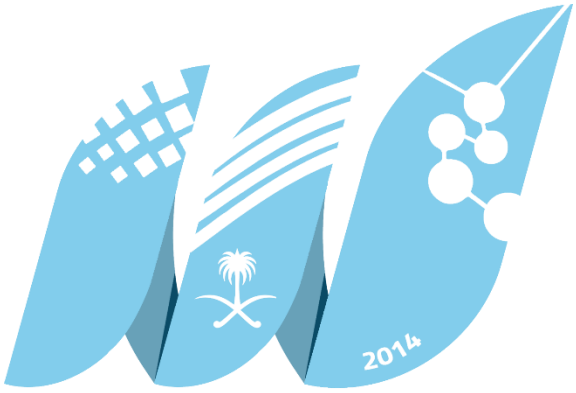


جامعة جدة

University of Jeddah

College of Computer Science and Engineering

Department of Software Engineering

Software Process Models

CCSW 315

Bloom	
Team Members	1. Lama Munir Noor | 2211796 2. Lama Fathi Akbar | 2212609 3. Joud Jalal Batarfi | 2210353 4. Raghad Al-saiari | 2210207 5. Ranad Waleed Aljhdali | 2211806 6. Hajar habiburahman | 2211350
Section	E2

Contents

Sprint #0: Project Initiation

Project Description

Bloom is a desktop application designed to connect plant enthusiasts, sellers, and buyers in a user-friendly platform. The system aims to create a digital marketplace for plants while providing valuable plant care information and management tools. Built using JavaFX and deployable as an executable JAR file, **Bloom** offers a dual-interface system catering to both sellers and customers, facilitating plant sales, care guidance, and personal plant collection management.

Problem Definition

The plant enthusiast community faces a fragmented marketplace and lacks a unified platform for plant purchases and care management. Novice owners struggle with accessing expert care instructions and maintaining proper care routines, while experienced growers have limited avenues to share their knowledge and sell plants. These disconnects results in inefficient plant care, potential plant health issues, and missed opportunities for knowledge sharing and sales. The absence of a comprehensive solution that combines a plant marketplace with personalized care management tools hinders the growth and success of both plant owners and sellers in this expanding hobby.

The Solution

Bloom addresses these problems through a comprehensive, user-friendly application:

1. A unified platform for plant sellers and buyers
2. A seller interface for uploading plant details and care instructions
3. A customer interface for browsing plants and managing personal collections
4. Integrated plant care management with watering reminders and care guides
5. User-friendly design built with JavaFX
6. Offline accessibility via executable JAR file

This comprehensive solution simplifies plant care, facilitates sales, and connects plant enthusiasts. **Bloom** bridges the gap between sellers and buyers while offering valuable tools for plant care management, promoting successful plant ownership and fostering a community of plant lovers.

The Scope

The scope of the **Bloom** includes:

1. **Seller interface:** sellers can upload plant pictures, name, type, and care instructions
2. **Customer interface:** enable customers to browse and search for plants to add to their collection, providing options to view all plants in their collection, including the ability to delete plants.
3. **Plant Purchasing:** Users can browse and purchase plants through the app.
4. **Plant delivery:** user can schedule plant delivery.
5. **Watering countdown:** users can set countdown to remind them when to water their plants depending on its instructions.
6. **Notifications:** Customizable reminders for watering, and other plant care tasks, with the ability to enable or disable reminders at any time.

Out of Scope:

1. **Plant database integration:** The app does not include features for identifying plants through images or descriptions.
2. **Community features:** No social sharing, forums, or community interaction within the app.
3. **Payment processing:** The app does not support payment processing for the purchase of plants or integrate with other e-commerce systems.

The Users

1. **Sellers:**

These are users who register with the intent to sell plants. They use the system to:

   a. Upload pictures of plants they want to sell, their name and type.
   b. Set prices and quantity for their plants.
   c. Provide detailed care instructions for each plant (including soil, watering, and fertilization information).

2. **Customers:**

These are users who register with the intent to browse, learn about, and potentially purchase plants. They use the system to:

   a. Browse through plant listings.
   b. Add plants to their personal collection, purchase and get them delivered.
   c. View care instructions for plants.
   d. Set timers and countdowns for watering their plants.

Product Backlog

Functional Requirements

1. **User Management:**
   - **1.1.** Users shall be able to register with a username, email and password
   - **1.2.** Users shall select their account type (seller or customer) during registration
   - **1.3.** Sellers must specify their store name.
   - **1.4.** Users shall be able to log in and log out of their accounts

2. **Seller Features:**
   - **2.1.** Sellers shall be able to upload plants picture, name, characteristics, care information, fertilization options, price and quantity.
   - **2.2.** The system shall allow sellers to edit or remove plants they have added to the system
   - **2.3.** The system shall allow sellers to view the purchase history of their plants.
   - **2.4.** The system must inform the seller when a plant's quantity is running lower than 5.

3. **Customer Features:**
   - **3.1.** Customers shall be able to browse plant listings
   - **3.2.** The system shall allow users to search for plants by name.
   - **3.3.** If a plant is not found, the system shall notify the user that the plant is unavailable.
   - **3.4.** Customers shall be able to add plants to their collection
   - **3.5.** Customers shall be able to view care instructions for plants in their collection.
   - **3.6.** Customer shall be able to explore fertilization possibility between two plants of their collection.
   - **3.7.** The system shall allow users to delete plants from their collection.

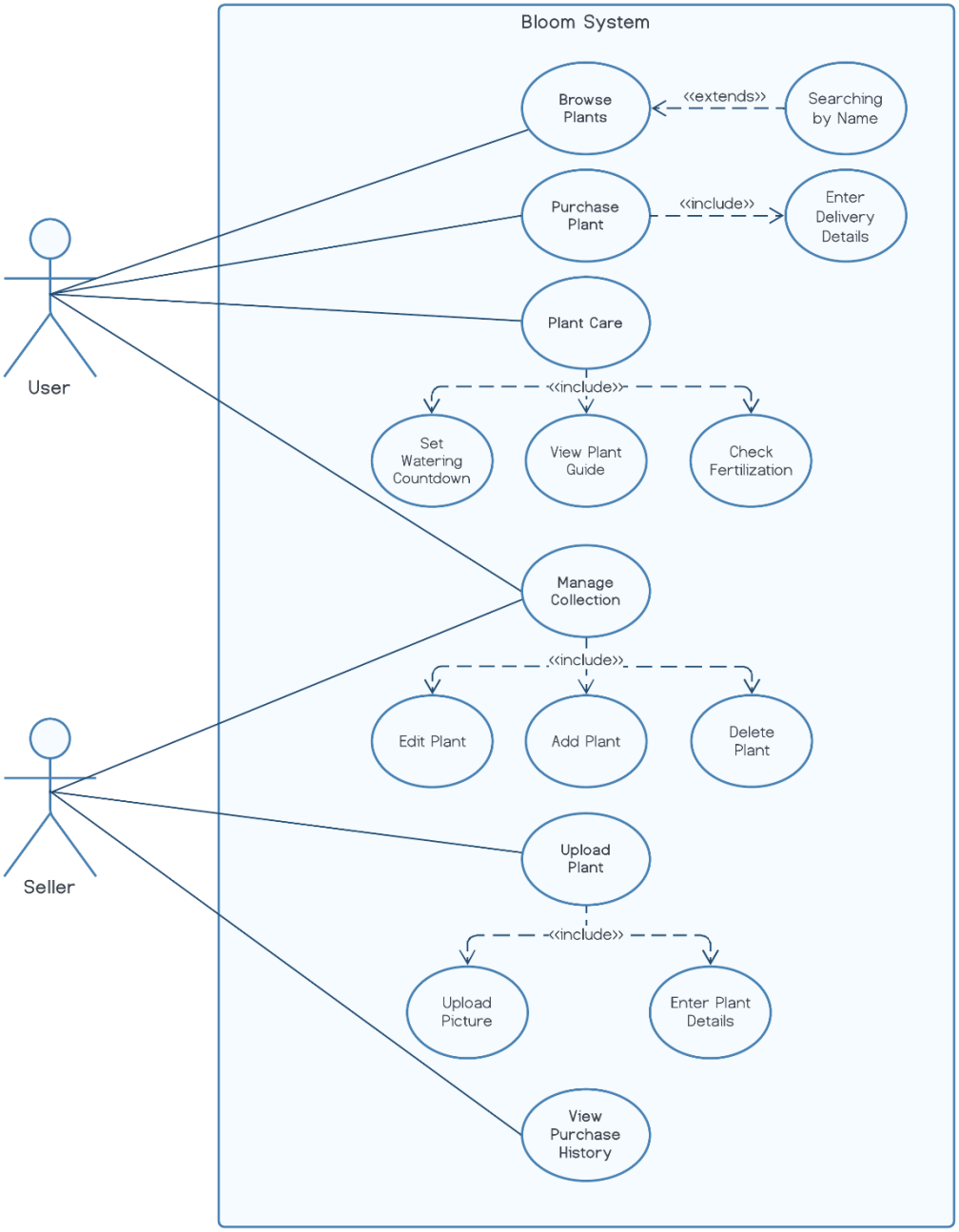
4. **Timer Functionality:**
   - **4.1.** The system shall allow setting of watering timers for each plant
   - **4.2.** The system shall provide notifications when watering is due

Non-Functional Requirements:

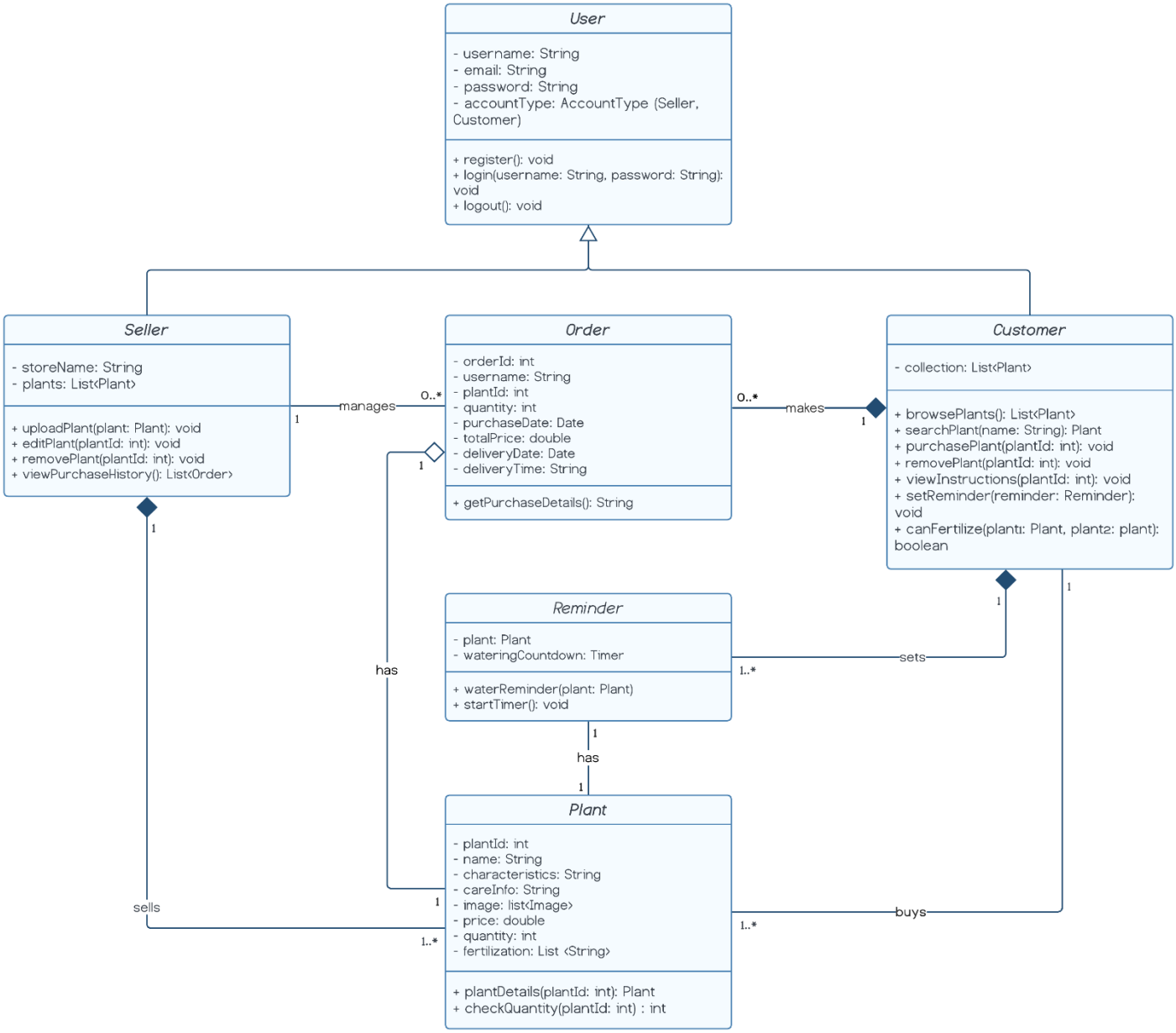
1. **Performance:**
   - **1.1.** The application shall load plant listings within 3 seconds on a standard broadband connection
2. **Usability:**
   - **2.1.** The user interface shall be intuitive and require no more than 5 minutes of learning for basic operations
3. **Reliability:**
   - **3.1.** The system shall have an uptime of 99.9% excluding scheduled maintenance
4. **Security:**
   - **4.1.** The system shall enforce strong password policies to enhance user account security.
5. **Compatibility:**
   - **5.1.** The application shall run on Windows, macOS, and Linux operating systems with Java Runtime Environment 8 or higher
6. **Maintainability:**
   - **6.1.** The codebase shall follow object-oriented design principles and be well-documented
7. **Storage:**
   - **7.1.** The application shall not exceed 100MB in size for the executable JAR file
8. **Response Time:**
   - **8.1.** User interactions (e.g., adding a plant to collection) shall have a response time of less than 1 second

System modelling

Use Case Diagram



Class Diagram



Tools

Integrated Development Enviroment (IDE)

Visual Studio Code, Netbeans, and Eclipse



Agile Planning Tool

Jira

Modeling App

Lucidchart

Git Tools

GitBash and Github


Messaging App

Slack

Sprint #1

Duration: October 3, 2024 - October 10, 2024

1. Sprint Overview

In this sprint, our primary focus was on developing and refining the **Customer** and **Buyer** classes, implementing a registration feature, and enabling sellers to upload plant information to the system. These features are crucial for enhancing user interaction and ensuring a seamless experience within our platform.

2. Sprint Goals

- **Develop Customer and Buyer Classes**: Implement the core functionalities needed for user management.
- **Registration Feature**: Create a user-friendly registration process for customers and sellers.
- **Plant Upload Functionality for Sellers**: Allow sellers to upload detailed plant information to the system.

3. Completed Work

- **User and Plant classes**:
    - Inheritance relation between the User class (parent) and customer, buyer classes (children).
    - Basic setters and getters, and toString methods along with registration.
    - Upload plant method for the sellers, check fertilization in plants.
- **Registration Feature**:
    - Designed a registration form with necessary fields (name, email, password, account type) for both customers and sellers and store name for sellers.
    - Implemented backend validation and file storage.
- **Plant Upload Functionality**:
    - Enabled sellers to upload plant details, including images, descriptions, and care instructions.
    - Established a database schema to store plant information and seller details.

Sprint Backlog

- **Seller Registers to the System**

  Component Name: User Registration

  Story Name: Seller Sign Up

  Story Sequence No: 1.1

  Story Short Description: Ability for sellers to sign up for new accounts, and filing their information in the users' directory.

  Story Long Description: As a seller, I want to register for a new account using my username, email address, password, and store name so that I can access my seller dashboard to upload new plants, with my information filed in a directory.

- **Customer Registers to the System**

  Component Name: User Registration

  Story Name: Customer Sign Up

  Story Sequence No: 1.2

  Story Short Description: Ability for customers to sign up for new accounts, and filing their information in the users' directory.

  Story Long Description: As a customer, I want to register for a new account using my username, email address, and password, so that I can access my customer dashboard, with my information filed in a directory.

- **Seller Upload New Plant**

  Component Name: Plants Uploading

  Story Name: Upload Plants

  Story Sequence No: 2

  Story Short Description: Ability for sellers to upload Plants with their details, and filing their information in the plants' directory.

  Story Long Description: As a seller, I want to upload the plant ID, name, characteristics, care information, image, price, and quantity of my plants so that I can effectively add them for sale on the Bloom platform and provide potential buyers with all necessary information for plant care, with my plants' information filed in a directory.

- **Upload Confirmation Message**

  Component Name: Plants Uploading

  Story Name: Confirm Uploading

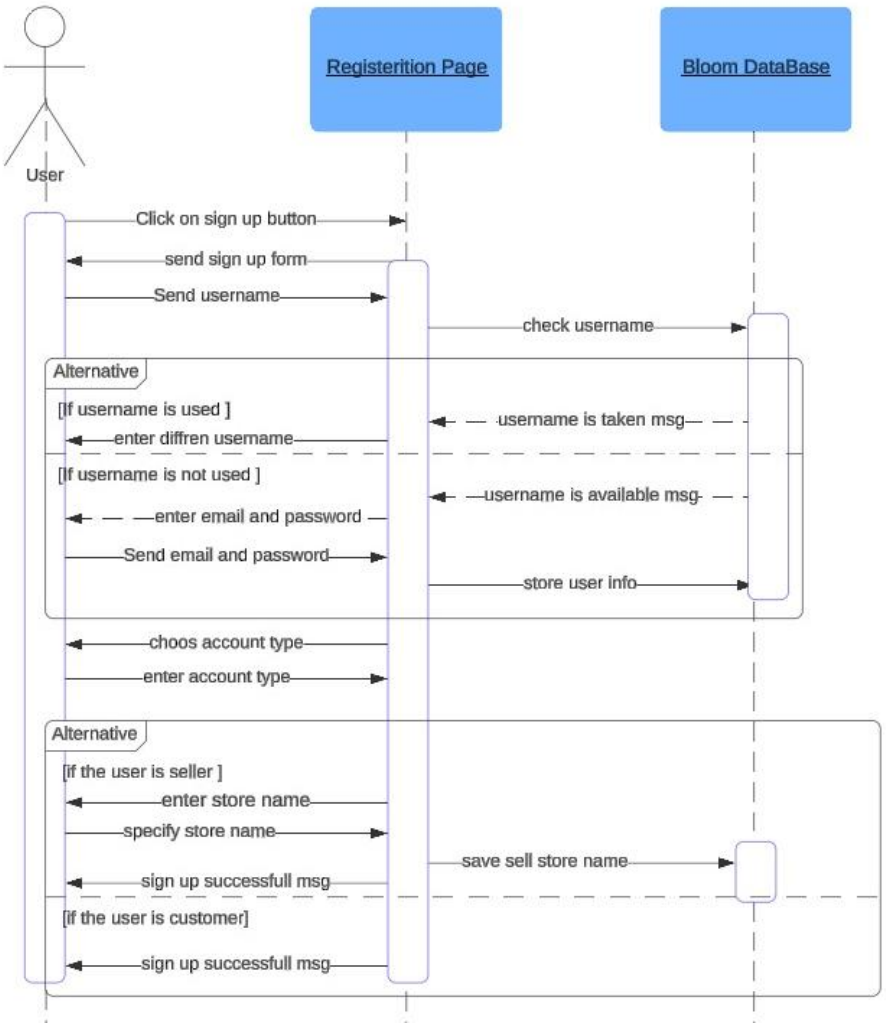
  Story Sequence No: 2.1

  Story Short Description: Confirmation message after successful plant uploading.

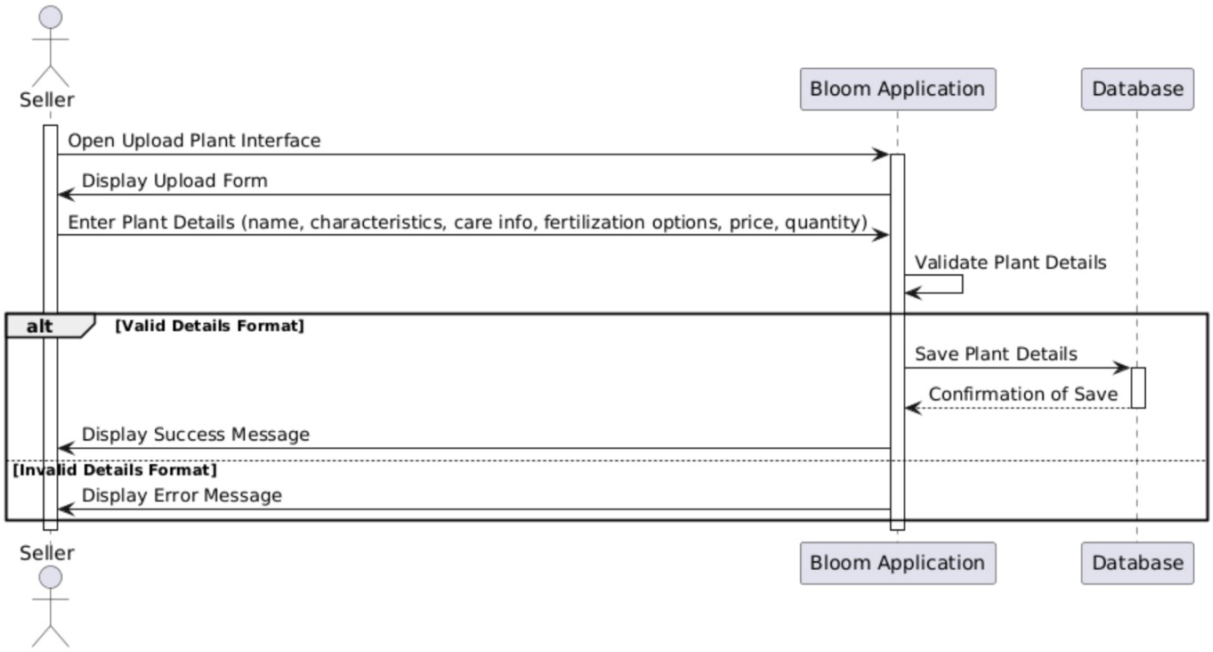
  Story Long Description: As a seller, I want to receive a confirmation message after successfully uploading my plant details so that I know the upload was completed successfully and my listing is now available for buyers to view.

Dynamic Models

Sequence diagram 1: User Register



Sequence Diagram 2: Seller Uploads a plant



Testing

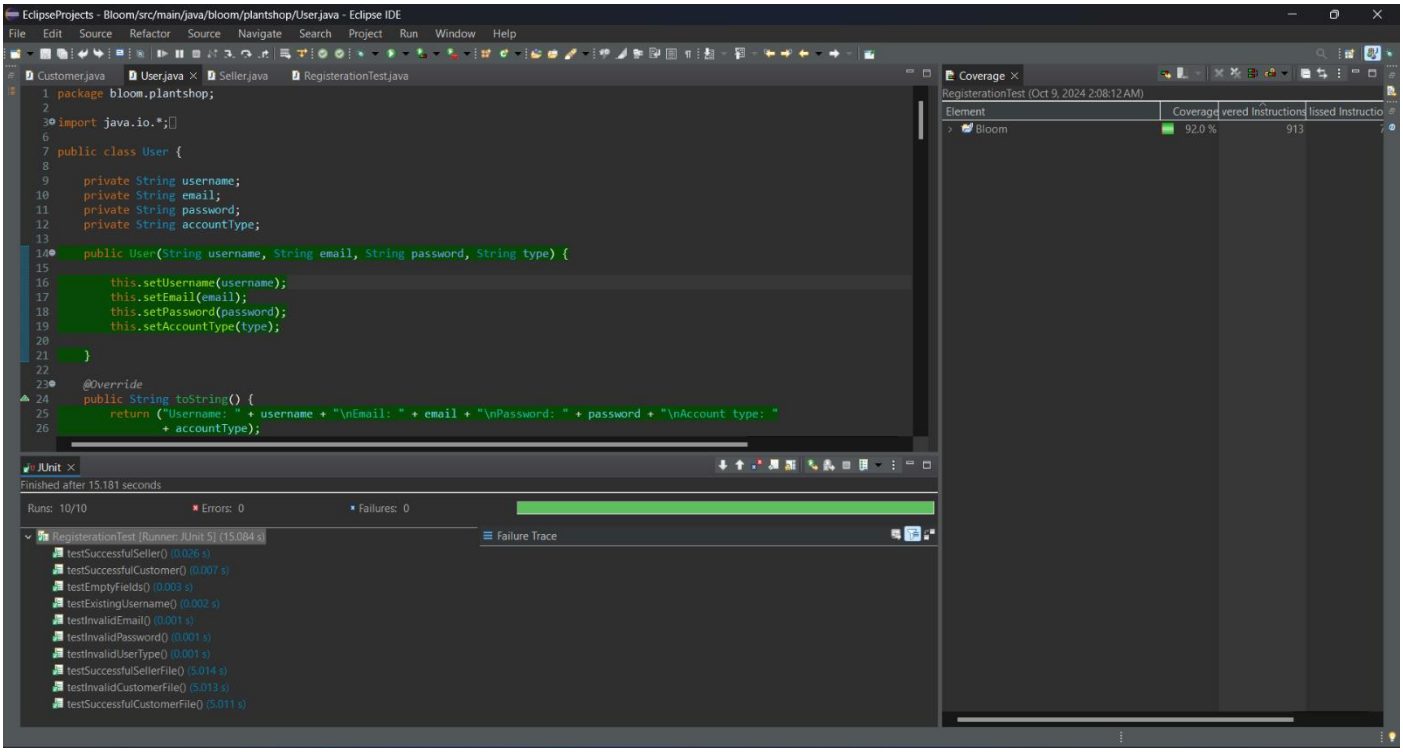
Functionality 1: User Registration

Model Structure: Input Space Domain

Coverage Criteria: Base Choice Coverage (valid, valid, valid, valid)

Characteristic	B1	B2	B3
Q1: Username	Valid	Invalid	Null
Q2: Email	Valid	Invalid	Null
Q3: Password	Valid	Invalid	Null
Q4: Account Type	Valid	Invalid	Null

Coverage Level: 92.0%



Test cases

Sprint #1 Test Cases - Wednesday, October 9, 2024
Test Case Name: Sprint 1 – Registration and user directory.
"Extra test cases were done to test the correctness of error messages without conflicts".

Test Case ID	Description	Expected Result	Test Result
TC1 – test successful seller	A user registers with valid username, email, password and chooses the type seller.	A User object of type seller should be created with matching fields to the inputs.	Passed
TC2 – test successful customer	A user registers with valid username, email, password and chooses the type customer.	A User object of type customer should be created with matching fields to the inputs.	Passed
TC3 – test empty user	User leaves all fields blank.	User should be displayed with a message: "All fields are required.", and the program would prompt the empty field.	Passed
TC4 – test existing username	User enters an existing username.	User should be displayed with a message "Username already exists. Please try again.", and repeats the registration process.	Passed
TC5 – test invalid email	A user enters an invalid email format.	User should be displayed with a message "Invalid email format. Please try again.", and the program requests for the email again.	Passed
TC6 – test invalid password	A user enters an invalid password format.	User should be displayed with a message declaring the valid password format, and be prompted for another password.	Passed

Test Case ID	Description	Expected Result	Test Result
TC7 – test invalid user type	A user enters an invalid input for the account-type prompt.	User should be displayed with a message "Invalid input. Please enter 'y' if you are registering as a seller or 'n' for no."	Passed
TC8 – test successful seller file	The user registers successfully in the system, the system logs their information in a file.	System creates a file for the user with their username under the users' directory in case of a successful seller's registration.	Passed
TC9 – test successful customer file	The user registers successfully in the system, the system logs their information in a file.	System creates a file for the user with their username under the users' directory in case of a successful customer's registration.	Passed
TC10 – test invalid user file	A user's failed registration attempts should not be logged in the system.	System must not create a file with the user's invalid inputs.	Passed

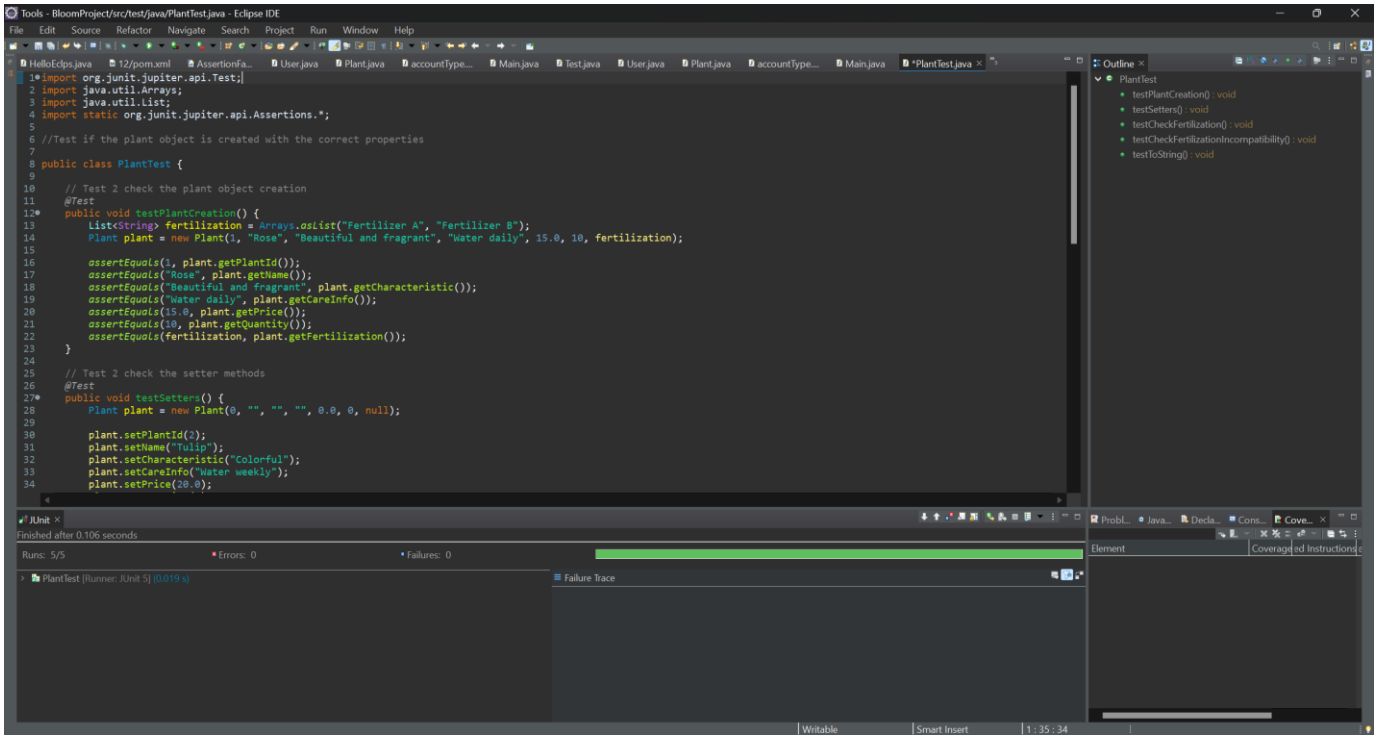
Functionality 2: Upload Plant Details

Model Structure: Input Space Domain

Coverage Criteria: Base Choice Coverage (valid, valid, valid, valid, valid, valid, valid)

Characteristic	B1	B2	B3
Q1: Plant ID	Valid	Invalid	Null
Q2: Name	Valid	Invalid	Null
Q3: Characteristics	Valid	Invalid	Null
Q4: Care Info	Valid	Invalid	Null
Q5: Price	Valid	Invalid	Null
Q6: Quantity	Valid	Invalid	Null
Q7: Fertilization	Valid	Invalid	Null

Coverage Level: 31.6%



Test cases

Sprint #1 Test Cases - Wednesday, October 9, 2024
Test Case Name: Sprint 1 – Plant Upload

Test Case ID	Description	Expected Result	Test Result
TC1	A plant object is created with valid Plant ID, Name, Characteristics, Care Info, Price, Quantity, and Fertilization.	A Plant object should be created with matching fields to the valid inputs.	Passed
TC2	A plant object is created with invalid Plant ID while other fields are valid.	The system should throw an error or exception for invalid Plant ID.	Passed
TC3	A plant object is created with valid Plant ID but with null Name, Characteristics, or Care Info.	The system should prompt a message declaring "Fields cannot be null."	Passed
TC4	A plant object is created with invalid Price (negative or too high) while other fields are valid.	The system should throw an error or validation message for invalid price value.	Passed
TC5	The “checkFertilization” method tests two plants with compatible fertilization.	The system should confirm the plants can be fertilized together.	Passed