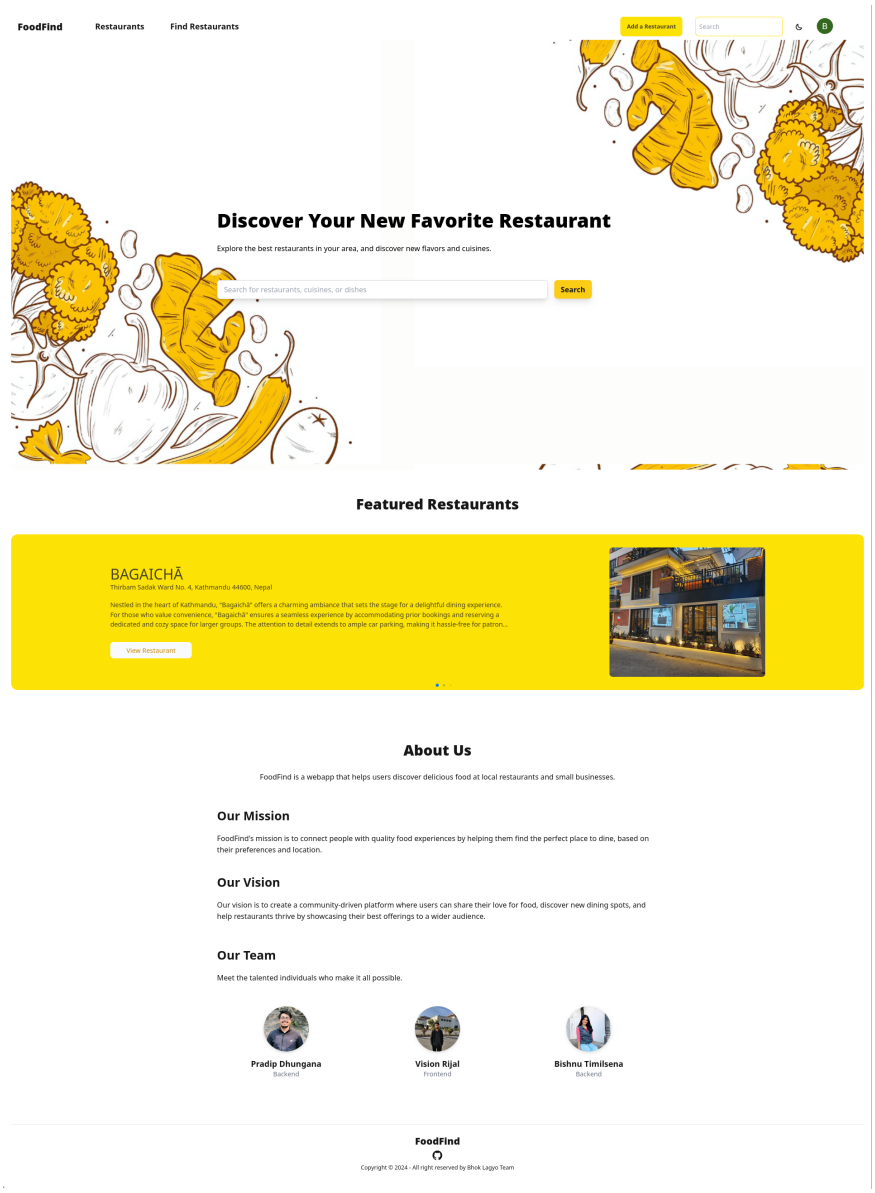


Figure 16: Home Page

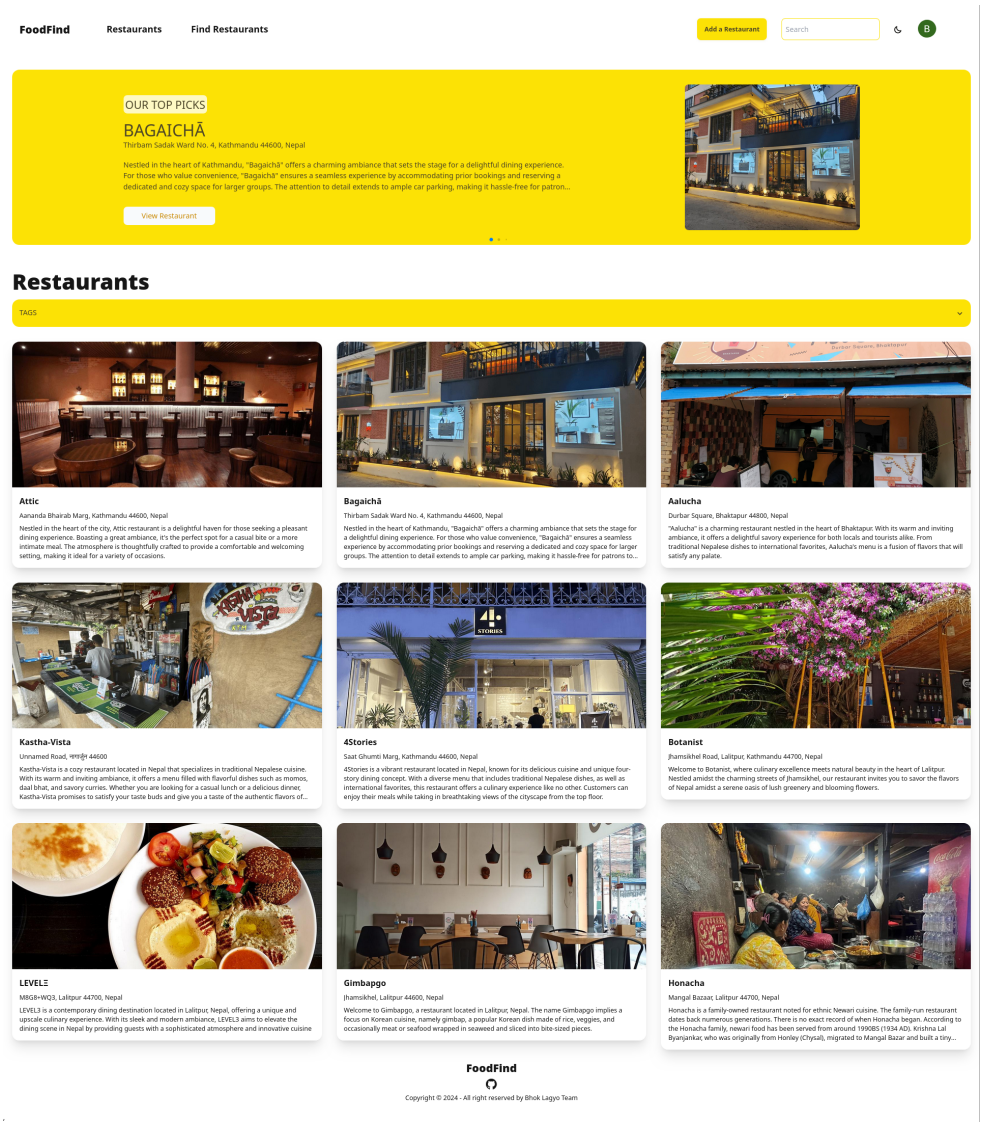


Figure 17: Restaurants List

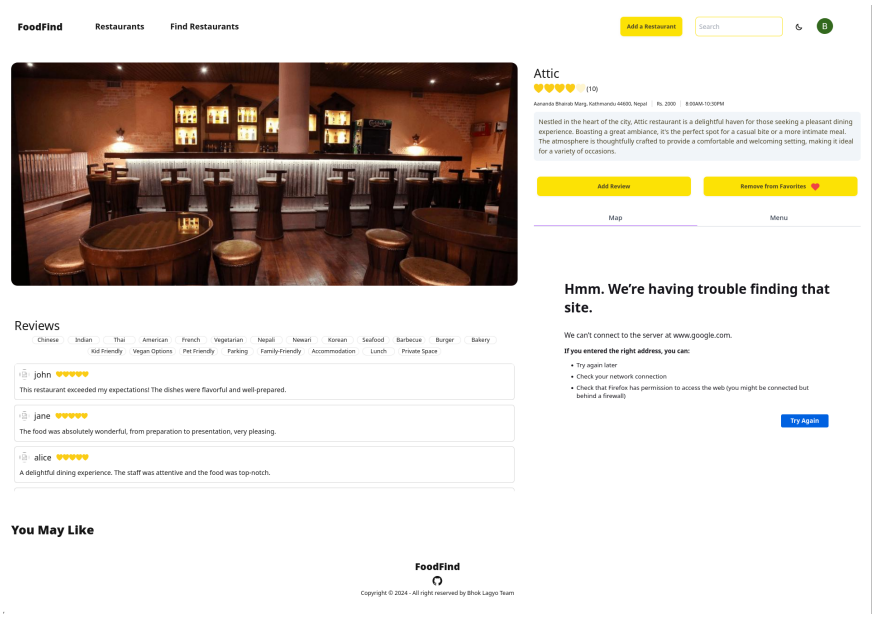


Figure 18: Restaurant Profile

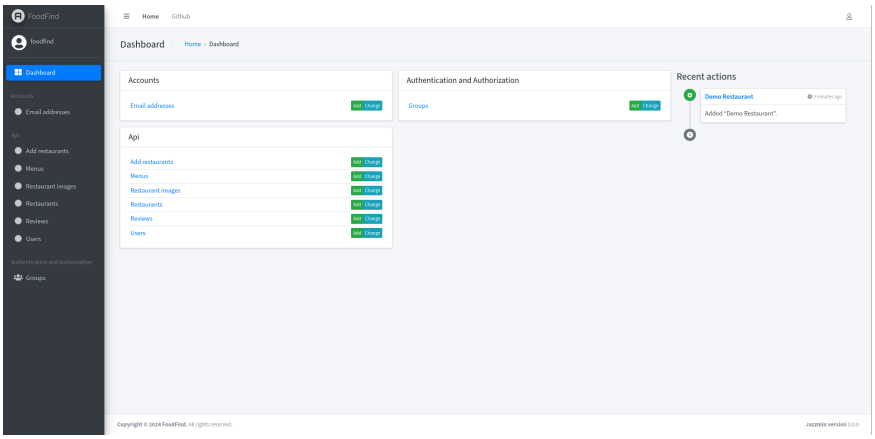


Figure 19: Admin Section