# Food Rescuer: Excess Food Management System

**Prepared by :** Logarathan S V - 2022503047

Sahana S - 2022503049

**Department :** Computer Technology

**Institution:** Madras Institute of Technology, Anna University.

**Date :** 14-11-2024

**Mentor:** Dr.R.Kathiroli

Computer Technology

Madras Institute of Technology, Anna University.

**Executive Summary**

The **Excess Food Management System** is a software solution designed to address food waste by connecting hotels with excess food to individuals and organizations in need. Hotels can list surplus food, which individuals can buy at discounted prices, while organizations receive food for free. Geospatial APIs are integrated to match users with nearby hotels within a **1-5 km radius**, ensuring efficient and local distribution. Developed using **Java Swing** for the GUI and **MySQL** for the database, the platform allows easy registration, view available food, and order notifications. This initiative helps minimize food waste, provides economic benefits to hotels, and supports communities in need.

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sno** | **Sections** | **Description** | **Page no** |
| **1** | Title Page | Project title, author, institution, date | 1 |
| **2** | Executive Summary | Summary | 3 |
| **3** | Introduction | Problem Statement , Objectives , Key features and Technical Highlights , Scope | 5 |
| **4** | Requirements Analysis | Functional and non-functional requirements | 6 |
| **5** | UML Diagrams | Use case , Class diagram , CRC , Dataflow , Sequence diagram , State machine diagram | 8 |
| **6** | System Design | System architecture, database, UI design, key challenges | 18 |
| **6** | Implementation | Technologies, database connection, code structure | 19 |
| **7** | Testing and Evaluation | Testing approach, test cases, performance evaluation | 29 |
| **8** | Conclusion | Summary of contributions and project outcomes | 33 |

**Introduction**

**Problem Statement**

Food waste is a significant issue faced by the hospitality industry, where surplus food often goes unused and discarded. At the same time, many individuals and organizations, such as orphanages and shelters, face food insecurity and lack access to affordable or free food. This project aims to bridge the gap between hotels with excess food and individuals or organizations in need. By providing a platform for hotels to list and distribute their surplus food, either at discounted prices or as donations, the system will reduce food waste, offer economic benefits to hotels, and support communities in need.

**Objectives**

* To provide a platform for hotels to donate or sell excess food, reducing unnecessary food waste.
* To enable organizations like orphanages, shelters, and other charitable groups to receive food donations free of charge.
* To help hotels earn revenue by selling excess food at lower prices to individuals, reducing waste and related costs.

**Key Features and Technical Highlights**

* Registration and Role Management : Hotels, individuals, and organizations can register and create personalized profiles. Different user roles ensure access to relevant features
* Geospatial Matching : The platform uses **geospatial APIs** to match individuals and organizations with hotels offering food within a **1-5 km radius**. Reverse geocoding is used to convert hotel coordinates into human-readable addresses
* Notifications: Real-time notifications are sent to user and hotels on placed orders , ensuring that both the hotel and the user stay updated on the order status.
* Database and Interface : The platform uses **MySQL** to securely manage user profiles, food listings, and orders, paired with an intuitive **Java Swing** GUI for easy registration, browsing, and transactions.
* Security and Social Impact : The platform secures user data while promoting food redistribution, reducing waste, and supporting community well-being.

**Scope**

This project focuses on creating a platform for hotels to manage and distribute excess food. Hotels can list surplus food for individuals at discounted rates or donate it to organizations. The platform includes features for location-based matching, real-time updates on food availability, and secure user interactions, all designed to reduce food waste and assist communities in need.

**Requirements Analysis**

**Functional Requirements**

1. **User Registration and Login**:

* Users (hotels, individuals, and organizations) must be able to register and log in securely.
* Users should have distinct profiles with relevant information based on their type (hotel, individual, organization).

1. **Hotel Food Listing**:

* Hotels must be able to list available surplus food, including details such as food type, quantity, price (for individuals), and availability.
* Hotels can specify whether the food is for sale or donation.

1. **Search and Location Matching**:

* Users (individuals and organizations) can search for hotels within a 5-10 km radius.
* The system must utilize geolocation and reverse geocoding APIs to match users to nearby food offers.

1. **Real-Time Availability Updates:**

* The system should update food availability in real-time to reflect when food is sold or donated.

1. **Order Notifications**:

* Users should be notified when their orders are confirmed or fulfilled.
* Hotels should receive notifications when an order or donation request is placed.

1. **Data Privacy**:

* Secure handling of user information, including encryption and privacy compliance.
* Protection of sensitive data, such as user personal details and transaction information.

**Non-Functional Requirements**

1. **Usability**:

* Designed with a simple, intuitive user interface to allow easy navigation for hotel owners, individuals, and organizations.
* Provides a smooth user experience with clear access to food listings, search functionality, and order placement.

1. **Performance**:

* Optimized to load food listings quickly and process user actions (e.g., orders, profile updates) without delays.
* Ensures responsive performance across both desktop and mobile devices for easy access to the platform.

1. **Scalability**:

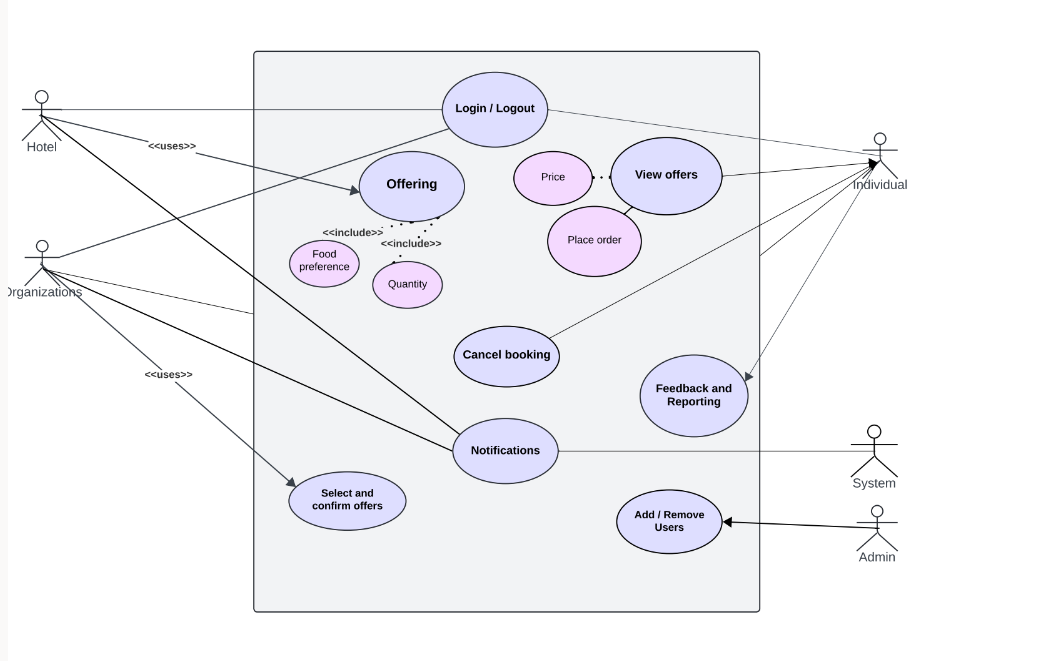
* Built to scale as the number of users, hotels, and food listings increases, ensuring the platform remains functional and responsive as demand grows.
* Database and application architecture support growth, allowing for additional features and higher user traffic.

1. **Security**:

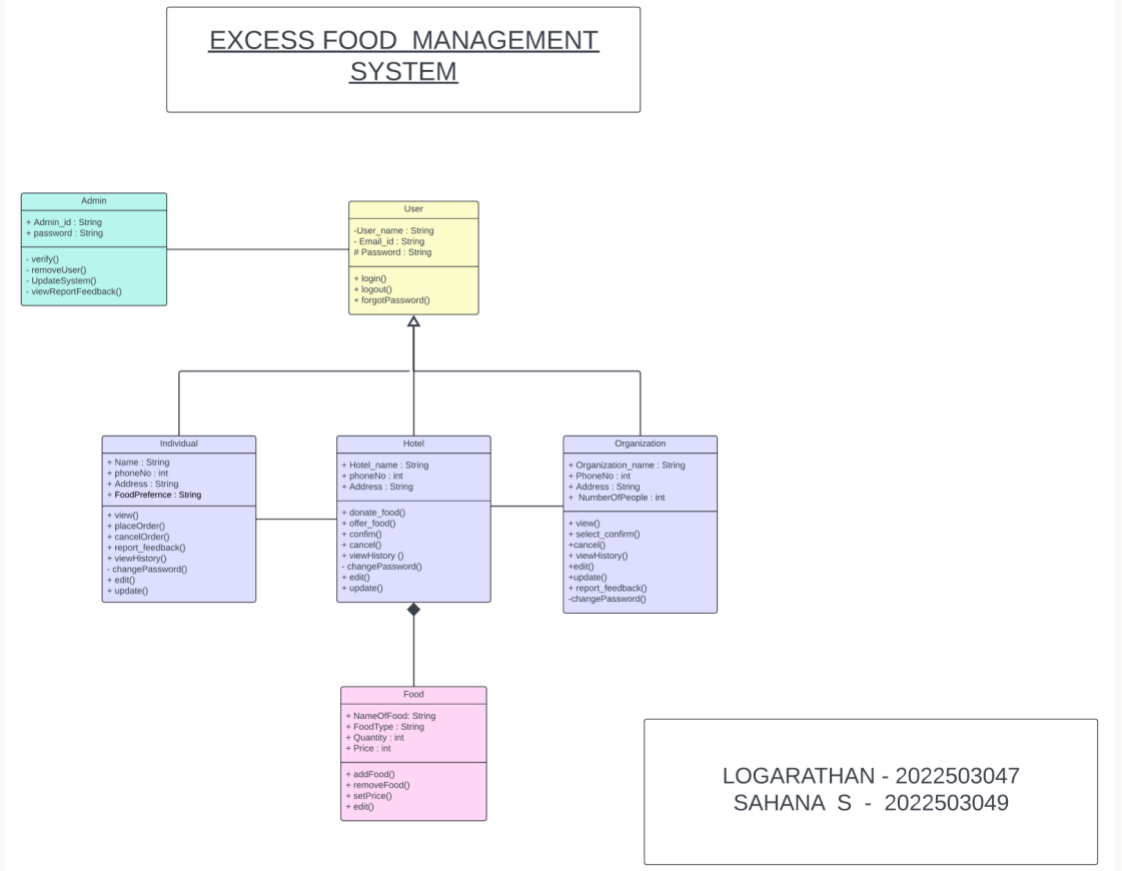
* Incorporates secure authentication mechanisms for user login and access control to protect sensitive data.
* Ensures compliance with data protection standards by encrypting sensitive information (e.g., personal details, food orders) and storing it securely.

**UML Diagrams**

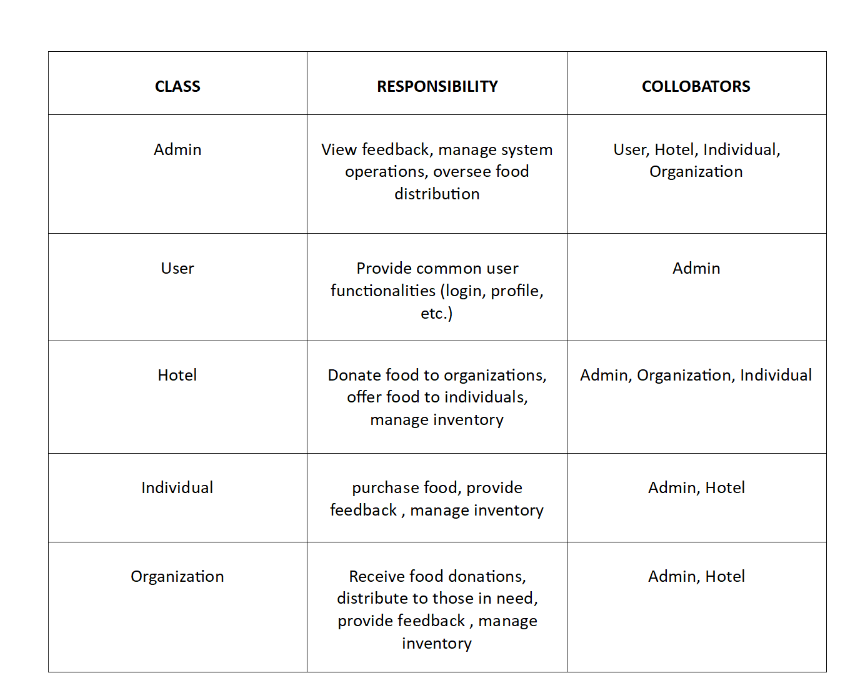
**Use Case Diagram:**

****

**Class Diagram**

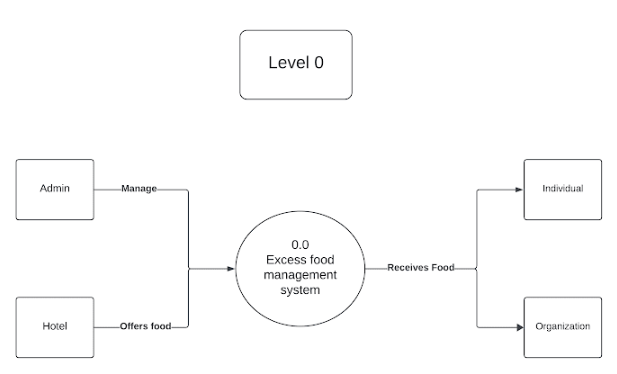
****

**Class Responsibility-Collaboration cardsCRC**

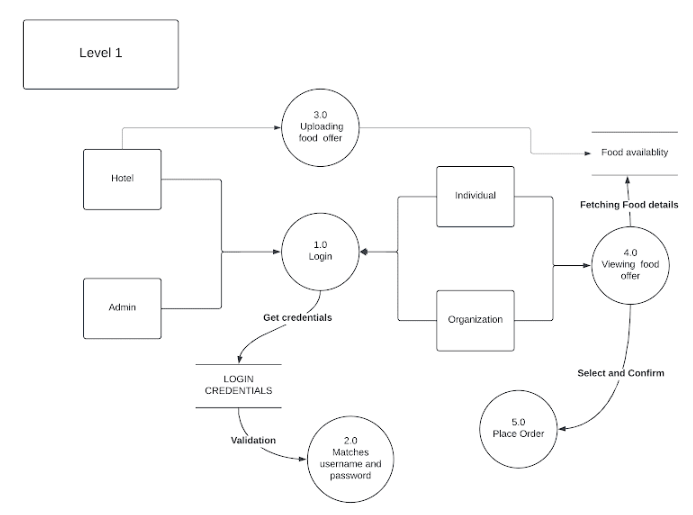
****

**Data Flow Diagram DFD**

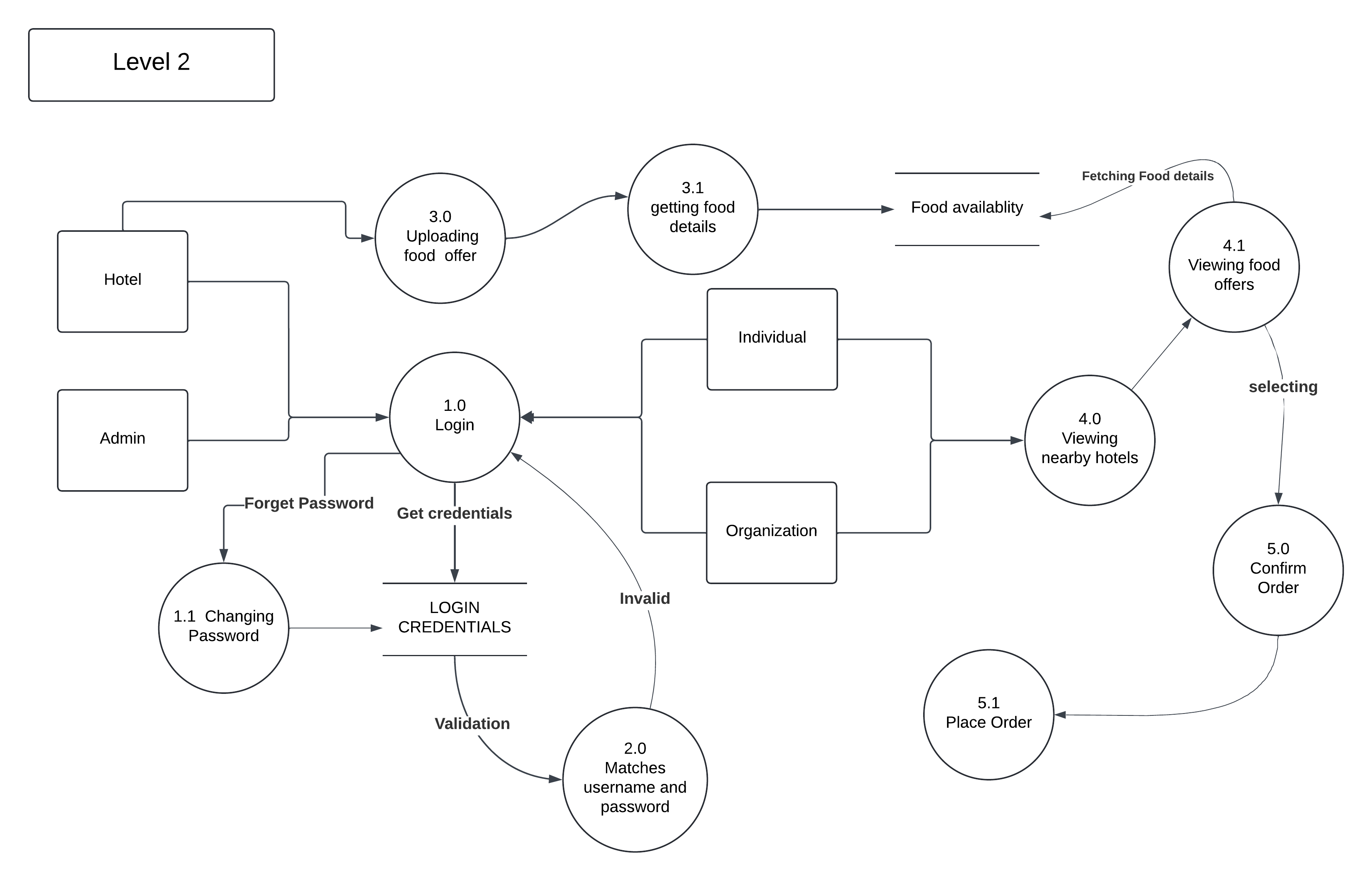
**Level 0:**

****

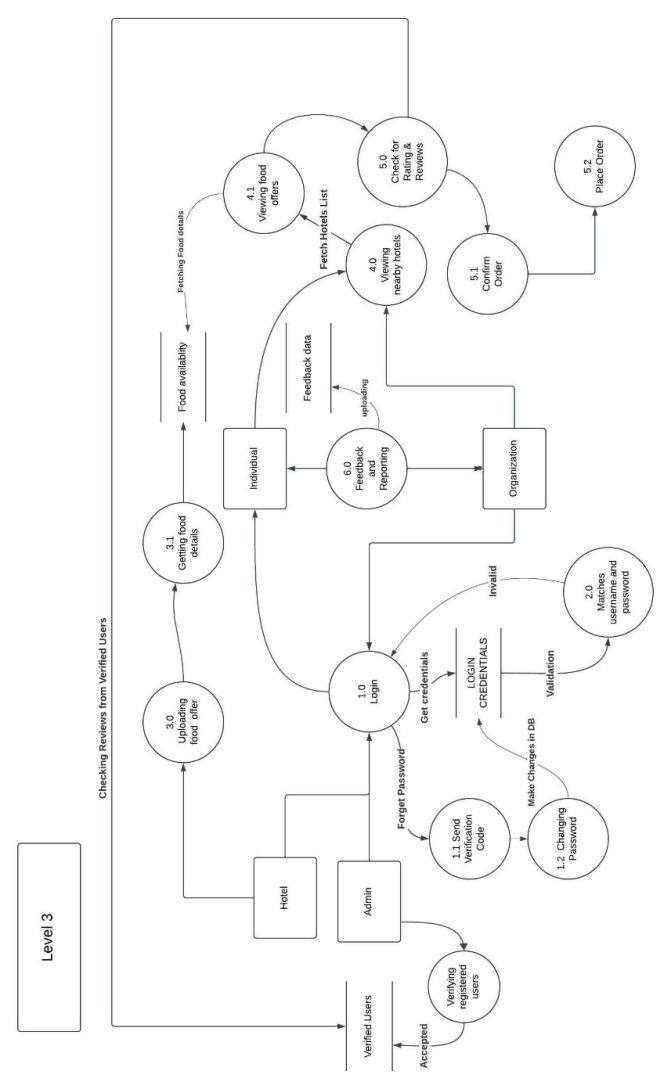
**Level 1:**

****

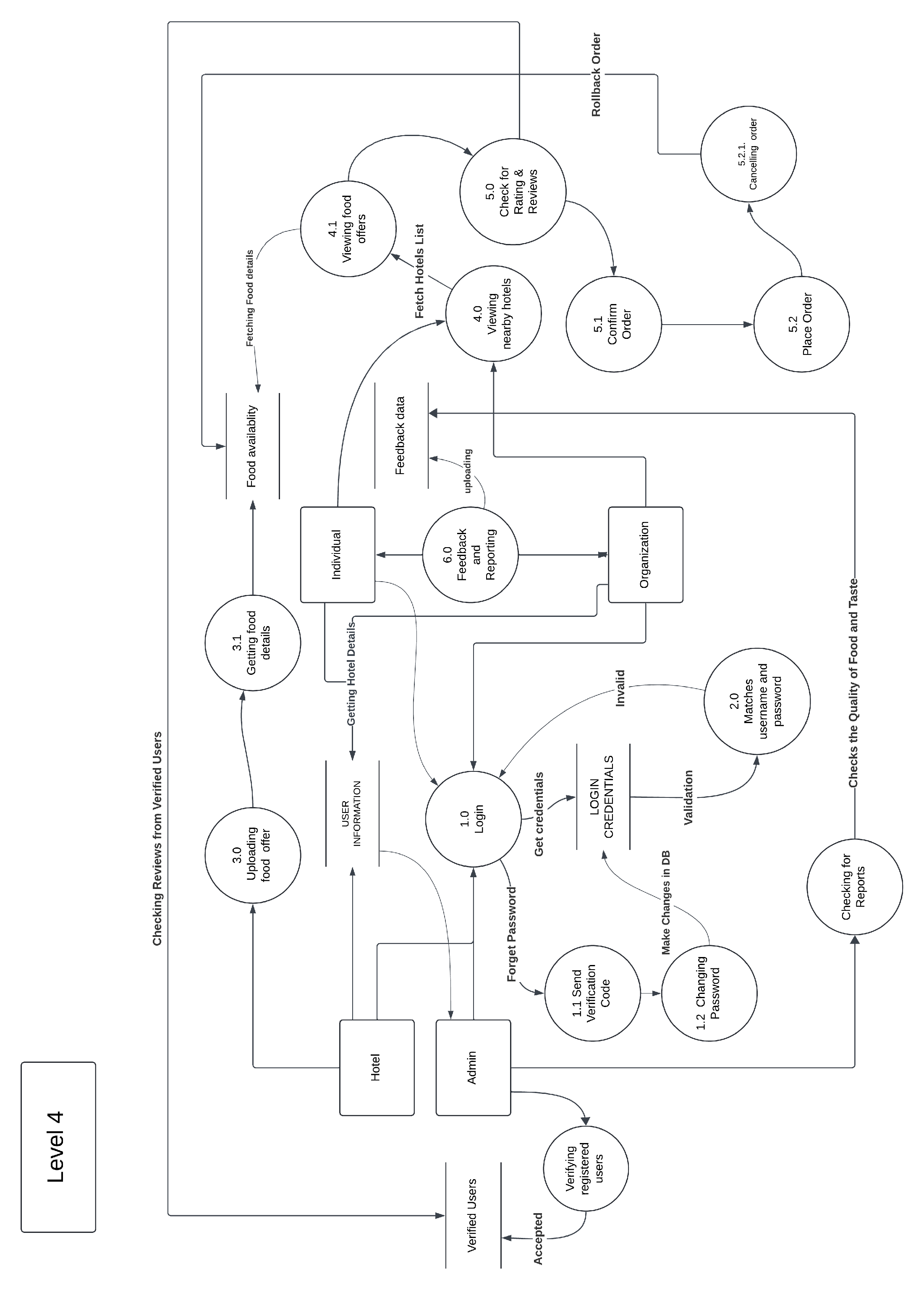
**Level 2:**

****

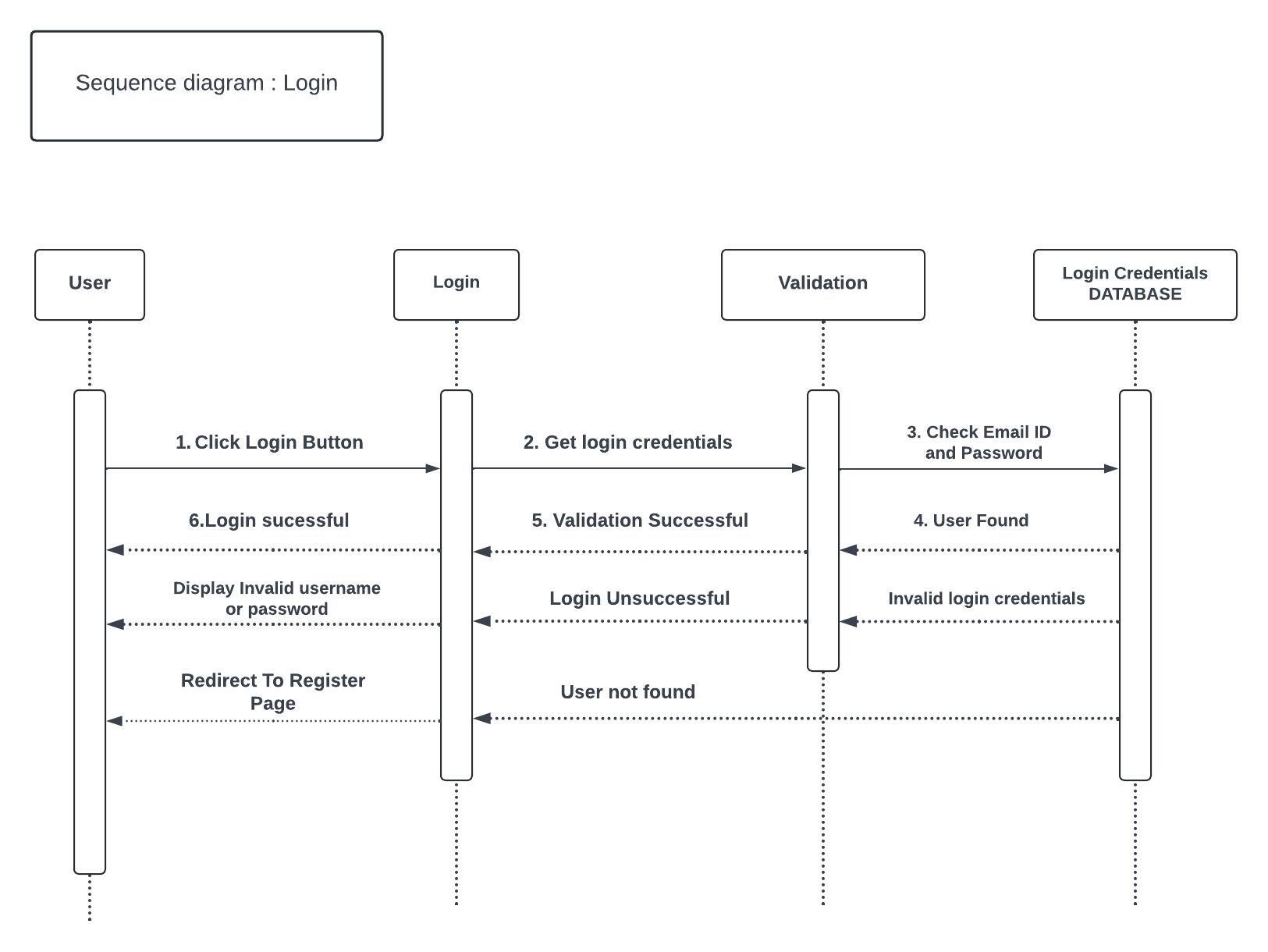
**Level 3**

****

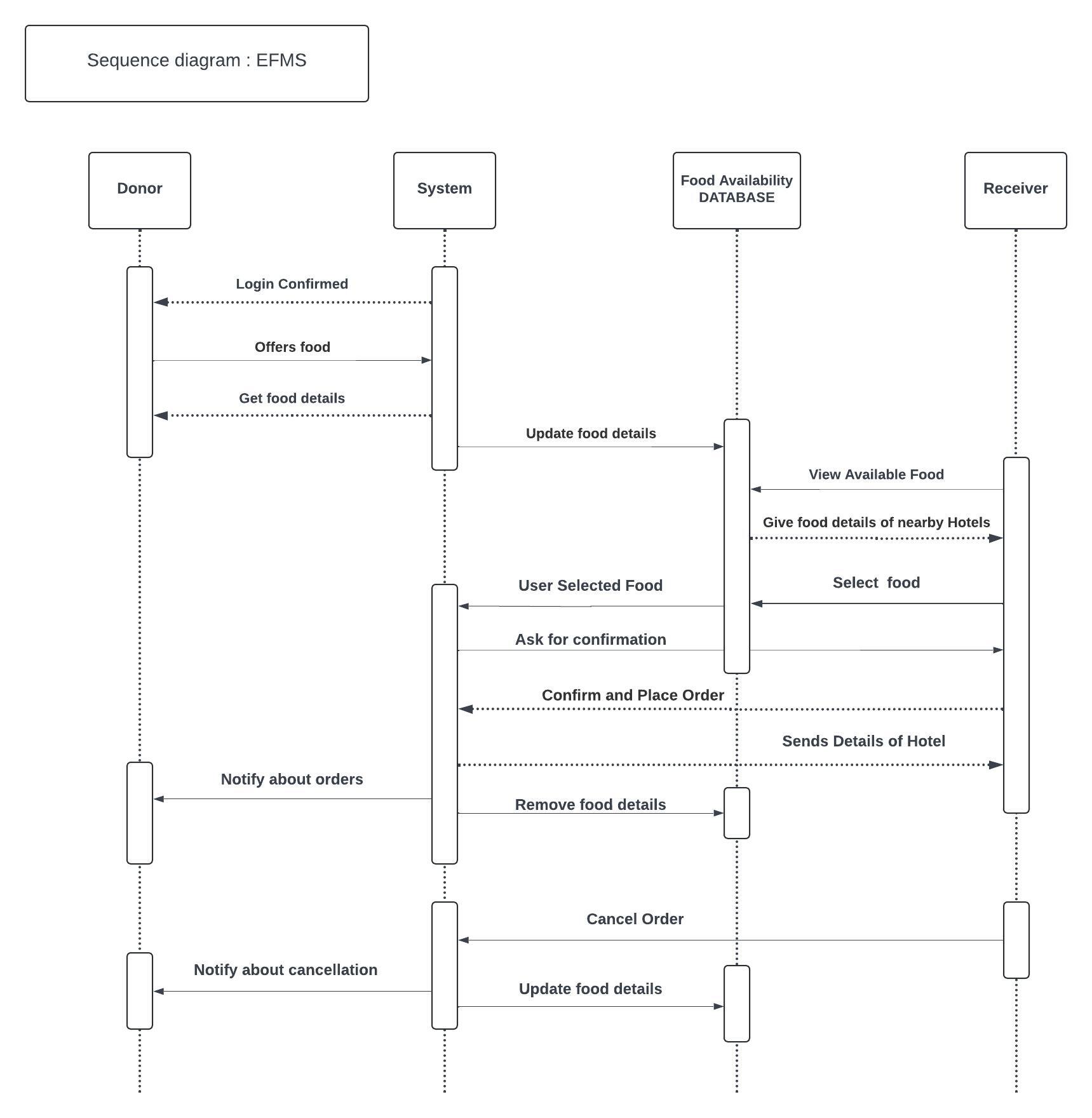
**Level 4:**

****

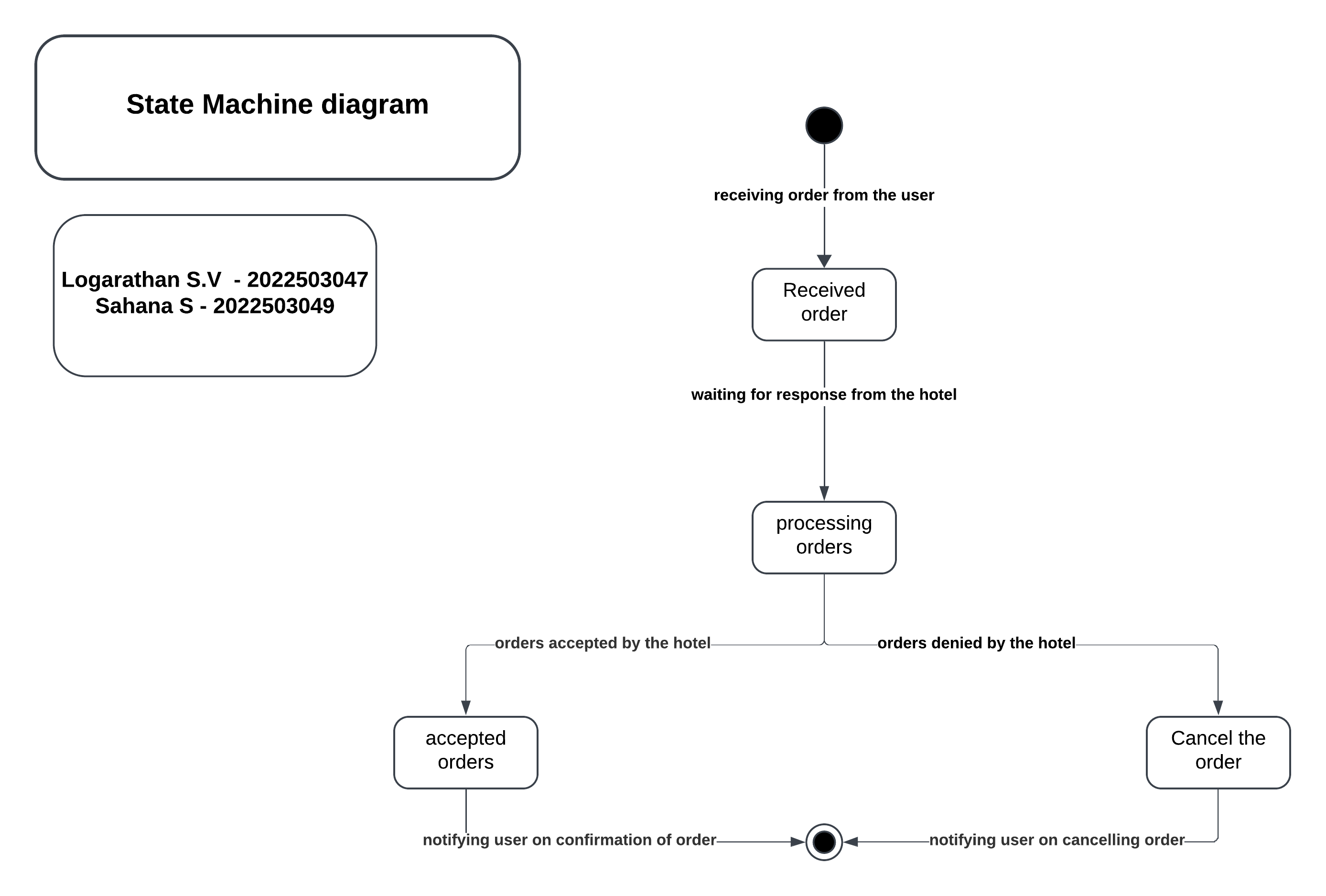
**Sequence Diagram :**



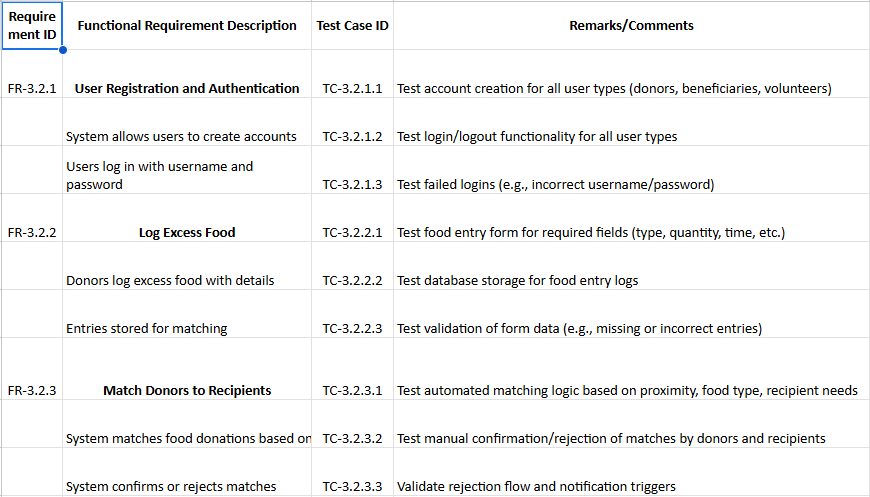
**System**



**State Machine Diagram**

****

**Traceability Matrix**

****

**System Design**

* **Architecture**: The platform will be built with a client-server architecture, separating the front-end (Java Swing GUI) and back-end (MySQL database) for better performance and maintenance.
* **Database Design**:MySQL will store user profiles, food listings, and transaction data, with normalized tables to ensure efficient data retrieval and maintain data integrity.
* **API Integration**:Geo-location services will use a reverse geocoding API to match users with nearby food listings based on their location.

**Key Challenges**

* Geospatial Data Accuracy and Matching : Integrating location-based services to match users with nearby hotels accurately (within a 1-5 km radius) and handling the complexities of reverse geocoding for proper address mapping
* Maintaining Data Consistency: Ensuring that information on food availability is consistently updated in real-time is vital to avoid inconsistencies between listed and actual availability, especially with high turnover
* Data Security and Privacy: Protecting sensitive user information, including personal data, requires implementing security measures to comply with privacy standards and prevent unauthorized access.
* Scalability and Performance: As the platform grows and user numbers increase, ensuring the system scales efficiently without compromising on performance or response time is crucial.

**Sample Implementation**

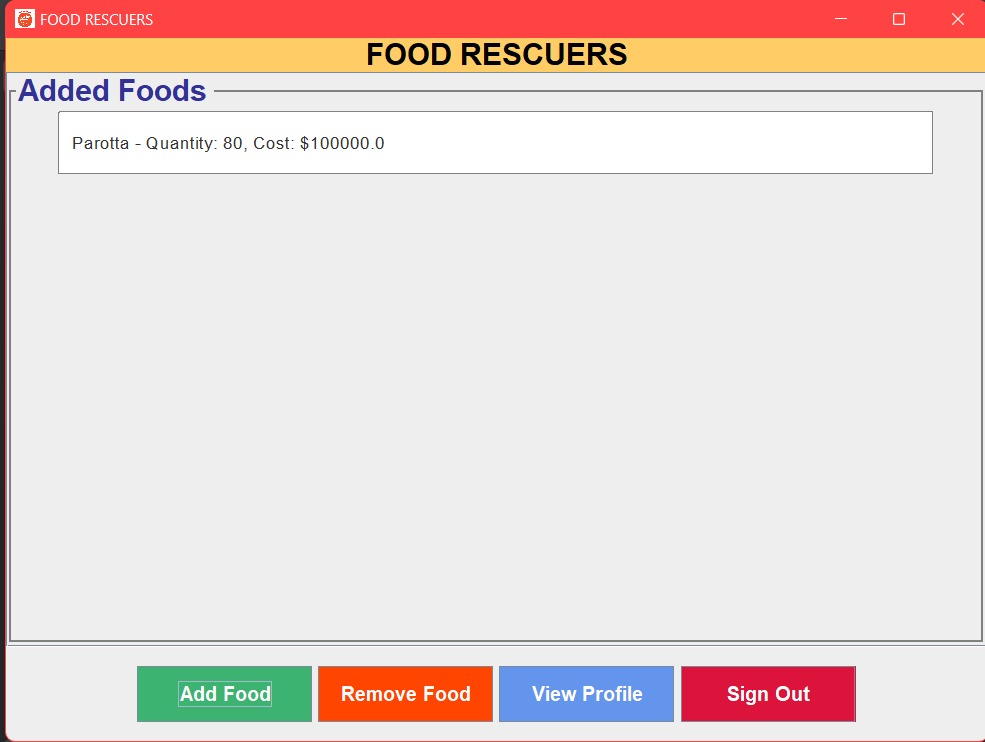
**Code:**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.\*;  
import java.util.ArrayList;  
  
public class HotelPage extends JFrame {  
  
 private CardLayoutcardLayout;  
 private JPanelmainPanel;  
 private JPanelhotelMainPage;  
 private JPaneladdFoodPage;  
 private JPanelremoveFoodPage;  
 private JPanelprofilePage;  
 private JPaneladdedFoodListPanel;  
 private JTextFieldfoodNameField, quantityField, costField;  
 private ArrayList<String>addedFoods;  
  
 // Constructor  
 public HotelPage() {  
setTitle("EFMS - Hotel Page");  
setSize(800, 600); // Set medium window size  
setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
cardLayout = new CardLayout();  
mainPanel = new JPanel(cardLayout);  
addedFoods = new ArrayList<>();  
  
initHotelMainPage();  
initAddFoodPage();  
initRemoveFoodPage();  
initProfilePage();  
  
mainPanel.add(hotelMainPage, "HotelMainPage");  
mainPanel.add(addFoodPage, "AddFoodPage");  
mainPanel.add(removeFoodPage, "RemoveFoodPage");  
mainPanel.add(profilePage, "ProfilePage");  
  
 add(mainPanel);  
cardLayout.show(mainPanel, "HotelMainPage");  
  
 // Load existing foods on startup  
loadFoodsFromDatabase();  
 }  
  
 // Method to load food items from the database  
 private void loadFoodsFromDatabase() {  
 String query = "SELECT \* FROM FoodItems";  
 try (Connection connection = connectToDatabase();  
 Statement stmt = connection.createStatement();  
ResultSetrs = stmt.executeQuery(query)) {  
  
addedFoods.clear();  
 while (rs.next()) {  
 String foodName = rs.getString("food\_name");  
 int quantity = rs.getInt("quantity");  
 double cost = rs.getDouble("cost");  
 String foodItem = foodName + " - Quantity: " + quantity + ", Cost: $" + cost;  
addedFoods.add(foodItem);  
 }  
updateAddedFoodList();  
  
 } catch (SQLException e) {  
e.printStackTrace();  
 }  
 }  
  
 // Method to clear the input fields in the Add Food Page  
 private void clearAddFoodFields() {  
foodNameField.setText("");  
quantityField.setText("");  
costField.setText("");  
 }  
  
 // Initialize the main hotel page  
 private void initHotelMainPage() {  
hotelMainPage = new JPanel(new BorderLayout());  
  
JLabeltitleLabel = new JLabel("EFMS - Hotel Page", JLabel.*CENTER*);  
titleLabel.setFont(new Font("Arial", Font.*BOLD*, 20));  
hotelMainPage.add(titleLabel, BorderLayout.*NORTH*);  
  
addedFoodListPanel = new JPanel();  
addedFoodListPanel.setLayout(new BoxLayout(addedFoodListPanel, BoxLayout.*Y\_AXIS*));  
 addedFoodListPanel.setBorder(BorderFactory.*createTitledBorder*("Added Foods"));  
updateAddedFoodList();  
  
hotelMainPage.add(new JScrollPane(addedFoodListPanel), BorderLayout.*CENTER*);  
  
JPanelbuttonPanel = new JPanel();  
JButtonaddFoodButton = new JButton("Add Food");  
JButtonremoveFoodButton = new JButton("Remove Food");  
JButtonviewProfileButton = new JButton("View Profile");  
buttonPanel.add(addFoodButton);  
buttonPanel.add(removeFoodButton);  
buttonPanel.add(viewProfileButton);  
hotelMainPage.add(buttonPanel, BorderLayout.*SOUTH*);

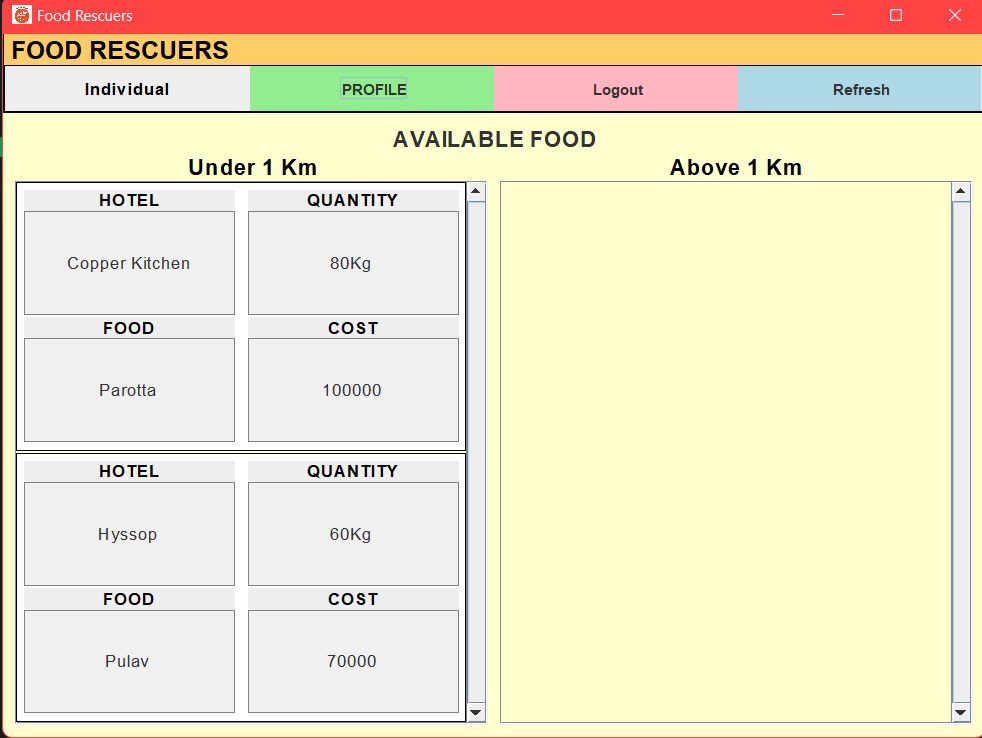
private void insertFoodToDatabase(String foodName, int quantity, double cost) {  
 // Validate inputs  
 if (foodName == null || foodName.isEmpty() || quantity <= 0 || cost <= 0) {  
System.*out*.println("Invalid input: foodName is empty or quantity/cost is non-positive.");  
 return; // Exit the method if the input is invalid, so food won't be inserted  
 }  
  
 // If input is valid, proceed to insert into the database  
 String query = "INSERT INTO FoodItems (food\_name, quantity, cost) VALUES (?, ?, ?)";  
 try (Connection connection = connectToDatabase();  
PreparedStatementpstmt = connection.prepareStatement(query)) {  
pstmt.setString(1, foodName);  
pstmt.setInt(2, quantity);  
pstmt.setDouble(3, cost);  
pstmt.executeUpdate();  
 } catch (SQLException e) {  
e.printStackTrace();  
 }  
 }  
  
  
 private void removeFoodFromDatabase(String foodName) {  
 String query = "DELETE FROM FoodItems WHERE food\_name = ?";  
 try (Connection connection = connectToDatabase();  
PreparedStatementpstmt = connection.prepareStatement(query)) {  
pstmt.setString(1, foodName);  
pstmt.executeUpdate();  
 } catch (SQLException e) {  
e.printStackTrace();  
 }  
 }  
  
 private Connection connectToDatabase() {  
 String url = "jdbc:mysql://localhost:3306/Available\_food";  
 String username = "root";  
 String password = "shaana";  
 try {  
 return DriverManager.*getConnection*(url, username, password);  
 } catch (SQLException e) {  
e.printStackTrace();  
 return null;  
 }  
 }  
  
 public static void main(String[] args) {  
SwingUtilities.*invokeLater*(() -> {  
 new HotelPage().setVisible(true);  
 });  
 }  
}

**Output:**

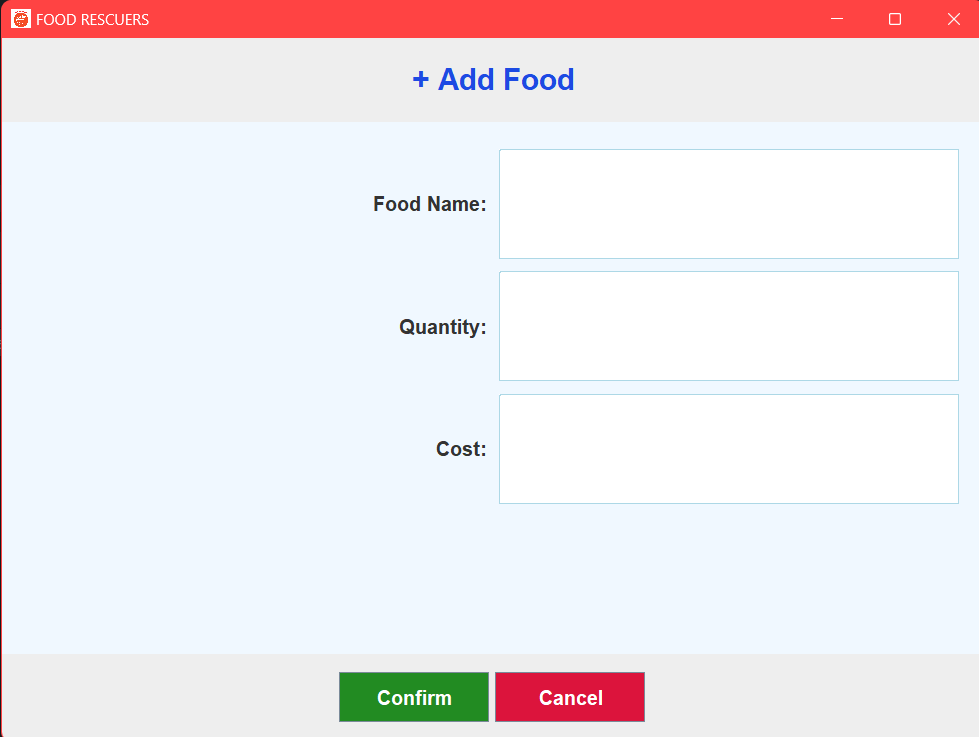
**HotelPage**



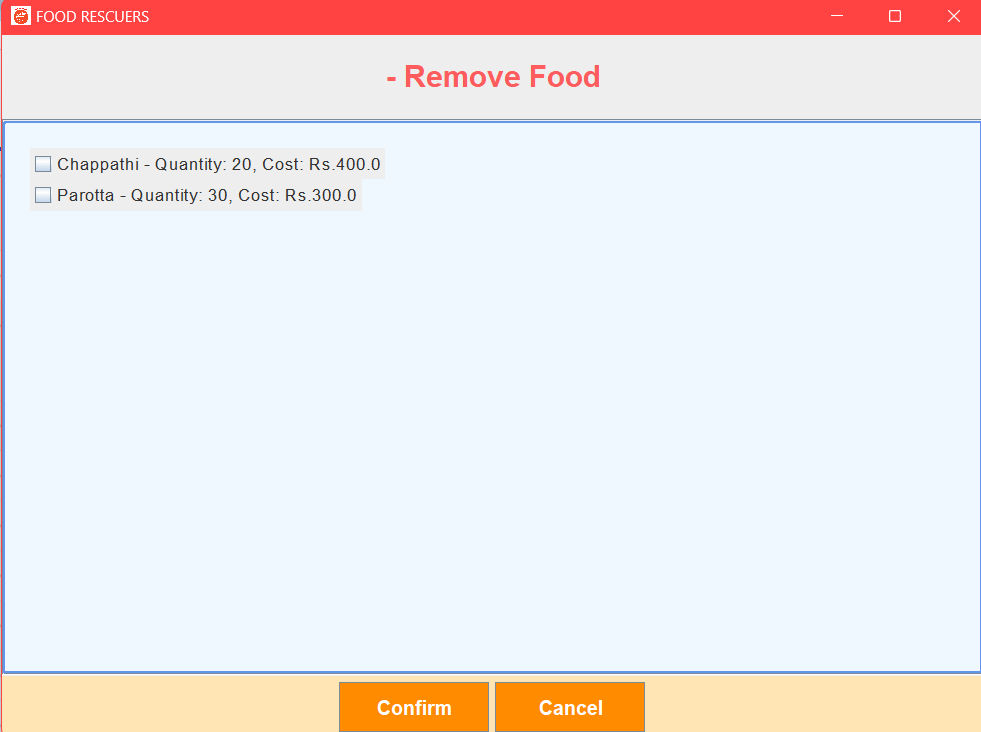
**User other than hotel**



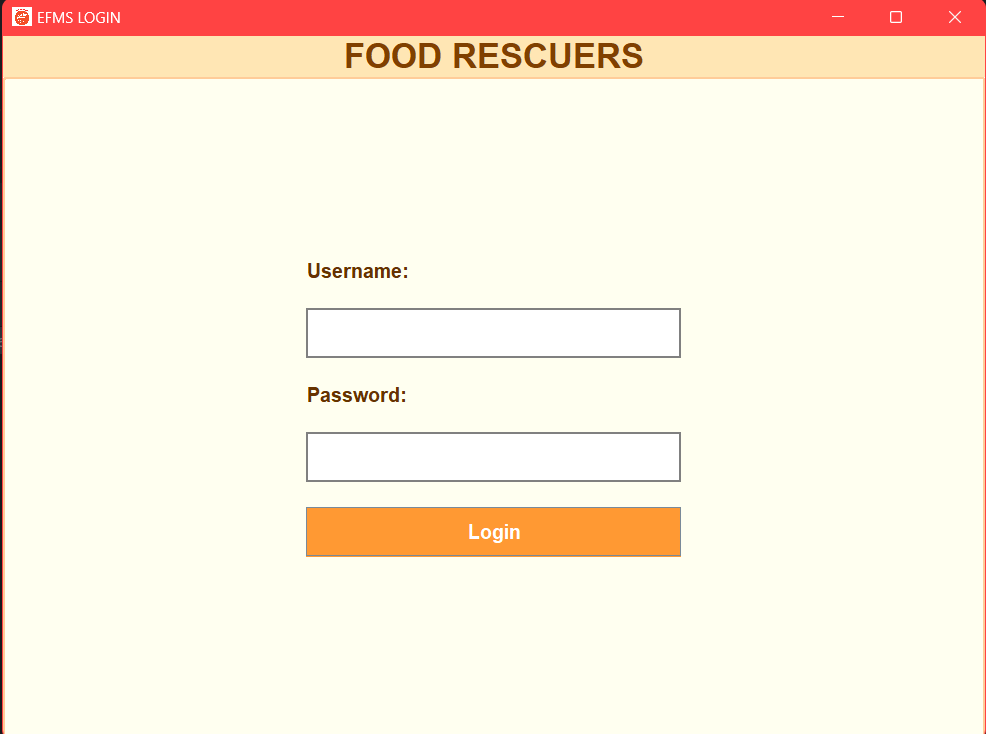
**Add Food**

****

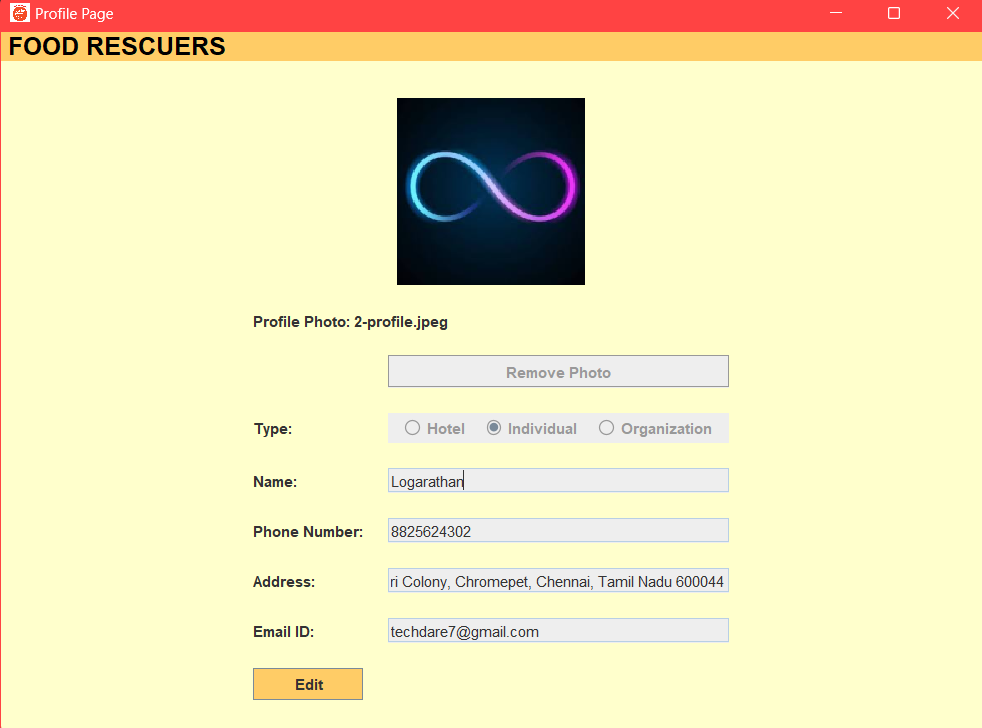
**Remove Food**

****

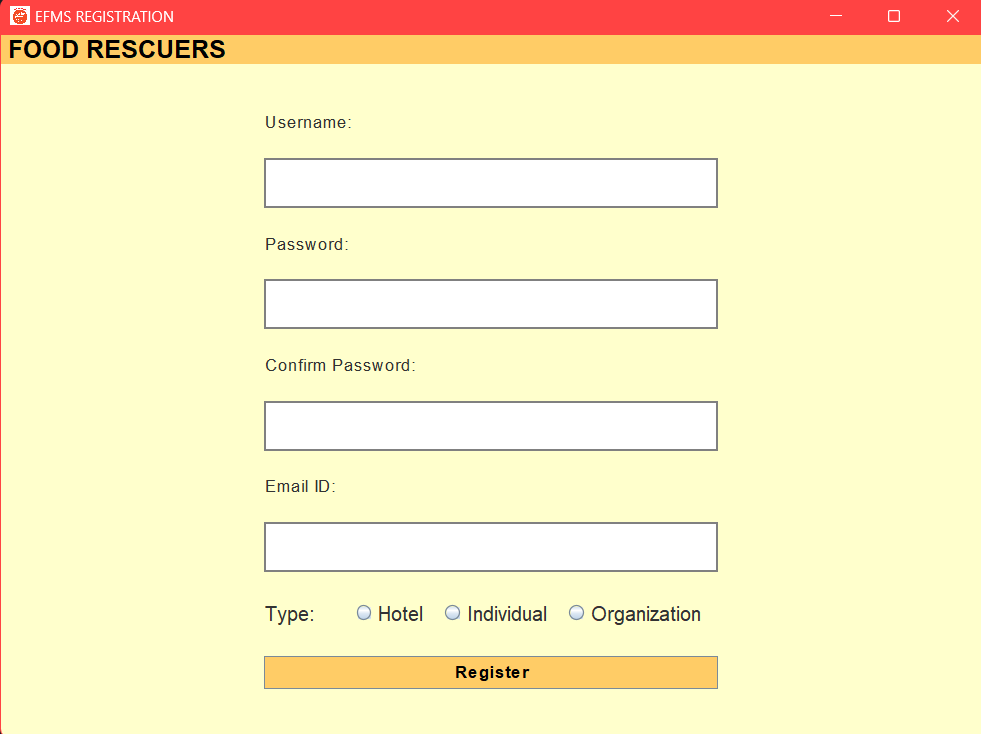
**Login**

****

**Profile**

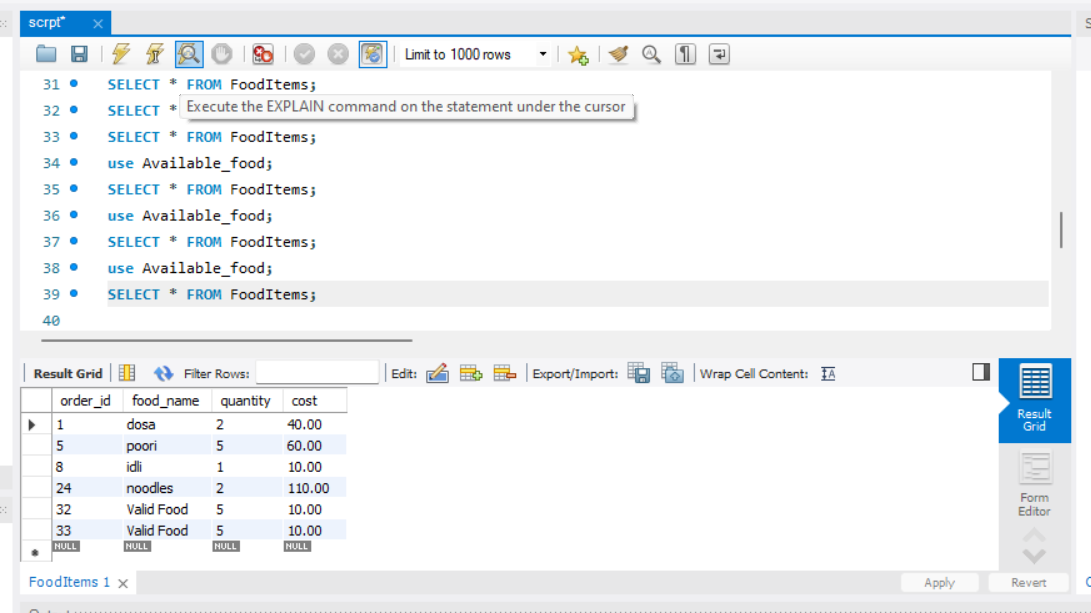
****

**Sign up**

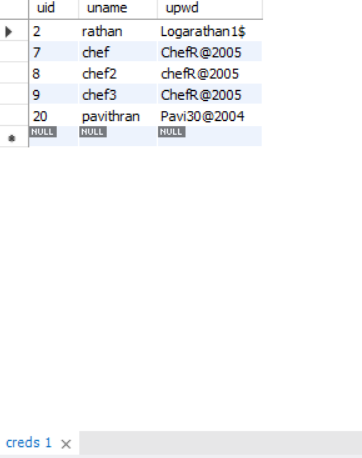
****

**Database:**

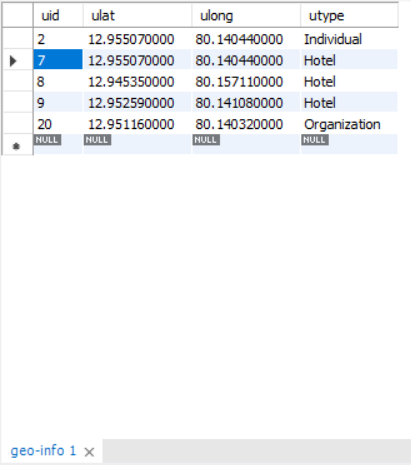
**Db of Hotel Page**

****

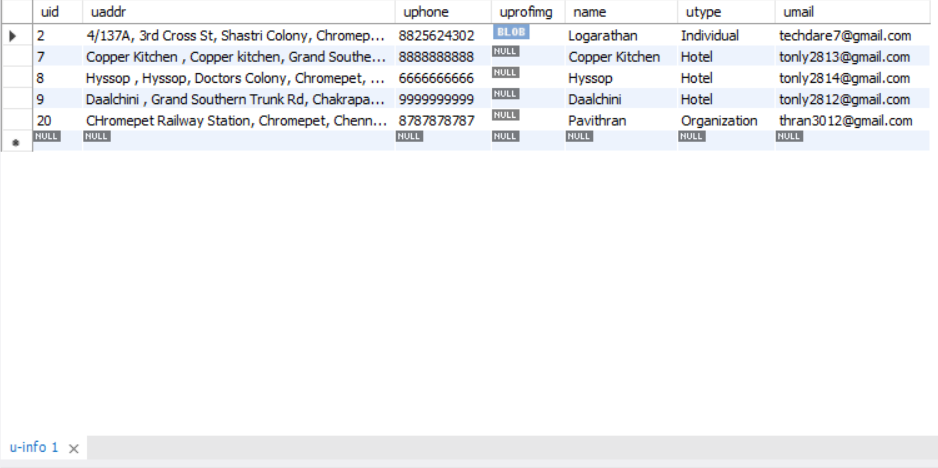
**Db of user Details**

****

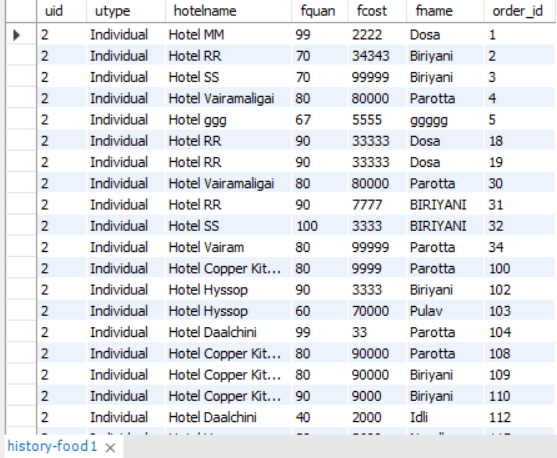
**Db of GeoInfo**

****

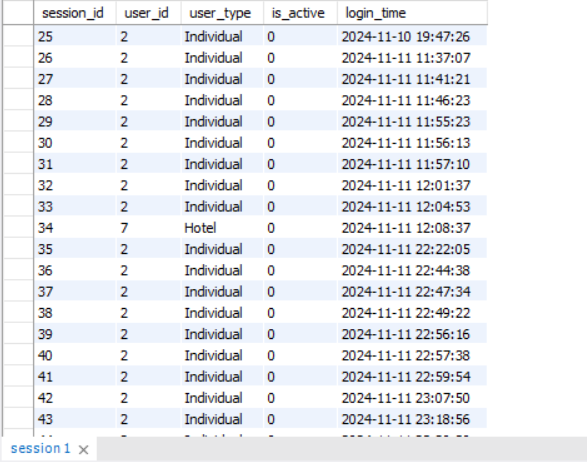
**Db of credentials**

****

**Db of food History**

****

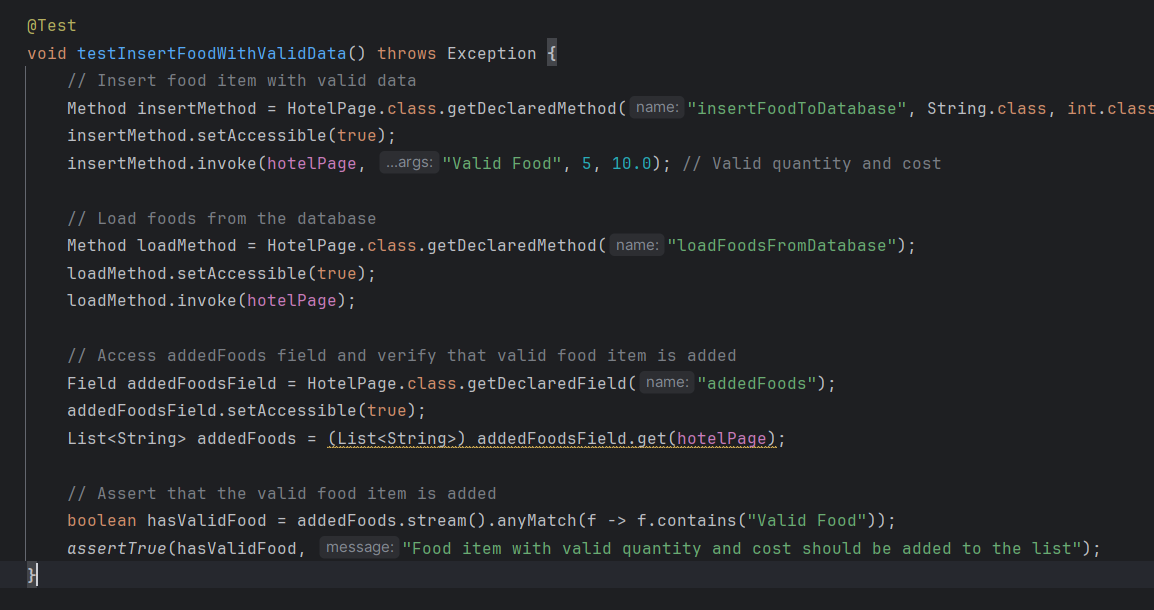
**Db of session management**

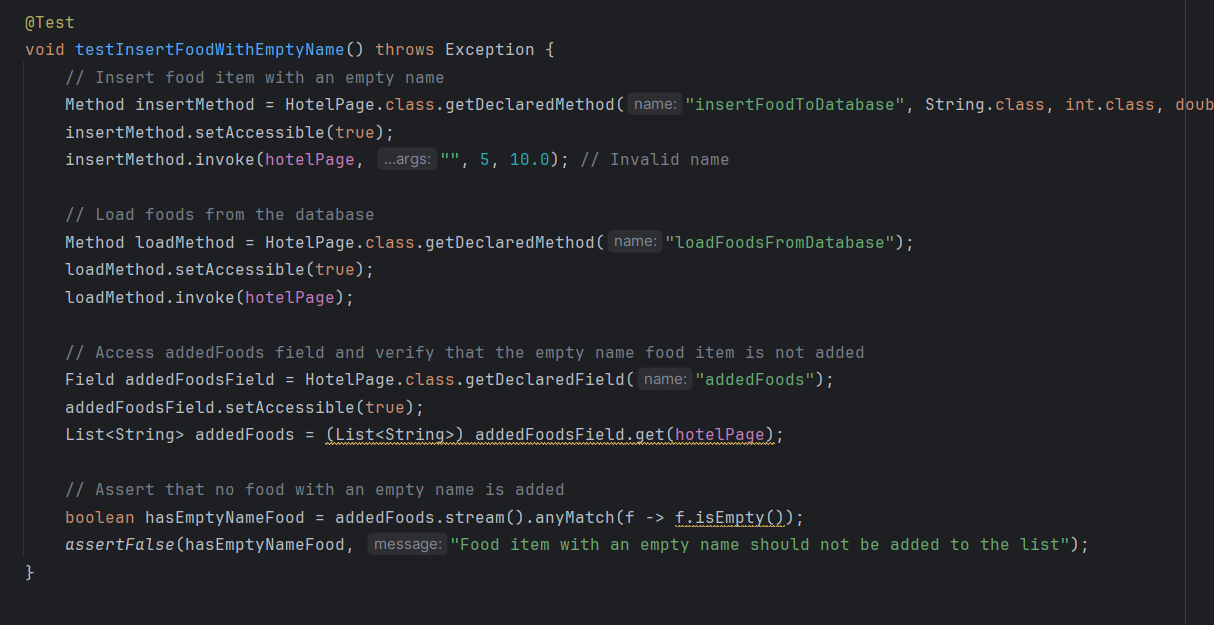
****

**Testing and Evaluation**

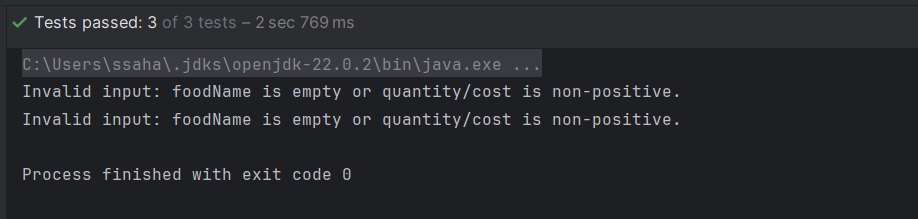
**Creating a Tester Class to test Hotel main Module:**

****

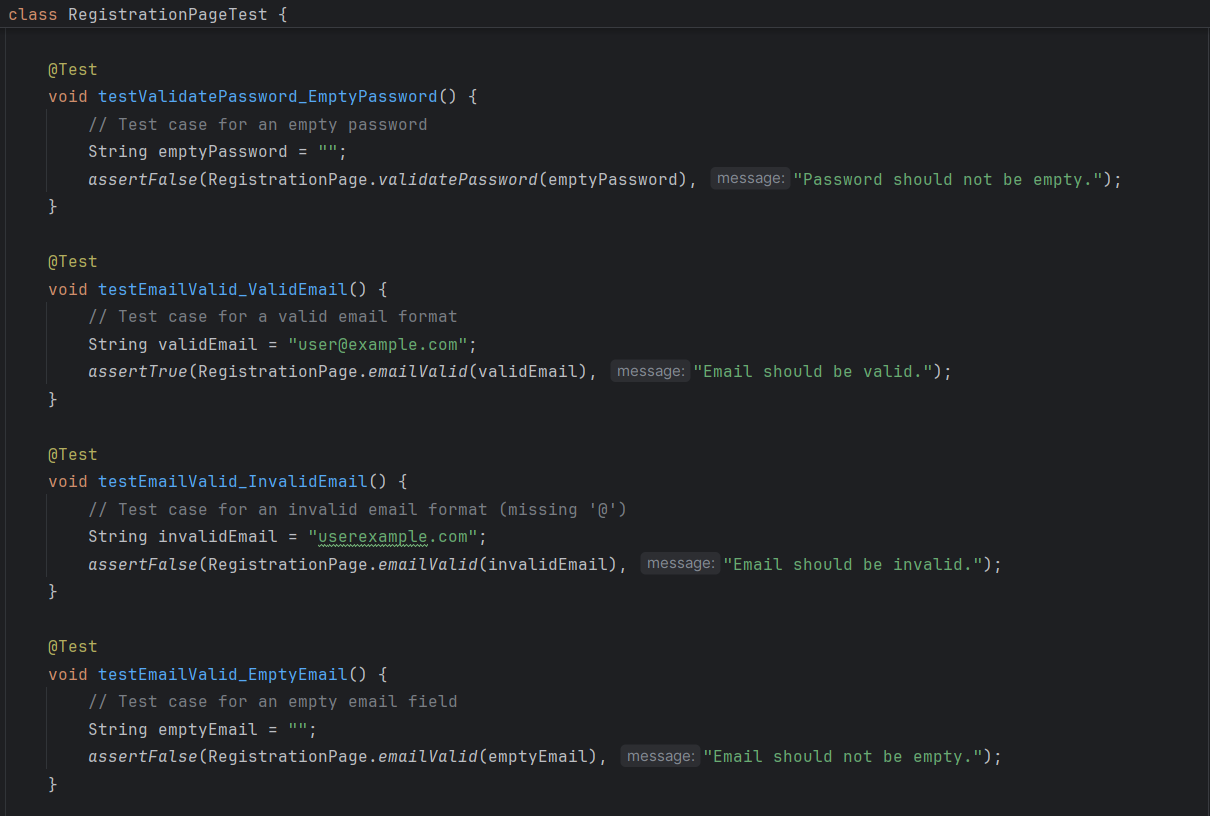
****

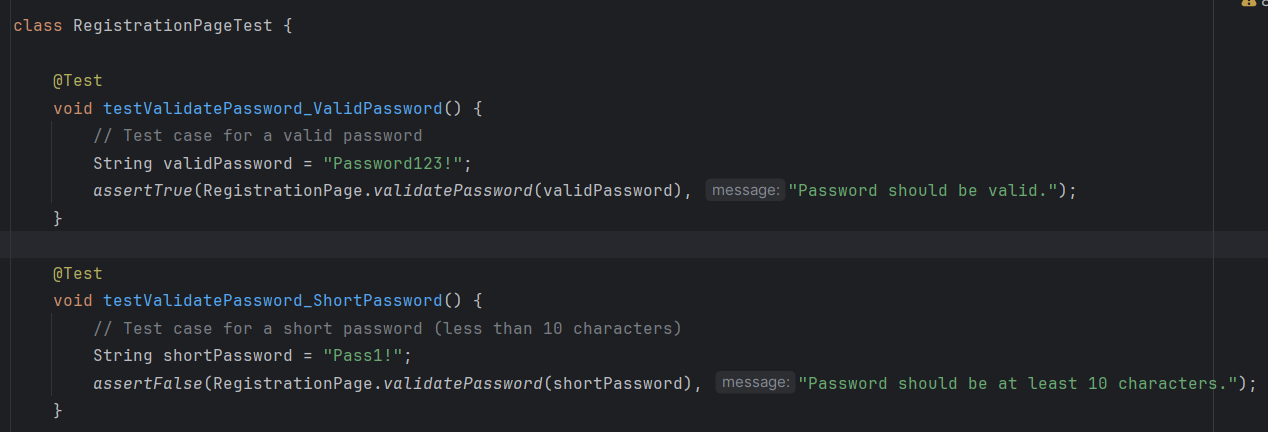
****

**Result on testing :**

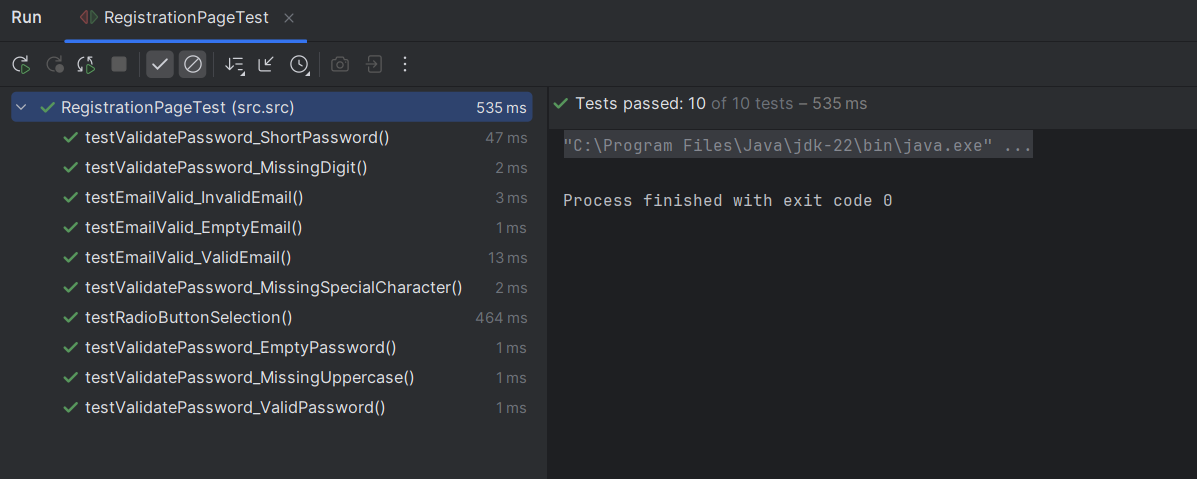
****

**Creating a Tester Class to test Registration Module**

****

****

**Result on testing :**

****

**Conclusion:**

This project presents a solution to the growing issue of food waste in the hospitality industry while addressing food insecurity in local communities. By developing a platform that connects hotels with excess food to individuals and organizations in need, we aim to reduce waste, provide economic benefits to hotels, and contribute to social good. The system’s design incorporates secure user authentication, efficient food listing management, and real-time updates to ensure smooth user experiences. The use of geolocation services enhances the platform’s ability to match users with nearby food offerings. With its intuitive user interface, robust database design, and secure environment, this project demonstrates a sustainable approach to tackling food waste and improving access to food for vulnerable groups. As the platform grows, it holds the potential to expand and adapt to new user needs, making a lasting impact on both the hospitality industry and society.