Product Backlog

Core Functional Requirements:

- **Customer Queuing System -** Implementing a single queue for customers.
- **Order Processing -** Each server processes orders sequentially.
- **Order Loading -** Read existing orders from csv file and add new orders to the queue.
- **Graphical User Interface (GUI) -** Display:
    - Customers in queue.
    - Servers processing orders.
    - Real-time order status.
- **Logging Events System -** Writing log entries for each event.
- **Closing system when queue empties**- Close shop and generate a summary report when the queue is empty.
- **Multi-Threading -** Implementing separate threads for Customer queue manager and handling orders.
- **Synchronization -** Ensuring threads runtime conditions without using built-in Java thread-safe collections.
- **Singleton Pattern -** Using Singleton for logging.
- **Observer Pattern -** Implementing Observer pattern for GUI updates.
- **MVC Architecture -** Structuring GUI in accordance to MVC pattern.
- **Agile Methods and Version Control** – Using Agilr methodologies for project completion such as iterations, pair programming, standups, and GitLab for version control.
- **JAR Export** – Exporting final app as a JAR executable file.

Sprint Backlog

Sprint 1: Core Queue and Threading Setup

Duration: March 16, 2025 – March 22, 2025 (7 days)
Sprint Goal: Set up the basic queue system and multi-threading functions.

User Story	**Task**	**Assignee**
System, I want to read existing orders so that I can increment the queue in order.	Implement extracting data from file (orders.csv) and using thread to gradually add orders to a data structure such as a LinkedList queue.	Ima and Shreyash
Server, I want to process orders one at a time so that customers are served in order.	Create order processing with two server (Employee) threads (Server 1, Server 2) pulling orders from the queue (OrderList)	Junaid and Ima
System, I want to log all actions so that I can track system events.	Implement Log event handler using Singleton method to record events and write to csv upon exit.	Aedh and Adil

Sprint 2: GUI and Design Patterns

Duration: March 23, 2025 – March 30, 2025 (7 days)
Sprint Goal: Implementing GUI with MVC and Observer patterns and applying thread synchronization.

User Story	**Task**	**Assignee**
User, I want to see the order queue and servers so that I can keep track of my order.	Implement GUI showing queued customers and their orders along with order status	Ima
Developer, I want the GUI to follow MVC methodology so it's maintainable.	Structure GUI with MVC: Model (queue/server state), View (JFrame panels), Controller (queue/server actions).	Junaid and Ima
System, I want to use Observer so that updates are automatically shown.	Implement Observer pattern for real time GUI updates.	Adil and Aedh
Developer, I want my threads to work correctly without interfering with each other.	Implement synchronization and avoid any runtime race conditions .	Shreyash and Junaid

Sprint 3: Completion, Extension, and Finalization

Duration: March 31, 2025 – April 6, 2025 (7 days)
Sprint Goal: Complete the application, add extensions, export as JAR, and generate the report.

User Story	**Task**	**Assignee**
System, I want to close the application when the queue is empty.	Implement logic to check queue size, generate a report and exit.	Junaid and Ima
User, I want a good looking experience, it should be visually clear.	Improve GUI styling.	Adil and Aedh
Developer, I want the app as a JAR for easy execution.	Export the application to a JAR file and test execution.	Ima and Shreyash
Team, I want to document our work so we meet coursework requirements.	Write Stage 2 report.	Ima, Junaid, Adil, Aedh and Shreyash

Sprint Planning

Each sprint starts with a planning meeting where we:

- Review the Sprint Backlog.

- Assign tasks to developers (suggested: Ima: threading/tests, Junaid: GUI/log, Adil: MVC/report, Shreyash: queue/report, Aedh: GUI/integration).
- Discuss risks (e.g., thread deadlocks, report time,etc).

Tools to use:

- **Scrum:** Trello
- **GitLab**: Version control with branches (e.g., sprint1-queue, sprint2-gui, sprint3-final).
- **Sprint 1 Planning (March 15, 1 hour):** Define tasks, set up branches, agree on meeting time.
- **Sprint 2 Planning (March 22, 1 hour):** Review Sprint 1 and assign GUI/pattern tasks.
- **Sprint 3 Planning (March 30, 1 hour):** Finalize application and finalize report.

Sprint Retrospective

Sprint 1 Retrospective (March 22, 1 hour)

- **What went well?**
  The core queuing system was successfully implemented using a LinkedList, and multi-threading for order loading and processing worked as planned. The team collaborated effectively to set up the initial structure, and the Singleton logging system was functional early, providing useful event tracking from the start.
- **What didn't go well?**
  Initial thread implementation led to some race conditions due to the unclear synchronization requirements. Task allocation felt uneven, with some team members working on multiple complex tasks like order loading and server threading simultaneously.
- **What can be improved?**
  Define clearer synchronization expectations during planning to avoid runtime issues. Distribute threading tasks more evenly across the team to prevent one person working much more than the others. Schedule a mid-sprint check-in to get to know about the current issues earlier.

Sprint 2 Retrospective (March 30, 1 hour)

- **What went well?**
  The GUI was implemented successfully, displaying the queue and server status in real-time. The Observer pattern integration worked great for GUI updates, and thread synchronization was redone which resolved earlier race condition problems. Pair programming between Junaid and Ima improved code quality for the MVC structure.
- **What didn't go well?**
  GUI styling took longer than expected, delaying testing of real-time updates. Some team members had less involvement due to a focus on GUI and heavy coding tasks, leading to uneven contribution this sprint.
- **What can be improved?**
  Allocate GUI styling tasks earlier in the sprint to leave enough time for testing. Involve

all team members in GUI related discussions to properly distribute workload and leverage everyone’s skills. Document Observer pattern decisions as they are made to ease future debugging.

Sprint 3 Retrospective (April 6, 1 hour)

- **What went well?**
  The application was finalized with all core features working, including the shutdown when queue is empty and summary report generation. Exporting to a JAR file was smooth, and GUI improvements enhanced the user experience. The team collaborated well on the Stage 2 report, finishing it ahead of schedule.
- **What didn’t go well?**
  Last minute tweaks to the report layout caused some difficulty, as documentation was left mostly to the end. Testing the JAR file revealed a small configuration issue that required quick fixes, taking out of our final review time.
- **What can be improved?**
  Start report drafting incrementally during earlier sprints to avoid a final rush. Test the JAR export earlier to catch configuration issues sooner.

Summary of Scrum Plan

Phase	Task	Duration
Sprint 1	Core Queue and Threading	March 16–22 (7 days)
Sprint 2	GUI and Patterns	March 23–30 (7 days)
Sprint 3	Completion and Finalization	March 31–April 6 (7 days)
Retrospective	Review sprint & improve	1 hour per sprint (March 15, 22, 30)
Finalization	Final testing & submission prep	April 6-7 (2 days)