**Dine-Deal - Mobile Application**



**By:**

**Syed Kamran Shah**

**35453**

**Safiullah Shahid**

**35857**

**Muhammad Ismail**

**35897**

**Supervised by:**

**Dr. Musharaf Ahmed**

**Faculty of Computing**

**Riphah International** **University, Islamabad**

**Year 2024**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of**

**the Degree of**

**Bachelor of Science in Software Engineering**

# Faculty of Computing

# Riphah International University, Islamabad

Date: ??????

**Final Approval**

# Committee:

# 1

Dr. Musharaf Ahmad

(Supervisor) (Head of Department/chairman)

**Declaration**

We hereby declare that this document, titled “Dine-Deal,” neither as a whole nor in part has been copied from any other source. It is further declared that this project was carried out by the undersigned entirely based on our personal efforts under the proficient guidance of our supervisor Dr. Musharaf Ahmed. If any part of the system is proven to be copied or reproduced from any source, we shall face the consequences.

**Syed Kamran Qadeer Shah**

**35453**

**Safiullah Shahid**

**35857**

**Muhammad Ismail**

**35897**

**Dedication**

We dedicate this project to **Allah Almighty**, the source of wisdom, inspiration, and knowledge. We also dedicate this project to our families, friends, and teachers, whose support and encouragement have been fundamental to the success of this project. A special thanks goes to our supervisor, Dr. Musharaf Ahmed, for his consistent guidance and support.

**Acknowledgement**

We are immensely grateful to **Allah Almighty** for giving us the strength and perseverance to complete this project. Our deepest gratitude goes to **Dr. Musharaf Ahmed**, whose insights, patience, and support were essential to the success of this work. We would also like to extend our appreciation to all faculty members of Riphah International University for their constant encouragement.

**Abstract**

Dine-Deal is a mobile-first restaurant reservation system designed to optimize table occupancy during off-peak hours by offering dynamic discounts. It bridges the gap between users seeking discounted dining experiences and restaurants aiming to fill empty tables during non-peak periods.

The system incorporates **real-time Reservation management**, **dynamic discounting**, and **pre-payment options** to reduce no-show rates.

The project aims to tackle shortcomings in existing systems like OpenTable and FirstTable.

### Table of Contents

#### 1. ****Introduction****

* 1.1 Opportunity and Stakeholder
  + Opportunity
  + Stakeholders
    - Guests/Users
    - Registered Users
    - Restaurants
    - Admin
* 1.2 Problem Statement
  + Lack of Dynamic Discounting
  + No Pre-Payment Options
  + Limited Restaurant Control
* 1.3 Project Scope
  + Guests
  + Registered Users
  + Restaurants
  + Admin Dashboard
* 1.4 Proposed Solution
  + Real-Time Dynamic Discounts
  + Pre-Payment Options
  + Dashboard for Restaurants
* 1.5 Objectives

#### 2. ****Literature/Market Survey****

* 2.1 Introduction
* 2.2 Literature Review/Market Survey
  + Existing Systems
    - OpenTable
    - FirstTable
    - Key Gaps Identified
  + User Survey Insights
    - Age Group
    - Dining Frequency
    - Awareness of Table Reservation Apps
    - Use of Mobile Apps to Browse Restaurants
    - Important Features in a Restaurant App
    - Usefulness of Reading User Reviews
    - Likelihood of Recommending an App Offering Discounts
    - Willingness to Pay a Reservation Fee
* 2.3 Brainstorming
* 2.4 Summary

#### 3. ****Requirement Analysis****

* 3.1 Introduction
* 3.2 Problem Scenarios
  + Problem Statement 1: Optimizing Off-Peak Restaurant Hours
  + Problem Statement 2: High No-Show Rates
  + Problem Statement 3: Limited Restaurant Control
* 3.3 Functional Requirements
  + User Features
    - Browse Restaurants
    - Register
    - Explore Deals
    - Reservation Management
    - Table Availability
    - Bookmark Restaurants
    - Add to Favorites
    - Leave Reviews
  + Restaurant Owner Features
    - Update Information
    - Manage Reservations
    - Offer Discounts
    - Set Pre-Payment Options
  + Admin Features
    - Monitor and Manage Restaurants
    - Generate Reports
* 3.4 Non-Functional Requirements
  + Performance
  + Scalability
  + Usability
  + Reliability
  + Security
  + Maintainability
* 3.5 Summary

#### 4. ****System Design****

* 4.1 Introduction
* 4.2 Architectural Design
* 4.3 Detailed Design
  + 4.3.1 Use Case Diagram
  + 4.3.2 Descriptive Use Cases
    - * Register
      * Login
      * Forgot Password
      * Manage location
      * make reservation
      * make payment
      * cancel reservation
      * write review
      * Edit profile
      * update payment method
      * apply for a partner ship
      * manage reservations
      * add discount
      * Edit rest. Info
      * manage registration
      * manage restaurants
      * Manage users
* 4.4 Activity Diagram

#### Register

#### Login

#### Forgot Password

#### Manage location

#### make reservation

#### make payment

#### cancel reservation

#### write review

#### Edit profile

#### update payment method

#### apply for a partner ship

#### manage reservations

#### add discount

#### Edit rest. Info

#### manage registration

#### manage restaurants

#### Manage users

**Chapter 1:**

**Introduction**

**Chapter 1:**

### Introduction

Dine-Deal is a mobile application focused on improving restaurant table occupancy during off-peak hours by offering flexible and dynamic discounts to users. The idea is to help restaurants maximize their revenue during quieter times while providing users with a system to book tables at discounted rates. By leveraging technology, Dine-Deal provides a user-friendly interface for seamless reservation management.

This project addresses the limitations in existing systems, such as lack of real-time discount flexibility, high no-show rates.

### 1.1 Opportunity and Stakeholder

Dine-Deal capitalizes on the growing demand for online restaurant Reservations, particularly during off-peak hours when restaurants struggle to fill tables. By offering dynamic discounts and easy-to-use mobile interfaces, the app creates a win-win situation for both users and restaurants.

* **Opportunity**:
  1. The global restaurant reservation market is growing at a rate of **15% per year**.
  2. Online Reservation systems have seen a **20-25% increase** in usage, with higher engagement on mobile devices.
  3. Restaurants face challenges filling tables during off-peak hours but can increase Reservations by **15-30%** with discounts.
* **Stakeholders**:
  1. **Guests/Users**: Can browse restaurants, view deals, and check ratings but cannot make reservations until they create an account.
  2. **Registered Users**: Can set up and edit profiles, add or remove locations, explore restaurants and deals, reserve tables, make payments, receive confirmations, write reviews, update payment methods, reset passwords, and view reservation history.
  3. **Restaurants**: Can apply for partnership, manage deals and discounts, view and reply to user reviews, and manage reservations and reservation history.
  4. **Admin**: Can approve or reject partnership requests, remove users, and remove restaurant registrations

### 1.2 Problem Statement

The current restaurant reservation systems, such as OpenTable and FirstTable, have significant gaps in their offerings, including:

1. **Lack of Dynamic Discounting**: Discounts are either non-existent (OpenTable) or fixed (FirstTable), limiting restaurant flexibility.
2. **No Pre-Payment Options**: High no-show rates are common on existing systems due to the lack of pre-payment or deposit systems.
3. **Limited Restaurant Control**: Restaurants have limited ability to control availability or customize promotions on existing systems.

**Impact:**  
Restaurants miss out on potential revenue from unsold tables, and users are limited in their choices for discounted dining experiences. The absence of pre-payment options leads to lost revenue from no-shows.

### 1.3 Project Scope

**1.3.1**. **Guests**

**Role Overview**: Guests are users who can explore restaurants but need to create an account to make reservations.

**Key Responsibilities**:

1. **Browse Restaurants:** Search and view restaurant information such as deals, hours, and reviews.
2. **View Restaurant Details:** Browse restaurant profiles, including location and available deals.
3. **Edit Location:** Update or select locations to refine restaurant search results.
4. **Create an Account:** Register to make reservations.

**1.3.2. Registered Users**

**Role Overview**: Registered users are account holders who can reserve tables, manage profiles, and access additional features.

**Key Responsibilities**:

1. **Set Up and Edit Profile:** Create and update personal information like email number name.
2. **Add/Edit Locations:** Manage preferred locations for restaurant suggestions.
3. **Explore Restaurants and Deals**: Search for restaurant deals.
4. **Reserve Tables:** Reserve table and make payments.
5. **Manage Reservations:** View reservation history and receive reservation confirmations.
6. **Write Reviews:** Share dining experiences by reviewing restaurants.
7. **Update Payment Methods**: Manage payment preferences for faster transactions.
8. **Reset Password:** Secure account recovery through password reset functionality.

3. **Restaurants**

**Role Overview**: Restaurants are partners who manage their reservations, deals, and respond to customer feedback.

**Key Responsibilities**:

1. **Apply for Partnership:** Submit registration form for admin approval to join the system.
2. **Manage Deals and Discounts:** Create, update, and control restaurant-specific discounts.
3. **View Reservations:** view upcoming reservations.
4. **Reservation History:** view reservation history.
5. **Respond to User Reviews:** Engage with customers by replying to feedback and improving satisfaction.

4. **Admin Dashboard**

**Role Overview**: Admins oversee system operations, manage users and restaurants.

**Key Responsibilities**:

1. **Approve/Reject Restaurant Partnerships:** Evaluate and manage restaurant registration requests.
2. **Remove Users:** Manage the system's integrity by removing inactive or disruptive users.
3. **Remove Restaurants:** Facilitate quality by deactivating non-compliant or underperforming restaurants.
4. **Generate Reports:** Analyze system performance, reservations, and user activity for strategic decisions.

### 1.4 Proposed Solution

Dine-Deal offers a comprehensive solution to these challenges by:

1. **Providing Real-Time Dynamic Discounts**: Restaurants can create and adjust discounts based on real-time demand and table availability.
2. **Offering Pre-Payment Options**: To reduce no-show rates, Dine-Deal allow pre-payment for reservation.
3. **Restaurants Panel**: Restaurants are given powerful tools to manage Reservations, adjust availability, and customize discounts and pre-payments dynamically.

### 1.5 Objectives

The primary objectives of the Dine-Deal project are:

1. To provide users with a **convenient, mobile-friendly system** for Reservation discounted restaurant tables.
2. To offer **real-time discount flexibility** to restaurants, helping them maximize occupancy during off-peak hours.
3. To reduce the issue of **no-shows** through the implementation of **pre-payment Option**.
4. To deliver a **dashboard** for restaurants to manage Reservations, pre-payments, discounts, and availability seamlessly.

**Chapter 2:**

**Literature/Market Survey**

**Chapter 2: Literature/Market Survey**

### 2.1 Introduction

This chapter provides an overview of the restaurant reservation market, focusing on existing systems, their features, and gaps in their offerings. It also examines how **Dine-Deal** addresses these gaps. A market survey was conducted to gather insights about user behavior and preferences, which informed the development of Dine-Deal’s key differentiating features.

### 2.2 Literature Review/Market Survey

The market for restaurant reservations has several well-established systems, such as **OpenTable** and **FirstTable**, which offer a variety of features but also suffer from certain limitations. By understanding these systems and their gaps, we were able to identify opportunities for innovation with Dine-Deal.

**Existing Systems**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Open Table** | **First Table** | **Dine-Deal** |
| Table Reservations | ✔ | ✔ | ✔ |
| Dynamic Discounts(Real-Time) | ❌ | ❌ | ✔ |
| Fixed Discounts (Early Dining) | ❌ | ✔ | ✔ |
| Pre-Payment/Deposit Option | ❌ | ❌ | ✔ |
| No-Show Prevention (Pre-Payment) | ❌ | ❌ | ✔ |
| Restaurant Control Over Discounts | ❌ | ❌ | ✔ |
| User Reviews & Ratings | ✔ | ❌ | ✔ |
| Admin Dashboard for Restaurants | ✔ | ❌ | ✔ |
| Reservation History for Users | ✔ | ❌ | ✔ |
| Favorite Restaurants | ✔ | ✔ | ✔ |

**2.2.1. OpenTable**

**Limitations:**

* No dynamic discounting.
* High no-show rates.
* No pre-payment options to secure reservations.

**2.2.2. FirstTable**

**Limitations:**

* Fixed discount model (usually 50%), limiting flexibility.
* Limited restaurant control over table management.
* No tools for restaurants to manage real-time availability.

**Key Gaps Identified:**

* **Lack of Dynamic Discounting**: Both systems lack dynamic discount options, which limits a restaurant's ability to adjust pricing based on demand.
* **No Pre-Payment Options**: High no-show rates are a recurring issue as neither system offers pre-payment or deposit features to reduce cancellations.
* **Limited Restaurant Control**: Restaurants have restricted control over real-time Reservation adjustments and customizing promotions, resulting in lower occupancy during off-peak hours.

To better understand the target users and validate the need for the features Dine-Deal offers, a survey was conducted with **98 participants**. The following insights were gathered:

**1. Age Group**

Majority are **18-25 years old (88.7%), followed by 28-35 (9.2%).** Targeting younger demographics is essential for app adoption.

**2. Dining Frequency**  
**58.2%** dine out **2-3 times a week**. Regular diners prioritize convenience, making them the core user base.

**3. Awareness of Table Reservation Apps**  
**84.7%** were **unaware** of any table Reservation apps, indicating a market gap and growth potential for Dine-Deal.

**4. Use of Mobile Apps to Browse Restaurants**  
**44.9%** use apps **occasionally**, while **36.7%** use them **rarely**. The potential exists to increase usage with more engaging features.

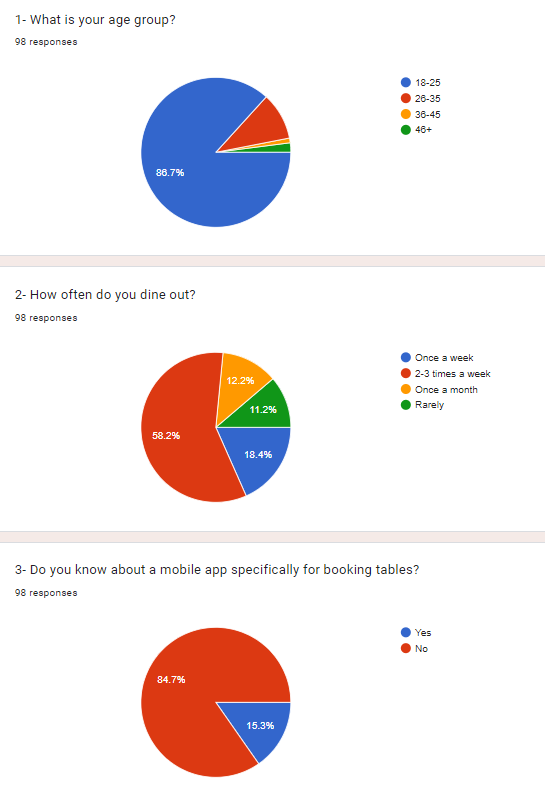
**5. Important Features in a Restaurant App**  
**Menu variety** (**58.2%**), **Pricing and deals** (**58.2%**), and **Table availability** (**50%**) are top user priorities. Focus on offering these features.

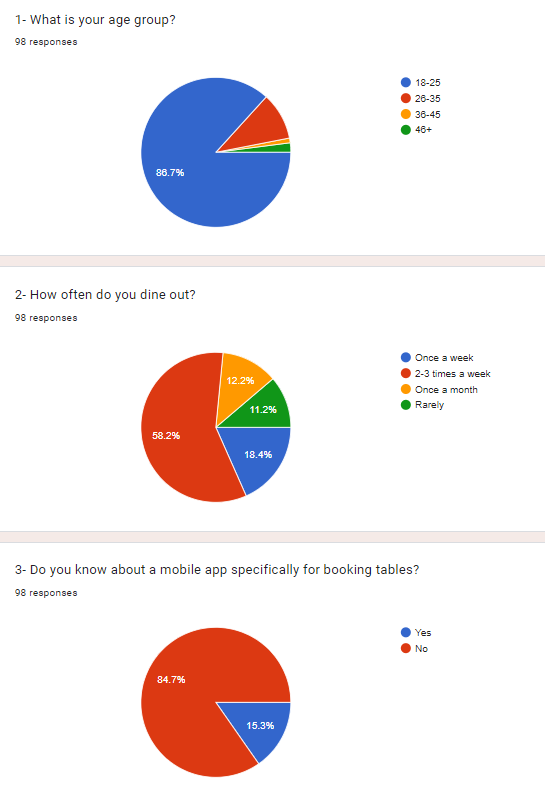
**6. Usefulness of Reading User Reviews**  
**71.4%** find reviews **very helpful**, suggesting that user feedback should be emphasized on the system.

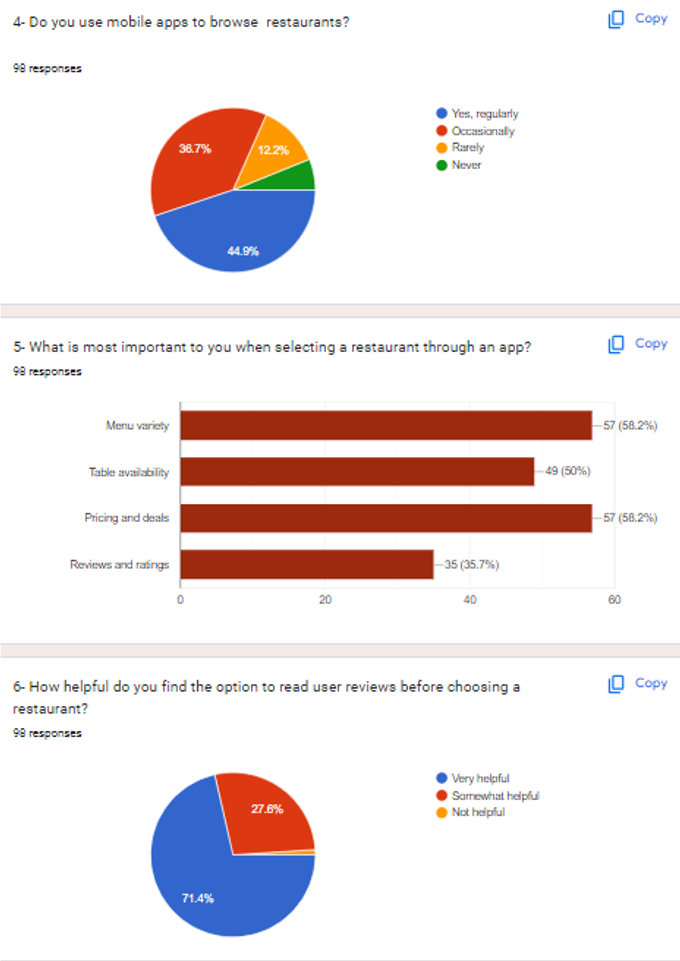
**7. Likelihood of Recommending an App Offering Discounts**  
**60.2%** would **very likely** recommend a restaurant app offering discounts, supporting the need for real-time, dynamic discounting.

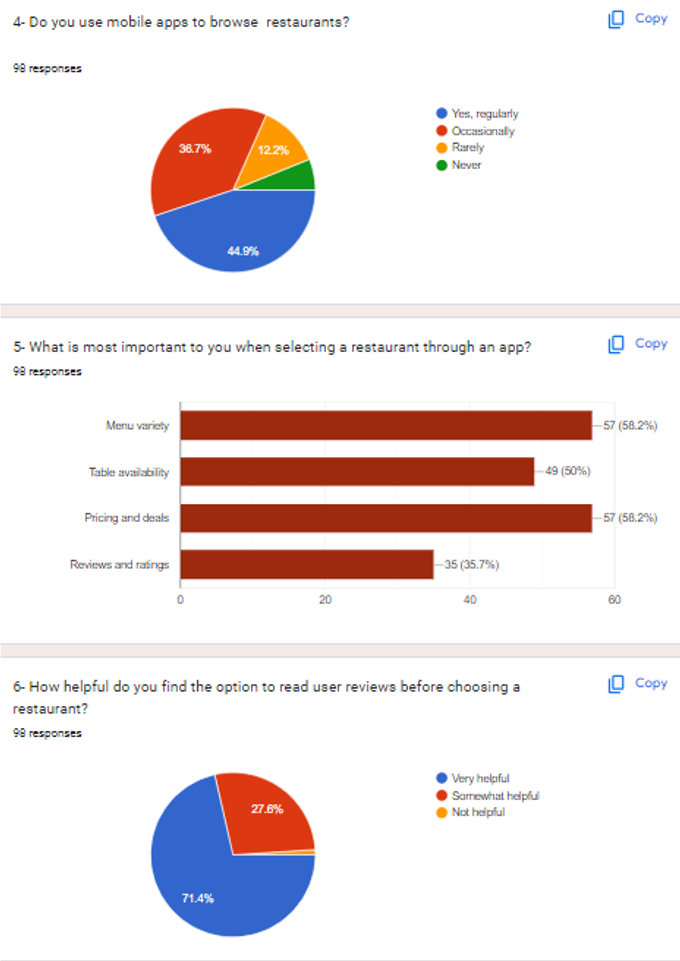
**8. Willingness to Pay a Reservation Fee**  
**65.3%** are willing to pay a small fee to guarantee table reservations, reinforcing the value of a pre-payment feature to reduce no-shows.

### 2.3 Unstructured interview

Several Unstructured interview were conducted to identify critical features that could differentiate **Dine-Deal** from existing systems. A major insight from these sessions was the need for dynamic, real-time discounts that adjust based on demand. By addressing the limitations of existing systems and focusing on customer pain points, Dine-Deal can provide a more versatile and efficient solution for both users and restaurants. 









### 2.4 Summary

The literature and market survey highlight significant gaps in existing restaurant reservation systems that Dine-Deal aims to address. Through features such as real-time discount flexibility, pre-payment options, and better control for restaurants over table management and availability. By targeting the identified preferences and behaviors of younger, frequent diners and offering the features they value most, Dine-Deal has the potential to capture a significant share of the restaurant reservation market.

**Chapter 3:**

**Requirement Analysis**

**Chapter 3: Requirement Analysis**

### 3.1 Introduction

In this chapter, we will discuss the Functional & Non-Functional requirements of our project “Dine-Deal”. Prior to that, we will discuss all the problem statements identified. The functional requirements were gathered through interviews and brainstorming. The non-functional requirements are derived based on system performance, security, and reliability needs.

### 3.2 Problem Scenarios

|  |  |
| --- | --- |
| **Problem Statement** | **Details** |
| **The problem of restaurants** | The problem of optimizing off-peak restaurant hours. |
| **Affects** | Restaurants |
| **The Result of which** | Missed revenue opportunities and underutilization of capacity during off-peak hours. |
| **Benefits of a Solution** | Real-time dynamic discounting based on demand allows restaurants to attract more customers during slow periods. |

|  |  |
| --- | --- |
| **Problem Statement** | **Details** |
| **The problem of restaurants** | High no-show rates due to lack of pre-payment options. |
| **Affects** | Restaurants |
| **The Result of which** | Loss of revenue and operational inefficiencies. |
| **Benefits of a Solution** | Introducing pre-payment or deposit systems can significantly reduce no-show rates, ensuring revenue even if cancellations occur. |

|  |  |
| --- | --- |
| **Problem Statement** | **Details** |
| **The problem of restaurants** | Limited control for restaurants over real-time Reservation adjustments and promotions. |
| **Affects** | Restaurants |
| **The Result of which** | Restaurants are unable to efficiently manage table availability or offer custom promotions in real-time, impacting customer satisfaction and revenue. |
| **Benefits of a Solution** | Providing a user-friendly admin dashboard gives restaurants full control over reservations, dynamic discounts, and promotions. |

* 1. **Functional Requirements**

**3.3.1 User (Guest, Registered user)**

* The user shall create an account or log in to user portal.
* The user shall create an account or log in to the user portal.
* The user shall reset their password securely.
* The user shall set up and edit their profile.
* The user shall browse restaurants, deals, and restaurant ratings.
* The user shall view detailed restaurant profiles (e.g., location, services).
* The user shall reserve tables
* The user shall make payments.
* The user can update payment methods.
* The user can give ratings and submit reviews for restaurants.
* The user can add or remove locations.
* The user can view reservation history.

**3.3.2 Restaurants**

* The restaurant shall apply for partnership via registration.
* The restaurant shall log in to account.
* The restaurant can manage their restaurant profile and deals.
* The restaurant shall manage deals and discounts offered to users.
* The restaurant shall view and reply to user reviews.
* The restaurant shall access and manage reservations.
* The restaurant shall view reservation history for operational insights.

**3.3.3 Restaurants**

* The admin shall log in to access the admin dashboard.
* The admin shall approve or reject restaurant partnership applications.
* The admin can remove user accounts when necessary.
* The admin shall remove restaurant registrations on the basis of reviews.

### 3.4 Non-Functional Requirements

### ****1. Performance****

* The system must respond quickly to user actions to ensure a seamless user experience.
* Deal and restaurant information should load within 3 seconds, and reservation processing must complete in 2 seconds.

### 3.5 Modules

### ****1. User Management Module****

### ****2. Restaurant Management Module****

### ****3. Deal Management Module****

### ****4. Search and Discovery Module****

### ****5. Reservation and Booking Module****

### ****6. Payment and Billing Module****

### ****7. Review and Feedback Module****

### ****8. Analytics and Reporting Module****

### ****9. Notification Module****

### ****10. Admin Management Module****

### ****11. Security and Compliance Module****

### ****12. Mobile App and Web Integration Module****

### 3.5 Summary

In this chapter, we outlined the functional and non-functional requirements and modules that are essential for the successful implementation of the Dine-Deal system. These requirements address the main

challenges identified, including off-peak table optimization, reduced no-show rates, and enhanced control for restaurants through an admin dashboard.

**Chapter 4:**

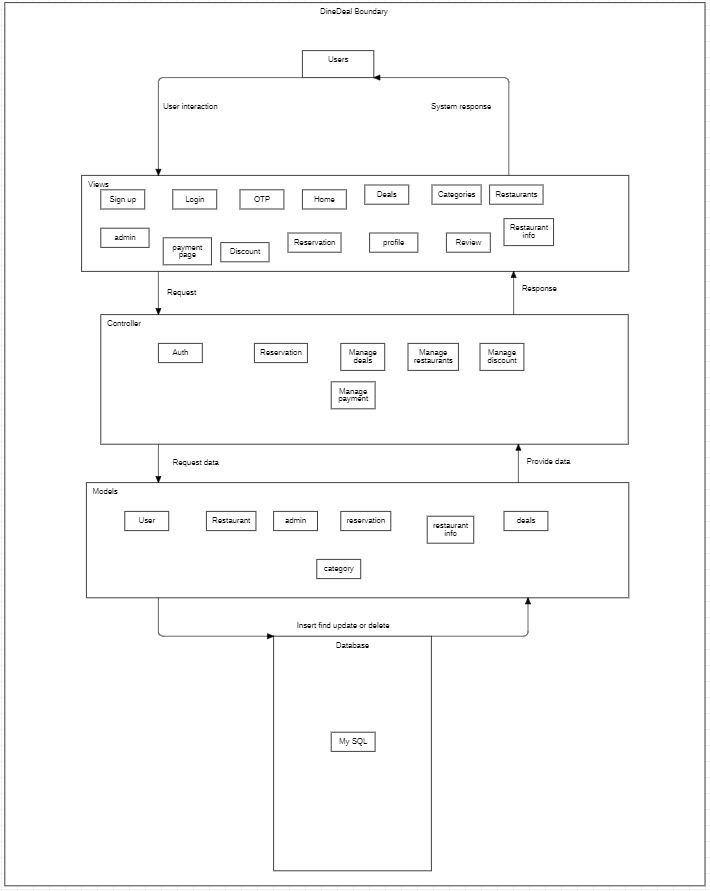
**System Design**

**Chapter 4: System Design**

**4.1 Introduction**

This chapter describes the design aspects of the Dine-Deal application, detailing the architectural framework, use cases that facilitate the implementation of the restaurant reservation system. The design is structured to facilitate scalability, usability, and efficient management of reservations, user accounts, and restaurant profiles.

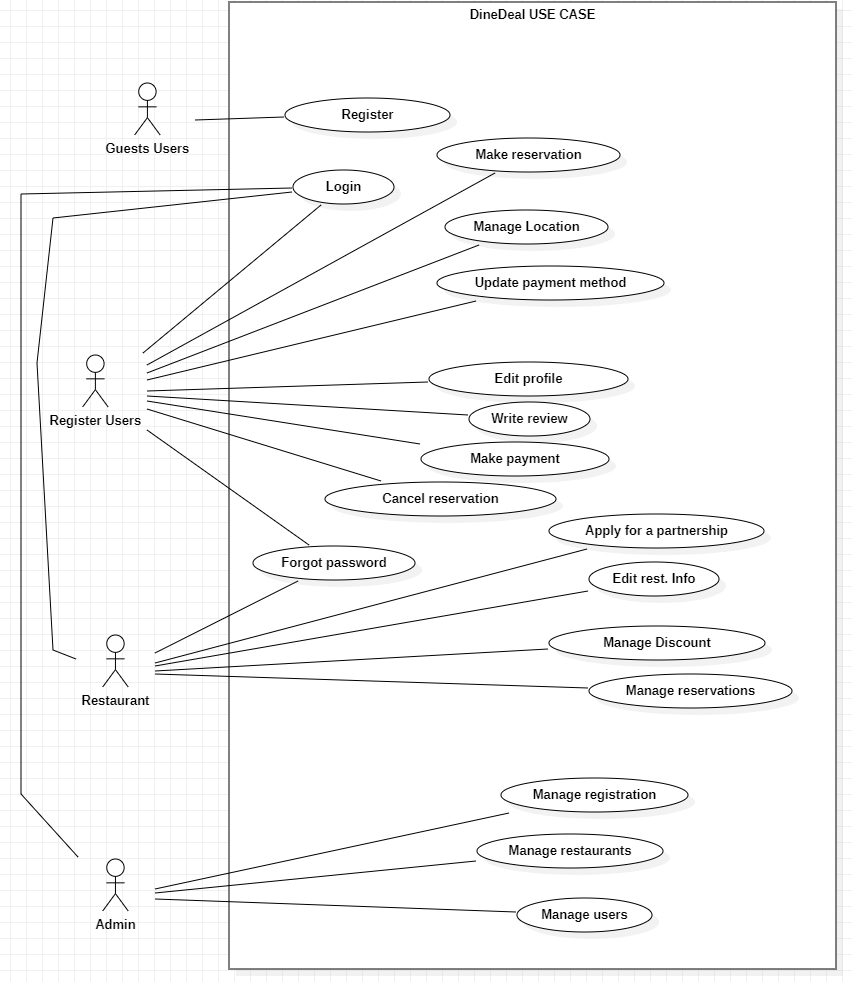
### 4.2 Architectural Design (MVC)

****

#### 4.3 Detailed Design

This section provides detailed designs for various components, including use case diagrams and descriptive use cases.

**Use Case Diagram:**

****

### 

### 4.3.1 Use Case

4.3.1.1 Register

4.3.1.2 Login

4.3.1.3 Forgot Password

4.3.1.4 Manage location

4.3.1.5 Make reservation

4.3.1.6 Make payment

4.3.1.7 Cancel reservation

4.3.1.8 Write review

4.3.1.9 Edit profile

4.3.1.10 Update payment method

4.3.1.11 Apply for a partnership

4.3.1.12 Manage reservations

4.3.1.3 Add discount

4.3.1.14 Edit rest. Info

4.3.1.15 Manage registration

4.3.1.16 Manage restaurants

4.3.1.17 Manage users

### 4.3.2 Descriptive Use Cases

**UC-ID: 4.3.1.1 Register**

|  |  |  |
| --- | --- | --- |
| **Use case: Register** | | |
| Brief Description**:** Allow user to register his/herself through email. | | |
| Actors: Guest users | | |
| Preconditions:   * 1. User has a unique email address and not associated with any existing account. | | |
| Main flow: | | |
| Actor:  1. User starts Registration process.  3. User enters information (name, email and password).  6. User enters verification code and lick on “verify” | | System:  2. System asks the user for email.  4. System validates and send a verification code to email.  5. System send verification code message on email.  7. System validates verification code and display “Registration Successful” |
| Post condition:   1. Account is created and updated in database. | | |
| Alternative flow: | | |
| Actor:  1. User enters an invalid email (i.e. dinedeal@@gmail.com).  3. User enters already registered email.  5. User enters a small password. | System:  2. System display “Invalid email”  4. System display “Email already exist”  6. System display “Password must be atleast 8 character long”. | |

**UC-ID: 4.3.1.2 Login**

|  |  |
| --- | --- |
| **Use case: Log in** | |
| Brief Description**:** Allow user to log in to the system through valid email and password. | |
| Actors: User, Restaurants , admin | |
| Preconditions:   1. User has already registered an account. 2. User have access to valid login credentials ( i.e. email, password) | |
| Main Flow: | |
| Actor:  1. User starts login process.  3. User add valid email and password. | System:  2. System asks for valid email and password.  4. The system displays “login successfully”. |
| Post condition:   1. Information of the user is stored. 2. User is redirected to their dashboard or homepage. 3. User can access system features based on their role and permissions. | |
| Alternative flow: | |
| Actor:  1. The user enters unregistered email (i.e. Dinedeal"@gmail.com).  2. User enters invalid email (i.e. dinedeal@@gmail.com, dinedeal@gmailcom).  3. User enters wrong password. | System:  1. The system displays ”Use valid email”  2. The system displays “Invalid email”.  The system displays “Invalid Password”. |

**UC-ID: 4.3.1.3 Forgot Password**

|  |  |
| --- | --- |
| **Use case: Forgot Password** | |
| Brief Description**:** Allow user to forgot password through valid email. | |
| Actors: User, Restaurants | |
| Preconditions:  User has already registered an account.  User is using valid email address. | |
| Main Flow: | |
| Actor:  1. User starts forgot Password process.  3. User enter a valid email.  5. User enters a verification code  7. User enters new password | System:  2. System asks for valid email.  4. The system send OTP on email.  6. The system validates and allow user to make a new password.  8. The system updates new password. |
| Post condition:   1. System Update the user password. | |
| Alternative flow: | |
| Actor:  1. The user enters unregistered email (i.e. Dinedeal1@gmail.com).  2. User enters invalid email (i.e. dinedeal@@gmail.com, dinedeal@gmailcom).  3. User enters wrong OTP  User enters a small password. | System:  1. The system displays ”Use valid email”  2. The system displays “Invalid email”.  The system displays “Invalid verification code”.  System display “Password Length must be more than 6 words or numbers”. |

**UC-ID: 4.3.1.4 Manage Location**

|  |  |
| --- | --- |
| **Use case: Manage Location** | |
| Brief Description**:** Allow user to change location their current location for personalized servics. | |
| Actors: User | |
| Preconditions:  1. User is logged into the system.  2. Location management is available. | |
| Main Flow: | |
| Actor:  1. User accesses location settings.  3. User updates their location. | System:  2. System displays current location.  4. System validates and saves the new location. |
| Post condition:   1. User's location is updated. 2. System provides recommendations based on the updated location. | |
| Alternative flow: | |
| Actor:  1. User enters an invalid location.  2. User cancels the update. | System:  1. System displays "Invalid location."  2. System retains the previous location. |

**UC-ID: 4.3.1.5 Make Reservation**

|  |  |
| --- | --- |
| **Use case: Make Reservation** | |
| Brief Description**:** Allow users to reserve a table at a restaurant. | |
| Actors: User | |
| Preconditions:   1. User is logged in. 2. Restaurant has available reservations. | |
| Main Flow: | |
| Actor:  1. User selects a restaurant.  3. User chooses date, time, and number of | System:  2. System displays reservation options.  4. System confirms availability and reserves the table. |
| Post condition:   1. Reservation is successfully made. 2. User receives a confirmation message. | |
| Alternative flow: | |
| Actor:  1. User selects an unavailable time slot.  2. User cancels the reservation process. | System:  1. System displays "Time slot unavailable."  2. System does not reserve a table. |

**UC-ID: 4.3.1.6 Make Payment**

|  |  |
| --- | --- |
| **Use case: Make Payment** | |
| Brief Description**:** Allow users to make payments for their reservations or deals. | |
| Actors: User | |
| Preconditions:   1. User has a valid reservation or deal to pay for. 2. Payment system is functional. | |
| Main Flow: | |
| Actor:  1. User selects a reservation or deal to pay for.  3. User enters payment details. | System:  2. System displays payment options.  4. System processes the payment and confirms it. |
| Post condition:   1. Payment is successfully processed. 2. User receives a confirmation receipt. | |
| Alternative flow: | |
| Actor:  1. User enters invalid payment details.  2. Payment system is down. | System:  1. System displays "Payment failed."  2. System displays "Service unavailable." |

**UC-ID: 4.3.1.7 Cancel Reservation**

|  |  |
| --- | --- |
| **Use case: Cancel reservation** | |
| Brief Description**:** Allow user to cancel an existing table reservation. | |
| Actors: User (Registered user) | |
| Preconditions:  The user must be logged into the system. The user must has an upcoming reservation. | |
| Main Flow: | |
| Actor:  The user navigates to the “reservations” section.  The user select the reservation they wish to cancel.  The user cancel the reservation | System:  The system display the details of all upcoming reservations.  The system process the cancellation and update status reservation cancelled.  The system sends a cancellation confirmation email to the user. |
| Post condition:   1. The reservation status is updated to “cancelled” in the system. | |
| Alternative flow: | |
| Actor:  The user confirms the cancellation request. | System:  The system encounters a technical technical issue while processing the cancellation.  The system displays an error message and suggest user to try later. |

**UC-ID: 4.3.1.8 Write Review**

|  |  |
| --- | --- |
| **Use case: Write Review** | |
| Brief Description**:** Allows the user to write a review for a restaurant based on their experience. | |
| Actors: User (Registered user) | |
| Preconditions:  The user must be logged into the system. The user has a reservation history with the restaurant. | |
| Main Flow: | |
| Actor:  1. The user navigates to the “restaurant review” section.  The user select the “write review” option  The User fills out the review form and submits it. | System:  2. The system displays the list of reviews and allow user to give rating and write review.  The system displays a rating option and an option to write.  The system saves the review. |
| Post condition:   1. The review is successfully added to the restaurant’s reviews list and visible to other users. | |
| Alternative flow: | |
| Actor:  1 The user submits the review without selecting rating. | System:  The system displays “Give rating also”. |

**UC-ID: 4.3.1.9 Edit Profile**

|  |  |
| --- | --- |
| **Use case: Edit Profile** | |
| Brief Description**:** Allow users to edit their personal information. | |
| Actors: Restaurant | |
| Preconditions:   1. User is logged in. | |
| Main Flow: | |
| Actor:  1. User navigates to the profile section.  3. User updates personal details. | System:  2. System displays editable user information.  4. System validates and saves changes. |
| Post condition:   1. User information is successfully updated. | |
| Alternative flow: | |
| Actor:  1. User enters invalid details.  2. User exits without saving. | System:  1. System displays "Invalid input."  2. System does not update the information. |

**UC-ID: 4.3.1.10 Update payment Method**

|  |  |
| --- | --- |
| **Use case: Update payment Method** | |
| Brief Description**:** Allow users to choose the payment method. | |
| Actors: User | |
| Preconditions:   1. User has already registered an account. 2. User is logged in. | |
| Main Flow: | |
| Actor:  1. User go to payment method page.  3. User change the payment method. | System:  2. System navigate to that page.  4. System changes the payment method. |
| Post condition:   1. The payment method is changed. 2. User can make payment. | |
| Alternative flow: | |
| Actor:  1. User enters invalid payment details.  2. User exits without saving. | System:  1. System displays "Invalid payment details."  2. System does not update payment methods. |

**UC-ID: 4.3.1.11 Apply for Partnership**

|  |  |
| --- | --- |
| **Use case: Apply for Partnership** | |
| Brief Description**:** Allow restaurants to apply for partnership by submitting required details. | |
| Actors: Restaurant | |
| Preconditions:   1. Restaurant owner has access to the partnership form. | |
| Main Flow: | |
| Actor:  1. Restaurant owner navigates to the partnership form.  3. Owner fills in required details and submits the form. | System:  2. System displays the form.  4. System validates and saves the application. |
| Post condition:   1. Partnership request is successfully submitted. 2. Admin is notified of the new request. | |
| Alternative flow: | |
| Actor:  1. Owner enters incomplete details.  2. Owner exits without submitting. | System:  1. System prompts for missing information.  2. System does not save the application. |

**UC-ID: 4.3.1.12 Manage Reservations**

|  |  |
| --- | --- |
| **Use case: Manage Reservations** | |
| Brief Description**:** Allow restaurants to manage customer reservations. | |
| Actors: Restaurant | |
| Preconditions:   1. Restaurant has active reservations. 2. Restaurant owner is logged in. | |
| Main Flow: | |
| Actor:  1. Owner navigates to the reservations section.  3. Owner reviews or updates a reservation. | System:  2. System displays a list of reservations.  4. System saves the changes. |
| Post condition:   1. Reservation details are updated successfully. | |
| Alternative flow: | |
| Actor:  1. Owner tries to edit an expired reservation.  2. Owner exits without saving changes. | System:  1. System displays "Reservation expired."  2. System does not update the reservation. |

**UC-ID: 4.3.1.13 Manage Discounts**

|  |  |
| --- | --- |
| **Use case: Manage Discounts** | |
| Brief Description**:** Restaurants can add or delete the discounts on deals. | |
| Actors: Restaurant | |
| Preconditions:   1. Restaurant is registered 2. Restaurant is logged in | |
| Main Flow: | |
| Actor:  1. Owner navigates to the deals section.  3. Owner selects a deal and adds a discount. | System:  2. System displays active deals.  4. System validates and applies the discount. |
| Post condition:   1. Discount is successfully added. 2. Updated deal is displayed to users. | |
| Alternative flow: | |
| Actor:  1. 1. Owner enters an invalid discount.  2. Owner exits without saving changes. | System:  1. System displays "Invalid discount."  2. System does not update the deal. |

**UC-ID: 4.3.1.14 Edit restaurant information**

|  |  |
| --- | --- |
| **Use case: Edit restaurant information** | |
| Brief Description**:** Restaurants can edit their information. | |
| Actors: Restaurant | |
| Preconditions:   1. Restaurant is registered 2. Restaurant is logged in | |
| Main Flow: | |
| Actor:  1. Owner navigates to the profile settings.  3. Owner updates the information. | System:  2. System displays editable restaurant details.  4. System validates and saves changes. |
| Post condition:   1. Restaurant information is successfully updated. | |
| Alternative flow: | |
| Actor:  1. Owner enters invalid information.  2. Owner exits without saving. | System:  1. System displays "Invalid input."  2. System does not update the information. |

**UC-ID: 4.3.1.15 Manage Registration Requests**

|  |  |
| --- | --- |
| **Use case: Manage Registration Requests** | |
| Brief Description**:** Allow admin to manage incoming restaurant registration requests. | |
| Actors: Admin | |
| Preconditions:   1. Admin is logged in. 2. There are pending registration requests. | |
| Main Flow: | |
| Actor:  1. Admin navigates to the registration requests section.  3. Admin reviews a request.  5. Admin approves or rejects the request. | System:  2. System displays a list of requests.  4. System displays request details.  6. System updates the request status. |
| Post condition:   1. Request status is updated. 2. Approved restaurants are added to the system. | |
| Alternative flow: | |
| Actor:  1. Admin exits without taking action.  2. System error occurs. | System:  1. System retains the request as pending.  2. Admin is notified to retry. |

**UC-ID: 4.3.1.16 Manage Restaurants**

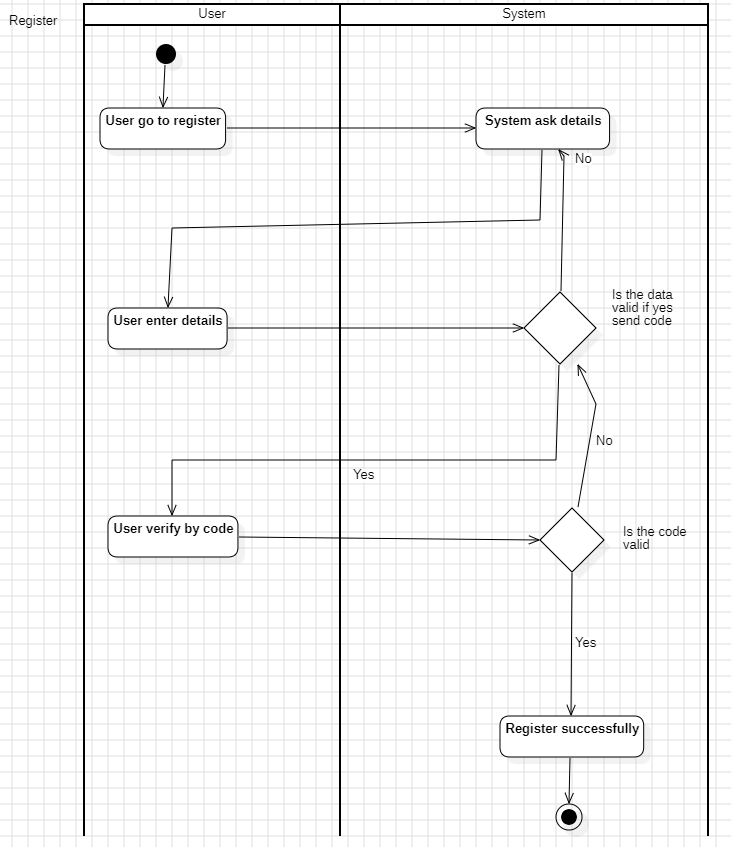
|  |  |
| --- | --- |
| **Use case: Manage Restaurants** | |
| Brief Description**:** Allow admin to manage registered restaurants. | |
| Actors: Admin | |
| Preconditions:   1. Admin is logged in. 2. Restaurants are registered in the system. | |
| Main Flow: | |
| Actor:  1. Admin navigates to the registered restaurants section.  3. Admin views or updates a restaurant's details.  5. Admin removes a restaurant. | System:  2. System displays a list of restaurants.  4. System saves the changes.  6. System confirms and removes the restaurant. |
| Post condition:   1. The restaurant is managed or removed. | |
| Alternative flow: | |
| Actor:  1. Admin cancels removal action.  2. System error occurs. | System:  1. System does not remove the restaurant.  2. Admin is notified to retry. |

**UC-ID: 4.3.1.17 Manage Users**

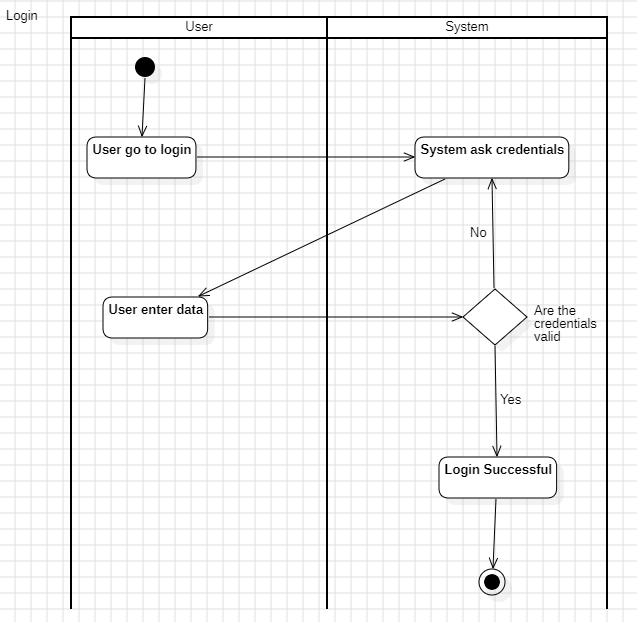
|  |  |
| --- | --- |
| **Use case: Manage Users** | |
| Brief Description**:** Allow admin to manage registered users. | |
| Actors: Admin | |
| Preconditions:   1. Admin is logged in. 2. Users are registered in the system. | |
| Main Flow: | |
| Actor:  1. Admin navigates to the users section.  3. Admin reviews user details.  5. Admin blocks a user. | System:  2. System displays a list of users.  4. System displays user information.  6. System updates the user's status. |
| Post condition:   1. User status is updated (e.g., blocked). | |
| Alternative flow: | |
| Actor:  1. Admin cancels the blocking action.  2. System error occurs. | System:  1. System does not block the user.  2. Admin is notified to retry |

#### 4.4 Activity Diagram

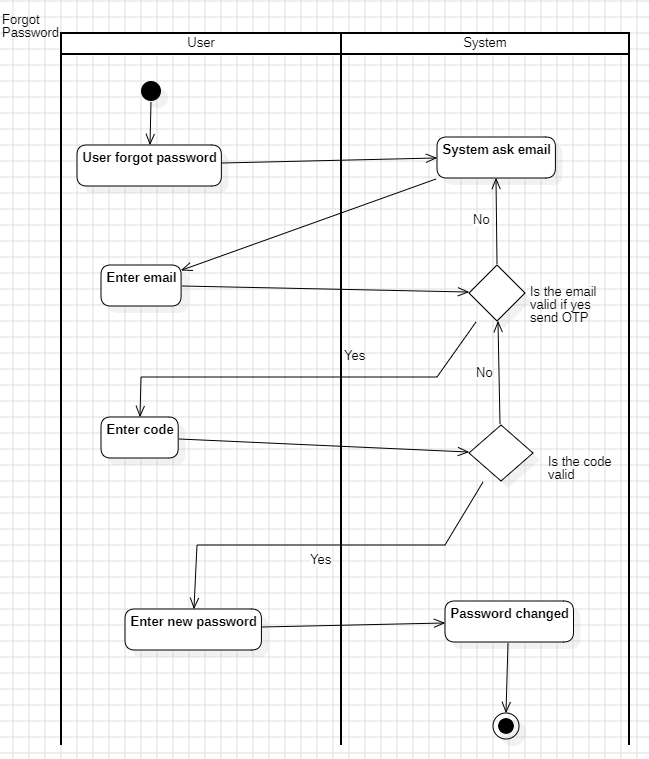
1. **Register**

****

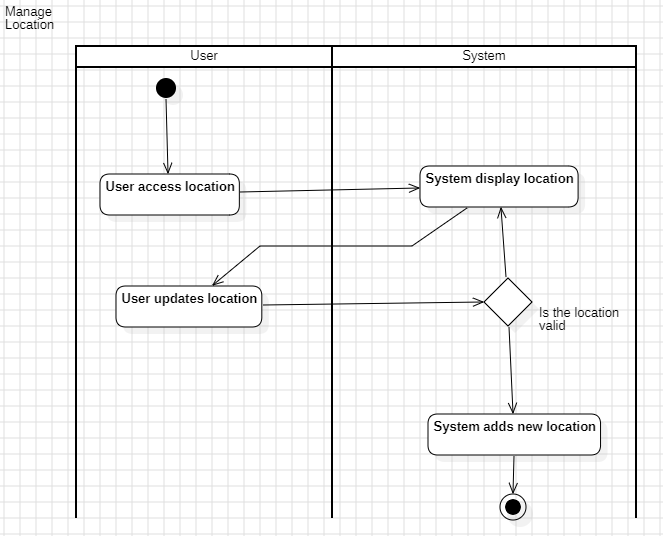
1. **Login**

****

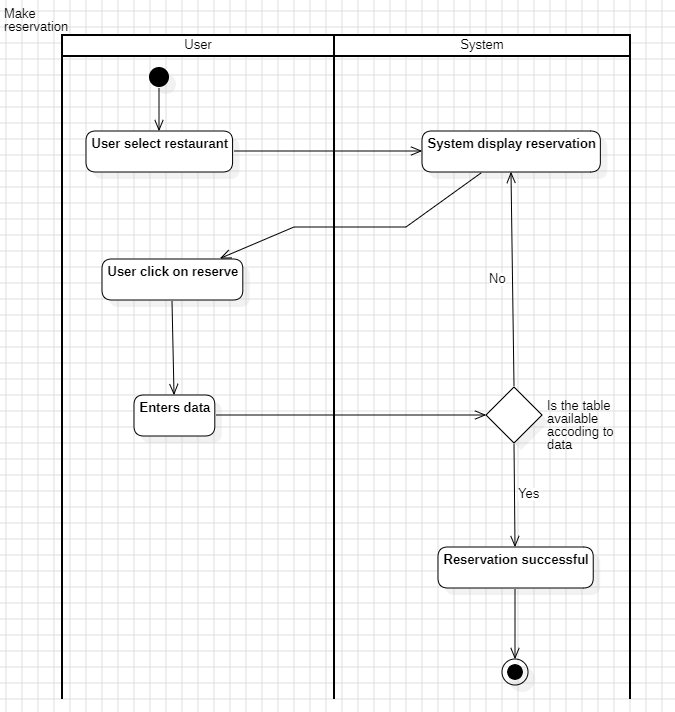
1. **Forgot password**

****

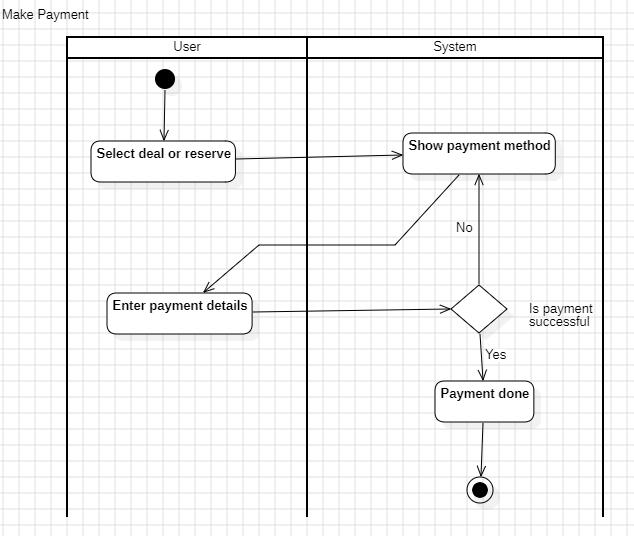
1. **Manage Location**

****

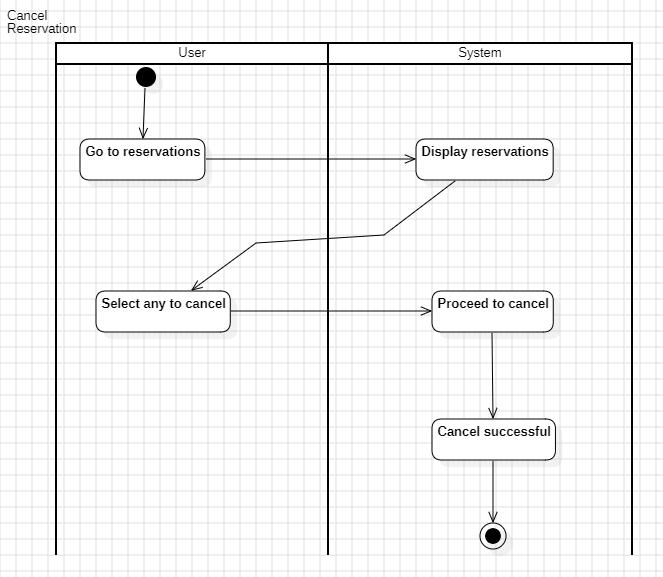
1. **Make reservation**

****

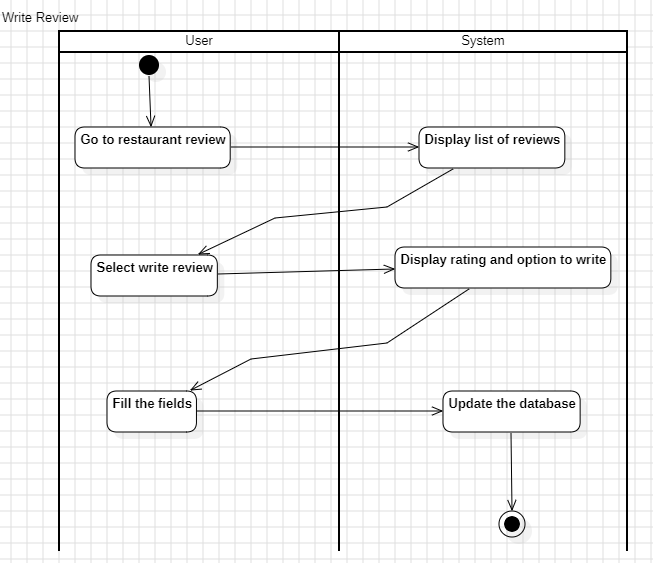
1. **Make payment**

****

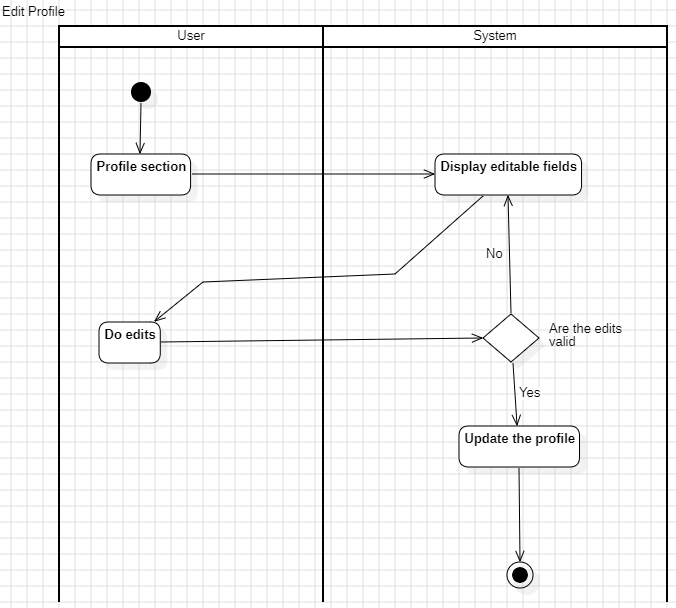
1. **Cancel reservation**

****

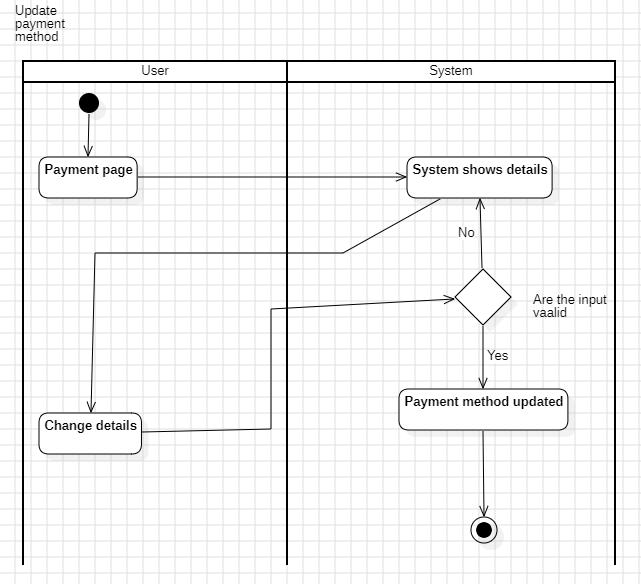
1. **Write review**

****

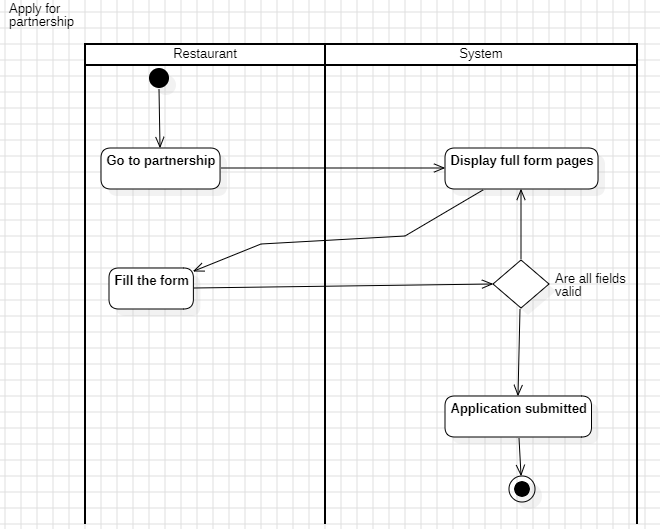
1. **Edit Profile**

****

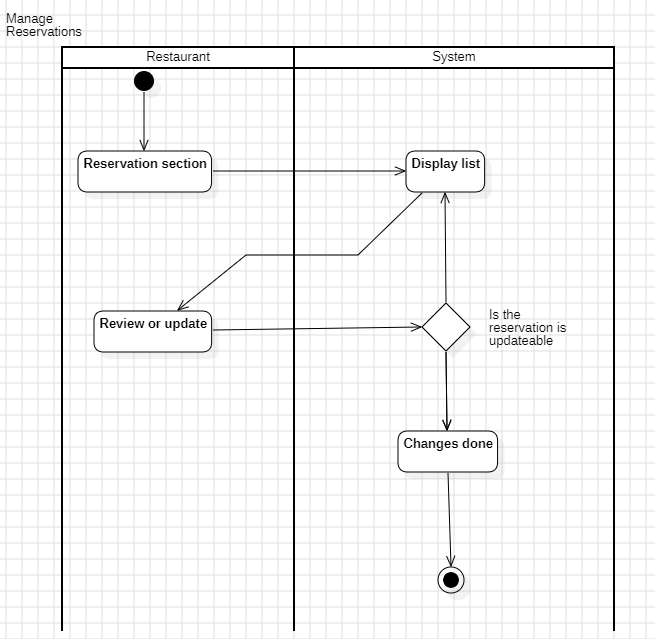
1. **Update payment method**

****

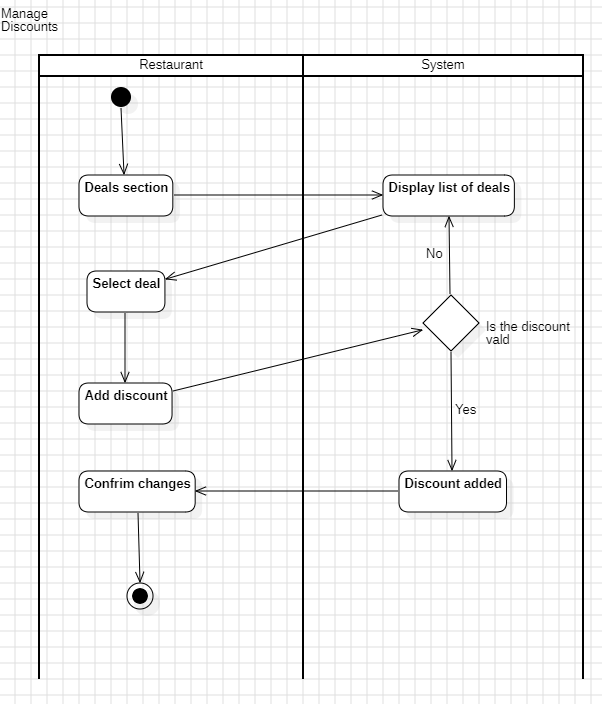
1. **Apply for partnership**

****

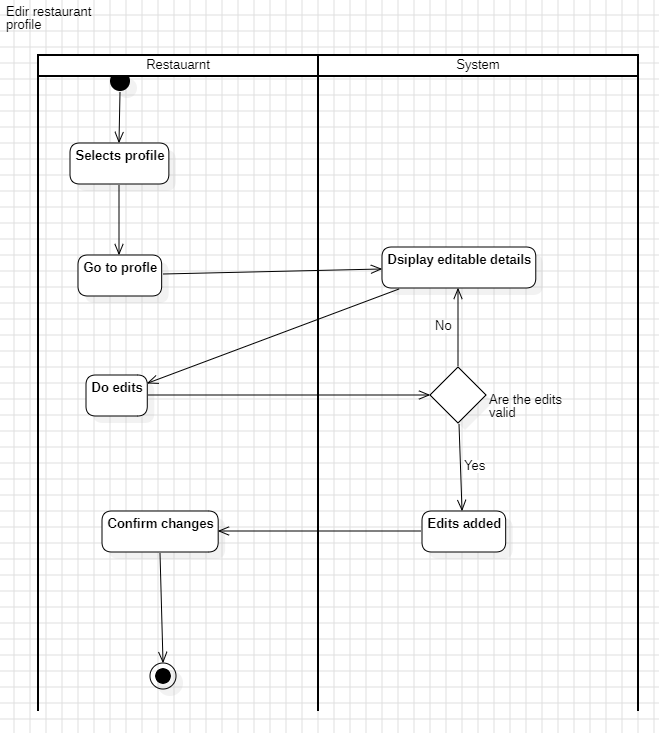
1. **Manage reservations**

****

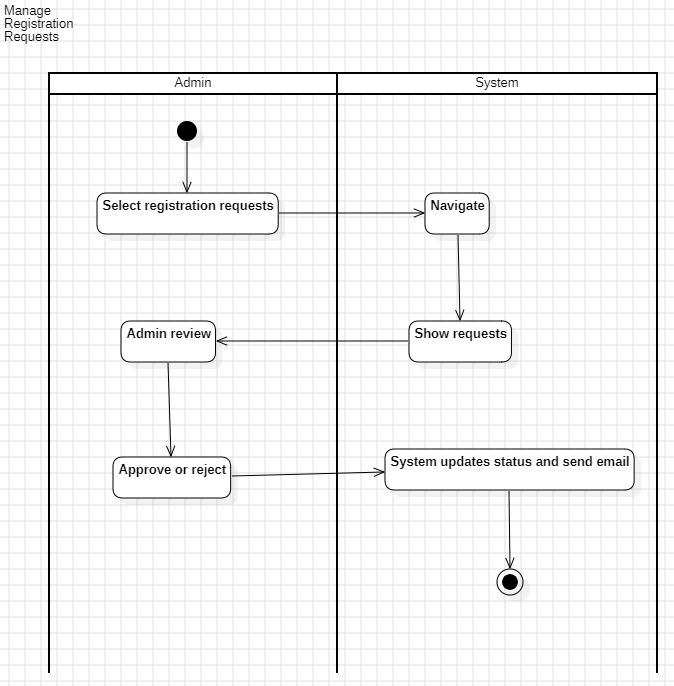
1. **Manage Discounts**

****

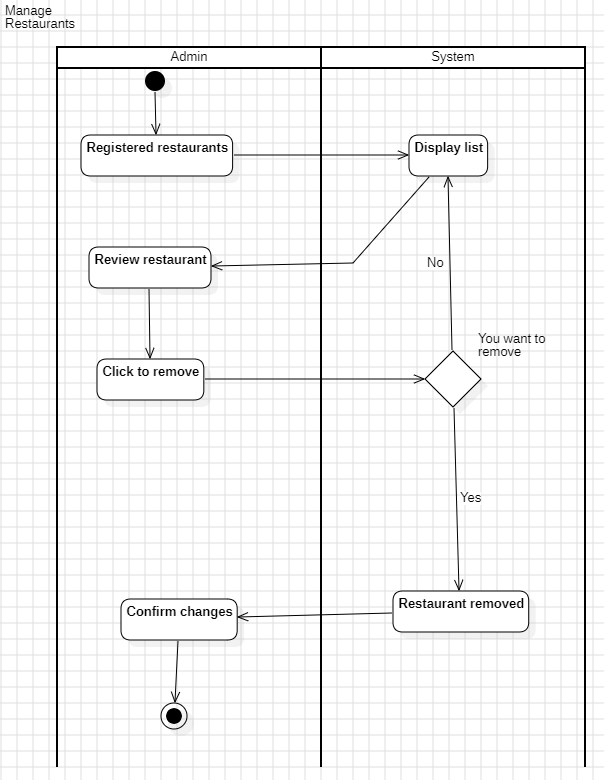
1. **Edit restaurant profile**

****

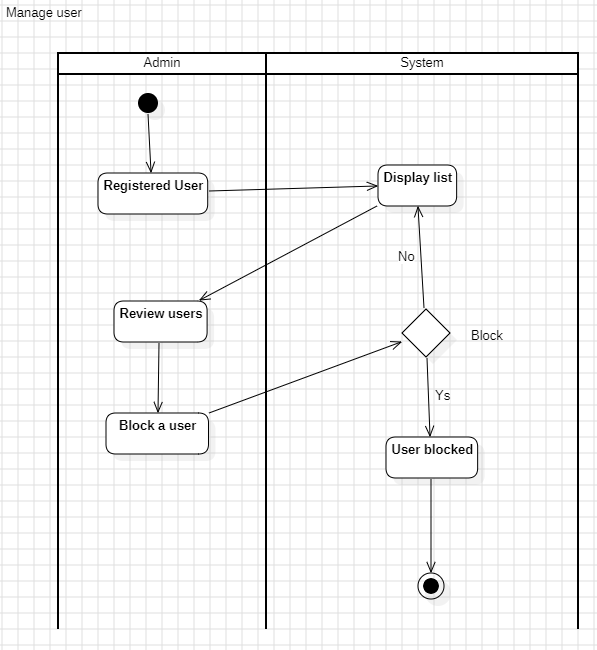
1. **Manage registration requests**

****

1. **Manage restaurant**

****

1. **Manage Users**

****

**Chapter 5:**

**Implementation**

**Chapter 5:**

**Implementation**

#### 5.1 Endeavour:

**5.1.1 Team**

Syed Kamran Shah

Safiullah Shahid

Muhammad Ismail

**5.1.2 Work Breakdown Structure**

**Project Management**

Work Breakdown Structure (WBS)

* Roles
  + **Syed Kamran Shah** -Frontend(Restaurant), Documentation, Design(Figma)
  + **Safiullah** **Shahid** - Backend(User, Restaurant), Frontend(User, Restaurant)
  + **Muhammad** **Ismail**-Frontend(Admin), Backend(Admin, Restaurant), Documentation
* Methodology
  + Incremental

**5.2 Components, Libraries, and APIs**

Key technologies and libraries used in the development include:

**Frontend Framework:** Flutter for cross-system mobile application development.

**Backend Framework:** Node.js with Express.js for API development.

**Database:** MongoDB for managing user, restaurant, and reservation data.

**Payment:** API for secure transactions.

**5.3 IDE, Tools, and Techniques**

The following tools and techniques facilitated the development process:

**IDE:** Visual Studio Code for coding and debugging.

**Version Control:** Git for collaboration and source code management.

**Testing Frameworks:** Postman for API testing.

**5.4 Best Practices and Coding Standards**

The development adhered to industry best practices to facilitate scalability, maintainability, and performance:

**Modular Code Structure:** To allow easy updates and new feature integration.

**Responsive Design:** Optimized for mobile devices with varying screen sizes.

**Secure Development:** Encryption for sensitive data.

**5.5 Test cases:**

For assuring the quality of the system, testing is the most essential step for assuring the quality of the system. The purpose of testing is to find out the system errors and bugs in the system. So, in this chapter we will discuss testing of our Dine-Deal Mobile app. We will create the test cases to check that the system is working correctly while completing the necessary requirements.

**5.6 List of Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Scenario** | **ECP** | **Expected Output** |
| **TC001** | Register with valid details | Valid Username, Valid Email, Valid Password | Registration successful, user redirected to login page. |
| **TC002** | Register with invalid email | Valid Username, Invalid Email, Valid Password | Error: "Invalid email." |
| **TC003** | Register with small password | Valid Username, Valid Email, small Password | Error: "Password Length must be more than 6." |
| **TC004** | Login with valid credentials | Valid Username, Valid Email, Valid Password | Login successful, user redirected to dashboard. |
| **TC005** | Login with invalid email | Valid Username, Invalid Email, Valid Password | Error: "Invalid email." |
| **TC006** | Login with wrong password | Valid Username, Valid Email, Invalid Password | Error: "Invalid password." |
| **TC007** | Forgot password with valid email | - | Verification code sent to email. |
| **TC008** | Make reservation with valid details | Valid Restaurant, Valid Date, Valid Time | Reservation successful |
| **TC09** | Make reservation with unavailable slot | Valid Restaurant, Valid Date, Unavailable Time | Automatically select first time. |