Software Requirements Specification

for

ReserV8

Version 1.3

Prepared by

Group Name: ReserV8

Aashita Parimi	**SE22UCSE002**	**se22ucse002@mahindrauniversity.edu.in**
Steffy Ranjith	**SE22UCSE217**	**se22ucse217@mahindrauniversity.edu.in**
S.Meghana	**SE22UCSE228**	**se22ucse228@mahindrauniversity.edu.in**
Shriya Mamidela	**SE22UCSE250**	**se22ucse250@mahindrauniversity.edu.in**
Tanasi Singarapu	**SE22UCSE251**	**se22ucse251@mahindrauniversity.edu.in**
Tathireddy Meghana	**SE22UCSE268**	**se22ucse268@mahindrauniversity.edu.in**
Vedha Mudhana	**SE22UCSE286**	**se22ucse286@mahindrauniversity.edu.in**
Shreyas A Baksi	**SE22UCSE317**	**se22ucse317@mahindrauniversity.edu.in**

Instructor: Dr. Vijay Rao

Course: Software Engineering

Lab Section: IT Lab

Teaching Assistant: Prof. Sravanthi Acchugatla

Date: 10-03-2025

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	ReserV8	Initial Draft	15/02/25
1.1	ReserV8	Use Cases	02/03/25
1.2	ReserV8	Functional and Non-Functional Requirements	06/03/25
1.3	ReserV8	Final Draft	10/03/25

1 Introduction

The ReserV8 project is a dynamic, multi-purpose reservation system designed to streamline the process of booking tables at restaurants and reserving university facilities. The system provides an intuitive web-based interface for users to search for available options, make reservations, and pre-book meals or drinks, ensuring a smooth and efficient experience.

1.1 Document Purpose

This Software Requirements Specification (SRS) document defines the functional and non-functional requirements of ReserV8, a dynamic, multi-purpose reservation system. The document provides a structured framework for developers, testers, and stakeholders, detailing the system's objectives and expected functionalities. It covers restaurant table reservations, meal pre-booking, and university facility bookings.

1.2 Product Scope

The ReserV8 system is a web-based reservation platform designed to facilitate restaurant table booking and university facility reservations. The system is built using React.js for the front end, Node.js and Express.js for the back end, and PostgreSQL for database management. It integrates geolocation services via Google Maps API and supports secure transactions using payment gateway APIs.

The key benefits of ReserV8 include time efficiency and an intuitive user experience. By integrating geolocation services and providing a centralized booking system, ReserV8 enhances convenience for users while helping restaurants and university administrators manage their reservations more effectively. The system ensures secure transactions, optimized performance, and seamless accessibility across different devices.

1.3 Intended Audience and Document Overview

This document is intended for various stakeholders involved in the ReserV8 project, including:

- Developers: To understand the system architecture, functional requirements, and constraints for implementation.
- Project Managers: To track progress, ensure timely development, and align features with project objectives.
- Testers: To verify and validate the system based on defined requirements.
- Clients: To ensure the platform meets the needs of students, faculty, and restaurant owners.
- Professors: To assess the project’s adherence to software engineering principles.

This SRS document is structured as follows:

- Section 1: Introduction, including purpose, scope, audience, and key definitions.

- Section 2: Overall description of the system, its functionalities, constraints, and dependencies.
- Section 3: Detailed functional and non-functional requirements, including external interfaces.
- Section 4: Other requirements, such as performance and security considerations.
- Appendices: Data dictionary and group activity log.

For effective reading, clients and professors should begin with Section 1 and Section 2 to understand the system's purpose and capabilities. Developers and testers should focus on Sections 2 and 3 for detailed implementation and validation guidelines. Project managers should refer to the entire document to ensure completeness and alignment with project milestones.

1.4 Definitions, Acronyms and Abbreviations

- API -Application Programming Interface
- UI/UX -User Interface / User Experience
- QA- Quality Assurance
- SRS- Software Requirements Specification
- GDPR General Data Protection Regulation

1.5 Document Conventions

This document follows IEEE SRS formatting guidelines:

- Font: Arial, size 12
- Spacing: Single
- Margins: 1-inch
- Section Titles: Bold and numbered for clarity

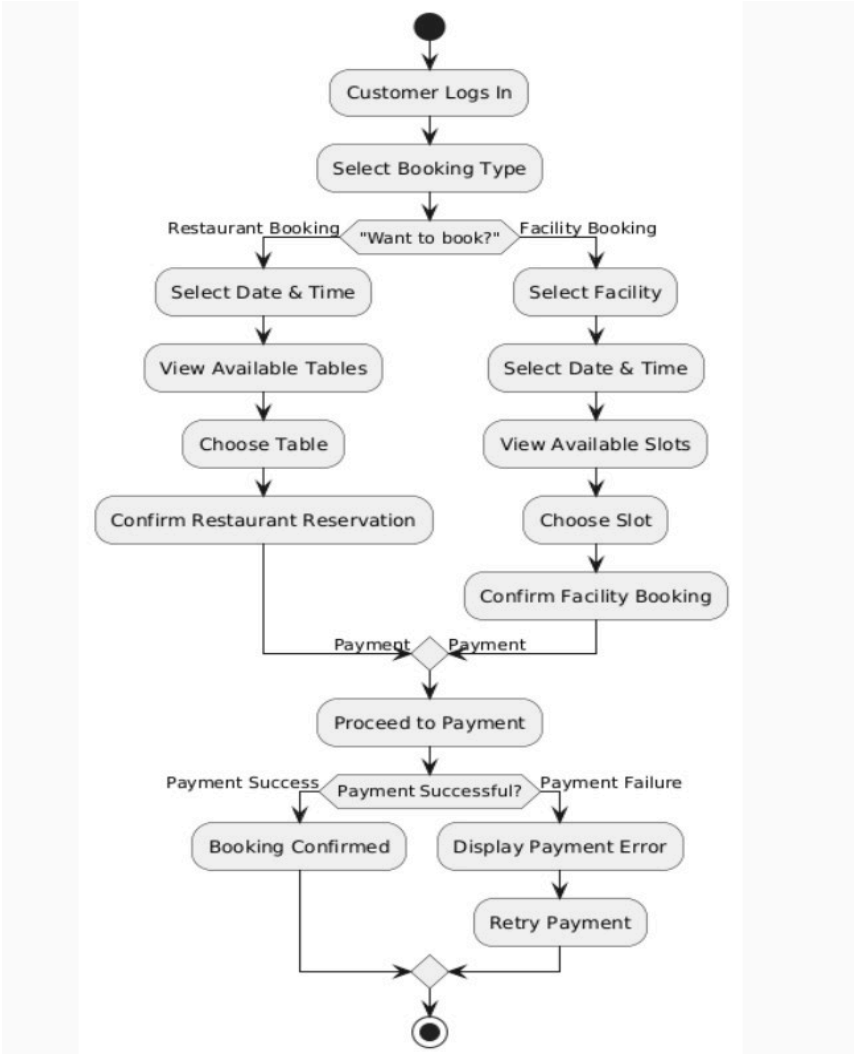
1.6 References and Acknowledgments

- ReserV8 Statement of Work (SOW)
- IEEE SRS Template
- Google Maps API Documentation
- Razor Pay API Documentation

2 Overall Description

2.1 Product Overview

ReserV8 is a web-based reservation system designed for restaurant and campus facility bookings. It
integrates geolocation services and ensures a seamless reservation process for users.



2.2 Product Functionality

- User Registration & Authentication
- Restaurant Table Reservations
- Meal & Drink Pre-Booking
- University Facility Booking
- Admin Management Portal

- Secure Payment Integration

2.3 Design and Implementation Constraints

The ReserV8 system development is subject to several constraints, which limit the options available to developers. These constraints ensure compatibility, maintainability, and performance optimization.

Hardware Limitations:

- The system is designed to run on web browsers and does not require specialized hardware.
- Performance may be affected on low-power devices with limited memory and processing power.

Software and Technology Constraints:

The system must be implemented using:

- Frontend: HTML, CSS, JavaScript (React.js for UI).
- Backend: Node.js with Express.js.
- Database: PostgreSQL for structured data storage.

Design and Modeling Constraints:

- COMET Methodology must be used for software design to ensure modularity and maintainability.
- UML (Unified Modeling Language) must be used for designing system architecture, including:
  - Use case diagrams
  - Class diagrams
  - Sequence diagrams

Communication Protocols:

- The system follows the RESTful API architecture for efficient data exchange between frontend and backend.
- Secure connections must be established using HTTPS and TLS encryption for data protection.

Security Considerations:

- User authentication is mandatory for all reservations.
- Role-based access control (RBAC) is enforced for admins managing reservations.
- Sensitive data must be stored securely with encryption.

Parallel Operations and Scalability:

- The system should handle at least 100 concurrent users without performance degradation.
- Load balancing mechanisms may be required for scalability in future iterations.

2.4 Assumptions and Dependencies

Stable Third-Party Services: It is assumed that external services like Firebase Auth, Google Maps API, and payment gateways (Razorpay/Stripe) will remain stable and available throughout the project's lifecycle. Any downtime or API changes could affect system functionality.
Scalability of Backend & Database: The PostgreSQL database and Node.js backend (Express.js) are assumed to handle expected traffic loads. If user demand exceeds initial estimates, performance optimization may be required.
Cross-Browser Compatibility: It is assumed that the frontend (React.js/Next.js with Tailwind CSS) will function correctly across major web browsers. Unexpected rendering issues might require additional fixes.
User Device & Network Conditions: It is assumed that users will access the system from modern devices with stable internet connections. Poor network conditions or outdated browsers may cause unexpected usability issues.
Integration with Future Features: The system is designed to be extendable. It is assumed that adding new features will not require major architectural changes.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The web-based application will have a modern, responsive UI designed for an intuitive user experience. The primary user interface components include:

- Homepage: Users can search for nearby restaurants using geolocation or by manually entering a location.
- Restaurant Listings: Displays a list of available restaurants with details such as name, rating, cuisine type, and distance.
- Reservation System: Users can select a restaurant, view available tables, pick a time slot, and confirm their booking.
- User Authentication: A login/signup system using Firebase Authentication, allowing users to manage their reservations.
- Payment System: If required, users can prepay for cover charges via Razorpay/Stripe.

User Interaction:

- Users interact with the website via mouse clicks and touch gestures (for mobile users).
- The UI will feature drop-down menus, input fields, and interactive buttons for navigation..

3.1.2 Hardware Interfaces

The system will interact with several hardware components to provide seamless functionality:

Supported Devices and Interfaces:

1. User Devices:
    - The website will be accessible on desktops, tablets, and mobile devices.
    - Optimized for modern browsers like Chrome, Firefox, Safari, and Edge.
2. Geolocation API:
    - Utilizes the Google Maps API to detect user location and fetch nearby restaurants.
3. Database Server:
    - PostgreSQL database for storing user data, restaurant details, and reservations.
4. Authentication System:
    - Firebase Auth will handle login and user verification.
5. Payment Gateway:
    - Razorpay/Stripe API will handle secure online transactions.

Each component will be integrated to provide a seamless end-to-end experience, ensuring that users can browse, book, and pay effortlessly.

3.1.3 Software Interfaces

The ReserV8 system interacts with various external software components to provide core functionalities. The key software interfaces include:

1. Frontend-Backend Communication:
    - The web application (React.js/Next.js) communicates with the backend (Node.js/Express.js) through RESTful API endpoints.
    - Users send requests (e.g., searching for restaurants, making reservations), and the backend processes and responds accordingly.

2. Database Interface:
    - The backend interacts with a PostgreSQL database to store and retrieve user accounts, restaurant details, and reservations.
    - Queries are executed using SQL to ensure efficient data management.
3. Authentication System:
    - The system uses Firebase Authentication to handle user login and registration securely.
    - The frontend interacts with Firebase via API calls to verify credentials and manage user sessions.
4. Geolocation API:
    - Google Maps API is used to fetch nearby restaurants based on the user's location.
    - The frontend sends location data to the API, and it responds with relevant restaurant options.
5. Payment Gateway Interface:
    - The system integrates with Razorpay or Stripe to handle online payments for cover charges.
    - The backend securely interacts with the payment gateway to process transactions and update reservation statuses.

3.2 Functional Requirements

3.2.1 F1: User Registration & Authentication

- The system shall allow users to register an account using email and password.
- The system shall support Firebase Authentication for secure login.
- The system shall allow users to reset their password via email.
- The system shall restrict access to authenticated users only for making reservations.

3.2.2 F2: Restaurant Table Reservations

- The system shall allow users to search for restaurants within a specific radius using Google Maps API.
- The system shall display restaurant details, including name, cuisine type, availability, and rating.
- The system shall allow users to select a restaurant and book a table for a preferred date and time.
- The system shall store reservation details in the PostgreSQL database.

3.2.3 F3: Meal & Drink Pre-Booking

- The system shall allow users to pre-select meals and drinks when making a reservation.
- The system shall display menu options provided by restaurants.
- The system shall store meal and drink selections in the reservation details.
- The system shall impose a cover charge for pre booking of meals.

3.2.4 F4: University Facility Booking

- The system shall allow students to book university facilities such as study rooms and sports grounds.
- The system shall check availability before confirming a booking.
- The system shall store facility reservations in the database.

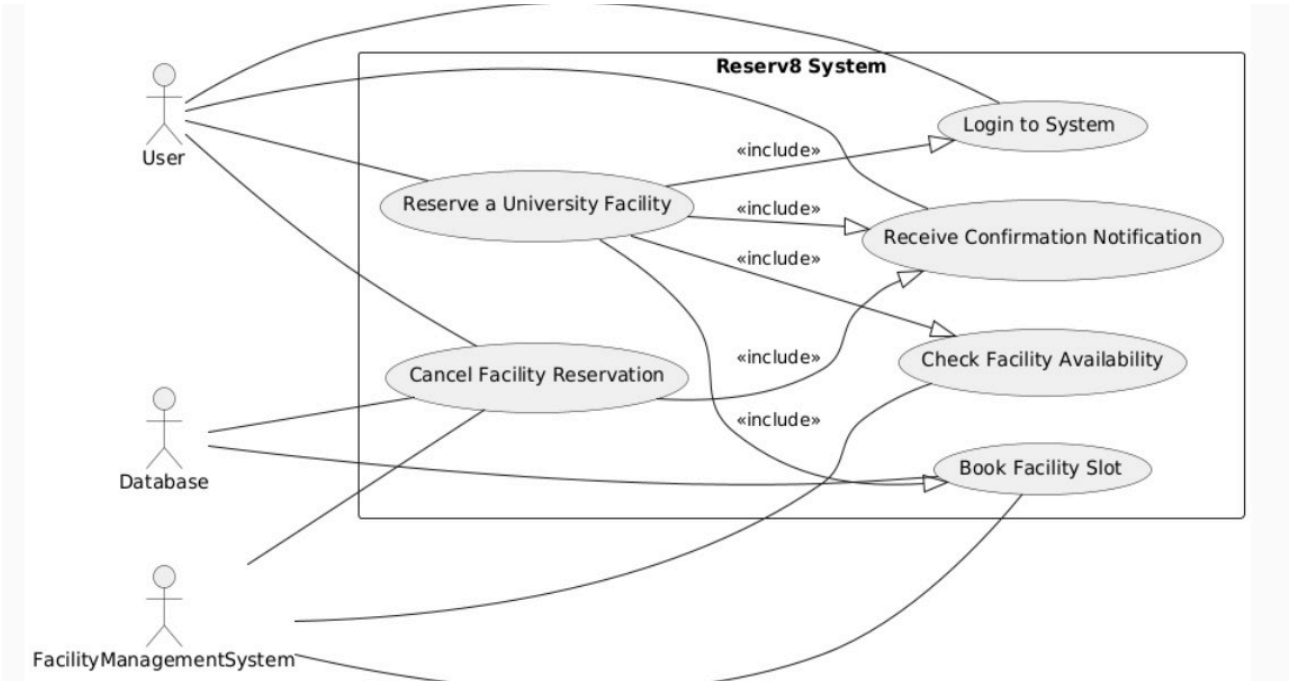
3.2.5 F5: Admin Management Portal

- The system shall provide an admin dashboard for restaurant owners and university staff.
- The admin portal shall allow managing restaurant availability and updating menus.
- The admin portal shall allow managing facility booking slots for university administrators.

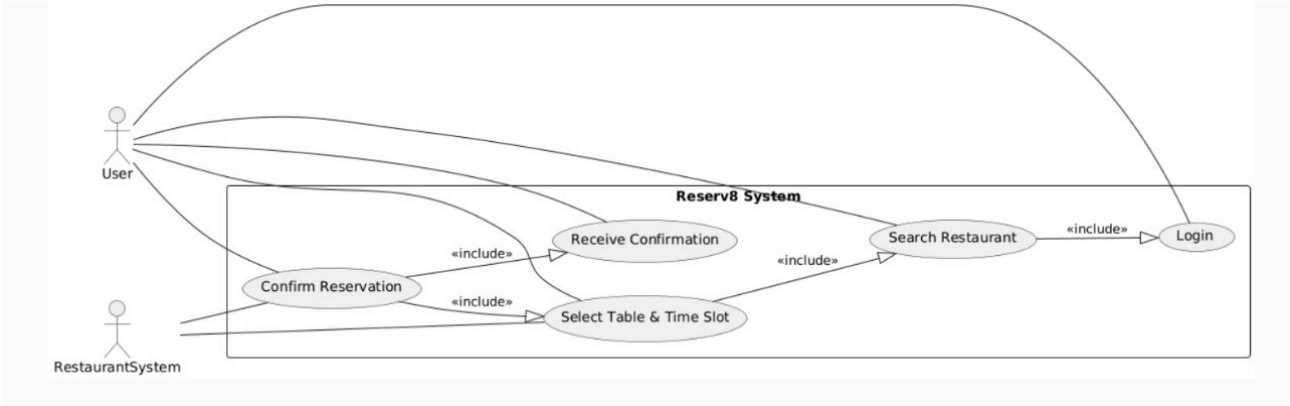
3.2.6 F6: Secure Payment Integration

- The system shall integrate Razorpay/Stripe for processing payments.
- The system will require users to pay a cover charge only if they are pre-booking a meal.
- The system shall securely store transaction details for reference.

3.3 Use Case Model



3.3.1 Use Case #1: Reserve a Restaurant Table (U1)



Author: Shreyas Aditya Baksi

Purpose:
The objective of this use case is to allow users to search for and reserve a table at a restaurant within their preferred radius using the web-based application.

Requirements Traceability:
- The system must allow users to search for restaurants based on location.
- The system must display available tables and allow users to select a preferred time slot.
- The system must confirm the reservation and notify the user.

Priority: High

Preconditions:
- The user must be logged into the system.
- The restaurant database must be up-to-date with availability.

Postconditions:
- The system successfully books a table for the user.
- The user receives a confirmation message (email/SMS).

Actors:
- User (Customer making a reservation)
- Reservation System (Handles search, booking, and notifications)
- Geolocation API (Provides nearby restaurant options)
- Database (Stores user and restaurant information)

Extends: None

Flow of Events
1. Basic Flow:
1. The user selects the "Reserve a Table" option.
2. The system prompts the user to enter a location or enable geolocation.
3. The system fetches nearby restaurants using the Geolocation API.
4. The user selects a restaurant from the list.
5. The system displays available tables with time slots.
6. The user selects a table and a time slot.
7. The system saves the reservation in the Database.
8. The system sends a confirmation notification to the user.

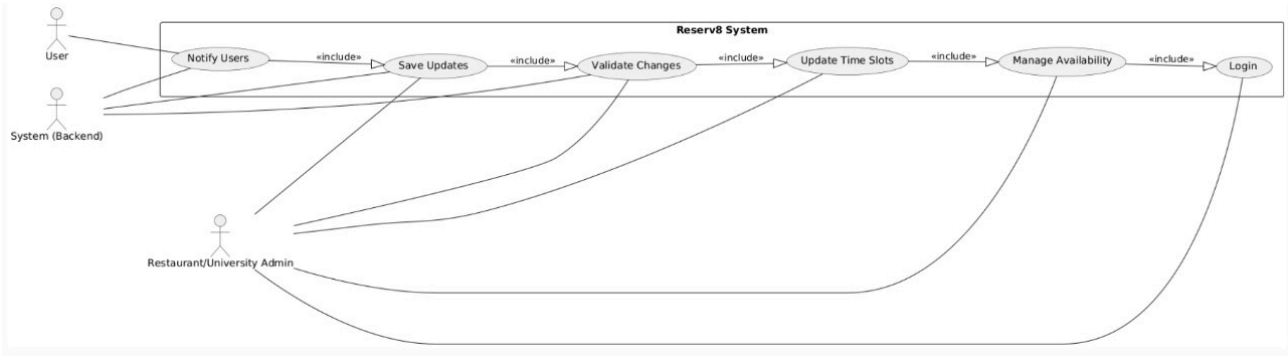
2. Alternative Flow:
User Cancels Reservation
1. The user cancels the reservation before the scheduled time.
2. The system removes the reservation from the database.
3. The system notifies the restaurant of the cancellation.

3. Exceptions:
- The system informs the user that no restaurants are available in the selected area.
- The system notifies the user that no tables are available for the selected time slot.
- If an error occurs during booking, the system notifies the user and prompts them to try again.

Includes:
- U2: Payment Processing
- U3: User Authentication

Notes/Issues:
- Need to confirm if some restaurants require pre-payment for reservations.
- Ensure the system updates table availability in real-time to avoid overbooking.

3.3.2 Use Case #2: Manage Restaurant Availability (U2)



Author: S.Meghana

Purpose:
The objective of this use case is to allow restaurants/university admins to update table/facility availability, modify reservation slots, and block out unavailable times. This ensures accurate booking availability and prevents overbooking.

Requirements Traceability:
- The restaurant/university admin must be able to update available time slots.
- The system must prevent overbooking.
- Users must be notified of any changes affecting their reservations.

Priority: High

Preconditions:
- The restaurant/university admin must be logged into the system.
- The restaurant/university admin must be registered in the system.

Postconditions:
- Availability data is updated in the system.
- Reservations are adjusted if necessary.
- Users with affected reservations receive notifications.

Actors:
-Restaurant/University Admin – Updates availability.
-System (Backend)– Saves changes and validates bookings.
-Users– Get notified of changes.

Extends: None

Flow of Events:
Basic Flow:
1. The restaurant/university admin logs into the system.
2. It navigates to the availability management section.
3. They update available time slots or blocks out specific dates/times.
4. The system validates changes to prevent conflicts.
5. The updated availability is saved.
6. Users with affected reservations are notified.

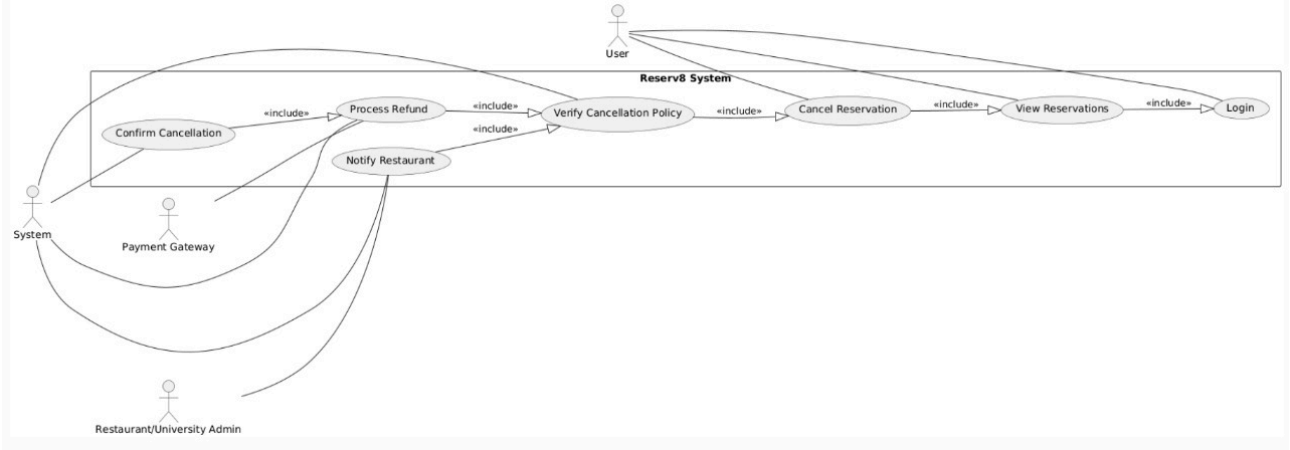
Alternative Flow:
• If a time slot is already booked, the system prompts to confirm adjustments and notify affected customers.

Exceptions:
• If the restaurant/university admin tries to delete an already booked slot, the system prevents the action or suggests an alternative solution.
• If the internet connection fails, changes are not saved, and an error message is displayed.

Includes:
-U1 Reserve a Restaurant Table

Notes/Issues:
• Provide an option to temporarily disable reservations for maintenance or special events.

3.3.3 Use Case #3: Cancel a Reservation (U3)



Author: Aashita Parimi

Purpose:
The objective of this use case is to allow users to cancel an existing reservation and notify the restaurant/university admin. Refunds are only applicable if the user had pre-booked a meal and paid the cover charge.

Requirements Traceability:

- Users must be able to cancel their reservations.
- The restaurant/university admin must be notified of the cancellation.
- If the reservation included a pre-booked meal and a cover charge was paid, the system must process a refund if applicable.

Priority: High

Preconditions:

- The user must be logged into the system.
- The reservation must exist in the system.
- The cancellation must be within the allowed time frame for a refund (if applicable).
- A refund is only applicable if the user pre-booked a meal and paid a cover charge.

Postconditions:

- The reservation is removed from the system.
- The restaurant/university admin is notified of the cancellation.
- If a cover charge was paid for a pre-booked meal, a refund is processed if applicable.
- The user receives a cancellation confirmation.

Actors:

- User – Initiates the cancellation.
- System – Processes cancellation and updates availability.
- Restaurant/University Admin – Receives cancellation notification.
- Payment Gateway – Processes refunds if a pre-booked meal was canceled.

Extends: None

Flow of Events

1. Basic Flow:

1. The user logs into the system.
2. The user navigates to their reservations.
3. The user selects a reservation to cancel.
4. The system verifies the cancellation policy and processes the cancellation.
5. The system updates availability and notifies the restaurant/university admin.
6. If a pre-booked meal was included, the system processes a refund through the payment gateway (if applicable).
7. The user receives a cancellation confirmation.

2. Alternative Flow:

- If the cancellation window has expired, the system informs the user and prevents cancellation.
- If no meal was pre-booked, no refund is processed.

3. Exceptions:

- If the internet connection is lost, the cancellation is not processed.
- If the payment gateway fails, the refund is not issued, and the user is notified.

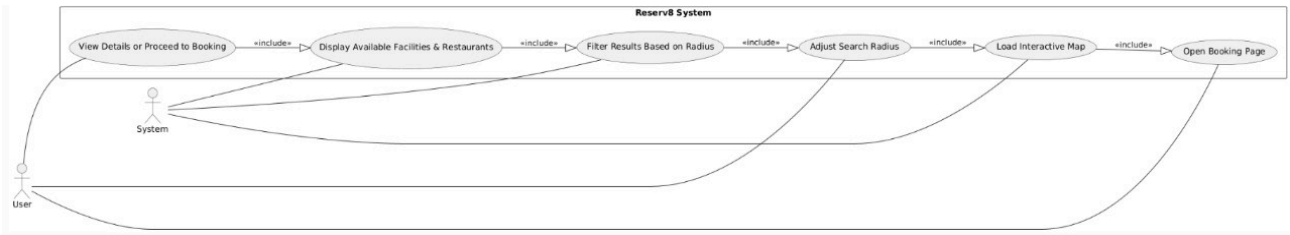
Includes:

- U1: Reserve a Restaurant Table
- U2: Manage Restaurant Availability

Notes/Issues:

- Ensure refunds are only applicable for pre-booked meals and not for general reservations.
- Consider allowing partial refunds based on restaurant policies.
- Ensure updates prevent overbooking due to canceled reservations.

3.3.4 Use Case #4: Selecting a Search Radius for Facility and Restaurant Booking (U4)



Author: Tanasi Singarapu

Purpose:
The objective of this use case is to allow users to dynamically adjust a circular radius on an interactive map and receive recommendations for restaurants and university facilities (e.g., sports complexes, study rooms, auditoriums) within the selected area.

Requirements Traceability:

- Users must be able to adjust a search radius dynamically on the map.
- The system must filter and display restaurants and university facilities within the selected radius.
- The system must allow manual location entry if location permissions are denied.

Priority: High

Preconditions:

- The user must be logged into the system (or have location permissions enabled for guest access).
- The application must have access to the user's location (if location-based search is enabled).
- The map must be loaded and interactive.
- University facility data must be available in the system.

Postconditions:

- The system filters and displays available restaurants and university facilities within the selected area.
- The user can view details and proceed with a booking if required.

Actors:

- User – Student, faculty, or visitor searching for facilities or restaurants.
- System – Application backend processing location-based recommendations.

Extends: None

Flow of Events

1. Basic Flow:

1. The user opens the booking page.
2. The system loads an interactive map centered around the user's current or selected location.
3. The user sees a circular selection tool overlaid on the map.
4. The user adjusts the radius of the circle by dragging the boundary or using a slider.
5. The system dynamically updates the area based on the new radius.
6. The user selects whether they are searching for restaurants, university facilities, or both.
7. The system fetches relevant data within the selected area and displays the results.
8. The user views the listings and can interact with them (e.g., view details, check availability, make a booking).

2. Alternative Flow:

- User Denies Location Permission
    - If the user denies location access, the system prompts them to enter a location manually.
    - The user enters a university block, hostel, or city name, and the system centers the map at that location.
    - The user can now adjust the radius as usual.
- No Facilities or Restaurants Found in Selected Radius

    - If no results are found, the system displays a message: "No restaurants or facilities are found in this area. Try expanding your search radius."
    - The user can increase the radius and retry.
- User Cancels the Search
    - The user decides not to select a radius and exits the search screen.
    - The system discards any selected radius and returns to the main screen.

3. Exceptions:

- Map Loading Failure
    - If the map fails to load, the system displays an error and retries loading.
- Server Error in Fetching Data
    - If the backend fails, the system shows an error message and prompts the user to retry later.

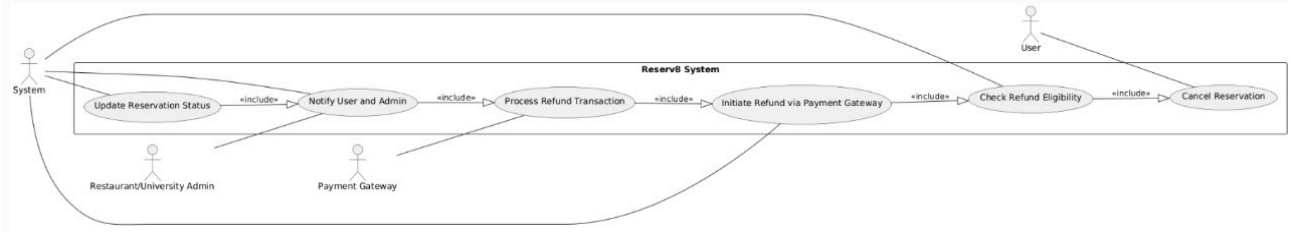
Includes:

- U1: Reserve a Restaurant Table
- U2: Manage Restaurant Availability

Notes/Issues:

- Ensure the system has up-to-date university facility data.
- Verify that the map library supports interactive radius selection.
- Optimize the backend for efficient location-based filtering.

3.3.5 Use Case #5: Process Payment for Pre-Booked Meals (U5)



Author: Steffy Ranjith

Purpose:
The objective of this use case is to process online payments when a user pre-books a meal as part of their restaurant reservation. The system must ensure secure payment handling through the payment gateway before confirming the booking. Additionally, the system must process refunds when a user cancels a pre-booked meal within the allowed refund period.

Requirements Traceability:

- The system must require payment only when the user pre-books a meal.
- The system must securely process payments through the payment gateway.
- The system must confirm the transaction and update the reservation status.
- The system must process refunds for canceled pre-booked meals within the allowed refund period.

Priority: High

Preconditions:

- The user must be logged in.
- The user must have selected a meal pre-booking option.
- The payment gateway must be operational.
- For refunds, the cancellation must occur within the refund eligibility period.

Postconditions:

- The payment is processed successfully.
- The reservation status is updated to "Paid."
- The user receives a confirmation of successful payment.
- The restaurant is notified of the pre-booked meal.
- If a cancellation occurs within the refund period, the refund is processed successfully.

Actors:

- User – Initiates the payment and/or refund request.
- System – Validates and processes the payment and refund.
- Payment Gateway – Handles the payment and refund transactions.
- Restaurant Admin – Receives payment and refund confirmations.

Extends: None

Flow of Events

1. Basic Flow:

1. The user selects a restaurant and pre-books a meal.
2. The system prompts the user to complete the payment.
3. The user enters payment details and confirms the transaction.
4. The system sends the payment request to the payment gateway.
5. The payment gateway processes the transaction and returns a confirmation.
6. The system updates the reservation status to "Paid."
7. The restaurant is notified of the confirmed pre-booked meal.
8. The user receives a confirmation message.

2. Alternative Flow:

- If the user cancels the payment before completing the transaction, the system discards the meal pre-booking and does not confirm the reservation.
- If the user cancels a pre-booked meal within the refund eligibility period, the system processes a refund through the payment gateway.

3. Exceptions:

- Payment Declined
  - The system notifies the user and prompts them to retry with another payment method.
- Payment Gateway Failure
  - The system informs the user and suggests trying again later.
- Cancellation Outside Refund Period
  - The system denies the refund and notifies the user of the policy.

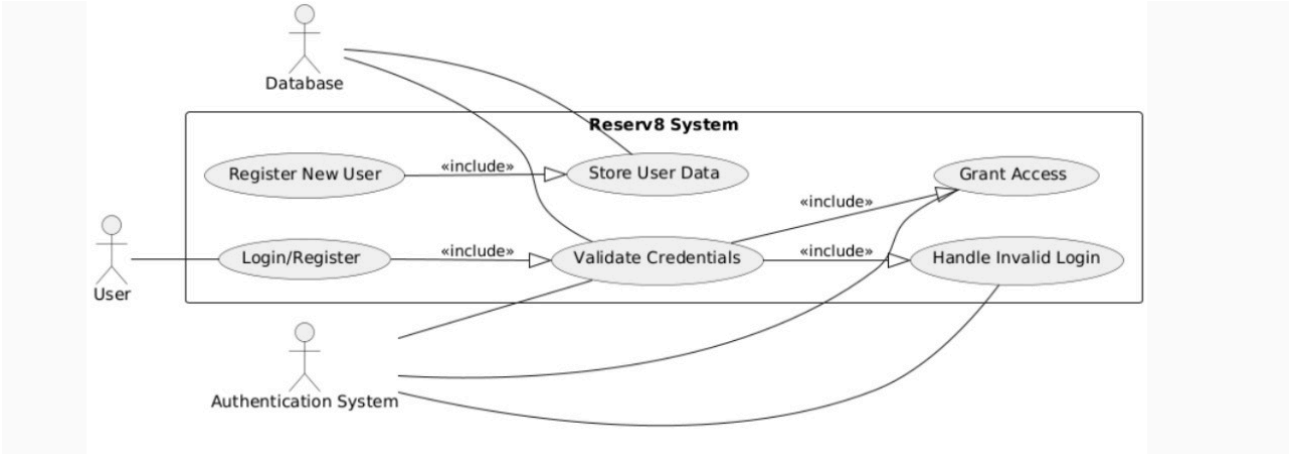
Includes:

- U1: Reserve a Restaurant Table
- U2: Manage Restaurant Availability
- U3: Cancel a Reservation

Notes/Issues:

- Ensure payment processing complies with financial security standards.
- Verify that pre-booked meals are correctly recorded in the reservation details.
- Ensure users cannot proceed with meal pre-booking unless payment is completed.
- Clearly communicate refund eligibility policies to users.

3.3.6 Use Case #6: User Authentication (U6)



Author: Vedha

Purpose:
The objective of this use case is to ensure users are authenticated before accessing the system's features, such as making reservations or payments.

Requirements Traceability:
- The system must allow users to register and log in securely.
- The system must enforce authentication before accessing certain features.
- The system must support third-party authentication.

Priority: High

Preconditions:
- The user must have a valid email/password or a third-party authentication account.

Postconditions:
- The user successfully logs in and gains access to the system.

Actors:
- User (Customer logging in)
- Authentication System (Validates login credentials)
- Database (Stores user credentials)

Extends: None

Flow of Events
1. Basic Flow:
   1. The user selects the "Login/Register" option.
   2. The system prompts for email/password or third-party authentication.
   3. The system validates the credentials.
   4. If valid, the user gains access to their account.

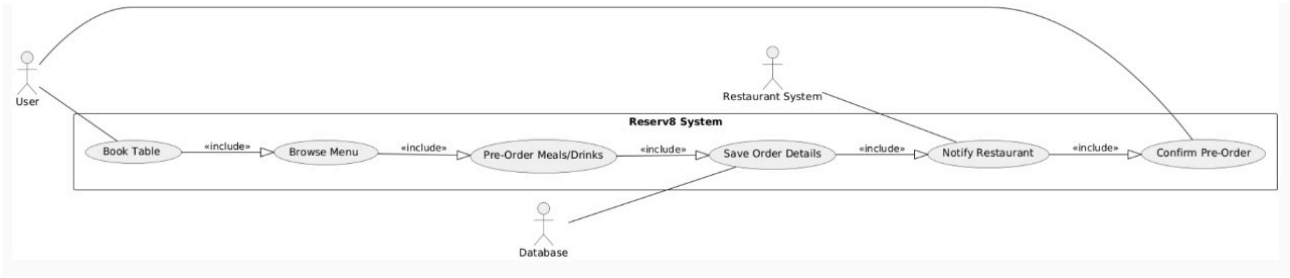
2. Alternative Flow:
User Chooses to Register
   1. The user selects "Register".
   2. The system prompts for email, password, and other details.
   3. The system creates a new user account.

3. Exceptions:
- The system notifies the user that login details are incorrect.
- The system suggests creating a new account.

Includes: None

Notes/Issues:
- Consider implementing multi-factor authentication for additional security.

3.3.7 Use Case #7: Pre-Book Meals and Drinks (U7)



Author: Shriya

Purpose:
This use case allows users to pre-order their meals and drinks while reserving a restaurant table, reducing wait time upon arrival.

Requirements Traceability:
- The system must allow users to browse restaurant menus.
- Users should be able to select and pre-order meals and drinks.
- The system should send the order details to the restaurant.

Priority: High

Preconditions:
- The user must have a confirmed table reservation.
- The restaurant must have an updated menu in the system.

Postconditions:
- The restaurant receives the pre-order.
- The user receives confirmation of the pre-booked items.

Actors:
- User (Customer pre-ordering meals)
- Restaurant System (Handles orders and menu management)
- Database (Stores order and menu details)

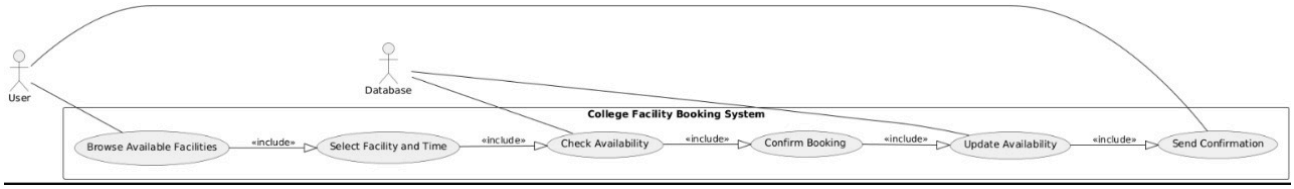
Flow of Events:
1. The user selects a restaurant and books a table.
2. The system prompts the user to browse the restaurant's menu.
3. The user selects meals and drinks to pre-order.
4. The system saves the order and links it to the reservation.
5. The system notifies the restaurant of the pre-order.

6 The user receives a confirmation message.

Exceptions:

- If the restaurant does not support pre-booking, the system disables the feature.
- If an item becomes unavailable, the system notifies the user.

3.3.8 Use Case #8: Reserving College Facilities (U8)



Author: Tathireddy Meghana Reddy

Purpose:
This use case allows users to reserve college facilities, such as sports grounds and library study rooms, ensuring availability at their preferred time.

Requirements Traceability:

- The system must allow users to browse available college facilities.
- Users should be able to select and book a facility for a specific date and time.
- The system should update facility availability upon successful booking.

Priority: High

Preconditions:

- The user is logged into the application.
- The system has up-to-date availability data for university facilities.
- The facility booking feature is enabled for the university.

Postconditions:

- The user successfully reserves a facility at the selected time.
- The system updates availability to reflect the reservation.
- The user receives a confirmation of the booking.

Actors:

- User – Student, faculty, or visitor reserving a facility.
- System – Application backend processing facility availability and reservations.
- Database – Stores facility availability and booking details.

Extends: None

Flow of Events:

1. Basic Flow:

1. The user navigates to the college facility booking section.
2. The system displays a list of available facilities.
3. The user selects a facility and chooses a date and time.
4. The system checks availability and confirms the selection.
5. The user proceeds to confirm the booking.
6. The system updates the facility's availability and sends a confirmation to the user.

2. Alternative Flow:

- Facility Not Available
    - If the selected facility is unavailable, the system suggests alternative times or nearby available facilities.
- User Cancels Reservation
    - If the user cancels before confirming, the system discards the selection and returns to the facility list.

3. Exceptions:

    - If the user remains inactive for a set period, the system cancels the reservation attempt and redirects them to the home screen.
    - If the backend fails, the system displays an error message and prompts the user to try again later.
    - If the user loses internet connection, the system notifies them and retries the booking once the connection is restored.

Includes:

- U1: Reserve a Restaurant Table
- U2: Manage Restaurant Availability
- U3: Cancel a Reservation
- U4: Selecting a Search Radius for Facility and Restaurant Booking

Notes/Issues:

- Verify that facility bookings cannot be double-booked due to system delays.

4 Other Non-functional Requirements

4.1 Performance Requirements

The following performance requirements ensure that the Dynamic, Multi-Purpose Reservation System operates efficiently under various circumstances, providing users with a seamless booking experience while maintaining system reliability.

1. General System Performance:

P1. The system must process a restaurant or facility reservation within 3 seconds after the user submits a request.
P2. The system must support at least 50 simultaneous reservations per minute without degradation in performance.
P3. The system must display search results (restaurants or facilities) within 2 seconds after the user sets their location and radius.
P4. The system must update table/facility availability in real-time to prevent overbooking or duplicate reservations.

2. Geolocation and Search Performance:

P5. The system must fetch and display nearby restaurants/facilities within 2 seconds after the user provides a location or enables geolocation.
P6. The system must allow users to dynamically adjust the search radius with results updating in under 1 second.
P7. The map feature must render and update location-based results within 1.5 seconds after the user changes their search radius.

3. Pre-Booking Meals and Orders:

P8. When a user pre-orders meals or drinks, the order confirmation must be processed within 3 seconds.
P9. The system must notify the restaurant of a pre-order within 5 seconds after confirmation.

4. Cover Charge Payment & Refund Processing:

4.4.1 Cover Charge Payment:

P10. When a customer opts for pre-booking meals, the cover charge payment must be processed within 5 seconds.
P11. The system must generate and send a confirmation receipt for the cover charge payment within 3 seconds after successful processing.
P12. If the payment fails, the user must receive an error message within 2 seconds with an option to retry.

4.4.2 Cover Charge Refund Process:

P13. If the customer attends their reservation, the cover charge must be refunded within 5 seconds after check-in confirmation.
P14. If the customer cancels their reservation, the system must determine the refund percentage within 2 seconds based on the meal preparation status.
P15. If the meal was not prepared, the system must refund 100% of the cover charge within 5 seconds.
P16. If the meal was partially prepared, the system must calculate and process the partial refund within 5 seconds.
P17. If the meal was fully prepared, the system must notify the user within 2 seconds that no refund is available.

4.4.3 Refund Notification and Processing(premium and high profile restaurants)

P18. The system must notify the restaurant of the cancellation and refund status within 3 seconds after processing the request.
P19. The user must receive a confirmation of their refund amount within 2 seconds after the cancellation is processed.
P20. If the refund fails due to a payment gateway issue, the system must notify the user within 2 seconds and retry automatically within 30 seconds.

5. Reservation Management and Admin Dashboard:

P21. Restaurant and university admins must be able to update availability, and changes should reflect in under 2 seconds.
P22. If a reservation is canceled, the slot should be made available to other users within 3 seconds.
P23. The admin dashboard should load all reservation and availability data within 2 seconds of login.

6. Payment Processing and Refunds:

P24. The payment gateway must complete a transaction within 5 seconds when a user makes a reservation.
P25. If a refund is issued, the system must process and initiate the refund request within 3 seconds.
P26. If a payment processing failure occurs, the user must be notified within 2 seconds with an appropriate error message.

7. Scalability and Load Handling:

P27. The system must be able to handle at least 100 concurrent users during peak hours without affecting response times.
P28. The database must support at least 1000 active reservations at any given time.

P29. The system must handle at least 1,000 location-based searches per minute without significant delays.

8. Security and Authentication:

P30. User authentication must be completed within 2 seconds after login credentials are submitted.
P31. If a login attempt fails, the user must receive feedback within 1 second.
P32. Session expiration should occur after 15 minutes of inactivity to maintain security

4.2 Safety and Security Requirements

4.2.1 Protecting User Information

ReserV8 will follow strict data privacy policies to ensure user details, such as names, contact information, and reservation history, remain secure. Only authorized users will have access to their own information. For students booking library study rooms, identity verification will be required to prevent unauthorized bookings by non-campus individuals.

4.2.2 Secure Login and Access

Users must log in to their accounts before making a reservation. To prevent unauthorized access, ReserV8 will support email or phone verification during account creation. If a user remains inactive for a certain period, they will be automatically logged out to protect their account. Additionally, restaurant/university administrators will have restricted access to manage their own reservations without accessing unrelated data.

4.2.3 Safe Payments and Fraud Prevention

For prepaid meal bookings, secure payment methods will be used to protect financial details. Users may need to confirm their reservations to prevent fake or last-minute cancellations. The system will monitor suspicious activities, such as multiple no-showsor frequent last-minute cancellations, and take action (e.g., temporary booking restrictions) to prevent misuse.

4.3 Software Quality Attributes

This section outlines the essential software quality attributes for the ReserV8 Dynamic, Multi-Purpose Reservation System, ensuring efficiency, usability, and reliability.

4.3.1 Reliability

The system should function consistently and provide uninterrupted reservation services.

Requirement: The system shall ensure minimal downtime, with planned maintenance occurring during off-peak hours.
Implementation Strategy:

Use automated backups to prevent data loss.
Conduct regular system testing to identify and fix issues proactively.

4.3.2 Usability

The platform must be user-friendly and intuitive for both students and restaurant customers.

Requirement: Users should be able to complete a reservation easily with minimal steps.
Implementation Strategy:
Use a simple and clean interface for easy navigation.
Ensure the platform is mobile-friendly for accessibility on different devices.

4.3.3 Maintainability

The system should allow for future updates and improvements with minimal effort.

Requirement: The system shall be structured for easy updates without disrupting existing functionality.
Implementation Strategy:
Follow clear coding standards and maintain documentation.
Use modular development to separate key features, making changes easier.

4.3.4 Scalability

The system should be able to handle an increasing number of users without performance issues.

Requirement: The system shall support a growing number of users without slowing down.
Implementation Strategy:
Optimize database queries for faster performance.
Use efficient data handling techniques to manage reservations smoothly.

4.3.5 Security

The system must protect user data and prevent unauthorized access.

Requirement: User data shall be secure and protected from unauthorized access.
Implementation Strategy:
Implement basic encryption for sensitive information.
Use secure login methods to prevent unauthorized access.
This ensures that ReserV8 is reliable, user-friendly, scalable, maintainable, and secure, providing a smooth and safe booking experience.

5 Other Requirements

This section defines additional requirements that are not covered elsewhere in the SRS.

5.1 Legal and Compliance Requirements

- The system must comply with GDPR for handling user data privacy.
- Payment processing must follow PCI DSS security standards for handling credit card transactions.
- Data retention policies must allow users to request account deletion per compliance guidelines.

5.2 Localization and Internationalization

- The system shall support multi-language capabilities for future scalability.
- Date and currency formats shall adapt based on user region settings.

Appendix A – Data Dictionary

Variable/Constant	Description	Possible States/Values	Related Operations/Requirements
UserID	Unique identifier for each user	Integer (Auto-incremented)	Used for authentication and reservations
UserType	Defines the role of a user	Student, Restaurant Admin, University Admin	Controls access permissions
ReservationID	Unique identifier for a reservation	Integer	Generated when a booking is confirmed
RestaurantName	Name of the restaurant	String	Displayed in search results and reservations
FacilityName	Name of the university facility	String	Used for university facility bookings
TableAvailability	Status of restaurant tables	Available, Reserved, Blocked	Updated when a booking is made or canceled
FacilityAvailability	Status of university facilities	Available, Reserved, Blocked	Managed by university admins

MealPreBooked	Indicates if a meal was pre-booked	Yes/No	Used for payment processing and refunds
PaymentStatus	Status of the payment for reservations	Pending, Completed, Refunded	Processed via the payment gateway
SearchRadius	User-defined search radius for restaurant/facility booking	Numeric (in km)	Adjusted via UI to refine search results
SessionTimeout	Auto logout after inactivity	15 minutes	Enforced for security reasons
RefundEligibility	Determines if a refund is allowed for cancellations	Full, Partial, No Refund	Based on restaurant policy
LoginStatus	Tracks whether a user is logged in	Logged In, Logged Out	Ensures authentication before booking

Appendix B - Group Log

Date	**Meeting Participants**	**Discussion Points**	**Decisions/Tasks Assigned**
15/02/25	Entire Team	Initial SRS Draft	Assigned sections for writing
20/02/25	T.Meghana, Shriya, Tanasi, Shreyas	Use Case Model	Created initial use cases
25/02/25	Steffy,Aashita, Vedha, S.Meghana	Functional & Non-Functional Requirements	Defined system functionalities
02/03/25	Full Team	Review of Use Cases	Updated flow of events
06/03/25	Full Team	Integration of all sections	Finalized the SRS structure
10/03/25	Full Team	Final Review & Submission	Proofread and checked for completeness