AKDENIZ UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

Shopciafa

TUĞBA GÜLER

GRADUATION SENIOR PROJECT
INTERMEDIATE REPORT

SUPERVISOR: Asst. Prof. Dr. Mustafa Berkay Yılmaz

CONTENT

1- SHOPCIAFA – ONLINE SHOP APPLICATION USING FACE DETECTION TECHNOLOGY

1.1 Project Description

The word "Shopciafa" was derived by combining the words shop and faccia. Faccia is an Italian word and it means face. The project's name is "Shopciafa". Because the subject chosen as the senior project; is to design an online shop application using face detection technology. The project name was created with inspiration from the subject.

Every person is private and they want to look beautiful both for themselves and those around them. Therefore, people go shopping and try to find the most suitable products for them. Technology is developing rapidly and becoming a part of our lives. People spend most of their time on phones and tablets. Most of the work can be done easily at the push of a button. One of them is undoubtedly shopping. Online shopping sites provide a great convenience for people who are in busy business pace, who do not like crowds, or who do not want to waste time walking around in stores. In this project, it is aimed to develop an application that will make it easier for users to find the product they are looking for.

Women shop more than men. For this reason, when you log into most online shopping sites, firstly women's products (clothes, shoes, accessories, cosmetics, etc.) are shown. In this project, the products that appear on the home page will be different for everyone. The user will log into the account and the user's picture should be uploaded to the profile page within the app. Nobody can see this picture except the user. The purpose of uploading a picture is to determine the person's gender and age. In line with this information, the products on the main page of the application will be the most suitable products for the user. When a male user enters the application, he will come across men's categories and products. When a female user enters the application, she will see women's products and categories. Besides, the person's age will be determined in the application. When a young person enters the system, he/she will see the products suitable for young fashion, and a middle-aged person will encounter age-appropriate products.

1.2 Mockups and Documentation for the Working Interface

This section shows the mockups of the project, and through these figures, information is given about how the application works**.**

Sign In, Sign Up, and Forgot Your Password Page is shown in Figure 1.1.1. If the user is using the application for the first time, they can register for the application from the sign up page. If user has logged in to the application before, user can continue to use the application using the sign in page. In case users forget their passwords, there is a field written forgot password on the Sign in page. Click here to go to the Forgot Your Password page and reset the password by entering the registered email address into the system. A mail is sent to the email address for the user to reset their password. The user changes his password by following the steps.

Figure 1.1.1 Sign In, Sign Up, and Forgot Your Password Page

When the user logs into the system, it is directed to the home page. At the top of the application there is a tabbar with categories. Just below there is a slider to inform the users about the discount campaigns. Since the user has not yet added a profile picture to the system, male and female products are displayed together. On the homepage, many products are shown to users under headings such as deals of the day, new season, best seller. Product picture, price information and brief descriptive information about the product are shown for users to have information about the products.

There is a hamburger menu on the home page that will make it easier for users to easily access other pages of the application. Users can easily follow up their orders by clicking on the field of my orders here. When they click on the My card field, they can see the products they have added to the basket and continue the purchase process. While the products they like are displayed in the My Wishlist section, my account section contains detailed information such as address information and past shopping information. In addition, by clicking the settings button here, the user can change the password or upload the profile picture to the application. If the user wants to exit the application, it will be sufficient to click on the sign out area.

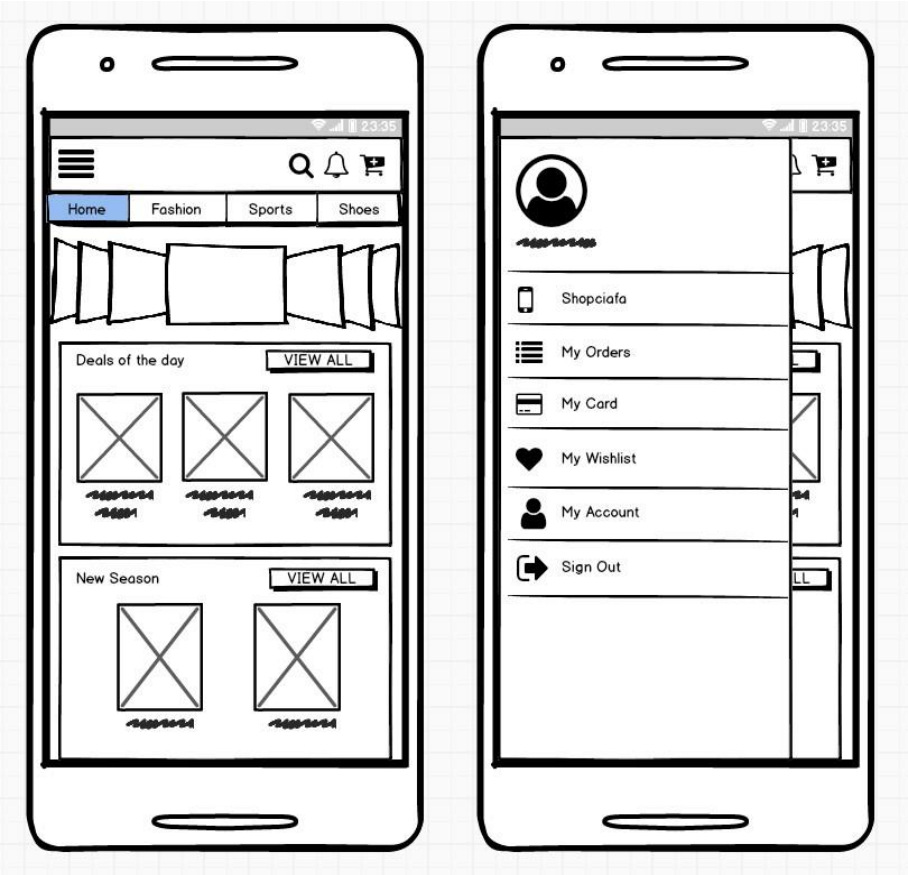


Figure 1.1.2 Home Page and Hamburger Menu

Not all products are listed on the homepage. Only a certain number of products are visible. To see all products, click on the view all button. The products in the deals of the day are given horizontally and the products are examined by scrolling. If there are less than 10 products, the view all button does not appear in the system. If there are more than 10 products, view all button appears.

When the view all button is clicked in the field where the products are displayed horizontally, the detailed name and brand of the products, how many votes and how many votes in total, the product price and the price of the product before the discount and whether it is cash on delivery are shown.

New season products are shown in grid structure. When the view all button is clicked, it is seen that the products in the same structure are listed. The product image, brand and product feature are specified. Also the price is shown.

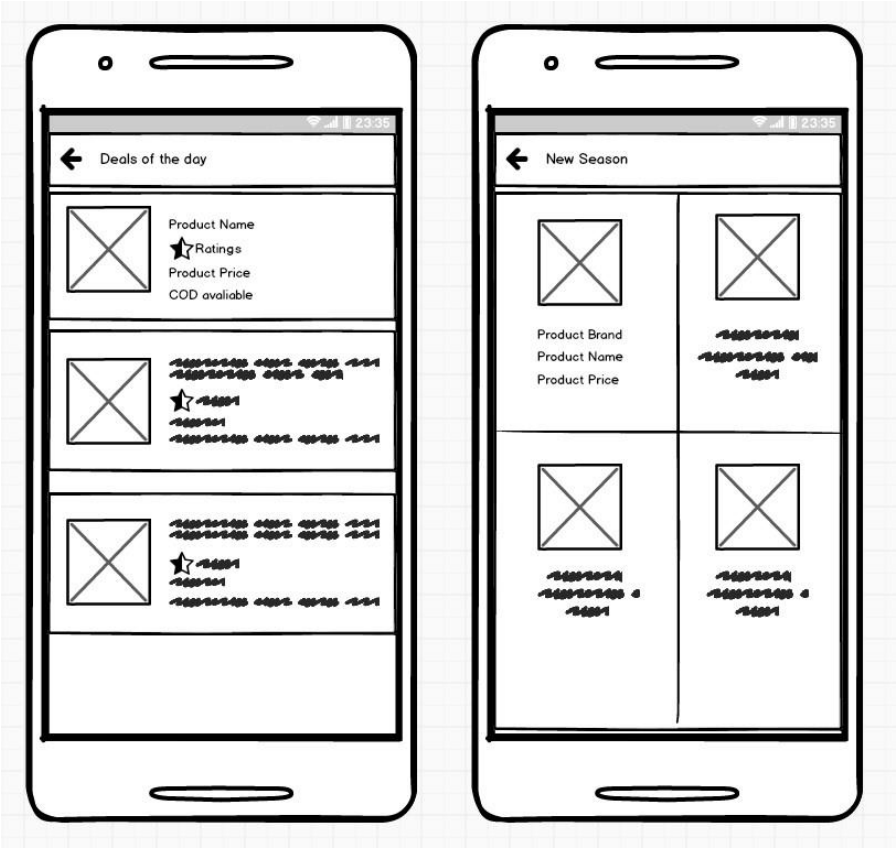


Figure 1.1.3 The View of the Pages Opened When Clicking the View All Buttons on the Home Page

The user has to click on the product to get detailed information about the product in the application. There are different photos of the product here. The full name of the product, the total number of votes, and how many people voted the price of the product and the price of the product without any discounts are shown.

The tab bar is located just below this information, and other information about the product is displayed here. Below is the detailed display of the product ratings. It depends on the votes used. Below here is a field for the user to vote on the product. As soon as the user votes, the data on the page is updated.

There are two buttons at the bottom of the page. If the user wants to add the product to the basket, click on the add to card button. The user has to click on the buy now button to instantly buy the product.

There is a button with a heart icon in the upper right corner of the page. The user can determine the products user likes by clicking this button. Popular products are listed on the My Wishlist page.

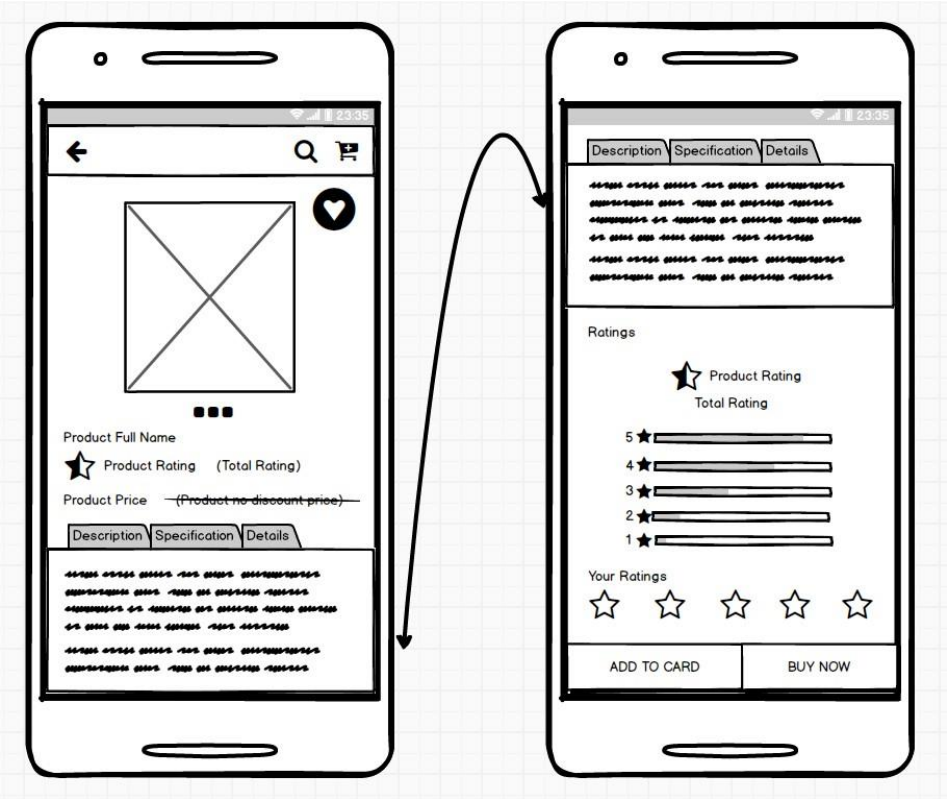


Figure 1.1.4 Product Description

When the user wants to see the products user likes again, user clicks on the "My Wishlist" text in the hamburger menu. The product picture, the full name of the product, the number of votes and votes received, the product price and the price before the discount is included. Also, if there is cash on delivery, it will be shown here. The user may want to remove the favorite product from the favorites list in the future. In this case, he can remove the product from the list by clicking the trash can icon.

The user can follow the ordered product from My Orders section. This page contains the product image, product name and the date ordered. If the product has been rated before, the rating for the product will also appear here. Click on the ordered product to get more detailed information. Here, there is additional information about how many orders have been placed. The user can follow the stage of his order here. These stages are grouped under 4 titles. Ordered, packed, shipping and delivered. Below this, the address information to which the product will be delivered and the total amount paid are shown.

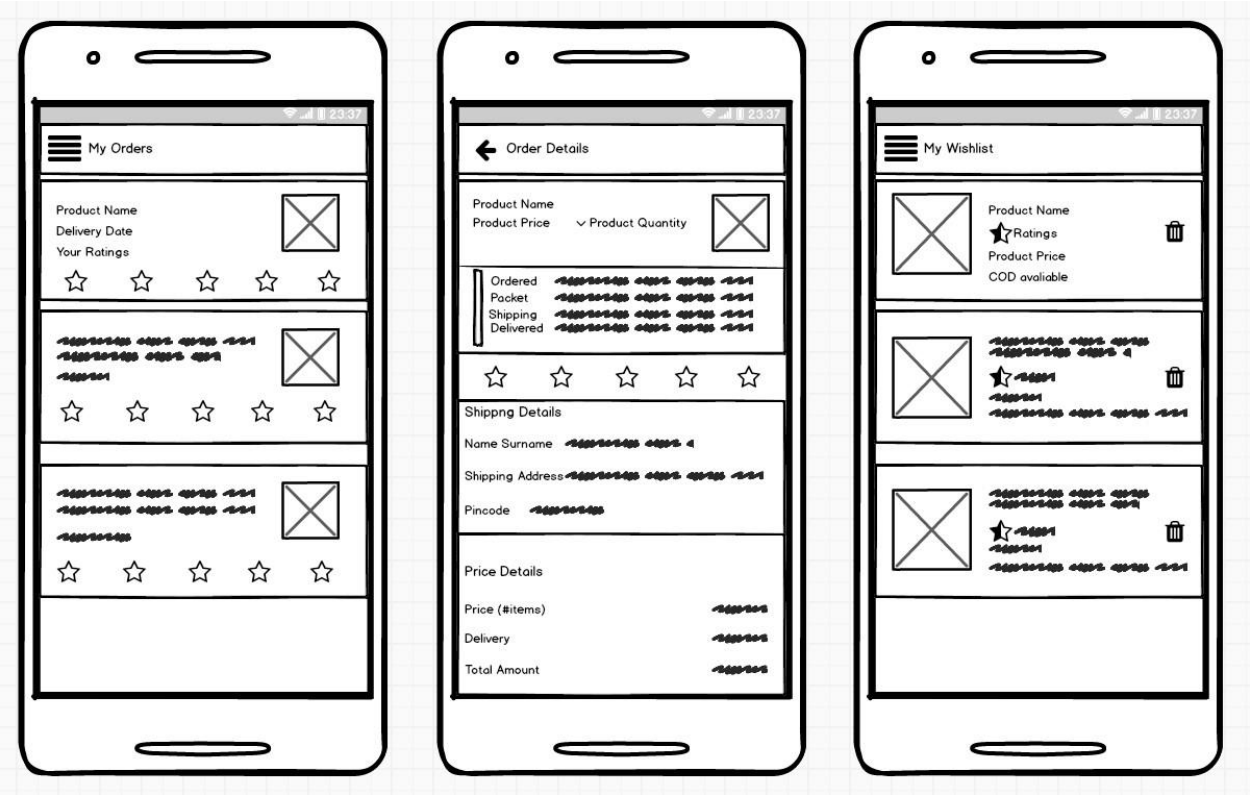


Figure 1.1.5 My Orders, Order Details and My Wishlist Page

The user added the product to the basket by clicking the add to card button on the page where the product details were given. The products added to the basket are displayed by clicking the basket icon at the top of the application or clicking the My Card section.

On the My Card page, there are pictures and full names of the products added, and there is a quantity field for the person to choose how many orders from the product. If the user does not want to buy the product, he can remove the product from the list by clicking the remove item button. At the bottom, there is price information.

If the user has a registered address information in the system, it is directed to the delivery page. It contains product information, price information and address information. The address information includes the name and surname, full address and pincode of the person who will receive the product. If the person will send a product to a different address, he can add a new address to the system by clicking the add or change new address button.

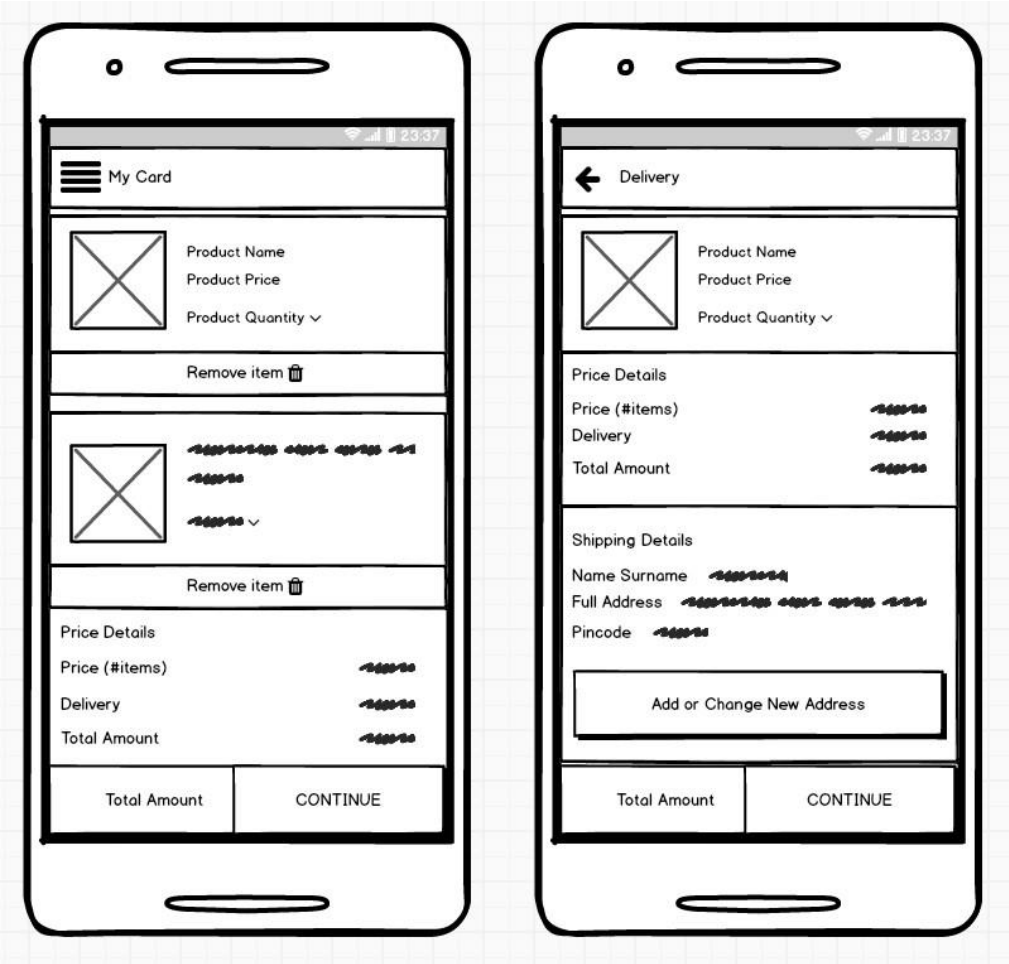


Figure 1.1.6 My Card and Delivery Page

Clicking on the Add or Change New Address button is directed to the My Addresses page. Here, the number of addresses and information of addresses added to the application is displayed. If the user wants to select one of the addresses he has saved, he must click on the address information. It will be displayed check box at the top right, indicating that the address has been selected. After the address is selected, the address information to be delivered is updated in the application by clicking the Deliver Here button. The updated address appears on the Delivery page. If the user wants to add a new address, the user is directed to the page where the address will be added by clicking the add a new address button.

The user enters all the information of the place where the product will be delivered. In addition, the person who will receive the product must enter the personal information. By clicking the Save button, the address information is added to the system and the my addresses page is updated.

If no address is added in the system, it is redirected to the add a new address page instead of the delivery page. After the address is saved in the system, it is directed to the delivery page.

Figure 1.1.7 My Addresses and Add a New Address Page

When the user comes to the my account page, user sees the profile picture and email address.

When the user orders any product; Includes which product was ordered and delivery information. If the user has previously placed an order in the system, the picture of the product that has been ordered and received is displayed on this page.

In addition, the user can view and edit the address information from here. If he wants to exit the application, he can click the sign out button at the bottom of the page.

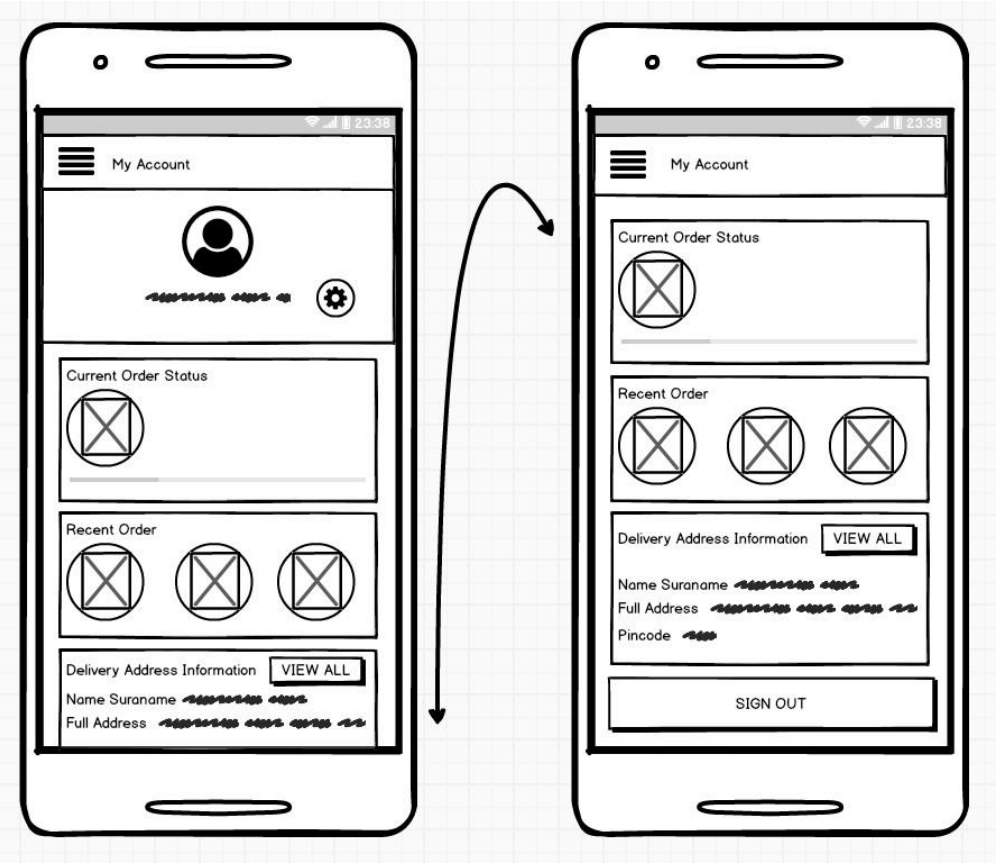


Figure 1.1.8 My Account Page

In Figure 1.1.8 there is a settings button next to the user's picture and e-mail address. Clicking on the settings button opens the page where the user information is updated. The user can upload a profile picture to the system in the user information section. He can delete his picture. Can edit the email address. Finally, it can update its information by pressing the update button. In the Password section, the user can change their password.

Figure 1.1.9 Update User Information

In the user information section, the user clicks the change photo button if user wants to upload the profile picture to the application or if a picture is loaded in the application and wants to change this picture. When the button is clicked, there are open camera buttons for taking snapshots for the user and upload image buttons for selecting pictures from the gallery. When the user takes the picture or selects it from the gallery, the picture appears where the picture icon is. The user determines the age and gender according to the picture by clicking on the process button. The specified gender and age process is displayed just below the button, and the button list products by age and gender appears. When the user clicks this button, user will be directed to the homepage. This time, the user will see the products in the system that best suit his age and gender on the homepage.

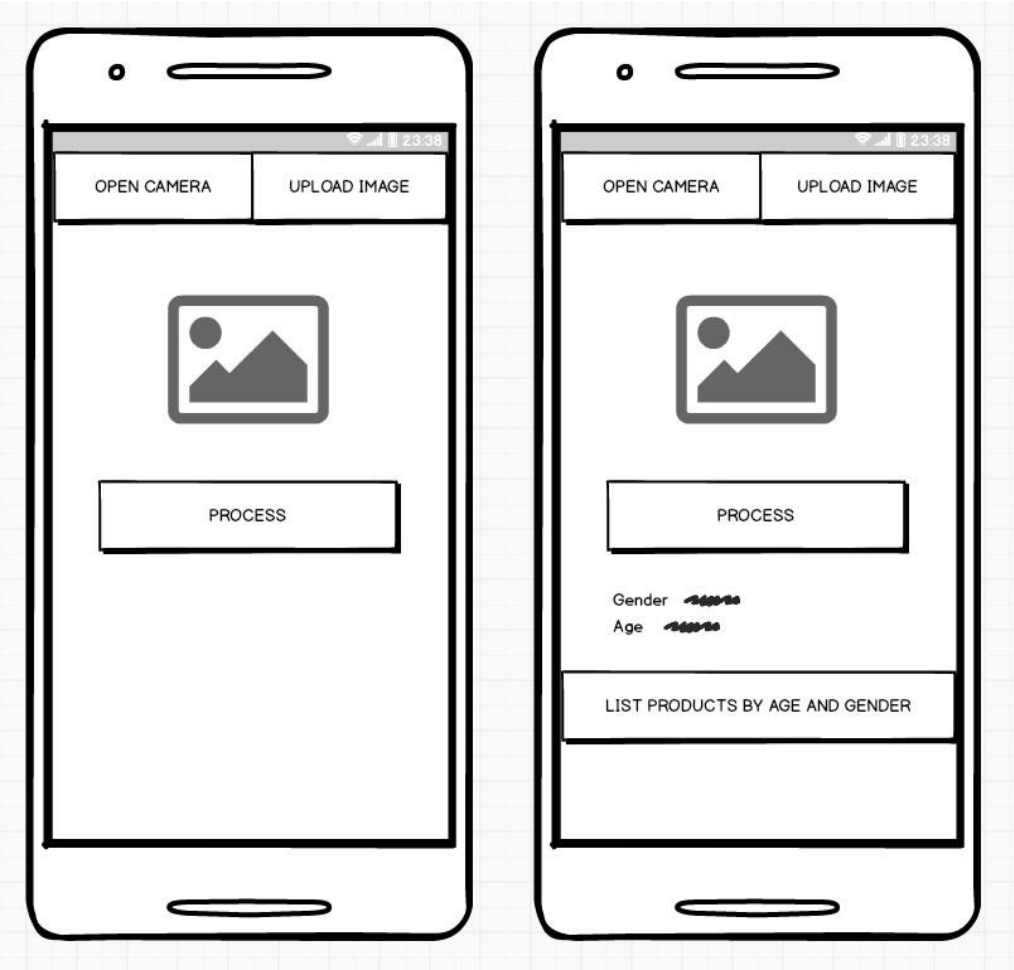
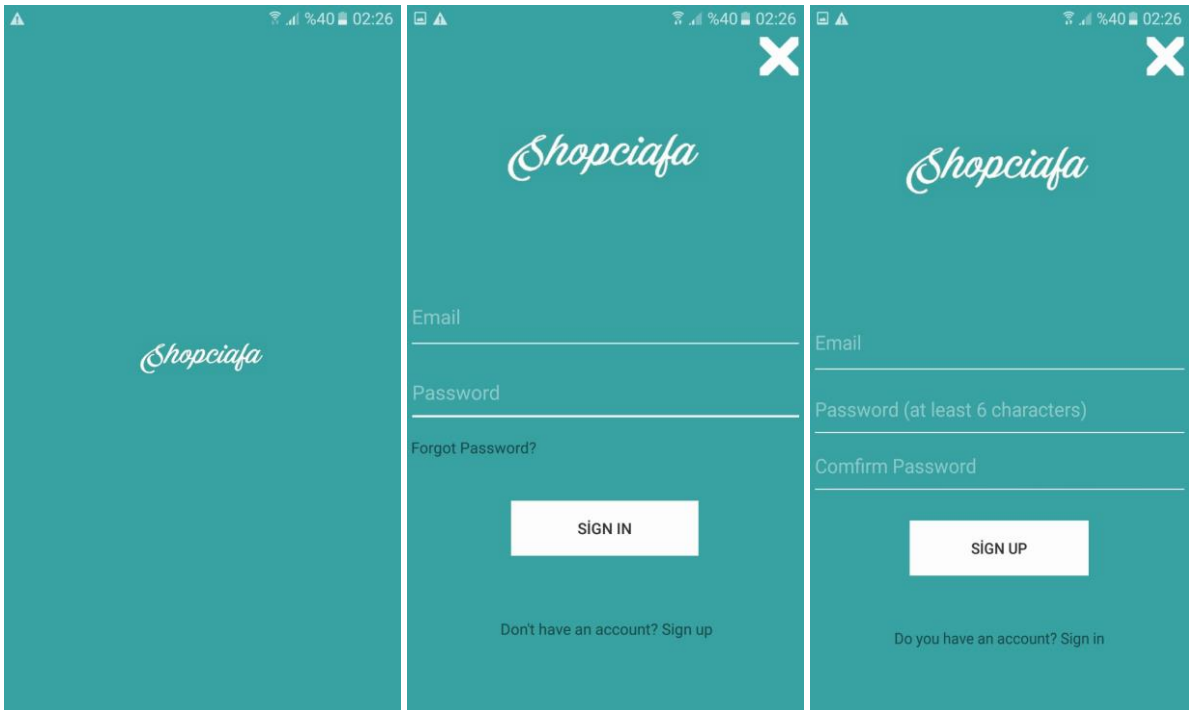
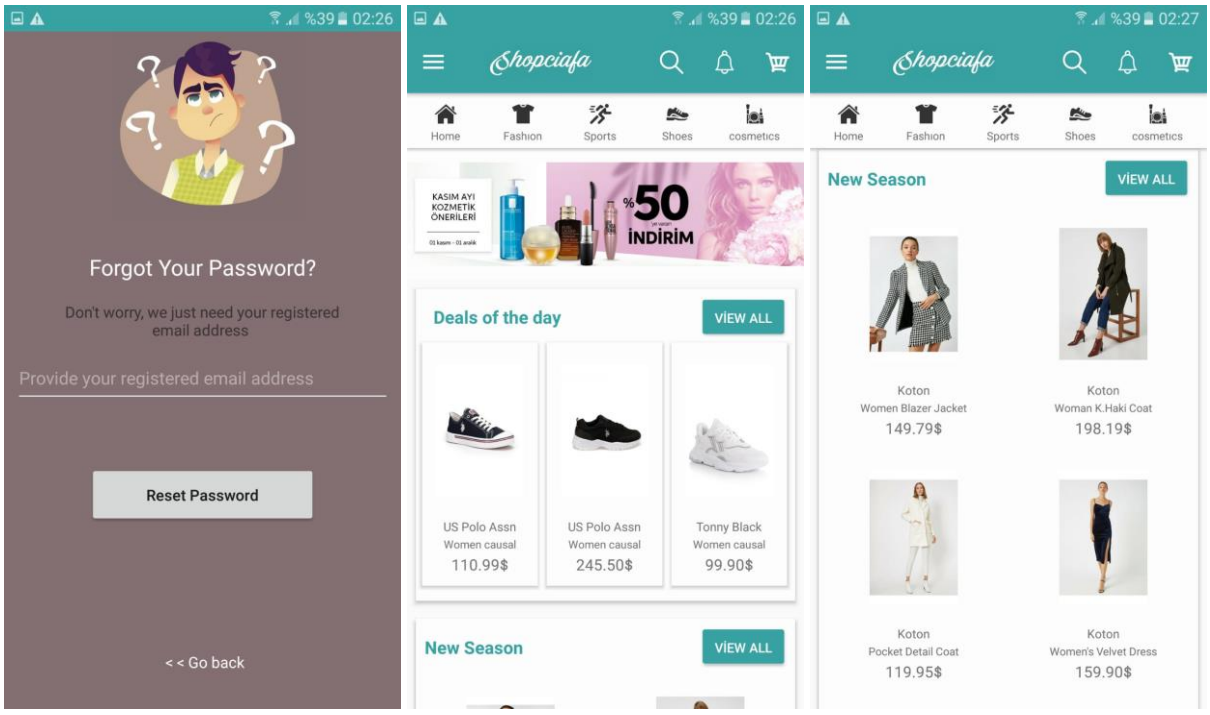
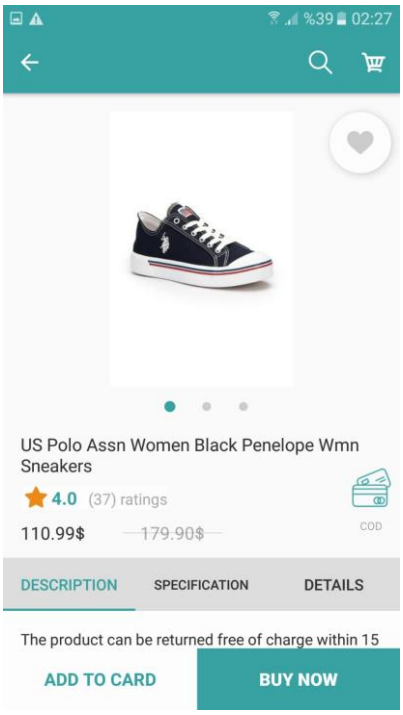


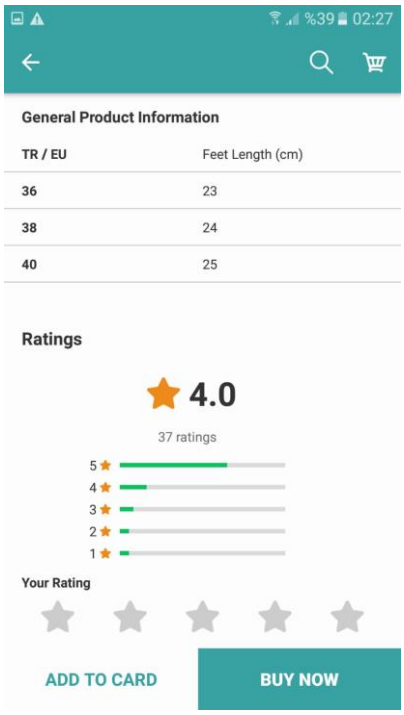
Figure 1.1.10 Changing the Profile Picture and Listing Products by Age and Gender in the Application

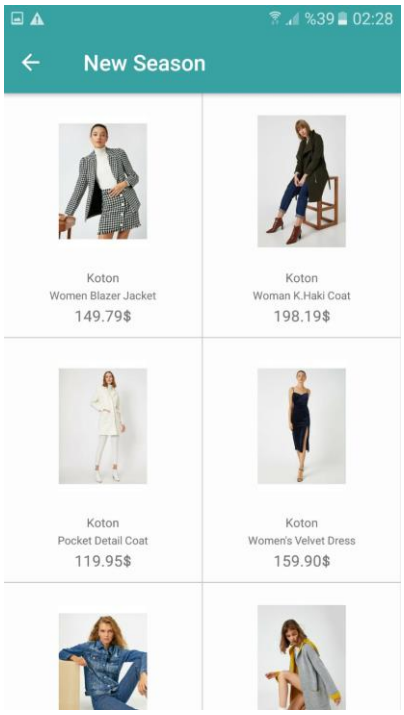
1.3 Application User Interfaces

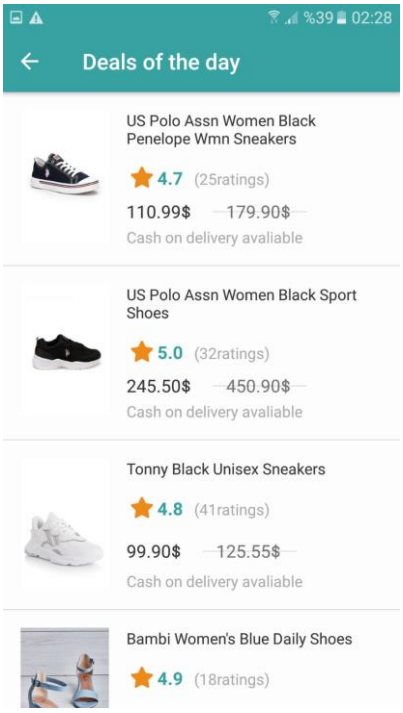


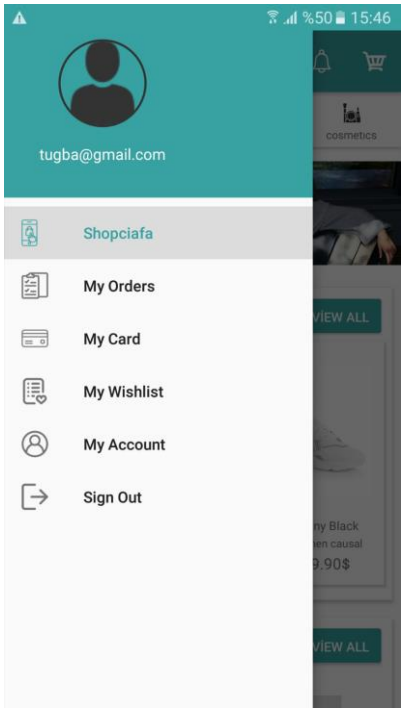


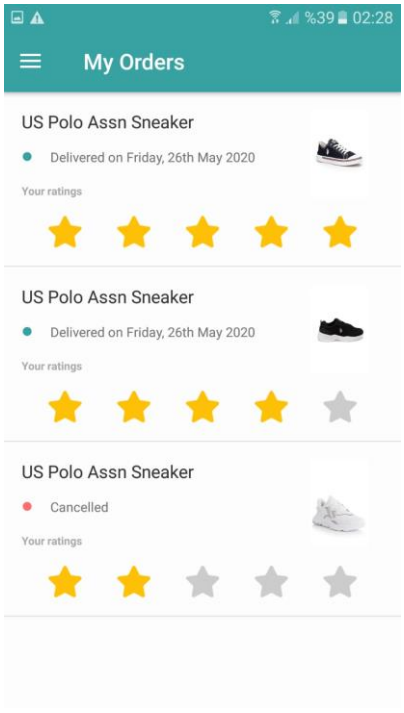

%39 02:27
US Polo Assn Women Black Penelope Wmn Sneakers
4.0 (37) ratings
110.99$ 179.90$
COD
DESCRIPTION
SPECIFICATION
DETAILS
The product can be returned free of charge within 15
ADD TO CARD
BUY NOW

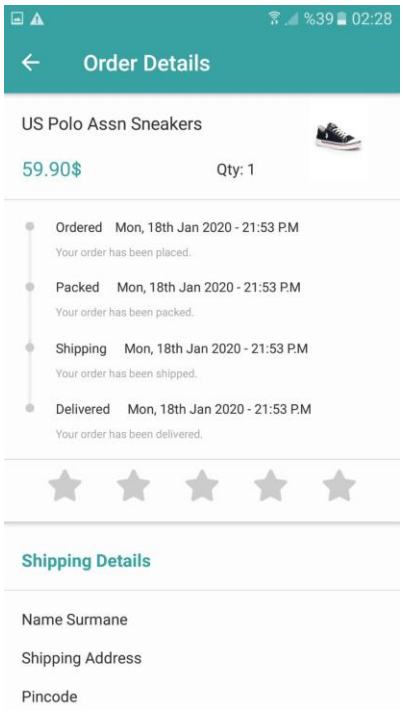

%39 02:27
General Product Information
TR / EU
Feet Length (cm)
36
23
38
24
40
25
Ratings
4.0
37 ratings
5
4
3
2
1
Your Rating
ADD TO CARD
BUY NOW

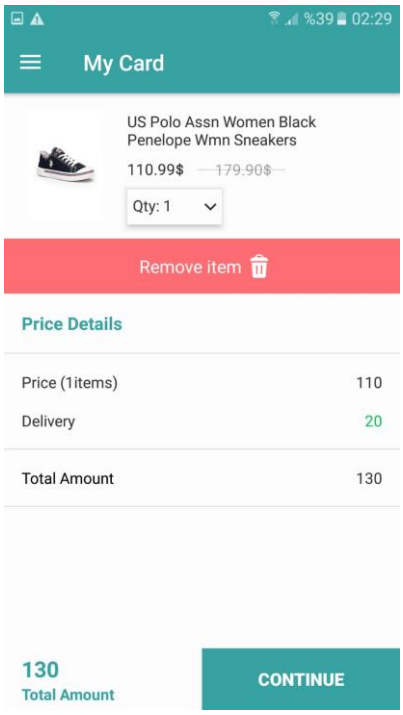

%39 02:28
New Season
Koton
Women Blazer Jacket
149.79$
Koton
Woman K.Haki Coat
198.19$
Koton
Pocket Detail Coat
119.95$
Koton
Women's Velvet Dress
159.90$

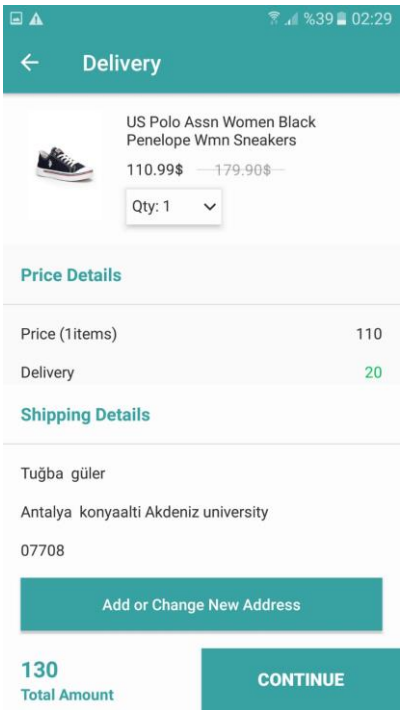

%39 02:28
Deals of the day
US Polo Assn Women Black Penelope Wmn Sneakers
4.7 (25ratings)
110.99$ 179.90$
Cash on delivery avaliable
US Polo Assn Women Black Sport Shoes
5.0 (32ratings)
245.50$ 450.90$
Cash on delivery avaliable
Tonny Black Unisex Sneakers
4.8 (41ratings)
99.90$ 125.55$
Cash on delivery avaliable
Bambi Women's Blue Daily Shoes
4.9 (18ratings)

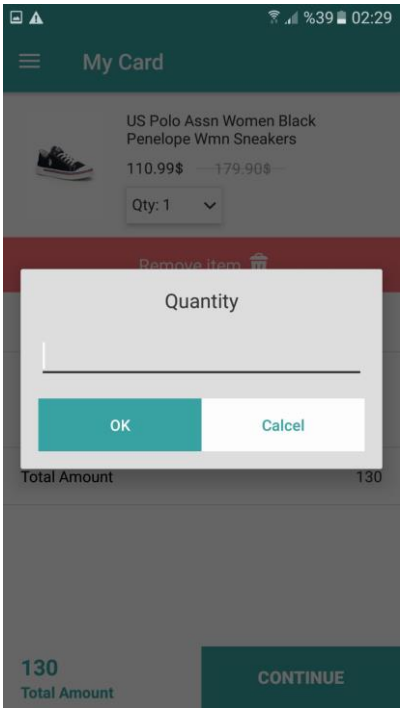

%50 15:46
tugba@gmail.com
Shopciafa
My Orders
My Card
My Wishlist
My Account
Sign Out

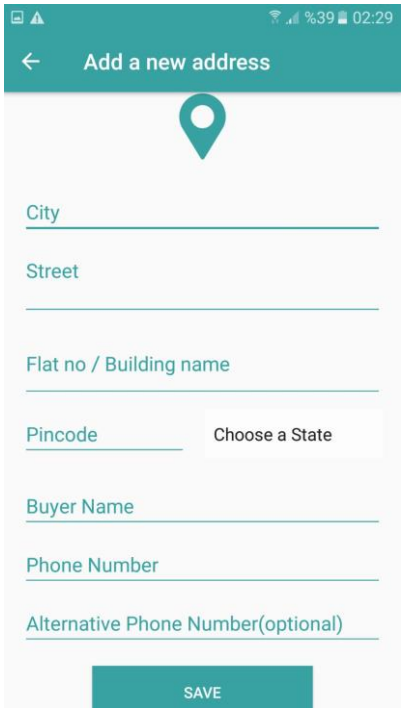

%39 02:28
My Orders
US Polo Assn Sneaker
Delivered on Friday, 26th May 2020
Your ratings
US Polo Assn Sneaker
Delivered on Friday, 26th May 2020
Your ratings
US Polo Assn Sneaker
Cancelled
Your ratings

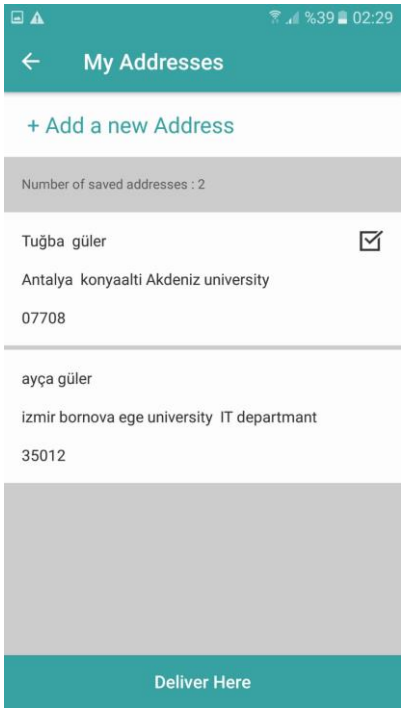

Order Details
US Polo Assn Sneakers
59.90$
Qty: 1
Ordered Mon, 18th Jan 2020 - 21:53 P.M
Your order has been placed.
Packed Mon, 18th Jan 2020 - 21:53 P.M
Your order has been packed.
Shipping Mon, 18th Jan 2020 - 21:53 P.M
Your order has been shipped.
Delivered Mon, 18th Jan 2020 - 21:53 P.M
Your order has been delivered.
Shipping Details
Name Surmane
Shipping Address
Pincode

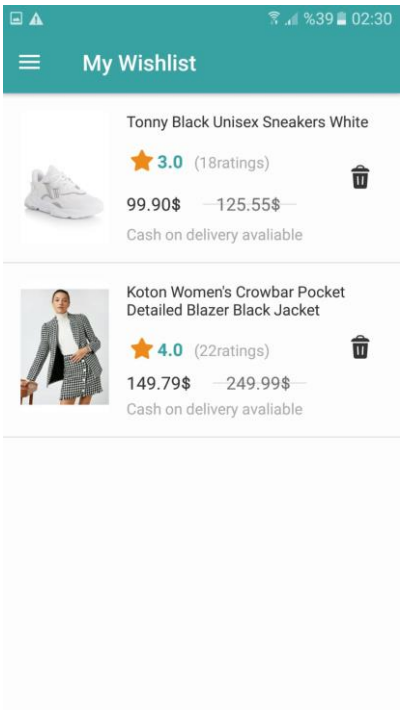

My Card
US Polo Assn Women Black Penelope Wmn Sneakers
110.99$ 179.90$
Qty: 1
Remove item
Price Details
Price (1items) 110
Delivery 20
Total Amount 130
130
Total Amount
CONTINUE

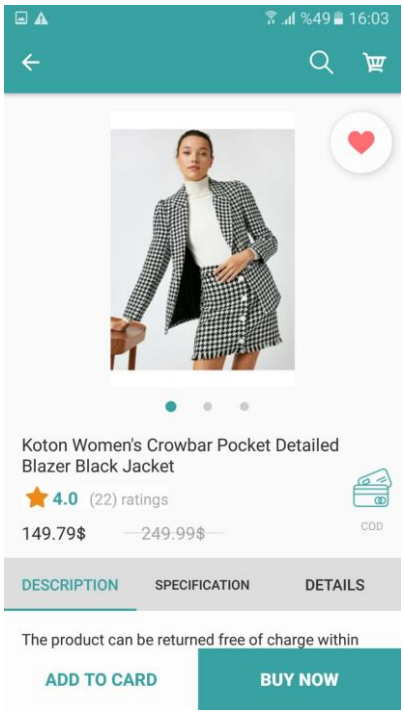

Delivery
US Polo Assn Women Black Penelope Wmn Sneakers
110.99$ 179.90$
Qty: 1
Price Details
Price (1items) 110
Delivery 20
Shipping Details
Tuğba güler
Antalya konyaalti Akdeniz university
07708
Add or Change New Address
130
Total Amount
CONTINUE

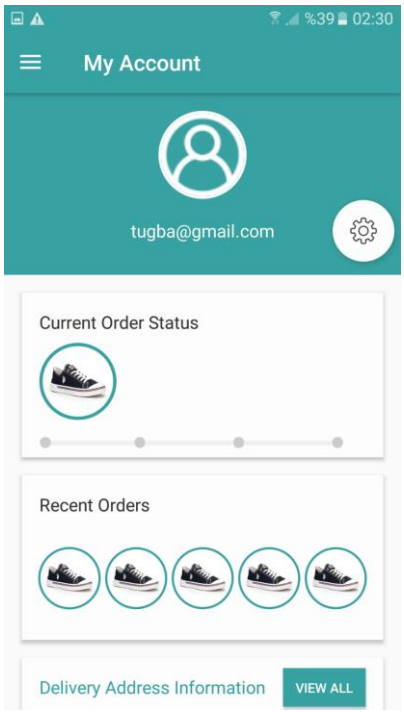

My Card
US Polo Assn Women Black Penelope Wmn Sneakers
110.99$ 179.90$
Qty: 1
Quantity
OK
Calcel
Total Amount 130
130
Total Amount
CONTINUE

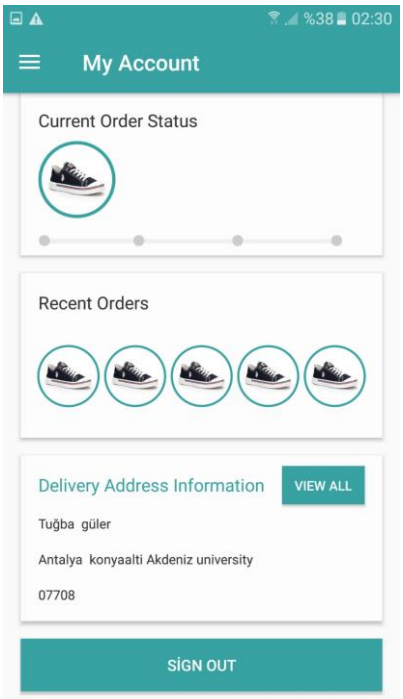

Add a new address
City
Street
Flat no / Building name
Pincode
Choose a State
Buyer Name
Phone Number
Alternative Phone Number(optional)
SAVE

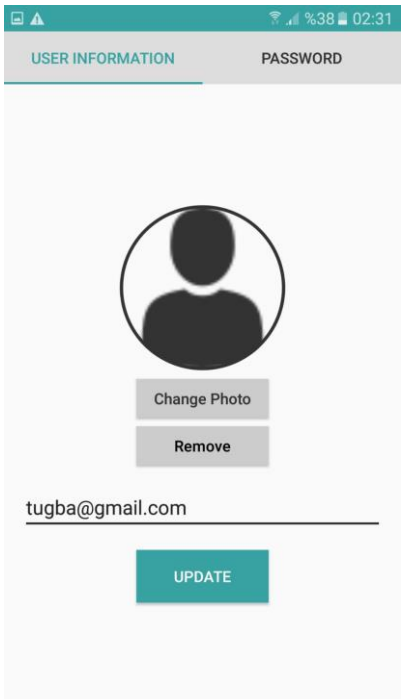

My Addresses
+ Add a new Address
Number of saved addresses : 2
Tuğba güler
Antalya konyaalti Akdeniz university
07708
ayça güler
izmir bornova ege university IT departmant
35012
Deliver Here

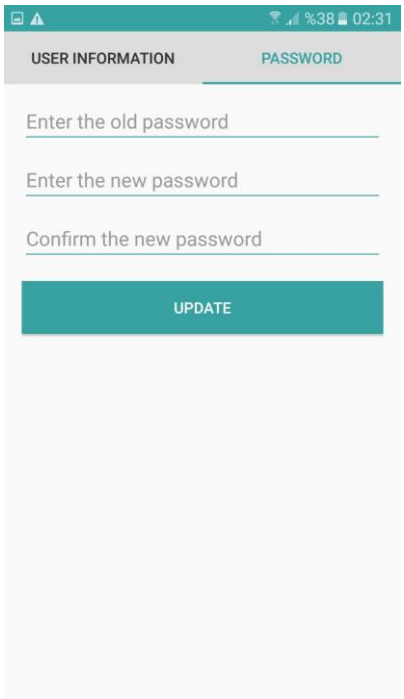

%39 02:30
My Wishlist
Tonny Black Unisex Sneakers White
3.0 (18ratings)
99.90$ 125.55$
Cash on delivery avaliable
Koton Women's Crowbar Pocket Detailed Blazer Black Jacket
4.0 (22ratings)
149.79$ 249.99$
Cash on delivery avaliable

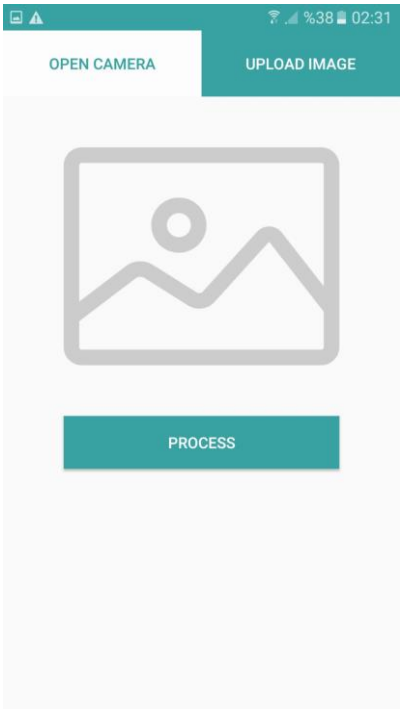

%49 16:03
Koton Women's Crowbar Pocket Detailed Blazer Black Jacket
4.0 (22) ratings
149.79$ 249.99$
COD
DESCRIPTION
SPECIFICATION
DETAILS
The product can be returned free of charge within
ADD TO CARD
BUY NOW

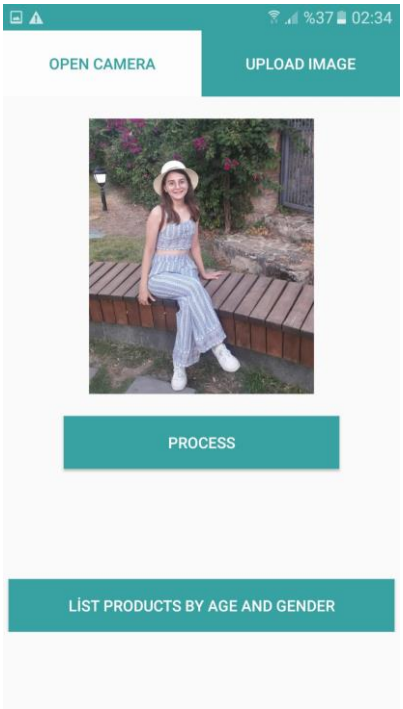

%39 02:30
My Account
tugba@gmail.com
Current Order Status
Recent Orders
Delivery Address Information
VIEW ALL

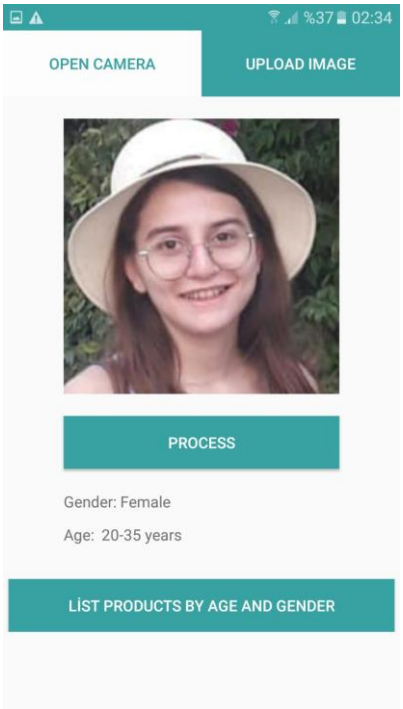

%38 02:30
My Account
Current Order Status
Recent Orders
Delivery Address Information
VIEW ALL
Tuğba güler
Antalya konyaalti Akdeniz university
07708
SİGN OUT

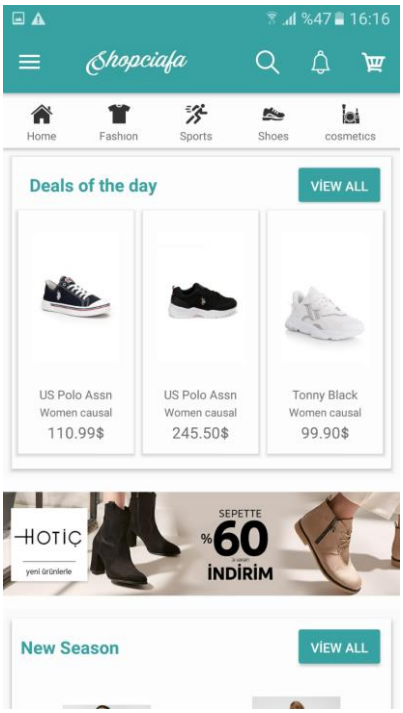

%38 02:31
USER INFORMATION
PASSWORD
Change Photo
Remove
tugba@gmail.com
UPDATE

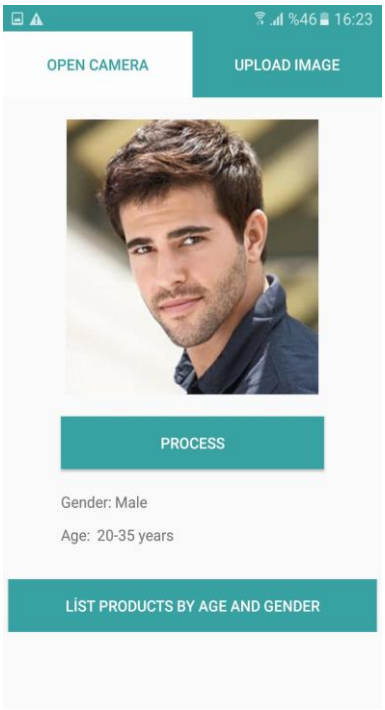

%38 02:31
USER INFORMATION
PASSWORD
Enter the old password
Enter the new password
Confirm the new password
UPDATE

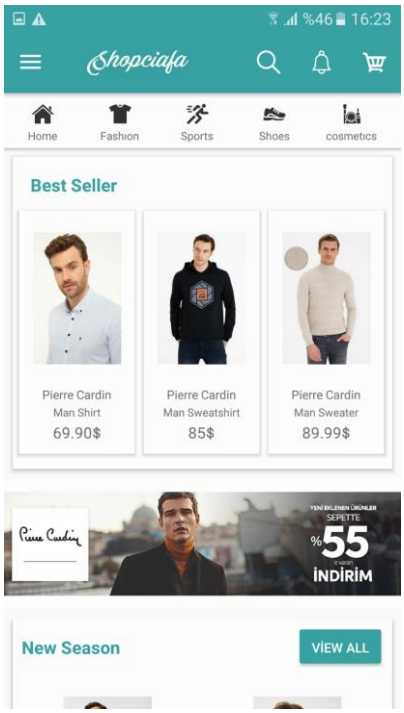

%38 02:31
OPEN CAMERA
UPLOAD IMAGE
PROCESS


%37 02:34
OPEN CAMERA
UPLOAD IMAGE
PROCESS
LİST PRODUCTS BY AGE AND GENDER


%37 02:34
OPEN CAMERA
UPLOAD IMAGE
PROCESS
Gender: Female
Age: 20-35 years
LİST PRODUCTS BY AGE AND GENDER


%47 16:16
Shopciafa
Home
Fashion
Sports
Shoes
cosmetics
Deals of the day
VIEW ALL
US Polo Assn
Women causal
110.99$
US Polo Assn
Women causal
245.50$
Tonny Black
Women causal
99.90$
HOTİÇ
SEPETTE
%60
İNDİRİM
New Season
VIEW ALL


%46 16:23
OPEN CAMERA
UPLOAD IMAGE
PROCESS
Gender: Male
Age: 20-35 years
LİST PRODUCTS BY AGE AND GENDER


%46 16:23
Shopciafa
Home
Fashion
Sports
Shoes
cosmetics
Best Seller
Pierre Cardin
Man Shirt
69.90$
Pierre Cardin
Man Sweatshirt
85$
Pierre Cardin
Man Sweater
89.99$
SEPETTE
%55
İNDİRİM
New Season
VIEW ALL

1.4 Explanation of the Implementation and the Methods

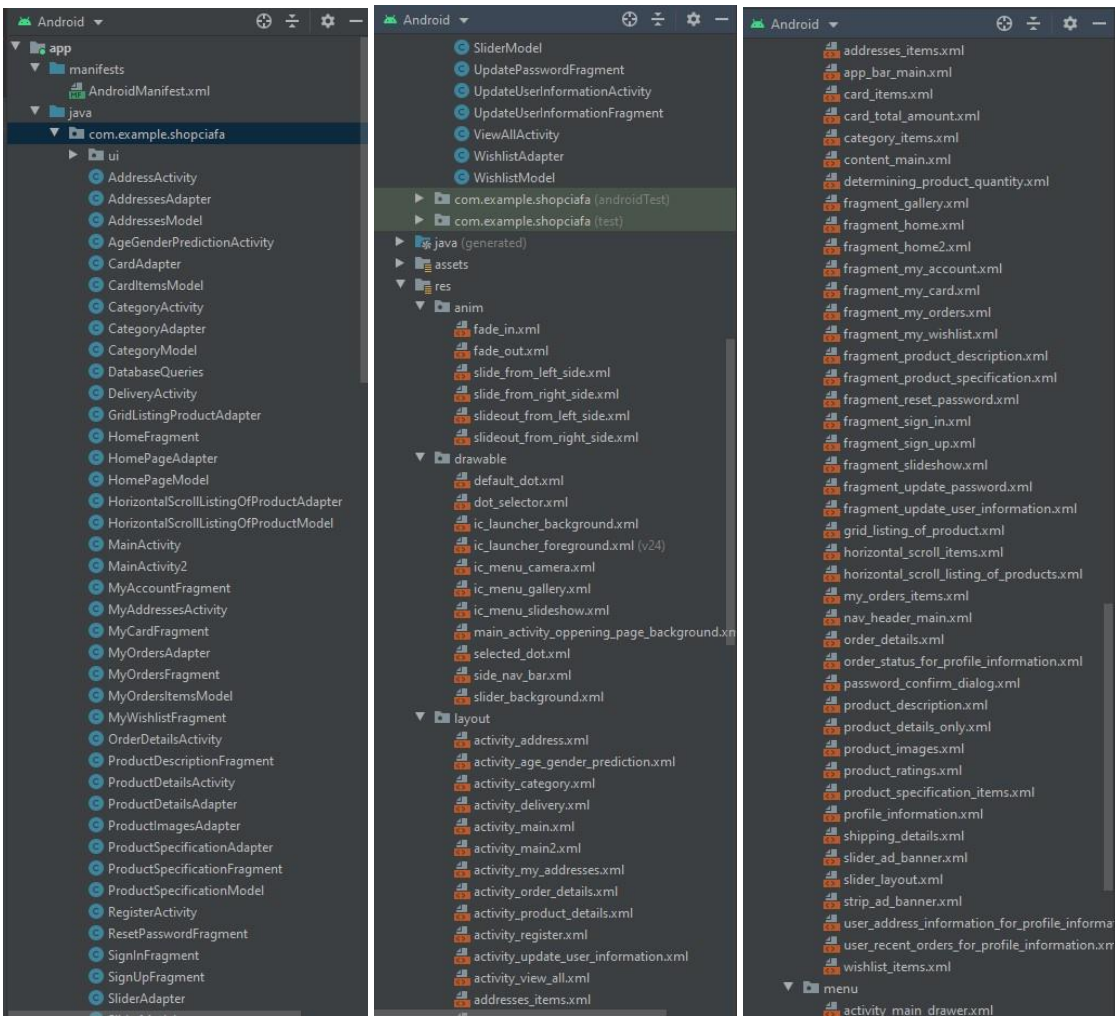


Figure 1.4.1 List of the Project Files

Activity is the page structure in the foreground in the application. When a new project is created in Android Studio, Main Activity will automatically come and be the first working class. Each Activity has an onCreate method and it contains setContentView. setContentView determines which visual structure the activity has. Each Activity class created must have a reference in AndroidManifest.xml.

MAIN ACTIVITY

- **MainActivity.Java**

OnStart method will be called when the event becomes visible to the user. onCreate method is called immediately after the first time the event is started.

When opening the Shopciafa project, the splash screen first appears. Here, the logo of the application remains visible for about 3 seconds. Then it is checked whether the user was logged into the system or not. If there is no user in the system, it is first directed to the register page. If the user is logged into the system, it is directed to the home page (Main Activity2). Transitions between activities are provided with an intent code.

```java
package com.example.shopciafa;

import ...

public class MainActivity extends AppCompatActivity {

    private FirebaseAuth firebaseAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        firebaseAuth = FirebaseAuth.getInstance();

        SystemClock.sleep( ms: 1000);
        Intent register_intent = new Intent(MainActivity.this, RegisterActivity.class);
        startActivity(register_intent);
        finish();
    }

    @Override
    protected void onStart() {
        super.onStart();

        FirebaseUser current_user = firebaseAuth.getCurrentUser();
        if (current_user == null){
            Intent register_intent = new Intent(MainActivity.this, RegisterActivity.class);
            startActivity(register_intent);
            finish();
        }else{
            Intent main_intent = new Intent(MainActivity.this, MainActivity2.class);
            startActivity(main_intent);
            finish();

        }
    }
}
```

If the user is using the application for the first time, user will be directed to the register page. For this, fragment was created, sign in and sign up designs were made. The purpose of using fragments is to create user interfaces without transitioning activity.

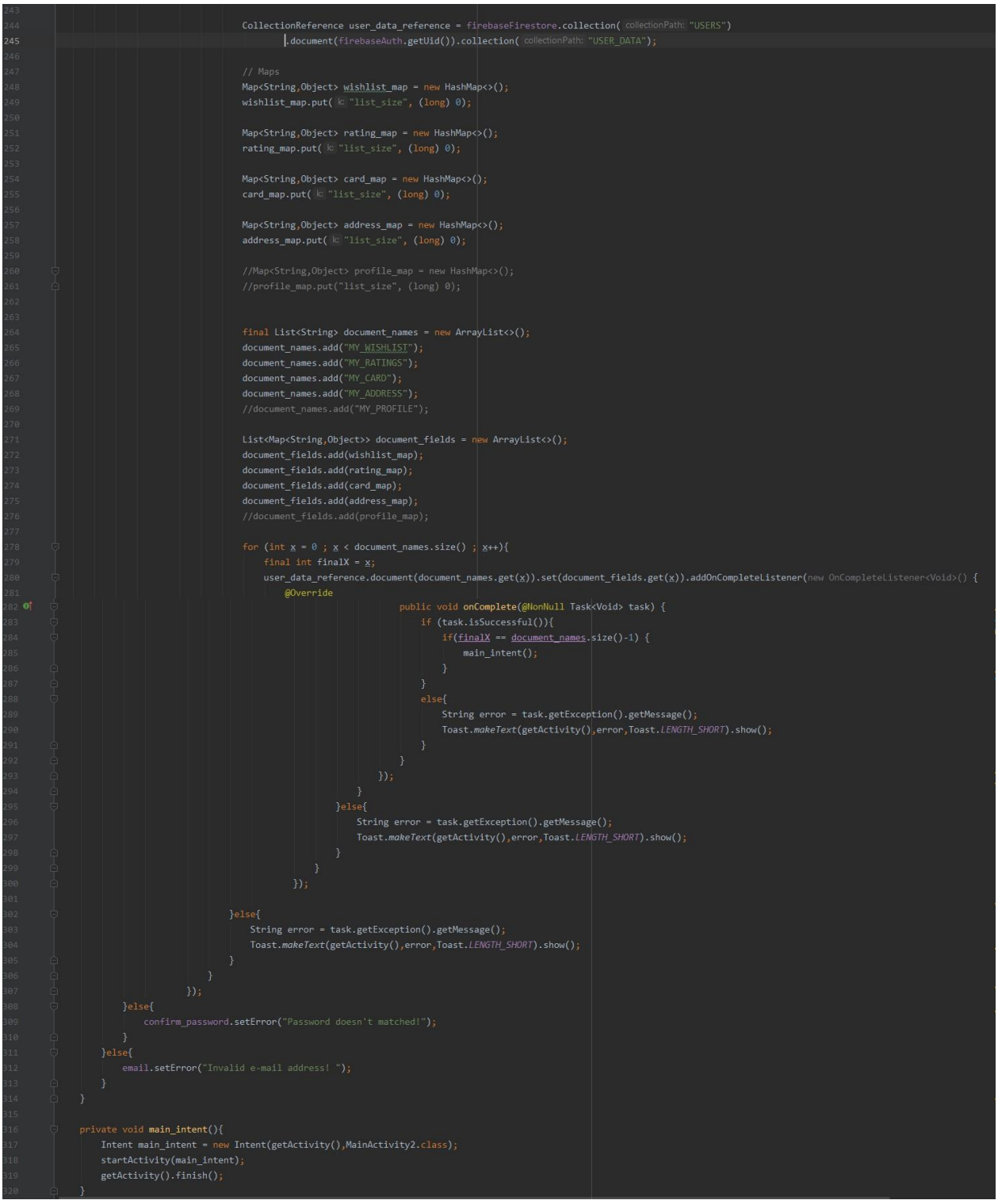
SIGN-UP

- **SignUpFragment.Java**

```java
package com.example.shopciafa;

import ...

/** A simple {@link Fragment} subclass. ...*/
public class SignUpFragment extends Fragment {

    //...
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public SignUpFragment() {
        // Required empty public constructor
    }

    private TextView doYouHaveAnAccount;
    private FrameLayout parentFrameLayout;
    private EditText email;
    private EditText password;
    private EditText confirm_password;

    private ImageButton btn_close;
    private Button btn_sign_up;

    private FirebaseAuth firebaseAuth;
    private FirebaseFirestore firebaseFirestore;

    private String emailPattern = "[a-zA-Z0-9._-]+@[a-z]+.[a-z]+";

    /** Use this factory method to create a new instance of ...*/
    // TODO: Rename and change types and number of parameters
    public static SignUpFragment newInstance(String param1, String param2) {
        SignUpFragment fragment = new SignUpFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
```

```
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_sign_up, container, attachToRoot: false);

        parentFrameLayout = getActivity().findViewById(R.id.register_framelayout);

        doYouHaveAnAccount = view.findViewById(R.id.sign_up_do_you_have_an_account);
        email = view.findViewById(R.id.sign_up_email);
        password = view.findViewById(R.id.sign_up_password);
        confirm_password = view.findViewById(R.id.sign_up_comfirm_password);

        btn_close = view.findViewById(R.id.sign_up_btn_cross);
        btn_sign_up = view.findViewById(R.id.btn_sign_up);

        firebaseAuth = FirebaseAuth.getInstance();
        firebaseFirestore = FirebaseFirestore.getInstance();

        return view;
    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        doYouHaveAnAccount.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { setFragment(new SignInFragment()); }
        });

        btn_close.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { main_intent(); }
        });

        //EditText uses TextWatcher interface to watch change made over EditText. For doing this, EditText calls the addTextChangedListener() method.
        //Tracks changes made to user information
        email.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
                checkInputs();
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });
        password.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
                checkInputs();
            }
```

```
            @Override
            public void afterTextChanged(Editable editable) {

            }
        });
        confirm_password.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
                checkInputs();
            }

            @Override
            public void afterTextChanged(Editable editable) {

            }
        });

        btn_sign_up.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //send users data to firebase
                checkEmailAndPassword();
            }
        });
    }

    private void setFragment(Fragment fragment) {
        FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction();
        //animation running while the user is directed to the sign in page if the user has a registered account
        fragmentTransaction.setCustomAnimations(R.anim.slide_from_left_side,R.anim.slideout_from_right_side);
        fragmentTransaction.replace(parentFrameLayout.getId(),fragment);
        fragmentTransaction.commit();
    }

    private void checkInputs(){
        //controls the user registration information
        if (!TextUtils.isEmpty(email.getText())){
            if (!TextUtils.isEmpty(password.getText()) && password.length() >= 6){
                if(!TextUtils.isEmpty(confirm_password.getText())){
                    btn_sign_up.setEnabled(true);
                }else{
                    btn_sign_up.setEnabled(false);
                }
            }else{
                btn_sign_up.setEnabled(false);
            }
        }else{
            btn_sign_up.setEnabled(false);
        }
    }

    private void checkEmailAndPassword(){
        if(email.getText().toString().matches(EmailPattern)){
            if (password.getText().toString().equals(confirm_password.getText().toString())){

                firebaseAuth.createUserWithEmailAndPassword(email.getText().toString(),password.getText().toString())
                        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult> task) {
                                if (task.isSuccessful()){

                                    Map<String,Object> user_data = new HashMap<>();
                                    user_data.put( k: "email",email.getText().toString());
                                    user_data.put( k: "profile", v: "");

                                    firebaseFirestore.collection( collectionPath: "USERS").document(firebaseAuth.getUid())
                                            .set(user_data)
                                            .addOnCompleteListener(new OnCompleteListener<Void>() {
                                                @Override
                                                public void onComplete(@NonNull Task<Void> task) {
                                                    if (task.isSuccessful()){
```



```
                                CollectionReference user_data_reference = firebaseFirestore.collection( collectionPath: "USERS")
                                        .document(firebaseAuth.getUid()).collection( collectionPath: "USER_DATA");

                                // Maps
                                Map<String,Object> wishlist_map = new HashMap<>();
                                wishlist_map.put( k: "list_size", (long) 0);

                                Map<String,Object> rating_map = new HashMap<>();
                                rating_map.put( k: "list_size", (long) 0);

                                Map<String,Object> card_map = new HashMap<>();
                                card_map.put( k: "list_size", (long) 0);

                                Map<String,Object> address_map = new HashMap<>();
                                address_map.put( k: "list_size", (long) 0);

                                //Map<String,Object> profile_map = new HashMap<>();
                                //profile_map.put("list_size", (long) 0);


                                final List<String> document_names = new ArrayList<>();
                                document_names.add("MY_WISHLIST");
                                document_names.add("MY_RATINGS");
                                document_names.add("MY_CARD");
                                document_names.add("MY_ADDRESS");
                                //document_names.add("MY_PROFILE");

                                List<Map<String,Object>> document_fields = new ArrayList<>();
                                document_fields.add(wishlist_map);
                                document_fields.add(rating_map);
                                document_fields.add(card_map);
                                document_fields.add(address_map);
                                //document_fields.add(profile_map);

                                for (int x = 0 ; x < document_names.size() ; x++){
                                    final int finalX = x;
                                    user_data_reference.document(document_names.get(x)).set(document_fields.get(x)).addOnCompleteListener(new OnCompleteListener<Void>() {
                                        @Override
                                                public void onComplete(@NonNull Task<Void> task) {
                                            if (task.isSuccessful()){
                                                if(finalX == document_names.size()-1) {
                                                    main_intent();
                                                }
                                            }
                                            else{
                                                String error = task.getException().getMessage();
                                                Toast.makeText(getActivity(),error,Toast.LENGTH_SHORT).show();
                                            }
                                        }
                                    });
                                }
                                else{
                                    String error = task.getException().getMessage();
                                    Toast.makeText(getActivity(),error,Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    }else{
                        confirm_password.setError("Password doesn't matched!");
                    }
                }else{
                    email.setError("Invalid e-mail address! ");
                }
            }

            private void main_intent(){
                Intent main_intent = new Intent(getActivity(),MainActivity2.class);
                startActivity(main_intent);
                getActivity().finish();
            }
```

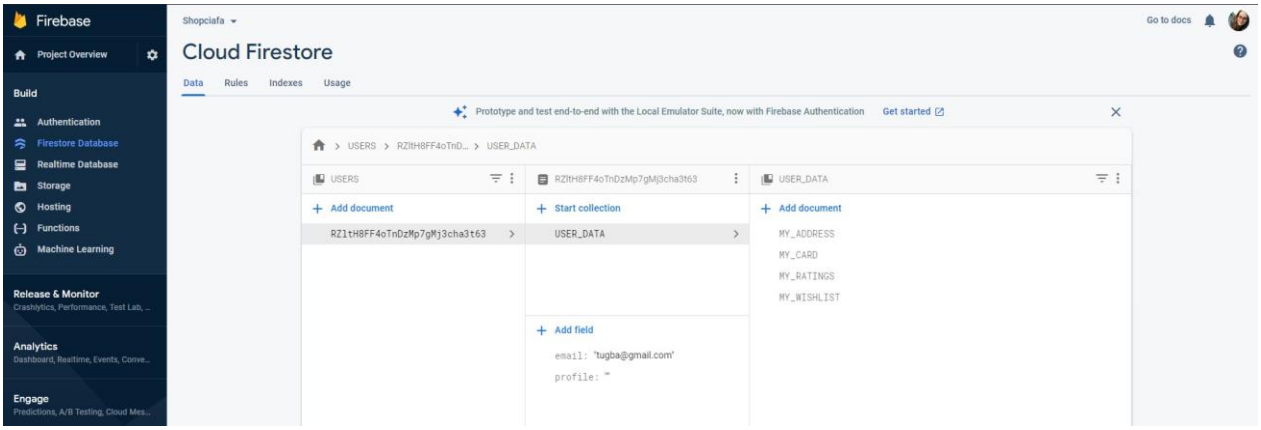
The user must enter his e-mail address and specify a password while registering in the system. EmailPattern has been defined to prevent security vulnerabilities. A valid e-mail address must be entered in accordance with the rules specified in the pattern. Otherwise, the application will fail.

CheckInputs and checkEmailAndPassword methods are written. The data in the fields written in the edit text are checked with the CheckInputs method. If the e-mail address is blank, if the password is empty or

the length is shorter than the specified length and the confirm password field is blank, sign up will not be performed and the user will be warned. The Edittext addTextChangedListener () method comes with three different functions, onTextChanged, beforeTextChanged, afterTextChanged. Using all three methods, the user can do anything when starting to write edittext. Therefore, when writing edittext, the addTextChangedListener () function is used to dynamically retrieve and change the text view value.

The checkEmailAndPassword method checks whether the email matches with the pattern defined above, and checks whether the password entered by the user matches the confirm password. If it matches, database transactions begin to take place.

In this project, Firebase Firestore was used as a database. When the user successfully registers to the application, the following section in the database is created automatically. The user's email address is saved. And a field has been created for the address information to be added during the application, the products it likes, the products to be added to the card, etc. This field is unique for every user because each user has a unique id. FirebaseAuth.getUid () can be used easily according to the user's id.



- **Fragment_sign_up.xml**

In Android applications, screen designs are determined by layout files under the res folder. These files are files prepared in XML format and report the placements and properties of visual items using labels specific to Android applications.

The appearance of a screen is usually determined using two different types of layouts. These are divided into RelativeLayout and LinearLayout. In addition to these, there is ConstraintLayout, which can be used in all APIs of API 9 and above, with its strengths that can easily reveal complex designs and its high performance compared to other layout types, which is used much more actively in a short time. He mostly used ConstraintLayout in his project designs.

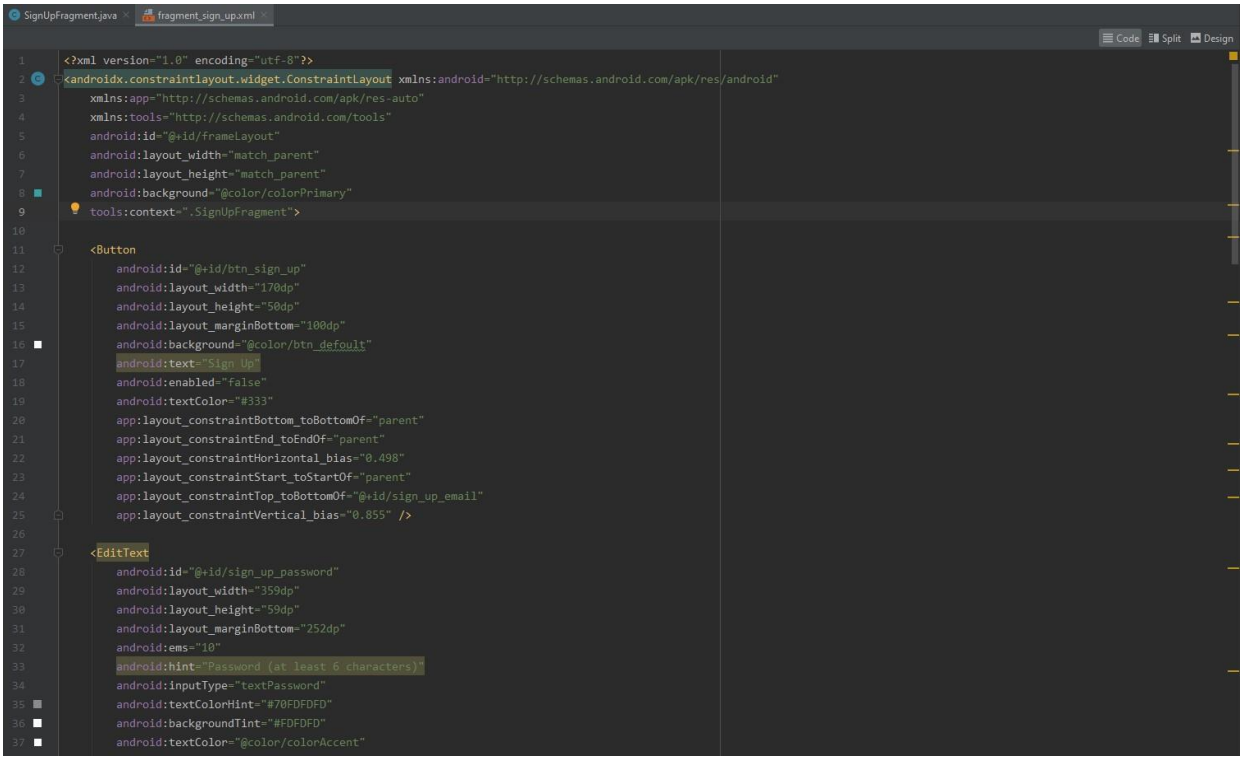
When creating designs, there are some basic terms and the one that best suits the design is preferred. The length and width of the elements are determined according to the android: layout_width and android: layout_height properties.

fill_parent (the larger the main element, the more it fills them) and wrap_content (takes up as much space as text or an image in the element) and length in pixels.

If we are going to determine the size by giving pixels, it will be useful to use DP unit instead of px unit. Since Android devices have screens of different sizes, the design of pixels given as pixels will create different on each device. The DP unit scales according to the screen size and allows you to get similar views on different devices. The dimensions of the elements are determined by the android: layout_height and android: layout_width methods.

The Android: id attribute allows each item to be given an identifier and helps us access and manipulate items in source code. The id values created here are automatically created in the R file and accessed from the source code.

In layout files, @ + id is used to assign an ID to an item. In this way, the specified names can be given to the products. The @id is used to reference a predefined element in the layout file.



```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.508"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/sign_up_email"
        app:layout_constraintVertical_bias="0.166" />

    <EditText
        android:id="@+id/sign_up_comfirm_password"
        android:layout_width="359dp"
        android:layout_height="59dp"
        android:layout_marginBottom="200dp"
        android:ems="10"
        android:hint="Comfirm Password "
        android:inputType="textPassword"
        android:textColorHint="#70FDFDFD"
        android:backgroundTint="#FDFDFD"
        android:textColor="@color/colorAccent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.511"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/sign_up_password" />

    <EditText
        android:id="@+id/sign_up_email"
        android:layout_width="359dp"
        android:layout_height="65dp"
        android:layout_marginStart="10dp"
        android:layout_marginTop="250dp"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="241dp"
        android:ems="10"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:textColorHint="#70FDFDFD"
        android:backgroundTint="#FDFDFD"
        android:textColor="@color/colorAccent"
        app:layout_constraintBottom_toTopOf="@+id/sign_in_dont_have_an_account"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/sign_up_btn_cross"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:layout_marginTop="5dp"
        android:layout_marginEnd="5dp"
        android:background="@android:color/transparent"
        android:padding="16dp"
        android:src="@mipmap/close_cross"
        android:visibility="visible"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/sign_up_img"
        android:layout_width="188dp"
        android:layout_height="0dp"
        android:layout_marginTop="50dp"
        android:layout_marginBottom="50dp"
        android:src="@drawable/main_activity_oppening_page_background"
        app:layout_constraintBottom_toTopOf="@+id/sign_up_email"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.115" />

    <TextView
        android:id="@+id/sign_up_do_you_have_an_account"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="170dp"
        android:text="Do you have an account? Sign in"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/sign_up_email" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

- **slideout_from_left_side.xml**

Another folder named anim was created in the res folder. Anim means animation. Sign in has been written to make the transition between sign up pages in a different way.

slideout_from_left_side.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="250" android:interpolator="@android:anim/accelerate_decelerate_interpolator">

    <translate
        android:fromXDelta="0%"
        android:toXDelta="-100%"
        android:fromYDelta="0%"
        android:toYDelta="0%"/>
</set>
```

MAIN ACTIVITY2

- **activity_main_drawer.xml**

There is a folder named menu under the res file where the designs of the homepage are arranged. For example, while editing the fields to appear in the side bar in activity_main_drawer.xml, the field where the user's picture and mail address will appear in nav_header_main.xml has been arranged.

activity_main_drawer.xml | main_activity2.xml

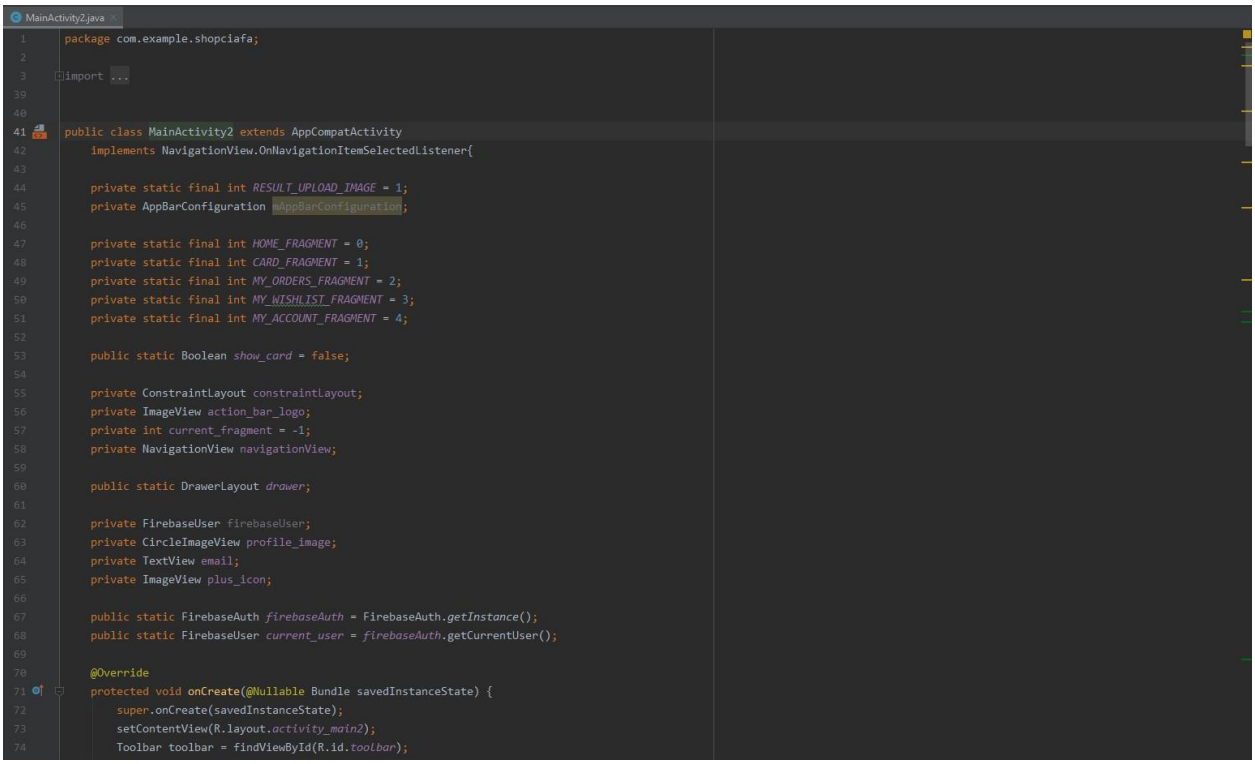
```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/main2_shopciafa"
            android:icon="@mipmap/shopciafa_icon"
            android:title="Shopciafa" />
        <item
            android:id="@+id/main2_my_orders"
            android:icon="@mipmap/checklist"
            android:title="My Orders" />
        <item
            android:id="@+id/main2_my_card"
            android:icon="@mipmap/card"
            android:title="My Card" />
        <item
            android:id="@+id/main2_my_wishlist"
            android:icon="@mipmap/wishlist"
            android:title="My Wishlist" />
        <item
            android:id="@+id/main2_my_account"
            android:icon="@mipmap/user"
            android:title="My Account" />
    </group>

    <item
        android:id="@+id/main2_sign_out"
        android:icon="@mipmap/logout"
        android:title="Sign Out"
    />

</menu>
```

Generally, the empty activity was chosen while creating an activity, but the navigation drawer activity was selected here.

- **MainActivity2.Java**

The menu and the operation of the menu are arranged in the Main activity2. The Android Navigation Drawer is a left-scrolling menu used to view important links in the app. The navigation drawer makes it easy to navigate between these links. It is invisible by default and needs to be opened by swiping left or clicking its icon in the ActionBar.

ActionBarDrawerToggle sets the app icon located to the left of the action bar or toolbar to open and close the Android navigation drawer. The addDrawerListener () method is called on a DrawerLayout to connect an ActionBarDrawerToggle with a DrawerLayout.



```
        action_bar_logo = findViewById(R.id.action_bar_logo);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayShowTitleEnabled(false);

        drawer = findViewById(R.id.drawer_layout);

        navigationView = (NavigationView) findViewById(R.id.nav_view);;
        navigationView.setNavigationItemSelectedListener(this);
        navigationView.bringToFront();
        navigationView.getMenu().getItem( i: 0).setChecked(true);

        // Passing each menu ID as a set of Ids because each menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow)
                .setDrawerLayout(drawer)
                .build();

        constraintLayout = findViewById(R.id.constraintlayout_content_main);

        profile_image = navigationView.getHeaderView( index: 0).findViewById(R.id.main2_user_profile_picture);
        email = navigationView.getHeaderView( index: 0).findViewById(R.id.main2_user_email_address);
        plus_icon = navigationView.getHeaderView( index: 0).findViewById(R.id.main2_add_profile_photo_btn);

        plus_icon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) { chooseImageFromGallery(); }
        });

        if (show_card) {
            drawer.setDrawerLockMode(DrawerLayout.LOCK_MODE_LOCKED_CLOSED);
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            gotoFragment( fragment_title: "My Card", new MyCardFragment(),  fragment_no: -2);
        } else {
            ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawer, toolbar,
                    "Open navigation drawer", "Close navigation drawer");
            drawer.addDrawerListener(toggle);
            toggle.syncState();
            setFragment(new HomeFragment(), HOME_FRAGMENT);
        }

    }

    @Override
    protected void onStart() {
        super.onStart();
        current_user = FirebaseAuth.getInstance().getCurrentUser();
        if(current_user == null){
            navigationView.getMenu().getItem( i: navigationView.getMenu().size()-1).setEnabled(false);
        }
        else{
            FirebaseFirestore.getInstance().collection( collectionPath: "USERS").document(current_user.getUid()).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    if (task.isSuccessful()){
                        DatabaseQueries.email = task.getResult().getString( field: "email");
                        DatabaseQueries.profile = task.getResult().getString( field: "profile");

                        email.setText(DatabaseQueries.email);
                        if (DatabaseQueries.profile.equals("")){
                            plus_icon.setVisibility(View.INVISIBLE);
                        }else {
                            plus_icon.setVisibility(View.INVISIBLE);
                            Glide.with( activity: MainActivity2.this).load(DatabaseQueries.profile).apply(new RequestOptions().placeholder(R.mipmap.user_dark2)).into(profile_image);
                        }
                    }else{
                        String error_message = task.getException().getMessage();
                        Toast.makeText( context: MainActivity2.this,error_message,Toast.LENGTH_SHORT).show();
                    }
                }
            });

            navigationView.getMenu().getItem( i: navigationView.getMenu().size()-1).setEnabled(true);
        }
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
```

The Android options menu is organized using getMenuInflater (). inflate, onCreateOptionsMenu and onOptionsItemSelected. Overrides the onCreateOptionsMenu using getMenuInflater ().inflate to create a menu inflating a menu hierarchy from XML source. The onOptionsItemSelected element is overridden in the Activity class to handle the Click event.

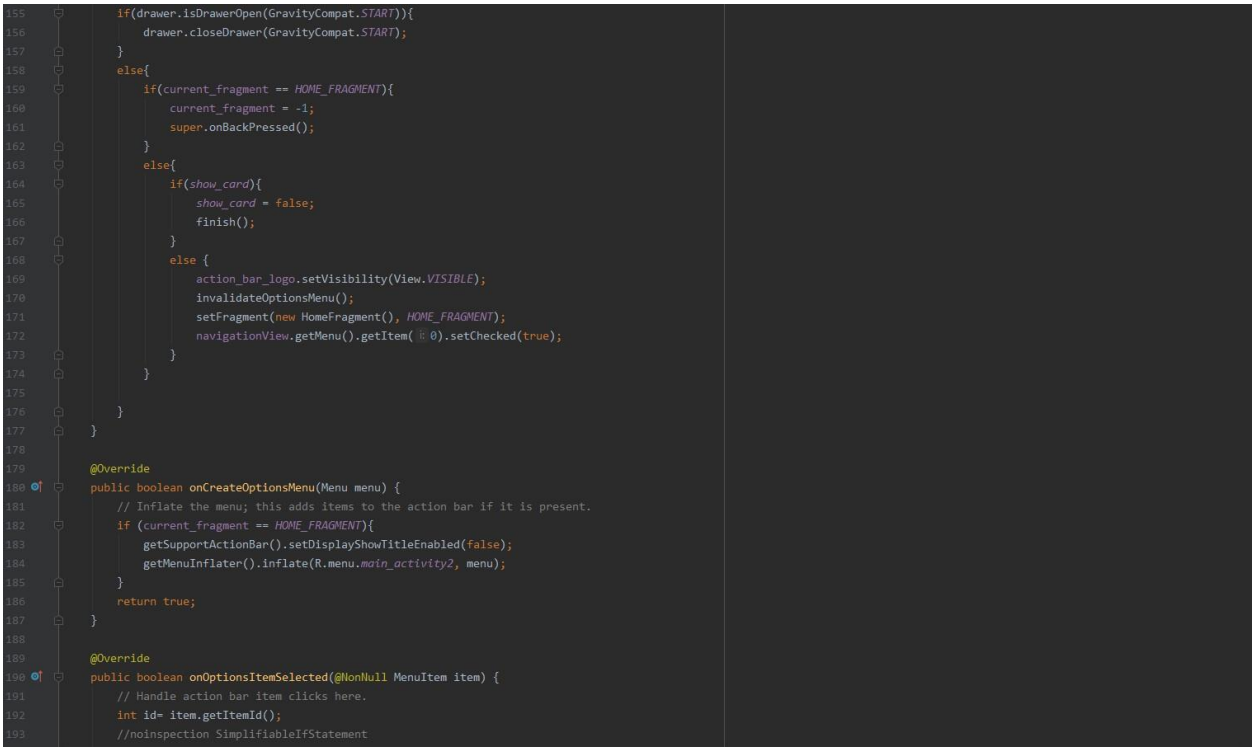
The code for the onCreateOptionsMenu (Menu menu) method has been written, which should be overridden in the Activity class. This creates the menu and returns the Boolean value. inflate inflates a menu hierarchy from XML source.

To create the click event on the menu item, the onOptionsItemSelected (MenuItem item) is applied in the Activity class and the Boolean value is returned.

The fragment clicked on is specified in the gotoFragment method is determined. When the user clicks on whichever fragment, a background color is assigned to indicate which fragment it is in while directing to that page, and the color of the selected fragment text changes.

Clicking on any item/fragment, the user must be directed to a new Activity or Fragment and this is called the navigation drawer. By implementing this convention or interface, the only method should now be overridden onNavigationItemSelected (). This method is invoked when an item in the navigation menu is selected. Redirection is made to the relevant fragment according to the id.

In cases where the drawer menu design is open, the onBackPressed method is used to provide back closure.



```
        if(drawer.isDrawerOpen(GravityCompat.START)){
            drawer.closeDrawer(GravityCompat.START);
        }
        else{
            if(current_fragment == HOME_FRAGMENT){
                current_fragment = -1;
                super.onBackPressed();
            }
            else{
                if(show_card){
                    show_card = false;
                    finish();
                }
                else {
                    action_bar_logo.setVisibility(View.VISIBLE);
                    invalidateOptionsMenu();
                    setFragment(new HomeFragment(), HOME_FRAGMENT);
                    navigationView.getMenu().getItem( i: 0).setChecked(true);
                }
            }

        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        if (current_fragment == HOME_FRAGMENT){
            getSupportActionBar().setDisplayShowTitleEnabled(false);
            getMenuInflater().inflate(R.menu.main_activity2, menu);
        }
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        // Handle action bar item clicks here.
        int id= item.getItemId();
        //noinspection SimplifiableIfStatement
```

```
        if(id==R.id.main2_search_icon){
            //search area
            return true;
        }else if(id==R.id.main2_notification_icon){
            //notification area
            return true;
        }else if(id==R.id.main2_basket_icon){
            //basket - card area
            gotoFragment( fragment_title: "My Card",new MyCardFragment(),CARD_FRAGMENT);

            /*
            * The main purpose here is to direct the payment-related card fragmenta when the basket in Product Details is clicked.
             * But gotoFragment ("My Card", new MyCardFragment (), CARD_FRAGMENT) on line 134; it only redirects to the basket button on the home page.
             * How can we control them all in one place.

            Intent card_intent = new Intent(this,ProductDetailsActivity.class);
            show_card = true;
            startActivity(card_intent);
             */
            return true;
        }else if(id == android.R.id.home){
            if (show_card){
                show_card = false;
                finish();
                return true;
            }
        }
        return super.onOptionsItemSelected(item);
    }

    private void gotoFragment(String fragment_title, Fragment fragment, int fragment_no) {
        action_bar_logo.setVisibility(View.GONE);
        getSupportActionBar().setDisplayShowTitleEnabled(true);
        getSupportActionBar().setTitle(fragment_title);
        invalidateOptionsMenu();
        setFragment(fragment,fragment_no);
        if(fragment_no == MY_ORDERS_FRAGMENT){
            navigationView.getMenu().getItem( i: 1).setChecked(true);
        }
         else if(fragment_no == CARD_FRAGMENT){
             //In the side menu; 3 elements of array (with index 2)
             navigationView.getMenu().getItem( i: 2).setChecked(true);
        }else if(fragment_no == MY_WISHLIST_FRAGMENT){
             //In the side menu; 4 elements of array (with index 3)
             navigationView.getMenu().getItem( i: 3).setChecked(true);
        }else if(fragment_no == MY_ACCOUNT_FRAGMENT){
             //In the side menu; 5 elements of array (with index 4)
             navigationView.getMenu().getItem( i: 4).setChecked(true);
        }else{
             //In the side menu; 5 elements of array (with index 5)
             navigationView.getMenu().getItem( i: 5).setChecked(true);
        }
    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        if(current_user != null) {
            //Handle navigation view item clicks here
            int id = item.getItemId();
            if (id == R.id.main2_shopciafa) {
                action_bar_logo.setVisibility(View.VISIBLE);
                invalidateOptionsMenu();
                setFragment(new HomeFragment(), HOME_FRAGMENT);
            } else if (id == R.id.main2_my_orders) {
                gotoFragment( fragment_title: "My Orders", new MyOrdersFragment(), MY_ORDERS_FRAGMENT);
            } else if (id == R.id.main2_my_card) {
                gotoFragment( fragment_title: "My Card", new MyCardFragment(), CARD_FRAGMENT);
            } else if (id == R.id.main2_my_wishlist) {
                gotoFragment( fragment_title: "My Wishlist", new MyWishlistFragment(), MY_WISHLIST_FRAGMENT);
            } else if (id == R.id.main2_my_account) {
                gotoFragment( fragment_title: "My Account", new MyAccountFragment(), MY_ACCOUNT_FRAGMENT);
            } else if (id == R.id.main2_sign_out) {
                FirebaseAuth.getInstance().signOut();
                DatabaseQueries.clearData();
                Intent register_intent = new Intent(MainActivity2.this,RegisterActivity.class);
                startActivity(register_intent);
                finish();
```

```
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }

    private void setFragment(Fragment fragment, int fragment_no){
        if(fragment_no != current_fragment){
            current_fragment = fragment_no;
            FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();
            fragmentTransaction.setCustomAnimations(R.anim.fade_in,R.anim.fade_out);
            fragmentTransaction.replace(constraintLayout.getId(),fragment);
            fragmentTransaction.commit();
        }
    }

    private void chooseImageFromGallery() {
        Intent galleryIntent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.INTERNAL_CONTENT_URI);
        startActivityForResult(galleryIntent, RESULT_UPLOAD_IMAGE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == RESULT_UPLOAD_IMAGE && resultCode == RESULT_OK && data != null){
            Uri select_Image = data.getData();
            profile_image.setImageURI(select_Image);
        }
    }
}
```

HOME PAGE

- **HomeFragment.Java**

The products in the application in HomeFragment.Java are listed on the homepage. When the user first registers to the system, since there is no picture loaded in the system, user will see the start page where men and women products are mixed. However, if the user uploads a picture to the system, the age and gender of the person are estimated from the picture. The estimated gender is kept in genderPredict. If the value of genderPredict is Male, male products are listed, and female and women's products are listed. The categories in the application are also listed on this page. Categories and products are pulled from the database. The application works when there is an internet connection. If there is an internet connection, the user can view the products in the system. Otherwise, a warning message is displayed saying please check the internet connection.

HomeFragment.java

```
package com.example.shopciafa;

import ...

/** A simple {@link Fragment} subclass. ...*/
public class HomeFragment extends Fragment {

    //...
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public HomeFragment() {
        // Required empty public constructor
    }

    /** Use this factory method to create a new instance of ...*/
    // TODO: Rename and change types and number of parameters
    public static HomeFragment newInstance(String param1, String param2) {
        HomeFragment fragment = new HomeFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    private RecyclerView categoryRecyclerView;
    private CategoryAdapter categoryAdapter;
    private RecyclerView home_page_recyclerView;
    private HomePageAdapter home_page_adapter;
    private ImageView no_internet_connection;
    private TextView check_internet_connection;
```

```java
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    //Layoutinflater takes XML file as input and creates View objects.
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // The inflate () method returns a view back. The first parameter of this method is the layout we want to be converted to java,
        // the second parameter is the ID of the layout element that we want to add this layout to, and the third parameter is the second parameter
        // and whether it will be added to the view, the parent of the inflated view, as true or false.
        View view = inflater.inflate(R.layout.fragment_home2, container,  attachToRoot: false);
        no_internet_connection = view.findViewById(R.id.no_internet_connection);
        check_internet_connection = view.findViewById(R.id.check_internet_connection);

        //internet connection check
        ConnectivityManager connectivityManager = (ConnectivityManager) getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connectivityManager.getActiveNetworkInfo();

        if (networkInfo != null && networkInfo.isConnected() == true) {
            //if there is internet connection
            //MainActivity2.drawer.setDrawerLockMode(0);
            no_internet_connection.setVisibility(View.GONE);
            check_internet_connection.setVisibility(View.GONE);

            categoryRecyclerView = view.findViewById(R.id.category_recyclerView);
            LinearLayoutManager layoutManager = new LinearLayoutManager(getActivity());
            layoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
            categoryRecyclerView.setLayoutManager(layoutManager);

            categoryAdapter = new CategoryAdapter(categoryModellist);

            if (categoryModellist.size() == 0) {
                loadCategories(categoryAdapter, getContext());
            } else {
                //MainActivity2.drawer.setDrawerLockMode(1);
                categoryAdapter = new CategoryAdapter(categoryModellist);
                categoryAdapter.notifyDataSetChanged();
            }
            categoryRecyclerView.setAdapter(categoryAdapter);

            home_page_recyclerView = view.findViewById(R.id.home_page_recyclerView);
            LinearLayoutManager testingLayoutManager = new LinearLayoutManager(getContext());
            testingLayoutManager.setOrientation(LinearLayoutManager.VERTICAL);
            home_page_recyclerView.setLayoutManager(testingLayoutManager);


            // LISTING PRODUCTS BY GENDER ON THE HOME PAGE
            if (genderPredict != null && genderPredict.getText().toString() != "" && genderPredict.getText().toString().contains("Female")) {
                // Listing products by female
                if (female_lists.size() == 0) {
                    loadedTheCategoryName.add("HOME");
                    female_lists.add(new ArrayList<HomePageModel>());
                    home_page_adapter = new HomePageAdapter(female_lists.get(0));
                    loadFemaleFragmentData(home_page_adapter, getContext(),  index: 0,  categoryName: "Home");
                    DatabaseQueries.pageIndex = 0;
                } else {
                    home_page_adapter = new HomePageAdapter(female_lists.get(0));
                    home_page_adapter.notifyDataSetChanged();
                }
                // Listing products by male
            } else if (genderPredict != null && genderPredict.getText().toString() != "" && genderPredict.getText().toString().contains("Male")) {
                if (male_lists.size() == 0) {
                    loadedTheCategoryName.add("HOME");
                    male_lists.add(new ArrayList<HomePageModel>());
                    home_page_adapter = new HomePageAdapter(male_lists.get(0));
                    loadMaleFragmentData(home_page_adapter, getContext(),  index: 0,  categoryName: "Home");
                    DatabaseQueries.pageIndex = 0;
                } else {
                    home_page_adapter = new HomePageAdapter(male_lists.get(0));
                    home_page_adapter.notifyDataSetChanged();
                }
            } else {
```

```
                // Mixed products seen when users do not upload pictures to the application
                if (lists.size() == 0) {
                    //When you click the home icon, the name of the relevant page will appear on the application. This process will be valid for all categories.
                    loadedTheCategoryName.add("HOME");
                    lists.add(new ArrayList<HomePageModel>());
                    home_page_adapter = new HomePageAdapter(lists.get(0));
                    loadFragmentData(home_page_adapter, getContext(), index: 0, categoryName: "Home");
                    DatabaseQueries.pageIndex = 0;
                } else {
                    home_page_adapter = new HomePageAdapter(lists.get(0));
                    home_page_adapter.notifyDataSetChanged();
                }
            }
            home_page_recyclerView.setAdapter(home_page_adapter);
        } else {
            // if there is not internet connection
            Glide.with( fragment: this).load(R.mipmap.no_internet_connection).into(no_internet_connection);
            no_internet_connection.setVisibility(View.VISIBLE);
            check_internet_connection.setVisibility(View.VISIBLE);
        }
        return view;
    }

}
```

- **HomePageAdapter.Java**

Firstly, bind our data to the AdapterView using an Adapter. The GetViewTypeCount () and getItemViewType () methods are overridden. getItemViewType () returns the view that should be loaded based on position, so here it is used to define the views that should be returned. By default, 0 is returned, so view types start enumerating from 0. For example, the slider is defined as 0.

In OnCreateViewHolder (), the viewType returned by getItemViewType () is determined and the viewHolder instance is created.

Finally, in onBindViewHolder, the viewType returned by getItemViewType () is determined and the view is binded with data using viewHolder instances.

Here, the interfaces that will be used to list the products on the homepage are arranged. These are slider, strip ad banner, horizontal and grid part. Later, relevant codes were added for the operation of these 4 modules.

```
@Override
public int getItemViewType(int position) {
    switch (homePageModelList.get(position).getType()){
        case 0:
            return HomePageModel.BANNER_SLIDER;
        case 1:
            return HomePageModel.STRIP_AD_BANNER;
        case 2:
            return HomePageModel.LISTING_PRODUCT_HORIZOTALLY;
        case 3:
            return HomePageModel.GRID_LISTING_PRODUCT;
        default:
            return -1;
    }
}

@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {

    switch (viewType){
        case HomePageModel.BANNER_SLIDER:
            View slider_banner_view = LayoutInflater.from(parent.getContext()).inflate(R.layout.slider_ad_banner,parent, attachToRoot: false);
            return new BannerSliderViewholder(slider_banner_view);
        case HomePageModel.STRIP_AD_BANNER:
            View strip_ad_banner_view = LayoutInflater.from(parent.getContext()).inflate(R.layout.strip_ad_banner,parent, attachToRoot: false);
            return new StripAdBannerViewholder(strip_ad_banner_view);
        case HomePageModel.LISTING_PRODUCT_HORIZOTALLY:
            View listing_product_horizontally = LayoutInflater.from(parent.getContext()).inflate(R.layout.horizontal_scroll_listing_of_products,parent, attachToRoot: false);
            return new ListingProductHorizontallyViewholder(listing_product_horizontally);
        case HomePageModel.GRID_LISTING_PRODUCT:
            View grid_listing_product = LayoutInflater.from(parent.getContext()).inflate(R.layout.grid_listing_of_product,parent, attachToRoot: false);
            return new GridListingProductViewHolder(grid_listing_product);
        default:
            return null;
    }
}
```

```java
@Override
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position) {
    switch (homePageModelList.get(position).getType()){
        case HomePageModel.BANNER_SLIDER:
            List<SliderModel> sliderModelList = homePageModelList.get(position).getSliderModelList();
            ((BannerSliderViewholder)holder).setBannerSliderViewPaper2(sliderModelList);
            break;

        case HomePageModel.STRIP_AD_BANNER:
            String resource = homePageModelList.get(position).getResource();
            String color = homePageModelList.get(position).getBackgroundColor();
            ((StripAdBannerViewholder)holder).setStripAd(resource,color);
            break;

        case HomePageModel.LISTING_PRODUCT_HORIZOTALLY:
            String layout_color = homePageModelList.get(position).getBackgroundColor();
            String listing_horizontal_product_title = homePageModelList.get(position).getTitle();
            List<WishlistModel> viewAllProductList = homePageModelList.get(position).getViewAllProductList();
            List<HorizontalScrollListingOfProductModel> horizontalScrollListingOfProductModelList = homePageModelList.get(position).getHorizontalScrollListingOfProductModelList();
            ((ListingProductHorizontallyViewholder)holder).setListingProductHorizontally(horizontalScrollListingOfProductModelList,listing_horizontal_product_title,layout_color,viewAllProductList);
            break;

        case HomePageModel.GRID_LISTING_PRODUCT:
            String grid_layout_color = homePageModelList.get(position).getBackgroundColor();
            String listing_grid_product_title = homePageModelList.get(position).getTitle();
            List<HorizontalScrollListingOfProductModel> gridListingProductModelList = homePageModelList.get(position).getHorizontalScrollListingOfProductModelList();
            ((GridListingProductViewHolder)holder).setGridListingProduct(gridListingProductModelList,listing_grid_product_title,grid_layout_color);
            break;

        default:
            return;
    }
}
```

Below is the module code, which has a hozizontalView view. The data are pulled from the database. If there are less than 10 products in this field, the view all button in the design will not appear in the application. If there are more than 10 products, then the button will be visible and while only the first 10 products are displayed on the homepage, other products will be displayed to the user when the button is clicked. This control is done in the code here.

```java
public class ListingProductHorizontallyViewholder extends RecyclerView.ViewHolder{

    private ConstraintLayout container;
    private TextView horizontal_title;
    private Button btn_horizontal_view_all;
    private RecyclerView horizontal_recyclerView;

    public ListingProductHorizontallyViewholder(@NonNull View itemView) {
        super(itemView);
        container = itemView.findViewById(R.id.container_horizontal_scroll_listing_of_products);
        horizontal_title = itemView.findViewById(R.id.horizontal_scroll_title);
        btn_horizontal_view_all = itemView.findViewById(R.id.btn_horizontal_scroll);
        horizontal_recyclerView = itemView.findViewById(R.id.horizontal_scroll_recyclerView);
        horizontal_recyclerView.setRecycledViewPool(recycledViewPool);
    }
    private void setListingProductHorizontally(List<HorizontalScrollListingOfProductModel> horizontalScrollListingOfProductModelList, final String title, String color, final List<WishlistModel> vi
        container.setBackgroundTintList(ColorStateList.valueOf(Color.parseColor(color)));
        horizontal_title.setText(title);
        if (horizontalScrollListingOfProductModelList.size() > 10){
            btn_horizontal_view_all.setVisibility(View.VISIBLE);
            btn_horizontal_view_all.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    ViewAllActivity.wishlistModelList = viewAllProductList;
                    Intent view_all_intent = new Intent(itemView.getContext(),ViewAllActivity.class);
                    view_all_intent.putExtra( name: "layout_code", value: 0);
                    view_all_intent.putExtra( name: "title",title);
                    itemView.getContext().startActivity(view_all_intent);
                }
            });
        }else{
            btn_horizontal_view_all.setVisibility(View.INVISIBLE);
        }
        HorizontalScrollListingOfProductAdapter horizontalScrollListingOfProductAdapter = new HorizontalScrollListingOfProductAdapter(horizontalScrollListingOfProductModelList);
        LinearLayoutManager linearLayoutManager = new LinearLayoutManager(itemView.getContext());
        linearLayoutManager.setOrientation(LinearLayoutManager.HORIZONTAL);
        horizontal_recyclerView.setLayoutManager(linearLayoutManager);
        horizontal_recyclerView.setAdapter(horizontalScrollListingOfProductAdapter);
        horizontalScrollListingOfProductAdapter.notifyDataSetChanged();
```

DATABASE QUERIES

- **Loading Categories**

In the code given below, the code in which the names and icons of the categories shown on the home page are loaded is shown.

```
public static void loadCategories(final CategoryAdapter categoryAdapter, final Context context){
    categoryModelList.clear();
    firebaseFirestore.collection( collectionPath: "CATEGORIES").orderBy("index").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()){
                // Contains data read from a document in your Firestore database.
                for (QueryDocumentSnapshot documentSnapshot : task.getResult()){
                    categoryModelList.add(new CategoryModel(documentSnapshot.get("icon").toString(),documentSnapshot.get("category_name").toString()));
                }
                categoryAdapter.notifyDataSetChanged();
            }
            else {
                String error_message = task.getException().getMessage();
                Toast.makeText(context,error_message,Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

- **Loading Modules**

The codes that allow the modules to be loaded on the main page have been written. The first module is a slider and its value is 0. An arraylist with its own data is defined for each module. Then a for loop is written to show how many products there are, all of them.

long no_of_banners = Long.parseLong (String.valueOf (documentSnapshot.get ("no_of_banners"))); With this, the information about how many banner images there are for the slider in the database is obtained.

The second module name is banner design and corresponds to type 1. This design is a module to be used when advertising is desired in practice. It is not yet used in the application.

```
public static void loadFragmentData(final HomePageAdapter homePageAdapter, final Context context, final int index, String categoryName) {
    // There is an i letter in categories such as fashion, appliances, cosmetics. When the ToUpperCase method is applied, it occurs and this causes an error in the program.
    // Here the letter İ has been changed to I.
    String target = categoryName.toUpperCase();
    Character harf = 'İ';
    if (target.contains("İ")) {
        target = target.replace(harf, newChar: 'I');
    }

    firebaseFirestore.collection( collectionPath: "CATEGORIES").document(target).collection( collectionPath: "TOP_DEALS").orderBy("index").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {

            if (task.isSuccessful()) {

                lists.get(index).clear();
                for (QueryDocumentSnapshot documentSnapshot : task.getResult()) {
                    // type 0 is slider banners

                    if (Long.parseLong(String.valueOf(documentSnapshot.get("view_type"))) == 0) {
                        List<SliderModel> sliderModelList = new ArrayList<>();
                        long no_of_banners = Long.parseLong(String.valueOf(documentSnapshot.get("no_of_banners")));
                        for (long x = 1; x < no_of_banners; x++) {
                            sliderModelList.add(new SliderModel(documentSnapshot.get("banner_img_" + x).toString()
                                    , documentSnapshot.get("banner_img_" + x + "_background").toString()));
                        }

                        lists.get(index).add(new HomePageModel( type: 0, sliderModelList));

                    }

                    // type 1 is ad banner. (It will be used if you want to advertise in the application.)
                    else if (Long.parseLong(String.valueOf(documentSnapshot.get("view_type"))) == 1) {
                        //lists.get(index).add(new HomePageModel(1,documentSnapshot.get("strip_ad_banner").toString(),documentSnapshot.get("strip_ad_banner_background").toString()));
                    }
```

Type 2 corresponds to the third module horizontal view. Here, products as much as the number of products in the database will appear in the application. However, there is a separate code (horizontalScrollListingOfProductModelList) for the part that appears on the homepage, and a code block with more detailed information (viewAllProducts) that will appear when you want to get detailed information on the product.

```
// type 2 is listing deals of the day! (horizontal view)
else if (Long.parseLong(String.valueOf(documentSnapshot.get("view_type"))) == 2) {
    List<WishlistModel> viewAllProducts = new ArrayList<>();
    List<HorizontalScrollListingOfProductModel> horizontalScrollListingOfProductModelList = new ArrayList<>();
    long no_of_products = Long.parseLong(String.valueOf(documentSnapshot.get("no_of_products")));
    for (long x = 1; x < no_of_products; x++) {
        horizontalScrollListingOfProductModelList.add(new HorizontalScrollListingOfProductModel(documentSnapshot.get("product_ID_" + x).toString()
                , documentSnapshot.get("product_image_" + x).toString()
                , documentSnapshot.get("product_title_" + x).toString()
                , documentSnapshot.get("product_subtitle_" + x).toString()
                , documentSnapshot.get("product_price_" + x).toString()));

        viewAllProducts.add(new WishlistModel(documentSnapshot.get("product_ID_" + x).toString()
                , documentSnapshot.get("product_image_" + x).toString()
                , documentSnapshot.get("product_full_title_" + x).toString()
                , documentSnapshot.get("average_rating_" + x).toString()
                , (long) documentSnapshot.get("total_ratings_" + x)
                , documentSnapshot.get("product_price_" + x).toString()
                , documentSnapshot.get("product_no_discount_price_" + x).toString()
                , (boolean) documentSnapshot.get("COD_" + x)));
    }
    lists.get(index).add(new HomePageModel( type: 2, documentSnapshot.get("layout_title").toString(), documentSnapshot.get("layout_background").toString(), horizontalScrollListingOfProductModelList,
}
```

The last module corresponds to type 3. Here, the data that will appear in a grid view design is pulled from the database. The product id number, picture, name, short subtitle description and price information were taken from the database.

```
                // type 3 is listing new season products. (grid view)
                else if (Long.parseLong(String.valueOf(documentSnapshot.get("view_type"))) == 3) {
                    List<HorizontalScrollListingOfProductModel> GridLayoutModelList = new ArrayList<>();
                    long no_of_products = Long.parseLong(String.valueOf(documentSnapshot.get("no_of_products")));
                    for (long x = 1; x < no_of_products; x++) {
                        GridLayoutModelList.add(new HorizontalScrollListingOfProductModel(documentSnapshot.get("product_ID_" + x).toString()
                                , documentSnapshot.get("product_image_" + x).toString()
                                , documentSnapshot.get("product_title_" + x).toString()
                                , documentSnapshot.get("product_subtitle_" + x).toString()
                                , documentSnapshot.get("product_price_" + x).toString()));
                    }
                    lists.get(index).add(new HomePageModel( type: 3, documentSnapshot.get("layout_title").toString(), documentSnapshot.get("layout_background").toString(), GridLayoutModelList));
                }
            }
            homePageAdapter.notifyDataSetChanged();

        } else {
            String error = task.getException().getMessage();
            Toast.makeText(context, error, Toast.LENGTH_SHORT).show();
        }
    }
});
}
```

When users want to list the products according to their gender by uploading photos to the system, a small change has been made in the database query.

The following query is written for all users.

```
firebaseFirestore.collection( collectionPath: "CATEGORIES").document(target).collection( collectionPath: "TOP_DEALS").orderBy("index").get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
```

Only women's products are listed in this query.

```
firebaseFirestore.collection( collectionPath: "CATEGORIES").document( documentPath: "HOME").collection( collectionPath: "TOP_DEALS").whereEqualTo( field: "gender",  value: 1).get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
```

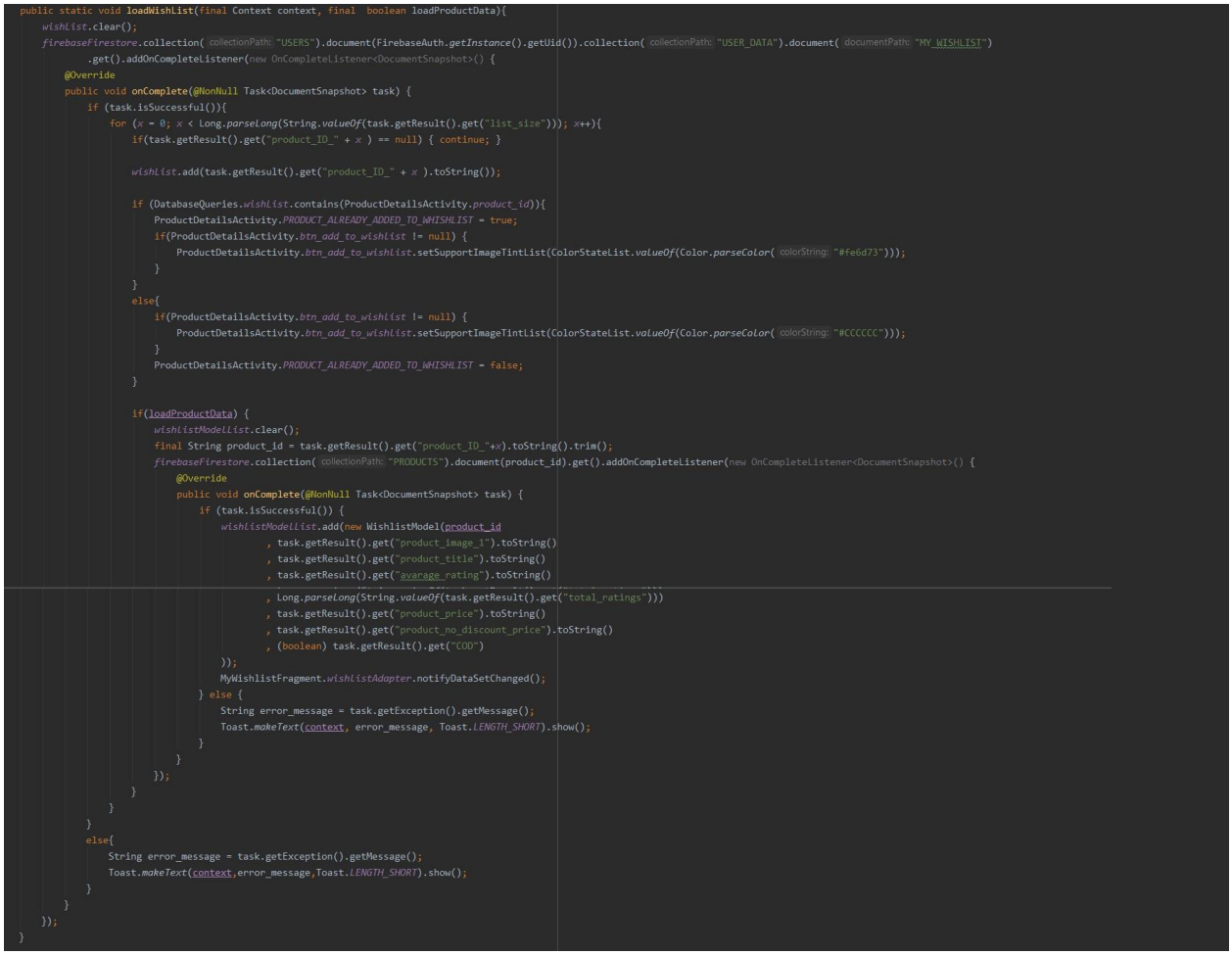
In this query, only men's products are listed.

```
firebaseFirestore.collection( collectionPath: "CATEGORIES").document( documentPath: "HOME").collection( collectionPath: "TOP_DEALS").whereEqualTo( field: "gender",  value: 2).get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
@Override
public void onComplete(@NonNull Task<QuerySnapshot> task) {
```

While defining gender in the database, it was defined as integer values. If the gender value in the product detail information is 1, that product is specific to women. If entered as 2, that product is specific to men.

- **Loading Wishlist**

There are codes that the user can add the favorite product to the wishlist. First of all, the place where the product is added to the wishlist is the page where all product details are shared. Therefore, all the information of the product is taken in the codes here. In addition, a check was made to see if the product is included in the wishlist. If it is not attached, the color of the like icon changes when the product is added. It is attached and if the user clicks this icon again, it changes to the default color.



```
public static void loadWishList(final Context context, final  boolean loadProductData){
    wishlist.clear();
    firebaseFirestore.collection( collectionPath: "USERS").document(FirebaseAuth.getInstance().getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_WISHLIST")
            .get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()){
                for (x = 0; x < Long.parseLong(String.valueOf(task.getResult().get("list_size"))); x++){
                    if(task.getResult().get("product_ID_" + x ) == null) { continue; }

                    wishList.add(task.getResult().get("product_ID_" + x ).toString());

                    if (DatabaseQueries.wishlist.contains(ProductDetailsActivity.product_id)){
                        ProductDetailsActivity.PRODUCT_ALREADY_ADDED_TO_WHISHLIST = true;
                        if(ProductDetailsActivity.btn_add_to_wishlist != null) {
                            ProductDetailsActivity.btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#fe6d73")));
                        }
                    }
                    else{
                        if(ProductDetailsActivity.btn_add_to_wishlist != null) {
                            ProductDetailsActivity.btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
                        }
                        ProductDetailsActivity.PRODUCT_ALREADY_ADDED_TO_WHISHLIST = false;
                    }

                    if(loadProductData) {
                        wishlistModellist.clear();
                        final String product_id = task.getResult().get("product_ID_"+x).toString().trim();
                        firebaseFirestore.collection( collectionPath: "PRODUCTS").document(product_id).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                            @Override
                            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                                if (task.isSuccessful()) {
                                    wishlistModellist.add(new WishlistModel(product_id
                                            , task.getResult().get("product_image_1").toString()
                                            , task.getResult().get("product_title").toString()
                                            , task.getResult().get("avarage_rating").toString()
                                            , Long.parseLong(String.valueOf(task.getResult().get("total_ratings")))
                                            , task.getResult().get("product_price").toString()
                                            , task.getResult().get("product_no_discount_price").toString()
                                            , (boolean) task.getResult().get("COD")
                                    ));
                                    MyWishlistFragment.wishlistAdapter.notifyDataSetChanged();
                                } else {
                                    String error_message = task.getException().getMessage();
                                    Toast.makeText(context, error_message, Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    }
                }
            }
            else{
                String error_message = task.getException().getMessage();
                Toast.makeText(context,error_message,Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

- **Remove Wishlist**

If the user wants to remove the product he likes from the wishlist, the id of the product to be removed is taken and the deletion process is performed according to the id. After the deletions, the database should be updated.

If the number of liked products is 0, it means that no product is liked, that is, it cannot be deleted. And the colors on the like icon are updated again according to whether the user likes the product or not.

```
public static void removeProductFromWishlist(final int index, final Context context){
    //final String product_id = ("PRODUCT_ID_"+ x).trim();
    // if(wishList.size() <= 0) { return; }
    final String removed_product_id = wishList.get(index);
    wishList.remove(index);
    Map<String,Object> updateWishlist = new HashMap<>();

    for (int x = 0 ; x < wishList.size() ; x++){
        updateWishlist.put( k: "product_ID_" + x, wishList.get(x));
    }
    updateWishlist.put( k: "list_size", (long) wishList.size());

    firebaseFirestore.collection( collectionPath: "USERS").document(FirebaseAuth.getInstance().getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_WISHLIST")
            .set(updateWishlist).addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()){
                if (wishlistModelList.size() != 0){
                    wishlistModelList.remove(index);
                    MyWishlistFragment.wishlistAdapter.notifyDataSetChanged();
                }
                ProductDetailsActivity.PRODUCT_ALREADY_ADDED_TO_WHISHLIST = false;
                ProductDetailsActivity.btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
                Toast.makeText(context, text: "Product Remove Successfully!",Toast.LENGTH_SHORT).show();
            }
```

```
            else{
                if(ProductDetailsActivity.btn_add_to_wishlist != null){
                    ProductDetailsActivity.btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
                }
                wishList.add(index,removed_product_id);
                String error_message = task.getException().getMessage();
                Toast.makeText(context, error_message, Toast.LENGTH_SHORT).show();
            }
            if(ProductDetailsActivity.btn_add_to_wishlist != null) {
                ProductDetailsActivity.btn_add_to_wishlist.setEnabled(true);
            }
        }
    });
}
```

- **Load Ratings**

The code in which the votes are arranged according to the ID of the product is given below.

```
public static void loadRatingList(final Context context){
    user_rated_id.clear();
    user_rating.clear();
    firebaseFirestore.collection( collectionPath: "USERS").document(FirebaseAuth.getInstance().getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_RATINGS")
            .get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()){
                for (long x = 0 ; x < (long)task.getResult().get("list_size") ; x++){
                    user_rated_id.add(task.getResult().get("product_ID_" + x).toString());
                    user_rating.add((long)task.getResult().get("rating_" + x));
                    if (task.getResult().get("product_ID_" + x).toString().equals(ProductDetailsActivity.product_id) && ProductDetailsActivity.giving_star_to_the_product != null){
                        ProductDetailsActivity.initial_rating = Integer.parseInt(String.valueOf((long)task.getResult().get("rating_" + x)))-1;
                        ProductDetailsActivity.setRating(ProductDetailsActivity.initial_rating);
                    }
                }
            }
            else{
                String error_message = task.getException().getMessage();
                Toast.makeText(context, error_message, Toast.LENGTH_SHORT).show();
            }
        }
    });
}
```

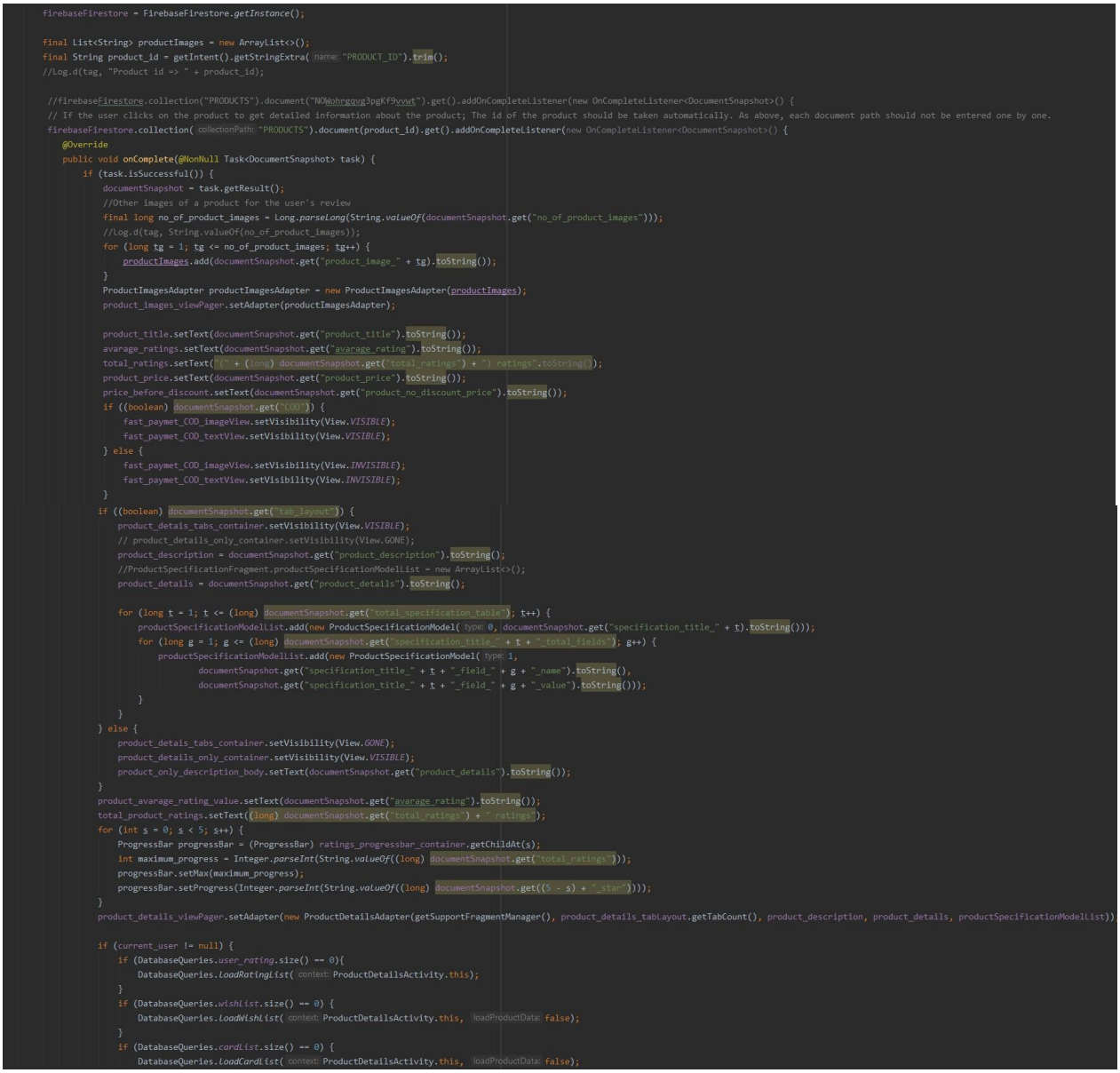
PRODUCT DETAILS

- **ProductDetailsActivity.Java**

There are many products in the application. Each product has its own id value. When the user clicks on a product to examine the product, the id value of that product is taken. Thus, there is no need to write the same codes over and over for each product.

Here, product information is pulled from the database. A product can have more than one picture. Here, how many pictures of each product, all pictures are taken from the database. A lot of information such as the detailed name of the product, how many votes it has received, average rating, current price, price before the discount, whether it is COD, detailed description of the product, detailed details, feature table are shown here.

Some checks have been made. For example, the user cannot use the application without being a member of the system. Those who log into the system without a valid e-mail address in the database will be greeted with a blank page. The main control made here is if there is someone using the application without logging in to the application, current_user will appear as null. And the user cannot make any comments about the products by voting. They cannot add the products they like to the wishlist or they cannot add products and buy any products in the snow.



```
firebaseFirestore = FirebaseFirestore.getInstance();

final List<String> productImages = new ArrayList<>();
final String product_id = getIntent().getStringExtra( name: "PRODUCT_ID").trim();
//Log.d(tag, "Product id => " + product_id);

 //firebaseFirestore.collection("PRODUCTS").document("NOWohrgqyg3pgKf9yvwt").get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
 // If the user clicks on the product to get detailed information about the product; The id of the product should be taken automatically. As above, each document path should not be entered one by one.
 firebaseFirestore.collection( collectionPath: "PRODUCTS").document(product_id).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            documentSnapshot = task.getResult();
            //Other images of a product for the user's review
            final long no_of_product_images = Long.parseLong(String.valueOf(documentSnapshot.get("no_of_product_images")));
            //Log.d(tag, String.valueOf(no_of_product_images));
            for (long tg = 1; tg <= no_of_product_images; tg++) {
                productImages.add(documentSnapshot.get("product_image_" + tg).toString());
            }
            ProductImagesAdapter productImagesAdapter = new ProductImagesAdapter(productImages);
            product_images_viewPager.setAdapter(productImagesAdapter);

            product_title.setText(documentSnapshot.get("product_title").toString());
            avarage_ratings.setText(documentSnapshot.get("avarage_rating").toString());
            total_ratings.setText("(" + (long) documentSnapshot.get("total_ratings") + ") ratings".toString());
            product_price.setText(documentSnapshot.get("product_price").toString());
            price_before_discount.setText(documentSnapshot.get("product_no_discount_price").toString());
            if ((boolean) documentSnapshot.get("COD")) {
                fast_paymet_COD_imageView.setVisibility(View.VISIBLE);
                fast_paymet_COD_textView.setVisibility(View.VISIBLE);
            } else {
                fast_paymet_COD_imageView.setVisibility(View.INVISIBLE);
                fast_paymet_COD_textView.setVisibility(View.INVISIBLE);
            }
            if ((boolean) documentSnapshot.get("tab_layout")) {
                product_details_tabs_container.setVisibility(View.VISIBLE);
                // product_details_only_container.setVisibility(View.GONE);
                product_description = documentSnapshot.get("product_description").toString();
                //ProductSpecificationFragment.productSpecificationModelList = new ArrayList<>();
                product_details = documentSnapshot.get("product_details").toString();

                for (long t = 1; t <= (long) documentSnapshot.get("total_specification_table"); t++) {
                    productSpecificationModelList.add(new ProductSpecificationModel( type: 0, documentSnapshot.get("specification_title_" + t).toString()));
                    for (long g = 1; g <= (long) documentSnapshot.get("specification_title_" + t + "_total_fields"); g++) {
                        productSpecificationModelList.add(new ProductSpecificationModel( type: 1,
                                documentSnapshot.get("specification_title_" + t + "_field_" + g + "_name").toString(),
                                documentSnapshot.get("specification_title_" + t + "_field_" + g + "_value").toString()));
                    }
                }
            } else {
                product_details_tabs_container.setVisibility(View.GONE);
                product_details_only_container.setVisibility(View.VISIBLE);
                product_only_description_body.setText(documentSnapshot.get("product_details").toString());
            }
            product_avarage_rating_value.setText(documentSnapshot.get("avarage_rating").toString());
            total_product_ratings.setText((long) documentSnapshot.get("total_ratings") + " ratings");
            for (int s = 0; s < 5; s++) {
                ProgressBar progressBar = (ProgressBar) ratings_progressbar_container.getChildAt(s);
                int maximum_progress = Integer.parseInt(String.valueOf((long) documentSnapshot.get("total_ratings")));
                progressBar.setMax(maximum_progress);
                progressBar.setProgress(Integer.parseInt(String.valueOf((long) documentSnapshot.get((5 - s) + "_star"))));
            }
            product_details_viewPager.setAdapter(new ProductDetailsAdapter(getSupportFragmentManager(), product_details_tabLayout.getTabCount(), product_description, product_details, productSpecificationModelList));

            if (current_user != null) {
                if (DatabaseQueries.user_rating.size() == 0){
                    DatabaseQueries.LoadRatingList( context: ProductDetailsActivity.this);
                }
                if (DatabaseQueries.wishList.size() == 0) {
                    DatabaseQueries.LoadWishList( context: ProductDetailsActivity.this,  loadProductData: false);
                }
                if (DatabaseQueries.cardList.size() == 0) {
                    DatabaseQueries.LoadCardList( context: ProductDetailsActivity.this,  loadProductData: false);
```

```
            if(DatabaseQueries.user_rated_id.contains(product_id)){
                int index = DatabaseQueries.user_rated_id.indexOf(product_id);
                initial_rating = Integer.parseInt(String.valueOf(DatabaseQueries.user_rating.get(index)))-1;
                setRating(initial_rating);
            }
            if (DatabaseQueries.wishlist.contains(product_id)) {
                PRODUCT_ALREADY_ADDED_TO_WHISHLIST = true;
                btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#fe6d73")));
            } else {
                PRODUCT_ALREADY_ADDED_TO_WHISHLIST = false;
            }
            if (DatabaseQueries.cardlist.contains(product_id)) {
                PRODUCT_ALREADY_ADDED_TO_CARD = true;
            } else {
                PRODUCT_ALREADY_ADDED_TO_CARD = false;
            }
        } else {
            String error_message = task.getException().getMessage();
            Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
        }
    }
});
```

It also organized wishlist, user ratings and card fields here. Below is the code block for the wishlist.

```
viewPager_tabLayout_indicator.setupWithViewPager(product_images_viewPager, autoRefresh: true);

btn_add_to_wishlist.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //the button changes color when the product is added to favorites
        if(current_user != null) {
            if (PRODUCT_ALREADY_ADDED_TO_WHISHLIST) {
                int index = DatabaseQueries.wishlist.indexOf(product_id);
                DatabaseQueries.removeProductFromWishlist(index, context: ProductDetailsActivity.this);
                btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#fe6d73")));
            } else {
                btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
                Map<String,Object> add_product = new HashMap<>();
                add_product.put( k: "product_ID_" + String.valueOf(DatabaseQueries.wishlist.size()), product_id);

                firebaseFirestore.collection( collectionPath: "USERS").document(current_user.getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_WISHLIST")
                        .update(add_product).addOnCompleteListener(new OnCompleteListener<Void>() {
                    @Override
                    public void onComplete(@NonNull Task<Void> task) {
                        if (task.isSuccessful()){

                            Map<String,Object> update_list_size = new HashMap<>();
                            update_list_size.put( k: "list_size",  v: (long) DatabaseQueries.wishlist.size() + 1);

                            firebaseFirestore.collection( collectionPath: "USERS").document(current_user.getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_WISHLIST")
                                    .update(update_list_size).addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()){

                                        if(DatabaseQueries.wishlistModellist.size() != 0){
                                            DatabaseQueries.wishlistModellist.add(new WishlistModel(product_id
                                                    ,documentSnapshot.get("product_image_1").toString()
                                                    ,documentSnapshot.get("product_title").toString()
                                                    ,documentSnapshot.get("avarage_rating").toString()
                                                    ,(long)documentSnapshot.get("total_ratings")
                                                    ,documentSnapshot.get("product_price").toString()
                                                    ,documentSnapshot.get("product_no_discount_price").toString()
                                                    ,(boolean)documentSnapshot.get("COD")
                                            ));
                                        }

                                        PRODUCT_ALREADY_ADDED_TO_WHISHLIST = true;
                                        btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#fe6d73")));
                                        DatabaseQueries.wishlist.add(product_id);
                                        Toast.makeText( context: ProductDetailsActivity.this, text: "Product Added to Wishlis Successfully",Toast.LENGTH_SHORT).show();
                                    }
                                    else{
                                        btn_add_to_wishlist.setSupportImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
                                        String error_message = task.getException().getMessage();
                                        Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
                                    }
                                    btn_add_to_wishlist.setEnabled(true);
                                }
                            });

                        }
                        else{
                            btn_add_to_wishlist.setEnabled(true);
                            String error_message = task.getException().getMessage();
                            Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
                        }
                    }
                });
            }
        }
    }
});

product_details_viewPager.addOnPageChangeListener(new TabLayout.TabLayoutOnPageChangeListener(product_details_tabLayout));
product_details_tabLayout.addOnTabSelectedListener(new TabLayout.OnTabSelectedListener() {
    @Override
    public void onTabSelected(TabLayout.Tab tab) {
        product_details_viewPager.setCurrentItem(tab.getPosition());
    }
```

The codes below are arranged for the user to vote for the product. The user can only vote once for a product. When you vote, the total number of votes is updated and the overall product average rating is recalculated.

```
    /*-----Product Rating Area-----*/
    giving_star_to_the_product = findViewById(R.id.your_ratings_stars_container);
    for (int x = 0; x < giving_star_to_the_product.getChildCount(); x++){
        final int starPosition = x;
        giving_star_to_the_product.getChildAt(x).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                    if (current_user != null) {
                        setRating(starPosition);
                        if (DatabaseQueries.user_rated_id.contains(product_id)) {

                        } else {
                            Map<String, Object> product_rating = new HashMap<>();
                            product_rating.put( k: starPosition + 1 + "_star", v: (long) documentSnapshot.get(starPosition + 1 + "_star") + 1);
                            product_rating.put( k: "avarage_rating", AverageRatingCalculation( current_user_rating: starPosition + 1));
                            product_rating.put( k: "total_ratings", v: (long) documentSnapshot.get("total_ratings") + 1);

                            firebaseFirestore.collection( collectionPath: "PRODUCTS").document(product_id).update(product_rating).addOnCompleteListener(new OnCompleteListener<Void>() {
                                @Override
                                public void onComplete(@NonNull Task<Void> task) {
                                    if (task.isSuccessful()) {
                                        Map<String, Object> rating = new HashMap<>();
                                        rating.put( k: "list_size", v: (long) DatabaseQueries.user_rated_id.size() + 1);
                                        rating.put( k: "product_ID_" + DatabaseQueries.user_rated_id.size(), product_id);
                                        rating.put( k: "rating_" + DatabaseQueries.user_rated_id.size(), v: (long) starPosition + 1);

                                        firebaseFirestore.collection( collectionPath: "USERS").document(current_user.getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_RATINGS")
                                                .update(rating).addOnCompleteListener(new OnCompleteListener<Void>() {
                                            @Override
                                            public void onComplete(@NonNull Task<Void> task) {
                                                if (task.isSuccessful()) {
                                                    DatabaseQueries.user_rated_id.add(product_id);
                                                    DatabaseQueries.user_rating.add((long) starPosition + 1);

                                                    total_ratings.setText("(" + ((long)documentSnapshot.get("total_ratings") + 1) + ") ratings");
                                                    total_product_ratings.setText((long) documentSnapshot.get("total_ratings") + 1 + " ratings");
                                                    avarage_ratings.setText(String.valueOf(AverageRatingCalculation( current_user_rating: starPosition + 1)));
                                                    product_avarage_rating_value.setText(String.valueOf(AverageRatingCalculation( current_user_rating: starPosition + 1)));
                                                    product_avarage_rating_value.setText(String.valueOf(AverageRatingCalculation( current_user_rating: starPosition + 1)));

                                                    for (int s = 0; s < 5; s++) {
                                                        ProgressBar progressBar = (ProgressBar) ratings_progressbar_container.getChildAt(s);
                                                        int maximum_progress = Integer.parseInt(String.valueOf((long) documentSnapshot.get("total_ratings") + 1 ));
                                                        progressBar.setMax(maximum_progress);
                                                        progressBar.setProgress(Integer.parseInt(String.valueOf((long) documentSnapshot.get((5 - s) + "_star"))));
                                                    }
                                                    Toast.makeText( context: ProductDetailsActivity.this, text: "Thank you for rating", Toast.LENGTH_SHORT).show();
                                                } else {
                                                    setRating(initial_rating);
                                                    String error_message = task.getException().getMessage();
                                                    Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
                                                }
                                            }
                                        });

                                    } else {
                                        setRating(initial_rating);
                                        String error_message = task.getException().getMessage();
                                        Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });
                        }
                    }
                }
        });
    }
public static void setRating(int starPosition) {
    for (int x = 0; x < giving_star_to_the_product.getChildCount(); x++) {
        ImageView btn_star = (ImageView) giving_star_to_the_product.getChildAt(x);
        btn_star.setImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#CCCCCC")));
        if (x <= starPosition) {
            btn_star.setImageTintList(ColorStateList.valueOf(Color.parseColor( colorString: "#ffc107")));
        }
    }
}

private float AverageRatingCalculation(long current_user_rating){
    long total_stars = 0;
    for (int x = 1 ; x <= 5 ; x++){
        total_stars = total_stars + (long) documentSnapshot.get(x + "_star") * x;
    }
    total_stars = total_stars + current_user_rating;
    long calculate_average_rating = total_stars / ((long) documentSnapshot.get("total_ratings") + 1);
    return calculate_average_rating;
}
```

When the user clicks the buy now button, he will be directed to the delivery page. When the user clicks the add-to-card button, the product will be added to the card. If the user unwittingly wants to add the product to the card again, user will receive a warning message that this product is already added.

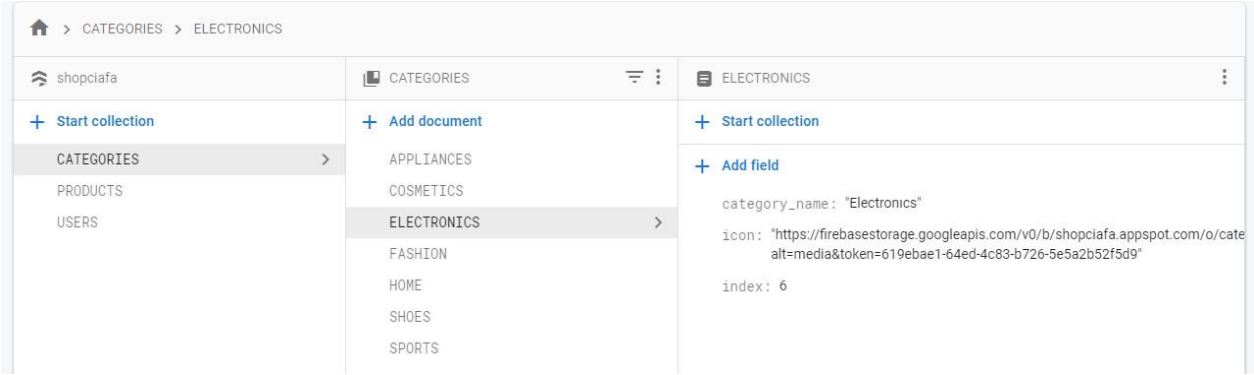
```
    btn_buy_now.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(current_user != null) {
                Intent delivey_intent = new Intent(ProductDetailsActivity.this, DeliveryActivity.class);
                startActivity(delivey_intent);
            }
        }
    });

    btn_add_to_card.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(current_user != null) {
                if (PRODUCT_ALREADY_ADDED_TO_CARD) {
                    Toast.makeText( context: ProductDetailsActivity.this, text: "Already added to card",Toast.LENGTH_SHORT).show();
                    } else {
                    Map<String,Object> add_product = new HashMap<>();
                    add_product.put( k: "product_ID_" + String.valueOf(DatabaseQueries.cardList.size()), product_id);
                    add_product.put( k: "list_size", (long) (DatabaseQueries.cardList.size() + 1));

                    firebaseFirestore.collection( collectionPath: "USERS").document(current_user.getUid()).collection( collectionPath: "USER_DATA").document( documentPath: "MY_CARD")
                            .update(add_product).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if (task.isSuccessful()){
                                if(DatabaseQueries.cardItemsModelList.size() != 0){
                                    DatabaseQueries.cardItemsModelList.add(new CardItemsModel(CardItemsModel.CART_ITEM, product_id
                                            , documentSnapshot.get("product_image_1").toString()
                                            , documentSnapshot.get("product_title").toString()
                                            , documentSnapshot.get("product_price").toString()
                                            , documentSnapshot.get("product_no_discount_price").toString()
                                            , (long) 1
                                    ));
                                }
                                PRODUCT_ALREADY_ADDED_TO_CARD = true;
                                DatabaseQueries.cardList.add(product_id);
                                Toast.makeText( context: ProductDetailsActivity.this, text: "Product Added to Card Successfully",Toast.LENGTH_SHORT).show();
                                btn_add_to_card.setEnabled(false);
                            }
                            else{
                                btn_add_to_card.setEnabled(true);
                                String error_message = task.getException().getMessage();
                                Toast.makeText( context: ProductDetailsActivity.this, error_message, Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                }
            }
        }
    });
}
```

1.5 Test and Experiment

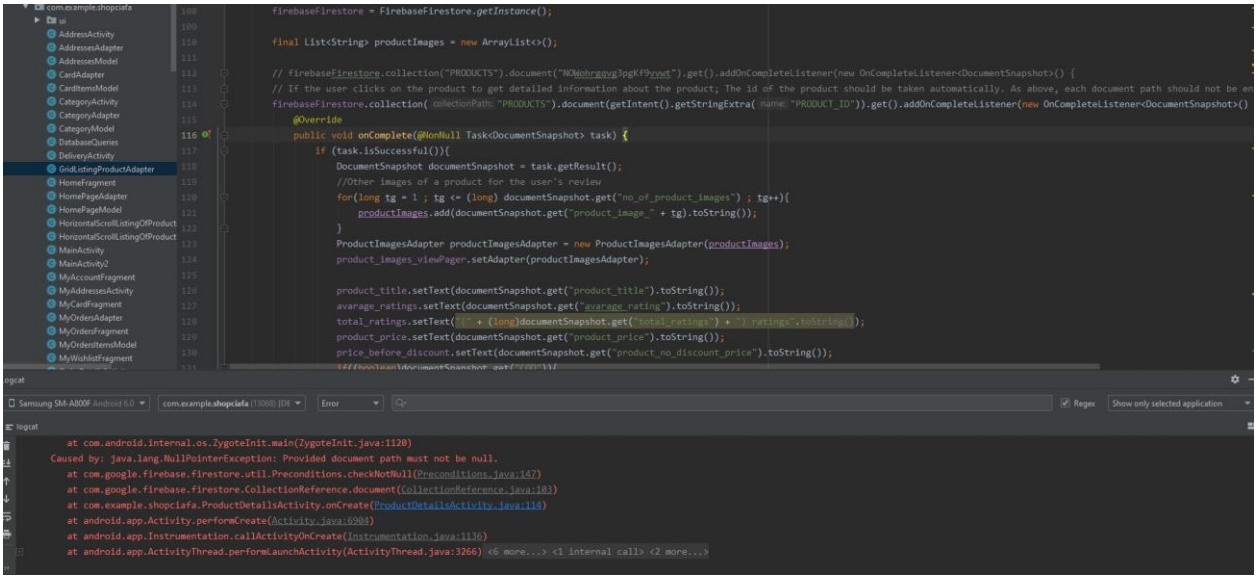
```
// There is an i letter in categories such as fashion, appliances, cosmetics. When the toUpperCase method is applied, it occurs and this causes an error in the program.
// Here the letter İ has been changed to I.
String target = categoryName.toUpperCase();
Character harf = 'İ';
if (target.contains("İ")) {
    target = target.replace(harf,  newChar: 'I');
}
```

While performing database operations, an error was encountered in the category name section. A category collection was created and the category names that will appear in the application were added with capital letters. These added data are in a document. Then, an index value is entered to set the names of these categories, icons and the order we want them to appear in the application.

For example; collection name is CATEGORIES. Document name is ELECTRONICS. However, while entering electronic information, it was entered as Electornics in the category_name field. While using category_name in the code, the uppercase category name was obtained as capital letters using the upperCase method. However, since it contains the letter i in the case it is written with a lowercase letter, it turned into an İ when it was enlarged. This is supposed to be I. That's why the application was terminate. The above code has been added to the codes written to eliminate this problem.



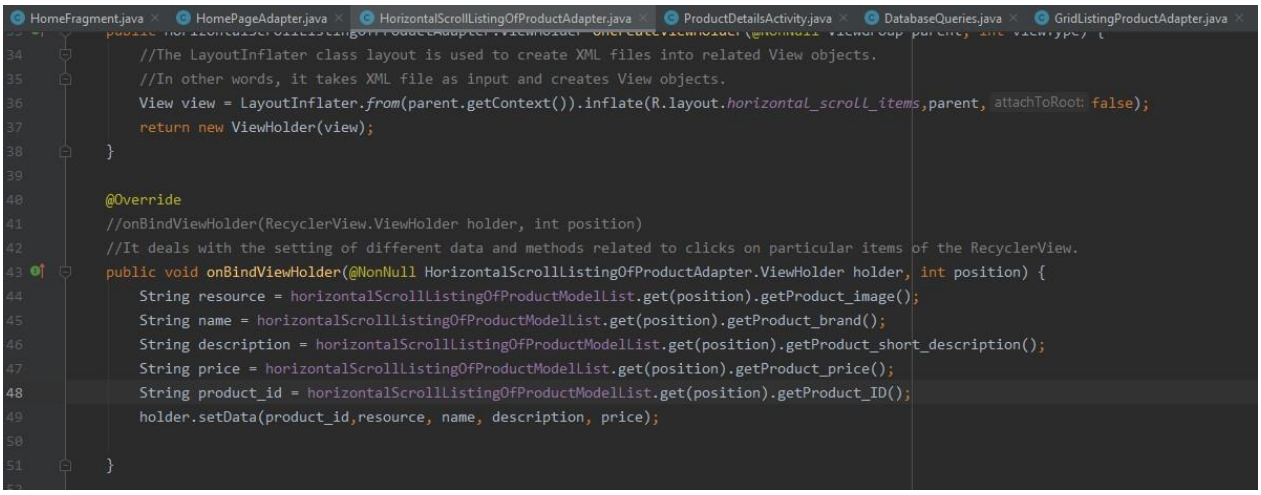
Another error received in the project;

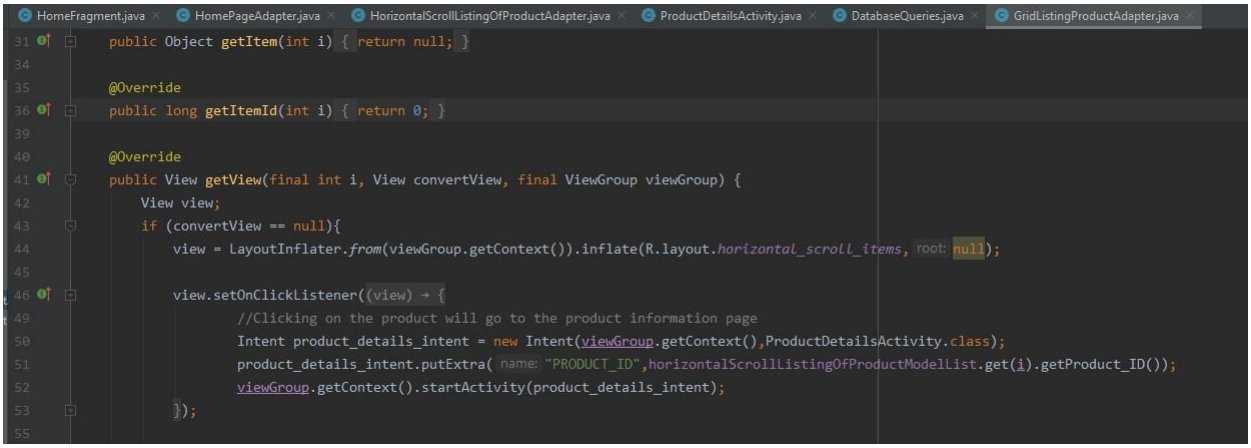


In order to solve this error, a separate page has been edited. These pages are HorizontalScrollListingOfProductAdapter, GridListingProductAdapter. While writing the code in the page, the id value information was not added at the beginning.

```
private void setData(final String product_id, String resource, String name, String description, String price){
    Glide.with(itemView.getContext()).load(resource).apply(new RequestOptions().placeholder(R.mipmap.picture)).into(product_image);
    product_brand.setText(name);
    product_short_description.setText(description);
    product_price.setText(price);

    itemView.setOnClickListener((view) → {
            //To know the details about the product, go to the product details page
            Intent product_details_intent = new Intent(itemView.getContext(),ProductDetailsActivity.class);
            product_details_intent.putExtra( name: "PRODUCT_ID", product_id);
            itemView.getContext().startActivity(product_details_intent);
    });
```

The error was resolved by adding the following codes.





But there was another problem. The other problem was errors due to space. While entering the product's id value, the space key was unknowingly pressed. This resulted in the length of the id value, which normally consists of 20 characters, to be perceived as 21. When you want to view information about the product, the application quits. When debugged, it was seen that the number of product ID value was different from normal. Clipping functionality has been added to the code to prevent similar errors.

The error is completely resolved after editing in this section.

```java
final List<String> productImages = new ArrayList<>();
final String product_id = getIntent().getStringExtra( name: "PRODUCT_ID").trim();
//Log.d(tag, "Product id => " + product_id);

 //firebaseFirestore.collection("PRODUCTS").document("NOWohrgqvg3pgKf9yvwt").get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
 // If the user clicks on the product to get detailed information about the product; The id of the product should be taken automatically. As above, each document path should not be
 firebaseFirestore.collection( collectionPath: "PRODUCTS").document(product_id).get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            documentSnapshot = task.getResult();
            //Other images of a product for the user's review
            final long no_of_product_images = Long.parseLong(String.valueOf(documentSnapshot.get("no_of_product_images")));
            //Log.d(tag, String.valueOf(no_of_product_images));
            for (long tg = 1; tg <= no_of_product_images; tg++) {
                productImages.add(documentSnapshot.get("product_image_" + tg).toString());
            }
            ProductImagesAdapter productImagesAdapter = new ProductImagesAdapter(productImages);
            product_images_viewPager.setAdapter(productImagesAdapter);

            product_title.setText(documentSnapshot.get("product_title").toString());
            avarage_ratings.setText(documentSnapshot.get("avarage_rating").toString());
            total_ratings.setText("(" + (long) documentSnapshot.get("total_ratings") + ") ratings".toString());
            product_price.setText(documentSnapshot.get("product_price").toString());
            price_before_discount.setText(documentSnapshot.get("product_no_discount_price").toString());
            if ((boolean) documentSnapshot.get("COD")) {
```

APPENDIX

The project is a big project and a lot of code has been written. It continues to be written. That's why not all codes are included. First, the home page and the modules and database operations and product details used there are shown. But in general, the logic of writing codes is always the same.

The variable names and method names are written in a very long and descriptive way so that the person who opens the project can easily understand the codes.

Also, many errors were encountered while coding the project, but screenshots of all errors were not taken. In the test and experiment part, errors that are challenging for me and I will spend a lot of time to solve are shown.

RESOURCES

1- https://developer.android.com/reference/androidx/viewpager/widget/PagerAdapter

2- https://www.codota.com/code/java/methods/android.widget.ArrayAdapter/notifyDataSetChanged

3- https://developer.android.com/topic/libraries/data-binding/binding-adapters

4- https://guides.codepath.com/android/viewpager-with-fragmentpageradapter

5- https://programmersought.com/article/6104114163/

6- https://stackoverflow.com/questions/53350635/navigation-drawer-onnavigationitemselected-not-responding

7- https://www.codota.com/code/java/methods/androidx.drawerlayout.widget.DrawerLayout/setDrawerLockMode

8- https://github.com/tranphunguyen98/MyMall

9- https://github.com/yassinekhaldi00/gender-and-age-detection

10- https://www.mobilhanem.com/android-navigation-drawer-ile-slider-menu-yapimi/

11- https://www.ahmetcevahircinar.com.tr/2016/09/10/androidte-navigation-drawer-ile-birlikte-fragment-nasil-kullanilir/

12- https://codelabs.developers.google.com/codelabs/recognize-flowers-with-tensorflow-on-android#4

13- https://stackoverflow.com/questions/29782808/what-does-fragmentmanager-and-fragmenttransaction-exactly-do