Inventory Management

Inventory Management Mobile Application For Restaurants And Similar Businesses

Washington State University

Team One

Amrit Banga

Tucker Brisbois

Zijian Zhang

[05/31/2024]

TABLE OF CONTENTS

I. Introduction

The main purpose of this document is to introduce the process of the project from concept to practice, from setting up the project to full realization. The audience of this document is developers who want to develop similar projects, hoping that this project can inspire their projects. This project is based on the iOS system and creates a restaurant inventory management mobile application. Users can track all recorded inventory and mark expiration dates and costs. It can be saved in the cloud and accessed on multiple devices.

II. System Overview

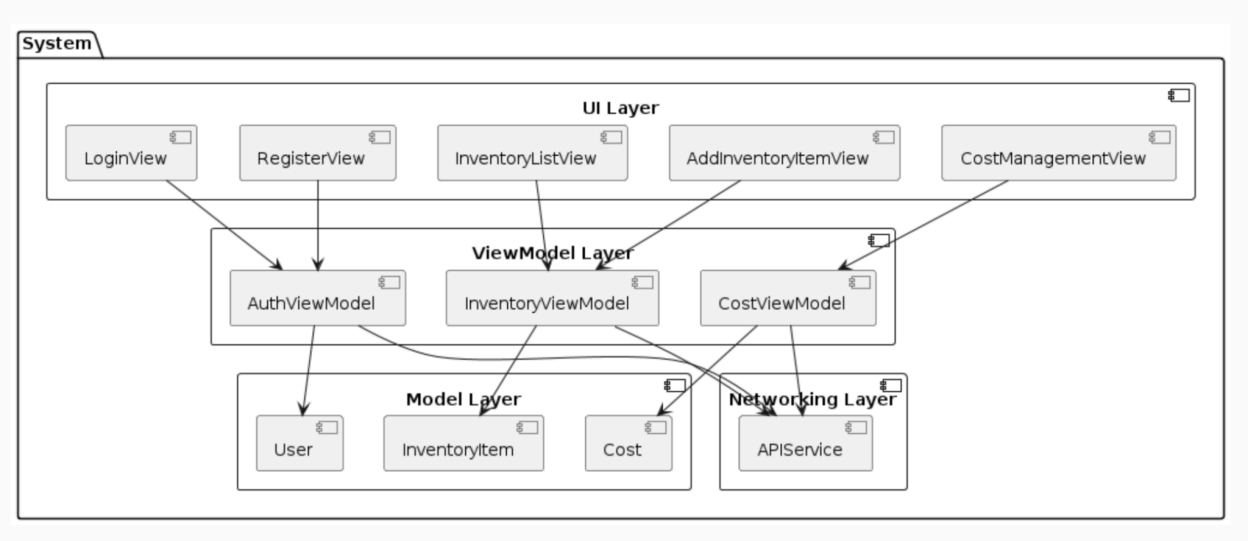
This project consists of four parts. The first part is the identity verification part, including 2FA, user registration, and user information verification. The second part is the inventory management part, including adding food information, deleting food information, modifying food information, and finding food information. The third part is the cost management part, including statistics on the cost of purchasing raw materials and food loss during storage. The fourth part is the network part, including connecting the system to the Internet so that data can be updated in real-time in multiple clients.

III. Architecture Design

III.1. Overview

The architectural design the team will be using for this application will be the well known Model-View-ViewModel (MVVM) layer design. This will allow clear separation between different parts of the application such as data logic, user interface (UI), etc. With this design, testing, updating, and maintaining the system will be much simpler and efficient.

- In the Model layer, we will have it handle all business logic and data manipulation. This layer will also communicate/call on the API to connect with the database
- In the View layer, we will have it handle all UI tasks such as the UI for login, logout, user creation, home page, user interaction, etc.
- The ViewModel layer is the in between for the Model and View layers. This will connect the data manipulation and everything else to the UI so the user can view it and/or manipulate it further.



III.2. Subsystem Decomposition

The system will be divided into 4 subsystems: Authentication, Inventory Management, Cost Management, and Networking. By dividing the system into these 4 subsystems, the team will be able to work on separate systems without much overlap or confusion.

- The Authentication subsystem will handle tasks such as user creation, user login, user logout, user session maintenance, and validation of user details.
- The Inventory subsystem will manage inventory items, creating new inventory items, deleting inventory items, and tracking inventory.
- The Cost subsystem will manage the costs of all inventory items past and present.
- The Networking subsystem will handle all API calls to the database for data storage and authentication service for user credentials.

I.1.1. 2FA (2 Factor Authentication) System [Authentication]

a) Description

The 2FA system will serve as an additional checkpoint to users in order to further protect user data. The 2FA will be evoked right after the log in page and ask the user to enter a unique code sent to their mobile number. If entered correctly, the user will be granted access to the application.

b) Concepts and Algorithms Generated

The 2FA system will need to evoke a third party backend service to send out messages using SMS. A popular one and one that the team here will most likely use is FIrebase. It will be imported into the code and then multiple functions will be created to implement subsystems such as sending the code, verifying the code, and then UI details as well.

c) Interface Description

Services Provided:

1. Service name: Send Verification Code

Service provided to: Firebase backend
Description:
This method will evoke the Firebase backend service to send a SMS to the user containing the verification code.

2. Service name: Verify Verification Code
Service provided to: Firebase backend
Description: This method will evoke the Firebase backend service in order to compare what code the user imputed vs what the Firebase service sent. If they match, then the user is allowed in. If not, then the user is prompted to try again.

Services Required:
Names of the required services and the subsystems that provide them.
Firebase Authentication provided by Firebase.

I.1.2 Log in System [Authentication]

d) Description

The log in system will be a simple username and password requiring system. This system will also include the create an account system as well as the two are intertwined.

e) Concepts and Algorithms Generated

The system will utilize a backend service to store credentials and have email and password authentication. We will also utilize Firebase for this system as well since the team is already planning on using it for the 2FA system. We will need to import Firebase into the code pages that are utilizing it. We will need to create functions for creating an account, logging in, and logging out.

f) Interface Description

Services Provided:

3. Service name: Create an account
Service provided to: Firebase backend
Description: This method will evoke the Firebase backend service to create an account which will do all the necessary actions such as prompting the user for their email and asking them to create a password.

Services Required:
Firebase Authentication provided by Firebase.

Service name: Logging in

Service provided to: Firebase backend
Description: This method will evoke the Firebase backend service to log into the application. The username/email and password will be prompted to be entered and then verified on the backend using Firebase,

Services Required:
Firebase Authentication provided by Firebase.

Service name: Logging out

Service provided to: Firebase backend
Description: This method will evoke the log out command from firebase and log the user out of the application and return then to the log in page.

Services Required:
Firebase Authentication provided by Firebase.

I.1.2. Inventory Management

a) Description

The inventory management subsystem is designed to track and manage the inventory of food items in a restaurant. This system will allow users to add, edit, and delete inventory items, including details such as item name, quantity, price, and expiration date. It will also provide functionalities to generate reports on inventory levels and expiration dates.

b) Concepts and Algorithms Generated

The system will utilize a backend service to store and manage inventory data. mySQL will be used to store inventory information. The system will include functions to add new inventory items, edit existing items, delete items, and retrieve inventory data for reporting purposes. Additionally, the system will implement data validation to ensure accurate and consistent data entry.

c) Interface Description

Services Provided:

- Service name: Add Inventory Item

Service provided to: mySQL

Description: This method will invoke the mySQL service to add a new inventory item. The user will be prompted to enter details such as item name, quantity, price, and expiration date, which will then be stored in the database.

- Service name: Edit Inventory Item

Service provided to: mySQL

Description: This method will invoke the mySQL service to update the details of an existing inventory item. The user can modify item details, and the updated information will be saved in the database.

- Service name: Delete Inventory Item

  Service provided to: mySQL

  Description: This method will invoke the mySQL service to remove an inventory item from the database. The user selects an item to delete, and the item will be removed from the database.

- Service name: Retrieve Inventory Data

  Service provided to: Application frontend

  Description: This method will retrieve inventory data from the database and display it on the application interface. This service will be used to generate reports and provide an overview of current inventory levels.

Services Required:

mySQL database

Cost Management

a) Description

The cost management subsystem is designed to track and manage the costs associated with inventory items in a restaurant. This system will allow users to input and track the cost of each inventory item, monitor overall expenses, and generate cost-related reports. The aim is to provide restaurant owners and managers with insights into their spending patterns and help optimize their purchasing decisions.

b) Concepts and Algorithms Generated

The system will utilize a backend service to store and manage cost data. mySQL will be used to store cost information due to its scalability and real-time synchronization capabilities. The system will include functions to add cost details to new and existing inventory items, update cost information, delete cost records, and retrieve cost data for reporting purposes. Additionally, the system will implement data validation to ensure accurate and consistent cost tracking.

c) Interface Description

Services Provided:

- Service name: Add Cost Details

  Service provided to: mySQL

Description: This method will invoke the mySQL service to add cost details to a new inventory item. The user will be prompted to enter the cost price of the item, which will then be stored in the database along with other inventory details.

- Service name: Update Cost Details

Service provided to: mySQL

Description: This method will invoke the mySQL service to update the cost details of an existing inventory item. The user can modify the cost price, and the updated information will be saved in the database.

- Service name: Delete Cost Details

Service provided to: mySQL

Description: This method will invoke the mySQL service to remove the cost details of an inventory item from the database. The user selects an item to delete its cost details, and the information will be removed.

- Service name: Retrieve Cost Data

Service provided to: Application frontend

Description: This method will retrieve cost data from the database and display it on the application interface. This service will be used to generate cost-related reports and provide an overview of overall expenses.

Services Required:

mySQL database

Network

a) Description

The network part is mainly used to solve the problem of information synchronization between multiple clients. The system ensures that the information of all clients, the activities and operation information of each user are updated in real time.

b) Concepts and Algorithms Generated

This system will use a client-server architecture. The server stores the database, and the user's iOS client will connect to the server for operation. For real-time data synchronization, WebSockets are used for operation. In this way, when the data changes, the server can push the data to all connected clients. During the operation, there will definitely be operation conflicts. At this time, the system will use the update timestamp to determine the latest update.

c) Interface Description

Service name: Real-Time Synchronization

Service Provided to: Clients

Description: The Synchronization log will be used at solving the problem of information synchronization between multiple clients. The service can provide consistent information updated in real time to all connected clients.

Service name: Conflict Resolution

Service Provided to: Clients

Description: This service mainly solves the problem of data conflicts when multiple clients try to update the same part of information. It maintains data integrity by implementing conflict detection and corresponding resolution strategies. When a conflict occurs, the system automatically determines that the updates of both parties are invalid, and notifies the conflicting users through WebSocket, thereby discovering and resolving the problem.

IV. Data design

We chose mySQL as the database. We have four parts in total, and each part needs some tables to store data. Here are the specific tables.

Part I: Authentication

Users table

user_id: Unique identifier for the user.

username: Unique username for the user.

password_hash: Hashed password for security.

email: Unique email address of the user.

two_factor_enabled: Boolean flag indicating if 2FA is enabled.

User Verification Table

user_id: Foreign key linking to the Users table.

password_hash: Foreign key linking to the Users table.

verification_code: Code used for verifying the user.

is_verified: Boolean flag indicating if the user is verified.

Part II: Inventory Management

Items Table

item_id: Unique identifier for the item.

item_name: Name of the item.

quantity: Quantity of the item in stock.

created_at: Timestamp of when the item was created.

Item Entries Table

item_id: Foreign key linking to the Items table.

item_name: Foreign key linking to the Items table.

entry_date: Date when the item was entered into the warehouse.

shelf_life_days: Shelf life of the item in days.

expiry_date: Expiry date of the item, calculated automatically.

quantity: Quantity of items in this entry.

created_at: Timestamp of when the entry was created.

Part III: Cost Management

Purchases Table

item_id: Foreign key linking to the Items table.

purchase_date: Date of the purchase.

purchase_cost: Cost of the purchase.

quantity: Quantity of items purchased.

created_at: Timestamp of when the purchase was recorded.

Part IV: Network

Sync Logs Table

sync_id: Unique identifier for the sync log.

sync_time: Timestamp of the sync event.

sync_status: Status of the sync event.

client_id: Identifier for the client performing the sync.

V. User Interface Design

Installation

1. Download the App:

- Visit the page where the application is avaliable.

- Download or install the application.

2. Initial Setup:

- Open the app after installation.

- Follow the on-screen instructions to complete the initial setup.

Configuration File Edits

1. Database Configuration:

- The app will automatically connect to the database. No manual configuration is needed from the user end.

Launch Daemon

1. Launching the App:

- Simply tap the app icon on your device to launch the application.

- The app will run in the background and sync data as needed, ensuring that inventory information is always up-to-date.

User Interface Description

The user interface is designed to be intuitive and user-friendly, catering to both tech-savvy and non-tech-savvy users. Below are detailed descriptions of each screen.

- Use Cases: User Registration and Login (Use Case 1)

Sign-Up Screen

Description: This screen allows new users to create an account by entering a username and password.

Sign-In Screen

Description: Existing users can log in by entering their credentials. After successful login, they are directed to the two-factor authentication screen.

- Use Cases: User Registration and Login (Use Case 1)

Two-Factor Authentication Screen

Description: Users enter the verification code received through a second source (email or SMS) to complete the login process.

- Use Cases: User Registration and Login (Use Case 1)

Main Dashboard

Description: After successful login, users are taken to the main dashboard, which provides access to inventory management and other functionalities.

- Use Cases: Adding a New Inventory Item (Use Case 2), Viewing and Editing Inventory (Use Case 3)

Inventory Management Screen

Description: Users can view, add, and edit inventory items. This screen displays a list of items with details like name, quantity, and expiration date.

- Use Cases: Adding a New Inventory Item (Use Case 2), Viewing and Editing Inventory (Use Case 3)

Add/Edit Inventory Item Screen

Description: This screen allows users to add new inventory items or edit existing ones. Users enter details such as item name, quantity, price, and expiration date.

- Use Cases: Adding a New Inventory Item (Use Case 2), Viewing and Editing Inventory (Use Case 3)

Use Case Mapping

- User Registration and Login (Use Case 1): Utilizes the Welcome Screen, Sign-Up Screen, Sign-In Screen, Two-Factor Authentication Screen, and Settings Screen.

- Adding a New Inventory Item (Use Case 2): Utilizes the Main Dashboard, Inventory Management Screen, and Add/Edit Inventory Item Screen.

- Viewing and Editing Inventory (Use Case 3): Utilizes the Main Dashboard, Inventory Management Screen, and Add/Edit Inventory Item Screen.

VI. Glossary

- iOS: The operating system on iPhones
- Java: A coding language used to built applications
- UI (User Interface): relates to the interface the user will interact with during the application
- 2FA: 2 Factor Authentication system. Refers to a system which adds further security to a login by asking for a randomly generated code sent to the user externally.
- MVVM: Model-View-ViewModel, a type of layered architecture for software development

VII. References

VIII. Appendices