Brew & Bite Café System

ICS 372 – Group Project

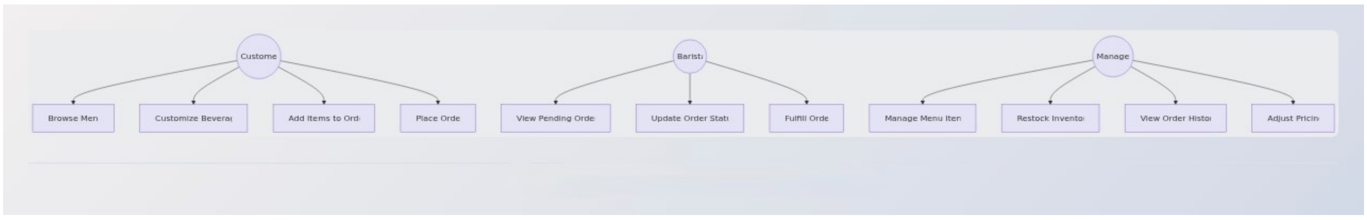
Semester: Fall 2025

Overview

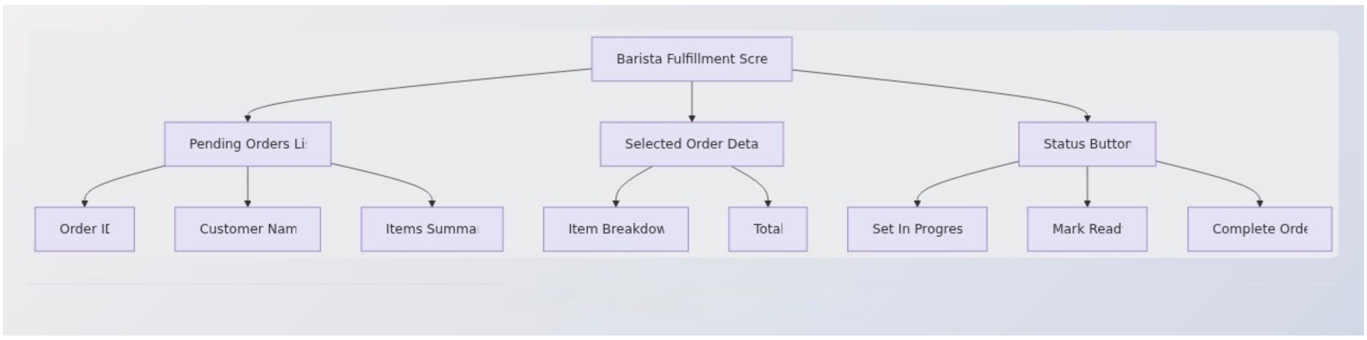
The Brew & Bite Café System is a JavaFX application designed to simulate a multi-role café environment, supporting Customer Ordering, Barista Fulfillment, and Manager Operations. The system follows an MVC architecture, incorporates object-oriented design principles, and uses JSON-based persistence to store menu, inventory, and order data across runs.

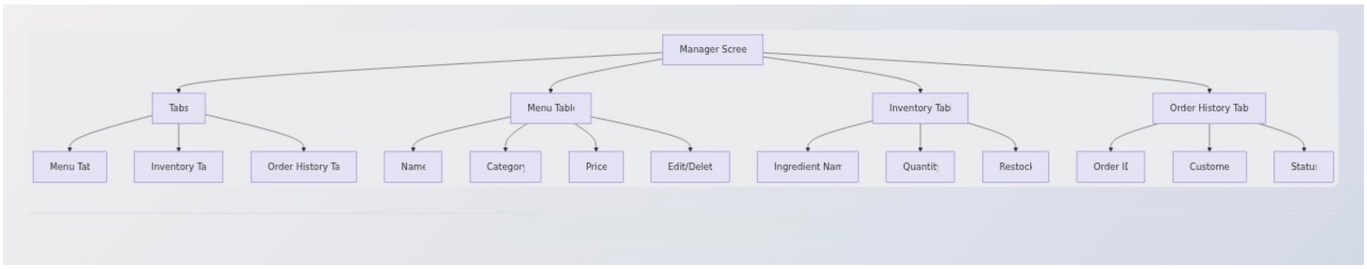
This project fulfills all functional and non-functional requirements described in the assignment specifications, including inventory tracking, menu customization, order management, Observer/Factory Method design patterns, and a role-based interface.

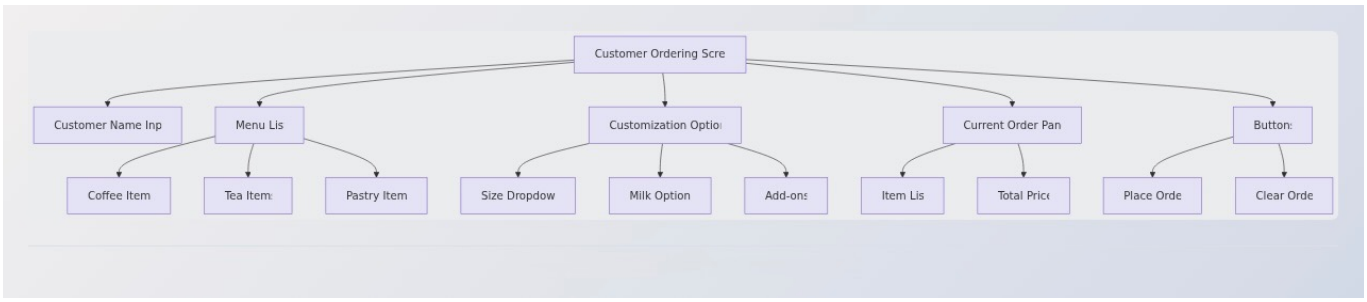
Use Case Diagrams



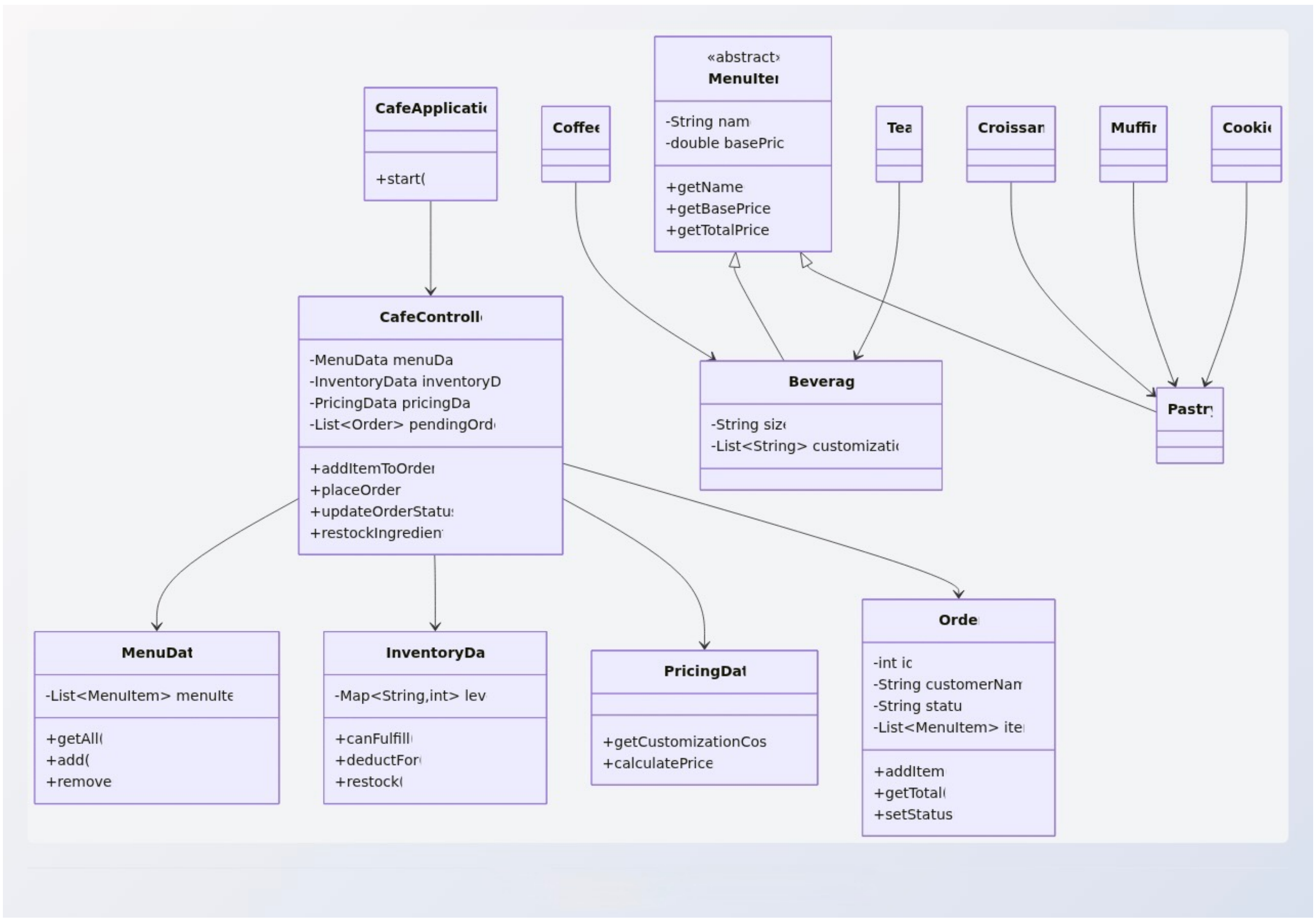
Wireframes







UML Diagram



Sequence Diagram 1: Barista Login

Participants (left → right)

- Barista (actor)
- Login UI (FXML Screen)
- CafeController
- System (Validation)

```
Barista         Login UI        CafeController       System
   |                |                  |                  |
   |---Enter ID--->|                    |                    |
   |---Enter PW--->|                    |                    |
   |---Click Login>|                    |                    |
   |               |---login(data)--->|                    |
   |               |                    |---validate()---->|
   |               |                    |<--result---------|
   |               |<--status---------|                    |
   |               |                    |                    |
   |        opt [login valid]                                |
   |               |---load Barista UI------------------>|
   |               |                    |                    |
   |        opt [login invalid]                              |
   |               |---display error-------------------->|
```

Sequence Diagram 2: Manager Login

Participants

- Manager (actor)
- Login UI
- CafeController
- System

```
Manager        Login UI        CafeController        System
 |               |                    |                    |
 |---Enter ID--->|                    |                    |
 |---Enter PW--->|                    |                    |
 |---Click Login>|                    |                    |
 |               |---login(data)--->|                    |
 |               |                    |---validate()---->|
 |               |                    |<--result---------|
 |               |<--status---------|                    |
 |               |                    |                    |
 |        opt [login valid]                                |
 |               |---load Manager UI------------------>|
 |               |                    |                    |
 |        opt [login invalid]                              |
 |               |---display error-------------------->|
```

Sequence Diagram 3: Customer Order Placement

Participants

- Customer (actor)

- Customer UI
- CafeController
- Order
- Menu

```
Customer     Customer UI     CafeController     Menu        Order
|              |                |              |            |
|---View Menu->|                |              |            |
|              |---getMenu()--->|              |            |
|              |                |---fetch----->|            |
|              |                |<--menu-------|            |
|              |<--display----- |              |            |
|              |                |              |            |
| loop [add items]              |              |            |
|---Select Item>|               |              |            |
|              |---addItem()--->|              |            |
|              |                |---add-------->|           |
|              |                |              |            |
|---Place Order>|               |              |            |
|              |---submit()---->|              |            |
|              |                |---validate---------------->|
|              |                |<--confirmed----------------|
|              |<--confirmation-|              |            |
```

Sequence Diagrams

Explained

The sequence diagrams were designed to clearly illustrate how responsibilities are delegated among system components to fulfill each use case while adhering to object-oriented design principles. Each diagram follows the structure of a UML sequence diagram used in our textbook, using lifelines to represent participating entities and horizontal messages to show communication over time.

Barista Login and Manger Login

For the Barista Login and Manager Login use cases, the diagrams emphasize separation of concerns by delegating authentication logic to the CafeController rather than embedding it within the user interface components. The login screens are responsible only for capturing user input and forwarding credentials, while the controller performs validation and determines the appropriate system response. The use of opt fragments represents conditional behavior based on the selected user role, and loop fragments model repeated login attempts when invalid credentials are entered. This design reflects the Single Responsibility Principle (SRP) by ensuring that user interface components do not directly manage business logic.

Customer Order Placement

In the Customer Order Placement sequence diagram, responsibilities are similarly distributed across distinct components. The Customer interacts with the ordering interface, which delegates menu retrieval and order processing tasks to the CafeController. A loop fragment is used to represent the repeated selection of menu items, accurately modeling the iterative nature of order construction. Conditional execution (opt) is used to show how the system verifies order validity before confirming the order to the Customer. This approach improves clarity by explicitly modeling decision points and repeated actions within the workflow. Across all sequence diagrams, the controller acts as the central coordinator, mediating interactions between the user interface and underlying system behavior. This design supports low coupling between components and high cohesion within classes, making the system easier to understand, maintain, and extend. By abstracting UI details and focusing on core business logic in the sequence diagrams, the design accurately reflects the implemented MVC architecture and demonstrates effective delegation of responsibilities in accordance with object-oriented design principles.

Features

1. User Roles & Authentication

- **Customer**: No login required; enters name and creates an order.
- **Barista**: Logs in with hardcoded credentials to see and fulfill orders.
- **Manager**: Logs in to manage inventory, menu items, pricing, and restocking.

2. Menu Management

- Handles Beverages (Coffee, Tea) with multiple sizes and customizations.
- Handles Pastries (Croissant, Muffin, Cookie) with variations.
- Manager can add, edit, or remove items dynamically.

3. Inventory Management

- Each item consumes ingredients; orders cannot be placed if ingredients are insufficient.
- Removed or added menu items updates instantly across views through Observer pattern.

4. Customer Ordering Interface

- Select beverages/pastries.
- Customize beverages (size, milk type, syrups, extra shots).
- View cart total, add multiple items, place or clear order.

5. Barista Fulfillment Interface

- View pending orders in FIFO order.
- Update status: Pending → In Progress → Ready → Fulfilled.
- Completed orders logged for manager review.

Architecture

MVC Pattern Implementation

- **Model**: Order, Coffee, Cookie, Croissant, Muffin, Tea,
- **View**: Application, JavaFX FXML files for each role interface. OrderUpdate Notification for updating a ListView.
- **Controller**: CaféController plus supporting controllers for UI flows.

Applied Object-Oriented Principles

- **SRP (Single Responsibility Principle)**:
  Each class encapsulates one responsibility (e.g., `InventoryData`, `MenuData`, `Order`).
- **Observer Pattern (Mandatory)**:
  Used for updating customer ordering details in views in real-time.
- **Inheritance & Composition**:
  Beverages and pastries are used in composition with the Order class; UpdateOrderNotification implements a UpdateOrderStrategy interface using inheritence.

Project Structure

```
src/
├── main/
│ ├── java/
│ │ └── com/example/cafesystem/
│ │ ├── CafeApplication.java
│ │ ├── CafeController.java
│ │ ├── Order.java
│ │ ├── InventoryData.java
│ │ ├── MenuData.java
│ │ ├── PricingData.java
│ │ ├── Coffee.java
│ │ ├── Tea.java
│ │ ├── Croissant.java
│ │ ├── Muffin.java
│ │ └── Cookie.java
│ └── resources/
│ └── com/example/cafesystem/
│ └── *.fxml (All UI screens)
│
├── gradle/
│
├── build.gradle.kts
└── settings.gradle.kts
```

Running the Application

Prerequisites

- Java **21 or newer**
- Gradle wrapper included in project

Run Command

```
./gradlew run
```

Design Artifacts (Submitted Separately)

Our design document includes:

- UML Use Case Diagram
- Wireframes for all major screens
- Conceptual-to-Software Class Mapping
- High-Level UML Class Diagram
- Multiple Sequence Diagrams
- MVC Layering Explanation
- Detailed application of Observer pattern

Group Reflection – Brew & Bite Café System

Challenges and Solutions

One of the biggest challenges our team faced was coordinating how the customer, barista, and manager interfaces would interact with shared data. Ensuring that updates to orders, inventory levels, and menu changes stayed consistent across the entire system required careful planning and several redesigns. Setting up JSON persistence alongside JavaFX also presented difficulties.

Beyond the technical work, we also encountered challenges related to the overall difficulty of the course and external complications that affected our availability and workflow. Balancing this project with other demanding coursework required extra coordination, and unexpected personal or schedule-related disruptions sometimes slowed our progress. We addressed these obstacles by increasing communication, adjusting timelines where possible, and supporting each other so the workload stayed manageable.

Learning Insights

This project pushed us to apply object-oriented design principles in a realistic scenario. Implementing the Observer pattern helped us understand how design patterns function in practice, not just in theory. Using MVC reinforced the importance of separating domain logic, user interfaces, and controllers to create maintainable software. We also gained experience in version control, collaborative development, and managing multiple features that depend on one another. These insights deepened our understanding of Chapters 5 and 6 and demonstrated how design decisions evolve as a project grows.

Improvements & Future Work

With more time, we would refine the user experience by improving interface transitions and adding more detailed feedback throughout the ordering process. We identified opportunities to enhance the manager interface with richer reporting tools and deeper insights into inventory usage. Expanding beverage customization options, adding customer order history, and implementing more comprehensive error handling are also future enhancements we discussed. Automated testing would further improve the system’s stability and long-term maintainability.

Team Dynamics & Project Management

Our team used GitHub Projects to assign tasks, track progress, and document work throughout the development cycle. Clear communication helped us resolve issues efficiently and kept everyone aligned on design decisions. Even when schedules and responsibilities shifted due to external complications, our workflow allowed us to integrate finished features without major conflicts. Version control played a crucial role in staying organized and ensuring that everyone worked from the same codebase.

Individual Contributions

- **Javier Frias (Team Leader):** Led the overall implementation of the system and coordinated the integration of customer ordering, barista workflows, and manager operations.
- **Anas Hussein:** Created UML diagrams including use case, sequence, class, and architectural diagrams. Assisted in wireframing and documenting conceptual-to-software class transitions.
- **Selemon Taddese:** Produced UML diagrams, contributed to design documentation, and supported the structural layout of system interactions.
- **Saaid Husein:** Managed the GitHub repository, handled version control and project organization, wrote the README file, and authored the group reflection document.

Together, our contributions resulted in a complete and functional Brew & Bite Café System. This project strengthened our technical skills, collaboration abilities, and our understanding of designing and implementing software within a team environment.