ÇANKAYA UNIVERSITY

SOFTWARE ENGINEERING DEPARTMENT

SOFTWARE PROJECT I

Name Surname	**Dilara Çağla Sarısu**
Identity Number	**202128201**
Course	**SENG-271**
Experiment	**Experiment-2**
E-mail	**c2128201@student.cankaya.edu.tr**

PROBLEM STATEMENT

Describe the goals of the programming assignment

- Design a simplified version of an online shopping site named "MyBazaar".
- Manage basic components such as users (service providers and customers) and products for sale.
- Implement system functionalities using data structures and algorithms.
- Provide practical experience in programming, data structures, and algorithm design.

What are the normal inputs to the program?

- **Input Files:** The program uses three specific tab-separated text input files:
    - **users.txt:** Contains information about all users in the system (admins, technicians, and customers). Each user type is indicated by a keyword (ADMIN, TECH, CUSTOMER) at the start of each line.
        - **Format for Admins**: ADMIN<TAB>name<TAB>email<TAB>birthDate<TAB>salary<TAB>password
        - **Format for Technicians:** TECH<TAB>name<TAB>email<TAB>birthDate<TAB>salary<TAB>isSeniorTechnician
        - **Format for Customers:** CUSTOMER<TAB>name<TAB>email<TAB>birthDate<TAB>balance<TAB>password
    - **item.txt:** Lists all items for sale, categorized by type (e.g., DESKTOP, LAPTOP, TABLET, TV, SMARTPHONE, BOOK, CDDVD, HAIRCARE, SKINCARE, PERFUME). Each line starts with a keyword indicating the item type.
        - **Example format for a Desktop:** DESKTOP<TAB>cost<TAB>manufacturer<TAB>brand<TAB>maxVolt<TAB>maxWatt<TAB>operatingSystem<TAB>CPUType<TAB>ramCapacity<TAB>HDDCapacity<TAB>boxColor
    - **commands.txt:** Contains a list of all commands to be executed by the program. These commands are structured with specific syntaxes for various actions like adding customers, showing customer information, creating campaigns, and more.
- **User Input Data:** Includes detailed information about users, such as name, email, date of birth, salary (for employees), customer balance, status, and passwords.
- **Item Input Data:** Covers detailed specifications of items, including cost, manufacturer, brand, technical specifications (like max voltage, wattage, operating system for electronics), and other relevant attributes depending on the item type.
- **Command Inputs**: Consists of structured commands for operations such as adding users, listing items, launching campaigns, etc., formatted according to the specified syntax in the commands.txt file.

What output should the program create?

- **Displaying User Information:**
    - All users (admins, technicians, and customers) can view their personal data (name, email, date of birth).

    - Customers can also view their customer IDs, balance, and status.
- **Listing and Information of Items:**
    - Admins and technicians can display stock quantities and types of products.
    - A list of all available products and their types, prices, stock statuses, and other related information is displayed.
- **Displaying Customer Information:**
    - Admins can view detailed information of a specific customer (name, ID, email, date of birth, status).
- **Displaying VIP Customers:**
    - Admins and senior technicians can display VIP customers with GOLDEN status.
- **Campaign Information:**
    - Information about campaigns initiated by admins (start and end dates, product type, discount rate) can be displayed.
    - Customers can view active campaigns.
- **Order Summaries and Transactions:**
    - When a customer's order is successful, an order summary is displayed, including the order date, total cost, list of purchased items, and customer ID.
    - After each order, the customer’s status and the stock quantities of purchased items are updated.
- **Displaying Product Information:**
    - Technicians can display detailed information of a specific product (price, manufacturer, brand, technical specifications).
- **Displaying Products with Low Stock:**
    - Admins and technicians can display products with stock quantities below a certain threshold.
- **Error Messages:**
    - In cases of invalid inputs or operations, the program displays appropriate error messages (e.g., attempting a transaction with a non-existent username).
- **Console Outputs:**
    - All outputs are displayed on the console in accordance with the specified formats.

What error handling was required?

- **Invalid Input Handling:**
    - The program detects invalid command inputs and syntax errors, displaying appropriate error messages.
    - Each command must be in a specific format and order (e.g., tab-separated arguments).
- **User Verification:**
    - Checks for the existence of users (e.g., attempting an operation with a non-existent admin name).
    - User authentication is critical, especially for customer orders and password changes.
- **Stock Management:**

    - Monitors product stock quantities and informs the user if a product is out of stock.
    - Updates stock quantities of relevant products after order transactions.
- **Order Validation:**
    - Verifies whether the customer has sufficient balance and displays an error message if the balance is insufficient.
    - Checks if the customer's shopping cart is empty and whether the password is correct when placing an order.
- **Campaign Management:**
    - Ensures campaigns are created with correct parameters (e.g., not exceeding the maximum discount rate).
    - Prevents adding a new campaign if there is already an active campaign for a given type.
- **Status Updates:**
    - Ensures accurate updating of customer status (e.g., from CLASSIC to SILVER).
    - Provides necessary notifications based on the customer's new status after an order.
- **Reading from and Writing to .txt Files:**
    - Error handling includes validating the data read from and written to the users.txt, items.txt, and commands.txt files.
    - Ensures the integrity and correctness of data in these text files.
- **Console Error Messages:**
    - All error messages are displayed on the console in a clear and understandable manner.
    - Error messages guide the user to correct their actions.
- **General Robustness:**
    - Continuously checks the correctness of user inputs and system operations.

DESIGN

Describe the design decisions you made.

- **Class Hierarchy and Design:**
    - Created a hierarchical class structure for users (admins, technicians, customers) and items (electronics, cosmetics, etc.), reflecting real-world relationships.
    - Designed user classes to include common attributes (name, email, birth date) and role-specific attributes (salary for employees, balance, and status for customers).
- **File-Based Data Initialization:**
    - Utilized tab-separated text files (users.txt, items.txt) for initializing users and items, ensuring flexibility and ease of data management.
- **Command Parsing and Execution:**
    - Developed a mechanism to parse and execute commands from the commands.txt file, enabling dynamic interaction based on user inputs.
- **Error Handling and Validation:**
    - Implemented robust error handling for input validation, ensuring the integrity of user actions and system operations.

- **Modular Design:**
    - **Adopted a modular design approach to facilitate code maintenance and future expansions.**
- **Data Encapsulation and Security:**
    - Ensured data encapsulation in classes, providing access through methods, and added security measures like password protection for sensitive user data.

What data structures did you use?

- **Classes:** To represent users (Admin, Technician, Customer) and products (Cosmetic, Electronic, Office Supplies).
- **Hierarchical Class Structure:** Different subclasses for users and products (e.g., CLASSIC, SILVER, GOLDEN statuses for Customer).
- **Lists:** For storing and sorting products and users.
- **Dictionaries:** To store product details and user information as key-value pairs.
- **Sets:** To store unique items like user names or product IDs.
- **Static and Constant Variables:** To store fixed data like customer status limits and discount rates.
- **Arrays:** To store certain data (e.g., product stock information) in an ordered manner.

What algorithms did you use?

- **Product and User Search Algorithms:** Used for searching users and products in the database or lists.
    - Binary Search
    - Hash Tables
- **Sorting Algorithms:** Applied for sorting product lists or user information, particularly by stock quantity or price.
    - QuickSort
    - MergeSort
- **Access Control and Authorization Algorithms:** Employed to ensure security in user login and authorization processes.
- **Stock Management and Updating Algorithms:** Used for updating and tracking product stocks.
    - FIFO (First-In-First-Out)
- **Discount and Campaign Management Algorithms:** Utilized in managing product pricing, discounts, and promotions.
- **Input.txt File Processing Algorithm:** For reading and parsing input files containing user and product information, and converting this data into appropriate classes and data structures within the system.

What were pros/cons of choices above?

- **Product and User Search Algorithms (Binary Search, Hash Tables):**
    - **Pros:** Fast and effective for small to medium-sized datasets.

- **Cons:** Hash Tables may have performance issues in large datasets; Binary Search requires sorted data.

- **Sorting Algorithms (QuickSort, MergeSort):**
  - **Pros:** QuickSort is very efficient for large datasets; MergeSort is stable and works well with large data.
  - **Cons:** QuickSort can be slower on already sorted data; MergeSort requires additional space.

- **Access Control and Authorization Algorithms:**
  - **Pros:** Enhances security, prevents unauthorized access.
  - **Cons:** Can be complex to implement and manage, especially in scalable systems.
- **Stock Management and Updating Algorithms (FIFO):**
  - **Pros:** FIFO ensures timely product rotation, good for perishable items.
  - **Cons**: Not always optimal for non-perishable items; can be complex in large inventories.

- **Discount and Campaign Management Algorithms:**
  - **Pros:** Flexible in managing various pricing strategies and promotions.
  - **Cons:** Requires constant updating to reflect market trends and inventory changes.

- **Input.txt File Processing Algorithm:**
  - **Pros:** Automates data entry, reduces manual errors.
  - **Cons:** Requires well-structured input; parsing errors can occur with malformed data.

IMPLEMENTATION DETAILS

- Describe your implementation process. (Flowchart)
- **What sample code did you start with?**

```
public class Person {
    private String name;
    private String email;
    private String birthDate;
```

```
public class Item {
    private String itemId;
    private String name;
    private double price;
```

```
public class ShoppingCart {
    private Map<String, Integer> items;

    public ShoppingCart() {
        items = new HashMap<>();
    }
```

```
public class MyBazaar {
    public static void main(String[] args) {

    }
}
```

```java
public class Order {
    private Date orderDate;
    private double totalCost;
    private Map<String, Integer> items;
    private String customerID;
```

In this initial code, I created the Person class, as it stores basic user information like names and email addresses. Then, I added the Item class, which holds details of products for sale, such as each item's name and price. The ShoppingCart class is used to manage users' shopping carts, adding selected products (Item objects) to a list. Finally, the Order class manages the details of an order, specifying which user (Person) bought which products (Items in ShoppingCart).

- **How did you extend or adapt this code?**
    - **Person Class:** Added methods to manage personal information, such as updating a user's email address.

```java
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}
```

```java
public void setEmail(String email) {
    this.email = email;
}

public String getBirthDate() {
    return birthDate;
}

public void setBirthDate(String birthDate) {
    this.birthDate = birthDate;
}
```

    - **Item Class:** Added methods to check product stock status and update the price.

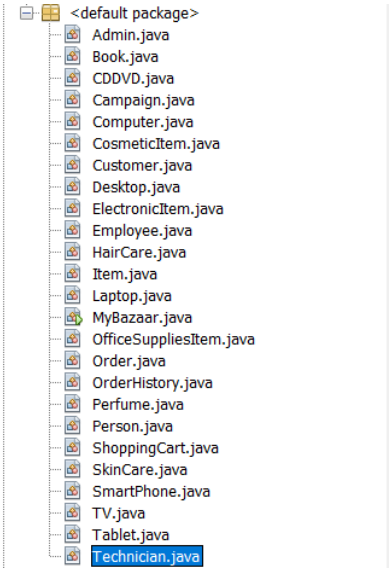
```java
public String getItemId() {
    return itemId;
}

public void setItemId(String itemId) {
    this.itemId = itemId;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getPrice() {
    return price;
}
```

```java
public void setPrice(double price) {
    this.price = price;
}

public String getBrand() {
    return brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getManufacturer() {
    return manufacturer;
}

public void setManufacturer(String manufacturer) {
    this.manufacturer = manufacturer;
}
```

- **ShoppingCart Class:**

```java
public void addItem(String itemName, int quantity) {
    items.put(itemName, items.getOrDefault(itemName, 0) + quantity);
}

public void removeItem(String itemName, int quantity) {
    int currentQuantity = items.getOrDefault(itemName, 0);
    if (currentQuantity <= quantity) {
        items.remove(itemName);
    } else {
        items.put(itemName, currentQuantity - quantity);
    }
}
```

- **Creating some of the other classes:**



- **What was your development timeline?**

  Here's how my first three weeks of development went:

  - **First Week:** Project planning and requirement analysis. Here, I determined the overall goals and functions of the application.
  - **Second Week:** Began coding by creating the **Person**, Item, **ShoppingCart**, and **Order** classes. These formed the foundation of the project.
  - **Third Week:** I developed additional helper classes and enhanced the existing ones. This was necessary to support more complex functions and user interactions of the project.

TESTING NOTES

- **Describe how you tested your program.**

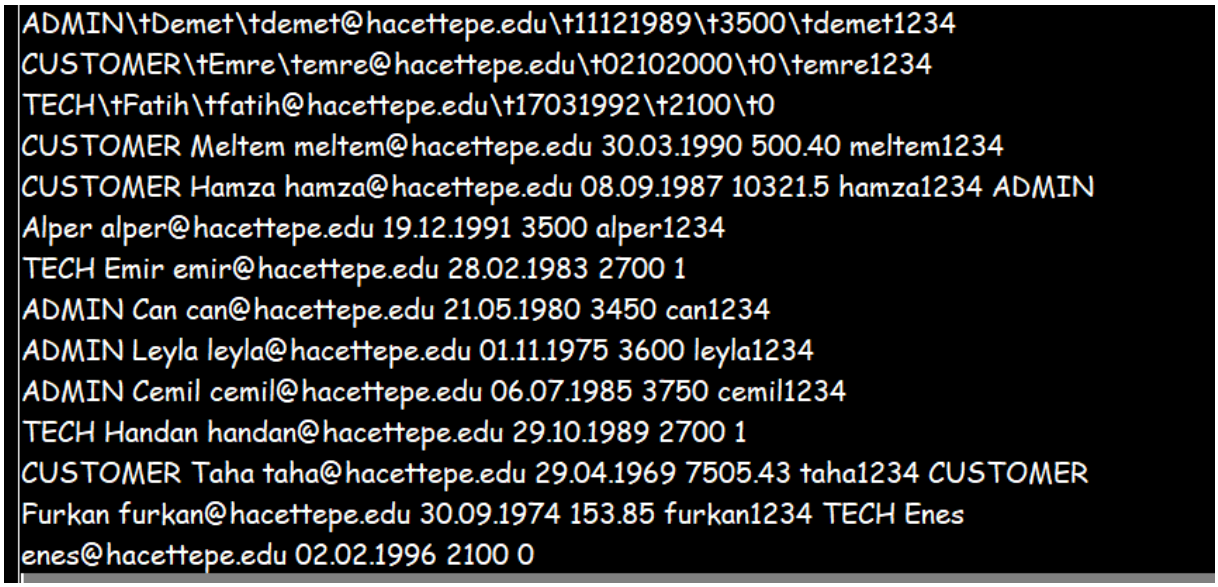
  - The testing process of the project was carried out using various methods. First, unit tests were written to test the basic functionality and flow of the application. These tests aimed to check whether base classes such as **Person, Item, ShoppingCart** and

Order were working correctly. Then, more complex functions and user interactions of the application were tested with user scenarios and flow tests. A systematic approach was followed to identify and correct errors.

- **What were the normal inputs you used?**

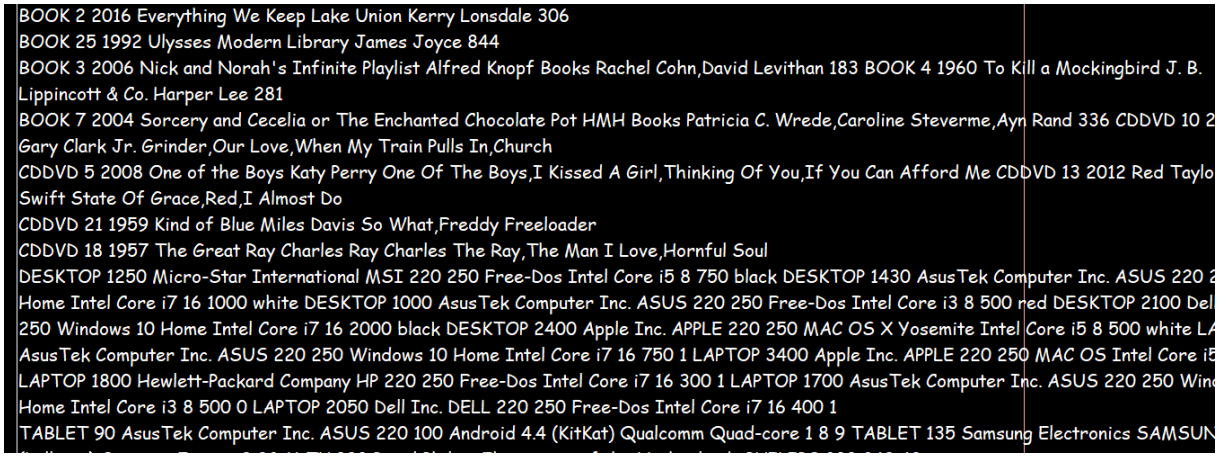
My normal test inputs(from the input file) included the following, with each scenario accompanied by example inputs:

- **User Inputs**:
  - Adding a new customer: Name, email, password.
  - Updating user information: Changing email or password. Example Input:



- **Item Inputs**:

  - Adding a new product: Product name, price, stock quantity.
  - Updating product information: Price changes, stock updates. Example Input:



- **Order and Shopping Cart Inputs**:
    - Adding items to the cart: Various products added to the cart.
    - Placing an order: Creating an order with products in the cart, entering payment and delivery details. Example Input:

```
ADDCUSTOMER Cemil Kerem kerem@yahoo.com 21.02.1993 100000 kerem1111
SHOWCUSTOMER Leyla 3
SHOWCUSTOMERS Demet
SHOWCUSTOMERS Ferit
ADDCUSTOMER Musa Ayten ayten@yahoo.com 05.10.1981 1000 ayten0000
ADDTOCART 7 37
ORDER 7 ayten0000
SHOWCUSTOMER Can 4
SHOWADMININFO Can
SHOWADMININFO Enes
CREATECAMPAIGN Alper 23.03.2017 01.06.2017 BOOK 25
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 DEKSTOP 90
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 PERFUME 20
SHOWCAMPAIGNS 2
ADDTOCART 3 10
ADDTOCART 3 3
ADDTOCART 3 5
ADDTOCART 3 15
ADDTOCART 5 10
ORDER 3 hamza1234
DEPOSITMONEY 6 210.6
CHPASS 1 emre1234 emre12345678
```

- ## What were the special cases you tested?

  The special case tests in the project included:

    - **Invalid or Missing Inputs**: I tested how the system handled incomplete or incorrect entries in fields like user information, product details, or order information.
    - **Out-of-Stock Products:** I tested the scenario where a user tries to add a product to the cart that is not in stock.
    - **Overload Scenarios:** I tested scenarios that would create a high load on the system, such as many users accessing the system simultaneously.

- ## Did everything work as expected?
    - During the testing process, I encountered significant issues with outputs. In some cases, the system did not generate the expected outputs correctly. This was especially critical in terms of data representation and the accuracy of operation results. I need to conduct additional tests and code corrections to resolve these issues.

- **Include sample input/output from your program.**
    - Input:

```
ADMIN\tDemet\tdemet@hacettepe.edu\t11121989\t3500\tdemet1234
CUSTOMER\tEmre\temre@hacettepe.edu\t02102000\t0\temre1234
TECH\tFatih\tfatih@hacettepe.edu\t17031992\t2100\t0
CUSTOMER Meltem meltem@hacettepe.edu 30.03.1990 500.40 meltem1234
CUSTOMER Hamza hamza@hacettepe.edu 08.09.1987 10321.5 hamza1234 ADMIN
Alper alper@hacettepe.edu 19.12.1991 3500 alper1234
TECH Emir emir@hacettepe.edu 28.02.1983 2700 1
ADMIN Can can@hacettepe.edu 21.05.1980 3450 can1234
ADMIN Leyla leyla@hacettepe.edu 01.11.1975 3600 leyla1234
ADMIN Cemil cemil@hacettepe.edu 06.07.1985 3750 cemil1234
TECH Handan handan@hacettepe.edu 29.10.1989 2700 1
CUSTOMER Taha taha@hacettepe.edu 29.04.1969 7505.43 taha1234 CUSTOMER
Furkan furkan@hacettepe.edu 30.09.1974 153.85 furkan1234 TECH Enes
enes@hacettepe.edu 02.02.1996 2100 0
```

```
BOOK 2 2016 Everything We Keep Lake Union Kerry Lonsdale 306
BOOK 25 1992 Ulysses Modern Library James Joyce 844
BOOK 3 2006 Nick and Norah's Infinite Playlist Alfred Knopf Books Rachel Cohn,David Levithan 183 BOOK 4 1960 To Kill a Mockingbird J. B.
Lippincott & Co. Harper Lee 281
BOOK 7 2004 Sorcery and Cecelia or The Enchanted Chocolate Pot HMH Books Patricia C. Wrede,Caroline Steverme,Ayn Rand 336 CDDVD 10 2
Gary Clark Jr. Grinder,Our Love,When My Train Pulls In,Church
CDDVD 5 2008 One of the Boys Katy Perry One Of The Boys,I Kissed A Girl,Thinking Of You,If You Can Afford Me CDDVD 13 2012 Red Taylo
Swift State Of Grace,Red,I Almost Do
CDDVD 21 1959 Kind of Blue Miles Davis So What,Freddy Freeloader
CDDVD 18 1957 The Great Ray Charles Ray Charles The Ray,The Man I Love,Hornful Soul
DESKTOP 1250 Micro-Star International MSI 220 250 Free-Dos Intel Core i5 8 750 black DESKTOP 1430 AsusTek Computer Inc. ASUS 220 2
Home Intel Core i7 16 1000 white DESKTOP 1000 AsusTek Computer Inc. ASUS 220 250 Free-Dos Intel Core i3 8 500 red DESKTOP 2100 Dell
250 Windows 10 Home Intel Core i7 16 2000 black DESKTOP 2400 Apple Inc. APPLE 220 250 MAC OS X Yosemite Intel Core i5 8 500 white LA
AsusTek Computer Inc. ASUS 220 250 Windows 10 Home Intel Core i7 16 750 1 LAPTOP 3400 Apple Inc. APPLE 220 250 MAC OS Intel Core i5
LAPTOP 1800 Hewlett-Packard Company HP 220 250 Free-Dos Intel Core i7 16 300 1 LAPTOP 1700 AsusTek Computer Inc. ASUS 220 250 Win
Home Intel Core i3 8 500 0 LAPTOP 2050 Dell Inc. DELL 220 250 Free-Dos Intel Core i7 16 400 1
TABLET 90 AsusTek Computer Inc. ASUS 220 100 Android 4.4 (KitKat) Qualcomm Quad-core 1 8 9 TABLET 135 Samsung Electronics SAMSUN
```

```
ADDCUSTOMER Cemil Kerem kerem@yahoo.com 21.02.1993 100000 kerem1111
SHOWCUSTOMER Leyla 3
SHOWCUSTOMERS Demet
SHOWCUSTOMERS Ferit
ADDCUSTOMER Musa Ayten ayten@yahoo.com 05.10.1981 1000 ayten0000
ADDTOCART 7 37
ORDER 7 ayten0000
SHOWCUSTOMER Can 4
SHOWADMININFO Can
SHOWADMININFO Enes
CREATECAMPAIGN Alper 23.03.2017 01.06.2017 BOOK 25
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 DEKSTOP 90
CREATECAMPAIGN Leyla 21.03.2017 01.09.2017 PERFUME 20
SHOWCAMPAIGNS 2
ADDTOCART 3 10
ADDTOCART 3 3
ADDTOCART 3 5
ADDTOCART 3 15
ADDTOCART 5 10
ORDER 3 hamza1234
DEPOSITMONEY 6 210.6
CHPASS 1 emre1234 emre12345678
```

- Output: The output cannot be provided yet because there are still deficiencies in the code and the code is not completed.

COMMENTS

- **Describe the overall result of the assignment.**
    - The overall result of the assignment was a partially successful development of the "MyBazaar" online shopping site. The project encompassed establishing a class hierarchy for users and products, handling file-based data, and creating a command processing system. Although key structures and functions were implemented, there were challenges in accurately generating expected outputs and fully addressing specific user scenarios. This indicates the need for further development and testing to fully realize all functionalities and ensure reliable system performance.

- **Was the programming project a success?**
    - The programming project achieved partial success. While it successfully established basic structures and functionalities for the "MyBazaar" online shopping site, it faced challenges in accurately producing outputs and handling complex user scenarios.

- **What would you do same or differently next time?**

  For future projects, I would maintain the modular and hierarchical design approach, as it facilitated organized and scalable development. However, I would implement more comprehensive testing strategies from the beginning, including unit testing and user scenario testing, to identify and address issues earlier in the development process. Additionally, I would focus on enhancing error handling and data validation mechanisms to ensure robustness against various input types and scenarios. These changes are aimed at improving the overall reliability and functionality of the system.