# Software Project Phase II

Keisi Breshanaj
Loard Bejko
Dejvi Kocilja
Ifigjenia Sopiqoti

**ii.i. Development Model:**
Developing a Supermarket ERP System requires a systematic and structured development process. The EasyMart Supermarket ERP System is a project with well predefined requirements, which is why we decided to implement the Waterfall development model for our system. The waterfall model allows us to divide our work into distinct phases and provide us with a clear structure of our project.

The Waterfall development model consists of five phases: Requirements analysis and definition, System and Software design, Implementation and Unit testing, Integration and system testing, Operation and maintenance.

1) Requirements analysis and definition: In this phase, the team will gather and define requirements and user expectations which includes managers, cashiers and administrators. We will analyze user requirements and also (if applicable) previous systems to understand user requirements completely. In this phase the team will also establish technical specifications such as database requirements, frameworks, designs etc. By the end of phase one we should have obtained a detailed documentation of functional and non-functional requirements for our ERP system.
2) System and Software Design: Our next phase consists of creating the system architecture and design based on the gathered user requirements. The development team will design documents such as UML diagrams, ER diagrams and interface mockups and prototypes. The design phase will provide a complete overall system architecture, user interfaces etc. We will gather feedback in order to make modifications if necessary, before jumping on to the next phase.
3) Implementation and Unit testing: In this phase, the team will start coding based on specific design specifications for different modules such as sales, inventory etc. to build the EasyMart ERP system. We will make sure that integration between modules will be completed for seamless functionality. Security measures and error handling mechanisms will be integrated and implemented. By the end of the implementation phase we will test individual code units to verify functionality and behavior. We will closely monitor and record test results in order to investigate any error that might occur and fix any failure.

4) Integration and system testing: For this next phase the team will test how modules interact with each other and how seamless their integration and functionality is in order to ensure proper data flow across the system. We will also be testing the system as a whole and evaluate how it meets specific requirements as expected in a real-world environment.
5) Operation and maintenance: As per the last phase, the team will deploy the ERP system into production. This phase consists of user training, system monitoring and we will also be involved in the ongoing support and maintenance of the system. Maintenance involves bug fixes and making sure that the system runs smoothly and meets all user requirements.

The Waterfall development model is a linear approach to system development. Each phase of the project must be completed before ongoing to the next one. This allows us to concentrate and dedicate our energy completely in each phase of the project development.

**ii.ii. User Requirements:**
The EasyMart software is crafted with a user-centric approach, ensuring a seamless and efficient experience for different users be it cashiers, managers, or administrators. Here are some user requirements:

a. Cashier requirements:
    - Secure login and logout functionality
    - Swift and accurate transactions processing
    - Ability to scan product barcodes and translate them into valuable product information
    - The possibility to handle every payment method whether it is a card or cash payment
    - Printing receipts
    - Being able to check loyalty points for each customer
    - Performing end of day sales reports
b. Manager requirements:
    - Secure login/logout with role based authentication
    - Inventory management tools that allow the management to add/remove products in real time, tracking stock etc.,
    - Sales management features that allows the management to monitor and analyze sales
    - Staff oversight features that allow scheduling, task assignment etc.,
    - Management tools for the loyalty points program
c. Administrator requirements:
    - Manage user accounts and access permissions
    - System configuration settings such as tax rates, currency rates, any applicable discount

- Ability to track system changes and user functions
- Database management procedures
- Security features that include data encryption and user authentication protocols
- Maintenance tools

The specified user requirements, which are our first phase of the project, will guide the software development for our EasyMart Supermarket ERP system. These build the groundwork for an ERP system that ensures to meet every specific need of each user.

**ii.iii. Functional Requirements**

a. Brief description:

- User-friendly design: Our system is not hard to operate. Cashiers, managers and administrators can use it simply without needing extra training.
- Distinct Modules: Our software combines several modules, where each one focuses on specific tasks for different roles in the supermarket.
- Loyalty Points System: This system is made for rewarding and promoting consumers by their past purchases.
- The ability to adapt: As the supermarket changes and grows, this software is built to adapt to these changes by implementing new features.

b. Acceptance Criteria

- User-friendly interface:
    1) With just three clicks or less, you can access your necessary functionalities.
    2) Buttons and menus are simple to use.
- Distinct Modules:
    1) Module of Cashier: The cashier is able to handle sales transactions precisely without the system breaking down.
    2) Module of the Manager: The system provides the necessary tools to manage inventory, by keeping track of sales and supervising employees.
    3) Module of the Administrator: The administrator is able to manage all user accounts.
- Loyalty Points System:
    1) Based on their past history of purchases, users can earn loyalty points.
    2) The total amount of loyalty points is displayed correctly.
    3) Smooth point redemption during sales transactions.

- The ability to adapt:
  1) The capacity of the system to handle big transactions without experiencing a drop in performance.
  2) It is simple to adapt changes while also interfering with regular supermarket operations.
  3) Regular updates to develop new technologies needed for new industry standards.
- Managing Data:
  1) A database will be kept up to date by the system.
  2) Mechanisms for data recovery are in place so there is no data loss.
- Security:
  1) Strong security measures to guard user data.
  2) Different users have different access to features and information.

**ii.iv. Non-functional Requirements**

a. Brief Description:

- Performance: The system should be able to handle multiple users simultaneously without slowing down.
- Reliability: The system should work at all times and recover quickly from errors.
- Security: Only authorized users should be able to access important information.
- Flexibility: The system must be able to withstand growing loads and adapt to change.
- Usability: All users should be able access the system easily.
- Maintenance: To continue operation, the system must be simple to fix and update.

b. Acceptance Criteria:

- Performance:
  1) Common tasks should respond in less than 2 seconds.
  2) The system should support at least 100 users at once.
- Reliability:
  1) 99.9% uptime is expected from the system, with planned maintenance resulting in least amount of downtime.
  2) If a breakdown occurs, the system needs to recover without deleting any data in 5 minutes.
- Security:
  1) Since there is sensitive information, data encryption methods should be applied.
  2) Role-based permissions should be enforced by access control.

- Flexibility:
    1) If transactions increase by 20%, the system should be able to continue its functionality.
    2) The system should allow new modules to be added without impacting the existing modules.
- Usability:
    1) For the system to be accessible, the user interface should support keyboard navigation and screen readers.
    2) For new users, training should be less than 1 hour.
- Maintenance:
    1) The code base should have clear comments.
    2) Applying updates should be easy and not cause any disturbance.

## ii.V. Application Specifications

a. Architecture:

```
+---------------------+
| Presentation Layer  |
+---------------------+
           |
           v
+---------------------+
| Application Layer   |
+---------------------+
           |
           v
+---------------------+
| Business Logic Layer|
+---------------------+
           |
           v
+---------------------+
| Data Access Layer   |
+---------------------+
           |
           v
+---------------------+
|      Database       |
+---------------------+
```

1. Presentation Layer:

This layer is responsible for presenting information to the users and gathering input from them. It includes the user interface components developed using JavaFX, such as screens, forms, buttons, and menus.
The presentation layer interacts with the application layer to retrieve data and trigger actions based on user input.

2. Application Layer:

The application layer serves as an intermediary between the presentation layer and the business logic layer.
It contains the logic for processing user requests, handling business rules, and coordinating interactions between different components of the system.
This layer interacts with the presentation layer to receive user input and with the business logic layer to execute business processes.

3. Business Logic Layer:

This layer encapsulates the core business logic and rules of the supermarket app.
It includes functionalities such as client management, billing, product inventory management, user authentication, and authorization.
The business logic layer interacts with the data access layer to retrieve and manipulate data from the database.

4. Data Access Layer:

The data access layer is responsible for accessing and manipulating data stored in the database.
It includes components such as data access objects (DAOs) or repositories that abstract the underlying database operations.
This layer interacts directly with the database to perform CRUD (Create, Read, Update, Delete) operations on data entities.

5. Database:

The database stores persistent data related to clients, bills, products, users, and other entities.
It may utilize a file-based storage system as indicated by the ".ser" files mentioned earlier, or it could be a relational database management system (RDBMS) such as MySQL, PostgreSQL, or SQLite.

b. Database Model:

- Clients.ser: This file stores information about clients or customers who interact with the supermarket app. It likely includes details such as client ID, name, contact information, and any other relevant data.

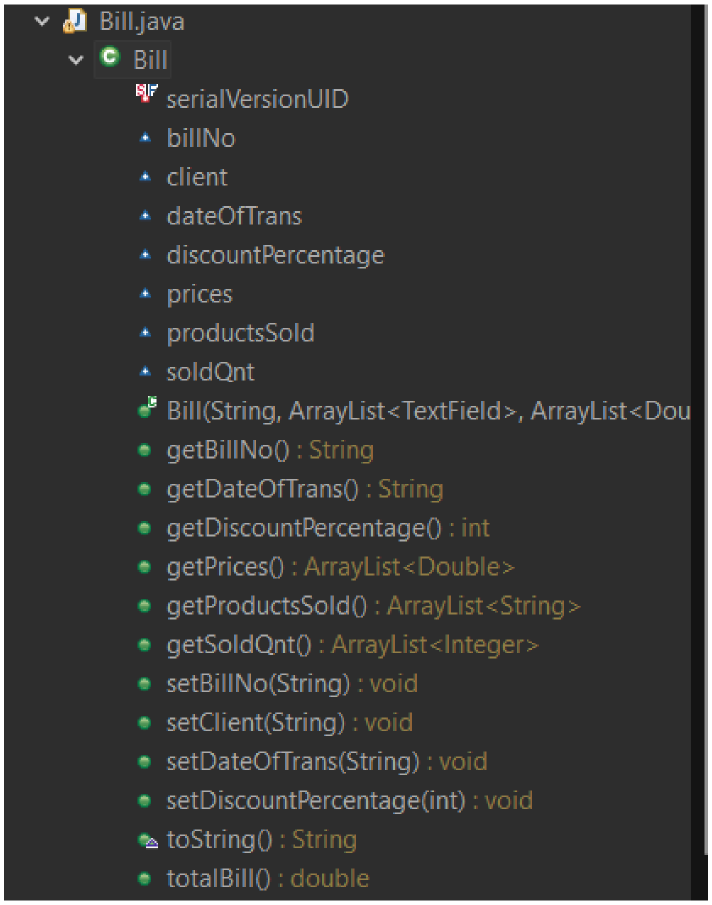
- Bills.ser: The "Bills" file contains records of transactions made by clients, including details such as bill ID, date and time of purchase, items purchased, quantities, prices, and total amount.

- Products.ser: This file serves as a repository for product information available in the supermarket. It likely includes details such as product ID, name, description, price, stock quantity, and any other attributes related to the products offered for sale.

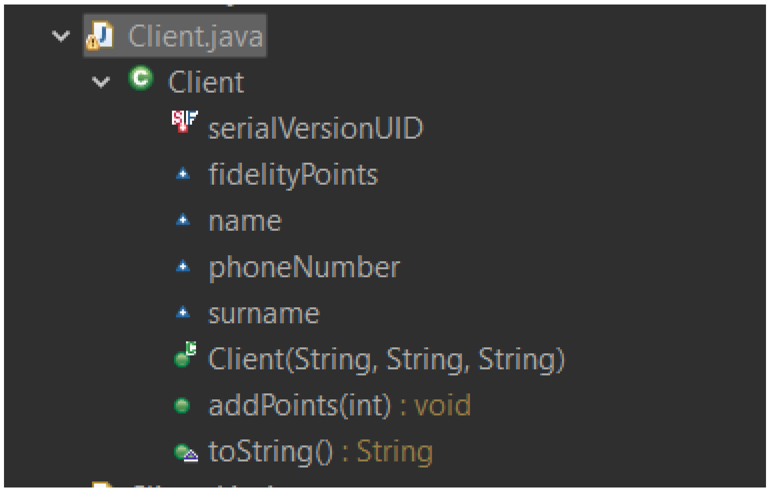
- Users.ser: The "Users" file is responsible for storing information about users who have access to the supermarket app. This may include user IDs, usernames, hashed passwords, roles or permissions, and any other relevant user-related data.

Constraints and relationships between these files may vary depending on the specific requirements of the application. For example:

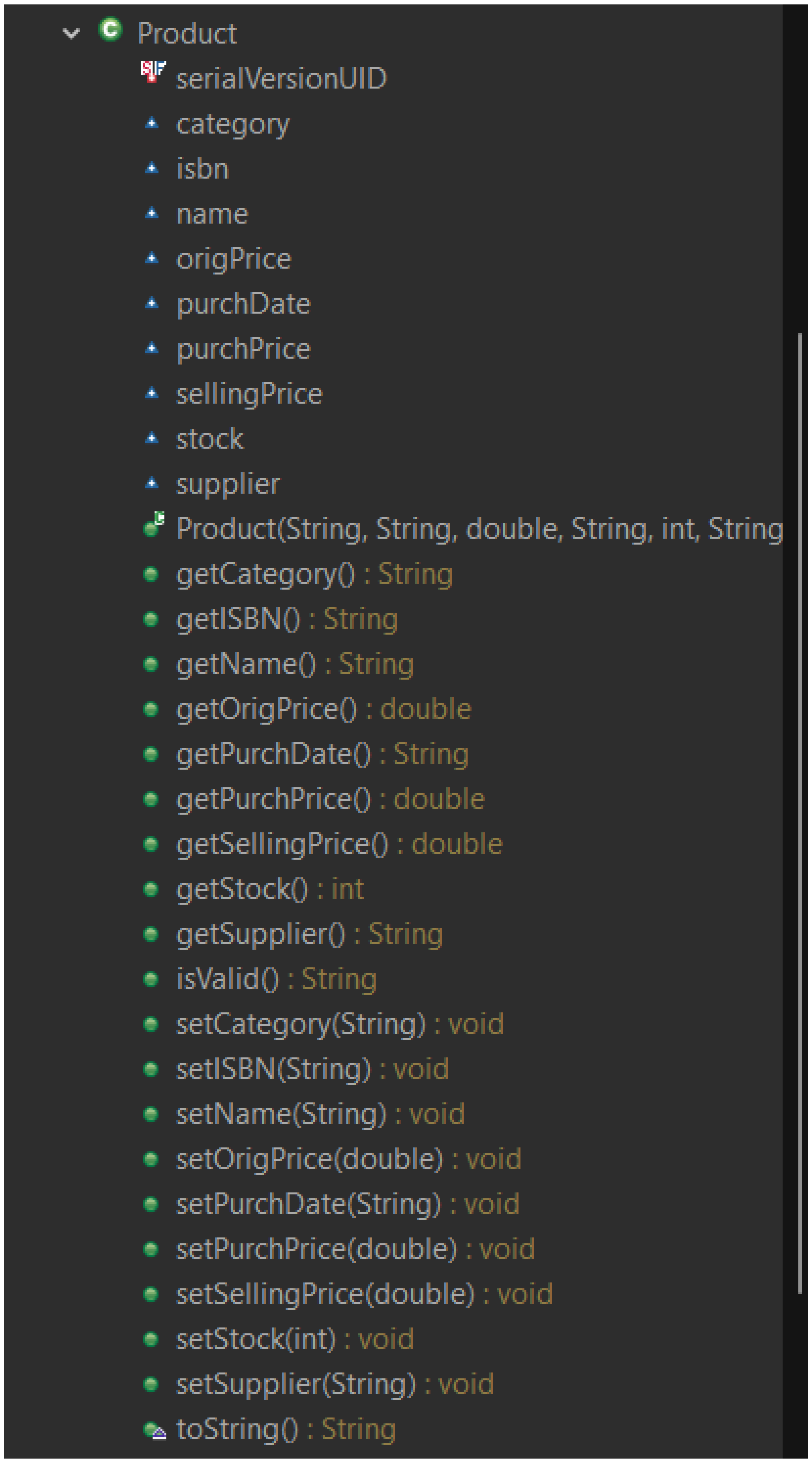
The "Clients" file may have a one-to-many relationship with the "Bills" file, indicating that each client can have multiple bills associated with them.
The "Bills" file may have a many-to-many relationship with the "Products" file, indicating that each bill can contain multiple products, and each product can be included in multiple bills.
The "Users" file may have constraints such as unique usernames to ensure each user has a distinct login identifier.



```
Bill.java
  Bill
    serialVersionUID
    billNo
    client
    dateOfTrans
    discountPercentage
    prices
    productsSold
    soldQnt
    Bill(String, ArrayList<TextField>, ArrayList<Dou
    getBillNo() : String
    getDateOfTrans() : String
    getDiscountPercentage() : int
    getPrices() : ArrayList<Double>
    getProductsSold() : ArrayList<String>
    getSoldQnt() : ArrayList<Integer>
    setBillNo(String) : void
    setClient(String) : void
    setDateOfTrans(String) : void
    setDiscountPercentage(int) : void
    toString() : String
    totalBill() : double
```



```
Client.java
  Client
    serialVersionUID
    fidelityPoints
    name
    phoneNumber
    surname
    Client(String, String, String)
    addPoints(int) : void
    toString() : String
```



```
Product
    serialVersionUID
    category
    isbn
    name
    origPrice
    purchDate
    purchPrice
    sellingPrice
    stock
    supplier
    Product(String, String, double, String, int, String
    getCategory() : String
    getISBN() : String
    getName() : String
    getOrigPrice() : double
    getPurchDate() : String
    getPurchPrice() : double
    getSellingPrice() : double
    getStock() : int
    getSupplier() : String
    isValid() : String
    setCategory(String) : void
    setISBN(String) : void
    setName(String) : void
    setOrigPrice(double) : void
    setPurchDate(String) : void
    setPurchPrice(double) : void
    setSellingPrice(double) : void
    setStock(int) : void
    setSupplier(String) : void
    toString() : String
```

c. Technologies Used:

For this project, we chose to use Java as the programming language because it's widely supported and known for its reliability. For the front-end design, we turned to JavaFX, a tool that helps us create user-friendly interfaces with buttons, menus, and other interactive elements. By using Java and JavaFX together, we're able to build a solid and visually appealing application that meets our needs.

- Cross-platform compatibility: Java is known for its platform independence, allowing the app to run on various operating systems without requiring major modifications.

- Rich user interface: JavaFX provides a robust set of tools for creating visually appealing and interactive user interfaces.

- Scalability: JavaFX is well-suited for building scalable applications, including enterprise-level solutions like a supermarket management system.

- Integration with backend systems: JavaFX seamlessly integrates with Java backend technologies, such as Spring Boot or Java EE, enabling smooth communication between the front-end and backend components of the application.

d. User Interface Design:
Showcase wireframes, mockups, or describe the user interface:
Provide a visual representation of how users will interact with the system.

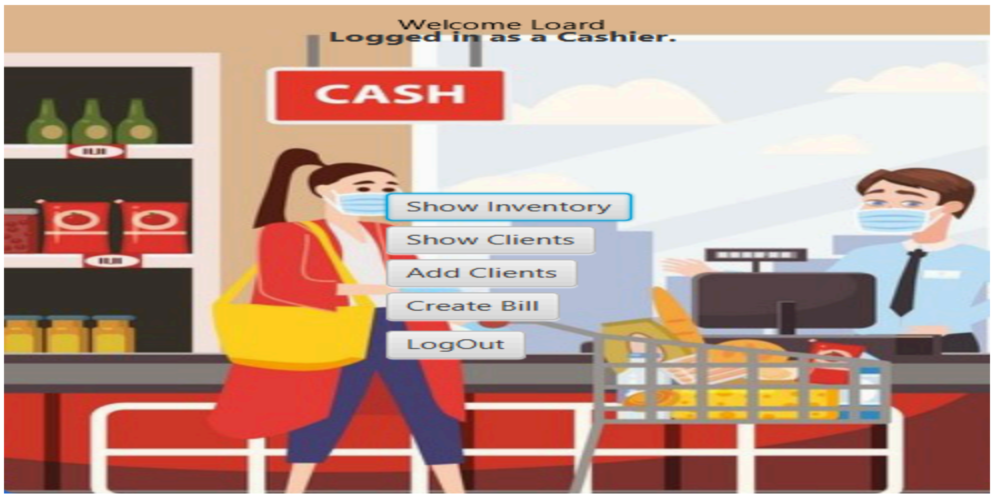
e. Security Measures:

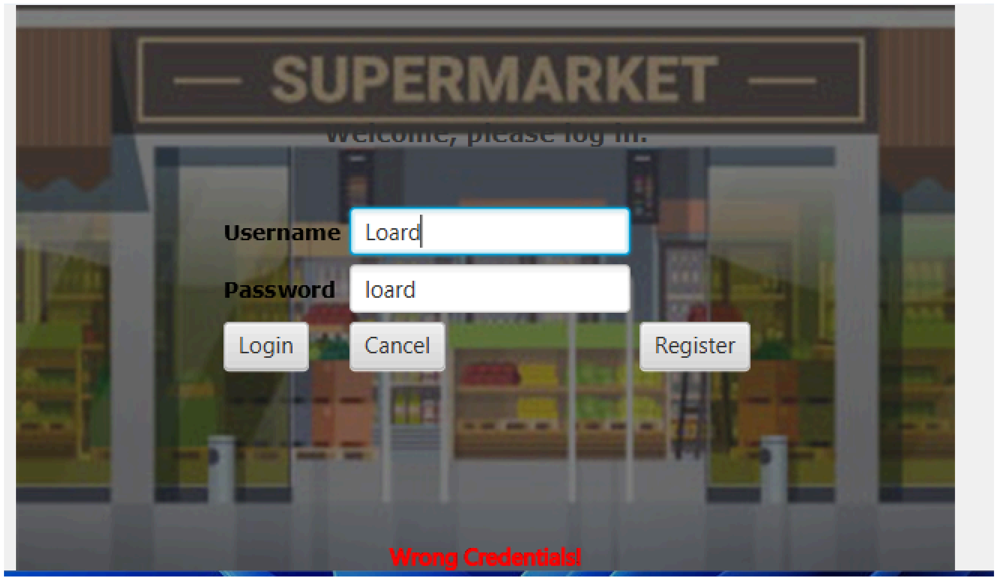
1.Authentication: Implementing authentication with username and password is a fundamental security measure to ensure that only authorized users can access the supermarket app. This helps prevent unauthorized access to sensitive data and functionalities.

2. Password Hashing: Hashing passwords is a critical step in protecting user credentials. By hashing passwords, you ensure that even if the database is compromised, attackers cannot directly access the plaintext passwords. Instead, they would need to perform a time-consuming brute-force attack to reverse-engineer the original passwords from the hashes.

3. Input Validation: Validating user input is essential for preventing common vulnerabilities such as SQL injection, cross-site scripting (XSS), and command injection. By validating and sanitizing input data, you reduce the risk of malicious exploitation of your application.

- Here are some visualizations on how the users interact with the system:

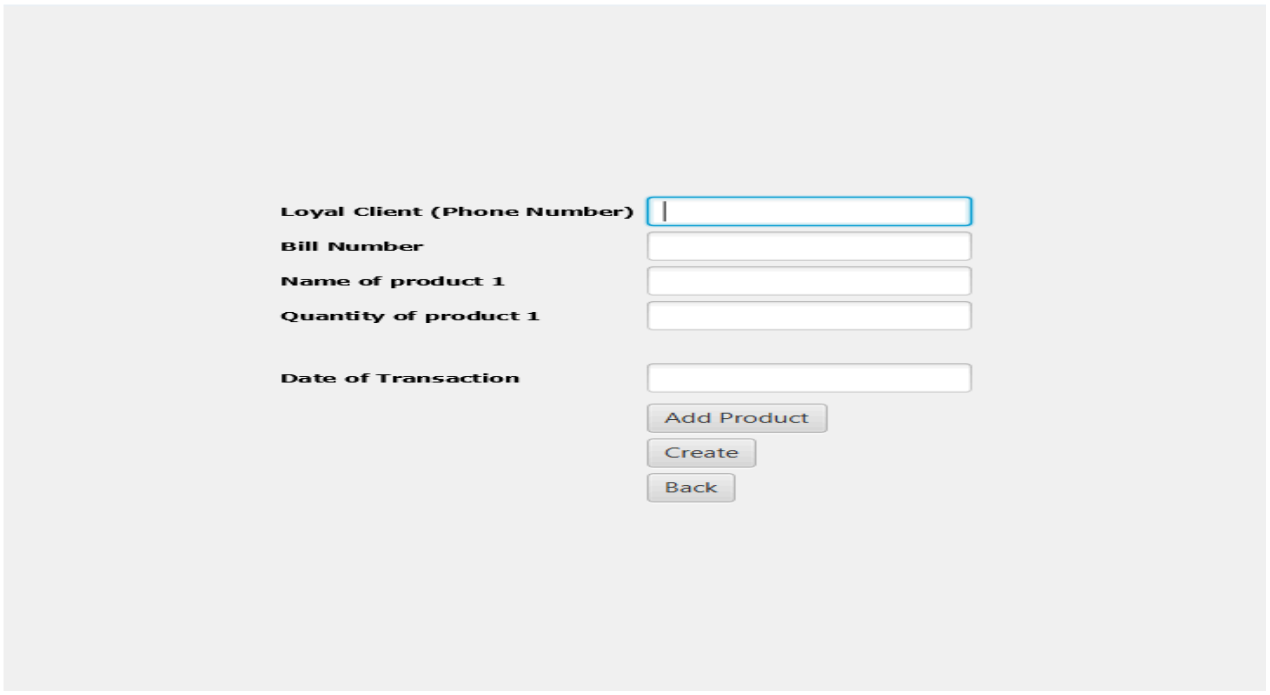




Username Loard

Password loard

Wrong Credentials!



| | |
|---|---|
| Loyal Client (Phone Number) | |
| Bill Number | |
| Name of product 1 | |
| Quantity of product 1 | |
| Date of Transaction | |

- This screenshot shows the registration of a specific client.
  If the registration is successful, you earn points based on your past purchases.