IT314 Software Engineering

Lab Session 6

Project: Canteen Automation System

Prepared by

Team No. 31 | Group 6

Group Members

202001023 Vandan Patel
202001409 Kashyap Panchani
202001271 Dhruv Patel
202001415 Pinak Trivedi
202001403 Jaimin Baurasi
202001420 Divya Patel
202001405 Kenil Vaghasiya
202001430 Aryan Patel
202001407 Karan Patel
202001432 Aditya Jadeja
202001465 Jaykumar Navadiya

Under the guidance of

Prof. Khare, Prof. Tiwari and
Teaching Assistant Ankita Ma'am

Date

April 1, 2023

Index

1. Domain Analysis Model

1. Boundary, entity and control objects

In the context of canteen automation, we can identify the following boundary, entity, and control objects:

Boundary Object:
Boundary objects are the objects that represent the interface between the application and the external world, such as the users, devices, and systems that interact with the app. The boundary objects for the canteen automation app are:

- User Interface: The mobile app used by customers to place orders. It includes the User interface (UI) components which customers use to interact with the app and place orders.
- Partner Interface: The app used by the canteen owner to receive orders and manage them. It includes the features that the canteen owner uses to manage the orders, inventory, and menu.
- Payment Gateway: The system that handles payments made by customers. It includes the payment system from entering the details to the communication protocols used to process the transaction.
- Notifications and Alerts: It includes the notifications and alerts that the app sends to the customers or the canteen owner to inform them about important updates.

Additionally, other factors such as the physical area of the canteen, hardware devices used by customers and canteen owners, etc, can be considered as boundary objects in the canteen automation app. They separate the internal environment of the canteen from the external environment and define the scope of the canteen's operations. However, they are not directly related to the design and implementation of the software application.

Entity Object:

Entity objects are the objects that represent the real-world entities that are managed by the application. The entity objects for the canteen automation App are:

- Food Item: Entity representing the food items available for ordering.
- Order: Entity representing the order placed by the customer.
- User: Entity representing the customer who placed the order.
- Canteen: Entity representing the entity that manages the food items and receives orders.
- Payment: Entity representing the payments or transactions made by the customer with details such as payment amount and status.

Food items, Order, User, Canteen, Payment are among the core entity objects that are used in the canteen automation app. They

represent the data which is stored and manipulated by the app to provide the desired functionality to the users.
There can be other entity objects which can be included in the application based on the extra features provided to the user. For eg. Loyalty points, Ratings, Reviews, etc. Also, some of the control objects such as payment manager, inventory manager can have their own entity objects.

Control Object:
Control objects are the objects that represent the actions or operations that are performed by the application. The control objects for the canteen automation App are:

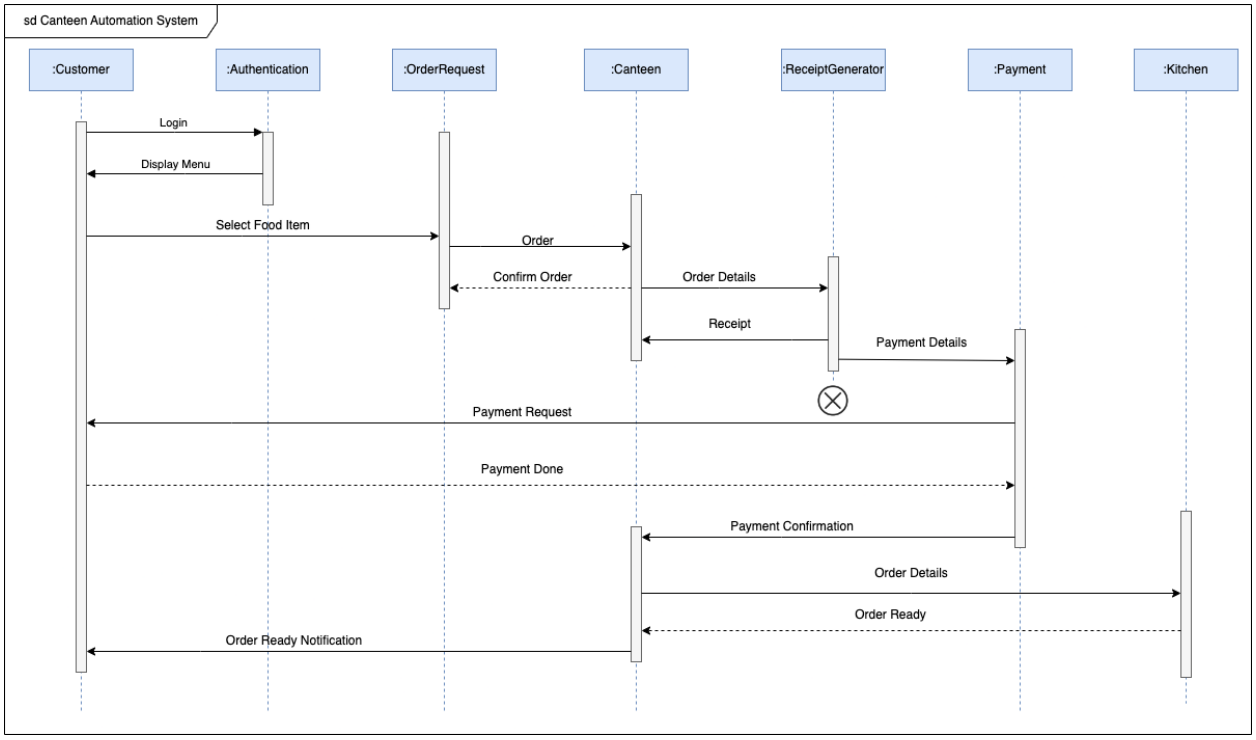
- Order Administration System to manage the flow of orders.
- Payment Processing System to manage the transactions.
- Menu Management System to manage the list of food items.
- User Authentication System to authorize users for accessing the app.
- Reports and Analytics System to generate reports and analytics based on the data collected by the app.

For the canteen to run well, these systems would be in charge of directing and managing the actions of the entity objects for their key functions and processes.

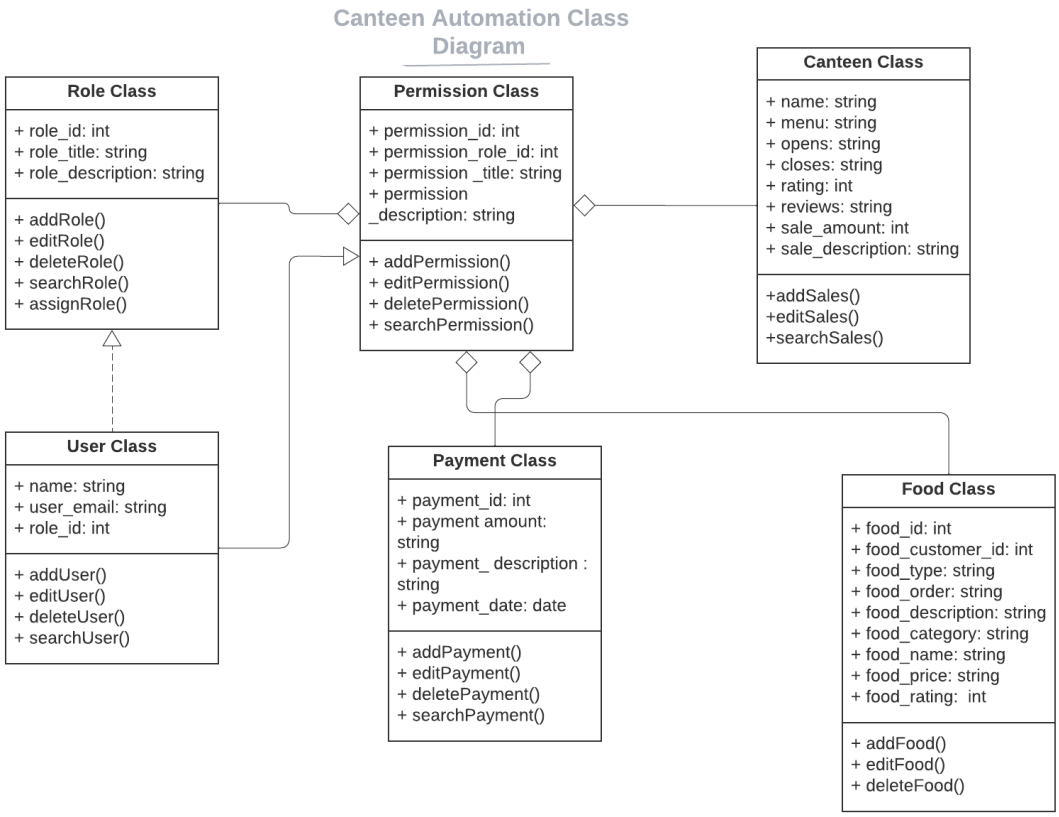
The design and implementation of these boundary, entity and control objects are critical to the success of the app and should

be carefully planned and tested during the development process.

2. Sequence diagram



3. Class Diagram



4. Design Goals

The specific needs and specifications of the canteen would determine the design objectives for the canteen automation system, however some examples of generic objectives might be:

- Efficiency Gains: The automation system should work to shorten wait times, speed up the serving process, and simplify the ordering and payment procedures. Mobile ordering, contactless payments, and self-service kiosks are a few examples of features that can help with this.

- Improving Customer Experience: The system should offer a pleasurable, straightforward, and customized customer experience. Features like reward programmes, individualized recommendations, and simple order customization can help with this.
- Ensure Accuracy: The system should work to reduce errors and guarantee that orders are handled and delivered precisely. Features like order confirmation displays, automated order tracking, and real-time inventory management can help with this.
- Protection of Customer Data and Secure Payment Processing: The system should make sure that payments are completed safely and securely. Features like encryption, secure payment gateways, and stringent data protection procedures can help with this.
- Delivering Analytics and Insights: The system must offer thorough analytics and insights that may be used to streamline processes, enhance customer satisfaction, and boost profitability. Predictive analytics, customer feedback analysis, and sales reports are a few aspects that can help with this.

5. High level system Design

5.1 Architecture of the Application

We are going to follow the multi-tier architecture for the development of the canteen automation system application.

The architecture used is described below.

The multi-tier architecture, also known as the multi-layer architecture, is a software architecture pattern used in designing large-scale, scalable and

secure applications. It is divided into three layers, each responsible for a specific set of functionalities:

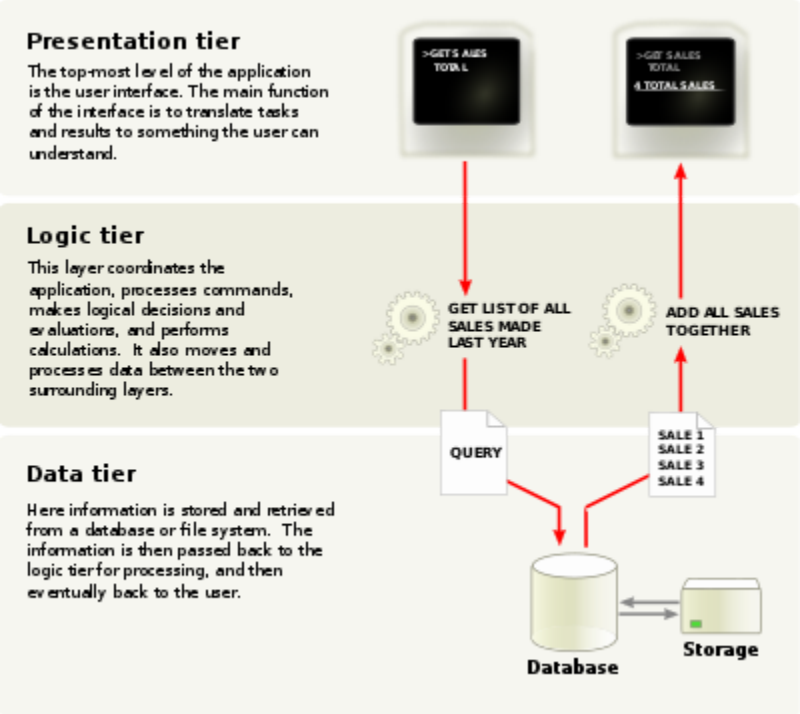
Presentation Layer: The presentation layer is responsible for presenting the data to the user in a user-friendly format. This layer is often referred to as the UI layer and includes user interfaces such as web pages, mobile applications, and desktop applications. It interacts with the business layer and displays the results to the user. We will be handling the frontend part via using the Flutter Dart UI elements such as material UI.

Application Layer (Back-end): The application layer is responsible for implementing the business logic of the application. It is also referred to as the middle or business layer. This layer contains the application's core logic and handles tasks such as data validation, processing, and manipulation. The application layer communicates with the presentation layer and the data layer to retrieve and manipulate data. We will be using the Flutter framework to connect with the database and for making REST http requests as well. Overall, the Flutter framework will handle the backend of the application.

Data Layer: The data layer, also known as the persistence layer, is responsible for managing the application's data. It includes the database management system and the data access layer. The data layer is responsible for storing and retrieving data from a database or file system. It communicates with the application layer to retrieve data and send it back to the user via the presentation layer. We are going to use Firebase and Elasticsearch for the data layer.

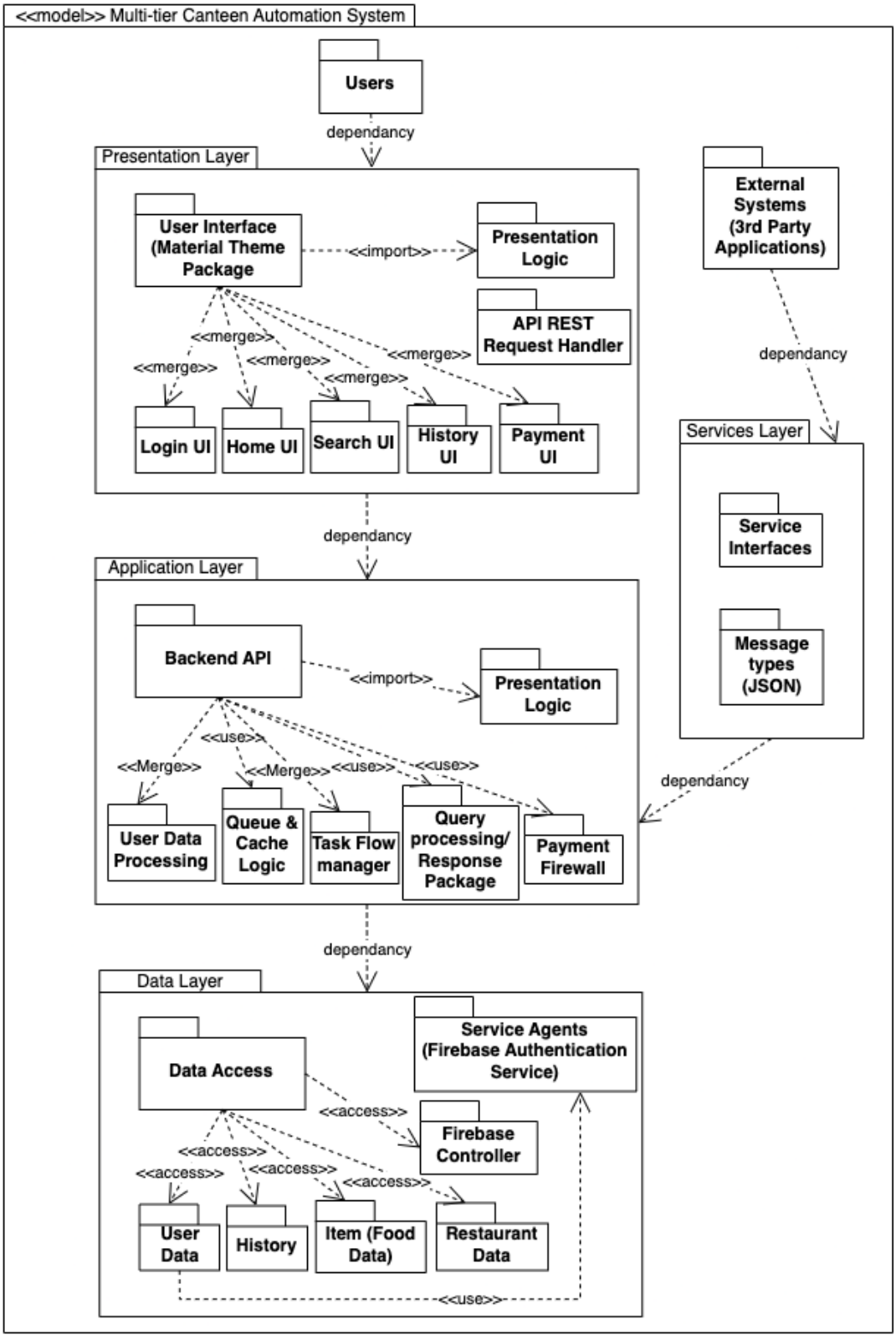
Services Layer: All the third party integration and other services that are already existing in the market, is integrated through the services layer and it is dependent on the application/business layer.

The multi-tier architecture provides separation of concerns, which allows for easier maintenance and scalability of the application. Each layer can be developed and maintained independently, and changes made in one layer do not affect the other layers. This is important as it allows every developer in the team to develop separately, the frontend, backend and database. This architecture also improves the security of the application by allowing for access control at each layer.



(Image from Wikipedia: describes how the 3-tier architecture works! We have multi-layered architecture, as more layers are added as the application gets complex)

5.2 Subsystems and package diagram


<<model>> Multi-tier Canteen Automation System
Users
dependancy
Presentation Layer
User Interface (Material Theme Package
<<import>>
Presentation Logic
API REST Request Handler
<<merge>>
<<merge>>
<<merge>>
<<merge>>
<<merge>>
Login UI
Home UI
Search UI
History UI
Payment UI
External Systems (3rd Party Applications)
dependancy
Services Layer
Service Interfaces
Message types (JSON)
dependancy
dependancy
Application Layer
Backend API
<<import>>
Presentation Logic
<<use>>
<<Merge>>
<<Merge>>
<<use>>
<<use>>
User Data Processing
Queue & Cache Logic
Task Flow manager
Query processing/ Response Package
Payment Firewall
dependancy
Data Layer
Data Access
Service Agents (Firebase Authentication Service)
<<access>>
Firebase Controller
<<access>>
<<access>>
<<access>>
<<access>>
User Data
History
Item (Food Data)
Restaurant Data
<<use>>