**Technologies Used:**

* React

**Technologies Approach:**

1. To build the system in-memory, all the states are being built on the top-level of the system (App.js) and being updated across the system for different list.
2. To simulate the order takes 10 seconds to complete processing, setTimeout is being used to update the bot status asynchronously.

**Concerns:**

1. There were two approaches to add the order into the queue:
   1. **Use a single array list:** The system needs to track which is the last VIP order, then the next VIP order will be added into the position after it in the array. **Chose this due to the easier managing of a single list**  
      **Concern:**
      1. Error-prone and hard to manage due to the complexity of the implementation.
      2. Insertion to the array is longer since the system need to shift the elements in the array to insert new VIP order
   2. **Use two array list for two Order Type and combine into one**: Easier to implement, test, check and fix if there is any error for either Order Types.   
      **Concern:** 
      1. Memory Inefficient.
      2. The runtime may be longer since the concatenation of two arrays will happen multiple times if it is being accessed frequently.
      3. When delete the order for processing, I need to delete the correct id in the combined order list and normal/vip order list. Otherwise, the order is not deleted properly.
2. setTimeout in React may not be the best solution to simulate the order processing need to take 10 seconds to complete. Reason => When setTimeout is triggered, it will save the data at that moment. So, there will be issues to update the bot status and completed order in real time.
   1. To fix this, I must implement few more states like processingOrder & removingOrder states. processingOrder is to keep track of which order is currently in process, removingOrder is to keep track of which order is to be removed from completedOrder since the bot processing it has been removed mid-way.
3. State Management Complexity. Since many useState has been used across the application, it will be hard to maintain and keep track.
4. Side effects in using UseEffect depending on the bots, pendingOrders & processingOrders. This may lead to unnecessary re-render execution.

**Optimisation Implemented:**

1. I have broken down several components into smaller part so that each component is responsible for their own job. This makes the code more reusable, easier to test and maintainability.
2. Minimised the re-render of useEffect depending on several states, this can help the performance of the website.
3. I have created utility functions (e.g. addOrder, addBot, removeBot etc) for handling specific tasks like adding order, adding bot, removing bot etc. With this implementation, the utility functions can be further reuse and easier to maintain.
4. OrderList component has been reused for both PENDING order and COMPLETED order list, since both are only displaying their queue of orders.
5. AddOrder utility function has also been reused for both Add Normal Order and Add VIP order to keep cleaner code.