# Software Requirements Specification

# PRJ566 – Winter 2025

# PRJ566 – Team No: 07

# Name of Project:   Restaurant Waitlist Application

# Project Leader: Marcus Brown (rotating weekly)

**Last updated: January 26, 2025**

**Team Members:**

**1. Marcus Brown**

**2. Sasawat Yimleang**

**3.**  **Jie Xu**

**4.**  **Kittikom Kornmanakij**

**5.** **Ario Nazemirad**

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. **Document Authors**
   2. **Revision History**
   3. **Document Conventions**
   4. **Document Purpose**
   5. **Intended Audience**
   6. **Group Agreement**
2. **Project Overview**
   1. **Project Proposal**
   2. **Stakeholders and Users**
   3. **Functional Requirements**
   4. **Non-Functional Requirements**
   5. **Project Scope**
   6. **System risks**
   7. **Operating Environment**
   8. **UI/UXD Interface Mockups**
3. **Process & Data Modeling**
   1. **UML Modeling: DFDs & Activity Diagrams**
   2. **Use Case Specification** 
      1. **Business Rules**
      2. **System Use Case Diagrams**
      3. **Use Case Description Tables**
4. **Domain Class Diagram**
5. **c(Select either 5.1 or 5.2)**
   1. **RDBMS Artifacts**
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
   2. **NoSQL Artifacts**
6. **Work breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## 1.1 Document Authors

* Marcus Brown
* Sasawat Yimleang
* Jie Xu
* Kittikom Kornmanakij
* Ario Nazemirad

## 1.2 Revision History

|  |  |
| --- | --- |
| Week 03 | 1. Introduction/Overview    1. Document Authors (Completed)    2. Revision History (Ongoing)    3. Document Conventions (Completed)    4. Document Purpose (Completed)    5. Intended Audience (Completed)    6. Group Agreement (Completed) 2. Project Overview    1. Project Proposal (Completed) |
| Week 04 | 1. Introduction/Overview    1. Revision History (Ongoing)    2. Document Conventions (Ongoing) 2. Project Overview   2.2. Stakeholders and Users (Completed)  2.5. Project Scope (Completed)  2.6. System Risks (Completed)  2.7. Operating Environment (Completed) |
| Week 05 | 1. Introduction/Overview   1.2. Revision History (Ongoing)  1.3. Document Conventions (Completed)  2. Project Overview  2.3. Functional Requirements (Completed)  2.4. Non-Functional Requirements (Completed)  3. Process and Data Modeling  3.1. UML Modeling: DFDs & Activity Diagrams (Ongoing) |
| Week 06 | 1. Introduction/Overview  1.3. Revision History (Ongoing)  2. Project Overview  2.8. UI/UXD Interface Mock-ups (Completed)  3. Process and Data Modeling  3.1. UML Modeling: DFDs & Activity Diagrams (Completed) |
| Week 07 | 1. Introduction/Overview  1.3 Revision History (Ongoing)  3. Process and Data Modeling  3.2. Use Case Specification (Completed) |
| Week 08 |  |
| Week 09 | 1. Introduction/Overview  1.3 Revision History (Ongoing)  4. Domain Class Diagram (Completed) |
| Week 10 | 1. Introduction/Overview  1.3 Revision History (Ongoing)  5. Database  5.1 Data Model (Completed)  5.2 Data Sample (Completed) |
| Week 11 | 1. Introduction/Overview  1.3 Revision History (Ongoing)  6. Work breakdown Structure (WBS)  6.1 Front-End (Completed)  6.2 Back-End (Completed)  6.3 Web Hosting (Completed)  6.4 Database (Completed)  6.5 Cloud Storage (Completed)  7. Milestones & Acceptance Criteria  7.1 Milestones (Completed)  7.2 Acceptance Criteria (Completed) |
| Final | 1. Introduction/Overview  1.3 Revision History (Completed)  8. Implementation Schedule (Agile/Waterfall) (Completed) |

## 1.3 Document Conventions

### Text Formatting Conventions

* 1. Any text in red indicates an exception or error.
  2. Any text in blue is in-progress.
  3. Any text highlighted in yellow is an important point.
  4. Any text in green was recently added.
  5. Any text *italicized* represents definitions.
  6. Any text with ~~strike-through~~ is deleted.

### Terminology and Definitions

* 1. **Administrator** is a system user with full access to manage restaurant profiles, waitlists, and promotional offers.
  2. **Restaurant Staff** are users who can manage and update restaurant waitlists and reservations.
  3. **Customers** are end-users who join waitlists, receive notifications, and interact with restaurant information.
  4. **Waitlist** is a virtual queue that allows users to reserve a table at a restaurant.
  5. **QR Code** is a unique scannable code that links users to a restaurant’s waitlist.
  6. **Push Notification** is real-time alerts sent to users about their waitlist status.

### Acronyms and Abbreviations

* 1. **API** is Application Programming Interface
  2. **QR Code** is Quick Response Code
  3. **SMS** is Short Message Service

## 1.4 Document Purpose

The purpose of the Software Requirements Specification (SRS) document is to serve as a guide for the development of the restaurant waitlist application. The document will be a reference for all stakeholders throughout the project lifecycle and will define the requirements, scope, and specifications needed to ensure the successful design, development, and deployment of the planned application.

Specifically, this document will outline:

* A detailed breakdown of key functional and non-functional requirements.
* The core features of the application, such as QR code check-in, interactive mini-games, and discounts.
* Identification of potential risks, constraints, and dependencies that may affect development or deployment.
* A framework for collaboration amongst team members, ensuring alignment at each stage of the project.

## 1.5 Intended Audience

The intended audience for this document includes:

* **The development team** (PRJ566 Team 7) – They will use this document to guide the design, implementation, and testing of the system.
* **The instructor or evaluator** – The document serves to communicate the team’s progress, design decisions, and approach to building the system for evaluation purposes.
* **Future developers or students** – This document may serve as a reference for others working on similar projects or studying software architecture, helping them understand the design decisions, structure, and implementation.
* **Project stakeholders** – If applicable, this document might be used to update stakeholders or others with an interest in the project’s goals and progress, providing insight into the system’s functionality and benefits.

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #: 7**

**Project Title: Restaurant Waitlist Application**

**Project Time Frame: January 6, 2025 – April 17, 2025**

**Team Members: Kittikom Kornmanakij, Marcus Brown, Jie Xu, Sasawat Yimleang, Ario Nazemirad**

**Team Leadership: Marcus Brown (rotating weekly)**

**Team Functions:**

* *We will share information through MS Teams, OneDrive, WhatsApp, e-mail and meetings.*
* Attend team online meetings on a regular basis and participate in discussions
* Complete assigned tasks, progress reports, and other documents needed

**Team Meetings:**

* **Frequency:** Weekly
* **Day and Time**: Every Wednesdays at 1:35PM
* **Location**: Online via MS Teams

**Team Problems:**

If any challenges arise, they will be addressed through the following process:

* Discuss the issue within the group and try to resolve it collaboratively.
* Seek guidance from the instructor if unresolved.
* Document the problem and agreed resolution for future reference.

**Team Commitment**

**The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.**

**Name Date**

|  |  |
| --- | --- |
| Ario Nazemirad | 26/01/2025 |
| Sasawat Yimleang | 26/01/2025 |
| Kittikom Kornmanakij | 26/01/2025 |
| Jie Xu | 26/01/2025 |
| Marcus Brown | 26/01/2025 |

ShapeShapeShapeShapeShapeShapeShapeShape

# 2 - Project Overview

## 2.1 Project Proposal

Project Background

Managing customer waitlists during peak hours poses a significant challenge for restaurants. Traditional systems, such as physical queues and SMS notifications, often result in inefficiencies, miscommunication, and customer dissatisfaction. Additionally, the lack of engaging options during waiting periods further detracts from the overall dining experience. As restaurants increasingly embrace digital tools like QR codes for menus, integrating waitlist management into this technological shift is a natural progression to improve operations and customer engagement.

**Problem Statement**

|  |  |
| --- | --- |
| The Problem of: | Inefficient waitlist management systems leading to operational challenges and customer dissatisfaction |
| Affects: | * People who face increased stress during peak hours of the restaurants or shops * People who experience delays and a lack of engagement |
| The impact of which is: | Delays, miscommunication, disputes, and a lack of engagement during wait times |
| A successful solution would: | Streamline the waitlist process, enhance customer satisfaction with gamification and rewards, and increase operational efficiency for restaurants |

**Product Vision**

|  |  |
| --- | --- |
| For | Restaurants and their customers |
| Who | Seek a seamless and enjoyable waitlist management experience |
| Restaurant Waitlist Application | Web-application |
| That | Simplifies waitlist operations, gamifies waiting, and provides insights for restaurants |
| Unlike | Traditional physical queues or basic SMS-based systems |
| Our product | Integrates QR code technology, gamification, and customer data analytics to provide a unique and engaging solution |

## 2.2 Stakeholders and Users

|  |  |
| --- | --- |
| Stakeholder Name/Identifier | Category |
| Yasser Elmankabady / CEO (Chief Executive Officer) | Sponsor |
| Yasser Elmankabady / Project Manager and Scheduler | Stakeholder (Needs accurate up to date information for costing and scheduling of project details) |
| Project Leader (Rotating Weekly) | Developers |
| Marcus Brown / Developer | Developers |
| Jie Xu / Developer | Developers |
| Kittikom Kornmanakij / Developer | Developers |
| Sasawat Yimleang / Developer | Developers |
| Marcus Brown / Developer | Developers |
| Restaurants / Administrators | Users |
| Restaurant staff | Users |
| Customers | Users |

## 2.3 Functional Requirements

1. User Registration and Authentication
   1. System will have the option for users to create an account using email or phone number.
   2. Users will be able to log in and log out securely.
   3. Users profile page will be displaying their waitlist history, preferences and rewards.
2. QR Code Integration
   1. Each restaurant will have a unique QR code linking to its waitlist page.
   2. Users will be able to scan the QR code to join the waitlist.
   3. The application will verify the restaurant’s unique QR code to prevent duplicating the waitlist.
3. Waitlist Management
   1. Users will be able to join a waitlist by providing their name and party size.
   2. Users will be able to view their real-time position in the queue and their estimated waiting time.
   3. Restaurants will be able to manage and update the waitlist (Such as remove no-show users)
4. Notification System
   1. Users will be able to receive notifications via SMS or application notifications.
   2. Users will be notified when their table is ready.
   3. Users will have the option to turn notifications on or off.
5. Mini Games and Rewards System
   1. Users will be able to play mini games while waiting.
   2. Users will earn points based on gameplay performance.
   3. Users will be able to redeem points for restaurant discounts.
6. Restaurant Information
   1. Restaurants will be able to upload and manage their menu through an admin panel.
   2. The application will display restaurant hours, contact details, promotions and other relevant information.
   3. Users will be able to browse the restaurant menu before their table is ready.
7. User Reviews and Social Features
   1. Users will be able to leave reviews for restaurants after their visit.
   2. Users will be able to connect with friends and view their reviews.
   3. Restaurants will be able to respond to customer reviews.

## 2.4 Non-Functional Requirements

### Operational Requirements

* 1. The system should be accessible through any modern browser (Chrome, Firefox, Edge, Safari).
  2. The system should be available as a mobile application for Android.
  3. The system should support QR Code scanning functionality.
  4. The system should support push notifications for real-time waitlist updates.

### Performance Requirements

* 1. The system will respond immediately to user actions (sends an API request). If a request requires database access, a loading indicator will be displayed.
  2. The application should process real-time updates with minimal delay.

### Security Requirements

* 1. Administrators will have access to:
     1. Managing restaurant profiles, waitlist setting and promotional offer.
     2. Updating restaurant hours and contact information.
  2. Restaurant staff will have access to:
     1. Managing and updating the waitlist.
     2. Viewing customer details required for reservation confirmation.
  3. Customers will have access to:
     1. Viewing their waitlist history and reservation details.
     2. Viewing restaurant's menu and information.
     3. Managing notification preferences and account settings.

### Cultural & Political

* 1. The system will use the local currency and language (Canadian dollars/ Canadian English).

## 2.5 Project Scope

### Project Goal and Objectives

The goal of this project is to develop a **web-based restaurant waitlist management**  **system** that enhances the customer experience and improves restaurant operations.

The **key objectives** are:

* To create an efficient digital waitlist system that reduces miscommunication and delays.
* To enhance customer engagement by integrating gamification and reward-based incentives.
* To provide real-time updates and notifications to customers through QR code integration.
* To generate actionable insights for restaurant owners using customer data analytics.

### Project Boundaries

**In-Scope:**

1. Development of a web-based application for restaurant waitlist management.
2. Integration of QR codes for easy check-in and tracking.
3. Implementation of gamification to keep customers engaged while waiting.
4. Notification system via SMS, email, or in-app alerts.
5. Basic customer data analytics for restaurant insights.

**Out-of-Scope:**

1. Development of a mobile app (this version is web-based only).
2. Advanced AI-based queue predictions (basic estimates will be provided).
3. Integration with external loyalty programs beyond the in-app rewards system.

### Project Deliverables

* 1. Web Application: A functioning restaurant waitlist management system.
  2. QR Code Integration: For customer check-in and tracking.
  3. Gamification Features: Points, rewards, or mini games to engage waiting customers.
  4. Admin Dashboard: Allowing restaurants to manage waitlists and view analytics.
  5. User Notifications: Real-time status updates via preferred channels
  6. Database and Cloud storage.

### Success Criteria

The project will be considered successful if:

* 1. Restaurants can efficiently manage their waitlist with minimal manual intervention.
  2. Customers experience reduced wait-time frustration due to gamification and rewards.
  3. The system achieves at least 80% adoption among restaurant customers in trials.
  4. Restaurant managers report on improved operational efficiency in a user survey.

### Project Assumptions

* Restaurants will have stable internet connections to run the web application.
* Customers will have smartphones capable of scanning QR codes.
* Restaurant staff will be trained to use the system effectively.
* The application will comply with basic data privacy and security standards.
* The initial version will support English only, with potential for future localization.

## 2.6 System Risks

|  |  |  |
| --- | --- | --- |
| **Risk** | **Description** | **Response** |
| System Downtime | The application may experience downtime due to server failures or high traffic. | Use a reliable cloud hosting service and conduct performance testing. |
| QR Code Malfunction | Issues with QR code generation or scanning could prevent users from joining the waitlist. | Provide an alternative manual entry method. |
| Cross-Platform Compatibility | The app may not function correctly on different devices and browsers. | Conduct thorough compatibility testing across various platforms and ensure responsive design. |
| User Adoption Resistance | Restaurants and customers may be reluctant to adopt the new system. | Provide clear onboarding instructions, incentives for first-time users, and training for restaurant staff. |
| Competition from Existing Solutions | Other waitlist solutions may try to compete with our offering. | Differentiate the app with unique features (mini-games, loyalty rewards/discounts) and conduct market research to discover consumer needs. |
| Monetization Strategy | The project may struggle to generate revenue if monetization isn’t planned. | Implement subscription models, in-app purchases, or restaurant partnerships to sustain long-term profitability. |
| Scalability Issues | If the app gains popularity, it may struggle to handle increased user load | Design a scalable architecture using cloud services (AWS) and conduct stress testing |
| Stale Mini Games | If users play the same mini games every time, they may get bored and not have an enjoyable experience. | Update the offered mini games on a regular basis to ensure customers are engaged. |

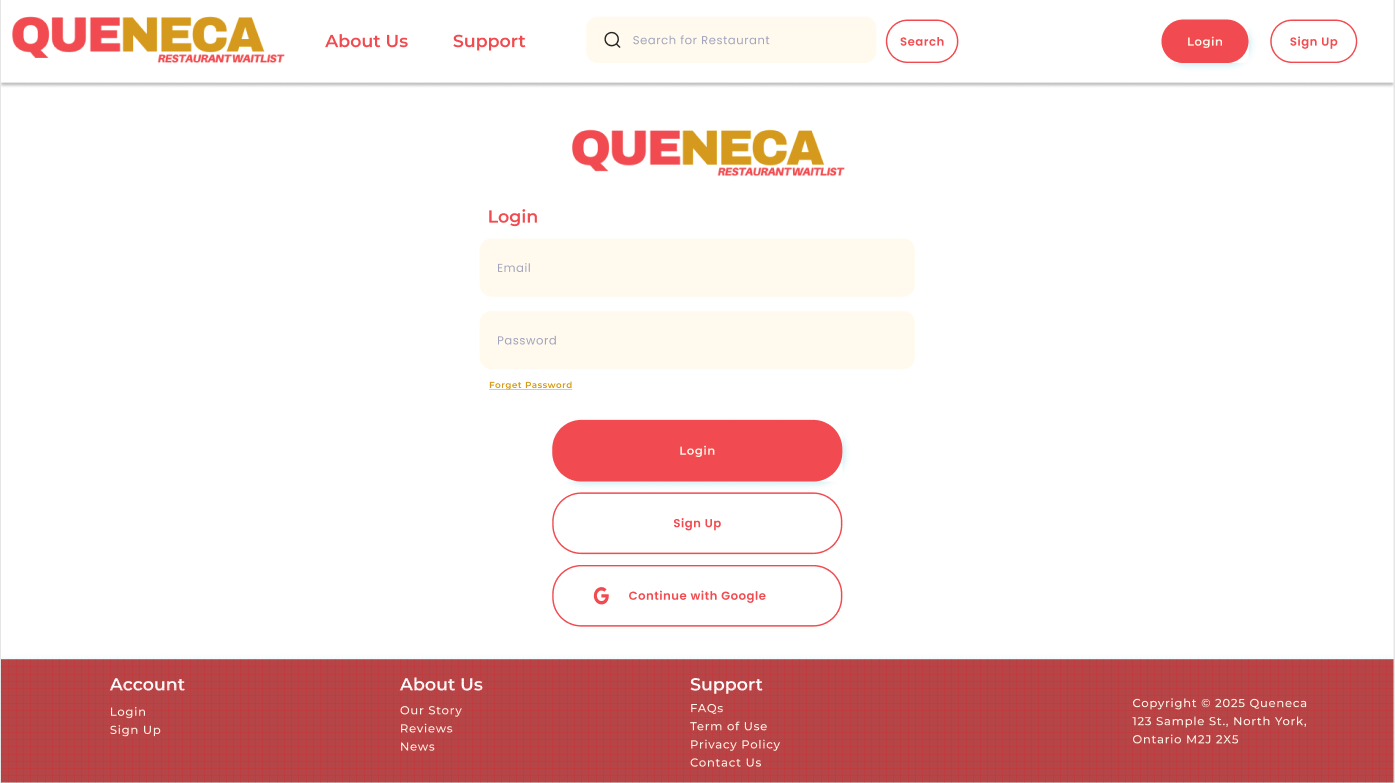
## 2.7 Operating Environment

* Server Operating System:
  + Windows
  + macOS
  + Linux
* Server Hosting: Netlify
* Database: MongoDB, PostgreSQL
* Platform:
  + Programming Language: JavaScript
  + API: NodeJS
  + Programming Framework: React
* Support Browsers
  + Google Chrome
  + Mozilla Firefox
  + Microsoft Edge
  + Safari

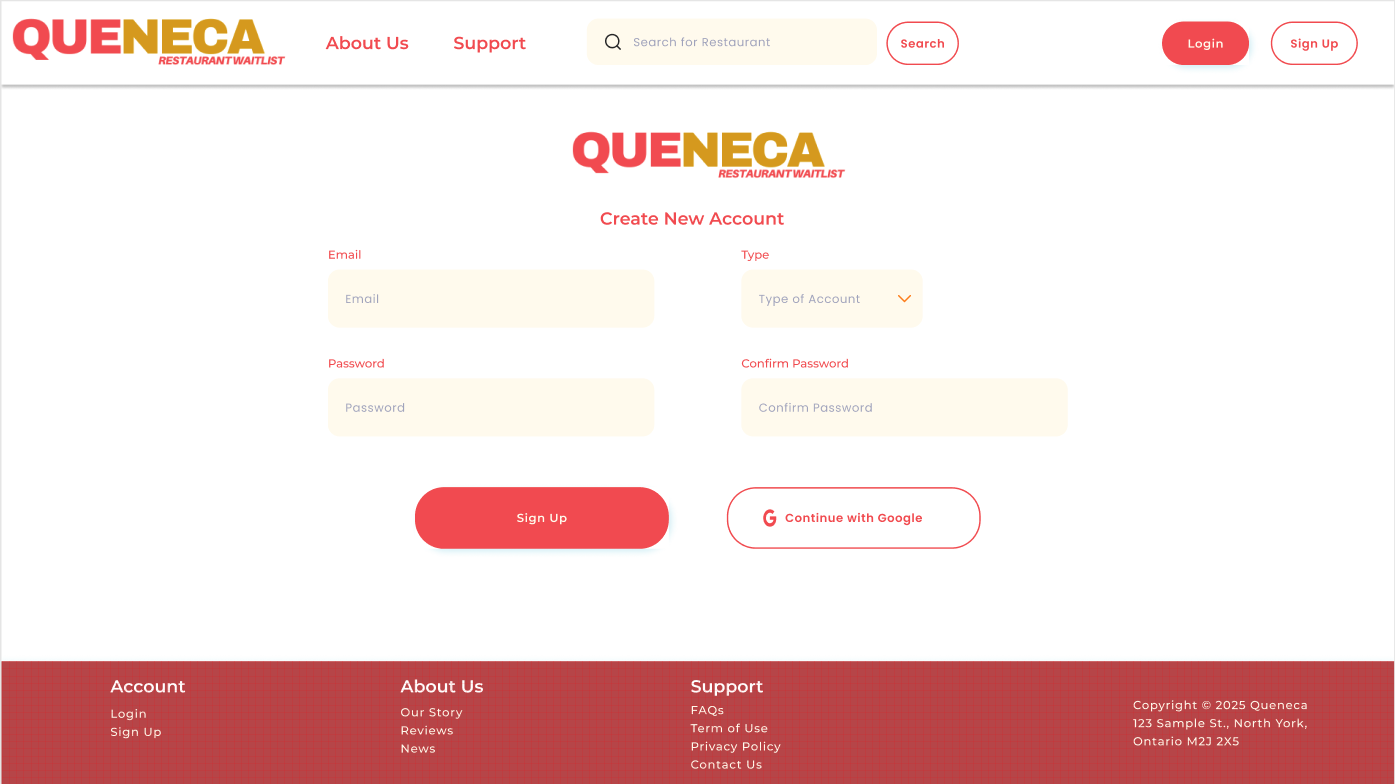
## 2.8 UI/UXD Interface Mock-ups

### 2.8.1 Login and Sign Up

#### 2.8.1.1 Login Page

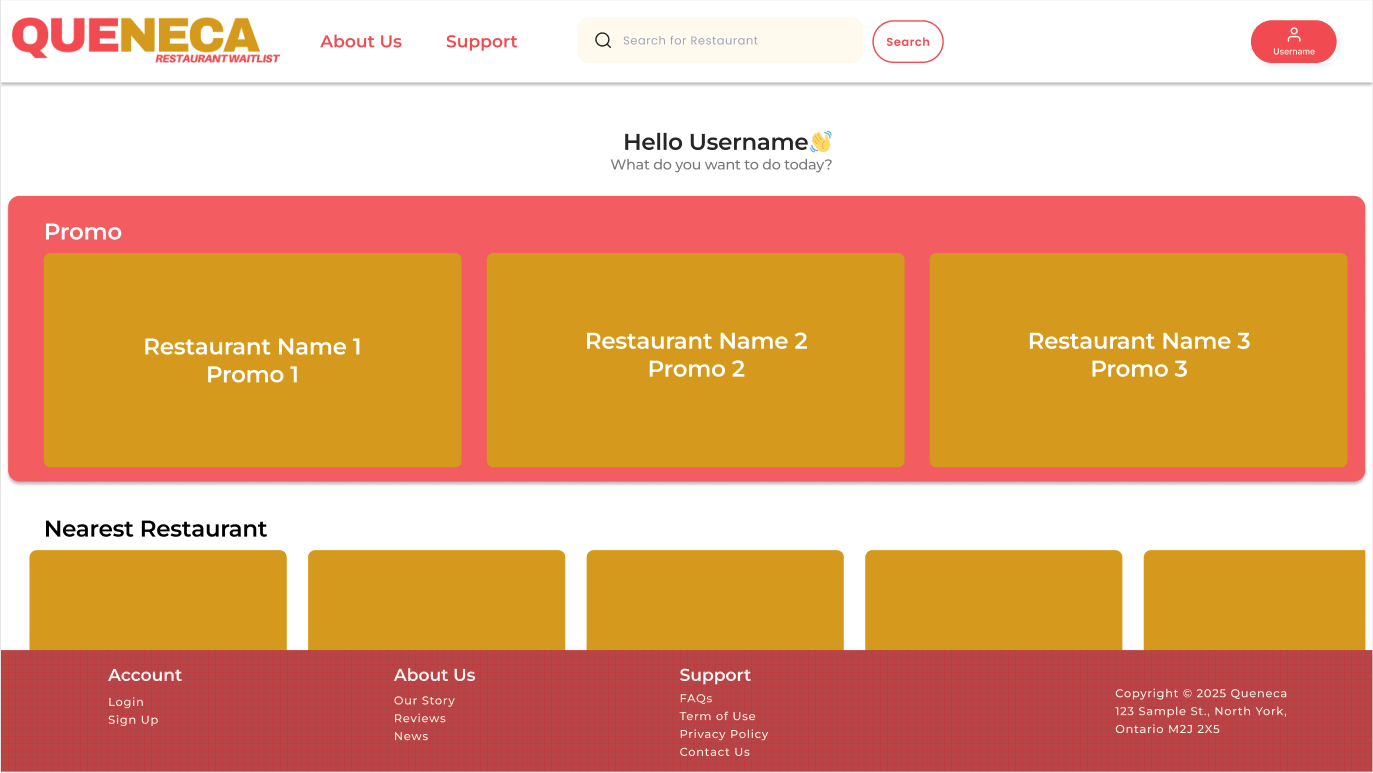


#### 2.8.1.2 Sign Up Page

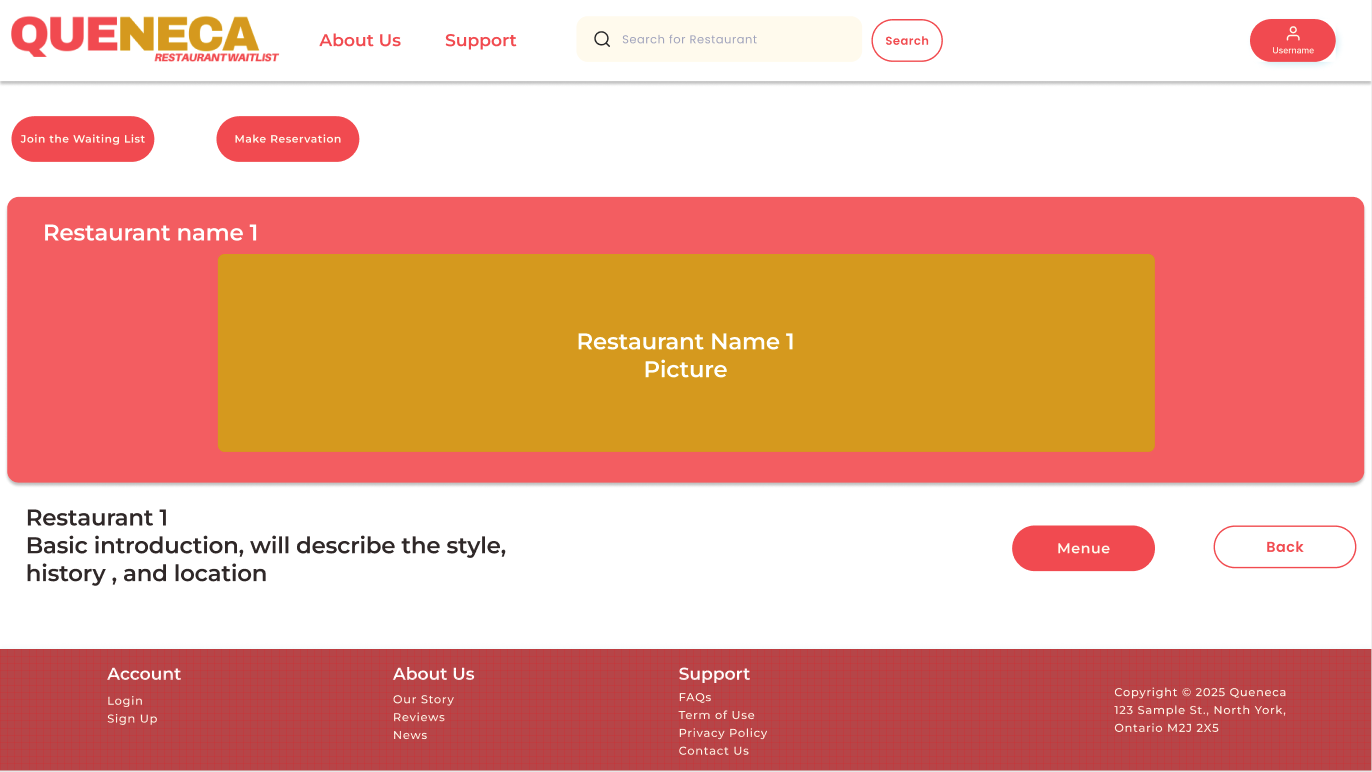


### 2.8.2 Customer

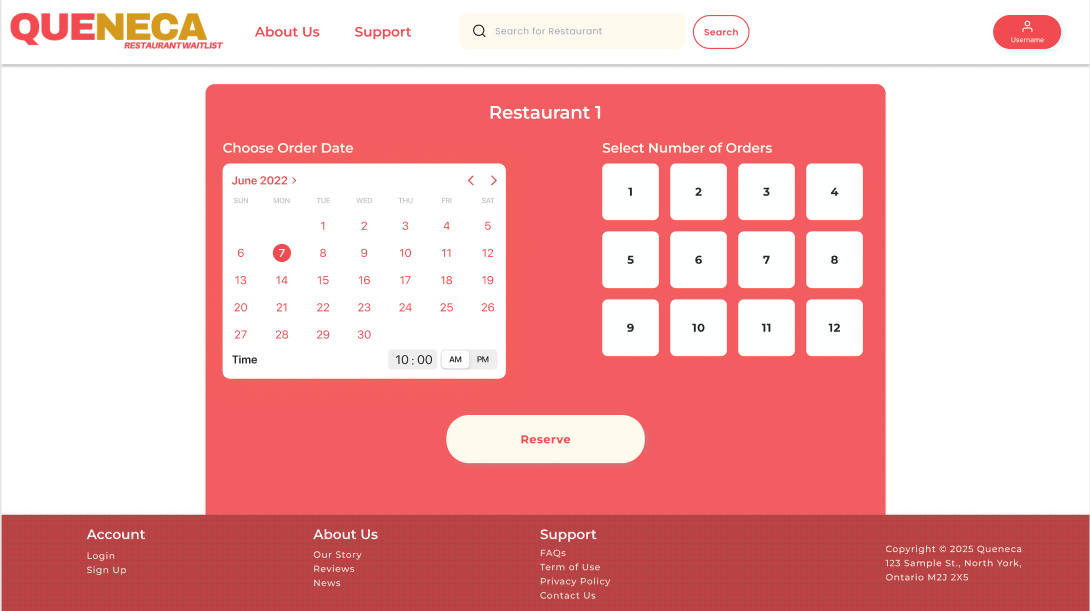
#### 2.8.2.1 Customer Landing Page



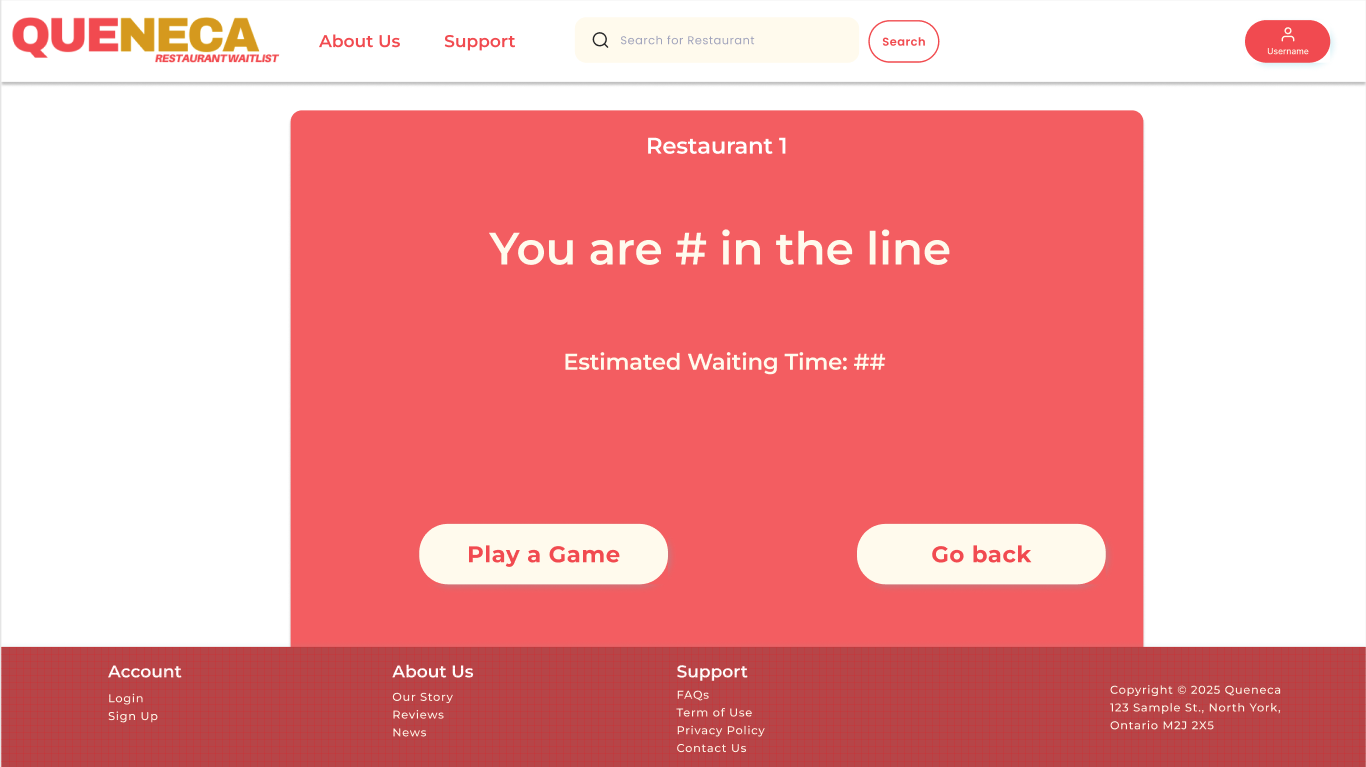
#### 2.8.2.2 Customer Select Restaurant Page



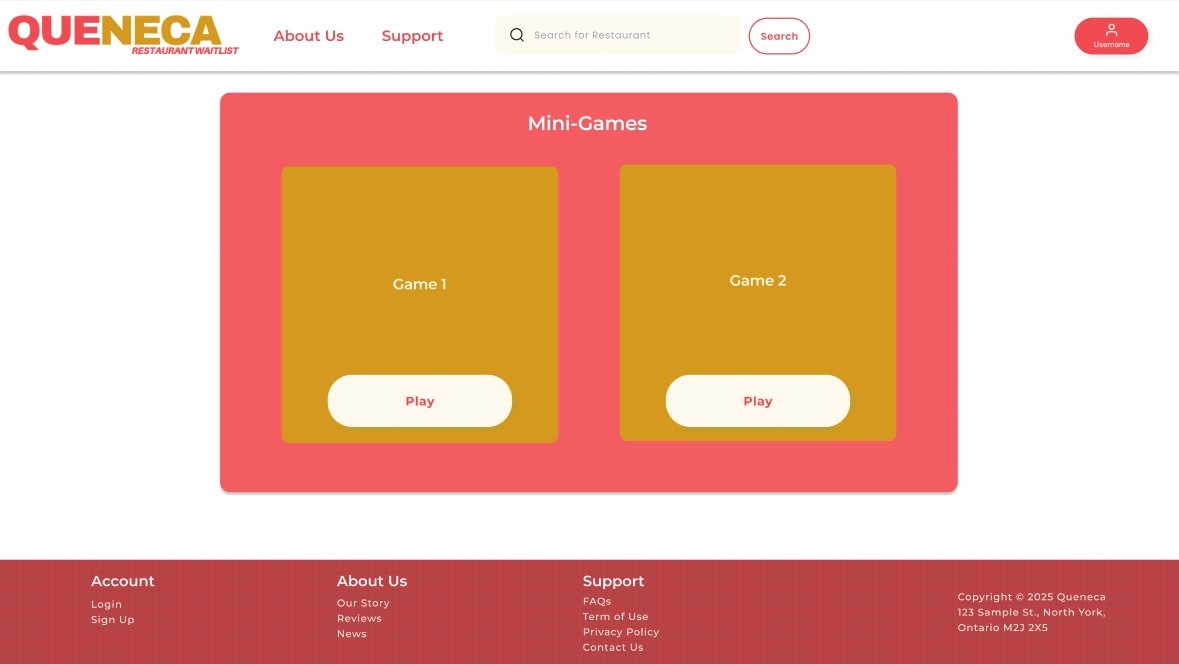
#### 2.8.2.3 Customer Reservation Page



#### 2.8.2.4 Customer Waiting List Page

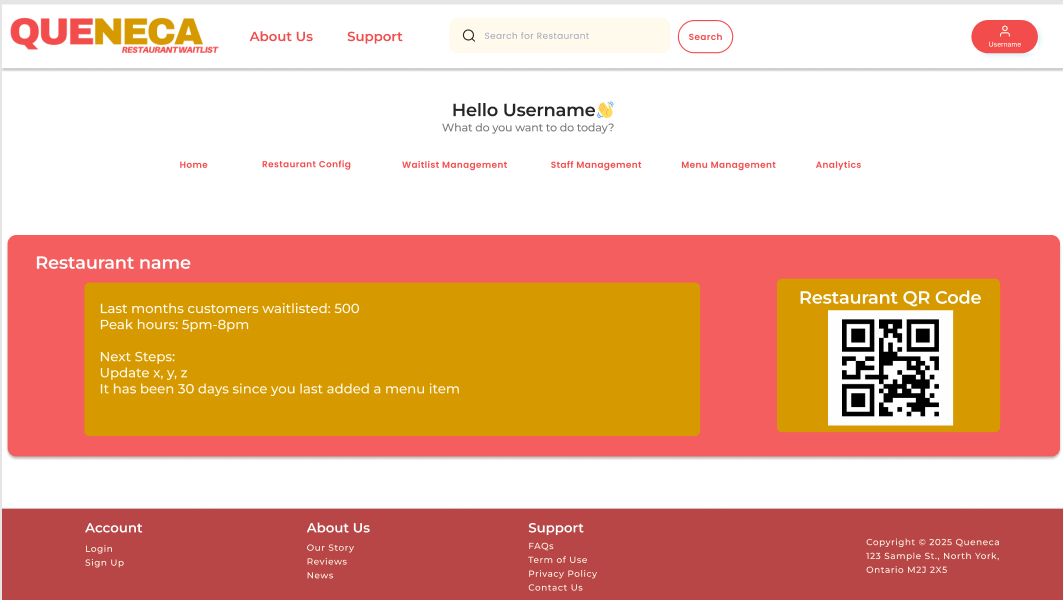


#### 2.8.2.4 Mini-Games Selection Page

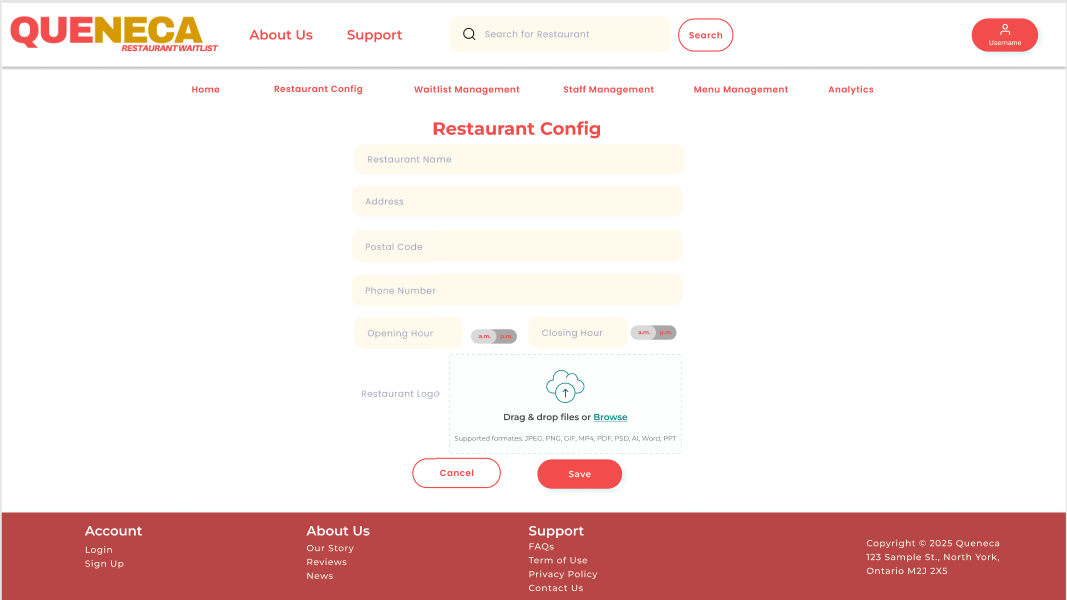


### 2.8.3 Administrator

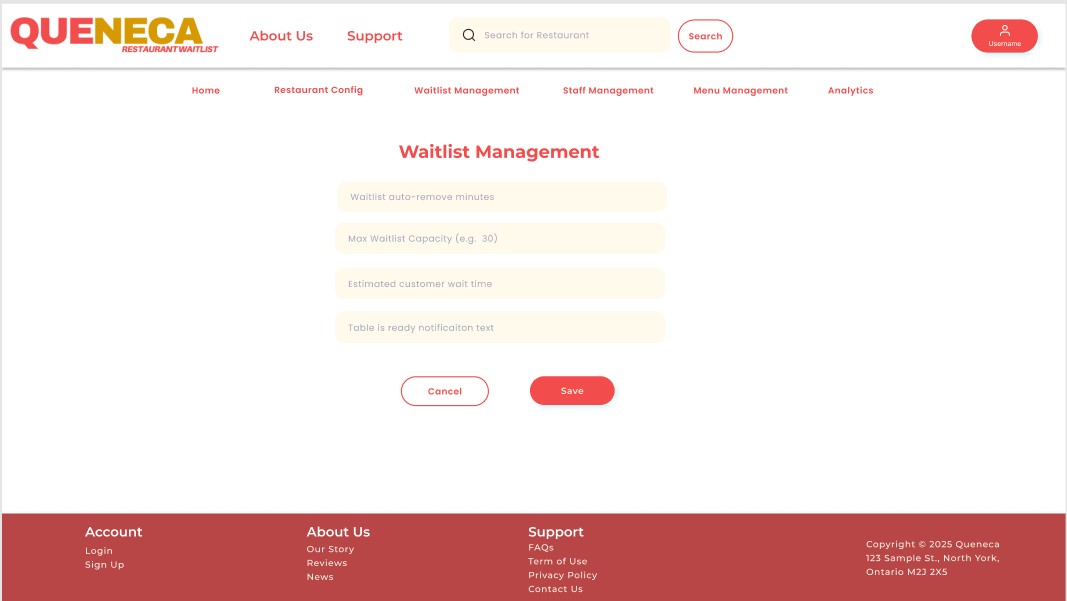
#### 2.8.3.1 Admin Home page



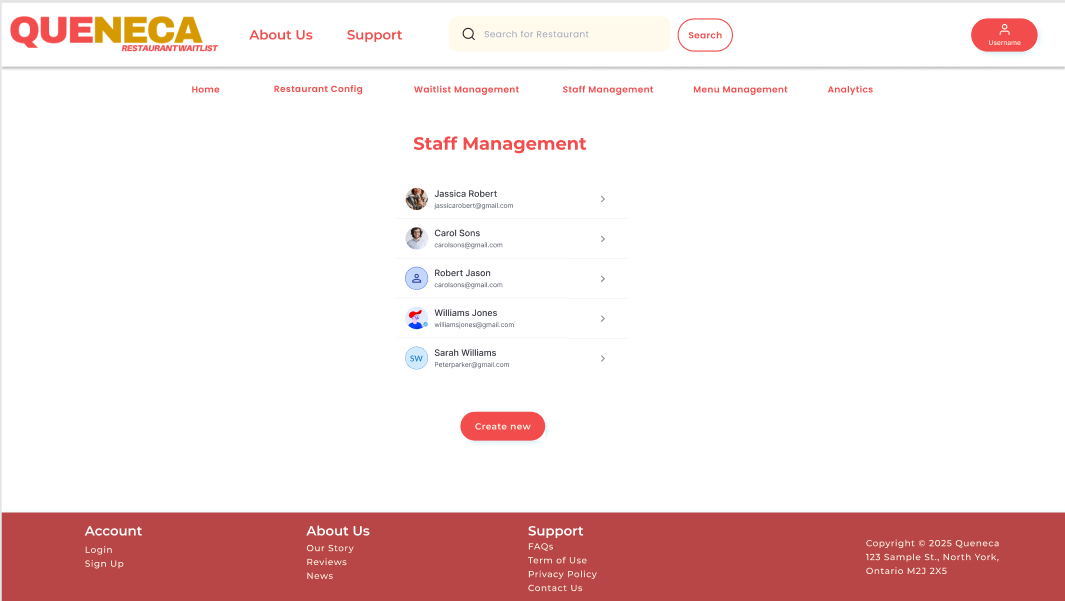
#### 2.8.3.2 Admin Restaurant Configuration page



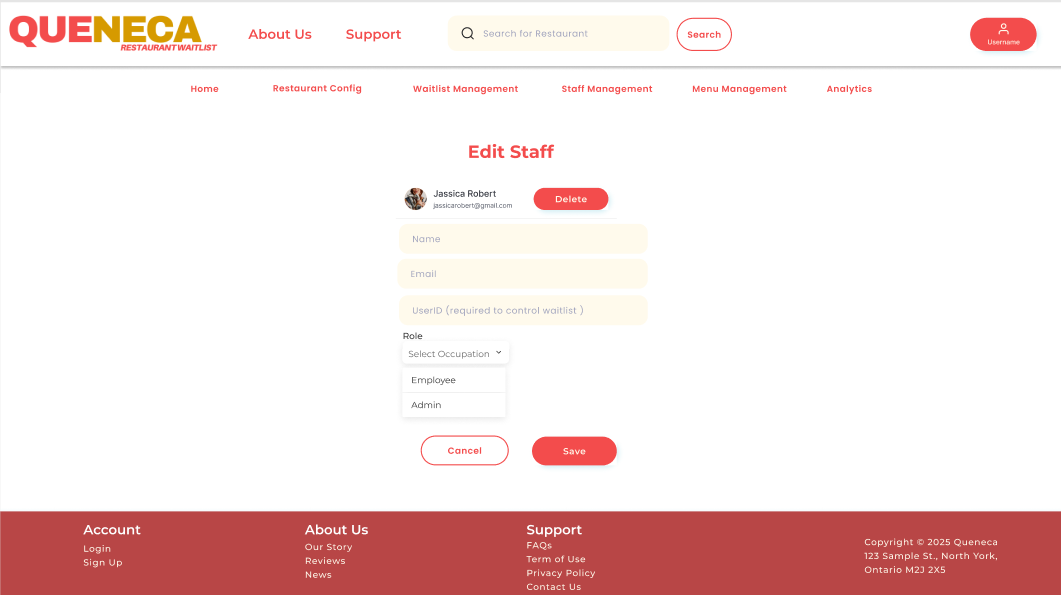
#### 2.8.3.3 Admin Waitlist Management Page



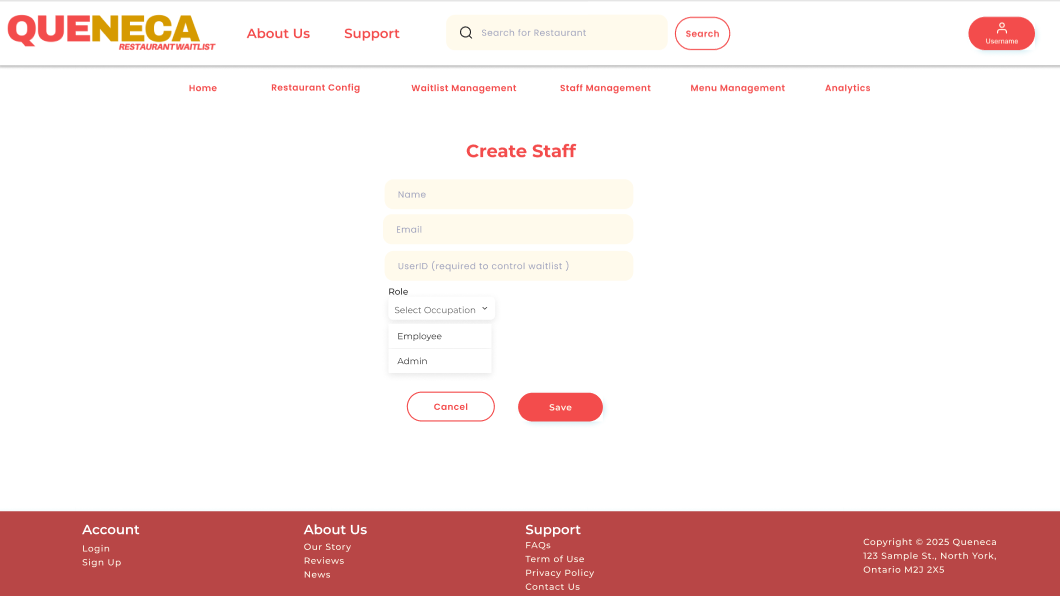
2.8.3.4 Admin Staff Management Page



##### *2.8.3.4.1 Admin Staff Edit Page*



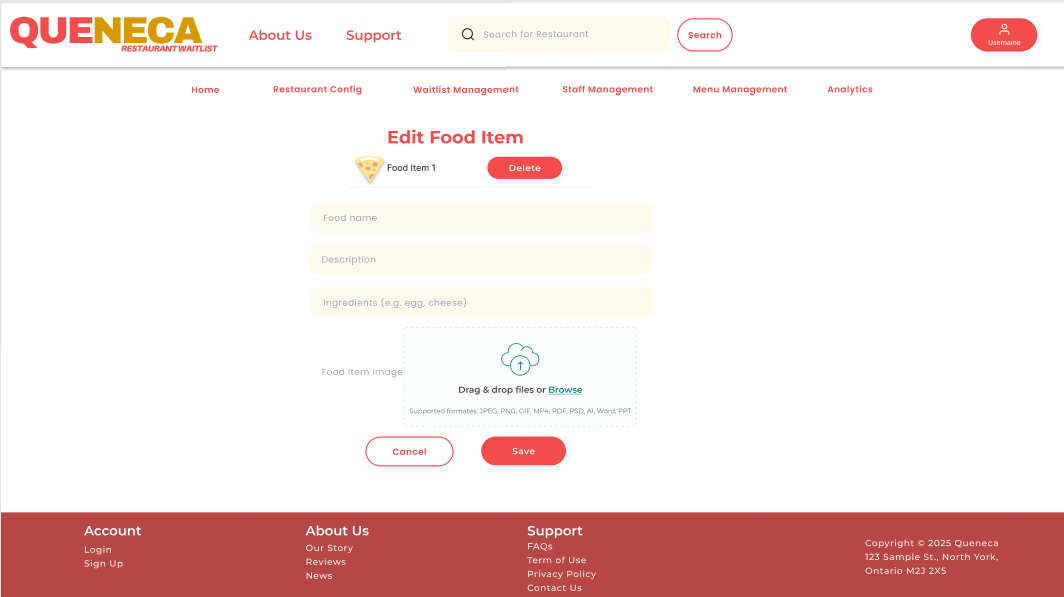
*2.8.3.4.2 Admin Staff Create Page*



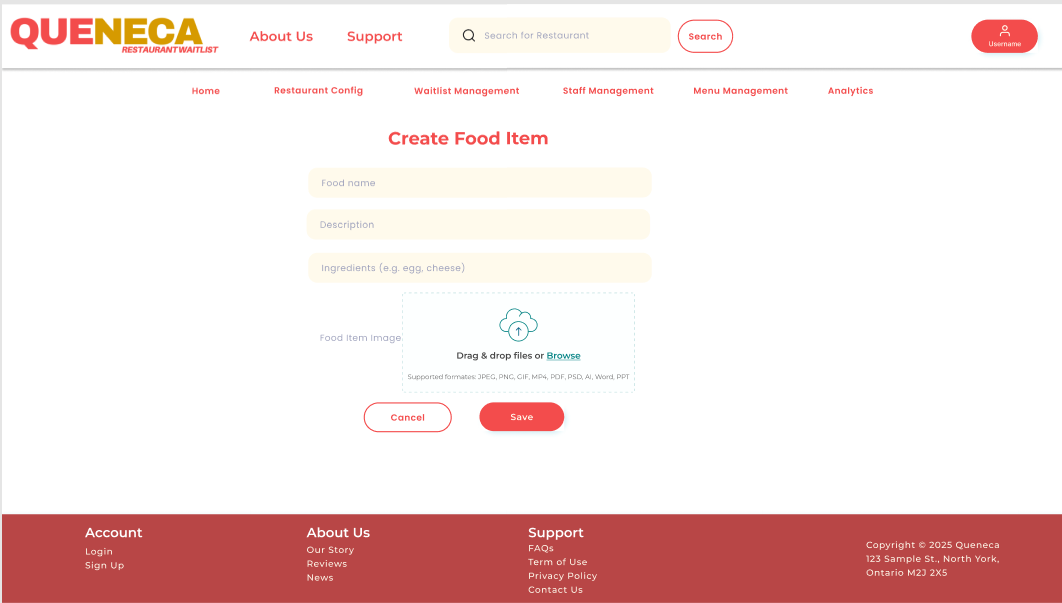
#### 2.8.3.5 Admin Menu Management Page

##### 

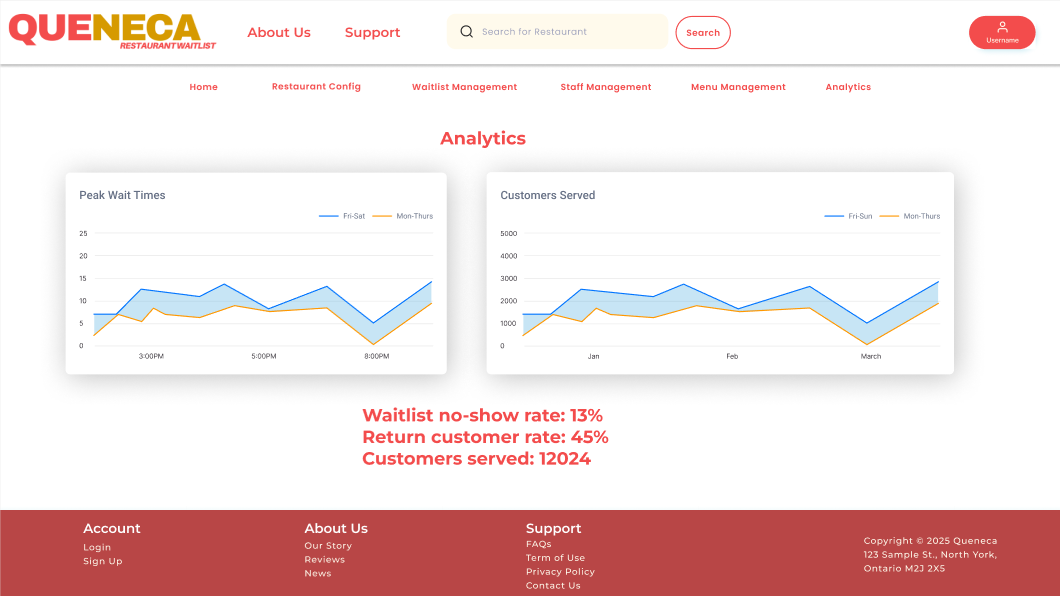
##### *2.8.3.5.1 Admin Menu Item Edit Page*



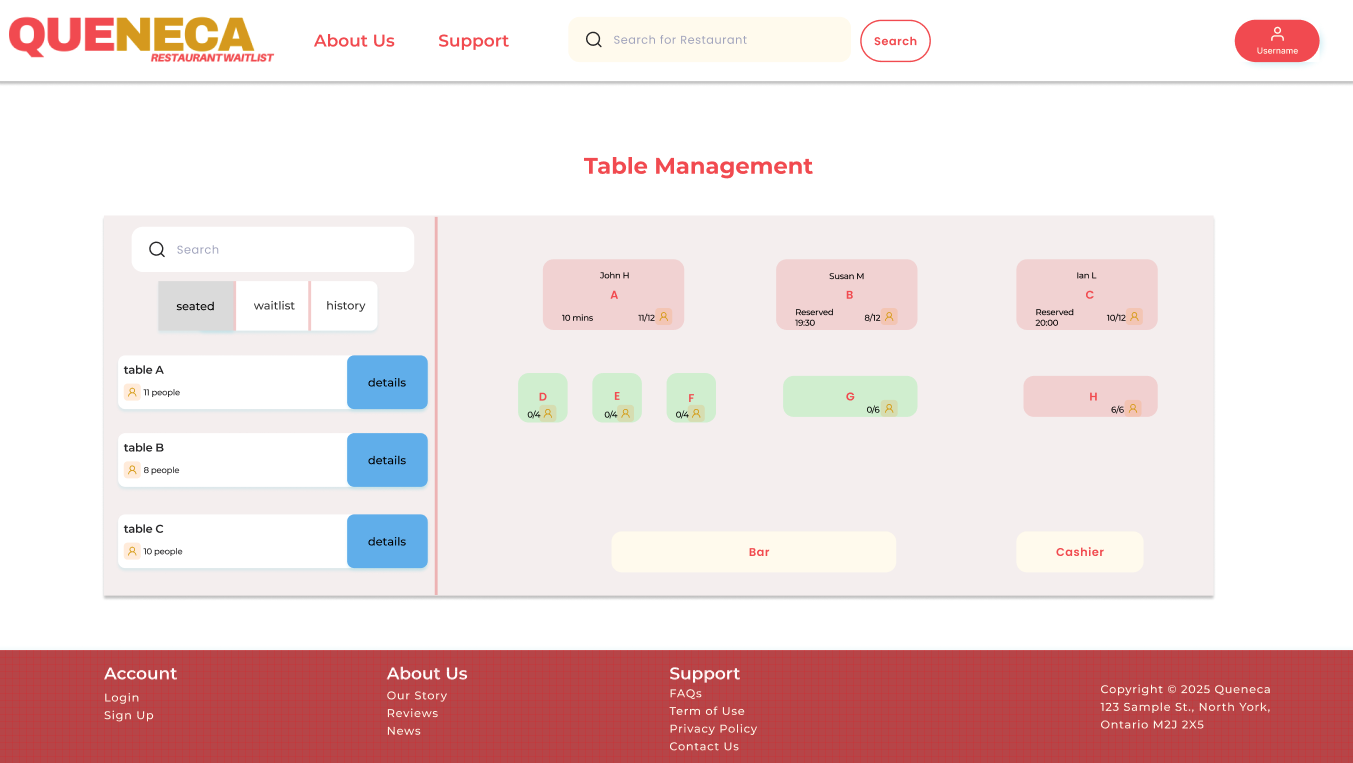
##### *2.8.3.5.2 Admin Menu Item Create Page*



#### 2.8.3.6 Admin Analytics Page



### 2.8.4 Restaurant Staff



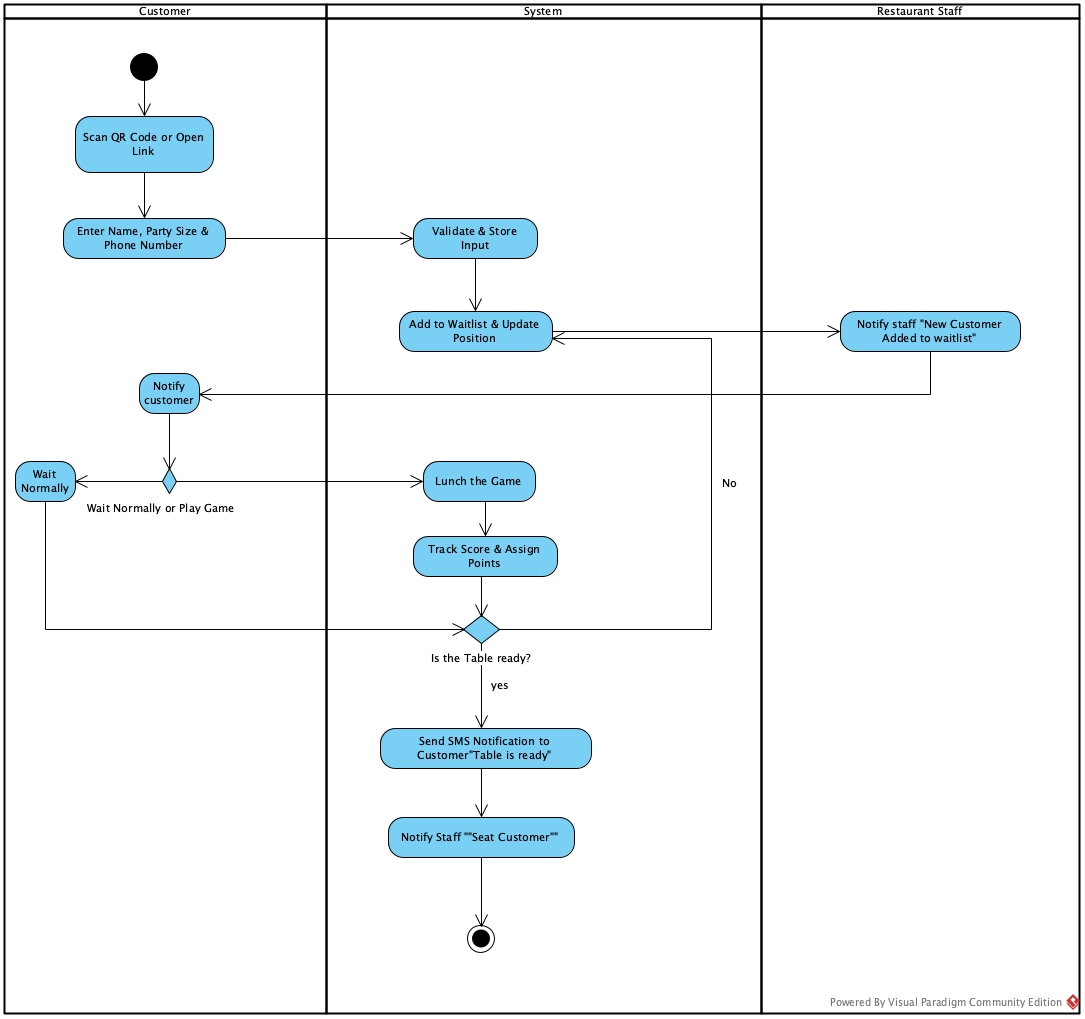
# Process and Data Modeling

## **3.1 UML/DFD Modeling and Data Modeling & Activity Diagrams**

### 3.1.1 Data Flow diagram

### 

### 3.1.2 Activity Diagrams

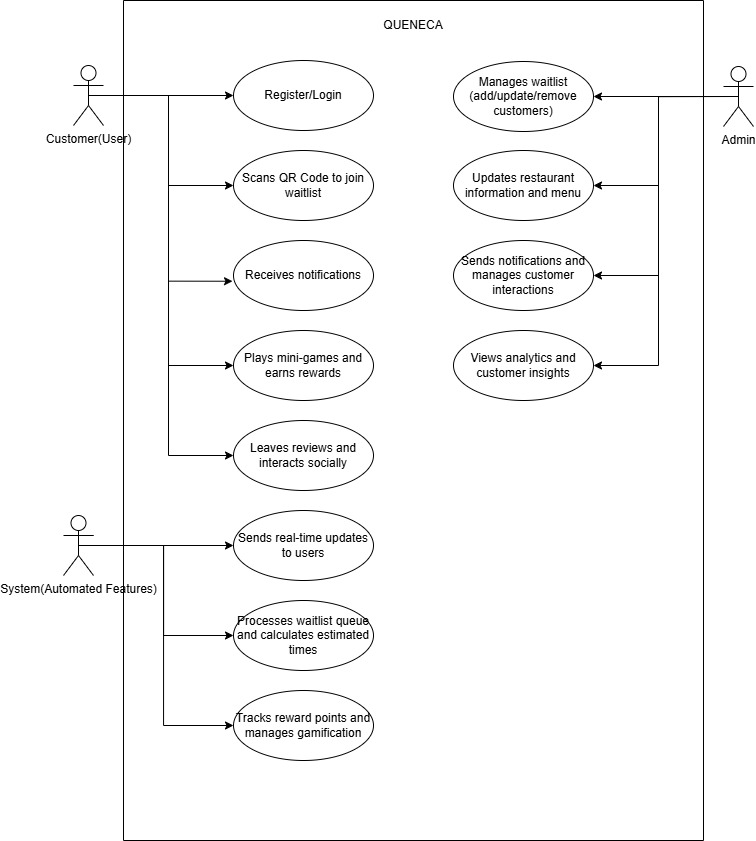


## **3.2 Use Case Documentation**

### 3.2.1 Business Rules

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | Users must provide a username, email and password to register for the app. | UC01 |
| BR02 | Post length can be no longer than 300 characters | UC09 |
| BR03 | Customers must scan the restaurant’s unique QR code to join the waitlist. | UC03 |
| BR04 | Customers must provide valid details (name, party size) to be added to the waitlist. | UC03, UC04 |
| BR05 | Administrators cannot misuse their privileges by accessing or sharing customers’ personal information beyond system requirements. | UC07 |
| BR06 | All restaurant administrators must process or decline waitlist updates (e.g., confirming reservations or removing no-show customers). | UC07 |
| BR07 | Administrators and restaurant staff must receive alerts for critical notifications, such as customer complaints or system issues. | UC07 |
| BR08 | If a customer is unavailable when their table is ready, they will be removed from the waitlist after a set grace period. | UC07 |
| BR09 | Restaurants must update estimated wait times dynamically based on real-time customer flow. | UC03, UC04 |
| BR10 | Users can only earn reward points by actively engaging with the mini games; inactive sessions will not generate rewards. | UC05 |
| BR11 | Reward points are non-transferable and can only be redeemed at the issuing restaurant. | UC05 |
| BR12 | Restaurants cannot delete customer reviews but are allowed to respond to them. | UC10 |
| BR13 | Customers must opt-in to receive notifications via SMS or app alerts; default settings will have notifications disabled. | UC01 |
| BR14 | Customers must confirm their arrival within a specific time after being notified that their table is ready; failure to do so results in removal from the queue. | UC03, UC04, UC07 |
| BR15 | System analytics will only collect non-sensitive customer data to generate insights for restaurant management. | UC07, UC09 |
| BR16 | Customers can only leave reviews for restaurants they have visited and checked in through the system. | UC09 |

### 3.2.2 System Use Case Diagram



### 3.2.3 Use Case Description Tables

#### UC01: User Registration

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | User Registration | | |
| Triggering Event | The user selects the "Sign Up" option. | | |
| Brief Description | Users register by providing a username, email, and password. | | |
| Actors | Customer | | |
| Related Use Cases | * User Login | | |
| Preconditions | * The user must have a valid email address. * The email provided must not already be registered in the system. | | |
| Post Conditions | The user account is successfully created. | | |
| Flow of activities | Actor | | System |
|  | 1. | User clicks the “Sign Up” button. | Display the registration page. |
|  | 2. | User enters username, email and password. | Validates the inputs. |
|  | 3. | User submits the form. | * Checks if email is unique. * Creates a new account. * Send a confirmation message. * Displays login page. |
| Exception Conditions | * If the email is already registered, an error message is displayed. * User cancels the registration. | | |

#### UC02: User Login

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | User Login | | |
| Triggering Event | The user selects the "Login" option. | | |
| Brief Description | User login by providing an email and password. | | |
| Actors | Customer | | |
| Related Use Cases |  | | |
| Preconditions | The user account must already be registered in the system. | | |
| Post Conditions | The user account is successfully login. | | |
| Flow of activities | Actor | | System |
|  | 1. | User clicks the “Login” button. | Display the login page. |
|  | 2. | User enters email and password. | Validates the inputs. |
|  | 3. | User submits the form. | * Checks if account existed. * Displays the landing page. |
| Exception Conditions | * If the email is already registered, an error message is displayed. * User cancels the login. | | |

#### UC03: Joining Waitlist via QR Code

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Joining Waitlist via QR Code | | |
| Triggering Event | The customer scans a restaurant's unique QR code | | |
| Brief Description | The customer joins a restaurant’s waitlist by scanning the QR code and providing required details. | | |
| Actors | Customer | | |
| Related Use Cases |  | | |
| Preconditions | * The restaurant must have an active waitlist. * The restaurant must have a valid QR code linked to the waitlist. * The customer already has an account. | | |
| Post Conditions | The customer is successfully added to the restaurant’s waitlist. | | |
| Flow of activities | Actor | | System |
|  | 1. | Customer scans the restaurant’s QR code. | Verifies the QR code and opens the restaurant’s reservation page. |
|  | 2. | * Customer selects the date and party size. * Customer clicks “Reserve” button. | * Add the customer to the waitlist. * Displays confirmation and estimated wait time. |
| Exception Conditions | * If the QR code is invalid, an error message is displayed. | | |

#### UC04: Joining Waitlist via Select Restaurant

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Joining Waitlist via Select Restaurant | | |
| Triggering Event | The customer clicks “Reserve” button on selected restaurant page. | | |
| Brief Description | The customer joins a restaurant’s waitlist by selecting the restaurant, reserving via the restaurant page and providing required details. | | |
| Actors | Customer | | |
| Related Use Cases |  | | |
| Preconditions | * The restaurant must have an active waitlist. * The customer already has an account. * The customer is already on the selected restaurant page. | | |
| Post Conditions | The customer is successfully added to the restaurant’s waitlist. | | |
| Flow of activities | Actor | | System |
|  | 1. | Customer clicks the “Reserve” button. | opens the restaurant’s reservation page. |
|  | 2. | * Customer selects the date and party size. * Customer clicks “Reserve” button. | * Add the customer to the waitlist. * Displays confirmation and estimated wait time. |
| Exception Conditions |  | | |

#### UC05: Playing Mini Games and Earning Rewards

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Playing Mini Games and Earning Rewards | | |
| Triggering Event | The customer starts a mini games session while waiting. | | |
| Brief Description | The customer plays mini games while waiting to earn reward points. | | |
| Actors | Customer | | |
| Related Use Cases | * Joining Waitlist via QR Code | | |
| Preconditions | * The customer must be on the waitlist. | | |
| Post Conditions | The customer earns points that can be redeemed at the restaurants. | | |
| Flow of activities | Actor | | System |
|  | 1. | Customer selects “Play a Game” button. | Load the mini game. |
|  | 2. | Customer plays the game actively. | Tracks user participation. |
|  | 3. | Customer completed the game. | * Awards points based on performance. * Update user’s reward balance. |
| Exception Conditions | * If the customer is inactive for a set period, the game session ends without awarding points. * If a system error occurs, the game restarts. | | |

#### UC06: Updating menu and promotions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Updating menu and promotions | | |
| Triggering Event | Staff updates menu and promotions of the restaurant | | |
| Brief Description | The restaurant staff updates menu items, add new promotions, or modify existing offers for customers | | |
| Actors | Staff | | |
| Related Use Cases | * Viewing menu and promotions | | |
| Preconditions | * The restaurant has active promotions and menu items * Staffs are logged in to the waitlist management system | | |
| Post Conditions | * The restaurant menu and promotions are updated * Customers can view the latest offers and menu items | | |
| Flow of activities | Actor | | System |
|  | 1. | Opens the menu management interface | Displays the current menu and promotions |
|  | 2. | Adds, updates, or removes menu items | Updates the restaurant database with new details |
|  | 3. | Adds or modifies promotional offers | Saves promotion details and adjusts customer-facing displays |
|  | 4. | Confirms and publishes changes | Updates the menu and promotions for customer viewing |
| Exception Conditions | * The system fails to save menu changes * Staffs enter invalid pricing or promotion details | | |

#### UC07: Managing waitlist

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Managing waitlist | | |
| Triggering Event | Staff updates queue information | | |
| Brief Description | The restaurant staff manages the customer waitlist by updating queue positions, seating customers, or removing them from the list | | |
| Actors | Staff | | |
| Related Use Cases | * Joining Waitlist via QR Code * Joining Waitlist via Select Restaurant | | |
| Preconditions | * Customers are actively joining the waitlist * Staffs are logged in to the waitlist management system | | |
| Post Conditions | * The waitlist is updated with accurate queue positions * Customers receive notifications when their tables are ready | | |
| Flow of activities | Actor | | System |
|  | 1. | Opens the waitlist management interface | Displays the current queue with customer details |
|  | 2. | Updates customer status (e.g., seated, removed, delayed) | Modifies the queue and updates the database |
|  | 3. | Sends a table-ready notification to a waiting customer | Sends an alert to the customer via the notification system |
|  | 4. | Removes a customer from the waitlist after seating | Updates the queue and adjusts waiting times |
| Exception Conditions | * A customer does not respond to a table-ready notification * System fails to update the queue | | |

#### UC08: Viewing menu and promotions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Viewing menu and promotions | | |
| Triggering Event | A customer accesses the restaurant app to check the available menu and ongoing promotions | | |
| Brief Description | Customers view and search for the restaurant's menu, view dish details, and check for ongoing promotions or discounts | | |
| Actors | Customer | | |
| Related Use Cases | * Updating menu and promotions | | |
| Preconditions | * The restaurant's menu and promotions are updated in the system | | |
| Post Conditions | * Customer views the updated menu items and promotions * Customer place an order based on the available options | | |
| Flow of activities | Actor | | System |
|  | 1. | Opens the restaurant | Displays the main menu page |
|  | 2. | Navigates to the menu section | Fetches and displays the restaurant menu |
|  | 3. | Clicks on a menu item to see details | Shows detailed information about the selected item |
|  | 4. | Navigates to the promotions section | Displays ongoing discounts, deals, or special offers |
|  | 5. | Exits the menu | Redirects to the selected feature or closes the app |
| Exception Conditions | * The system fails to load the menu * The customer tries to access an outdated or unavailable menu item | | |

#### UC09: Customer adds review

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Customer adds review | | |
| Triggering Event | A customer wants to share their experience by leaving a review | | |
| Brief Description | Customer submits a review, including ratings and optional comments, after dining at the restaurant | | |
| Actors | Customer | | |
| Related Use Cases | * Staff responds to a customer review | | |
| Preconditions | * The customer has dined at the restaurant * The restaurant app allows review submissions * The customer is logged in to the system | | |
| Post Conditions | * The review is successfully posted and visible to other users and staff | | |
| Flow of activities | Actor | | System |
|  | 1. | Opens the app and navigates to the review section | Displays past reviews and a "Write a Review" button |
|  | 2. | Selects a rating (1 to 5 star) | Captures the rating |
|  | 3. | Writes a comment about their experience | Stores the comment |
|  | 4. | Clicks "Submit" to post the review | Saves the review and updates the review section |
|  | 5. | Receives confirmation that the review has been posted | Displays the newly submitted review |
| Exception Conditions | * The customer submits a review without a rating * The system encounters an error while saving the review | | |

#### UC10: Staff responds to a customer review

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Name | Staff responds to a customer review | | |
| Triggering Event | A staff member wants to respond to a customer review | | |
| Brief Description | Staff replies to customer reviews to address concerns, express gratitude, or clarify information | | |
| Actors | Staff | | |
| Related Use Cases | * Customer adds a review | | |
| Preconditions | * The restaurant app has a review management system * The staff has permission to respond to reviews | | |
| Post Conditions | * The staff's response is posted under the customer's review | | |
| Flow of activities | Actor | | System |
|  | 1. | Logs into the app and navigates to the review section | Displays customer reviews |
|  | 2. | Selects a review to respond | Opens the selected review with a reply option |
|  | 3. | Writes a response to the review | Captures the response |
|  | 4. | Clicks "Submit" to post the reply | Saves and displays the response under the customer review |
|  | 5. | Customer is notified about the response | Sends a notification to the customer |
| Exception Conditions | * The staff member tries to reply without proper permissions * The system fails to save the response | | |

## **3.2 Use Case Specifications with corresponding interface mockups:**

**Each use case needs to have the following:**

1- **Business Rules.**

**2- System Use Case Diagrams.**

**3- Use Case Descriptions.**

**4- Corresponding Mockups**

# Domain Class Diagram

# 

# Database

## 5.1 Data Model (Conceptual Schema for NoSQL)

### Restaurant and Waitlist Model

Restaurants (Collection):

{

\_id: ObjectId,

name: String,

phone: String,

location: {

address: String,

city: String,

state: String,

zip: String

},

hours: {

open: String,

close: String

},

menu\_id: ObjectId,

rating: Number

}

Waitlist: {

\_id: ObjectId

customer\_id: ObjectId (Reference to Users)

party\_size: Number

status: String

estimated\_wait\_time: Number

joined\_time: Timestamp

seated\_time: Timestamp

removed\_time: Timestamp

note: String

}

}

### Menu Model

{

\_id: ObjectId,

restaurant\_id: ObjectId, // Reference to Restaurants

created\_at: Timestamp,

updated\_at: Timestamp,

menu\_items: [

{

\_id: ObjectId,

name: String,

description: String,

price: Number,

category: String, // e.g., Appetizer, Main, Dessert, Beverage

available: Boolean,

image\_url: String,

dietary\_info: {

vegetarian: Boolean,

vegan: Boolean,

gluten\_free: Boolean,

halal: Boolean

},

promotions: {

discount\_percentage: Number,

valid\_until: Timestamp

}

}

]

}

### Mini-Game Model

{

reviewID: ObjectId,

name: String // name of the game e.g., Spin wheel, Trivia Challenge,

description: String,

pointsEarned: Interger,

last\_updated: Timestamp

}

### Review Model

{

reviewID: ObjectId,

customer: Customer, //refer from Customer Model

restaurant: Restaurant, //refer from Restaurant Model

description: String,

price: double,

last\_updated: Timestamp

}

### Admin/Staff Model

{

\_id: ObjectId

restaurant\_id: ObjectId

name: String

email: String

password: String

role: String

created\_at: Timestamp

}

### QR Code Model

{

\_id: ObjectId

restaurant\_id: ObjectId

qr\_code\_url: String

created\_at: Timestamp

}

### Customer Model

Customers (Collection):

{

\_id: ObjectId,

name: String,

email: String,

phone\_number: String,

rewards\_points: Number,

waitlist\_history: [

{

restaurant\_id: ObjectId,

waitlist\_id: ObjectId,

joined\_at: Timestamp,

status: String // "Waiting", "Seated", "Cancelled"

}

],

created\_at: Timestamp

}

## 5.2 Data Sample (JSON Documents)

### Restaurant and Waitlist sample

{

"\_id": "65fa4bde1234567890abcd12",

"name": "Tasty Bites Restaurant",

"phone": "+1-416-555-1234",

"location": {

"address": "123 Main Street",

"city": "Toronto",

"state": "ON",

"zip": "M5H 2N2"

},

"hours": {

"open": "11:00 AM",

"close": "10:00 PM"

},

"menu\_id": "661fa0e21234567890abcd99",

"rating": 4.5

}

"waitlist": [

{

"\_id": "11,

"customer\_id": "1001",

"party\_size": 4,

"status": "queued",

"estimated\_wait\_time": 20,

"joined\_time": "2025-03-02T18:45:00Z",

"seated\_time": null,

"removed\_time": null,

"notes": "Requested window seating"

},

{

"\_id": "12",

"customer\_id": "1024",

"party\_size": 2,

"status": "seated",

"estimated\_wait\_time": 5,

"joined\_time": "2025-03-02T18:30:00Z",

"seated\_time": "2025-03-02T18:50:00Z”,

"removed\_time": null,

"notes": "Prefers outdoor seating"

}

]

}

### Menu sample

{

"\_id": "661fa0e21234567890abcd99",

"restaurant\_id": "65fa4bde1234567890abcd12",

"created\_at": "2025-03-20T18:00:00Z",

"updated\_at": "2025-03-21T14:00:00Z",

"menu\_items": [

{

"\_id": "item001",

"name": "Grilled Salmon",

"description": "Fresh Atlantic salmon grilled to perfection, served with lemon butter sauce.",

"price": 19.99,

"category": "Main Course",

"available": true,

"image\_url": "https://example.com/salmon.jpg",

"dietary\_info": {

"vegetarian": false,

"vegan": false,

"gluten\_free": true,

"halal": true

},

"promotions": {

"discount\_percentage": 15,

"valid\_until": "2025-03-30T23:59:59Z"

}

},

{

"\_id": "item002",

"name": "Vegan Brownie",

"description": "Delicious chocolate brownie made with vegan ingredients.",

"price": 6.50,

"category": "Dessert",

"available": true,

"image\_url": null,

"dietary\_info": {

"vegetarian": true,

"vegan": true,

"gluten\_free": false,

"halal": true

},

"promotions": {

"discount\_percentage": 0,

"valid\_until": null

}

}

]

}

### Mini-Game sample

### {

### "\_id": "game001",

### "name": "Spin the wheel",

### "description": "A fun mini-game where users spin a wheel to win rewards.",

### "pointsEarned": 50,

"last\_updated": "2025-03-15T12:00:00Z"

### }

### Review sample

{

"\_id": "review001",

"customer": {

"\_id": "1001",

"name": "John Doe",

"phoneNumber": "123-456-7890",

"email": "johndoe@example.com"

},

"restaurant": {

"\_id": "65fa4bde12345670abcd12",

"name": "The Food Spot",

"location": "123 Main St, Toronto, ON"

},

"description": "Amazing food and excellent service!",

"price": 25.99

"last\_updated": "2025-03-15T12:00:00Z"

}

### Staff/Admin sample

{

"\_id": "65f1a3eaf1b2c345670123",

"restaurant\_id": "65f1a3eaf1b2c345690124",

"name": "John Doe",

"email": "john.doe@example.com",

"password": "$2b$10$7Qh/N1QK7IhH1O1Y9iRgeOgP9jKyzTzQbY2R4G8OZB9h0J3y5ui",

"role": "admin",

"created\_at": "2025-03-19T12:00:00Z"

}

### QR Code sample

{

"\_id": "65f1a3eaf1b2c34567890125",

"restaurant\_id": "65f1a3eaf1b2c345678901",

"qr\_code\_url": "https://queneca.com/qr/65f1a3eaf1b2c345678901",

"created\_at": "2025-03-19T12:05:00Z"

}

### Customer sample

{

"\_id": "65f2b4cdf1b2c34567890125",

"name": "John Doe",

"email": "johndoe@example.com",

"phone\_number": "+1-555-123-4567",

"rewards\_points": 120,

"waitlist\_history": [

{

"restaurant\_id": "65f1a3eaf1b2c345678901",

"waitlist\_id": "65f3c5def1b2c345678902",

"joined\_at": "2025-03-19T12:10:00Z",

"status": "Seated"

},

{

"restaurant\_id": "65f4d6eff1b2c345678903",

"waitlist\_id": "65f5e7fff1b2c345678904",

"joined\_at": "2025-03-20T18:30:00Z",

"status": "Cancelled"

}

],

"created\_at": "2025-03-18T10:00:00Z"

}

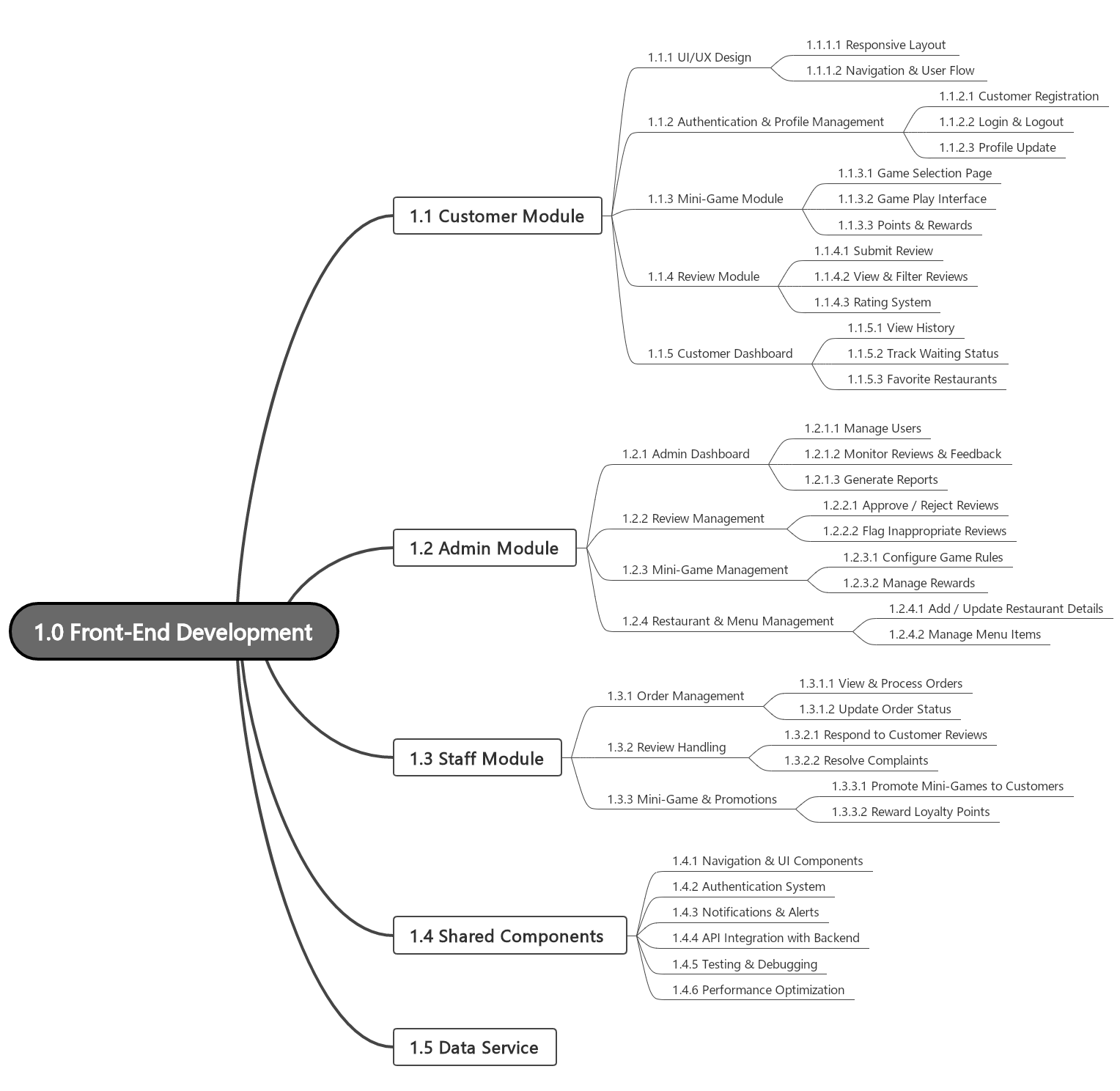
# Work Breakdown Structure (WBS)

Sample WBS:

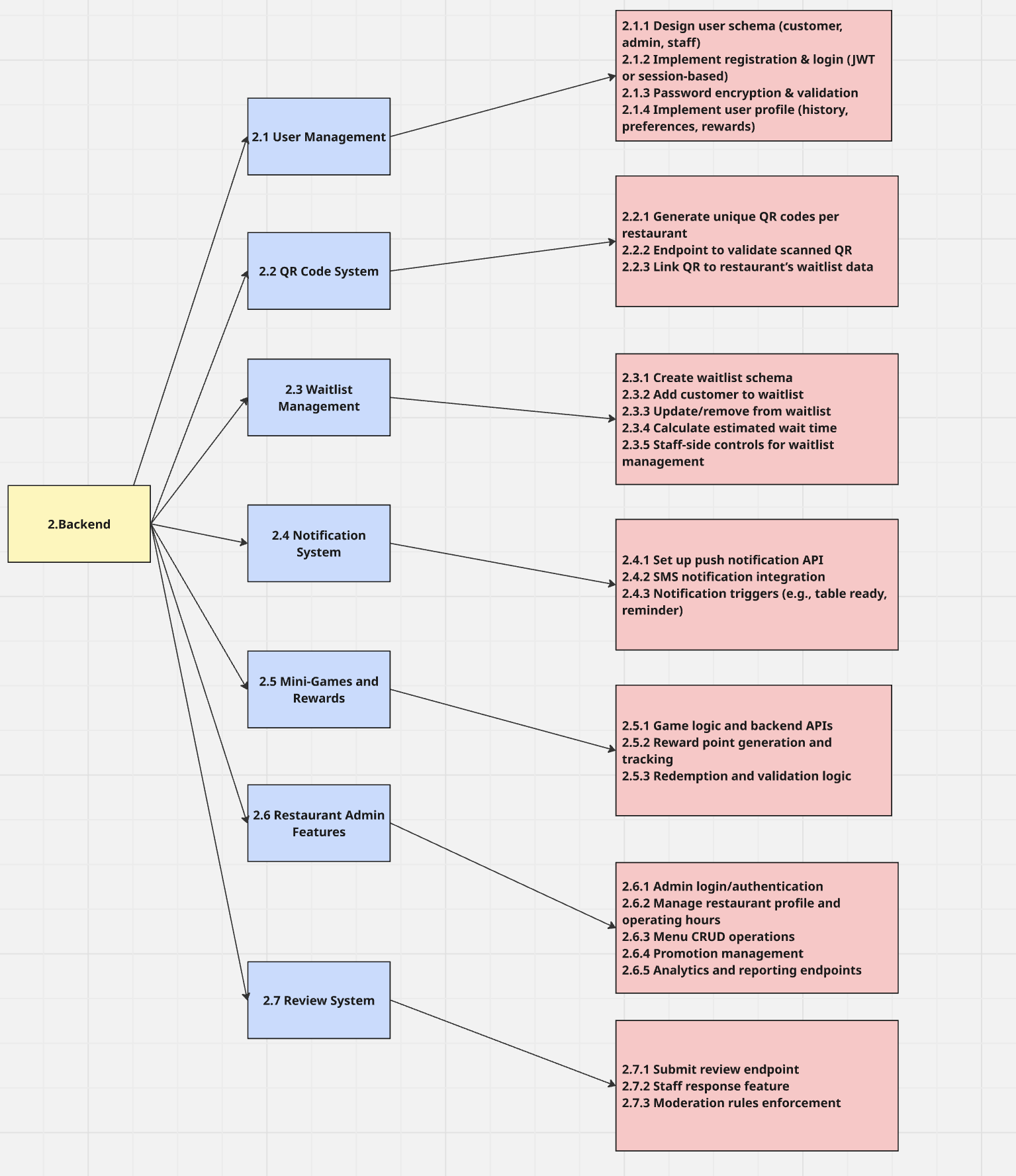
Diagram

Description automatically generated

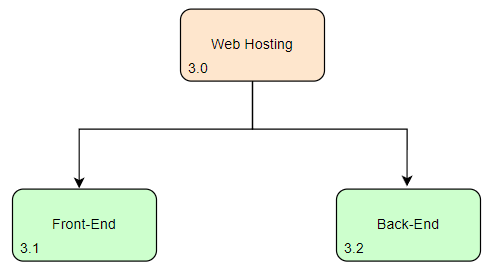
## Front-End



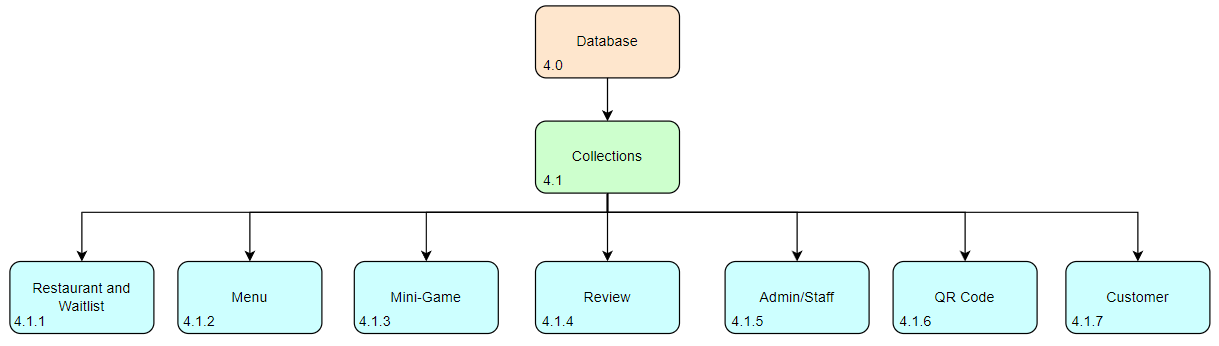
## Back-End



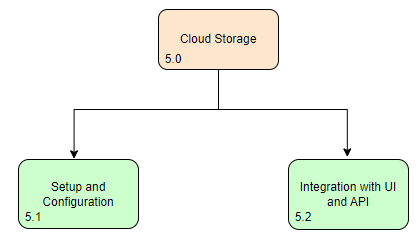
## Web Hosting



## Database



## Cloud Storage



# Milestones and Acceptance Criteria

## 7.1 Milestones

1. **Cloud storage is set up.** The database is created, and sample data is loaded. User accounts (restaurant staff and customers) can be created, logged in, and authorized. The web app can start with a basic login screen.
2. **Basic front-end layout is built, and UI components are standardized.** The navigation bar, color scheme, and fonts are finalized. Initial pages like the homepage and login screen are functional.
3. **Core user components are created** for different user roles (Administrator, Restaurant Staff, and Customers). Each role has access to its respective dashboard, and a professor can review these layouts.
4. **Navigation is completed, APIs are implemented, and the application is functional.** Users can register, join a waitlist, and receive updates. The QR code system is integrated, and restaurants can manage the waitlist.
5. **Testing phase begins.** API tests are created (if needed). Compare the app functionality with SRS (Software Requirement Specifications). Identify and fix bugs. Adjust based on professor feedback.

## 7.2 Acceptance Criteria

**Milestone 1 (Approximately 2 weeks)**

* Cloud storage is configured.
* Database is created, and sample data is added.
* Web app can connect to the database.
* User accounts can be created (Admin, Restaurant Staff, Customers).
* The web app can start, and the login system is functional.
* A professor can create an account and log in.

**Milestone 2 (Approximately 2 weeks)**

* Create the front-end header and footer layouts.
* Standardize fonts, colors, and graphics.
* Implement a basic restaurant dashboard.
* Administrator dashboards are created.
* Restaurant staff waitlist management screens are created.
* Restaurant management API is created.
* Waitlist management API is created.
* Discuss the application with the professor and present sample layouts.

**Milestone 3 (Approximately 4 weeks)**

* Begin creating user role components.
* Customer profiles and waitlist history screens are created.
* QR code scanning and waitlist entry are functional.
* User authentication API is created (login, signup).
* The professor can access layouts for all user roles.

**Milestone 4 (Approximately 4 weeks)**

* Notifications system is created.
* Review system is created.
* Users can navigate through the app and complete basic actions.
* Gamification and rewards UI is introduced.

**Milestone 5 (Approximately 2 weeks)**

* API tests are created (if necessary).
* Check for and fix major bugs.
* Ensure all main functionalities align with the SRS.
* Implement any feedback from the professor.
* Final deployment preparations.

# Implementation Schedule

Sprint 1 (Week 1-4):

Database Setup, Cloud Storage Setup, Admin & Staff core features implementation.

Sprint 2 (Week5-8):

Customer Core Features, QR Code System, Notification, Review and Mini-Game system.

Sprint 3 (Week 9-12):

Data persistence with Database, Web Hosting, Integration Testing, Final features adjustment and Deployment.

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor