University Distrital Francisco Jose De Caldas

Nelson de Jesús Navarro de la Rosa 20242020116

Jefferson David Rico Ruiz 20242020108

Engineering of Systems

Report Workshops

Development of a buying and selling app based of TEMU Application

Introduction

An application inspired by the Temu model was developed, replicating its main features and adapting them to an academic and experimental approach. Developed as part of an academic exercise, this application incorporates features such as product navigation, category filtering, detailed item viewing, purchasing process simulation, and shipping status.

Motivation

The main motivation for carrying out this project arises from the constant changes brought about by the digital age, which have significantly transformed the way people carry out their daily activities. In particular, the way we shop, access products, and consume services has rapidly evolved thanks to mobile technologies and e-commerce platforms. This change in consumption habits has generated new needs, such as immediacy, ease of use, and accessibility from anywhere. Considering this reality, we decided to develop a Temu-like application, with the goal of exploring how to build a modern digital solution that responds to these new behaviors and expectations.

1. Requirements

In this section we define the objectives or functional requirements that describe behaviors or actions that the application should have and the non-functional requirements that seek to show characteristics that the application should have at a general level.

1.1 Functional requirements:

1. User Registration: The system must allow users to register by providing their name, email address, and password.
2. User login: The system must allow users to log in using their registered email and password.
3. Product exploration: Users must be able to browse products categorized by type and view individual product details.
4. Order Tracking: Users must be able to view the status of their orders (pending, shipped, delivered).
5. Product Management (Seller): The system must allow sellers to upload products with a name, description, image, and price.
6. Sales Overview (Seller): Sellers must be able to view a list of products sold and basic sales statistics.
7. Buy product: The app must allow the user to purchase the desired product.
8. Record payments and sales: The system must have a section where the seller can view payments and sales for their products.
9. Implement Promotions: The seller can set up promotions for a specific product.
10. Display offers: The user can view product offers upon entering the app.

1.2 No functional requirements:

1. An interactive and easy-to-use interface: The application's user and seller interfaces must be clear and intuitive.
2. Security: Ensure the security of the user's information and confidential data, such as their password or payment information.
3. Performance: The system must respond efficiently to user interaction, with agile processes and minimal response times.
4. Modularity: The code must be divided into independent modules to facilitate maintenance and scalability.

1.3 Changes

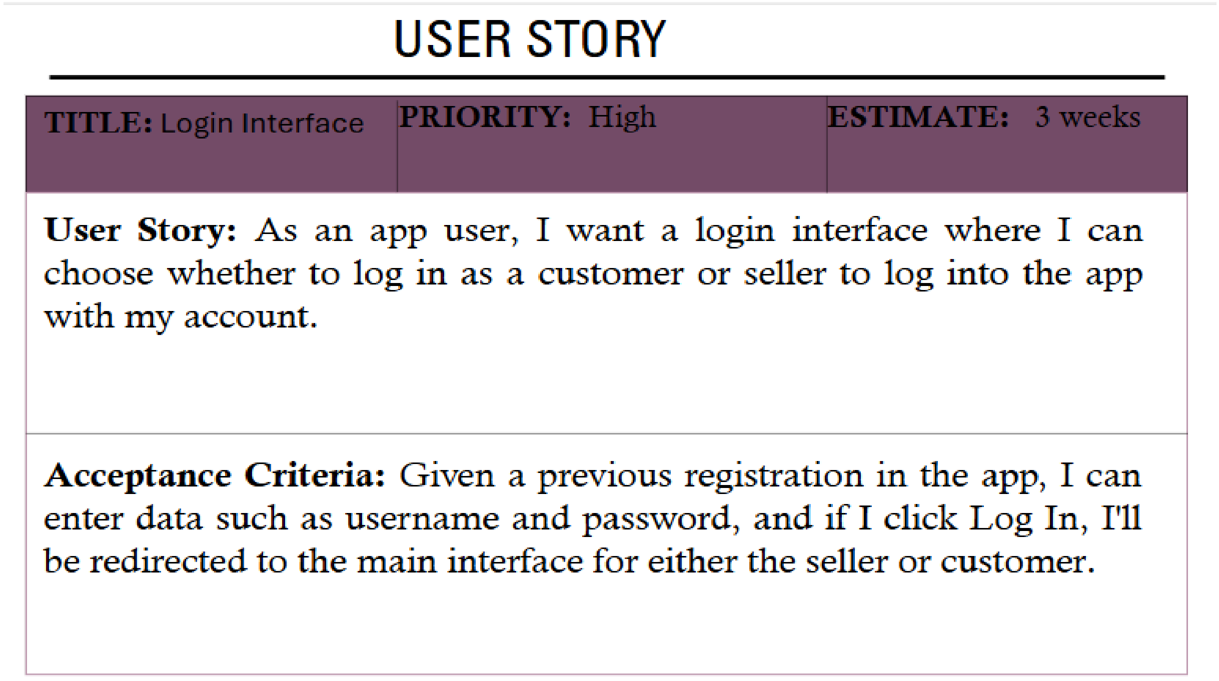
We analyzed the functional and non-functional requirements considering the solid principles and decided that it is not necessary to add more or make any modifications since they are the objectives or characteristics that our app must have.

2. User Stories

Here we create user stories that seek to briefly but specifically describe a functionality that the application will have, including aspects such as who it is aimed at, what it should do, with what objective, and when it will do it.

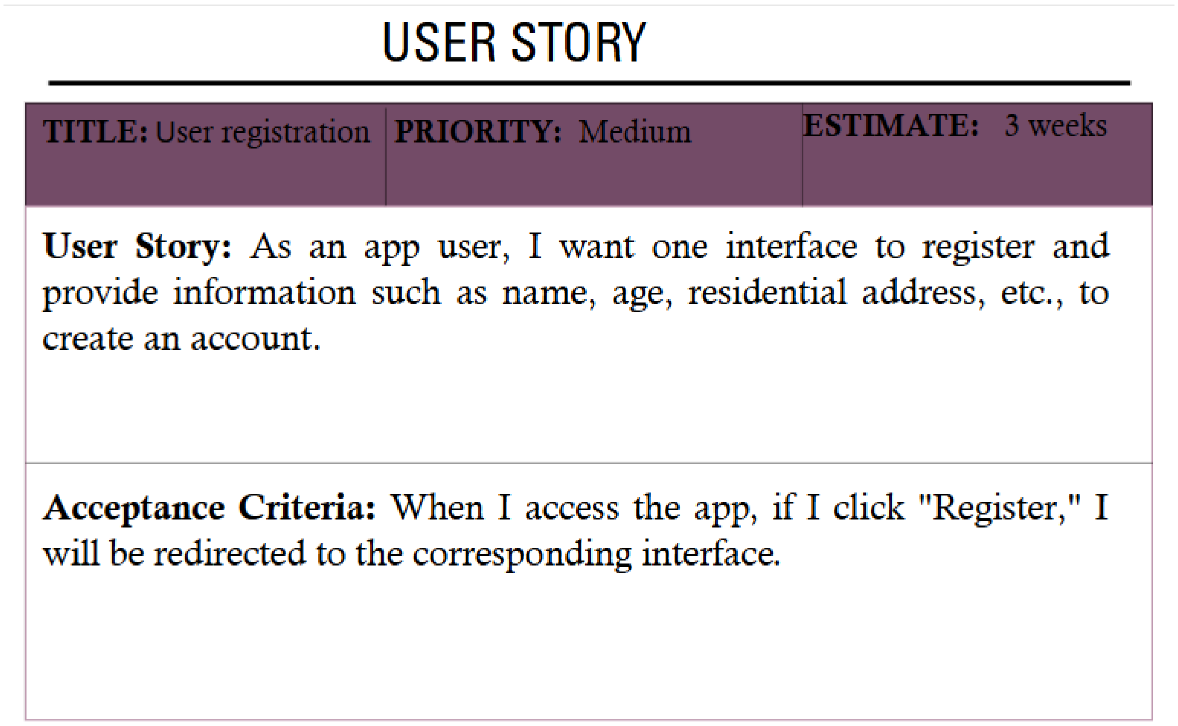
2.1 Changes

We analyzed the user stories and decided to create more of them, where we sought to ensure that each story had a clear functionality, explaining it more briefly, and adding new functionalities such as buttons and interfaces.



USER STORY

TITLE: Login Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As an app user, I want a login interface where I can choose whether to log in as a customer or seller to log into the app with my account.		
Acceptance Criteria: Given a previous registration in the app, I can enter data such as username and password, and if I click Log In, I'll be redirected to the main interface for either the seller or customer.		



USER STORY

TITLE: User registration	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks
User Story: As an app user, I want one interface to register and provide information such as name, age, residential address, etc., to create an account.		
Acceptance Criteria: When I access the app, if I click "Register," I will be redirected to the corresponding interface.		

USER STORY

TITLE: Main interface (Customer)	**PRIORITY:** High	**ESTIMATE:** 3 weeks
User Story: As an app user, I want to have a main interface that allows me to easily access the products I need through categories.		
Acceptance Criteria: Given a series of product categories, when the user interacts with them, he or she will be able to access the section of the selected category.		

USER STORY

TITLE: Products on Sale	**PRIORITY:** Low	**ESTIMATE:** 3 weeks
User Story: As a user, I want a window to appear with a discounted product upon entering the app. This is to attract attention and encourage people to buy.		
Acceptance Criteria: Given a few discounted products, choose one at random to display upon app launch. If the user interacts with the offer window, they'll be redirected to the discounted product. Otherwise, they'll be redirected to the initial interface.		

USER STORY

TITLE: Product Purchase Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a user, I want to be redirected to an interface that displays the product image, price, description, user reviews, and a "Buy Now" button . This allows the user to choose the product they want to purchase and obtain more information.

Acceptance Criteria: Once a product is selected, an interface will be displayed with the product and its information.

USER STORY

TITLE: Pay for product	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a buyer, I would like to be redirected to the payment methods when selecting the "Buy Now" button.

Acceptance Criteria: Once the product is selected, I can click the "Buy Now" button and then be redirected to the payment methods section.

USER STORY

TITLE: Payment methods	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As an app user, I would like to be able to choose between several payment methods when paying for a product, so I can use the one that's most convenient for me.

Acceptance Criteria: After choosing the payment method, I will be asked for the necessary information to complete the purchase.

USER STORY

TITLE: Return to home (button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a user, I want the "Return" button to redirect me to the main interface to continue interacting with the products.

Acceptance Criteria: Once the interface is deployed, when the user interacts with the button, it will redirect them to the main interface.

USER STORY

TITLE: Main (Seller) Interface	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a seller, I want a space where the products I sell are displayed, along with their price, image, and name. I also want a "Add Product" button and a "Sold Products" button to view them.

Acceptance Criteria: Once the user has registered as a seller, they will be redirected to this interface upon logging in.

USER STORY

TITLE: Add Product(button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a seller, I want a button on the main seller interface where I can add products to sell.

Acceptance Criteria: With the button implemented in the interface, if the seller interacts with it, they will be redirected to the section.

USER STORY

TITLE: Add Product Interface	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks

User Story: As a seller, I want a space where I can add the product to the application and enter data such as the name, price, image, and description, and assign it to the category to put the product for sale.

Acceptance Criteria: Given the add product button, when the seller interacts with it, they should be redirected to that interface.

USER STORY

TITLE: Publish Product(button)	**PRIORITY:** High	**ESTIMATE:** 3 weeks

User Story: As a seller, I want a "Publish Product" button to complete the product addition process.

Acceptance Criteria: With the "Add Product" interface, when the user interacts with the button, the product is published in the app, added to the seller's main interface, and redirected to that interface.

USER STORY

TITLE: Products sold (Button)	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks

User Story: As a seller user, I want the 'Sold Products' button to take me to the interface where all my sold products are so I can manage my sales.

Acceptance Criteria: Given the main interface of the seller, when the user interacts with the button, they will be directed to the respective sold products interface.

USER STORY

TITLE: Products sold Interface	**PRIORITY:** Medium	**ESTIMATE:** 3 weeks

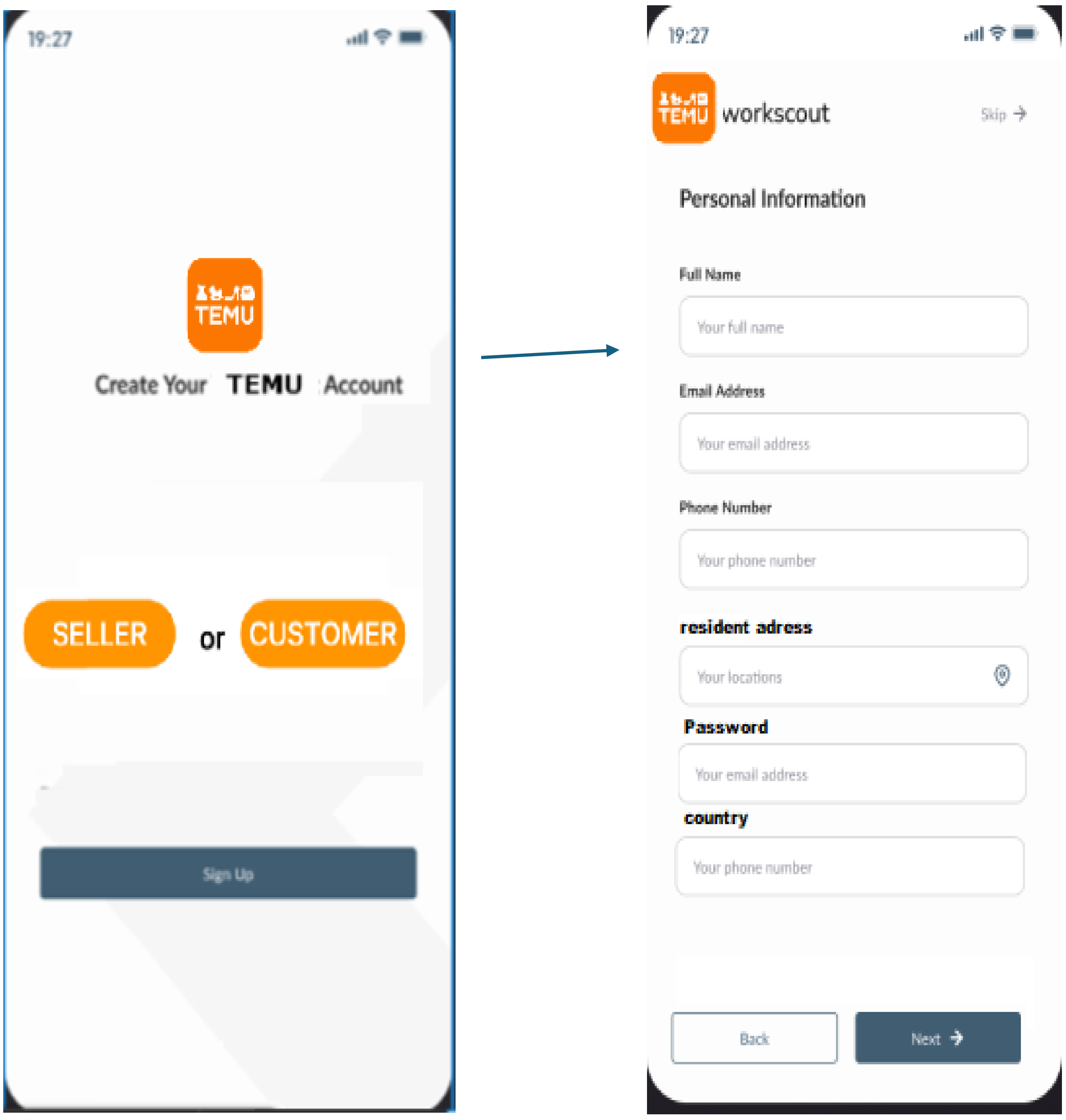
User Story: As a seller, I want an interface where I can see all the products I have sold in the application to manage my sales.

Acceptance Criteria: Given the main interface of the seller, when the seller interacts with the button, they will be directed to that interface.

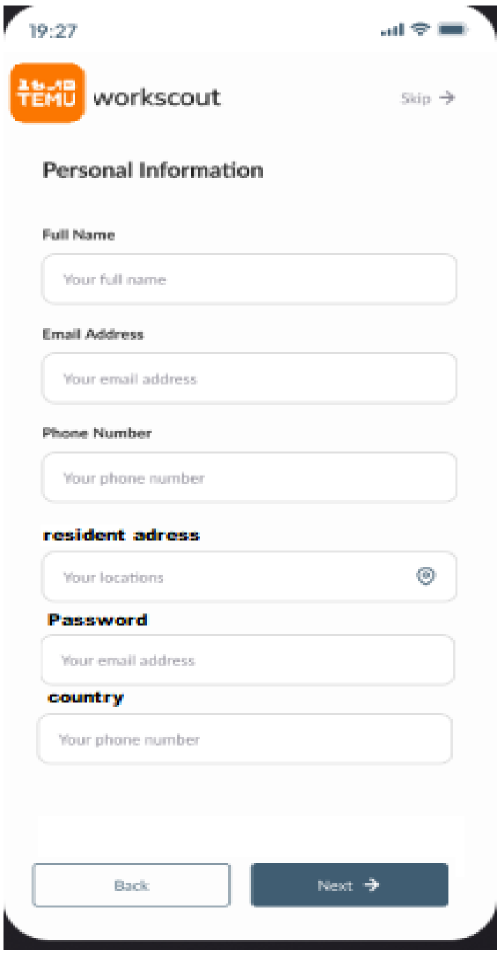
3. Mockups

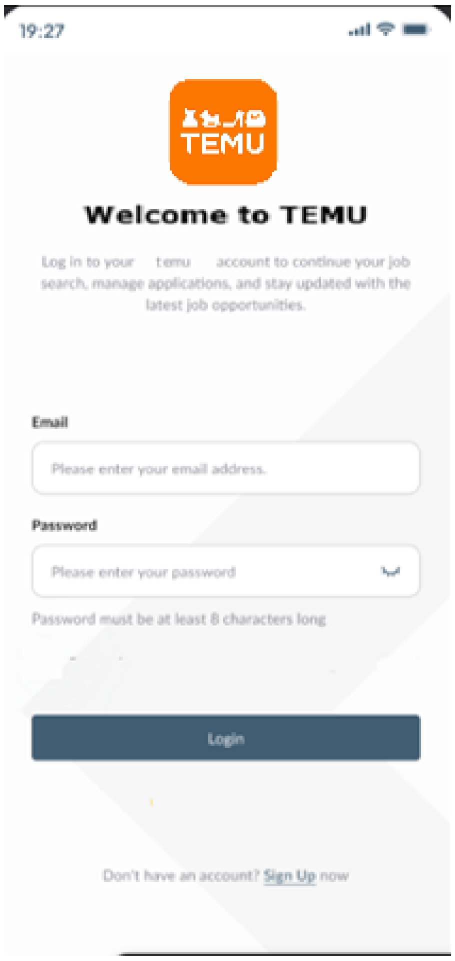
In this section we show the mockups for our application and the relationships between each of them.

3.1 Customer interfaces



Explanation: These two mockups are related since after selecting the role it redirects to the registration interface

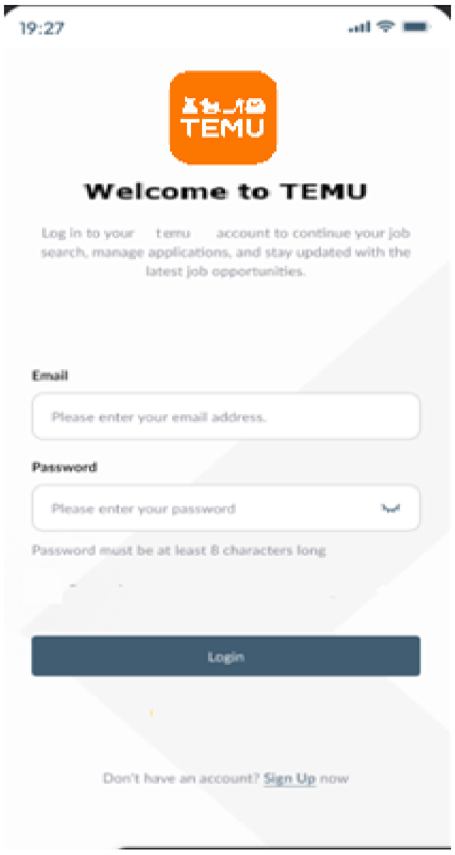



19:27
TEMU
Welcome to TEMU
Email
Please enter your email address.
Password
Please enter your password
Login
Don't have an account? Sign Up now

Log in to your temu account to continue your job search, manage applications, and stay updated with the latest job opportunities.

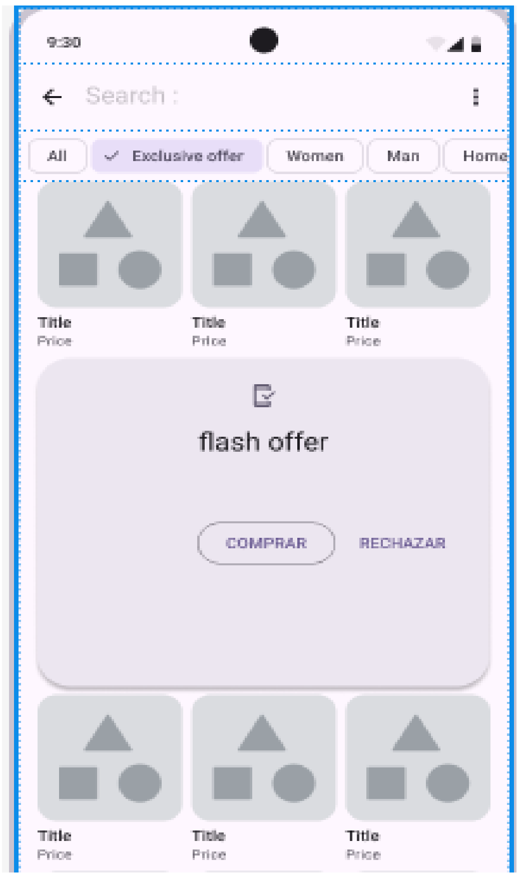
Password must be at least 8 characters long

Explanation: When the user registers, they will be redirected to the login interface.

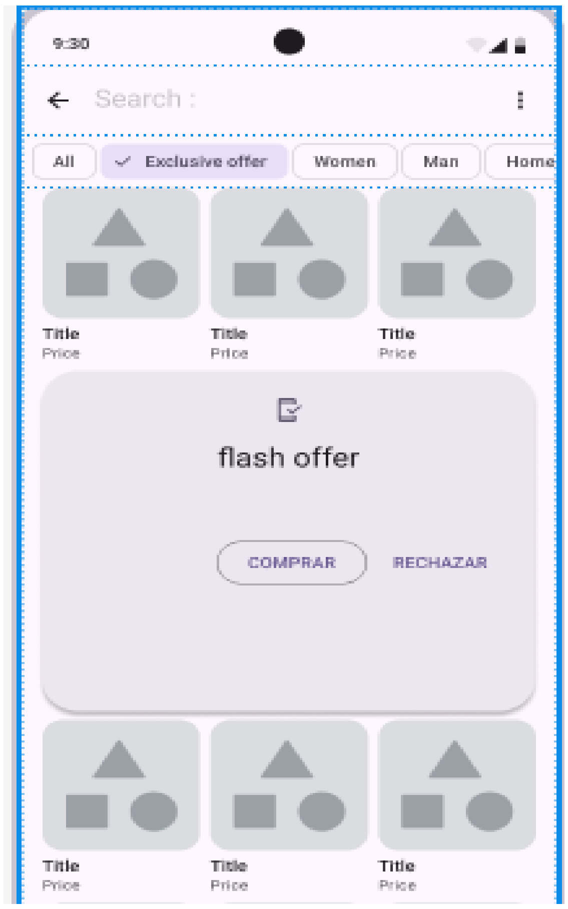

19:27
TEMU
Welcome to TEMU
Email
Please enter your email address.
Password
Please enter your password
Login
Don't have an account? Sign Up now

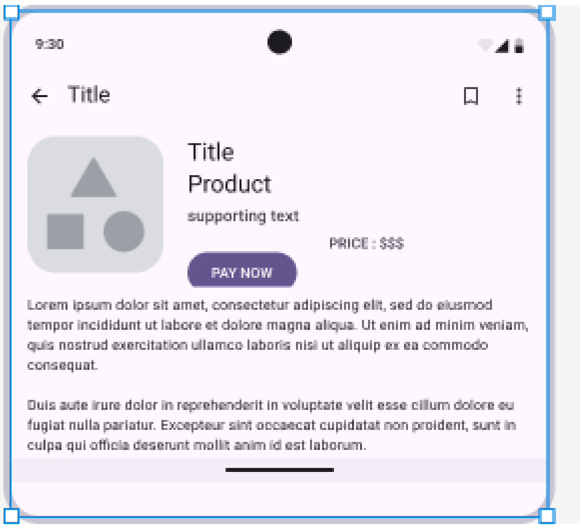
Log in to your temu account to continue your job search, manage applications, and stay updated with the latest job opportunities.

Password must be at least 8 characters long



Explanation: When the user logs in, an offer appears with a product from the app.

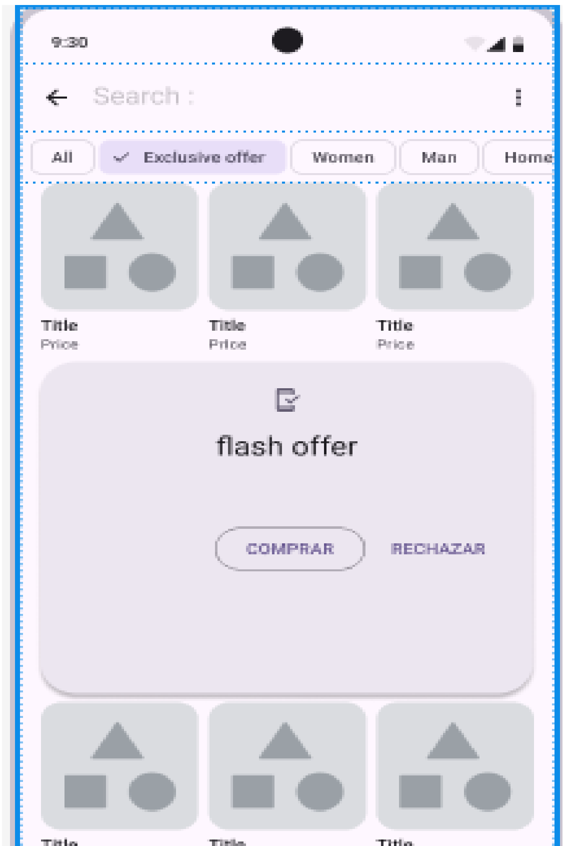


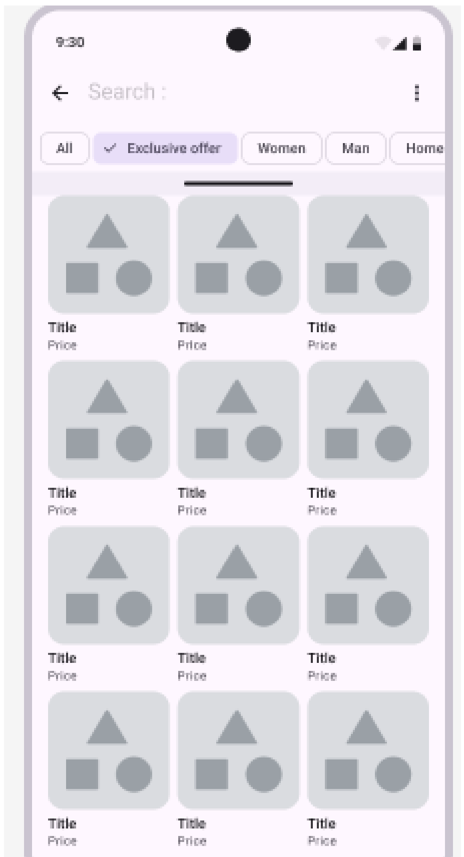


Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

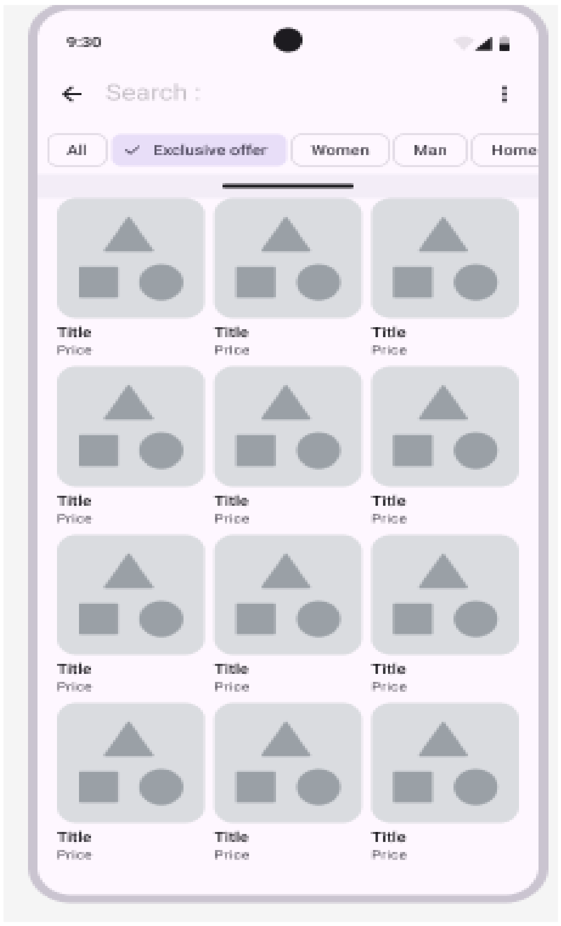
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

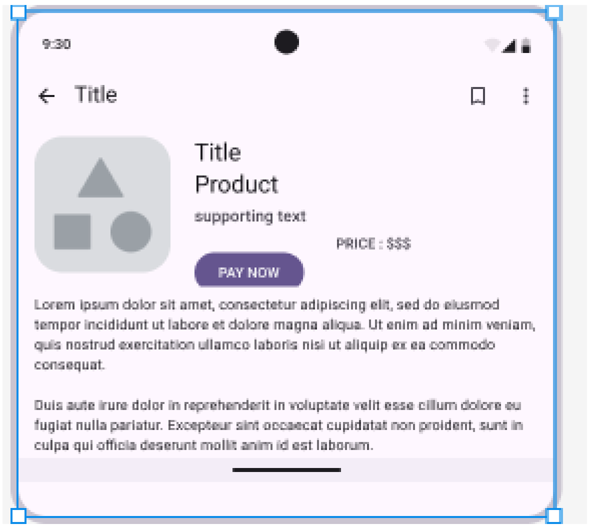
Explanation: If the user accepts the offer, they will be taken to the product.





Explanation: If the user does not accept the offer, they will be taken to the main interface.

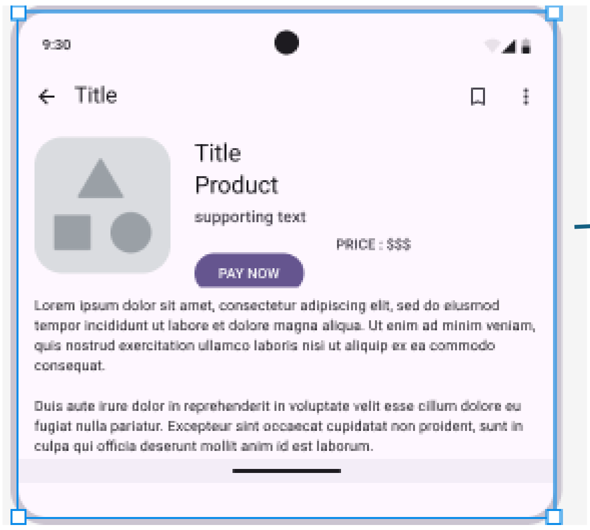




Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

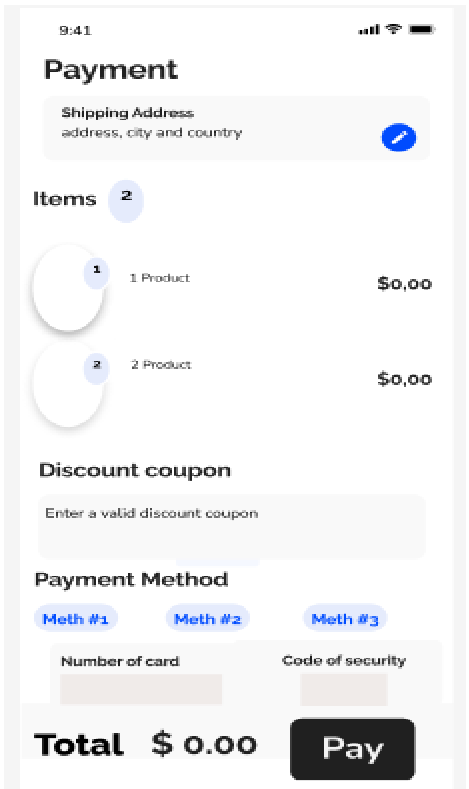
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Explanation: If the user selects a product, they will be taken to that product's interface.

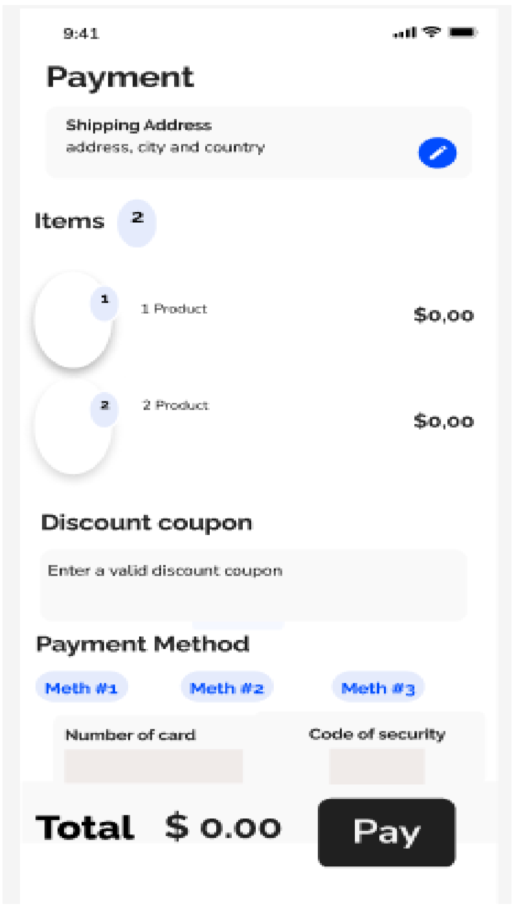


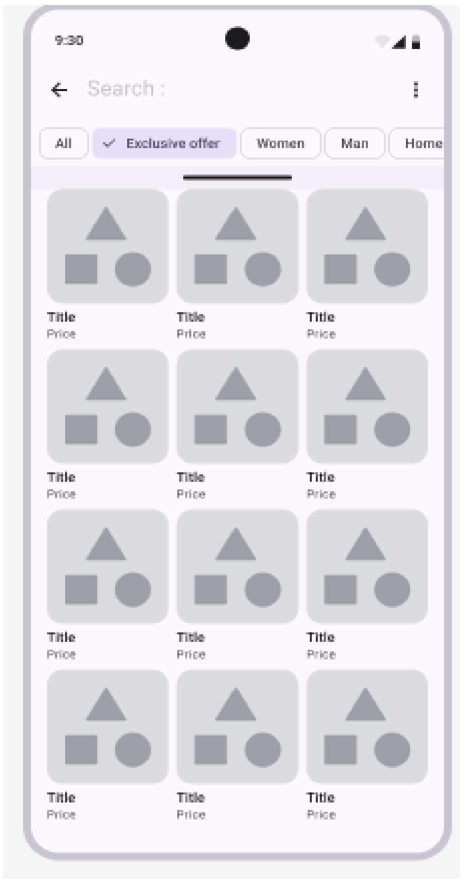
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



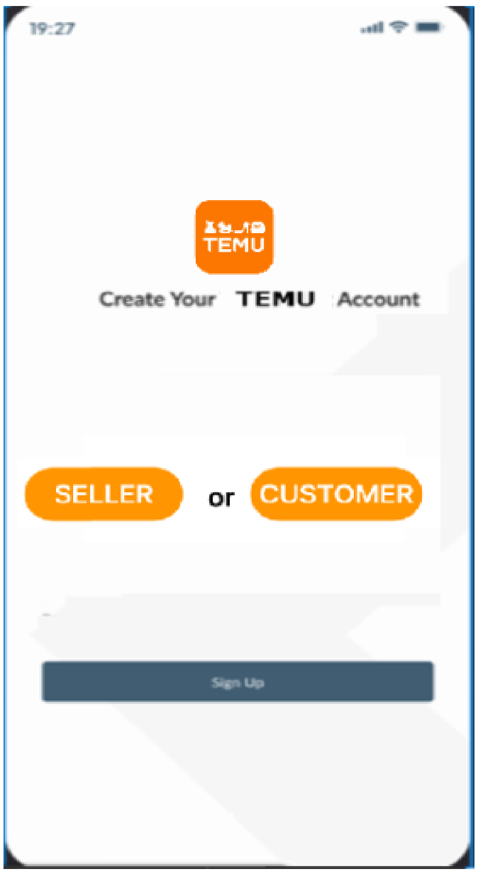
Explanation: If the user clicks on pay, they are redirected to the payment interface.

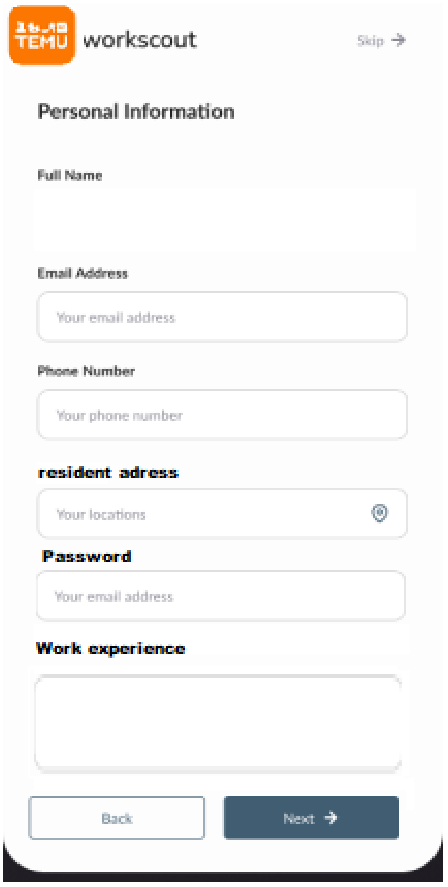




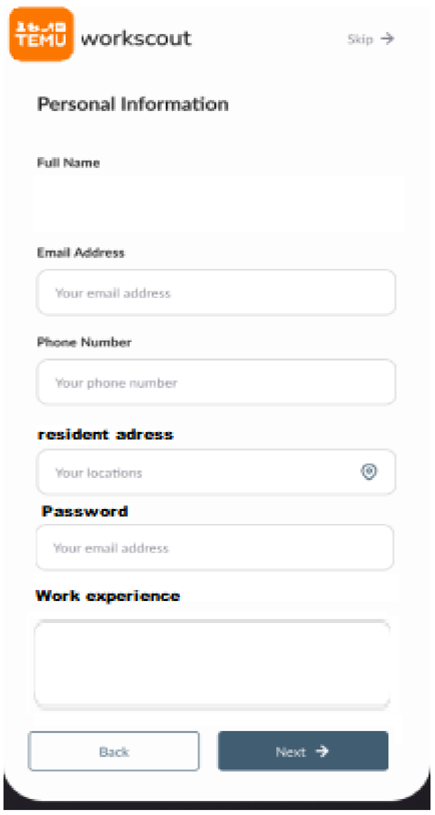
Explanation: When the user clicks on shutdown, he is redirected to the main interface.

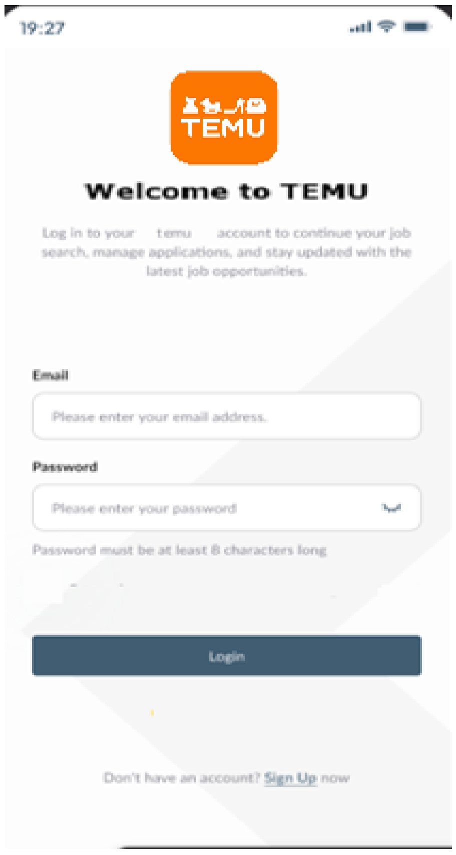
3.2 Seller interfaces





Explanation: If the user chooses the seller role, the registry will ask for the professional profile as additional information.

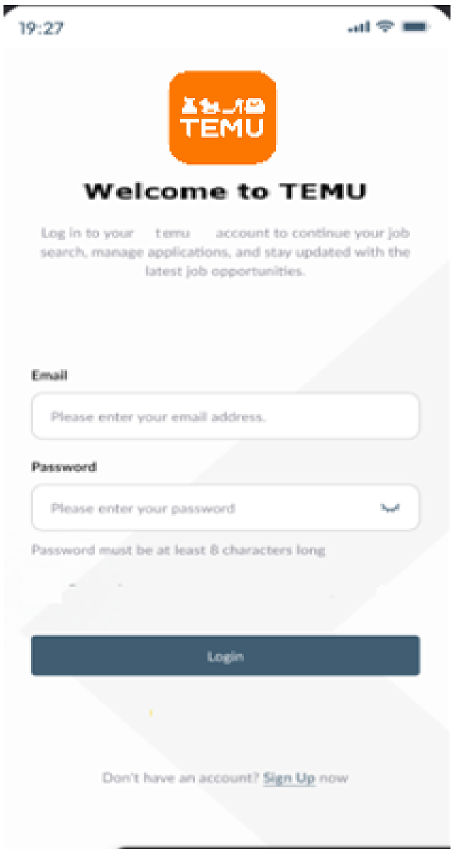



19:27
TEMU
Welcome to TEMU

Log in to your temu account to continue your job search, manage applications, and stay updated with the latest job opportunities.

Email
Please enter your email address.
Password
Please enter your password
Password must be at least 8 characters long
Login
Don't have an account? Sign Up now

Explanation: When the seller registers, he will be redirected to the login interface.


19:27

TEMU

Welcome to TEMU

Log in to your temu account to continue your job search, manage applications, and stay updated with the latest job opportunities.

Email

Please enter your email address.

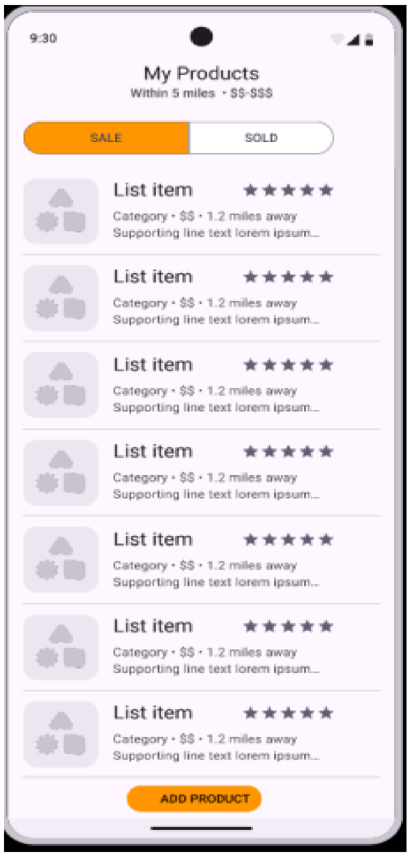
Password

Please enter your password

Password must be at least 8 characters long

Login

Don't have an account? Sign Up now


9:30

My Products

Within 5 miles · $$-$$$

SALE | SOLD

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

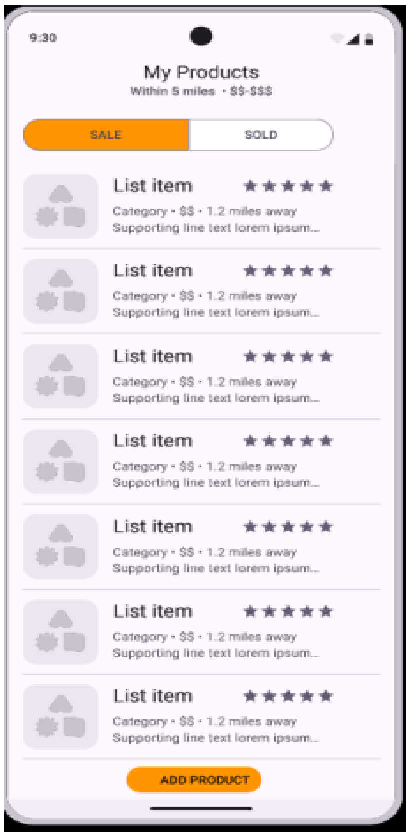
List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

ADD PRODUCT

Explanation: When the seller logs in, he will be redirected to the main interface.


9:30

My Products

Within 5 miles · $$-$$$

SALE | SOLD

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

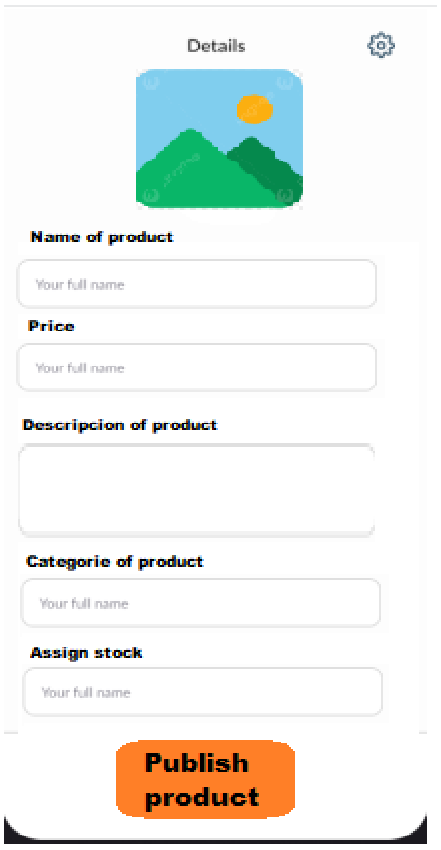
List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

List item ★★★★★
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...

ADD PRODUCT


Details

Name of product

Your full name

Price

Your full name

Descripcion of product

Categorie of product

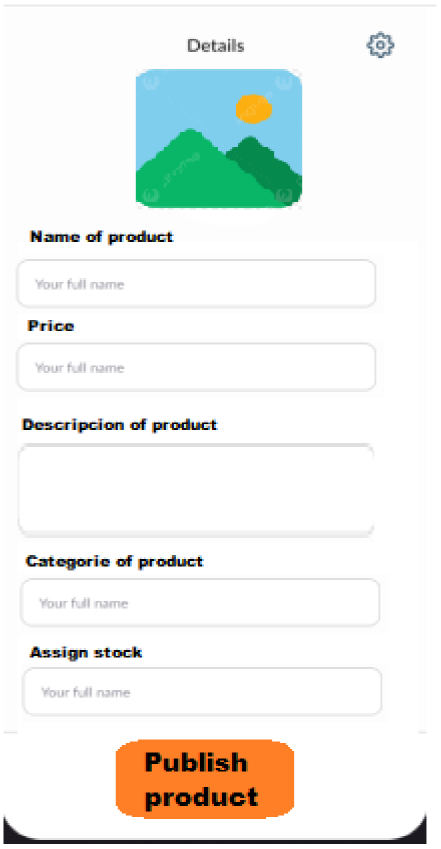
Your full name

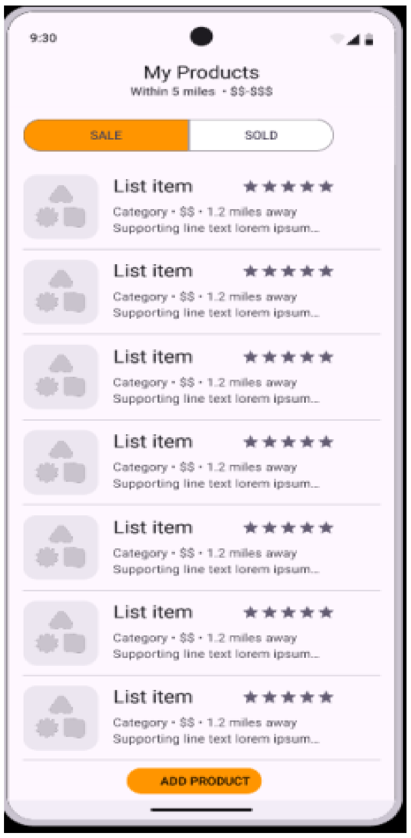
Assign stock

Your full name

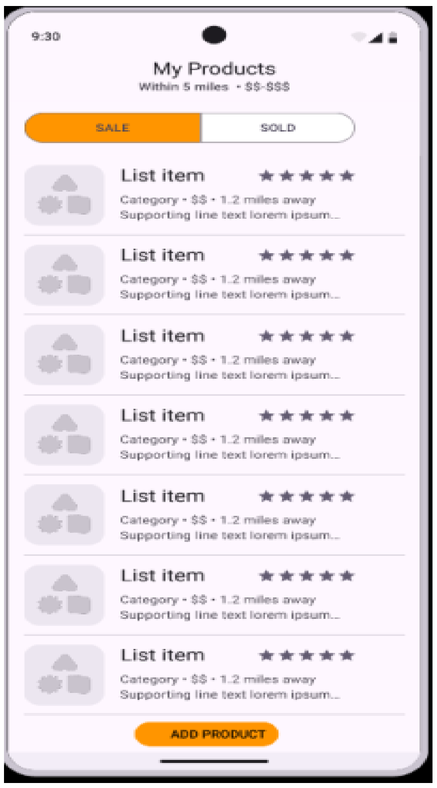
Publish product

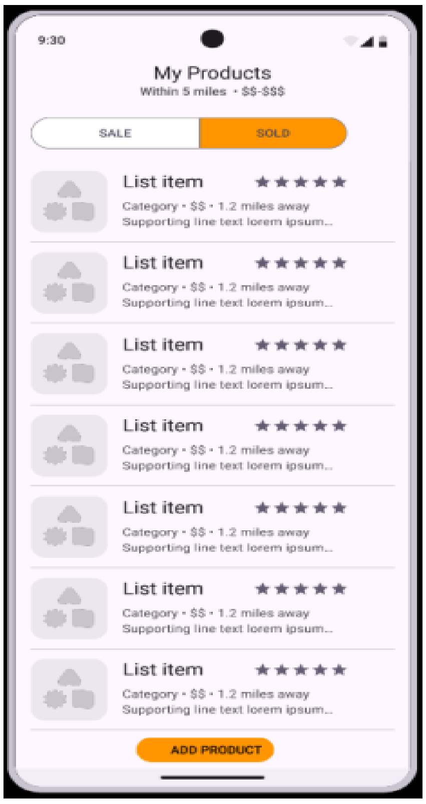
Explanation: When the seller clicks the add product button, he will be redirected to the add product interface.




9:30
My Products
Within 5 miles · $$-$$$
SALE
SOLD
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
ADD PRODUCT

Explanation: When the seller clicks the publish product button, it publishes it and redirects it to the main interface again.


9:30
My Products
Within 5 miles · $$-$$$
SALE
SOLD
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
ADD PRODUCT


9:30
My Products
Within 5 miles · $$-$$$
SALE
SOLD
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
List item
Category · $$ · 1.2 miles away
Supporting line text lorem ipsum...
ADD PRODUCT

Explanation: When the seller clicks the sold button, it will show the products he has sold.

4.3 Why the mockups?

Mockup Select Rol

We chose this mockup because it allows us to select whether the person wants to sell or buy products.

Mockup Interface User Login.

We chose this mockup because it shows the interface for logging in and registration if the user doesn't have an account associated with the app.

Mockup Offer Windows .

We chose this design because it has window of a product offering to generate customer attraction and interest.

Mockup Main interface (Customer).

We chose this design because it displays the products and categories in an orderly manner, giving an organized look to the main interface.

Mockup Product Interface.

We chose the design for mockup because the interface shows the product more specifically with its description and the respective payment button

Mockup Payment Methods Interface.

We chose the design of mockup because it is very complete when it comes to presenting the payment options, the discount coupon section, and the selected product, and the price along with the pay button.

Mockup Main interface (Seller)

We chose it because it allows the seller to see the products they have created

Mockup Add Product

We chose this design because it shows the necessary information to create a product in the app.

Mockup Sold products

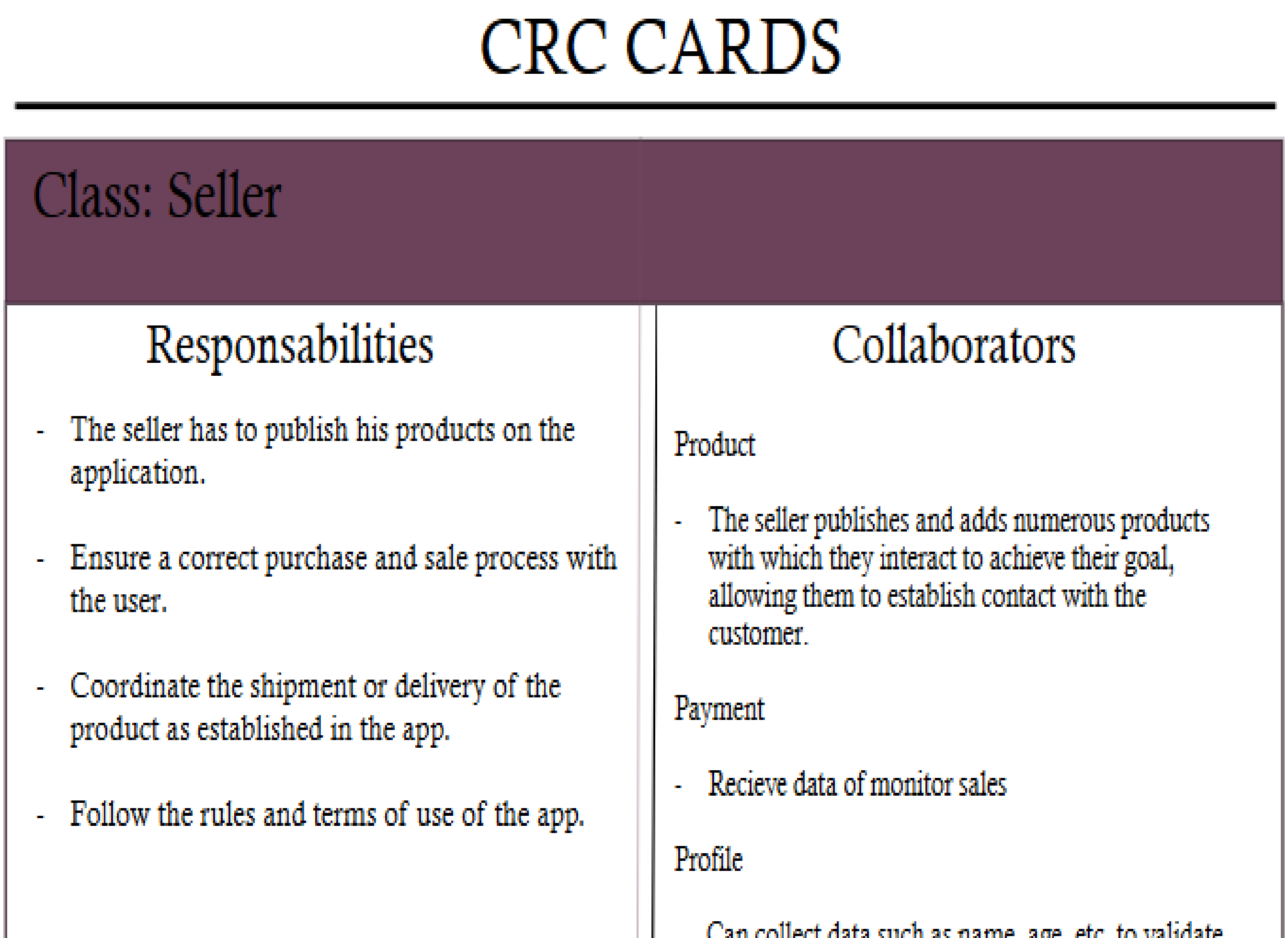
We chose this design because it shows the products the seller has sold.

4. Crcs Cards

In this section we present the main classes that the application will have, where we mention what roles or responsibilities they will perform, as well as the classes with which they collaborate and help fulfill their responsibilities and how they achieve this.

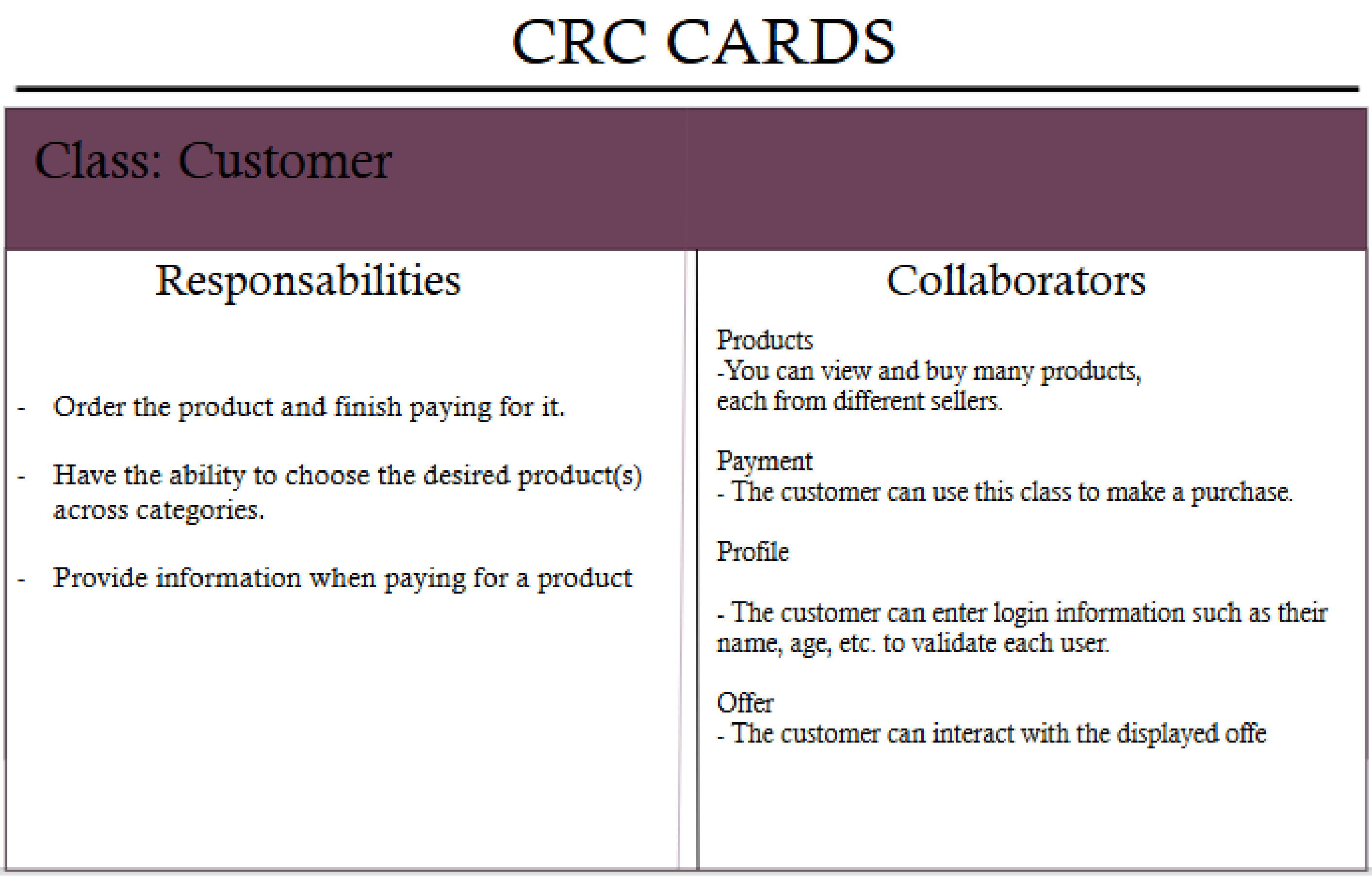
3.1 Changes

We analyzed CRC cards and sought to add more responsibilities to the classes. We decided to create a class registration and profile offering to comply with the single responsibility principle where each class has a clear responsibility.



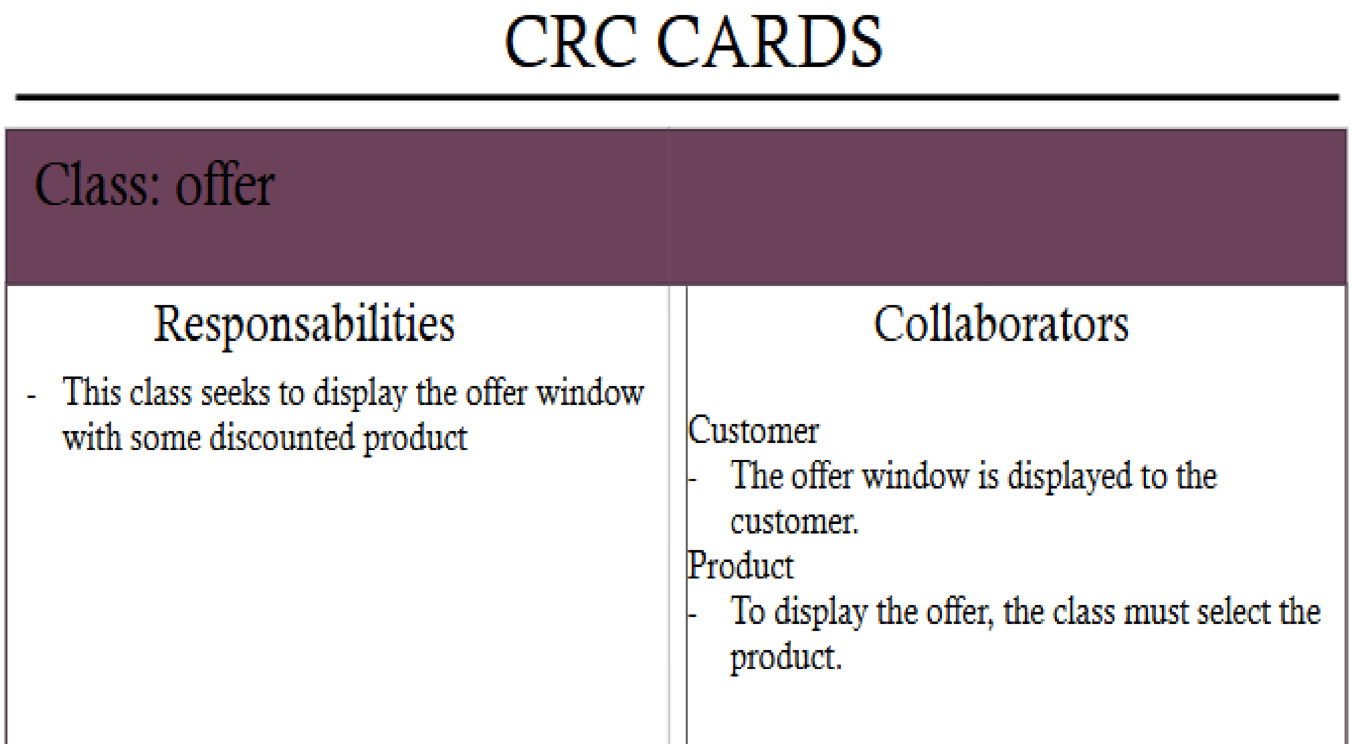
CRC CARDS

Class: Seller	
Responsabilities	**Collaborators**
- The seller has to publish his products on the application. - Ensure a correct purchase and sale process with the user. - Coordinate the shipment or delivery of the product as established in the app. - Follow the rules and terms of use of the app.	Product - The seller publishes and adds numerous products with which they interact to achieve their goal, allowing them to establish contact with the customer. Payment - Recieve data of monitor sales Profile Can collect data such as name, age, etc. to validate



CRC CARDS

Class: Customer	
Responsabilities	**Collaborators**
- Order the product and finish paying for it. - Have the ability to choose the desired product(s) across categories. - Provide information when paying for a product	Products -You can view and buy many products, each from different sellers. Payment - The customer can use this class to make a purchase. Profile - The customer can enter login information such as their name, age, etc. to validate each user. Offer - The customer can interact with the displayed offe



CRC CARDS

Class: offer	
Responsabilities	**Collaborators**
- This class seeks to display the offer window with some discounted product	Customer - The offer window is displayed to the customer. Product - To display the offer, the class must select the product.

CRC CARDS

Class: Product	
Responsabilities	**Collaborators**
- Have a photo, description, and price to provide information to the buyer - Know what type of item it is and, therefore, what category it should be associated with. - Going on offer eventually.	Seller - This person creates and manages product features. They can implement offers, modify prices, and modify images. Customer - The customer can view and purchase the product. Payment - The product can pass data to the Payment class, such as its price and image. Offer - The offer select a product

CRC CARDS

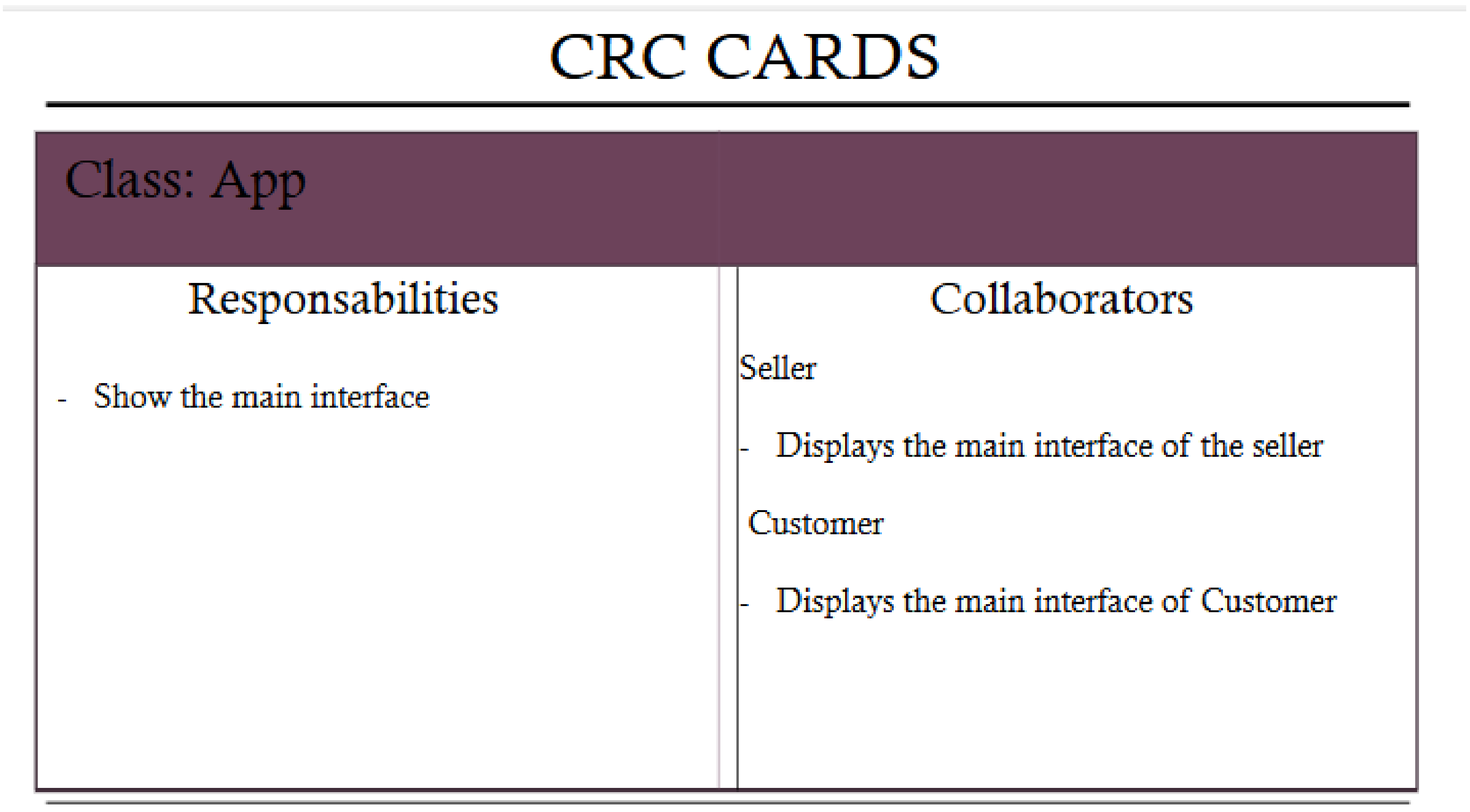
Class: Payment	
Responsabilities	**Collaborators**
- Record the sale of a certain product - Validate the data according to the card number, security number and amount of money on the card and discount the value of the product. - Calculate the new value to be paid depending on the discount percentage, if there is one.	Customer - Can provide information such as their name, home address, etc., for the payment process Product - Can provide the price and image of the item Seller - Receives payments made for this class and stores them in the sales record

CRC CARDS

Class: Register	
Responsabilities	Collaborators
- Collect information from the user	Profile - The register class passes user data to the profile class

CRC CARDS

Class: Profile	
Responsabilities	Collaborators
- Store the data supplied by the user.	Seller - Stores the data provided by the seller in the class. A registry is used to store it. Customer - Stores the data provided by the customer in the class. A registry is used to store it.


CRC CARDS

Class: App	
Responsabilities	Collaborators
- Show the main interface	Seller - Displays the main interface of the seller Customer - Displays the main interface of Customer

5. Packages directory

In this section we show and explain why we chose that folder distribution to develop the project.

The way we organize packages in Java is governed by model-view-controller architecture. We divide the project into three folders, each containing the classes responsible for managing these three aspects.

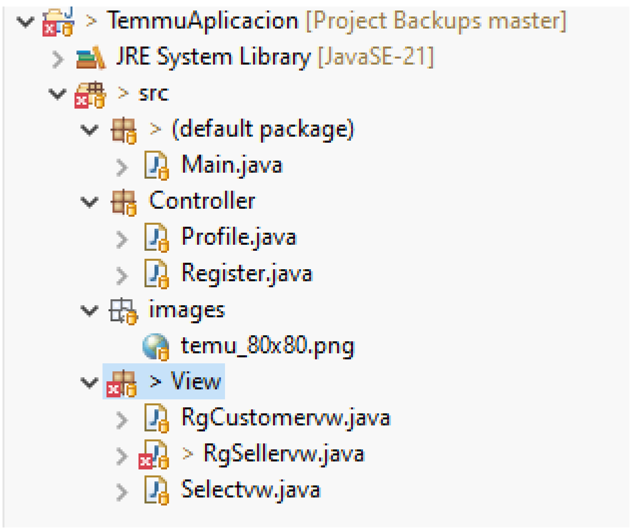
The controller contains classes responsible for managing the application logic and receiving data supplied by the user in the view.

The view is responsible for displaying graphical interfaces made up of different elements, such as text fields, buttons, etc., with which the user can interact and send requests to the controller when performing a specific action.

The model is responsible for managing the data and storing it, for example, in a database, so it can be accessed and validated.

In the package created by default, we have the main class, where we will call the class with which the application starts running.

Finally, we have an images folder where we will store images or icons for the application.



```
> TemmuAplicacion [Project Backups master]
  JRE System Library [JavaSE-21]
  > src
    > (default package)
      Main.java
    Controller
      Profile.java
      Register.java
    images
      temu_80x80.png
    > View
      RgCustomervw.java
      > RgSellervw.java
      Selectvw.java
```

6. UML diagrams

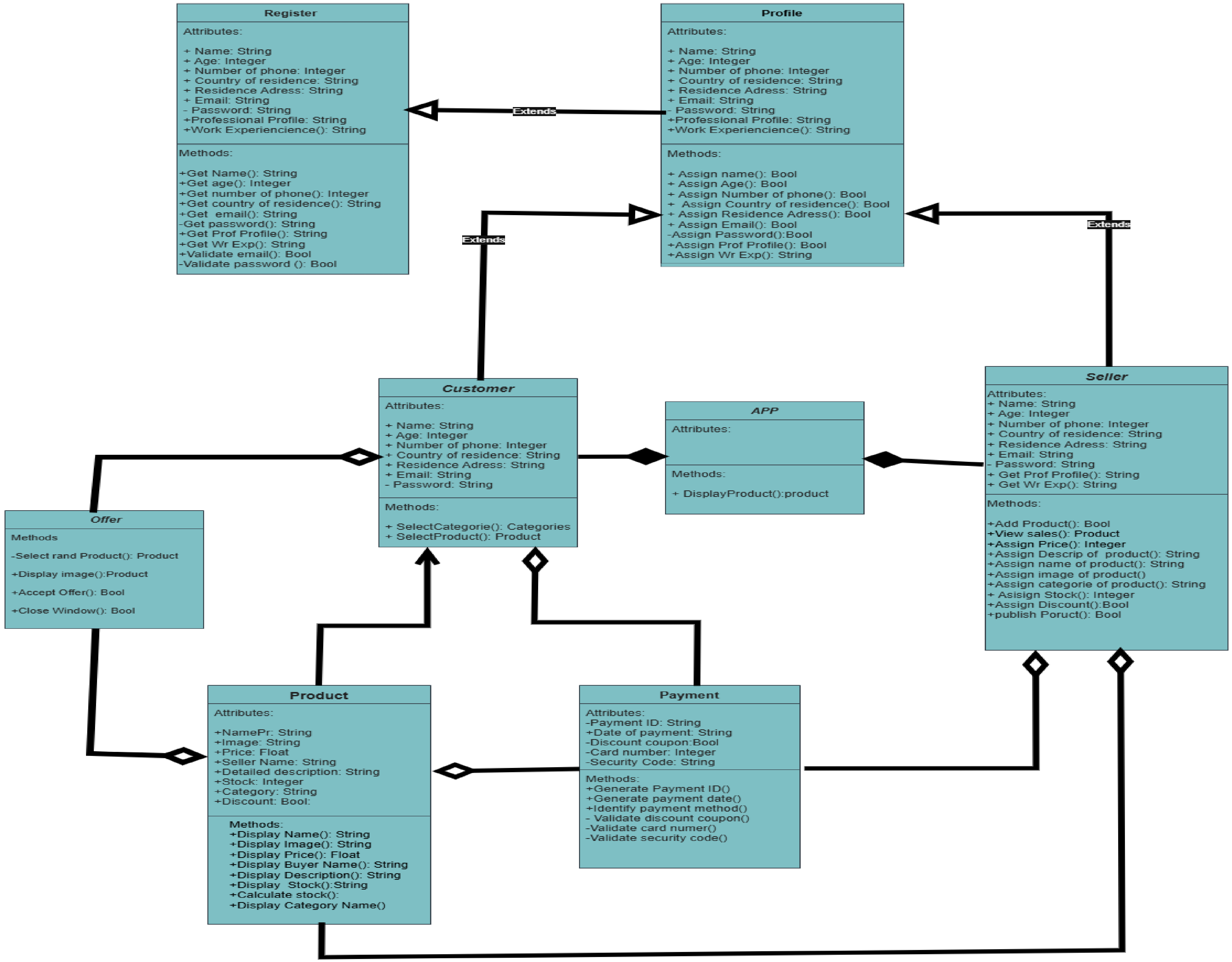
In this section, we will show class, activity, and sequence diagrams.

In class diagrams, we seek to show the relationships between different classes, with the methods and attributes expected for each class, in addition to ensuring and displaying relationships such as association, composition, aggregation, inheritance, or implementation.

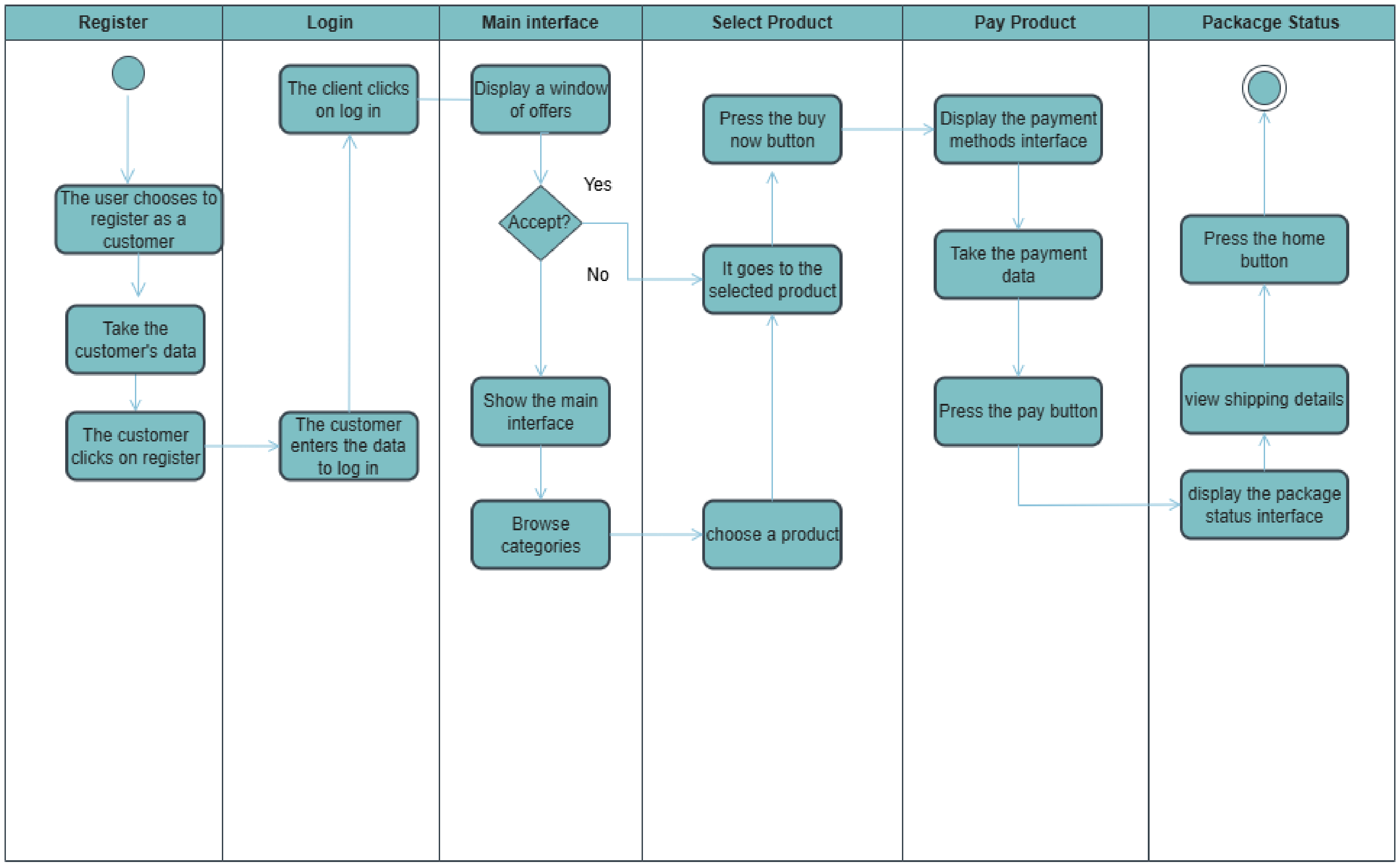
For activity diagrams, we seek to show the flow of information or the sequence of steps to carry out a certain activity in the application, such as actions and processes.

6.1 Changes

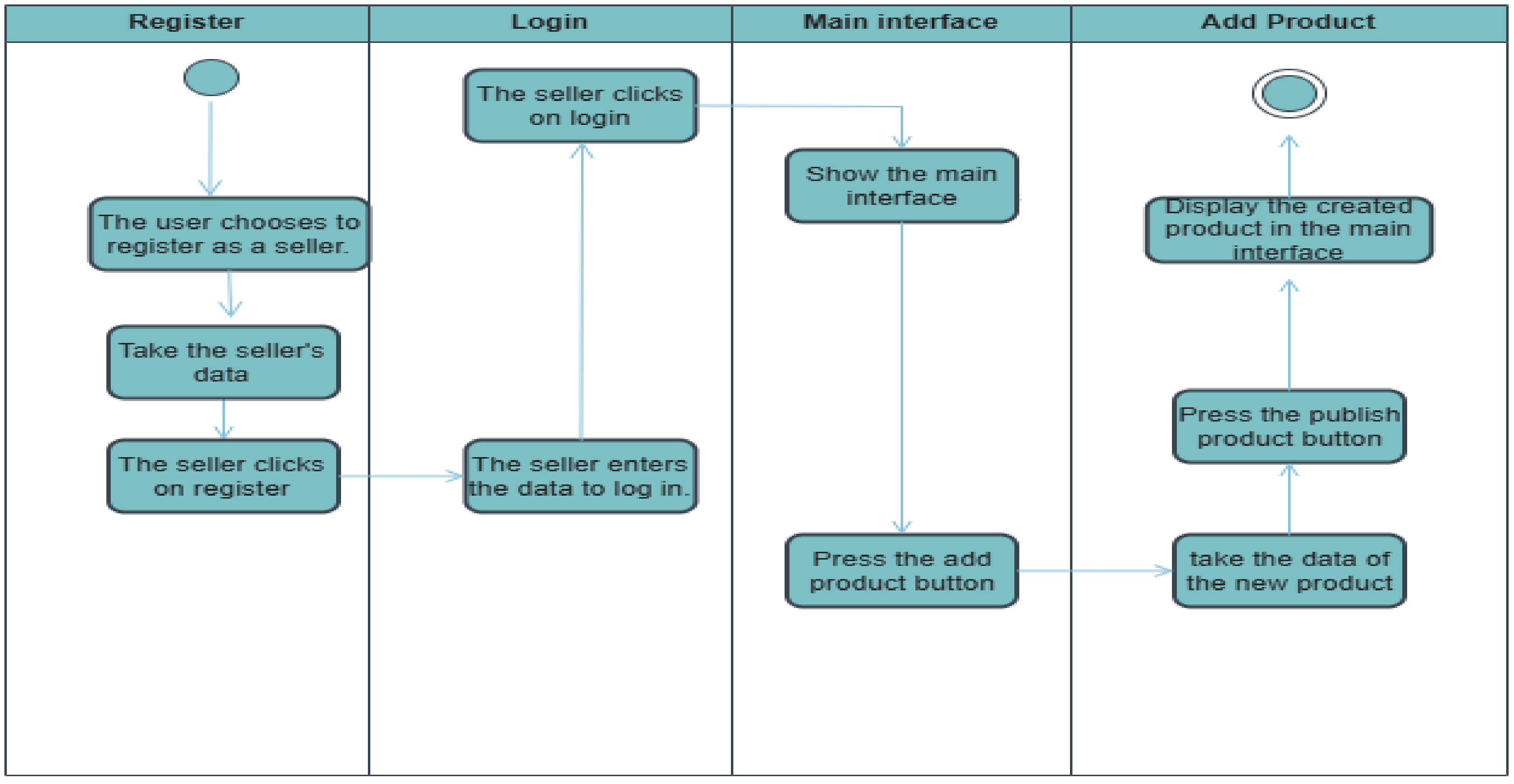
For the UML diagram, we place the arrows that indicate the type of relationship between the classes. We add a new class to try to guarantee the principle of single responsibility and that the classes also allow adding new functionalities if required, complying with the principle of open to extension, closed to modification.


Register
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+Professional Profile: String
+Work Experiencience(): String
Methods:
+Get Name(): String
+Get age(): Integer
+Get number of phone(): Integer
+Get country of residence(): String
+Get email(): String
-Get password(): String
+Get Prof Profile(): String
+Get Wr Exp(): String
+Validate email(): Bool
-Validate password (): Bool
Extends
Profile
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+Professional Profile: String
+Work Experiencience(): String
Methods:
+ Assign name(): Bool
+ Assign Age(): Bool
+ Assign Number of phone(): Bool
+ Assign Country of residence(): Bool
+ Assign Residence Adress(): Bool
+ Assign Email(): Bool
-Assign Password():Bool
+Assign Prof Profile(): Bool
+Assign Wr Exp(): String
Extends
Extends
Customer
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
Methods:
+ SelectCategorie(): Categories
+ SelectProduct(): Product
APP
Attributes:
Methods:
+ DisplayProduct():product
Seller
Attributes:
+ Name: String
+ Age: Integer
+ Number of phone: Integer
+ Country of residence: String
+ Residence Adress: String
+ Email: String
- Password: String
+ Get Prof Profile(): String
+ Get Wr Exp(): String
Methods:
+Add Product(): Bool
+View sales(): Product
+Assign Price(): Integer
+Assign Descrip of product(): String
+Assign name of product(): String
+Assign image of product()
+Assign categorie of product(): String
+ Asisign Stock(): Integer
+Assign Discount():Bool
+publish Poruct(): Bool
Offer
Methods
-Select rand Product(): Product
+Display image():Product
+Accept Offer(): Bool
+Close Window(): Bool
Product
Attributes:
+NamePr: String
+Image: String
+Price: Float
+Seller Name: String
+Detailed description: String
+Stock: Integer
+Category: String
+Discount: Bool:
Methods:
+Display Name(): String
+Display Image(): String
+Display Price(): Float
+Display Buyer Name(): String
+Display Description(): String
+Display Stock():String
+Calculate stock():
+Display Category Name()
Payment
Attributes:
-Payment ID: String
+Date of payment: String
-Discount coupon:Bool
-Card number: Integer
-Security Code: String
Methods:
+Generate Payment ID()
+Generate payment date()
+Identify payment method()
- Validate discount coupon()
-Validate card numer()
-Validate security code()

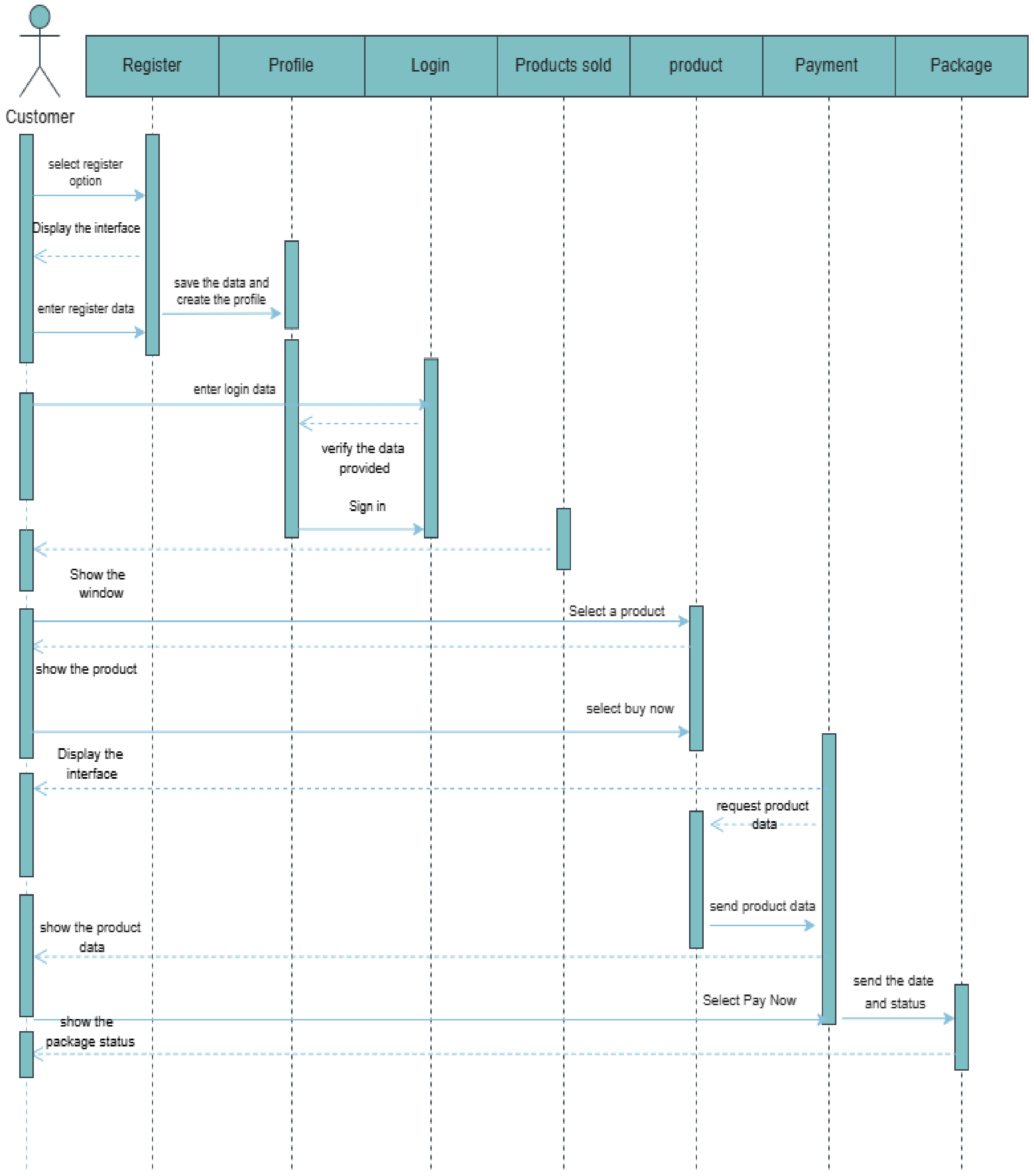
Activity Diagram Product Purchase

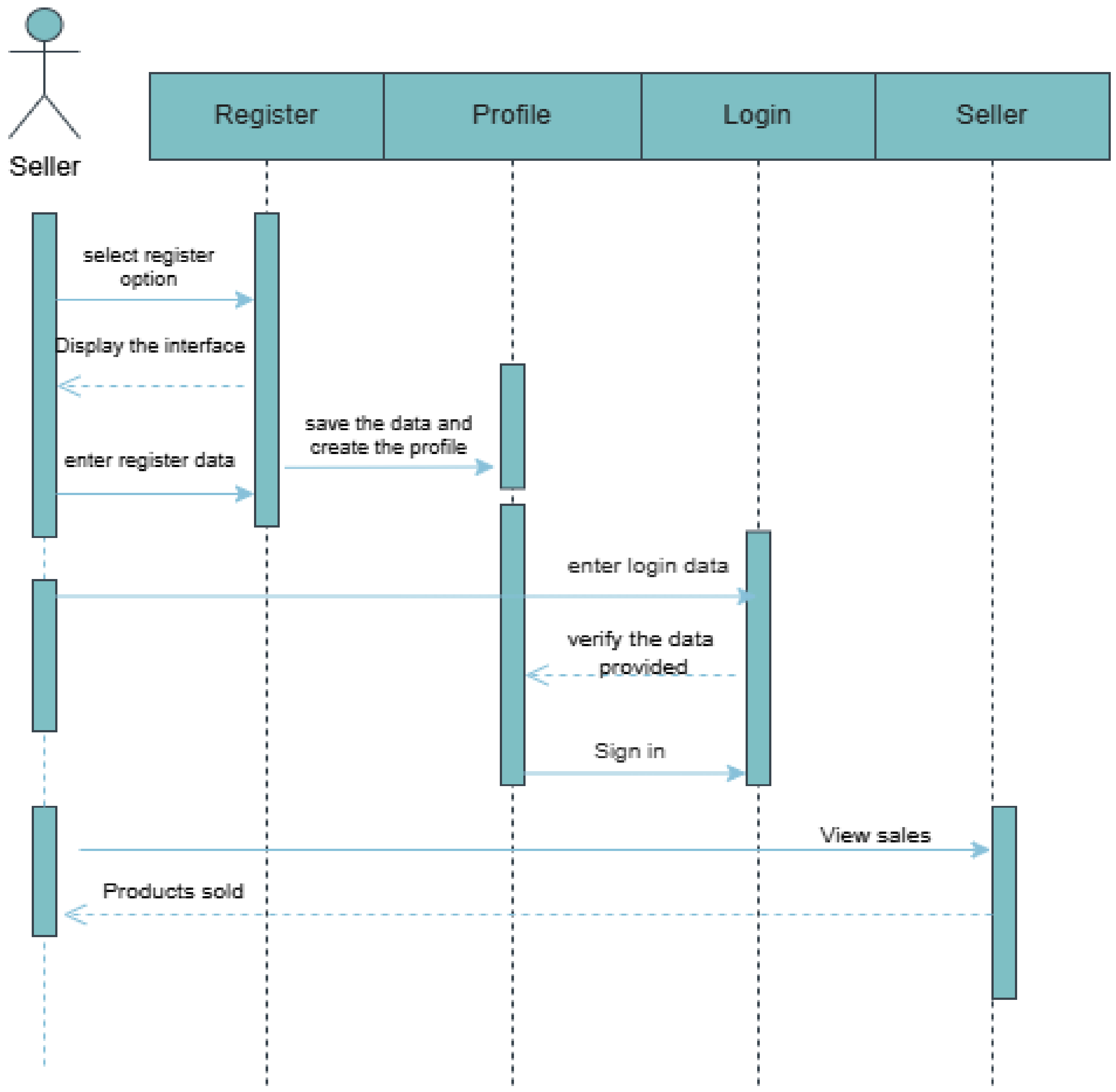

Register
Login
Main interface
Select Product
Pay Product
Packacge Status
The user chooses to register as a customer
Take the customer's data
The customer clicks on register
The customer enters the data to log in
The client clicks on log in
Display a window of offers
Accept?
Yes
No
Show the main interface
Browse categories
choose a product
It goes to the selected product
Press the buy now button
Display the payment methods interface
Take the payment data
Press the pay button
display the package status interface
view shipping details
Press the home button

Activity Diagram Add Product


Register
Login
Main interface
Add Product
The user chooses to register as a seller.
Take the seller's data
The seller clicks on register
The seller enters the data to log in.
The seller clicks on login
Show the main interface
Press the add product button
take the data of the new product
Press the publish product button
Display the created product in the main interface

And in sequence diagrams, we seek to show the relationships between classes sequentially, where importance is given to the order in which information flows.





7. How apply the OPP concepts

In this section we explain how we can apply object-oriented concepts in the project

Inheritance

Using the UML diagrams, we were able to demonstrate that we can inherit some attributes from the Create Profile and Login classes, such as name, age, and address, to the Customer and Seller classes.

Also, from the Seller class, we can, for example, inherit the Add Product, Assign Price, Description, and Product Name methods to this class.

Encapsulation

For encapsulation, we have examples of attributes that we do not want to be public, but rather private to ensure the security of user data, such as the password, the card number, the security code, and everything related to payment or bank details.

For example, you could request the password with a getter and try not to expose user information and only perform validation to be able to execute the setters or perform a specific function. Similarly, this can be applied when performing, for example, card discounting or number validation.

Polymorphism

Based on the diagram, we could not find any class to which we could apply either overloading or inheritance polymorphism.

8. Solid Implementation

In this section we explain how we apply each solid principle to our design

1 Single Responsibility:

This means that classes should have a single responsibility. This can be seen in our design, for example, in the product class, which is only responsible for displaying product information, and in the package class, which is only responsible for package data.

2 Open/Closed Principle:

This principle is used to add new functionality without altering the behavior of the base class. This principle is applied in our design, as we use classes such as Payment, Product, Seller, and Customer, which allow us to extend behavior without modifying specific classes. For example, if we want to add product reviews, we could add this functionality to the product class.

3 Liskov Substitution Principle:

This principle seeks to ensure that a subclass or child class can replace the parent class without affecting its functionality. In our design, without clearly defined child classes, we cannot apply this principle.

4 Interface Segregation Principle:

This principle seeks to ensure that each application's functionality or client has its own interface. This implies having multiple interfaces for a specific purpose or client, rather than having interfaces with multiple overloaded functions. In our design, since we don't have interfaces or more than one class that can implement the same interface, we can't apply this principle.

5 Dependency Inversion Principle:

This principle states that high-level classes should not depend on low-level classes, but rather on abstractions.

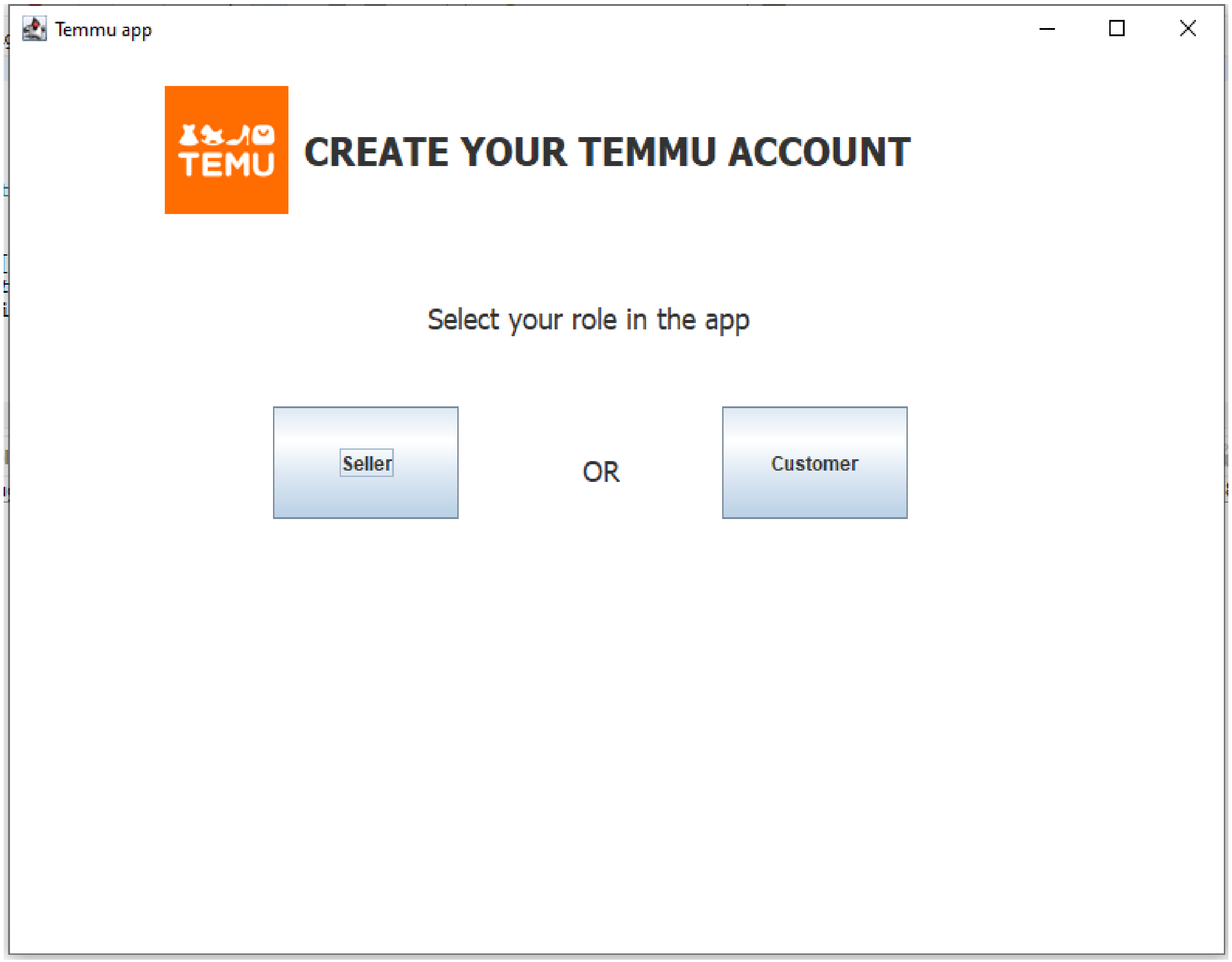
In our design, we don't have high-level and low-level classes, so we can't apply this principle.

9. Implementation GUIS

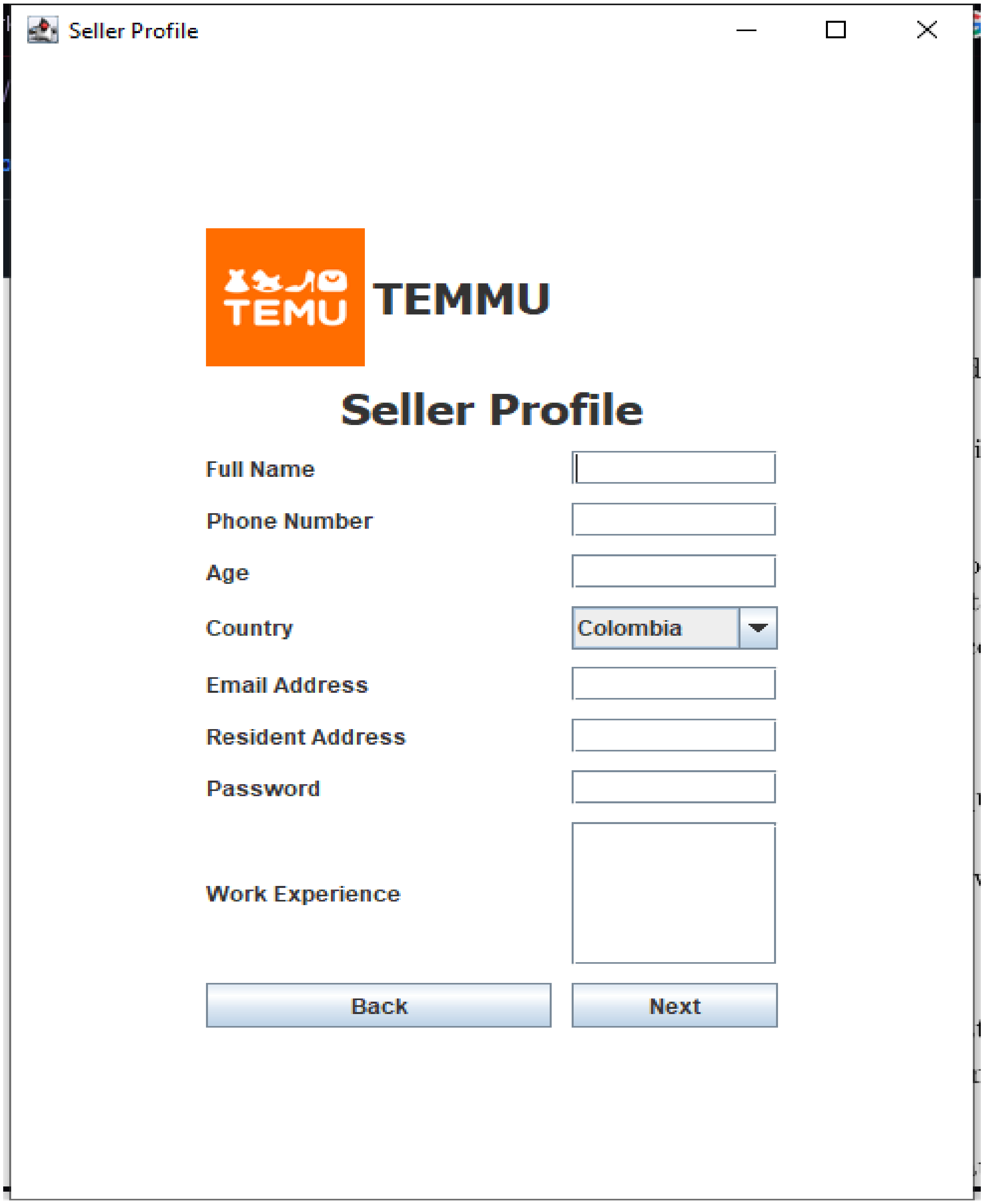
In this section we will show the graphical interfaces implemented using the functionalities of the java.Swing library where we demonstrate the implementation of the mockups at the code level.

Report Workshop 4

9.1 Gui Select role



9.2 Gui Register Seller



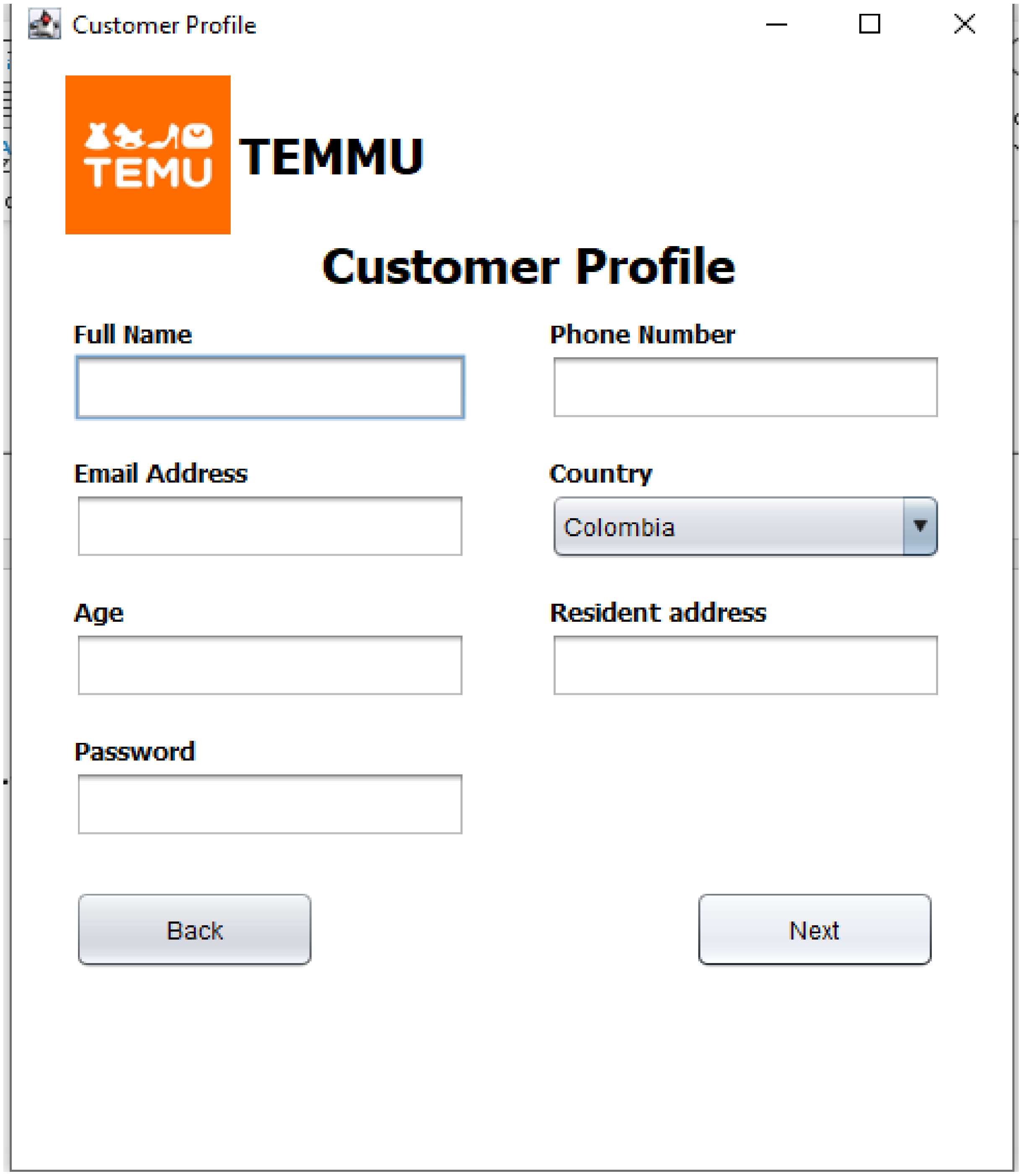
Seller Profile

TEMMU

Seller Profile

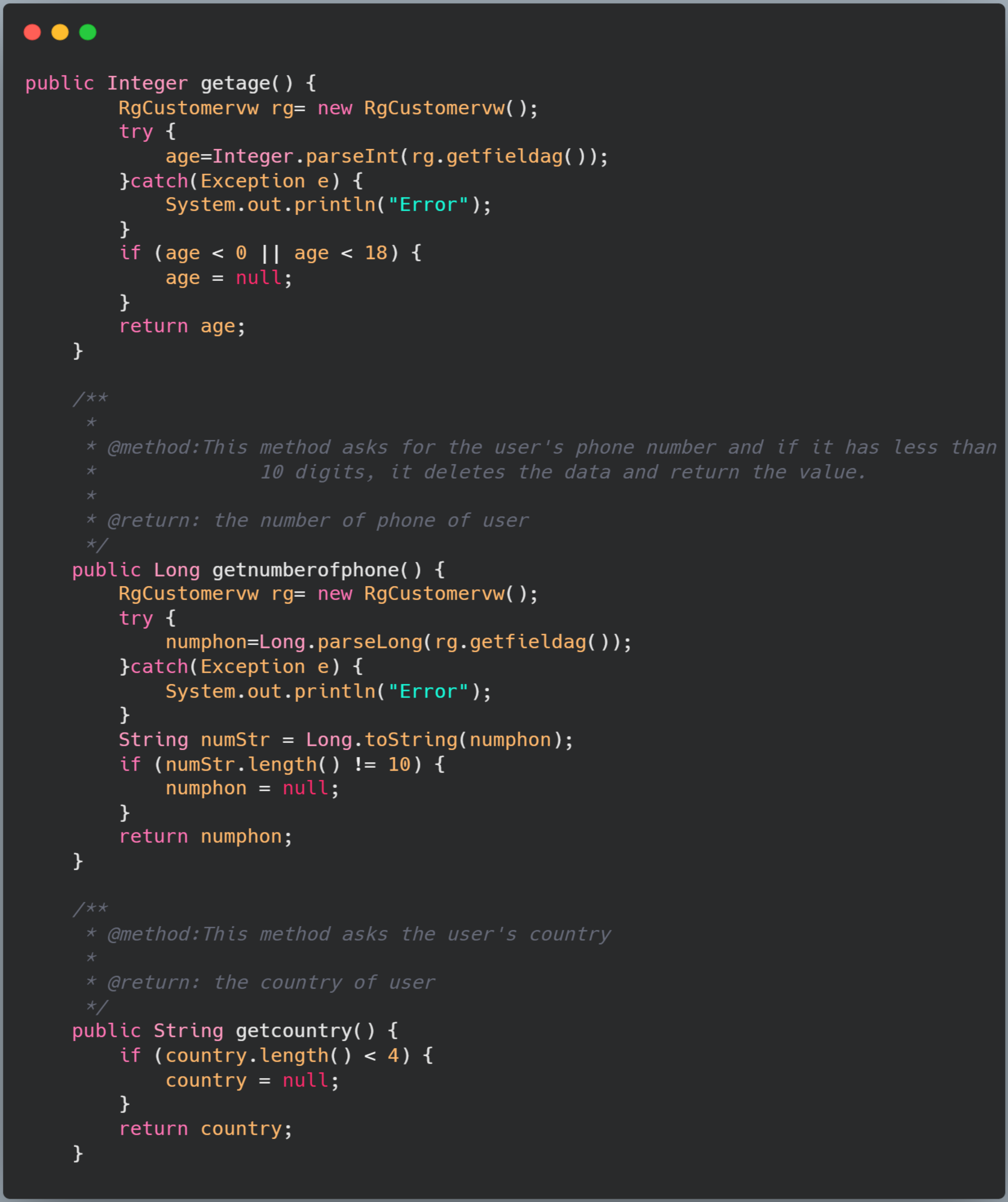
Full Name	
Phone Number	
Age	
Country	Colombia
Email Address	
Resident Address	
Password	
Work Experience	

9.3 Gui Register Customer

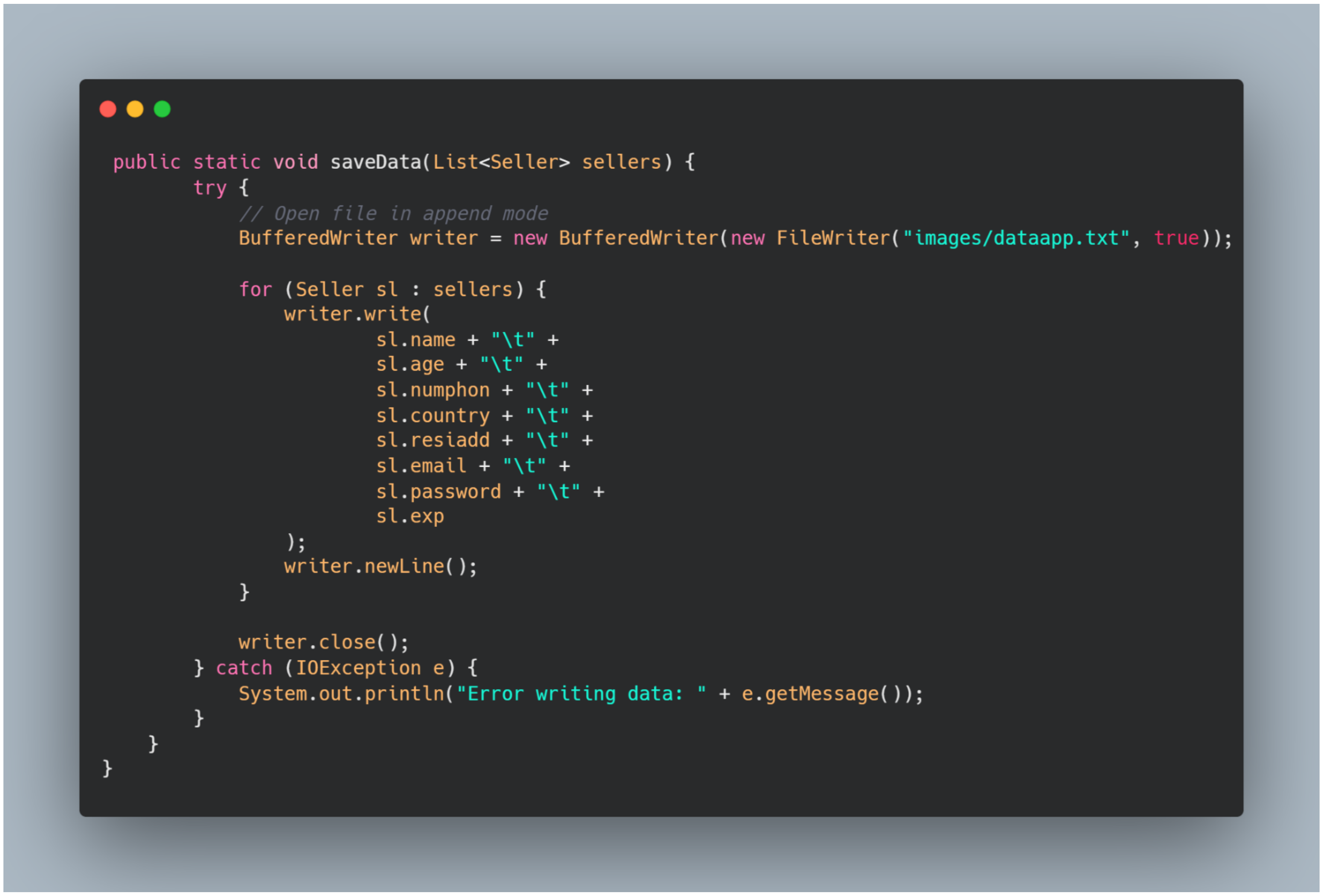


10. File Storage

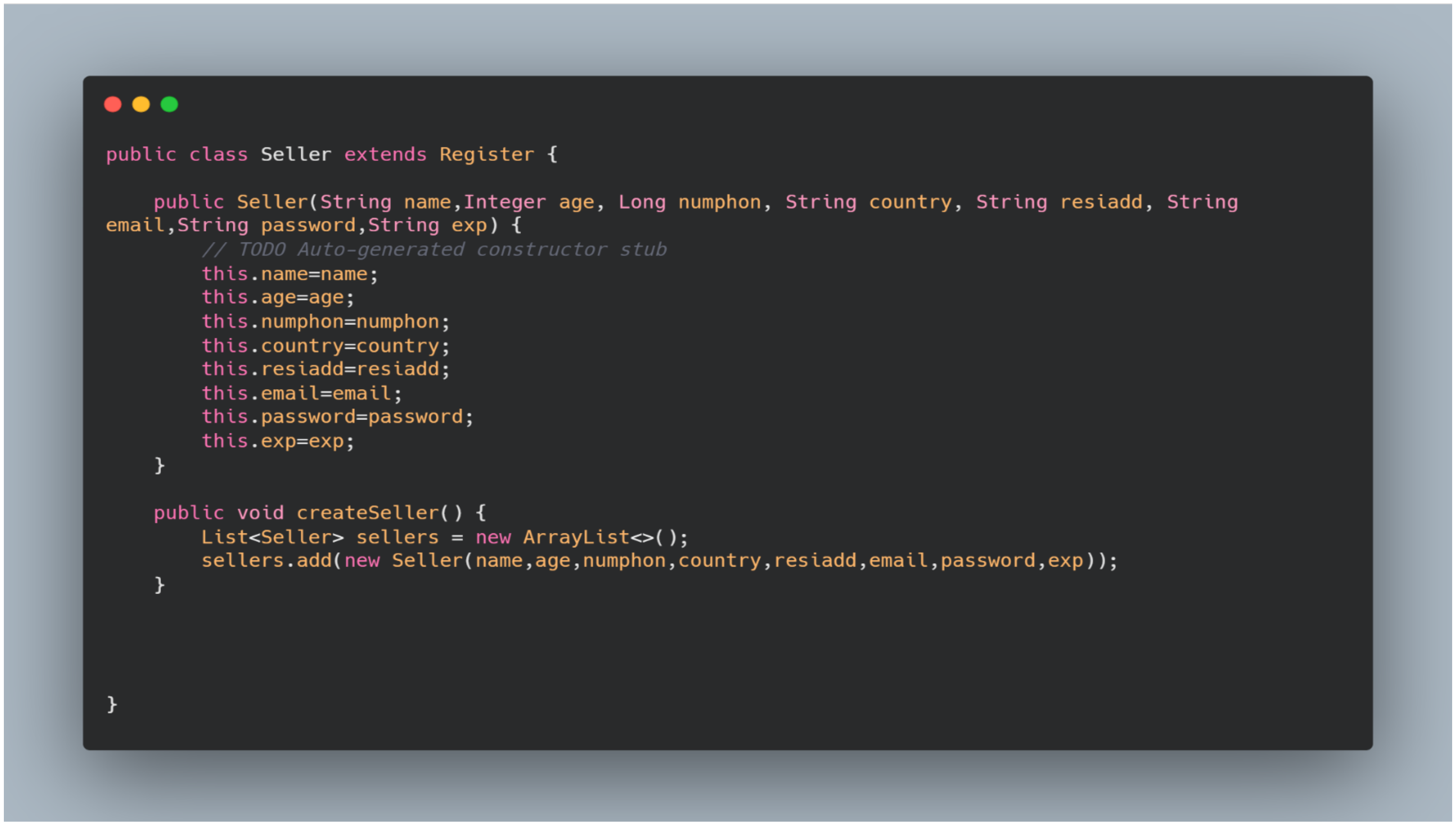
In this section we show some methods that will help us store user data in a txt file.



```java
public Integer getage() {
        RgCustomervw rg= new RgCustomervw();
        try {
            age=Integer.parseInt(rg.getfieldag());
        }catch(Exception e) {
            System.out.println("Error");
        }
        if (age < 0 || age < 18) {
            age = null;
        }
        return age;
    }

    /**
     *
     * @method:This method asks for the user's phone number and if it has less than
     *              10 digits, it deletes the data and return the value.
     *
     * @return: the number of phone of user
     */
    public Long getnumberofphone() {
        RgCustomervw rg= new RgCustomervw();
        try {
            numphon=Long.parseLong(rg.getfieldag());
        }catch(Exception e) {
            System.out.println("Error");
        }
        String numStr = Long.toString(numphon);
        if (numStr.length() != 10) {
            numphon = null;
        }
        return numphon;
    }

    /**
     * @method:This method asks the user's country
     *
     * @return: the country of user
     */
    public String getcountry() {
        if (country.length() < 4) {
            country = null;
        }
        return country;
    }
```



```java
public static void saveData(List<Seller> sellers) {
       try {
           // Open file in append mode
           BufferedWriter writer = new BufferedWriter(new FileWriter("images/dataapp.txt", true));

           for (Seller sl : sellers) {
               writer.write(
                       sl.name + "\t" +
                       sl.age + "\t" +
                       sl.numphon + "\t" +
                       sl.country + "\t" +
                       sl.resiadd + "\t" +
                       sl.email + "\t" +
                       sl.password + "\t" +
                       sl.exp
               );
               writer.newLine();
           }

           writer.close();
       } catch (IOException e) {
           System.out.println("Error writing data: " + e.getMessage());
       }
    }
}
```

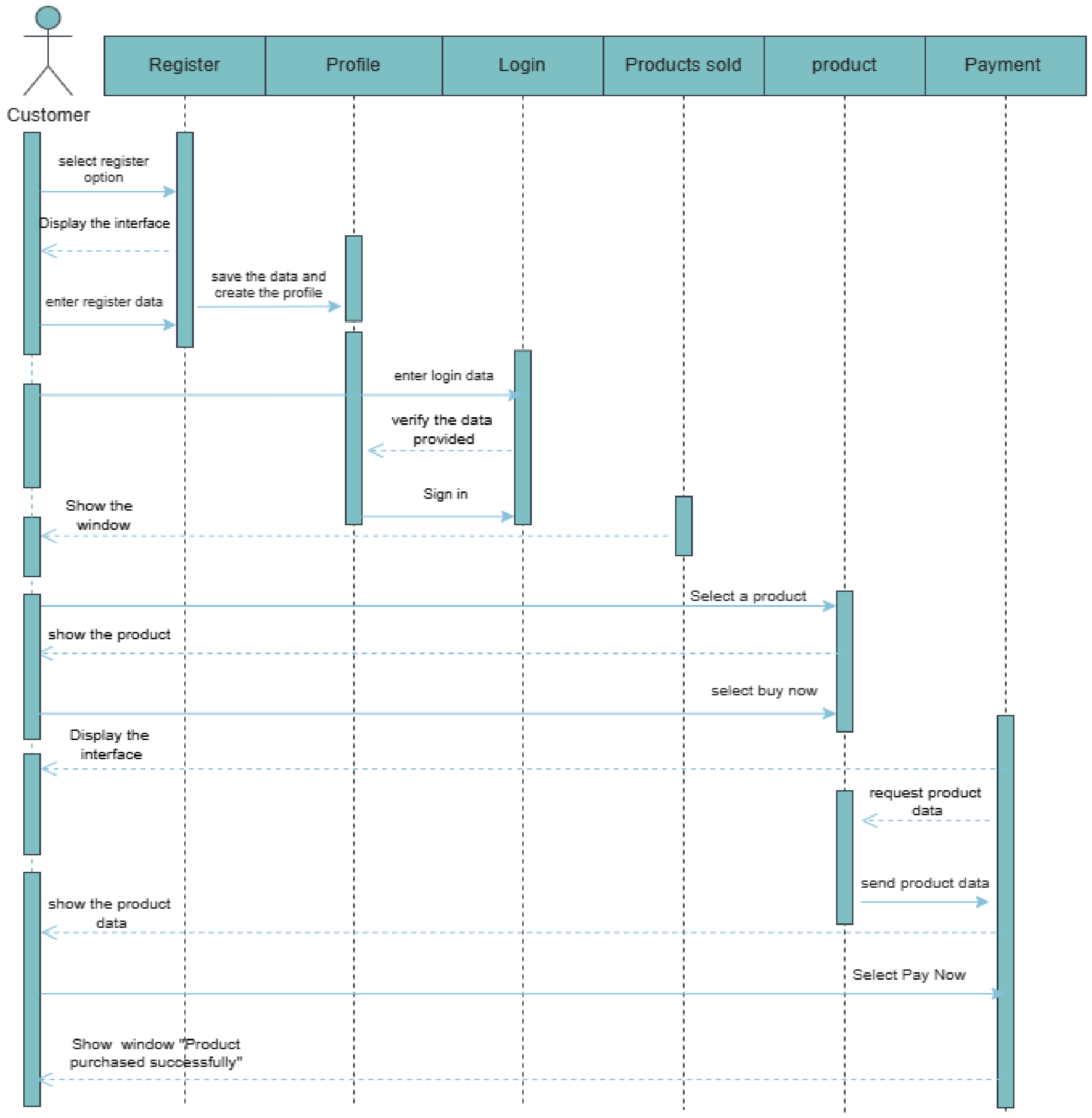


```java
public class Seller extends Register {

    public Seller(String name,Integer age, Long numphon, String country, String resiadd, String
email,String password,String exp) {
        // TODO Auto-generated constructor stub
        this.name=name;
        this.age=age;
        this.numphon=numphon;
        this.country=country;
        this.resiadd=resiadd;
        this.email=email;
        this.password=password;
        this.exp=exp;
    }

    public void createSeller() {
        List<Seller> sellers = new ArrayList<>();
        sellers.add(new Seller(name,age,numphon,country,resiadd,email,password,exp));
    }



}
```

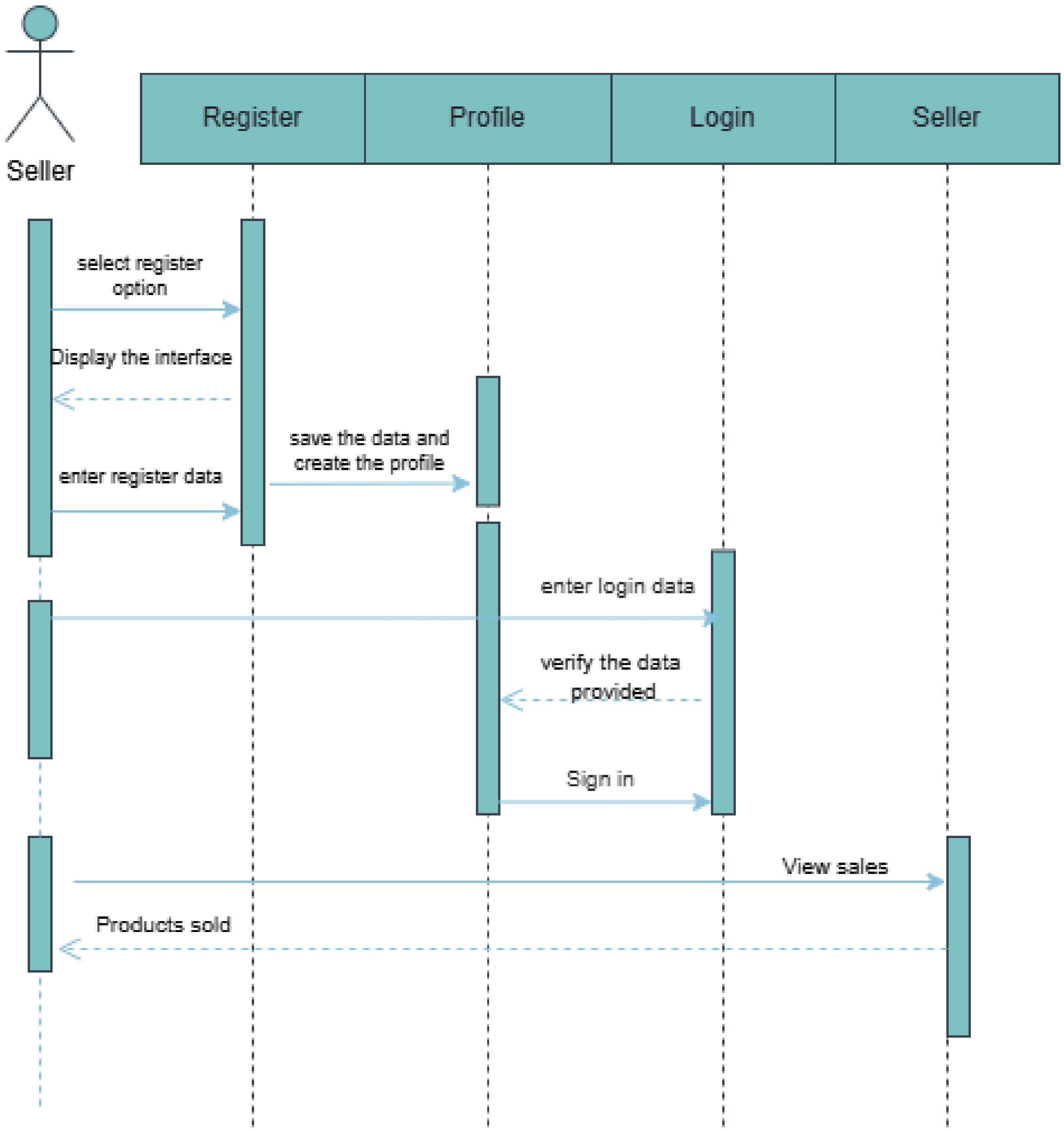
11. Diagrams of sequence (Updated) and code of GUIS

In this section we show sequence diagrams showing the data flow proposed for the final solution.

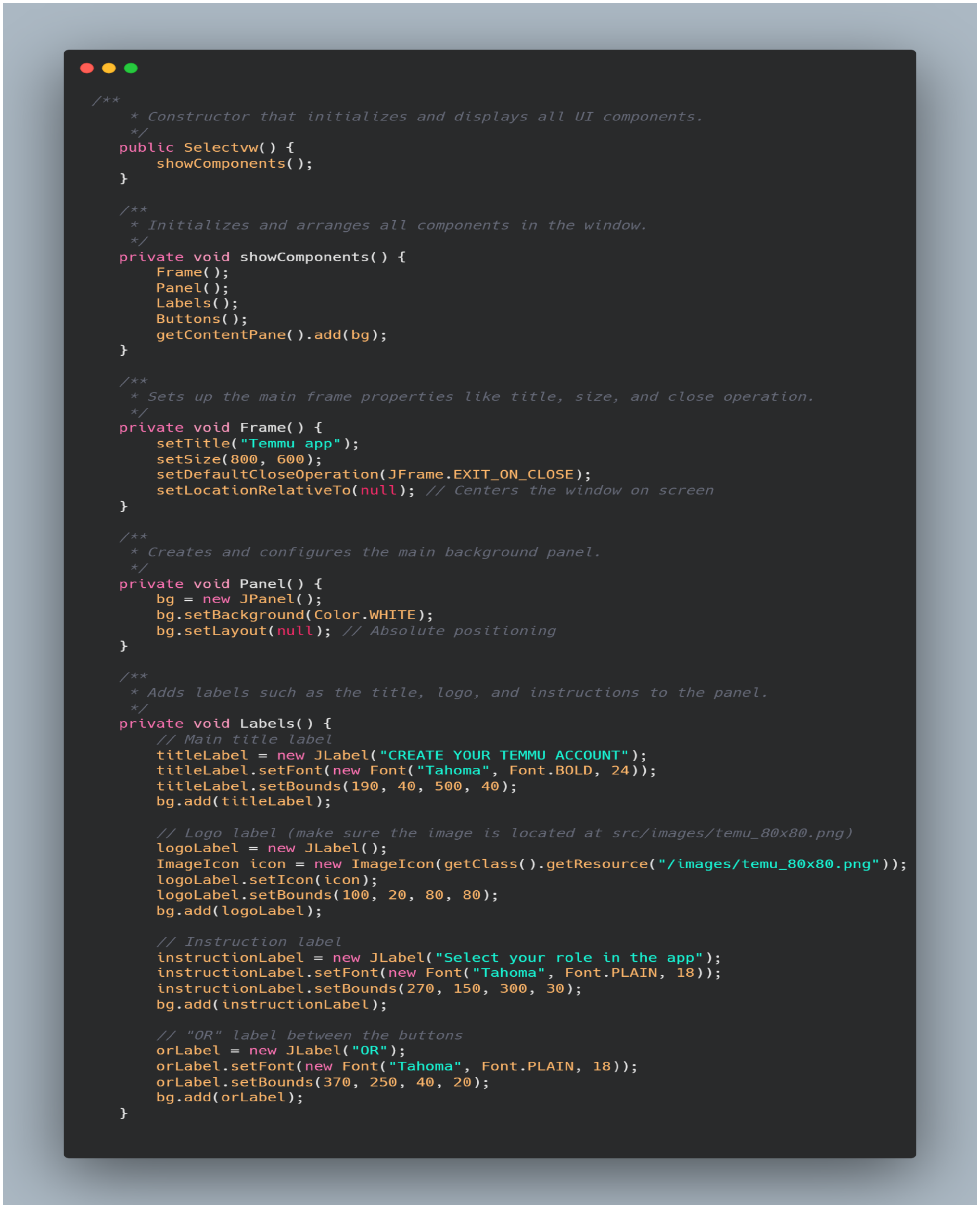
11.1 Sequence diagram of the customer:



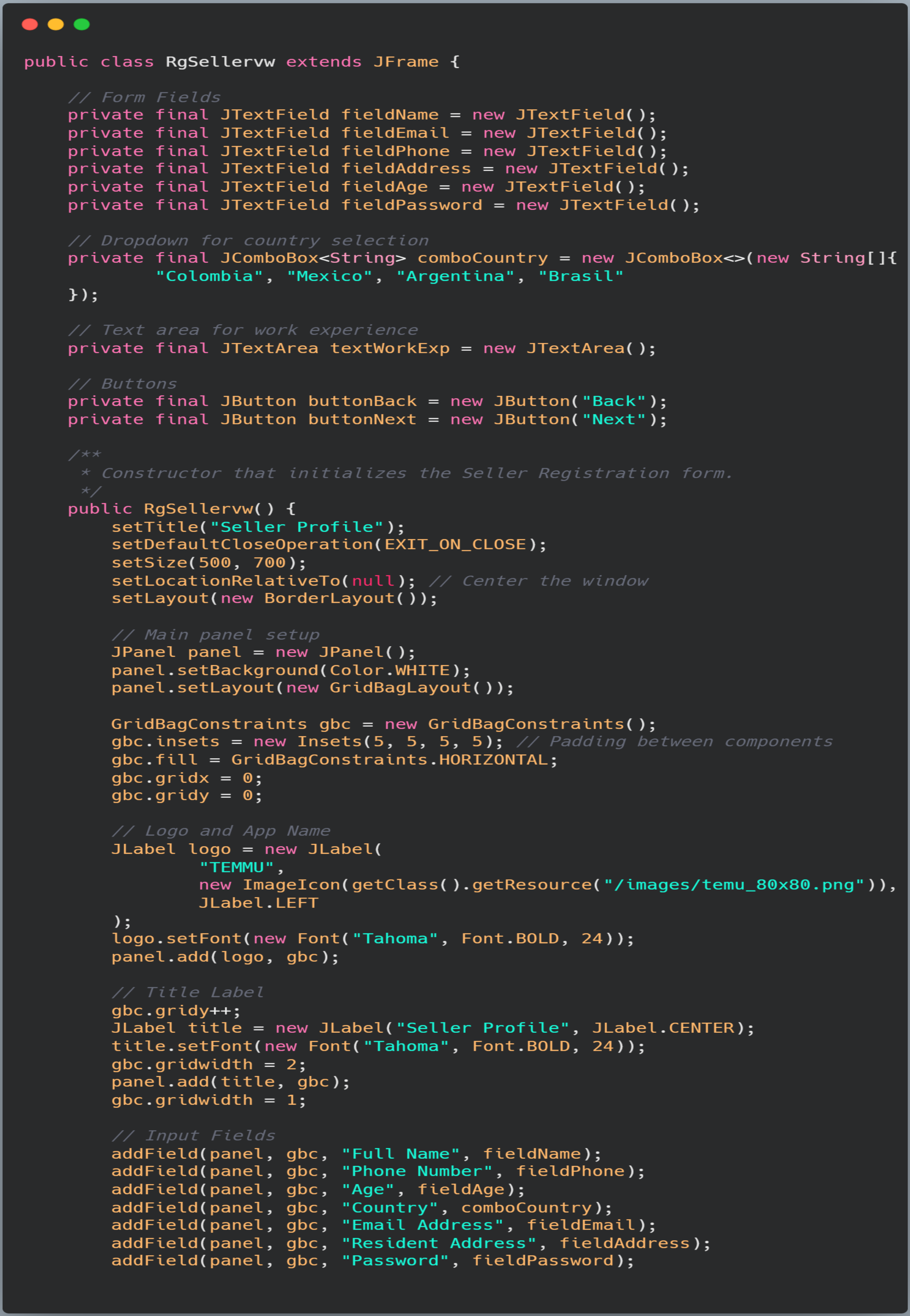
Sequence diagram of the seller:



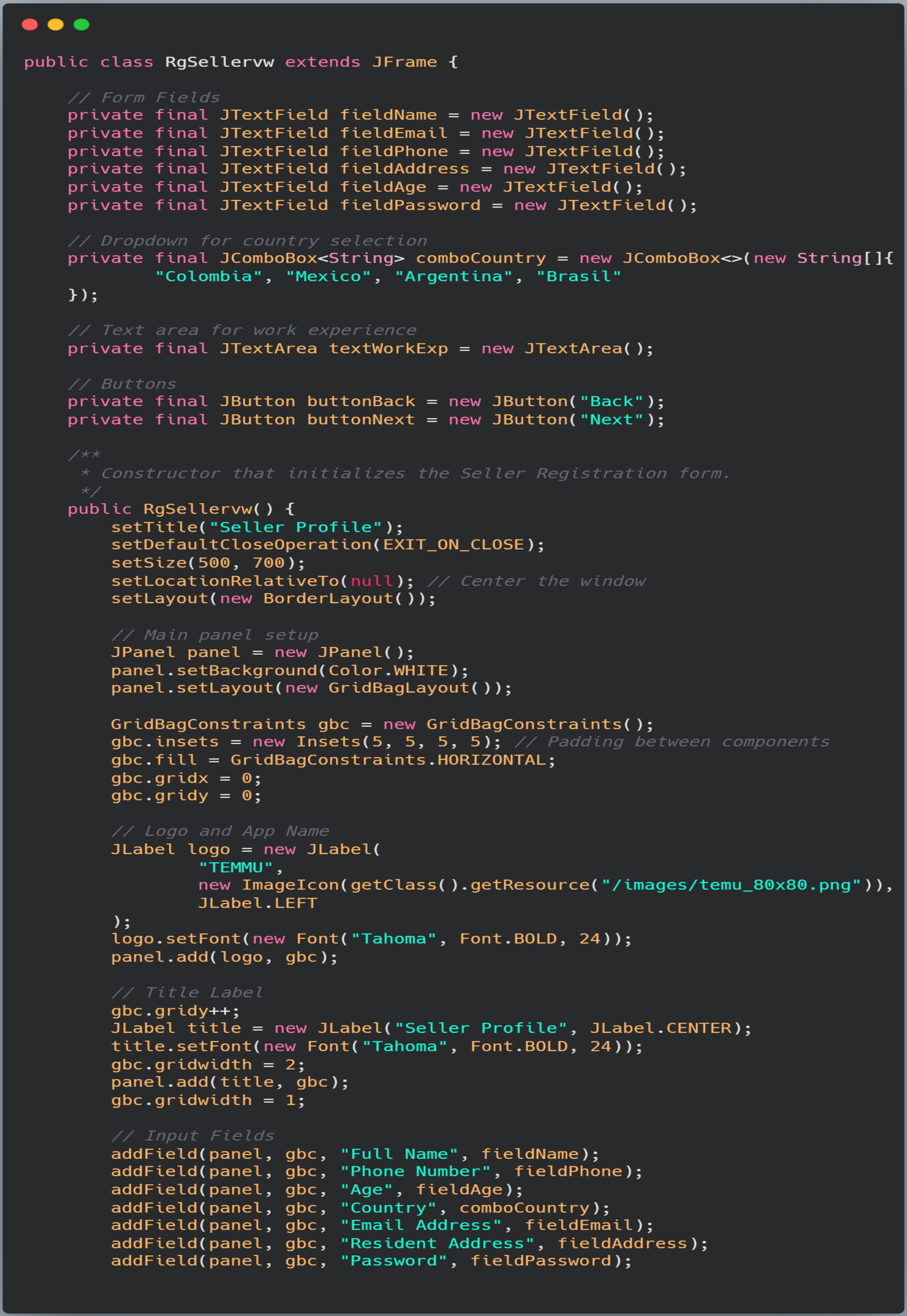
11.2 GUI code snippets



```
/**
     * Constructor that initializes and displays all UI components.
     */
    public Selectvw() {
        showComponents();
    }

    /**
     * Initializes and arranges all components in the window.
     */
    private void showComponents() {
        Frame();
        Panel();
        Labels();
        Buttons();
        getContentPane().add(bg);
    }

    /**
     * Sets up the main frame properties like title, size, and close operation.
     */
    private void Frame() {
        setTitle("Temmu app");
        setSize(800, 600);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Centers the window on screen
    }

    /**
     * Creates and configures the main background panel.
     */
    private void Panel() {
        bg = new JPanel();
        bg.setBackground(Color.WHITE);
        bg.setLayout(null); // Absolute positioning
    }

    /**
     * Adds labels such as the title, logo, and instructions to the panel.
     */
    private void Labels() {
        // Main title label
        titleLabel = new JLabel("CREATE YOUR TEMMU ACCOUNT");
        titleLabel.setFont(new Font("Tahoma", Font.BOLD, 24));
        titleLabel.setBounds(190, 40, 500, 40);
        bg.add(titleLabel);

        // Logo label (make sure the image is located at src/images/temu_80x80.png)
        logoLabel = new JLabel();
        ImageIcon icon = new ImageIcon(getClass().getResource("/images/temu_80x80.png"));
        logoLabel.setIcon(icon);
        logoLabel.setBounds(100, 20, 80, 80);
        bg.add(logoLabel);

        // Instruction label
        instructionLabel = new JLabel("Select your role in the app");
        instructionLabel.setFont(new Font("Tahoma", Font.PLAIN, 18));
        instructionLabel.setBounds(270, 150, 300, 30);
        bg.add(instructionLabel);

        // "OR" label between the buttons
        orLabel = new JLabel("OR");
        orLabel.setFont(new Font("Tahoma", Font.PLAIN, 18));
        orLabel.setBounds(370, 250, 40, 20);
        bg.add(orLabel);
    }
```



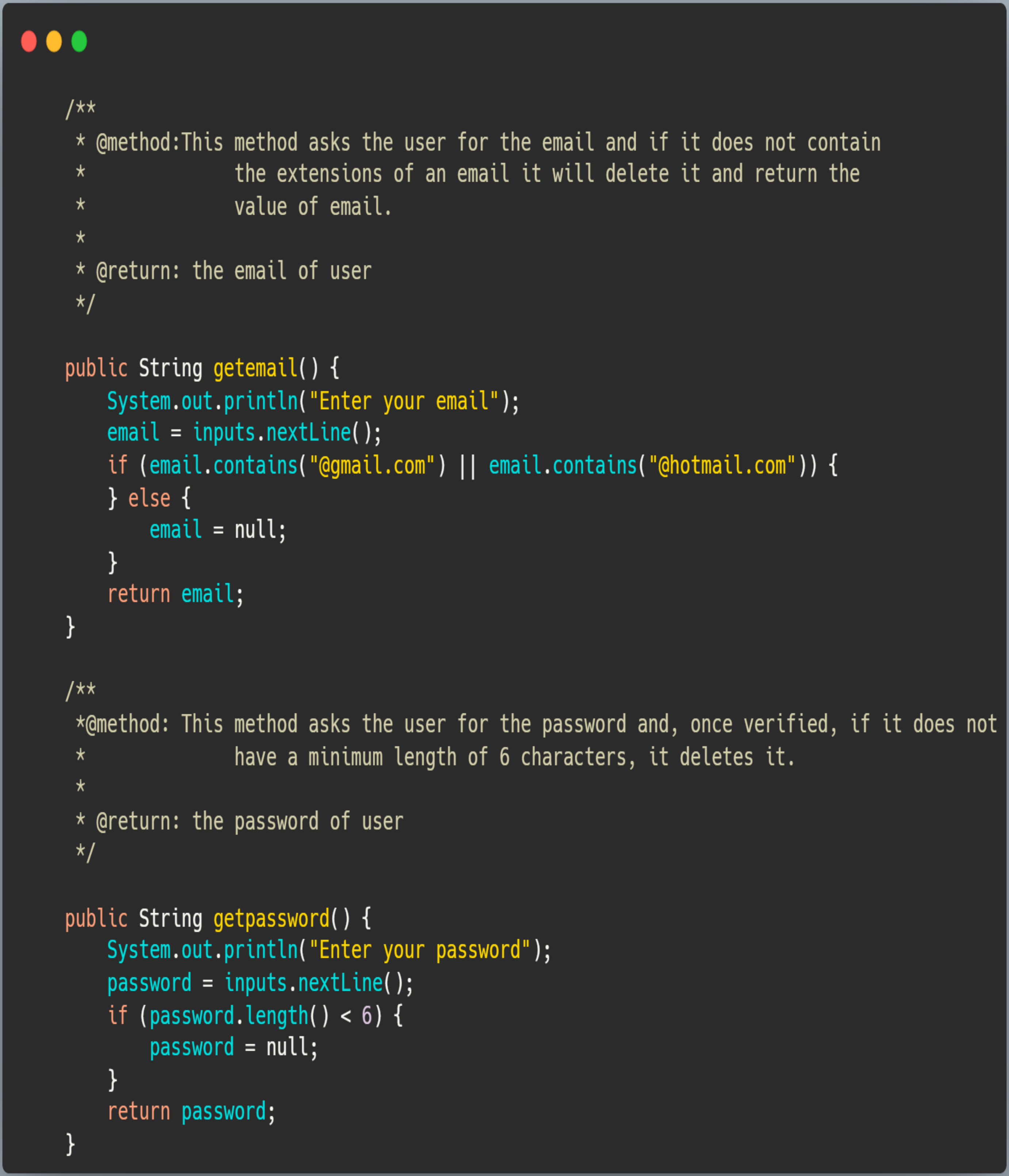
```java
public class RgSellervw extends JFrame {

    // Form Fields
    private final JTextField fieldName = new JTextField();
    private final JTextField fieldEmail = new JTextField();
    private final JTextField fieldPhone = new JTextField();
    private final JTextField fieldAddress = new JTextField();
    private final JTextField fieldAge = new JTextField();
    private final JTextField fieldPassword = new JTextField();

    // Dropdown for country selection
    private final JComboBox<String> comboCountry = new JComboBox<>(new String[]{
            "Colombia", "Mexico", "Argentina", "Brasil"
    });

    // Text area for work experience
    private final JTextArea textWorkExp = new JTextArea();

    // Buttons
    private final JButton buttonBack = new JButton("Back");
    private final JButton buttonNext = new JButton("Next");

    /**
     * Constructor that initializes the Seller Registration form.
     */
    public RgSellervw() {
        setTitle("Seller Profile");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(500, 700);
        setLocationRelativeTo(null); // Center the window
        setLayout(new BorderLayout());

        // Main panel setup
        JPanel panel = new JPanel();
        panel.setBackground(Color.WHITE);
        panel.setLayout(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5); // Padding between components
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
        gbc.gridy = 0;

        // Logo and App Name
        JLabel logo = new JLabel(
                "TEMMU",
                new ImageIcon(getClass().getResource("/images/temu_80x80.png")),
                JLabel.LEFT
        );
        logo.setFont(new Font("Tahoma", Font.BOLD, 24));
        panel.add(logo, gbc);

        // Title Label
        gbc.gridy++;
        JLabel title = new JLabel("Seller Profile", JLabel.CENTER);
        title.setFont(new Font("Tahoma", Font.BOLD, 24));
        gbc.gridwidth = 2;
        panel.add(title, gbc);
        gbc.gridwidth = 1;

        // Input Fields
        addField(panel, gbc, "Full Name", fieldName);
        addField(panel, gbc, "Phone Number", fieldPhone);
        addField(panel, gbc, "Age", fieldAge);
        addField(panel, gbc, "Country", comboCountry);
        addField(panel, gbc, "Email Address", fieldEmail);
        addField(panel, gbc, "Resident Address", fieldAddress);
        addField(panel, gbc, "Password", fieldPassword);
```



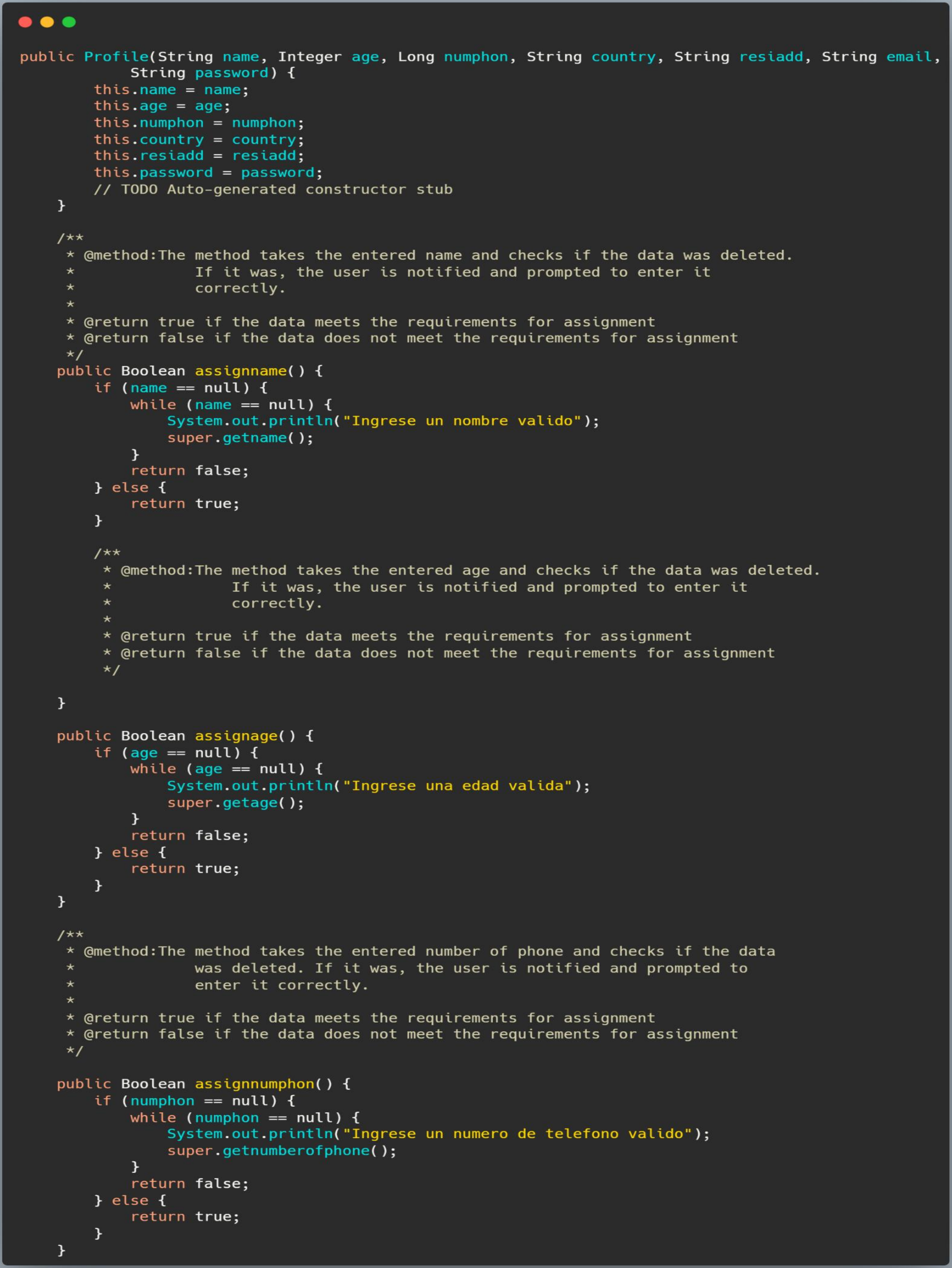
```java
public class RgSellervw extends JFrame {

    // Form Fields
    private final JTextField fieldName = new JTextField();
    private final JTextField fieldEmail = new JTextField();
    private final JTextField fieldPhone = new JTextField();
    private final JTextField fieldAddress = new JTextField();
    private final JTextField fieldAge = new JTextField();
    private final JTextField fieldPassword = new JTextField();

    // Dropdown for country selection
    private final JComboBox<String> comboCountry = new JComboBox<>(new String[]{
            "Colombia", "Mexico", "Argentina", "Brasil"
    });

    // Text area for work experience
    private final JTextArea textWorkExp = new JTextArea();

    // Buttons
    private final JButton buttonBack = new JButton("Back");
    private final JButton buttonNext = new JButton("Next");

    /**
     * Constructor that initializes the Seller Registration form.
     */
    public RgSellervw() {
        setTitle("Seller Profile");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(500, 700);
        setLocationRelativeTo(null); // Center the window
        setLayout(new BorderLayout());

        // Main panel setup
        JPanel panel = new JPanel();
        panel.setBackground(Color.WHITE);
        panel.setLayout(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5); // Padding between components
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridx = 0;
        gbc.gridy = 0;

        // Logo and App Name
        JLabel logo = new JLabel(
                "TEMMU",
                new ImageIcon(getClass().getResource("/images/temu_80x80.png")),
                JLabel.LEFT
        );
        logo.setFont(new Font("Tahoma", Font.BOLD, 24));
        panel.add(logo, gbc);

        // Title Label
        gbc.gridy++;
        JLabel title = new JLabel("Seller Profile", JLabel.CENTER);
        title.setFont(new Font("Tahoma", Font.BOLD, 24));
        gbc.gridwidth = 2;
        panel.add(title, gbc);
        gbc.gridwidth = 1;

        // Input Fields
        addField(panel, gbc, "Full Name", fieldName);
        addField(panel, gbc, "Phone Number", fieldPhone);
        addField(panel, gbc, "Age", fieldAge);
        addField(panel, gbc, "Country", comboCountry);
        addField(panel, gbc, "Email Address", fieldEmail);
        addField(panel, gbc, "Resident Address", fieldAddress);
        addField(panel, gbc, "Password", fieldPassword);
```

12 Code Snippets

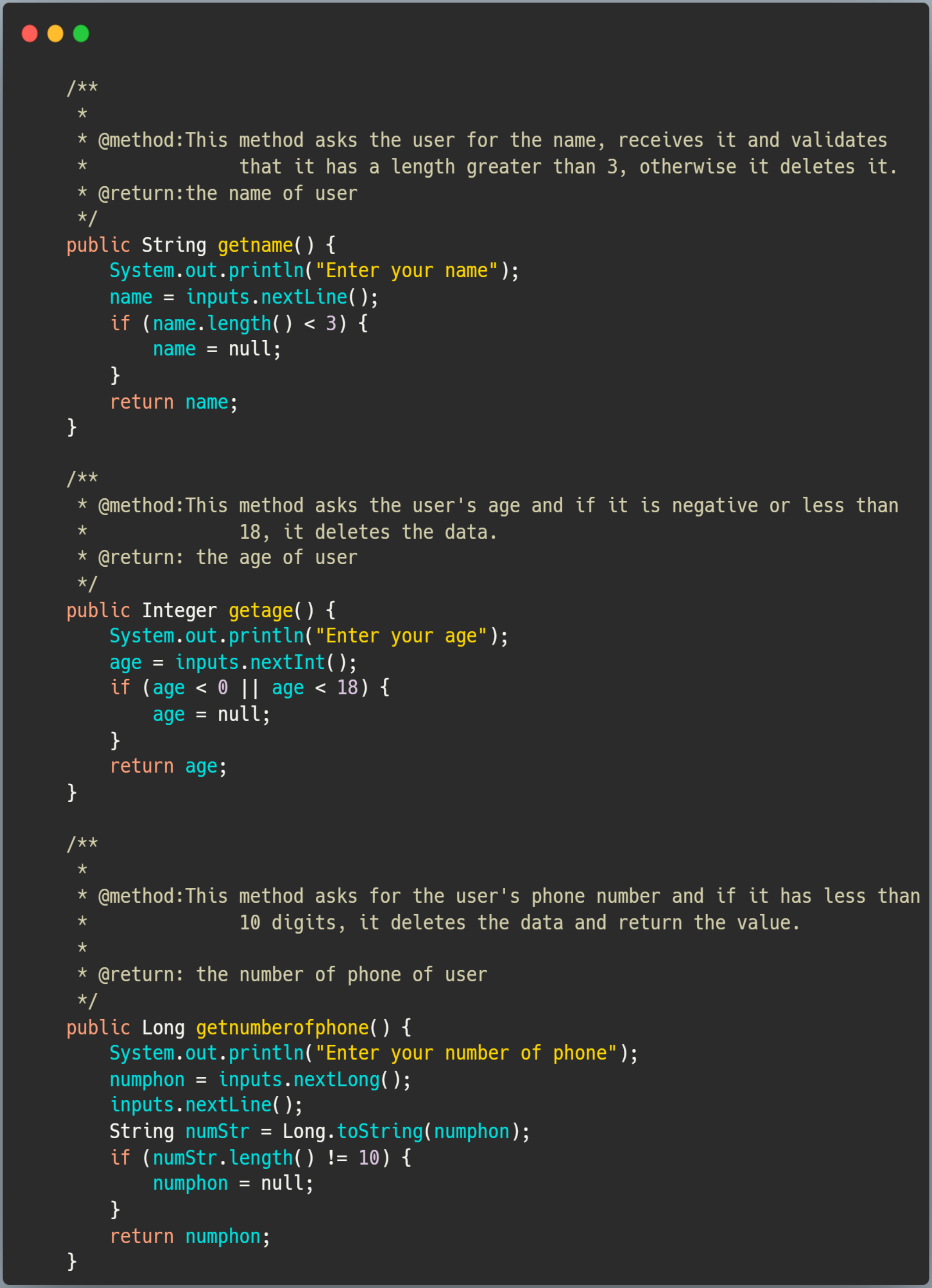
In this section we show code advances related to the logic of some of the relevant classes in the registration and creation of the user profile.



```java
/**
 * @method:This method asks the user for the email and if it does not contain
 *              the extensions of an email it will delete it and return the
 *              value of email.
 *
 * @return: the email of user
 */

public String getemail() {
    System.out.println("Enter your email");
    email = inputs.nextLine();
    if (email.contains("@gmail.com") || email.contains("@hotmail.com")) {
    } else {
        email = null;
    }
    return email;
}

/**
 *@method: This method asks the user for the password and, once verified, if it does not
 *              have a minimum length of 6 characters, it deletes it.
 *
 * @return: the password of user
 */

public String getpassword() {
    System.out.println("Enter your password");
    password = inputs.nextLine();
    if (password.length() < 6) {
        password = null;
    }
    return password;
}
```



```
public Profile(String name, Integer age, Long numphon, String country, String resiadd, String email,
            String password) {
        this.name = name;
        this.age = age;
        this.numphon = numphon;
        this.country = country;
        this.resiadd = resiadd;
        this.password = password;
        // TODO Auto-generated constructor stub
    }

    /**
     * @method:The method takes the entered name and checks if the data was deleted.
     *             If it was, the user is notified and prompted to enter it
     *             correctly.
     *
     * @return true if the data meets the requirements for assignment
     * @return false if the data does not meet the requirements for assignment
     */
    public Boolean assignname() {
        if (name == null) {
            while (name == null) {
                System.out.println("Ingrese un nombre valido");
                super.getname();
            }
            return false;
        } else {
            return true;
        }

        /**
         * @method:The method takes the entered age and checks if the data was deleted.
         *             If it was, the user is notified and prompted to enter it
         *             correctly.
         *
         * @return true if the data meets the requirements for assignment
         * @return false if the data does not meet the requirements for assignment
         */

    }

    public Boolean assignage() {
        if (age == null) {
            while (age == null) {
                System.out.println("Ingrese una edad valida");
                super.getage();
            }
            return false;
        } else {
            return true;
        }
    }

    /**
     * @method:The method takes the entered number of phone and checks if the data
     *             was deleted. If it was, the user is notified and prompted to
     *             enter it correctly.
     *
     * @return true if the data meets the requirements for assignment
     * @return false if the data does not meet the requirements for assignment
     */

    public Boolean assignnumphon() {
        if (numphon == null) {
            while (numphon == null) {
                System.out.println("Ingrese un numero de telefono valido");
                super.getnumberofphone();
            }
            return false;
        } else {
            return true;
        }
    }
```



```java
/**
 *
 * @method:This method asks the user for the name, receives it and validates
 *              that it has a length greater than 3, otherwise it deletes it.
 * @return:the name of user
 */
public String getname() {
    System.out.println("Enter your name");
    name = inputs.nextLine();
    if (name.length() < 3) {
        name = null;
    }
    return name;
}

/**
 * @method:This method asks the user's age and if it is negative or less than
 *              18, it deletes the data.
 * @return: the age of user
 */
public Integer getage() {
    System.out.println("Enter your age");
    age = inputs.nextInt();
    if (age < 0 || age < 18) {
        age = null;
    }
    return age;
}

/**
 *
 * @method:This method asks for the user's phone number and if it has less than
 *              10 digits, it deletes the data and return the value.
 *
 * @return: the number of phone of user
 */
public Long getnumberofphone() {
    System.out.println("Enter your number of phone");
    numphon = inputs.nextLong();
    inputs.nextLine();
    String numStr = Long.toString(numphon);
    if (numStr.length() != 10) {
        numphon = null;
    }
    return numphon;
}
```