E-sale

Presented by

Aritra Naha Likhan (2007092)

Sumaiya Rahim Suma (2007102)

Hanium Maria Joli (2007113)

Agenda

- Easy trade

- User friendly interface

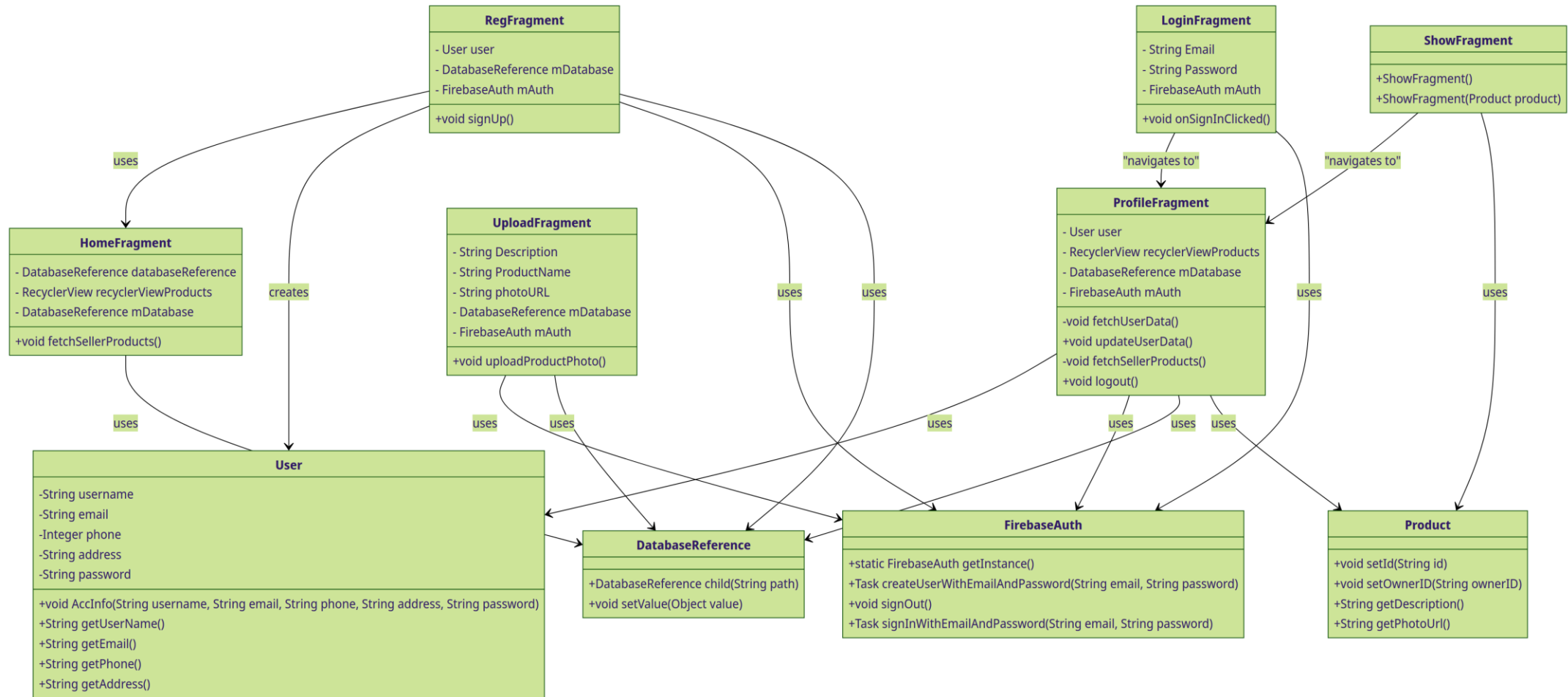- Interactive seller-buyer information

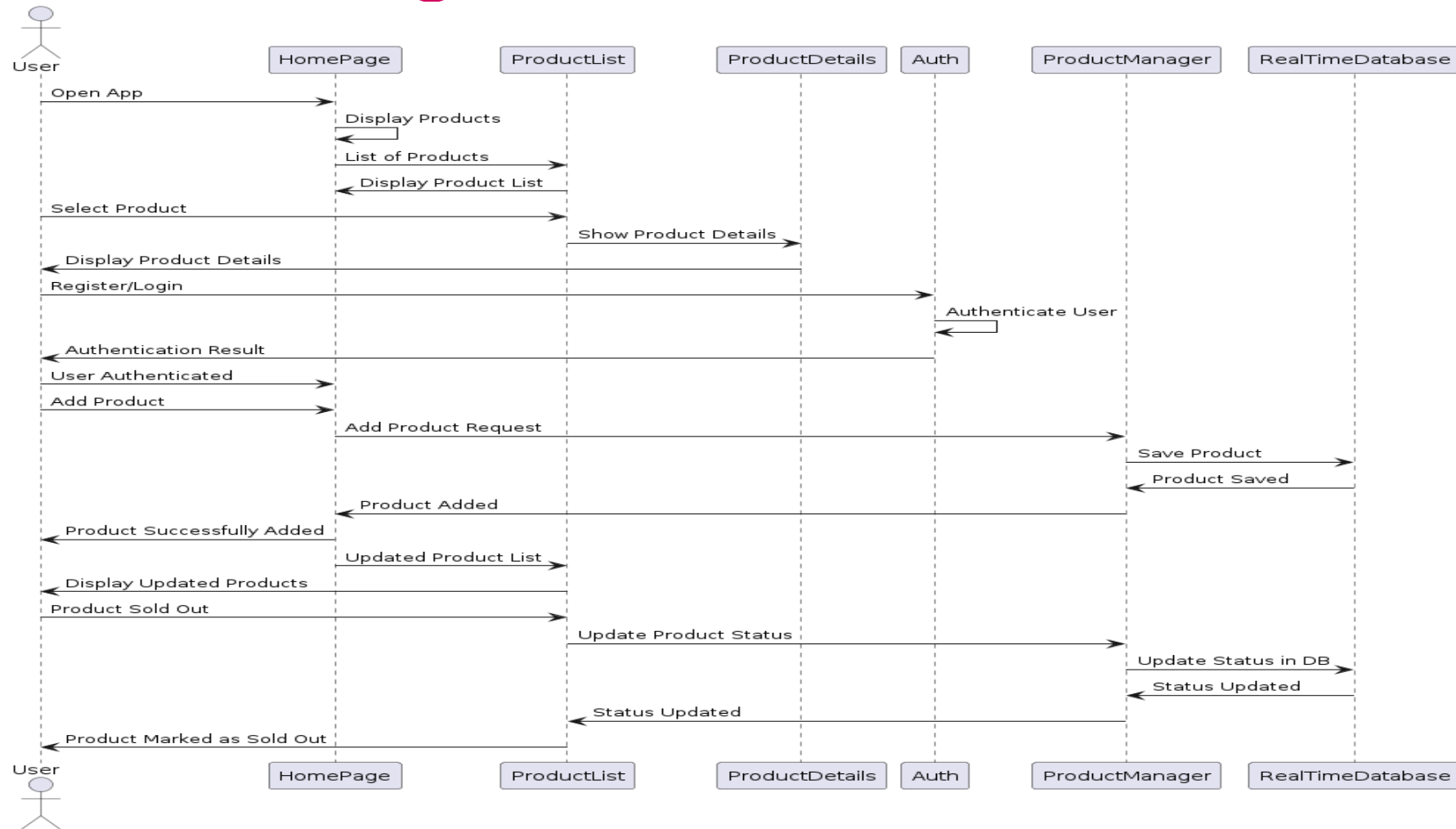- Seamless product management

Introduction

E-sale is an ecommerce android application that simplifies user registration, profile and product management and sales activities

APP OVERVIEW

UML Diagram

RegFragment

- User user
- DatabaseReference mDatabase
- FirebaseAuth mAuth

+void signUp()

LoginFragment

- String Email
- String Password
- FirebaseAuth mAuth

+void onSignInClicked()

ShowFragment

+ShowFragment()
+ShowFragment(Product product)

HomeFragment

- DatabaseReference databaseReference
- RecyclerView recyclerViewProducts
- DatabaseReference mDatabase

+void fetchSellerProducts()

UploadFragment

- String Description
- String ProductName
- String photoURL
- DatabaseReference mDatabase
- FirebaseAuth mAuth

+void uploadProductPhoto()

ProfileFragment

- User user
- RecyclerView recyclerViewProducts
- DatabaseReference mDatabase
- FirebaseAuth mAuth

-void fetchUserData()
+void updateUserData()
-void fetchSellerProducts()
+void logout()

User

-String username
-String email
-Integer phone
-String address
-String password

+void AccInfo(String username, String email, String phone, String address, String password)
+String getUserName()
+String getEmail()
+String getPhone()
+String getAddress()

DatabaseReference

+DatabaseReference child(String path)
+void setValue(Object value)

FirebaseAuth

+static FirebaseAuth getInstance()
+Task createUserWithEmailAndPassword(String email, String password)
+void signOut()
+Task signInWithEmailAndPassword(String email, String password)

Product

+void setId(String id)
+void setOwnerID(String ownerID)
+String getDescription()
+String getPhotoUrl()

uses · creates · uses · uses · uses · "navigates to" · "navigates to" · uses · uses · uses · uses · uses · uses · uses · uses

Sequence Diagram

Homepage

Frontend

- Image: Product image

- Title: Product name

Backend

- HomeFragment: A class that sets up the layout and data fetching mechanisms for displaying the products.

- HomeProduct: A class that models a product with attributes for name, photo URL, owner ID, and description.

- HomeProductAdapter: An adapter that displays a list of HomeProduct objects in a RecyclerView, allowing users to see product details and images, and navigate to a detailed view of the selected product.

Homepage (Design pattern)

- **Singleton**

> **HomeProductAdapter** class implements singleton pattern to ensure that only one instance of product list is active on homepage despite the list being frequently updated by the sellers

Homepage (Design pattern)

HomeProductAdapter

-List products
-static FragmentManager fragmentManager
-static HomeProductAdapter instance

-HomeProductAdapter(List products)
+static HomeProductAdapter getInstance(List products)
+void updateProducts(List products)

contains ↓

HomeProduct

-String name
-String photoUrl
-String ownerID
-String description

HomeProduct()
HomeProduct(String name, String description, String photoUrl)
+String getName()
+void setName(String name)
+String getDescription()
+void setDescription(String description)
+String getPhotoUrl()
+void setPhotoUrl(String photoUrl)
+String getOwnerID()
+void setOwnerID(String ownerID)

Singleton (class diagram)

Homepage (Design patterns)

Single tap

(Transition from homepage
to product details)
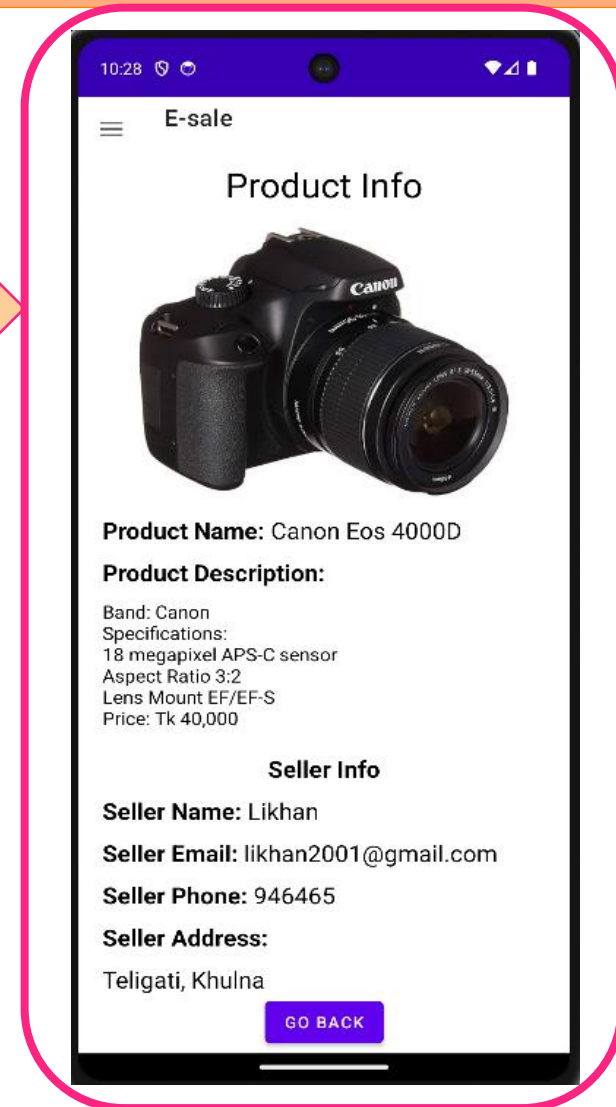
Product and Seller Details

Frontend

- Image: Product image

- Details: Product name, description and Seller's profile information

Backend

- HomeShowFragment: The class displays detailed information about a selected product, including the product's name, description, image, and the seller's contact information, and provides a button to navigate back to the HomeFragment.
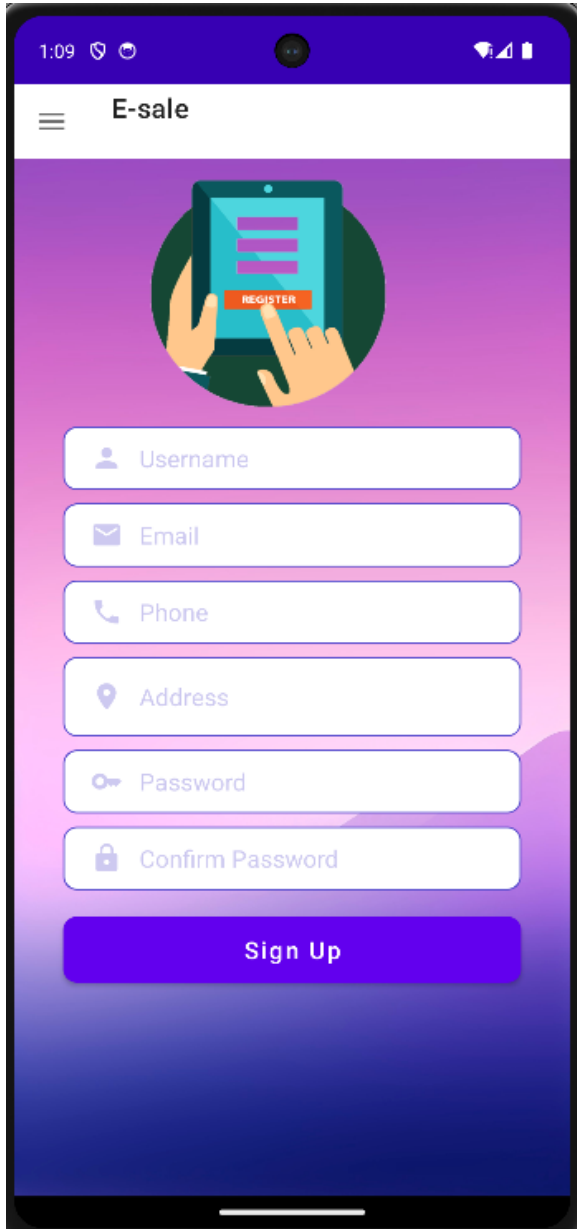
Signup

Frontend

- Fields: Username, Email, Phone, Address, Password and Confirm password fields

- Signup button: Button dedicated for signup action

Backend

- RegFragment: A class for -

  - Collecting user details

  - Creating a new user with Firebase Authentication

  - Saving user information in Firebase Realtime Database

- User: A class for a user's account information, including their username, email, phone, address, and password.
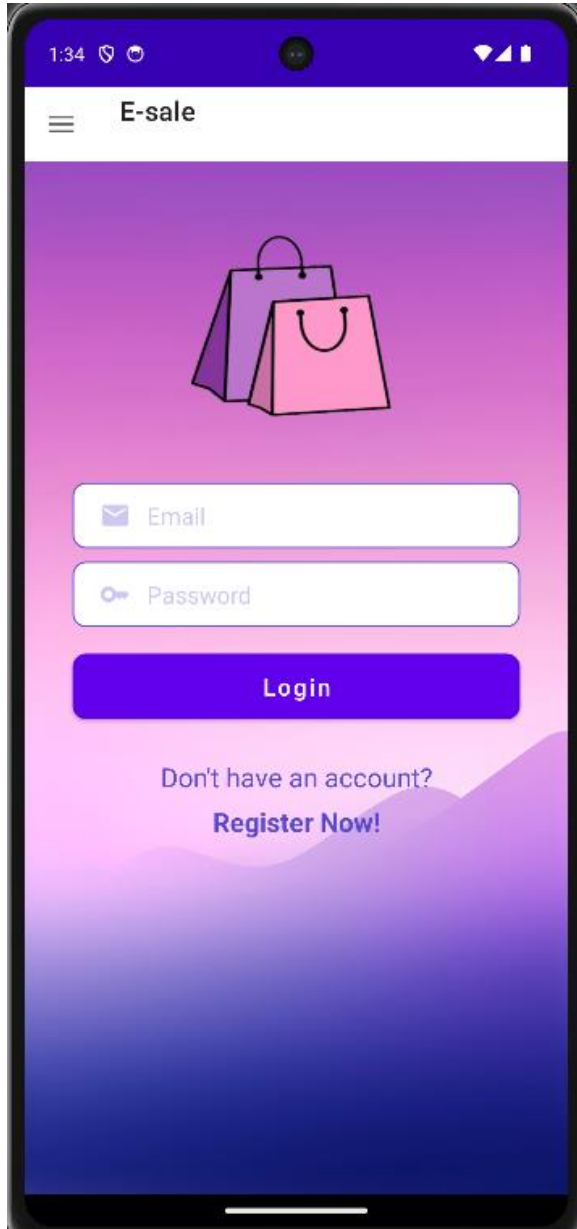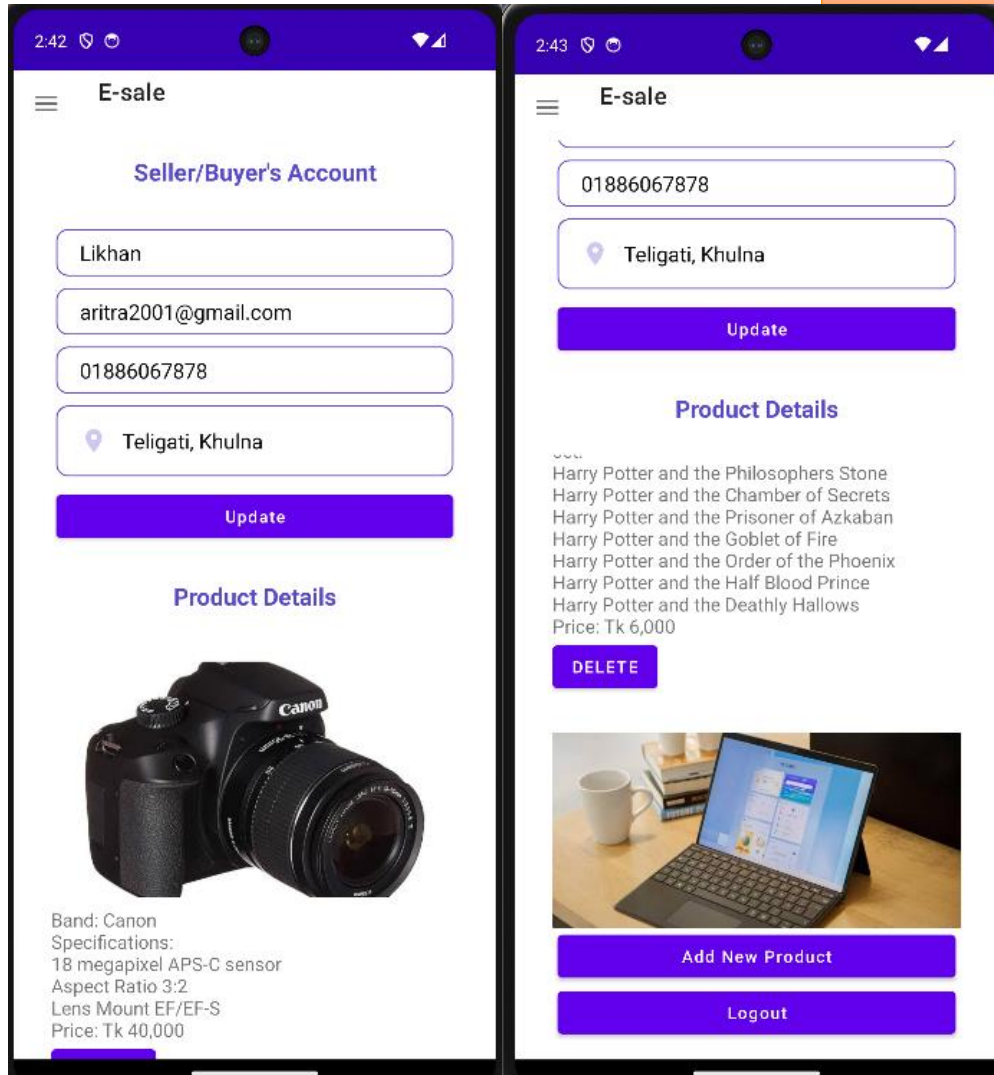
Login

Frontend

- Fields: Email and password fields

- Login button: Button dedicated for login action

Backend

- LoginFragment: A class for -

  - Collecting user details

  - Authenticating user using Firebase Authentication

  - Navigates to ProfileFragment upon successful login

  -Shows failure message if not authenticated

Profile

Frontend

- Fields: Username, email, phone and address fields

- Update button: Button dedicated for profile info update

- Image with description: Each product's image and its description including price, deletion is allowed

- Add New Product button: Button for adding new product

- Logout button: For logging out the user
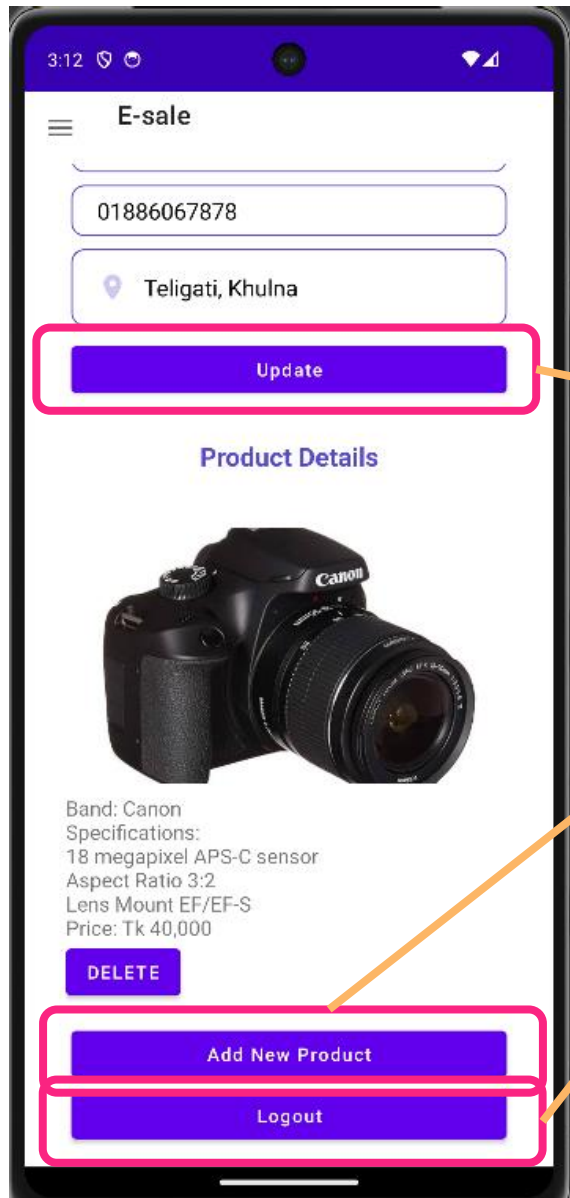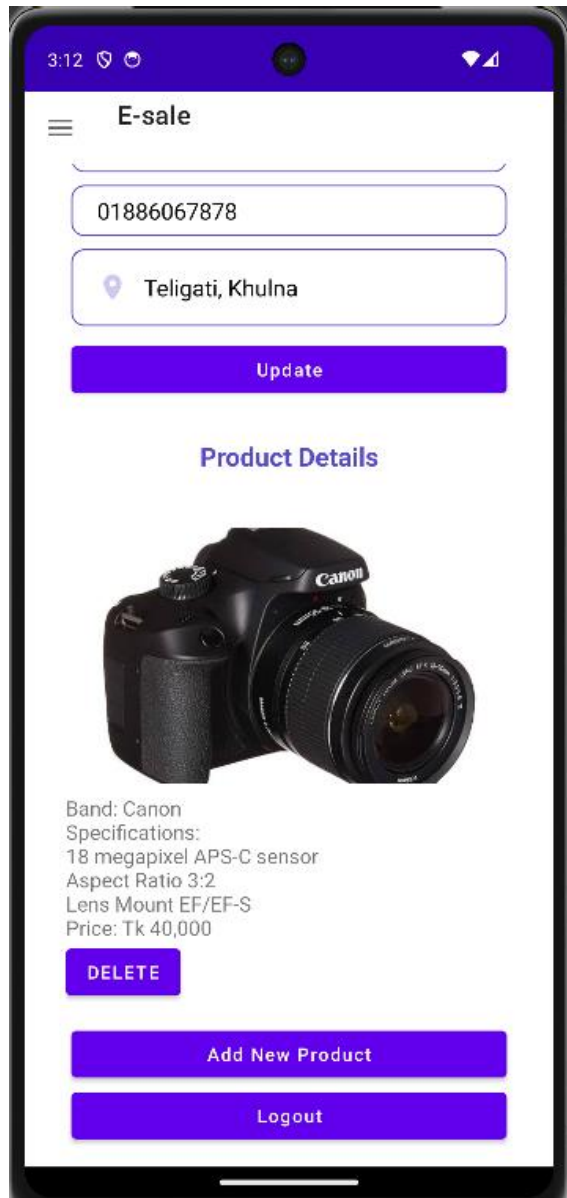
Profile
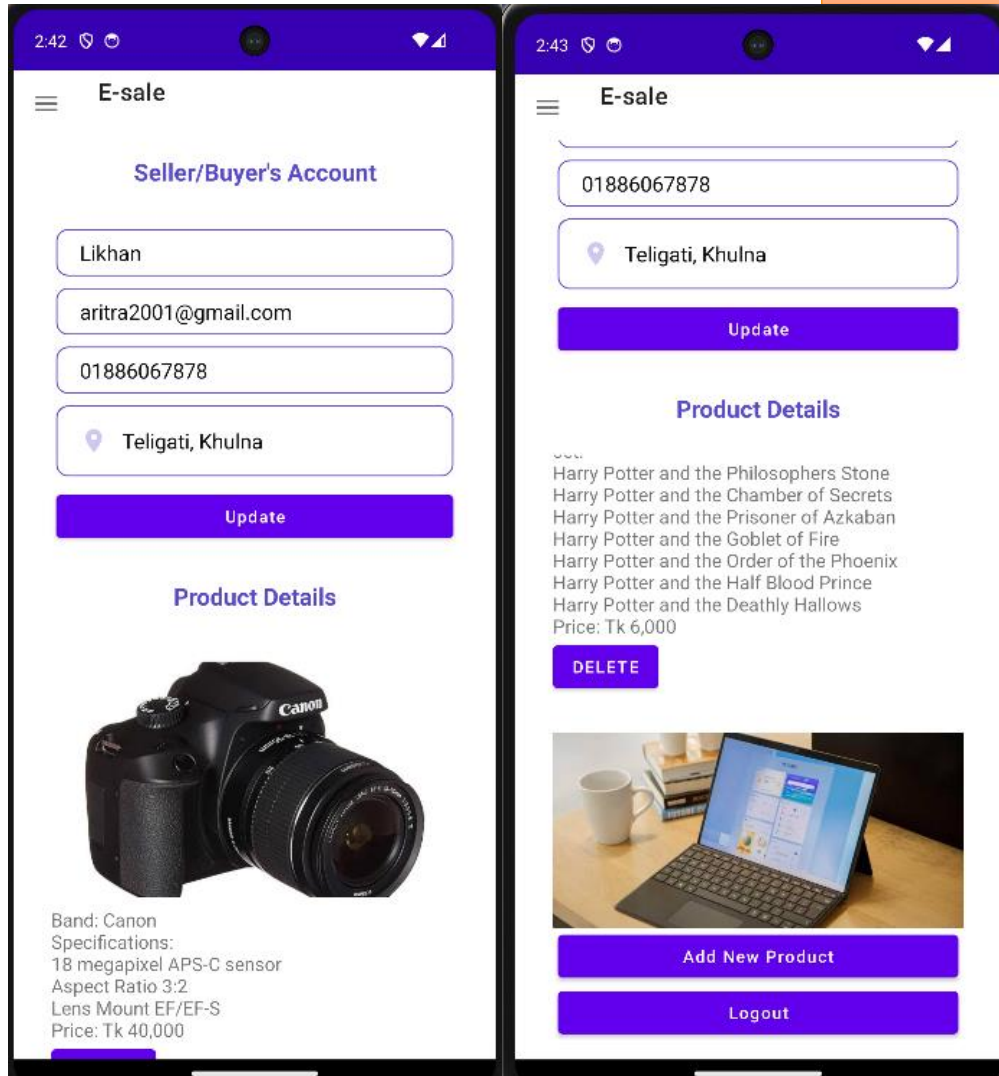
Backend:

- ProfileFragment: The class enables a user to –
  - view and update their profile information
  - view a list of their products
  - add new product
  - log out
- Product: The class represents a product with properties (unique ID, description, photo URL, and owner ID).
- ProductAdapter: A class that manages a list of `Product` objects in a `RecyclerView`, displaying their details and allowing users to view more details or delete products, with interactions supported by a `FragmentManager`.
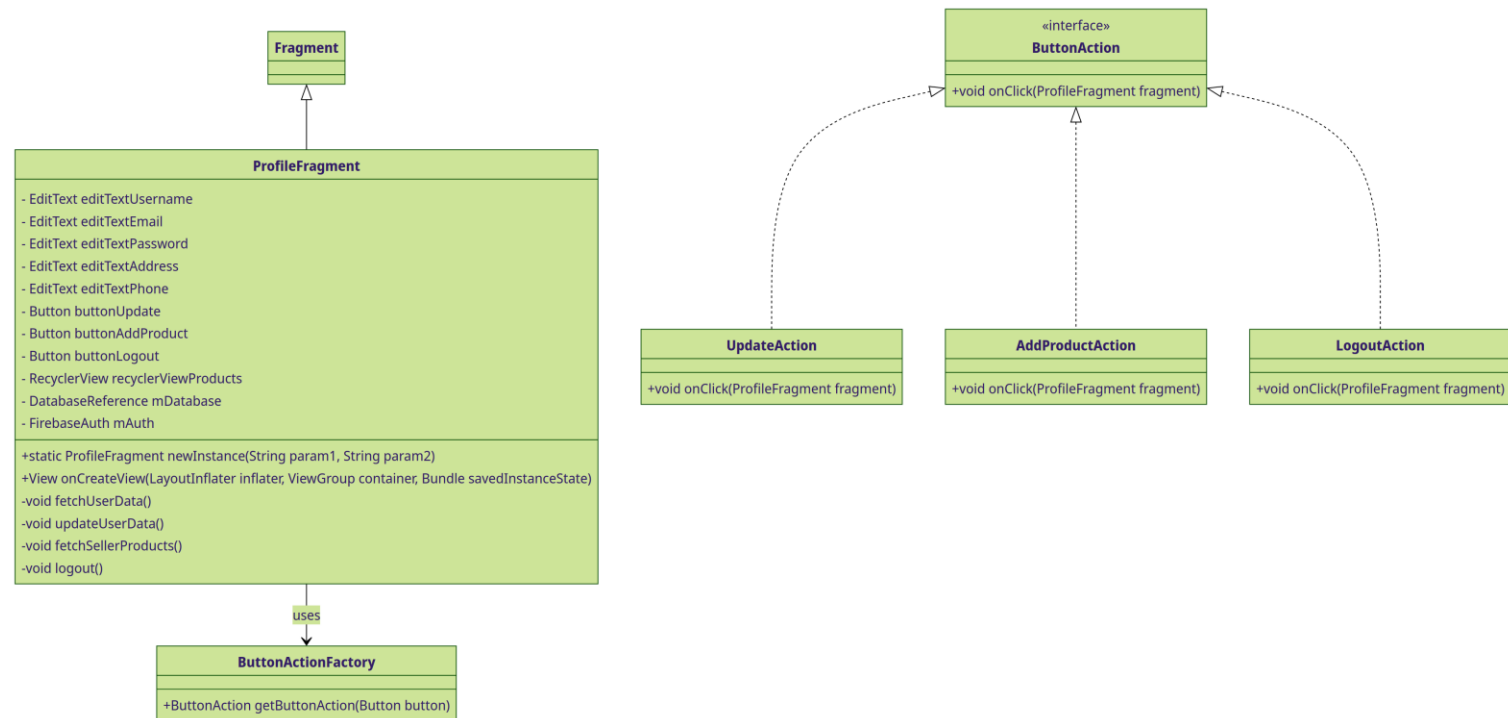
Profile (Design pattern)

- **Factory**

The factory pattern is implemented in the **ButtonActionFactory** class, which creates instances of different **ButtonAction** implementations (**UpdateAction**, **AddProductAction**, **LogoutAction**) based on the button that was clicked.

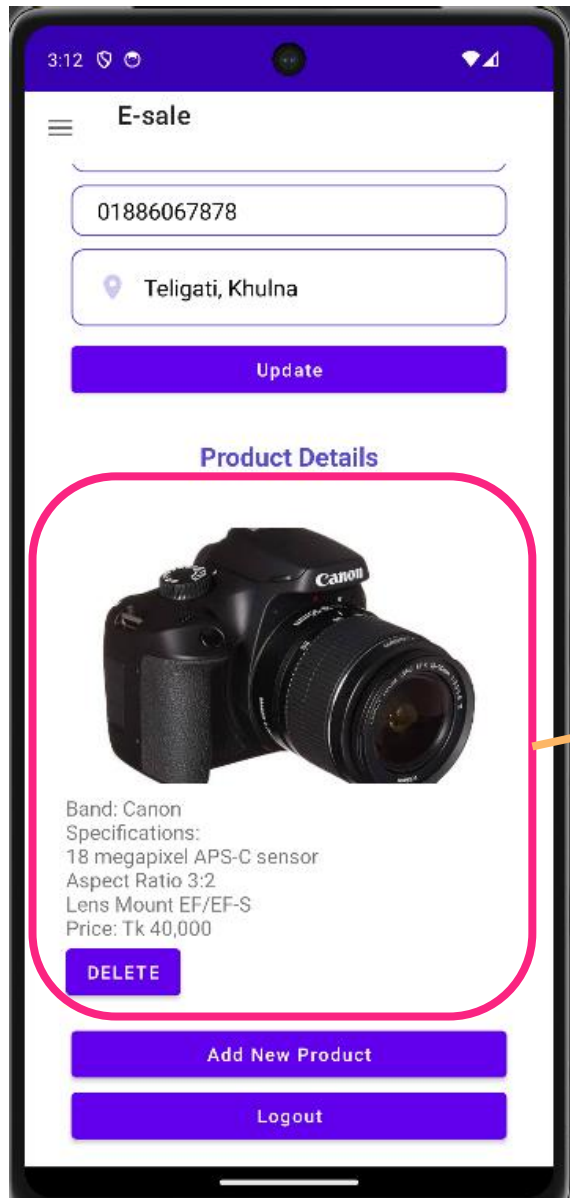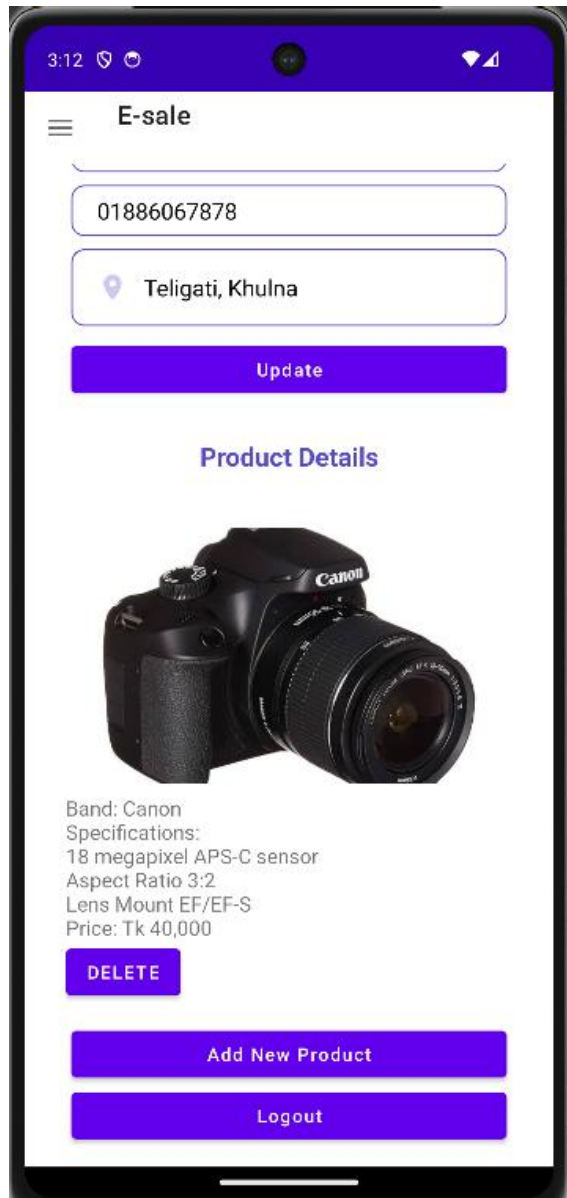# Profile (Design pattern)

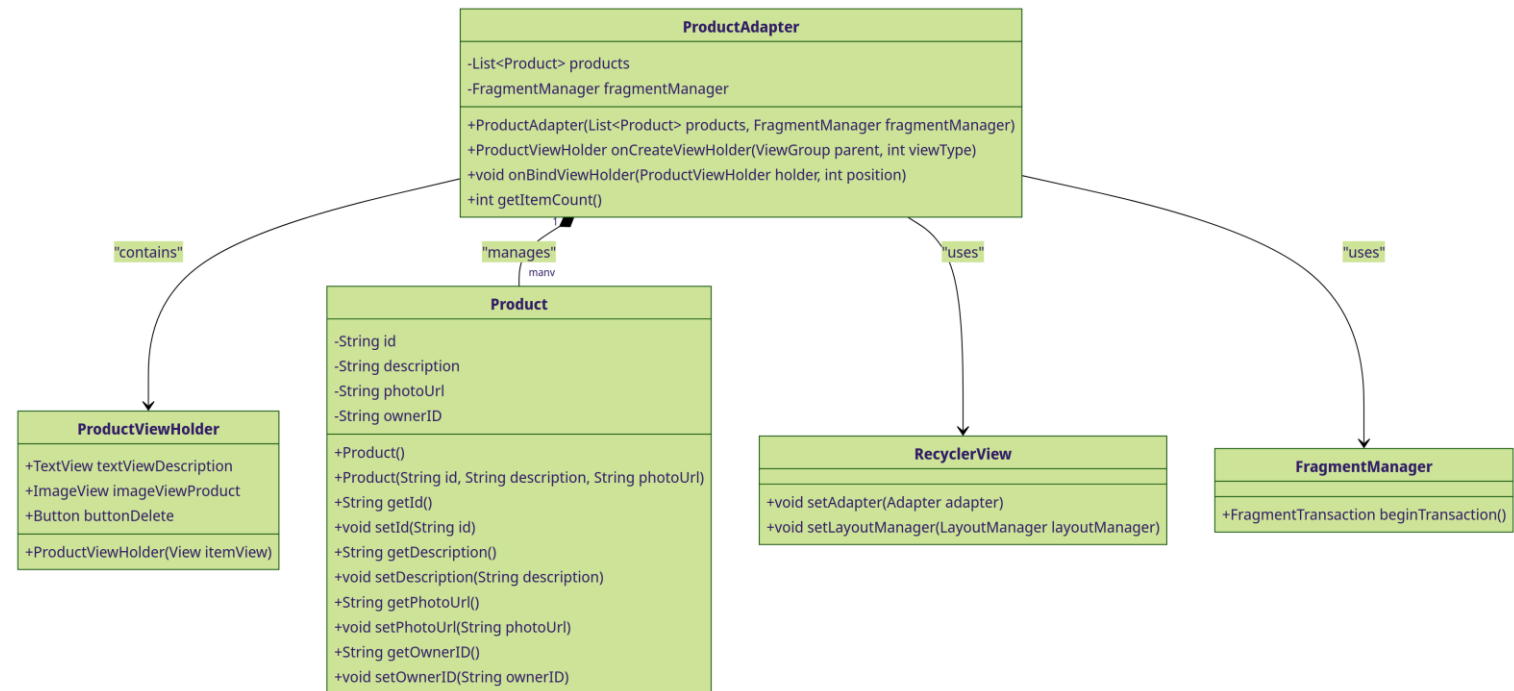**Factory (class diagram)**

Profile (Design pattern)

- **Adapter**

The adapter design pattern is implemented in the **ProductAdapter** class, which acts as an intermediary to fit in a list of **Product** objects into individual list items displayed by the **RecyclerView**.

Profile (Design pattern)

ProductAdapter

-List<Product> products
-FragmentManager fragmentManager

+ProductAdapter(List<Product> products, FragmentManager fragmentManager)
+ProductViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
+void onBindViewHolder(ProductViewHolder holder, int position)
+int getItemCount()

"contains"   "manages"   "uses"   "uses"
                     manv

ProductViewHolder

+TextView textViewDescription
+ImageView imageViewProduct
+Button buttonDelete

+ProductViewHolder(View itemView)

Product

-String id
-String description
-String photoUrl
-String ownerID

+Product()
+Product(String id, String description, String photoUrl)
+String getId()
+void setId(String id)
+String getDescription()
+void setDescription(String description)
+String getPhotoUrl()
+void setPhotoUrl(String photoUrl)
+String getOwnerID()
+void setOwnerID(String ownerID)

RecyclerView

+void setAdapter(Adapter adapter)
+void setLayoutManager(LayoutManager layoutManager)

FragmentManager

+FragmentTransaction beginTransaction()

Adapter (class diagram)

19

Unit Testing

Verifies the correct instantiation of the fragments and classes and the presence of interactive components within the fragments' layouts.

Unit test classes:

- HomeProductTest

- HomeProductAdapterTest

- HomeShowFragmentTest

- ShowFragmentTest

- LoginFragmentTest

- ProductTest

- RegFragmentTest

- UploadFragmentTest

Unit Testing

Integration Testing

Verifies that all the components within a fragment are working correctly together to serve the expected purpose.

Integration test classes:

- LoginFragmentIntegrationTest

- ProfileFragmentIntegrationTest

- RegFragmentIntegrationTest

UI Testing

Verifying the user interface of the application to ensure it behaves as expected and provides a good user experience.

UI test classes:

- HomeFragmentUITest

- HomeProductAdapterUITest

- ProductAdapterUITest

- ProfileFragmentUITest

- LoginFragmentUITest

- RegFragmentUITest

- UploadFragmentUITest

Thank you