FlowCart
Delivery Management System

INFO-5100 : FINAL PROJECT

2024 December 10

Team: Natasha Kiara,
Mudra Pandya,
Xiaotong Yong

Problem

Small businesses often struggle with efficiently managing deliveries due to the lack of a centralized system and easy access to reliable delivery personnel. This leads to:

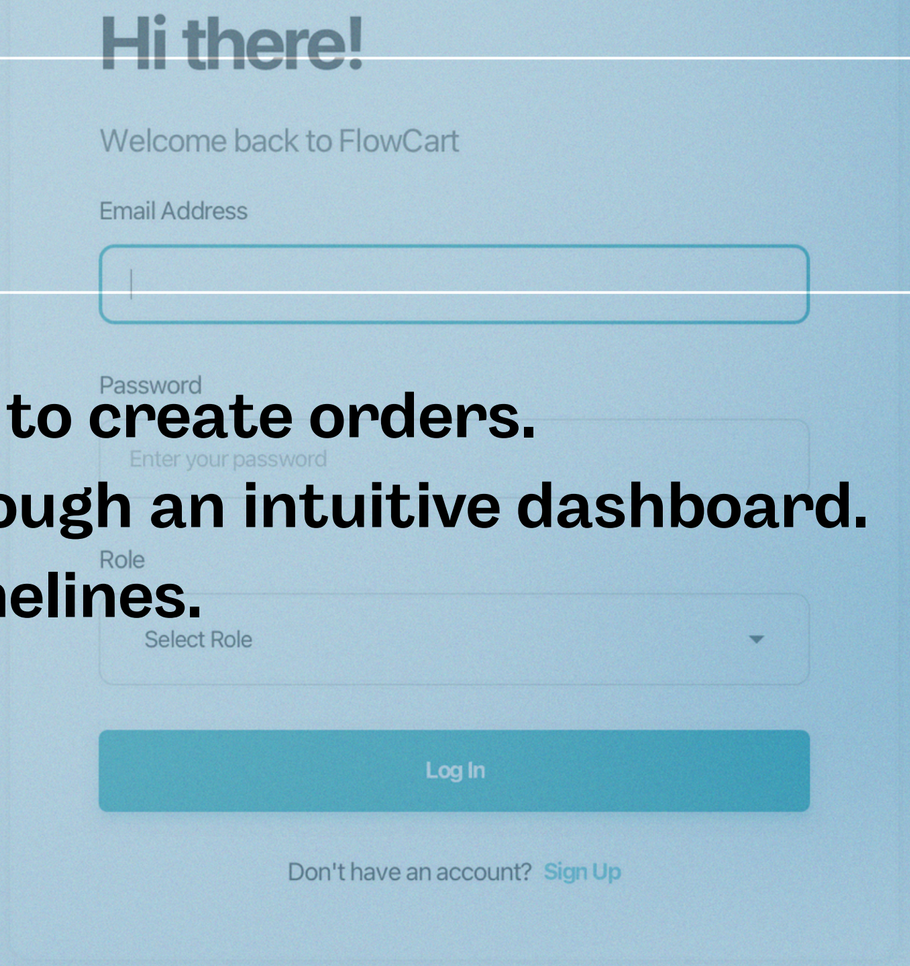
- Inefficient order tracking and delivery management, impacting customer satisfaction.
- Delays caused by manual processes and miscommunication between stores and delivery agents.
- Missed opportunities to streamline operations and scale their delivery capabilities.

Objectives

- Build a centralized delivery management system.
- Enable seamless communication between stores and delivery agents.
- Provide real-time updates on order statuses.
- Improve operational efficiency.

Solution

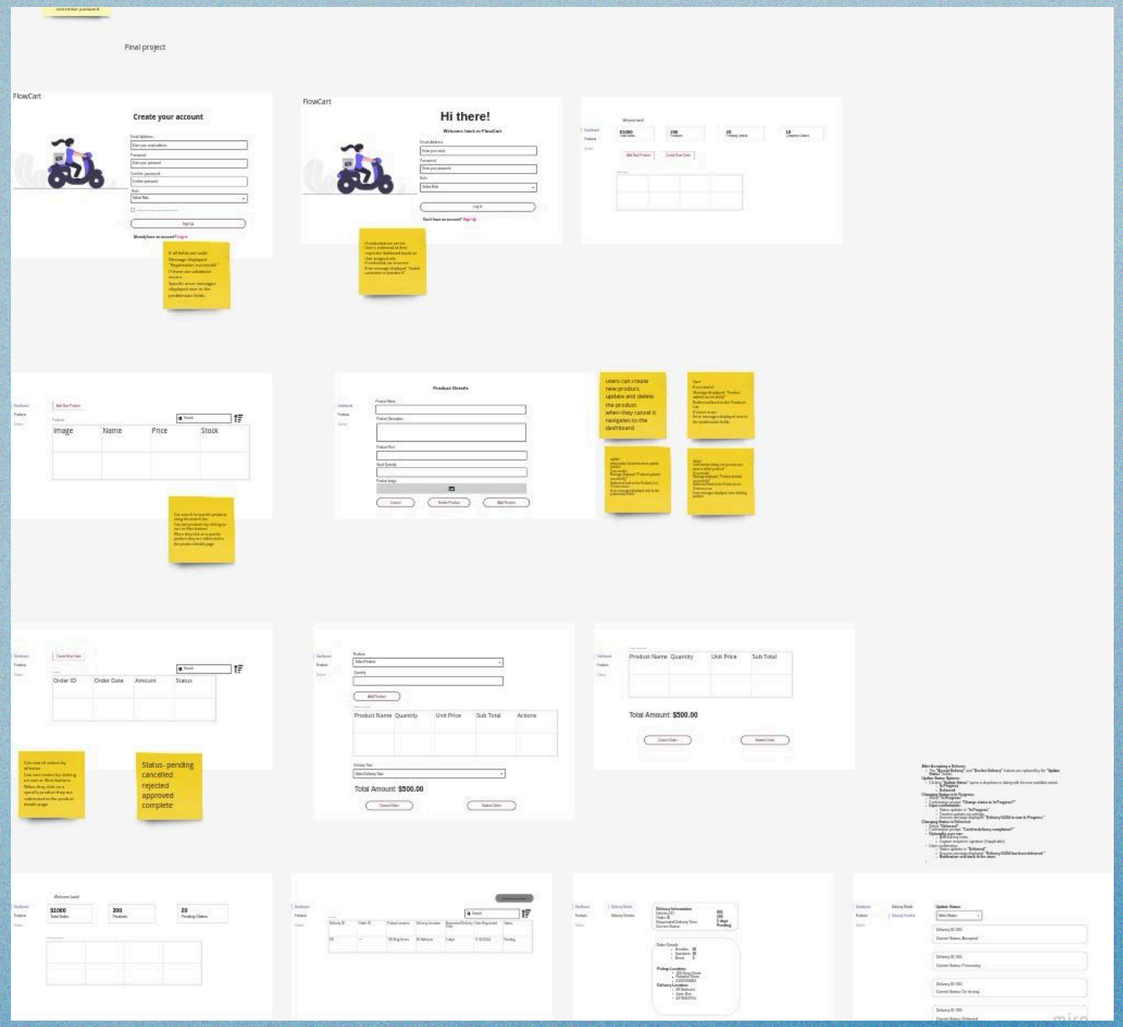
- A web-based platform enabling stores to create orders.
- Delivery agents manage deliveries through an intuitive dashboard.
- Real-time updates on statuses and timelines.



Features

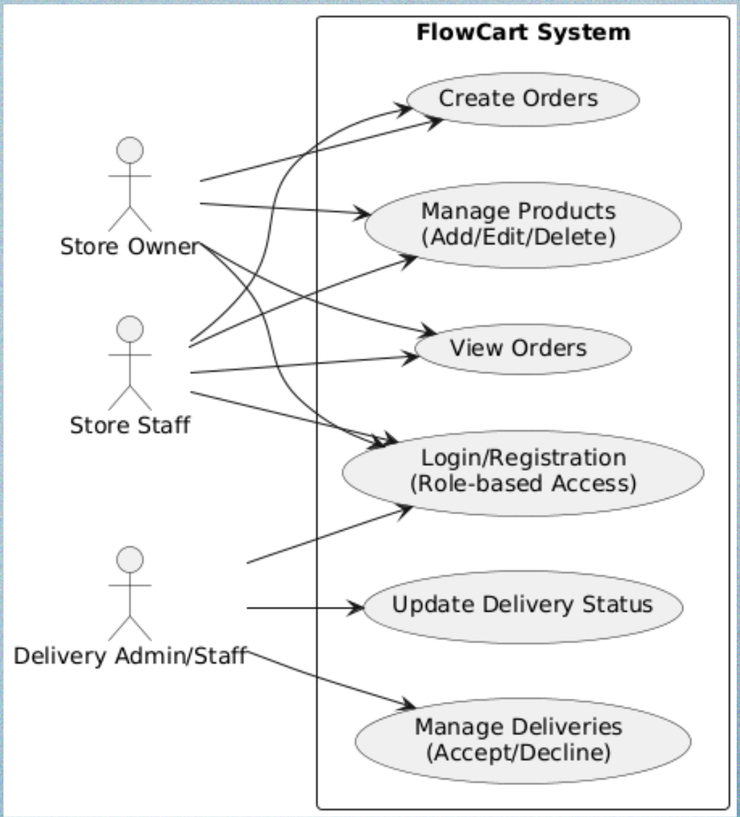
- **Store Dashboard: Manage orders and track statuses.**
- **Delivery Dashboard: Accept/decline deliveries, view timelines.**
- **Reporting: Generate Excel reports for performance tracking.**
- **Role-Based Access: Separate views for store and delivery personnel.**

Mockup



- Registration: User account creation with role selection.
- Login: Secure login with role-based dashboards.
- Store Dashboard: Manage products and orders.
- Product List: View, add, edit, or delete products.
- Order Management: Track and update orders with status changes.

Use Case Diagram



Actors:

Store Owner: Manages products and creates orders.

Store Staff: Views products and order details.

Delivery Admin/Staff: Handles delivery requests and updates statuses.

Key Use Cases:

Login/Registration: Users log in and access features based on their role.

Product Management: Add, edit, or delete products (Store Owner).

Order Management: Create and view orders (Store Owner/Staff).

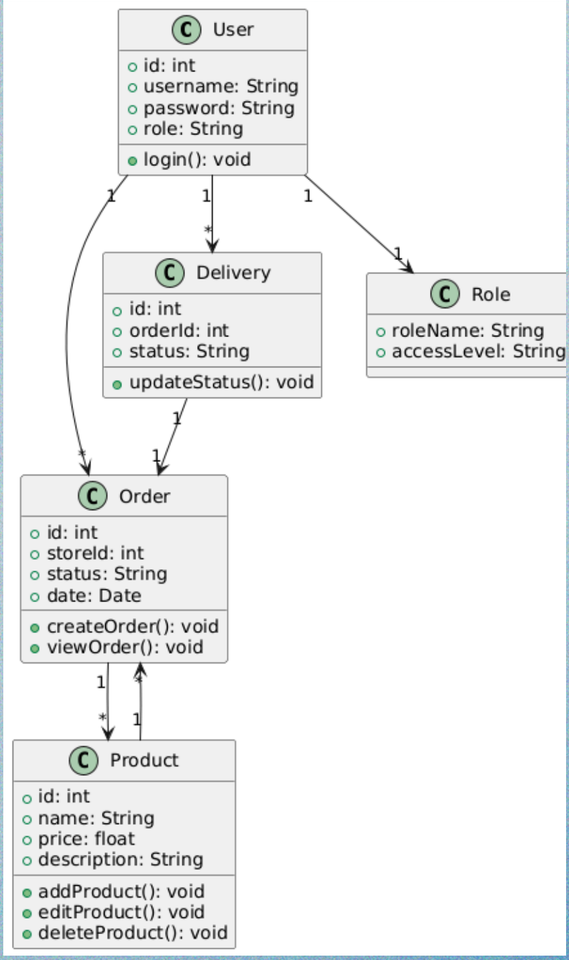
Delivery Management: Accept/decline requests and update delivery statuses (Delivery Admin/Staff).

Purpose:

Clearly defines role-based responsibilities.

Simplifies system navigation and task execution.

UML Class Diagram



Classes:

- User: Manages login credentials and role-based actions.
- Product: Handles product details with add, edit, and delete methods.
- Order: Tracks orders with status, date, and associated products.

Delivery: Manages delivery requests and status updates.

Relationships:

- Users interact with orders and deliveries.
- Orders are linked to products.

Database Design & MySQL Implementation

Database Design Overview

- **Schema:** The flowcart database contains five core tables to manage system operations effectively:
1. Users: Stores user credentials and roles (e.g., admin, staff).
2. Products: Manages product information (name, price, stock).
3. Orders: Tracks customer orders, including status and timestamps.
4. Order_Items: Links products to specific orders with quantities.
5. Deliveries: Handles delivery status, assignments, and logs.
- **Relationships:** Tables are connected via foreign keys to ensure data integrity.
- **Example:** orders reference users for the customer and order_items for product details.

MySQL Implementation

- The MySQL Database for FlowCart is fully functional with five key tables: users, products, orders, order_items, and deliveries.
- Query Execution: Basic SQL commands such as USE and SHOW TABLES confirm the schema and table structure.
- Key Features:
- Each table is designed to store and manage specific data efficiently, such as user credentials, product details, and delivery statuses.
- The database is optimized with primary and foreign keys to maintain data relationships and integrity.

System Architecture

Frontend: JavaFX (FXML for views).

Backend: MVC Architecture with DBUtils and Service Layers.

Database: MySQL for data persistence.

Workflow/Process

- Store Owner: Creates orders.
- Delivery Agent: Accepts/rejects deliveries and updates status.
- Reporting: Generates performance reports.

Implementation Highlights

- Singleton pattern for DBUtils.
- Reusable Service Layer for database operations.
- Modular design for scalability.

The End

Special thanks to Professor Zheng Zheng and TA Tapaswi for their guidance and support throughout this project.