# 1. Introduction

This project report covers the development of a file-based food ordering web service and client interface application, designed to provide a user-friendly experience for customers, restaurant managers, and admins. The project uses a RESTful API to handle login, registration, and menu-viewing functionalities. Future expansions include order placement, user role definitions, and billing capabilities.

# 2. Business Impact

Using web services for a food ordering platform brings significant advantages:

* **Enhanced User Accessibility**: The APIs provide seamless access to users, allowing customers to search for menus quickly and securely.
* **Increased Efficiency**: Automating login and menu display with token-based authentication reduces manual tasks for restaurant managers and admins.
* **Scalability**: The separation of frontend and backend services allows for independent scaling as needed.
* **Data Security**: Passwords are encrypted before storage, and tokens are used for authenticated sessions, protecting user data.

# 3. Design and Architecture

## 3.1 Web Service Design Architecture

* **Framework**: The backend web service is implemented as a Maven project using RESTful architecture. The endpoints include AuthController for authentication (login and registration) and MenuController for retrieving and filtering menu items.
* **API Endpoints**:
  + **Login (POST)**: Authenticates the user and returns a session token for secure, seamless access.
  + **Register (POST)**: Registers new users, storing their details in a tab-separated file, with encrypted passwords.
  + **View Menus (GET)**: Retrieves a list of menu items, with the ability to filter based on keywords, providing dynamic menu searching.

## 3.2 MVC Pattern

* **View**: A separate Maven project serves as the frontend, developed using Java Swing for GUI components.
* **Model**: Data models (User, AuthDetails, Menu) encapsulate core application data.
* **Controller**: Manages data flow between the frontend and backend, with AuthController and MenuController coordinating the primary application actions.

## 4. Data Structures Used

* **HashMap (Singleton Pattern)**: To ensure efficient data management, a singleton HashMap instance is used to store and retrieve user and menu information. This approach centralizes data handling and allows for easy maintenance of user and menu records.
* **File Storage**: Registered users and menus are saved in a tab-separated file, providing a lightweight, database-free storage solution for rapid access and updates.
* **Token Generation**: Tokens for login are generated using a simple hashing method, which allows secure session handling without the need for a formal database.

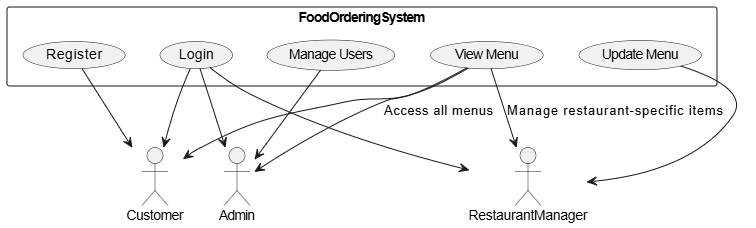
## 5. Security Considerations

* **Password Encryption**: User passwords are encrypted upon registration and stored securely, reducing the risk of unauthorized access.
* **Token-Based Authentication**: A hashing method is used to generate tokens, allowing secure session management and reducing repeated logins.
* **POST Requests for Sensitive Data**: Login and registration requests use POST to secure user credentials in transit.

## 6. Future Expansion

* **User Types**: Expanding roles for admins, managers, and regular users, each with varying access and management capabilities.
* **Order Placement**: Adding functionality for customers to place orders, view order history, and access billing information.
* **Billing and Payment Integration**: Implementing billing to support various payment methods and streamline transactions.

# 7. Use Case Diagram

Not all functionalities are implemented. This is a simple use case Diagram

# 8. Screenshots for use case scenarios loginEdited.pngregisterEdited.pngmenuEdited.png