A black and white logo

AI-generated content may be incorrect.

**Computer Science and Information Technology Department College of Engineering**

**CSC-202 SPRING 2024-25**

|  |  |
| --- | --- |
| NAME | ID |
|  |  |
|  |  |
|  |  |

**Prof. Adel Khelifi**

**Submission Date: 06/06/2025**

Contents

[1. Introduction 3](#_Toc200142910)

[2. System Overview 4](#_Toc200142920)

[3. Main Features 4](#_Toc200142921)

[3.1 User Management 4](#_Toc200142922)

[3.2 Product Search and Display 4](#_Toc200142923)

[3.3 Customer Portal 5](#_Toc200142924)

[3.4 Farmer Dashboard 6](#_Toc200142925)

[3.5 Delivery System 6](#_Toc200142926)

[4. Data Management 7](#_Toc200142927)

[5. Exception Handling 7](#_Toc200142928)

[6. Core Logic and Console-Based Components 8](#_Toc200142929)

[7. Graphical User Interface (GUI) 12](#_Toc200142930)

[8. Code Structure 14](#_Toc200142931)

[9. Challenges Faced 15](#_Toc200142932)

[10. Testing and Validation 15](#_Toc200142933)

[11. Sample UML 16](#_Toc200142934)

[12. Conclusion 19](#_Toc200142935)

# 1. Introduction

# Our group worked on creating a Farm Produce Delivery System as a project on farmers and customers. In many places, it’s hard for people to get fresh fruits and vegetables directly from local farmers. At the same time, farmers also find it difficult to reach customers. So, we decided to build a system that will help solve this problem.

# The goal of our project is to build a desktop application using Java. This app lets farmers share what they are growing and allows customers to subscribe and receive fresh produce. There are two main types of users:

# Farmers, who can add produce, see who subscribed, and manage deliveries.

# Customers, who can browse available produce, subscribe for deliveries, and track their orders.

# Our system has two main parts:

# Java Logic – This is the part of the app that does all the work in the background, like saving data, checking user input, and managing produce and delivery. It doesn’t need any graphics and can work from the command line.

# JavaFX GUI – This is the part of the app that the user sees. It has buttons, tables, and menus so users can easily use the system with a mouse and keyboard.

# We used basic object-oriented programming ideas like classes, inheritance, and error handling to build our system in a clean and organized way. We also used text files to save data so we didn’t need a database, which made things simpler.

# This report explains everything we did in our project, including how the system works, what each part does, how the users interact with it, and what we learned along the way.

# 2. System Overview

The system is designed to cater to two main stakeholders: farmers and customers. Farmers can log in to manage and list available produce, while customers can subscribe to receive fresh products and manage their subscriptions.

The system handles delivery planning, harvest date tracking, and alerting users for invalid actions such as ordering out-of-season produce.

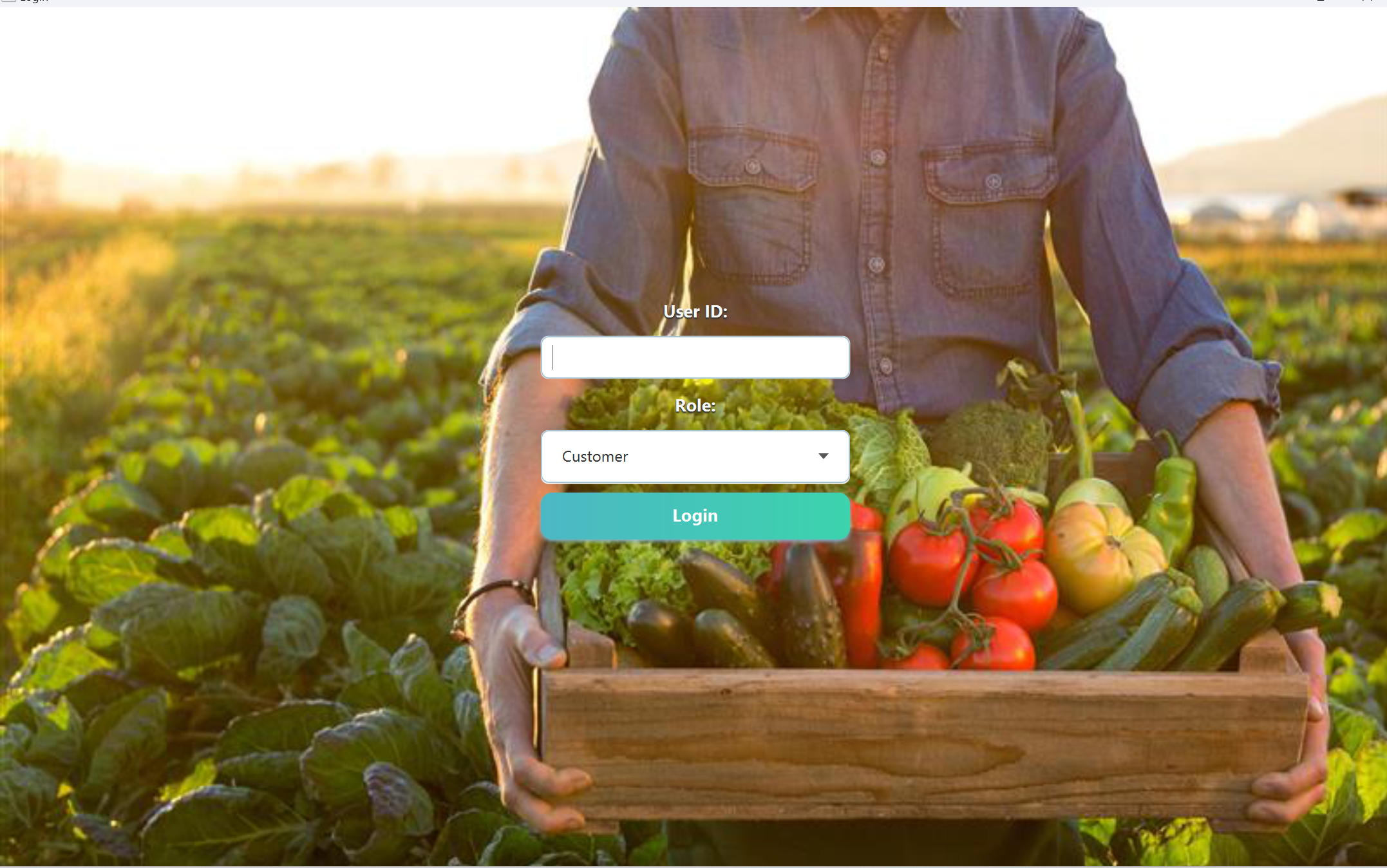
The project consists of multiple Java classes, structured around object-oriented programming principles, and uses plain text files for persistent data storage.

# 3. Main Features

The system includes several key modules that interact to provide a smooth experience for both farmers and customers.

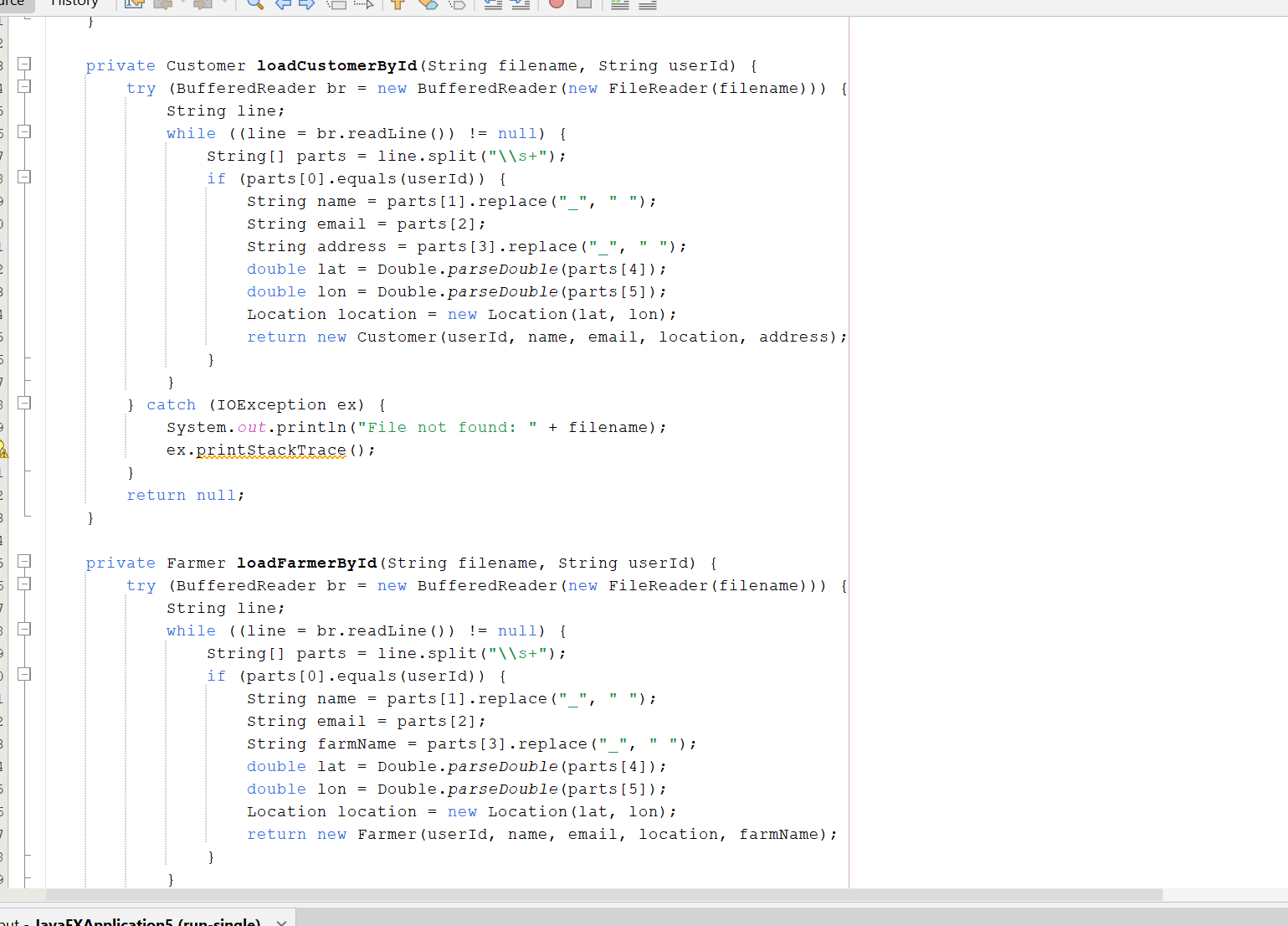
## 3.1 User Management

The `LoginScreen` class provides the login interface for users. The `User` class is a base class that stores shared attributes for `Customer` and `Farmer`.



## 3.2 Product Search and Display

The `ProductSearch`, `ProductSearchEngine`, and `IProductSearch` handle search operations allowing customers to find and subscribe to available produce.



## 3.3 Customer Portal

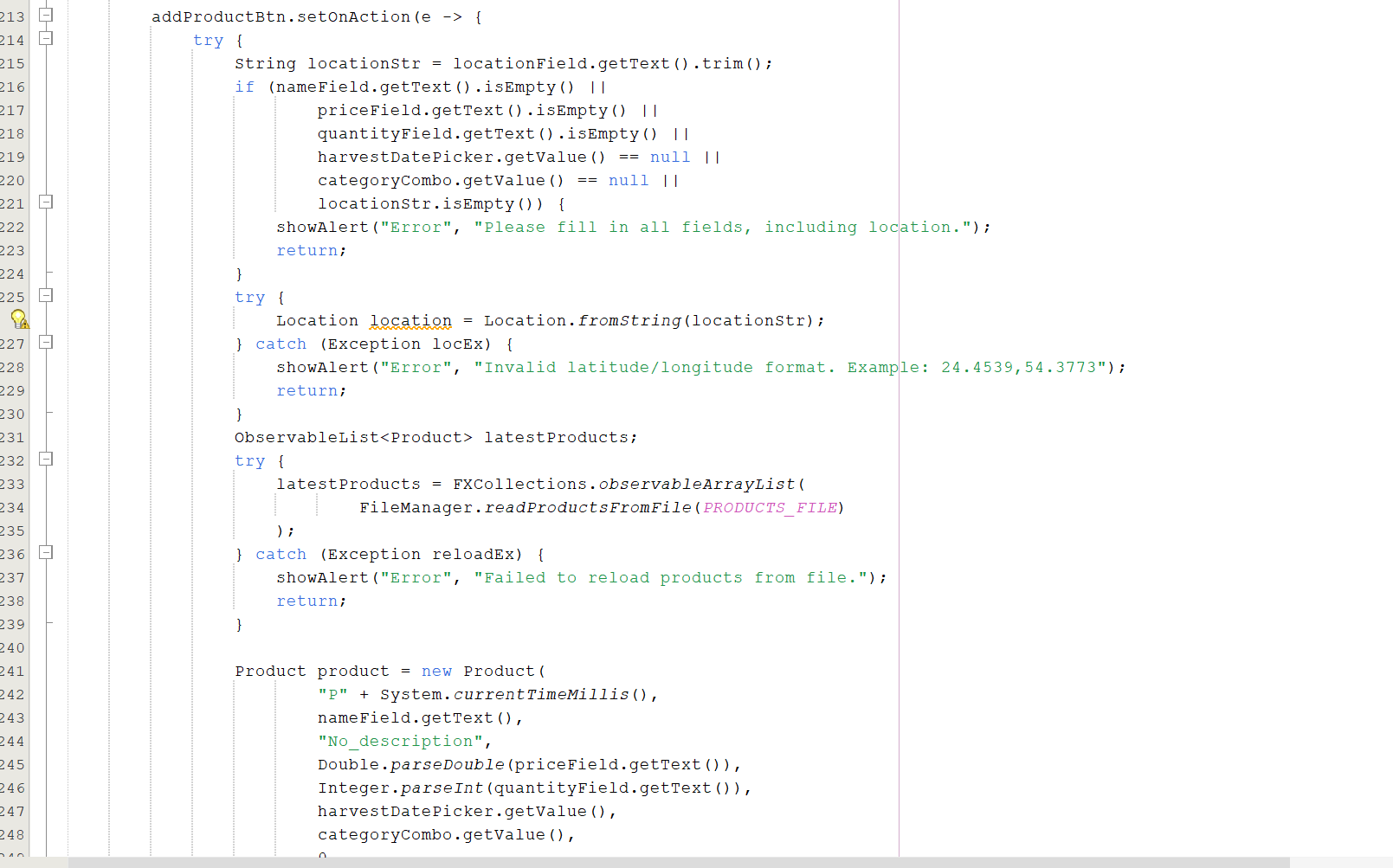
In `CustomerPortal.java`, customers can view available produce, manage subscriptions, and receive alerts. The `Subscription` class records a customer's subscription data.

A screenshot of a computer

AI-generated content may be incorrect.

## 3.4 Farmer Dashboard

Farmers use the `FarmerDashboard` to add and update produce information. It includes checks for harvest date validity using `HarvestDateException`.



## 3.5 Delivery System

The `DeliverySystem`, `DeliveryMap`, and `DeliveryRecord` manage the mapping of delivery paths and the record-keeping of dispatched produce.



# 4. Data Management

Data for customers, farmers, and produce is stored in plain text files under the `data/` directory.

- `customers.txt` contains registered customers.

- `farmers.txt` includes registered farmers.

- `produce\_updated.txt` store product information and updates.

The `FileManager` class abstracts the loading and saving of these files.

A screenshot of a computer

AI-generated content may be incorrect.

# 5. Exception Handling

Custom exceptions ensure that the system behaves correctly during unexpected conditions:

- `HarvestDateException`: Thrown when a product has an invalid harvest date.

- `OutOfSeasonException`: Ensures users don’t subscribe to products that are not in season.

- `DeliveryUnavailableException`: Raised when delivery is not available for a certain location.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# 6. Core Logic and Console-Based Components

Apart from GUI elements, the project includes substantial backend logic and core functionalities implemented purely in Java. These components are essential for maintaining the integrity, functionality, and data processing of the system, and they can be tested without the GUI as well.

- `Product.java`: Handles creation and management of produce objects with attributes like name, harvest date, and seasonality.

A screenshot of a computer screen

AI-generated content may be incorrect.

- `Customer.java` and `Farmer.java`: Extend the `User` class and provide logic for user-specific actions, such as subscribing to produce or listing new products.

A screenshot of a computer code

AI-generated content may be incorrect.  
- `Subscription.java`: Encapsulates the logic of managing customer subscriptions and links customers to their chosen produce.

A white screen with text

AI-generated content may be incorrect.

- `ProductSearch.java` and `ProductSearchEngine.java`: These classes enable logic-driven search of products based on filters.

A screenshot of a computer code

AI-generated content may be incorrect.  
- `DeliverySystem.java`: Implements the backend delivery logic, matching customers to farmers and validating availability.

A screen shot of a computer

AI-generated content may be incorrect.  
- `FileManager.java`: Handles reading and writing data from and to text files (e.g., customers.txt, produce.txt). This forms the storage logic of the application.

A screenshot of a computer program

AI-generated content may be incorrect.  
- `DeliveryRecord.java`: Maintains delivery history and updates subscription/delivery states.

A screen shot of a computer code

AI-generated content may be incorrect.

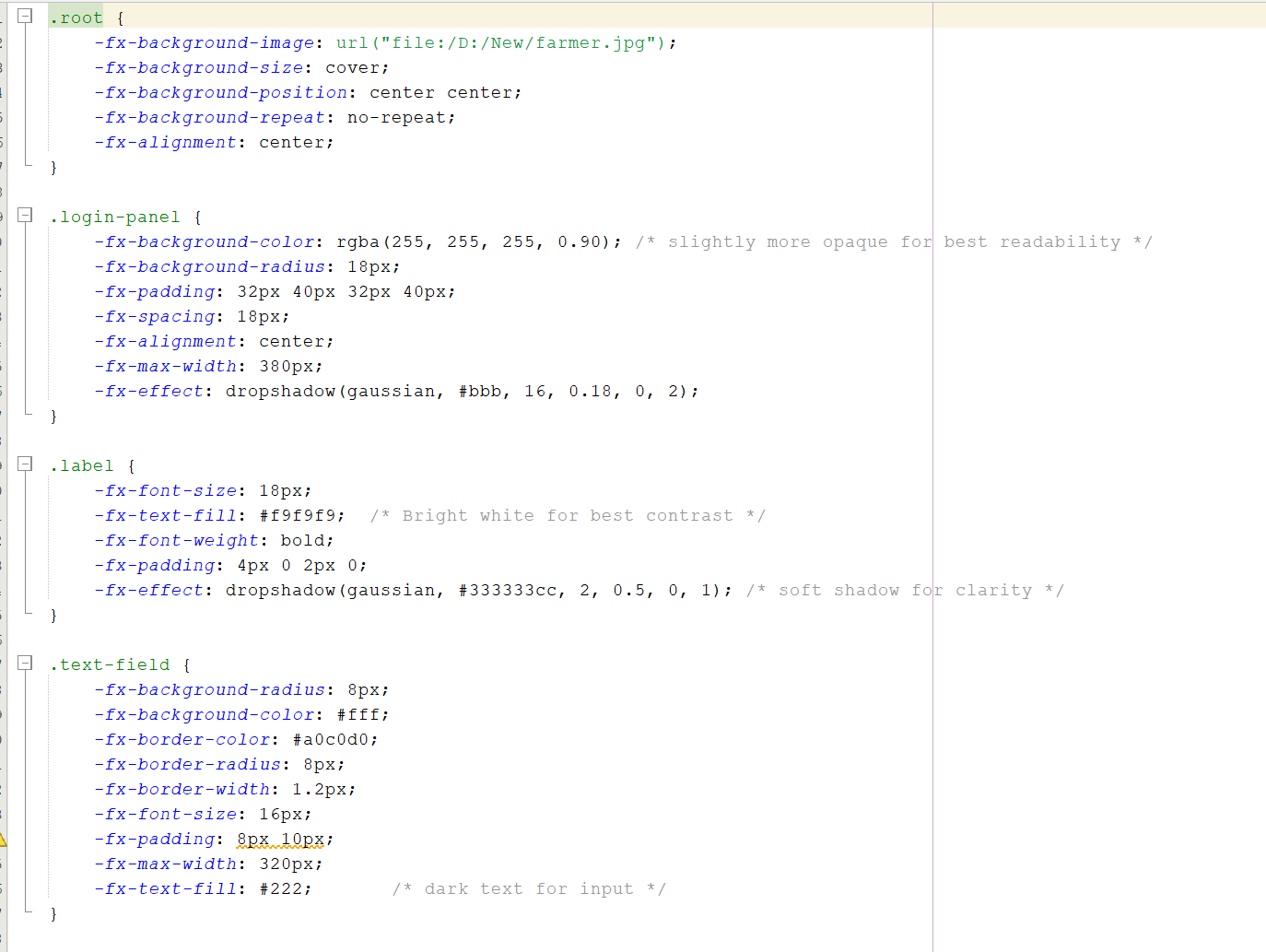
These components were structured using object-oriented principles. This design allows unit testing and further extension without relying on the user interface layer. Many of the classes use constructors, exception handling, interfaces, and polymorphism to encapsulate reusable logic.

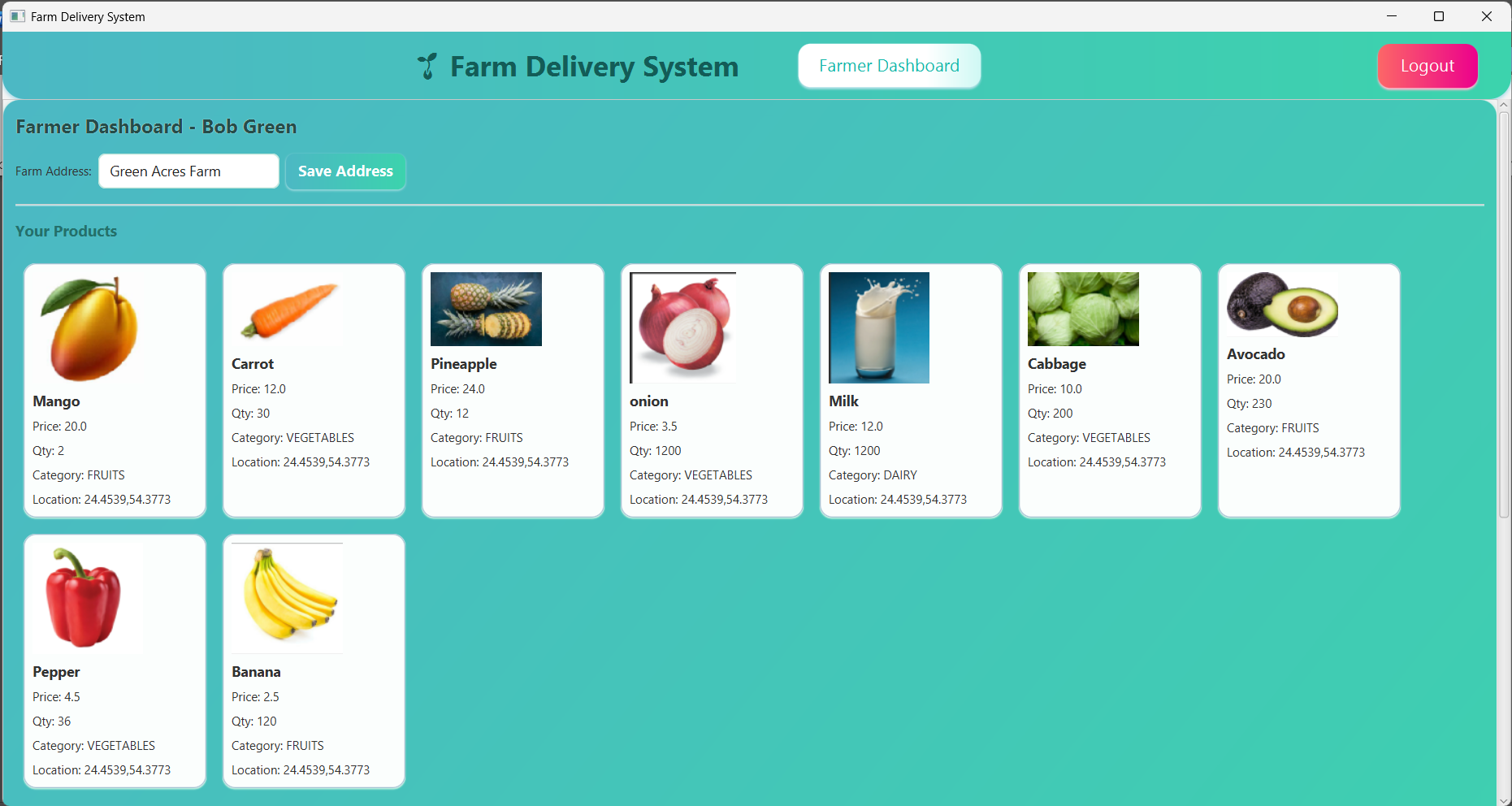
# 7. Graphical User Interface (GUI)

The GUI is built using JavaFX (or standard Swing, if specified).

The application starts from `MainApp.java`, which loads the main screens.

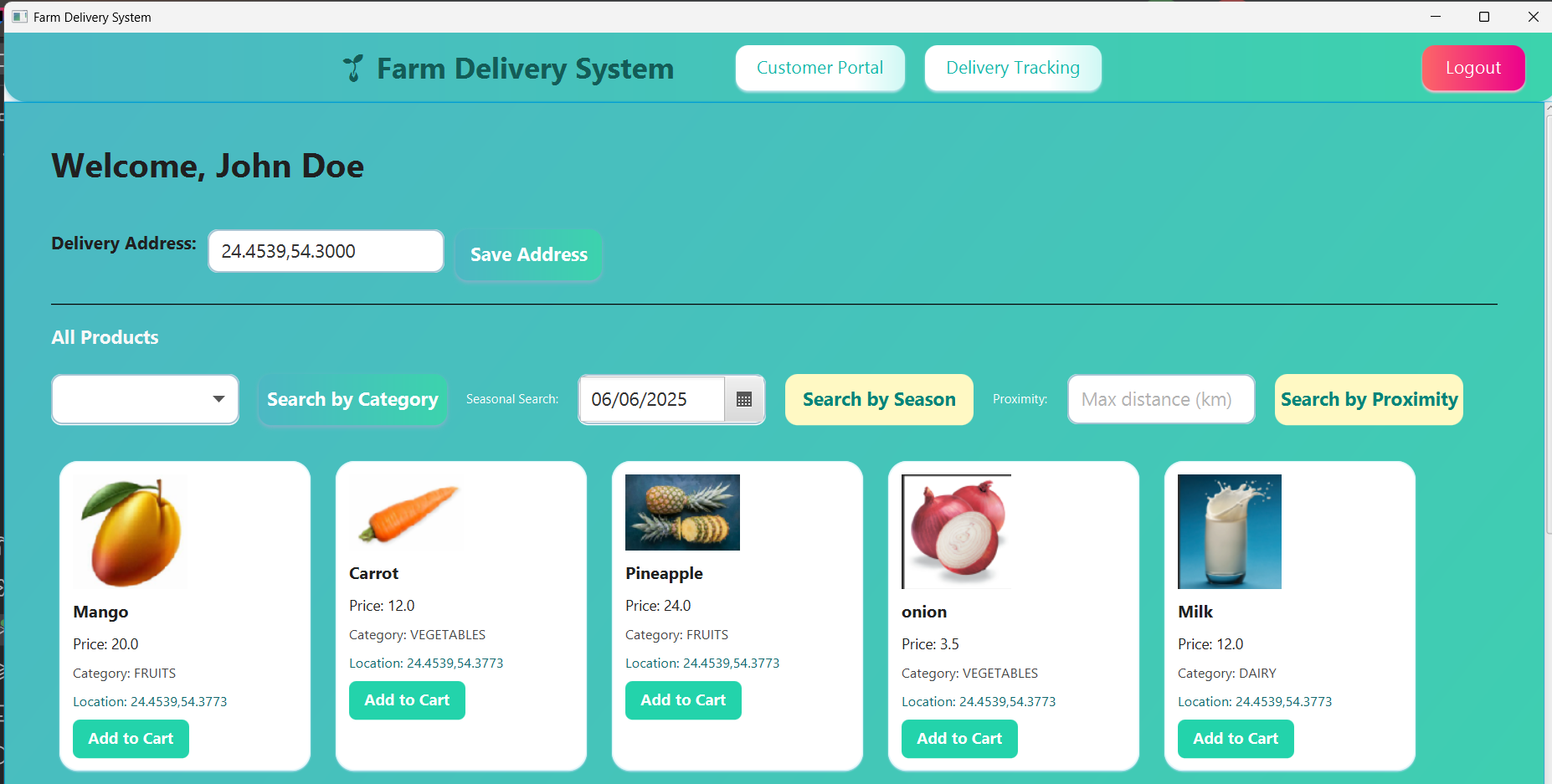
The `style.css` file provides styling rules for UI components.





A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a food delivery system

AI-generated content may be incorrect.

A screenshot of a map

AI-generated content may be incorrect.

# 8. Code Structure

The project follows Object-Oriented Programming (OOP) principles.

Interfaces like `IProductSearch` and `ILocation` define contracts that are implemented by corresponding classes.

The `MainApp.java` is the entry point and organizes the application's execution flow.

# 9. Challenges Faced

Some of the main challenges faced during development were:

- Managing file-based data without a database system.

- Implementing custom exceptions and ensuring robust validation.

- Designing intuitive user interfaces.

Despite these, we succeeded in creating a fully functional application that covers all use cases.

# 10. Testing and Validation

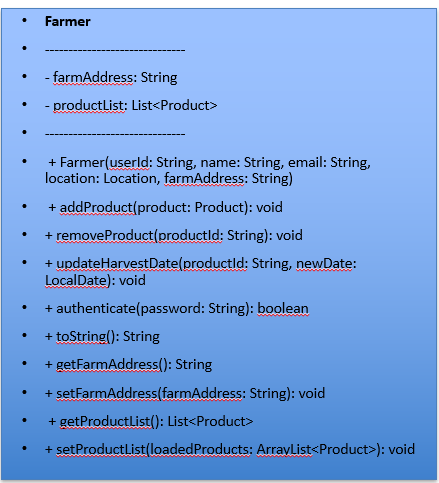
We tested the system manually by:

- Creating mock data files and loading them in the interface.

- Testing subscription, delivery, and login functionalities.

- Raising exceptions and verifying appropriate alerts are shown.

# 11. Sample UML



A blue screen with text

AI-generated content may be incorrect.

A blue screen with black text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# 

# 12. Conclusion

This assignment helped our group gain practical experience with Java programming, GUI development, and data management.

We successfully developed a working Farm Produce Delivery System and ensured all parts—from login to delivery—are functional.

In the future, we would consider migrating to a database backend and enhancing the GUI for a better user experience.