FeedMe Software Engineer Take Home Assignment

Documentation Report for Automated Cooking Bot Order Controller

1. Introduction

This document provides a comprehensive overview of the order controller system designed for the automated cooking bots at McDonald's. It details the user stories, state diagram, and the functionality of the system, ensuring a clear understanding of its architecture and operation.

2. System Overview

The automated cooking bot system aims to enhance McDonald's operational efficiency by reducing workforce reliance during COVID-19. This report outlines how the order controller manages order processing, prioritizes VIP customers, and handles bot availability.

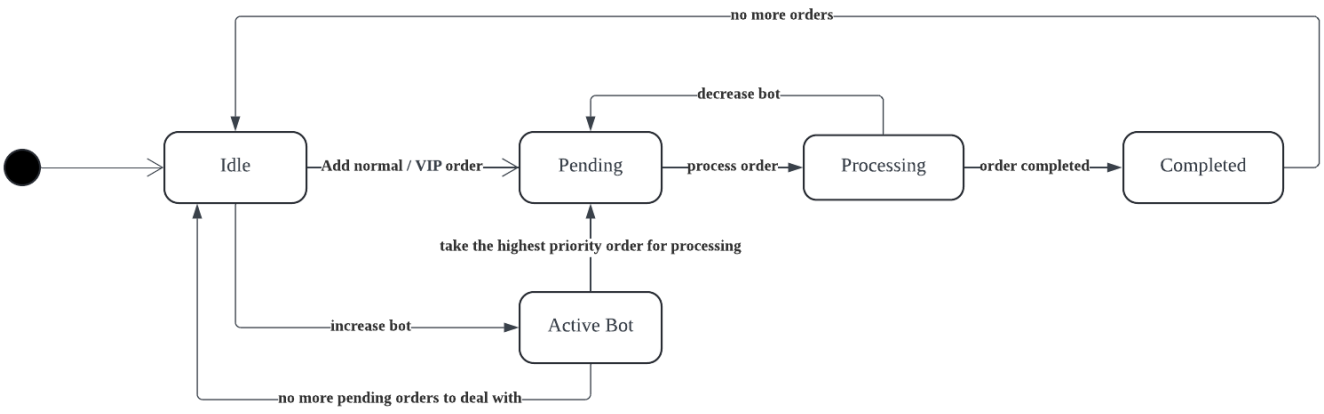
3. User Stories

The system is designed based on the following user stories:

- **Normal Customer**: After submitting an order, the customer wishes to see it in the "PENDING" area until it is completed.
- **VIP Member**: A VIP customer expects their order to be processed before normal orders but queued behind other VIP orders if applicable.
- **Manager**: A manager wants the ability to increase or decrease the number of cooking bots, affecting order processing immediately based on the current state.
- **McDonald's Bot**: As a McDonald's cooking bot, I can only pick up and process one order at a time, and each order requires 10 seconds to complete the process.

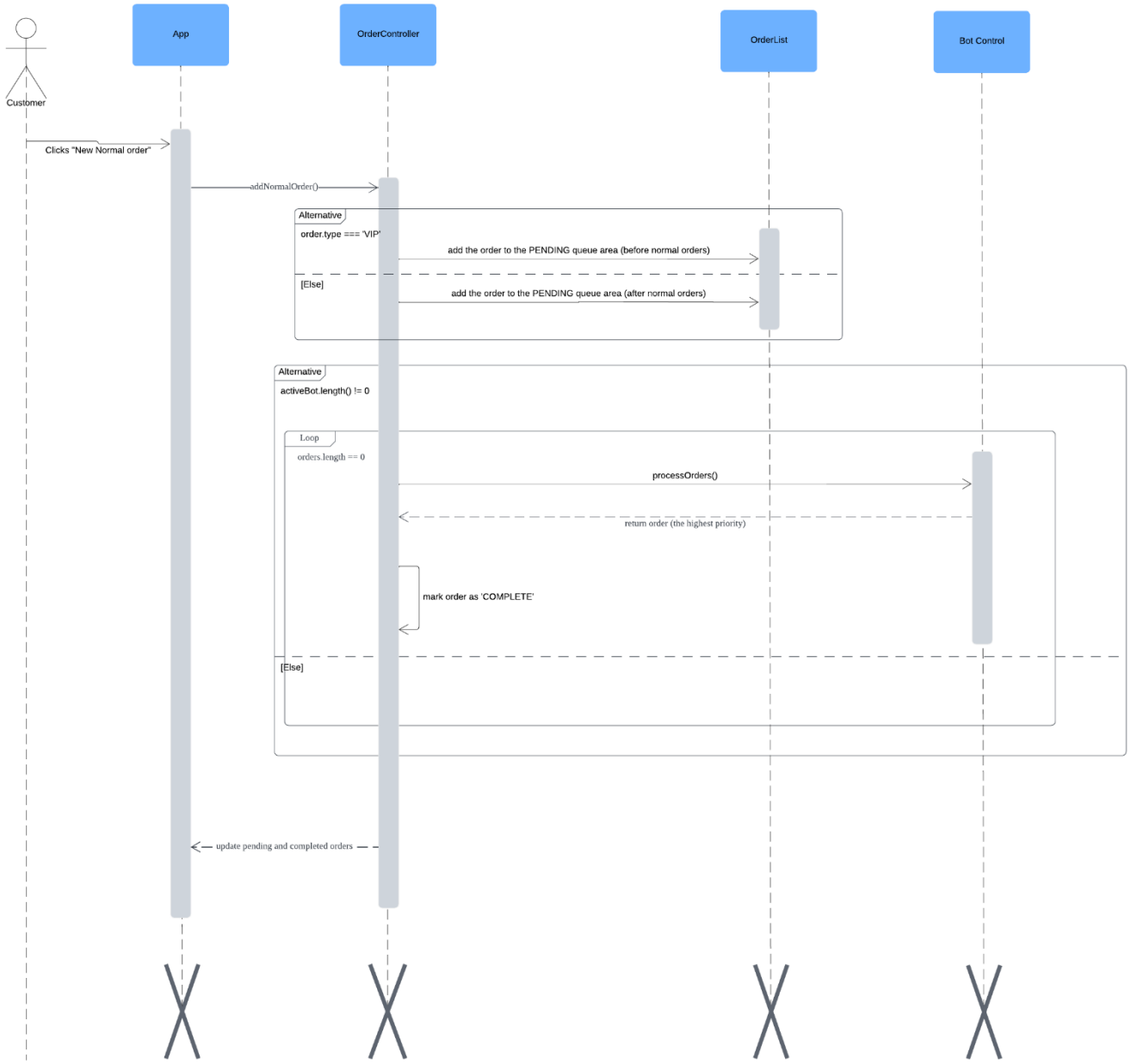
4. State Diagram

State Diagram Representation



The state diagram outlines the flow of orders from submission to completion. Orders enter the **Pending** state upon submission. Normal orders wait in the queue, while VIP orders are prioritized. When a bot is added, it processes an order for 10 seconds before marking it as **Complete**. If a bot is removed while processing, the current order returns to **Pending**. This design ensures efficient management of order processing and prioritization of customer types.

5. Sequence Diagram



This sequence diagram illustrates the interactions between the customer, application components, and order management system when a new order is placed in the McDonald's Order Processing System. It starts with the customer clicking the "New Normal Order" button, which triggers the `addNormalOrder()` method in the 'OrderController'. Depending on whether the order is VIP or normal, the order is added to the 'PENDING' queue accordingly. The diagram shows how the system handles active bots and processes orders. The 'processOrders()' method is called to manage the order completion, marking the highest priority order as 'COMPLETE' and updating the order lists for both pending and completed orders in the application. This diagram helps to visualize the flow of order processing, highlighting the roles of different components and their interactions in managing customer orders efficiently.

6. Implementation Details:

The McDonald's Order Processing System is built using **React** as the front-end framework for handling user interactions and updating the UI, along with **JavaScript** for logic and state management. The system follows a modular structure with different components to handle various aspects of the application like order management, bot control, and order processing.

1. **Component-Based Architecture:**

- The application is divided into reusable components to keep the code clean and manageable:
    - App: The main component that ties everything together.
    - OrderList: Handles the display of both pending and completed orders.
    - BotControl: Provides control for adding or removing bots.
    - OrderItem: Displays individual order details (ID, type, status).

2. **State Management:**

- **React Hooks** are used to manage the state across the application, especially for orders and bots. The useOrderController() hook encapsulates the core logic of the application, including:
    - Handling the addition of normal and VIP orders.
    - Managing bots (adding/removing).
    - Processing orders based on priority (VIPs first, then normal orders).

3. **Order Queueing and Processing:**

- Orders are divided into **normal** and **VIP** types. VIP orders take precedence over normal orders in the processing queue.
- When a bot is added, it starts processing pending orders one by one. Each order takes **10 seconds** to complete. Once completed, the order status changes from "PENDING" to "COMPLETE."
- Bots can only process one order at a time, ensuring efficient workflow management.

4. **Order Priority Handling:**

- VIP orders are inserted at the front of the pending queue, but behind any other existing VIP orders. Normal orders are appended to the end of the queue.

5. **Bot Management:**

- The system allows the dynamic addition and removal of bots:
    - When a new bot is added, it immediately begins processing orders in the pending queue.
    - If a bot is removed while processing an order, the current order is sent back to the pending queue for another bot to pick up.

6. **CSS & Styling:**

- Custom styles are applied using CSS to create a clean user interface for displaying orders and controlling the bots.

7. Tech Stack:

1. React:

- **React** is the core front-end framework for this project, chosen for its component-based architecture and efficient state management using hooks. It enables the smooth and dynamic interaction needed for real-time order processing and updates.

2. CSS:

- **CSS** is used for styling the application. Each component is styled with custom classes for visual clarity and distinction between pending and completed orders.

3. JavaScript (ES6+):

- JavaScript serves as the primary language for implementing the business logic. Modern ES6+ features such as arrow functions, template literals, and array methods are used to handle data manipulation and user interactions.