**MOB HUB**

*A Project Report Submitted in partial fulfilment of the*

*Requirements for the award of the degree of*

**BACHELOR OF COMPUTER APPLICATIONS (BCA)**

Submitted by

**SAMREEN FATHIMA M**

**(Reg No. 2113441033085)**

Under the Guidance of

**Dr(Mrs.)R.LALITHA,MCA.,M.Phil.,Ph.D.,** 

**DEPARTMENT OF COMPUTER SCIENCE**

**JUSTICE BASHEER AHMED SAYEED COLLEGE FOR WOMEN**

**(AFTERNOON SESSION), (AUTONOMOUS)**

**TEYNAMPET, CHENNAI-600 018**

**2023-2024**

**JUSTICE BASHEER AHMED SAYEED COLLEGE FOR WOMEN**

**(A.N)(AUTONOMOUS)**

**TEYNAMPET, CHENNAI-600 018.**

**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled as “**MOB HUB SHOP**” submitted to “**JUSTICE BASHEER AHMED SAYEED COLLEGE FOR WOMEN**” in partial fulfilment of the requirement for the award of the degree of Bachelor of Computer Applications (BCA), is an original work done by **SAMREEN FATHIMA M**, Register number **2113441033006**, under the guidance of **Dr(Mrs.)R.LALITHA,MCA.,M.Phil.,Ph.D.,**  during the period of December 2023 – April 2024.

**STAFF INCHARGE HEAD OF THE DEPARTMENT**

**DATE: [DEPARTMENT OF COMPUTER SCIENCE]**

**EXAMINER:**

**1.**

**2.**

**ACKNOWLEDGEMENT**

I thank the almighty for his blessings and mercy showered on me in each and every step to complete this project successfully.

I submit my under gratitude to our **Principal and Correspondent Dr (Mrs.)AMTHUL AZEEZ ,MSc.,M.Phil.,Ph.D.,** of our college for the facilities provided to carry out this project.

I would like to express my heartfelt thanks and deep gratitude to our **Head of the Department**, **Dr (Mrs.) Reshma, MCA, M.Phil., Ph.D.,** for her valuable advice and ideas at various stages of this work.

I render my thanks and high sense of gratitude to my **Internal Guide Dr(Mrs.)R.LALITHA,MCA.,M.Phil.,P.hd.,** Assistant Professor**,** Department of Computer Science for suggesting this topic and for her valuable support and keen interest with which she imparted her knowledge. I thank her for her accurate suggestion and proper direction, which enabled me to pursue this project work with great interest and curiosity and for her constant encouragement in making this project a successful one.

I sincerely thank all staff members of computer science department for their constant encouragement. I would like to express my sincere gratitude and deep appreciation to all those who have helped me in every possible way to successfully carry out this project.

I wish to submit my sincerity and thanks to my parents, friends, brothers and sisters for encouraging me in all walks of my life. Once again, I register my gratitude to almighty for the blessing showed on me.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NO** |
|  | **ABSTRACT** |  |
|  | **INTRODUCTION**  2.1. OBJECTIVES |  |
|  | **SYSTEM REQUIREMENT**  3.1 HARDWARE REQUIREMENTS  3.2 SOFTWARE REQUIREMENTS |  |
|  | **SYSTEM ANALYSIS**  4.1. EXISTING SYSTEM  4.2. PROPOSED SYSTEM  4.3. PROJECT DESCRIPTION |  |
|  | **SOFTWARE DESCRIPTION** |  |
|  | **SOFTWARE DESIGN**  6.1 ACTIVITY FLOW DIAGRAM |  |
|  | **SOFTWARE TESTING AND IMPLEMENTATION** |  |
|  | **APPENDICES**   * 1. SOURCE CODE   2. SCREENSHOTS |  |
|  | **CONCULSION & FUTURE ENHANCEMENT** |  |
|  | **BIBLIOGRAPHY & WEBLIOGRAPHY** |  |

**1.ABSTRACT**

The "Mob Hub" is a mobile application designed to provide users with a convenient and efficient platform for exploring, purchasing, and managing a wide range of accessories for their mobile devices. The app aims to enhance the overall mobile experience by offering a diverse selection of high-quality accessories such as cases, chargers, headphones, screen protectors, and more. The Mob Hub aims to create a one-stop-shop for mobile users, offering not only a vast selection of accessories but also an enriching and personalized shopping experience. Through a blend of user-friendly design and robust features, the app strives to become an indispensable companion for mobile enthusiasts seeking to enhance and customize their devices. The app aims to enhance the shopping experience for mobile accessory enthusiasts by offering a seamless and personalized platform accessible through mobile devices, allowing users to explore and shop for their favorite accessories anytime and anywhere.

The project involves developing a mobile accessories app using java.

**2. INTRODUCTION**

The dynamic world of mobile technology, the importance of accessories to complement and enhance the functionality of our devices cannot be overstated. Mobile accessories have evolved from mere add-ons to essential companions, offering users a personalized and optimized experience with their smartphones and tablets. From protective cases and stylish covers to advanced chargers and cutting-edge audio devices, mobile accessories have become an integral part of our daily lives. As mobile devices continue to evolve, so too will the landscape of mobile accessories. The synergy between advancements in technology and the creativity of accessory manufacturers ensures that users can tailor their devices to meet their unique needs and preferences. The world of mobile accessories is not just about functionality; it's a canvas for personal expression and a testament to the ever-expanding possibilities within the realm of mobile technology.

* 1. **OBJECTIVES**

The objective of the project is to make an application in android platform to make online orders In order to build such an application complete web support need to be provided. A complete and efficient web application which can provide the user interaction is the basic objective of the project. The central concept of the application is to allow the customer to make orders virtually using the Internet and allow customers to buy there day today needs.

1. **SYSTEM REQUIREMENTS**
   1. **HARDWARE REQUIREMENTS**

* l-core 64-biDuat processor
* 20 GB of memory
* Up to 24 GB of internal storage:

1. Android SDK: 2GB
2. Windows SDK: 4GB

**3.2 SOFTWARE REQUIREMENTS**

* Android SDK tools 22.0.5
* Android 4.2.2 platform API 1
* Android 2.3.3
* Java Runtime Environment 1.6
* Java SE JDK v6.0

**4 .SYSTEM ANALAYSIS**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |

**4.1 EXISTING SYSTEM**

The current system is shopping accessories for smartphones, tablets and other devices is to visit the center manually and from the available center to choose their needs accordingly and that customer wants to buy.

1.Limited Physical Examination

2.Internet Dependency

3.Security Concerns

4.Delivery Delays and Costs

5.Return and Refund Processes

**4.2 PROPOSED SYSTEM**

The proposed system for mobile accessories aims to enhance convenience, functionality and user experience through innovative products and services. Here are some ideas for a proposed system for mobile accessories:

1.Smart Cases with Integrated Features

2.Wireless Charging Solutions

3.Fashionable and Trendy Designs

4.Environmentally Friendly Materials..etc

Implementing these proposed system ideas can help differentiate mobile accessories in the market, cater to evolving consumer needs, and create a more engaging and innovative ecosystem of accessories for mobile devices.

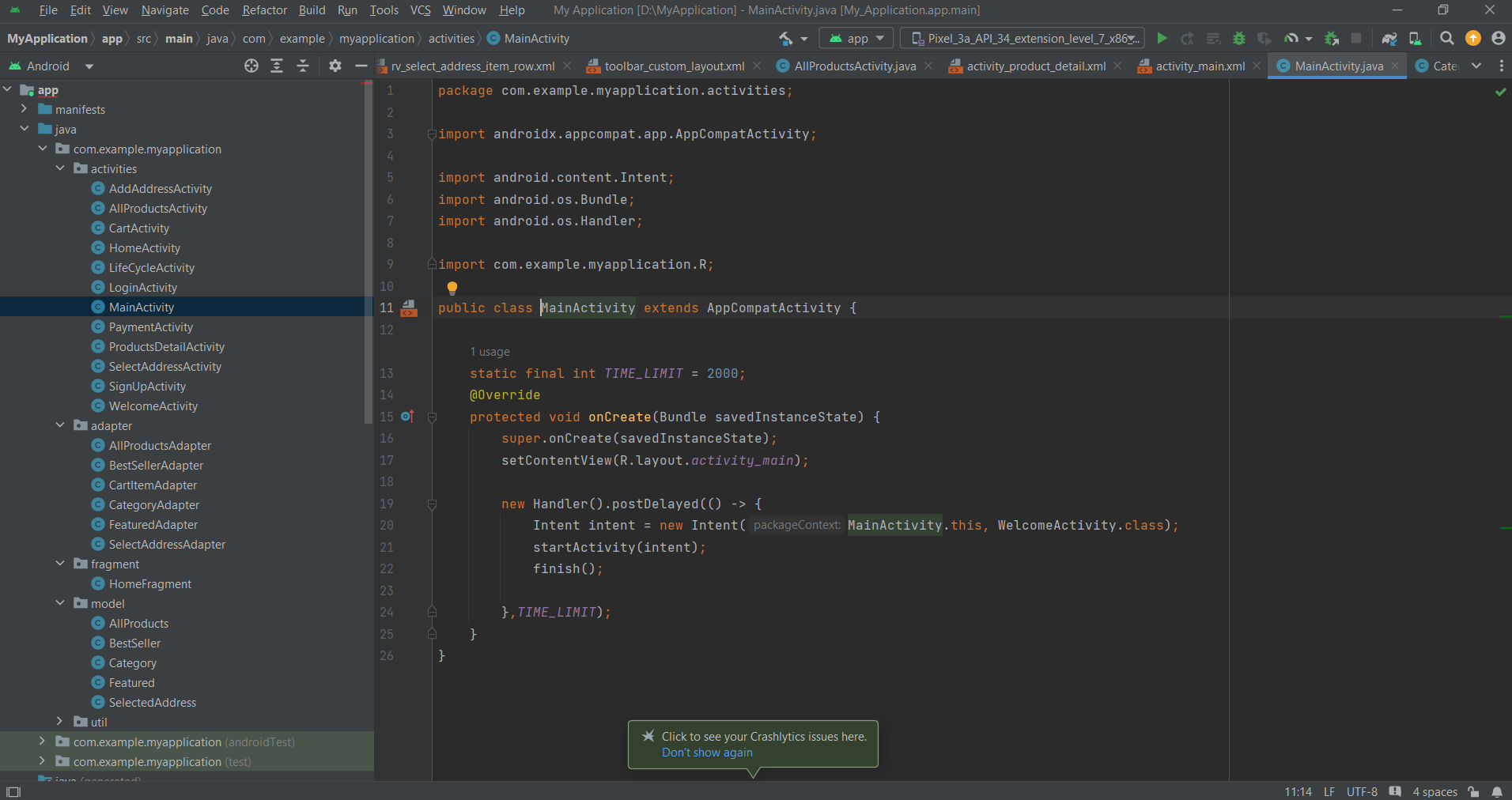
**4.3 PROJECT DESCRIPTION**

A mobile accessories shopping app is a specialized software application designed specifically for smartphones and tablets, offering a platform for users to browse, discover and purchase a wide range of accessories for their mobile devices. It aims to enhance the mobile shopping experience by offering a convenient, secure and feature-rich platform for users to discover.

**4.4 MODULE DESCRIPTION**

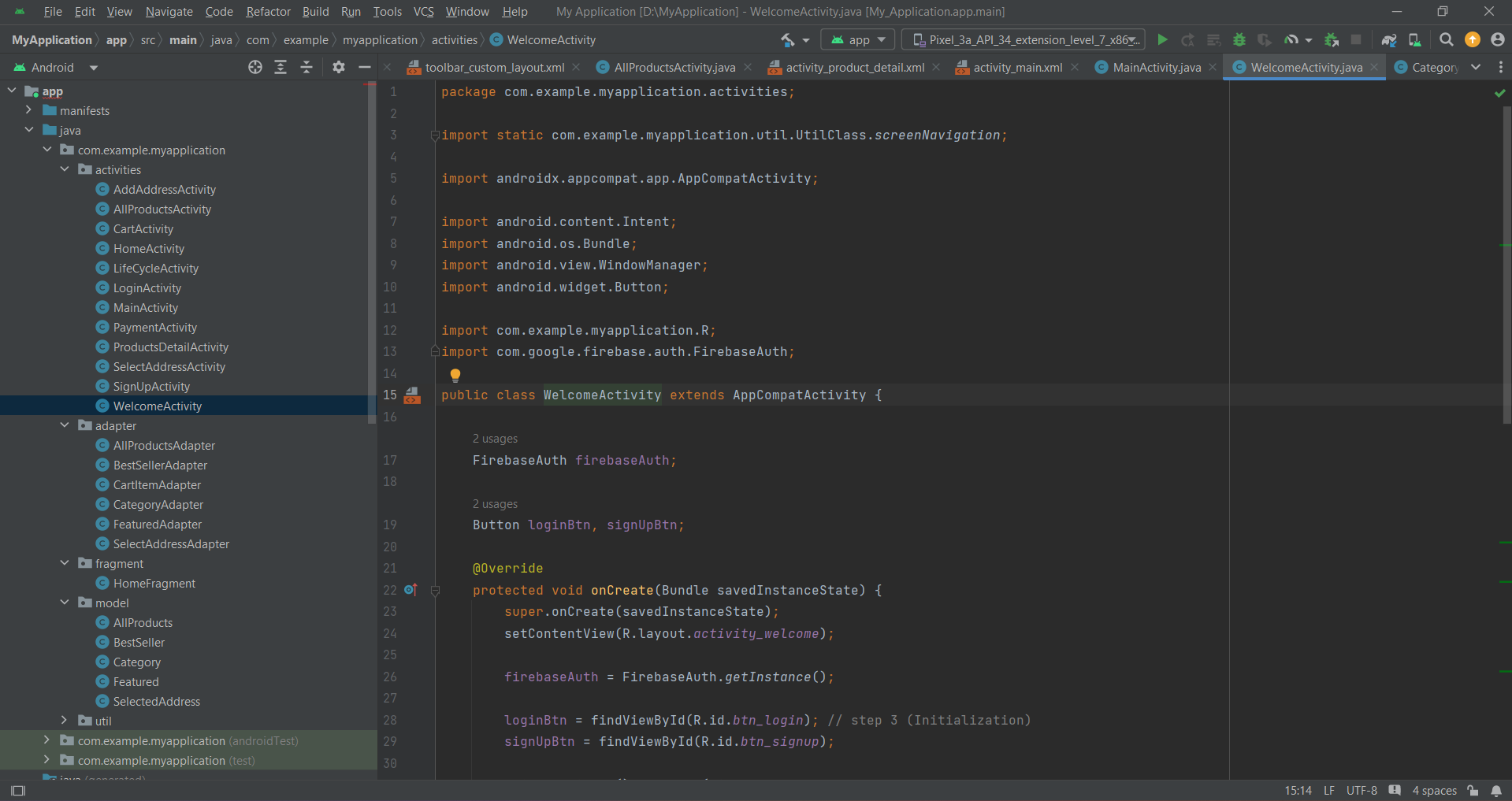
**4.4.1 Splash Screen**

TheSplash screen is a screen that loads when the app is launched. When the user opens the application, a loading screen, also known as a launch screen or startup screen, appears. It contains the logo of the application. A splash screen’s function is to immediately show an attractive screen while the application retrieves any relevant content (from a database or network calls). It contains the logo of the application.

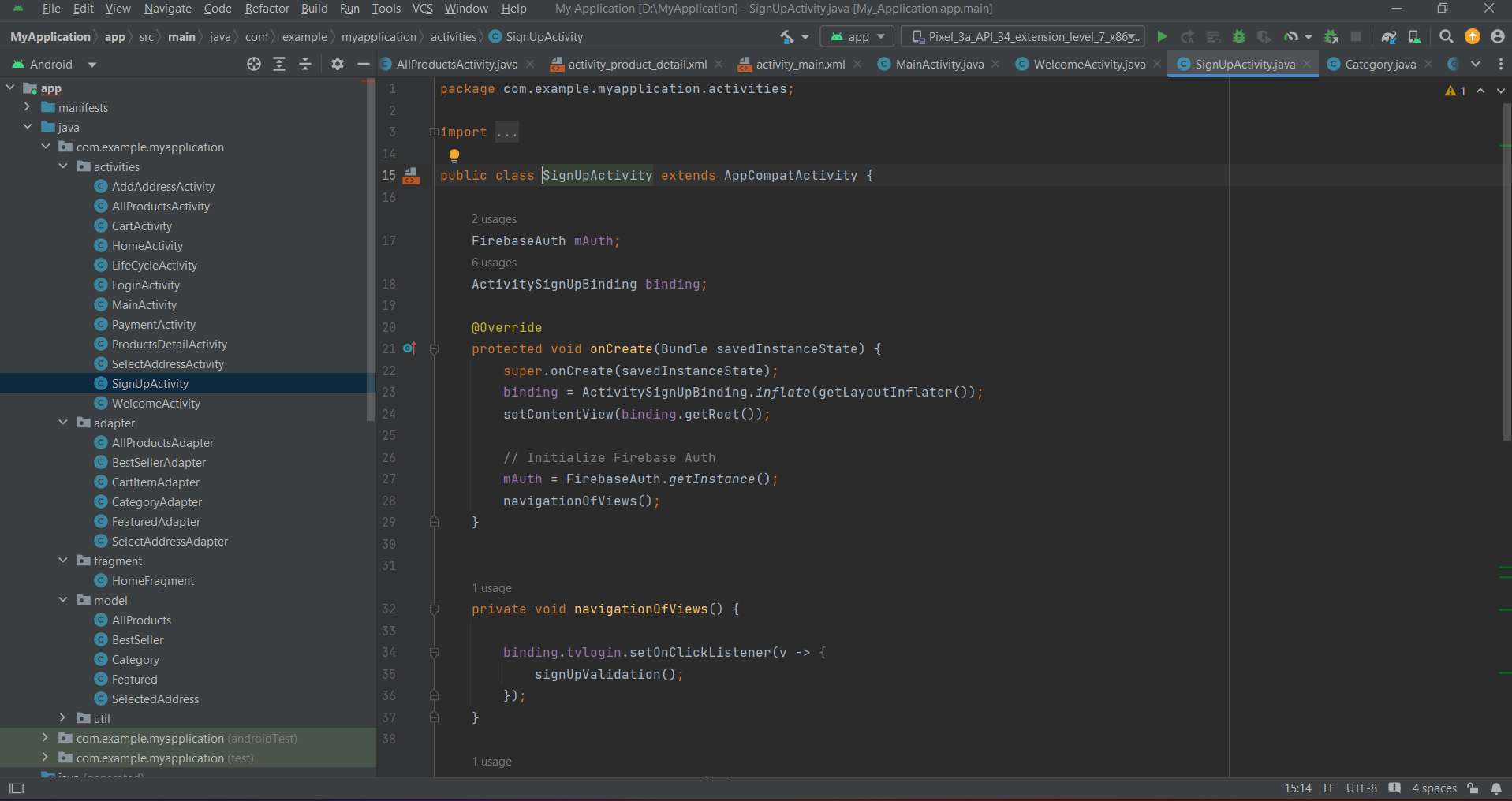


**4.4.2 Welcome Screen**

TheWelcome screen is used to greet the user with an attractive background and let’s the user to navigate to login screen or sign up screen. It has two buttons: LOGIN, SIGNUP which will lead to their respective screens for the user authentication.

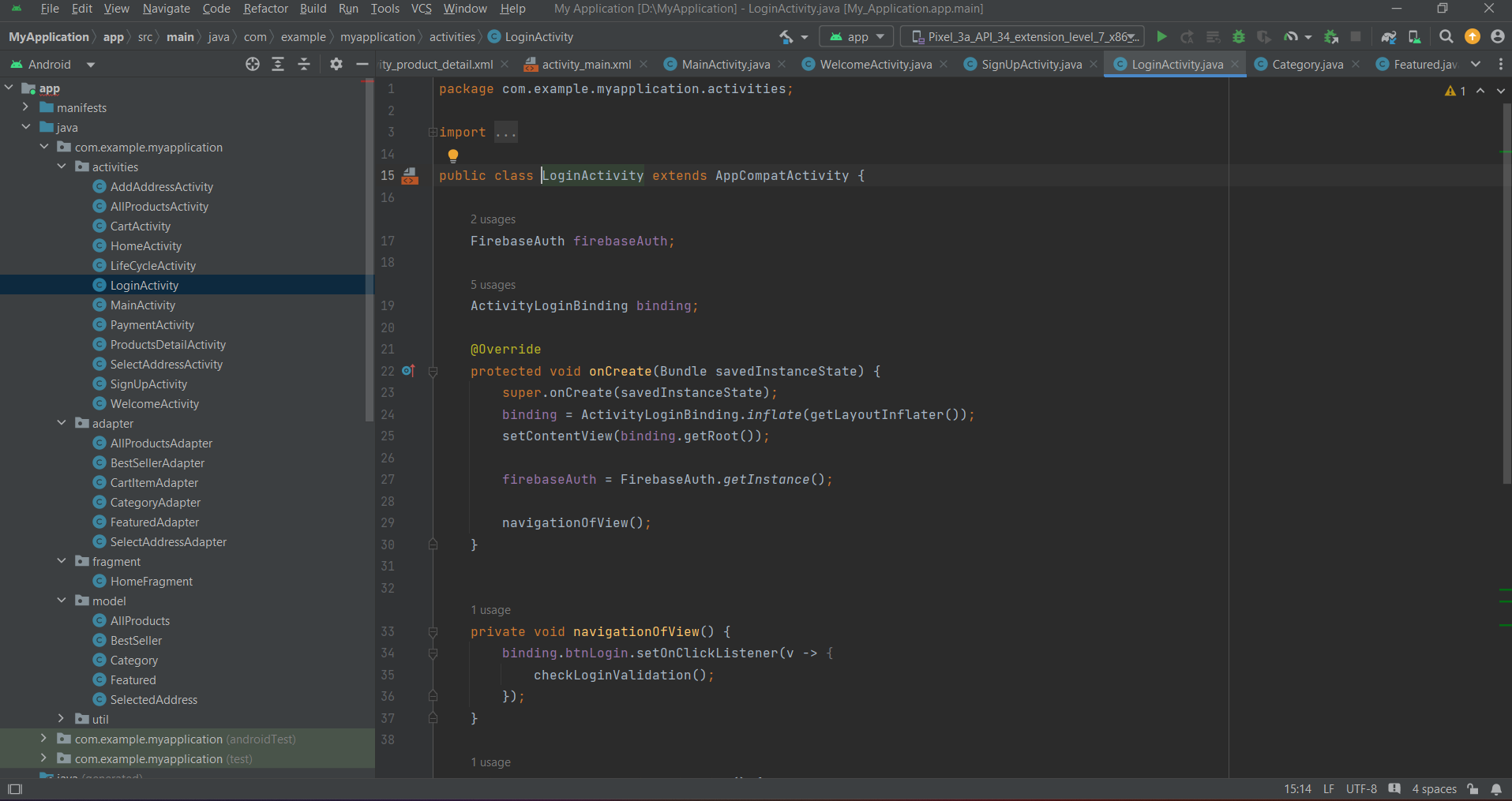


**4.4.3 Sign Up Screen**

The Sign up screen allows the new user to sign up to the application. It gets the username, email and password from the user and stores the email with an UID (user Id) that uniquely identifies each user using the app, in firebase authentication, which acts as the backend for user authentication. 

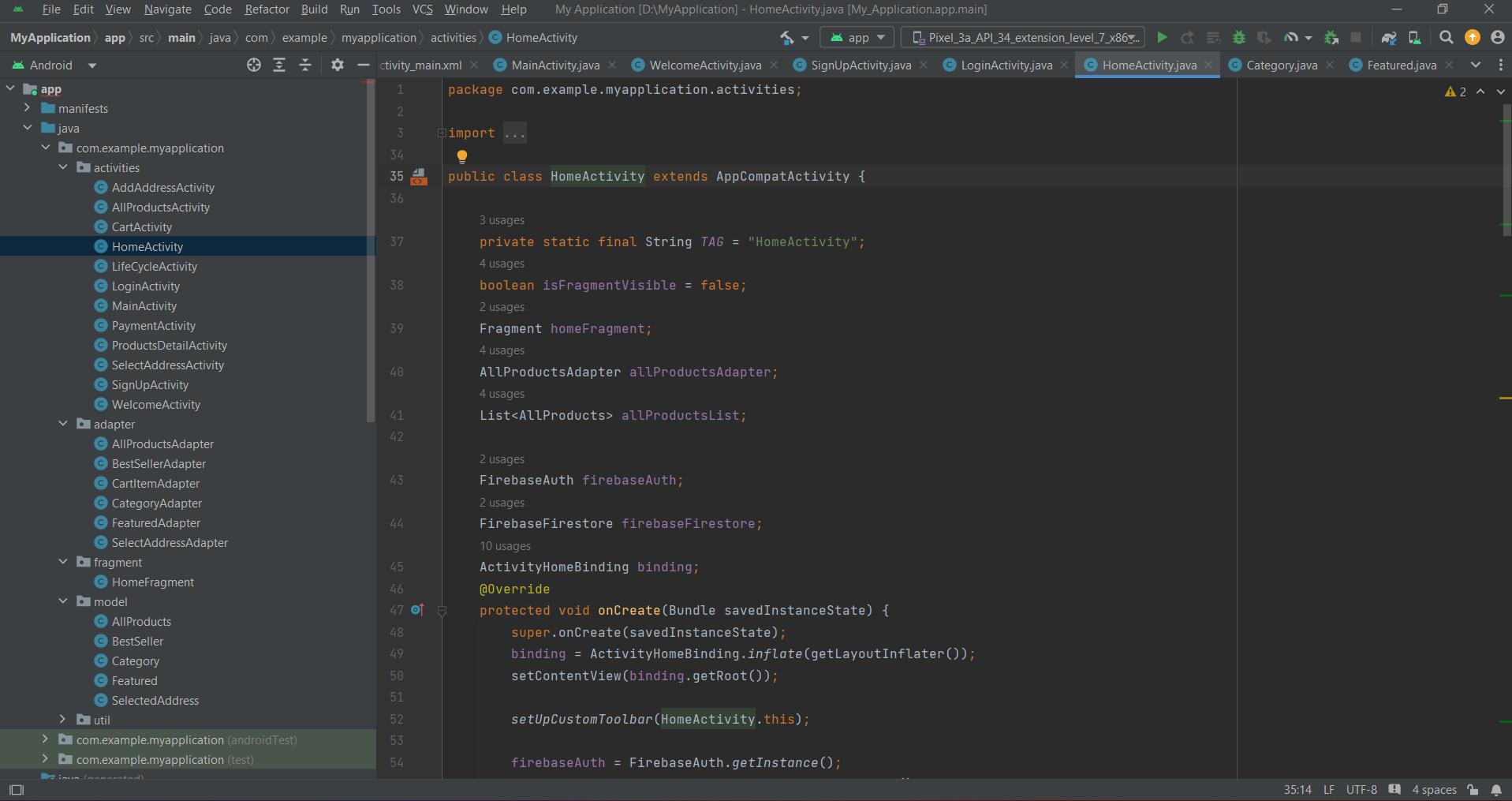
**4.4.4 Login Screen**

The Login screen allows the user to login into their existing account in the application. It gets the email and password from the user to authenticate the user using the app. The email is checked against the already stored information in firebase authentication, which authenticates the user if the email and its corresponding password match exactly. The main purpose of a login screen is to ensure that only authorized users are able to access the app.



**4.4.5 Home Screen**

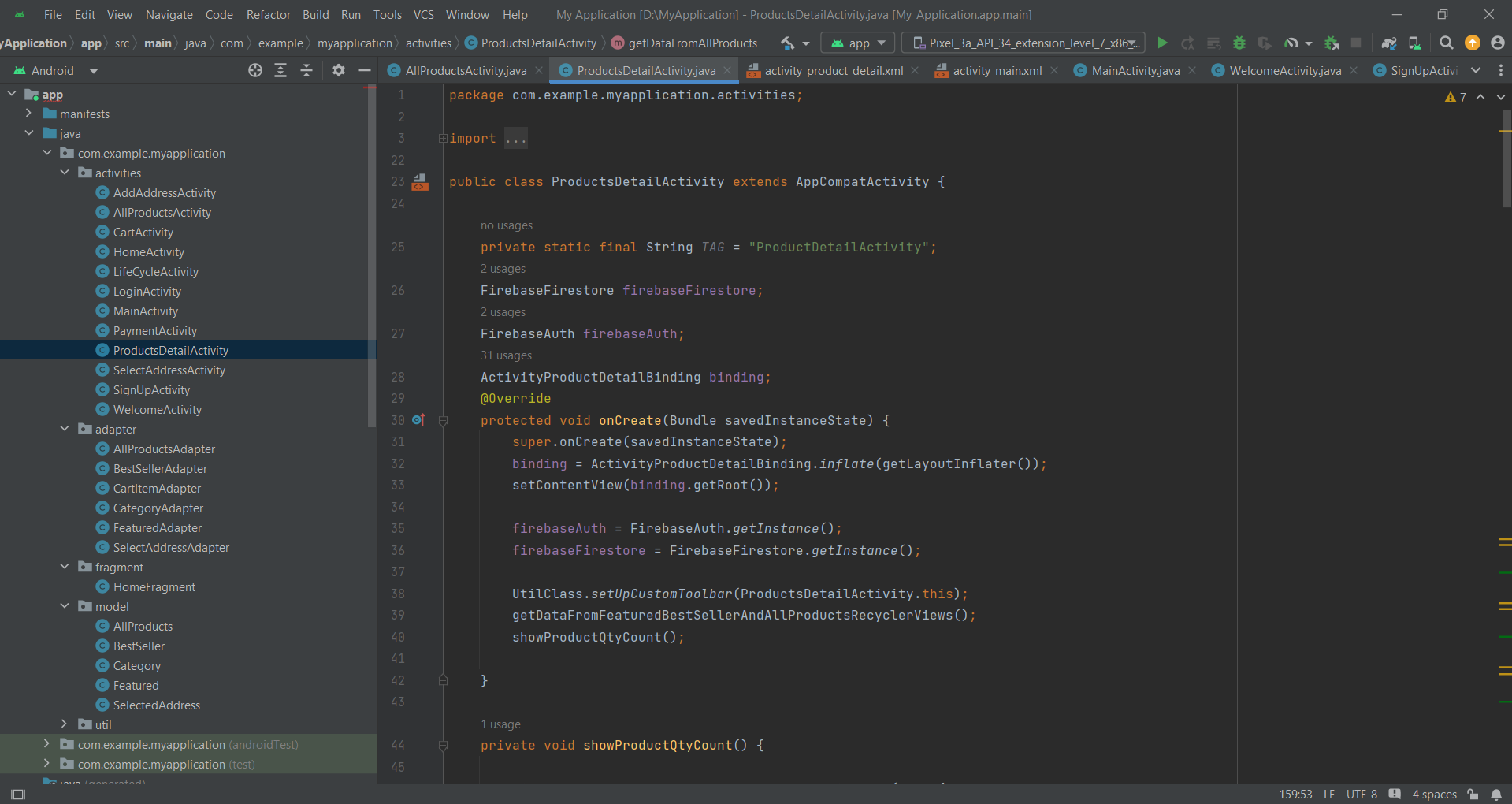
The Home Screen acts as gateway for the users shopping experience. It has 3 categories: Phone Cases, Air pods and Bluetooth Speakers. It shows a horizontally scrollable list of featured accessories in the app and the bestselling products in the application which also displays the product’s image, product’s name and price. The Cart option allows the user to navigate from home screen to cart screen.

**4.4.6 All Products Screen**

Once the user selects the genre, the screen navigates from genre screen to all products screen which displays the product image, name of the accessories and price in a grid layout. All these accessories displayed in this screen are of the genre selected by the user. All these data are fetched from the cloud firestore in real time through internet connection of the device running applications.

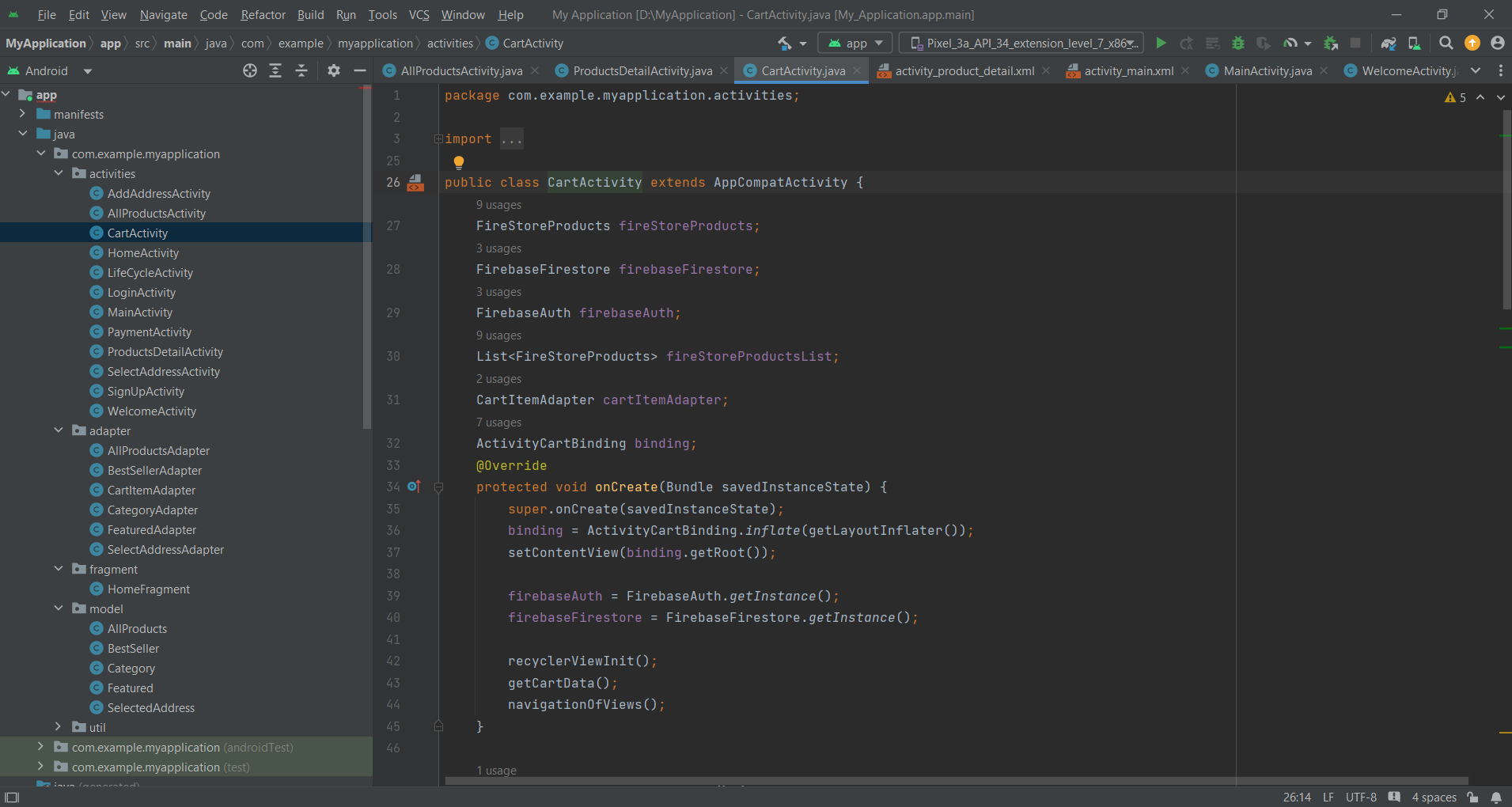
**4.4.7 Product Detail Screen**

Product detail screen displays all the details of the accessory being viewed. It displays the image of the product, description, price and the rating of the accessory is shown using a star rating bar, synopsis of the product.It allows the user to decrease and increase the quantity of the product to be purchased where the default value is 1. It also has an Add To Cart button which once clicked, will add the item in the cart and shows a toast message that the item is added is to the cart.



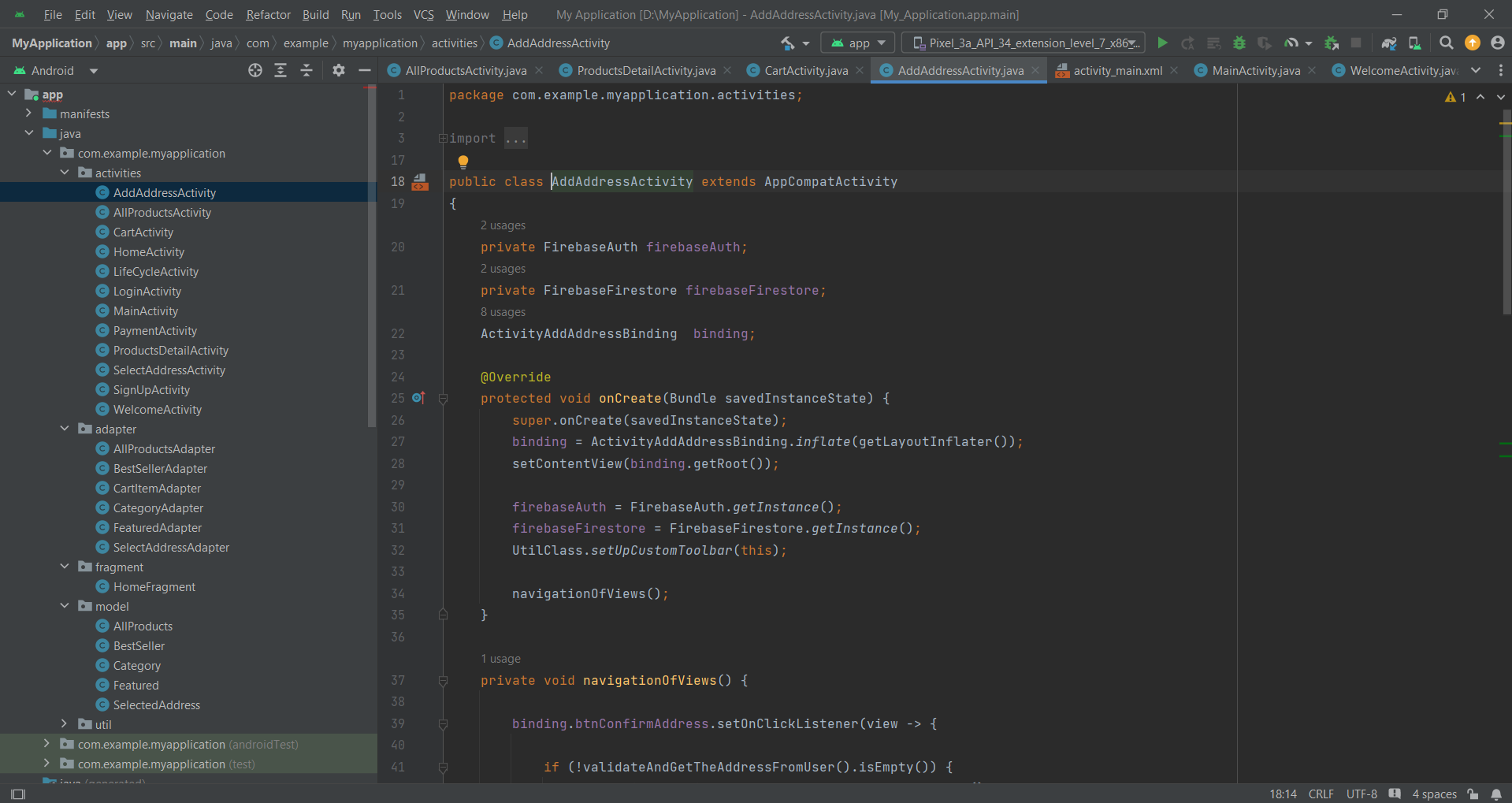
**4.4.8 Cart Screen**

The Cart screen displays the following information, product name, quantity to be purchased, price of one quantity, total price of the product and the remove option which removes the product when clicked for each product added in the cart. It displays overall total amount of all the products added to the cart.



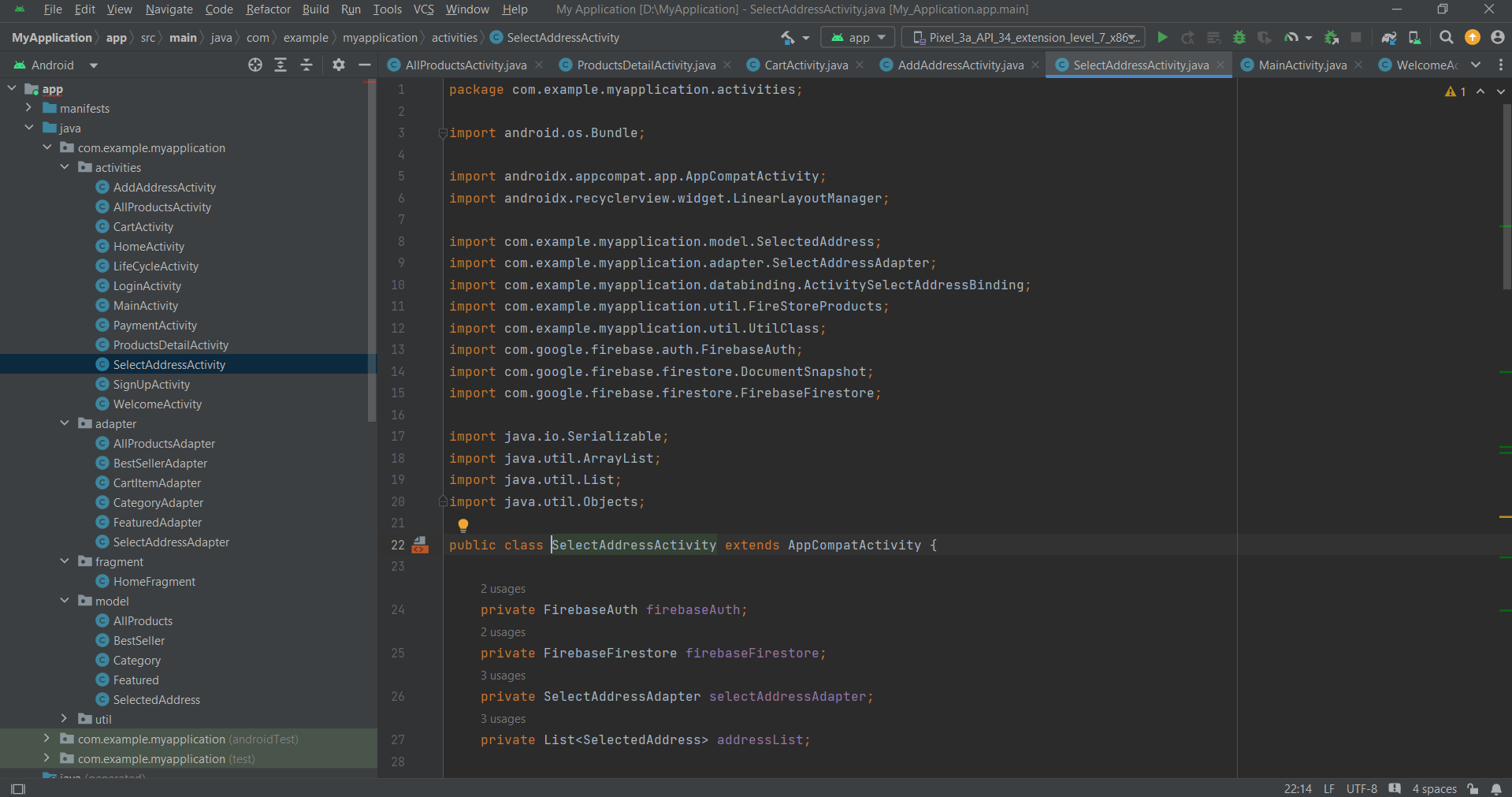
**4.4.9 Add Address Screen**

TheAdd Address screen allows the user to add an address of their own, The user will provide the following information: name, full address, city, postal code and phone number. The user can click the confirm address button when all the necessary details are given. It also stores the full address in the cloud, so the data can be displayed in the select address screen.



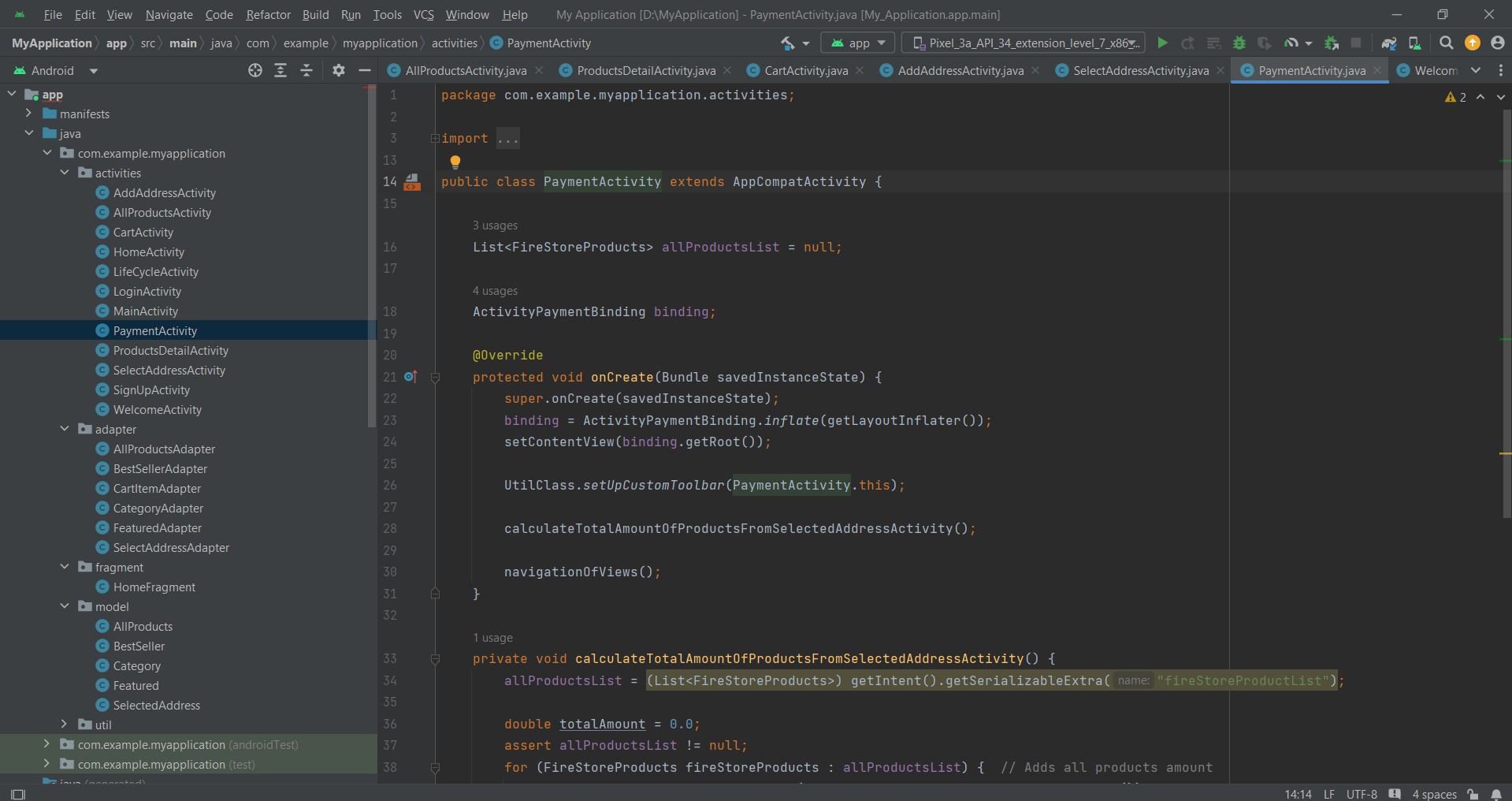
**4.4.10 Select Address Screen**

TheSelect Address screen allows the user to select an address of their own, if used before to confirm their location of delivery. Once selected, they can click the Continue to payment button for the order payment. If the user hasn’t given any address/new user then, the can click the add address button to add their address which is stored in the cloud database for an easy user interaction. The main purpose of this screen is to allow the user to choose their own location of delivery, for their ease of convenience. It also save time as the user doesn’t have to enter their full address for each purchase if the address is same.

****

**4.4.11 Payment Screen**

ThePayment screen displays the total amount of the current order. It has a pay now button which once the user clicks the payment is made and a toast message is displayed that the transaction is successful. It leads to the payment gateway through which the payment is made.



**5.SOFTWARE DESCRIPTION**

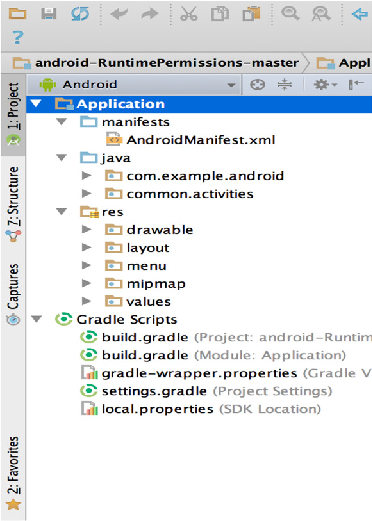
**Android Studio**

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

* A flexible Gradle-based build system
* A fast and feature-rich emulator
* A unified environment where you can develop for all Android devices
* Instant Run to push changes to your running app without building a new APK
* Code templates and GitHub integration to help you build common app features and import sample code
* Extensive testing tools and frameworks
* Lint tools to catch performance, usability, version compatibility, and other problems
* C++ and NDK support.
* Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

This page provides an introduction to basic Android Studio features. For a summary of the latest changes, see Android Studio release notes.

**Project structure**



**Figure 1.** The project files in Android view.

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

* Library modules
* Android app modules
* Google App Engine modules

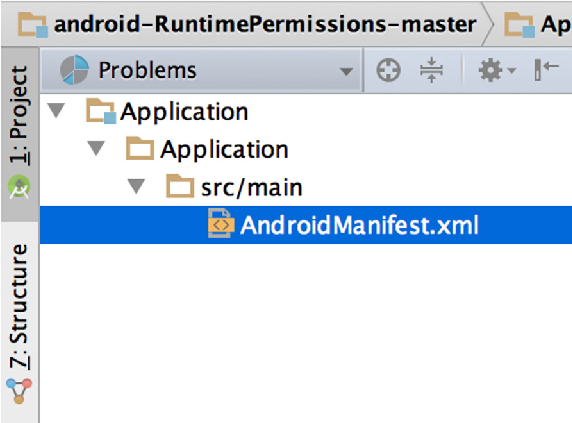
By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

* **Java**: Contains the Java source code files, including JUnit test code.
* **Manifests**: Contains theAndroidManifest.xmlfile.
* **Res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

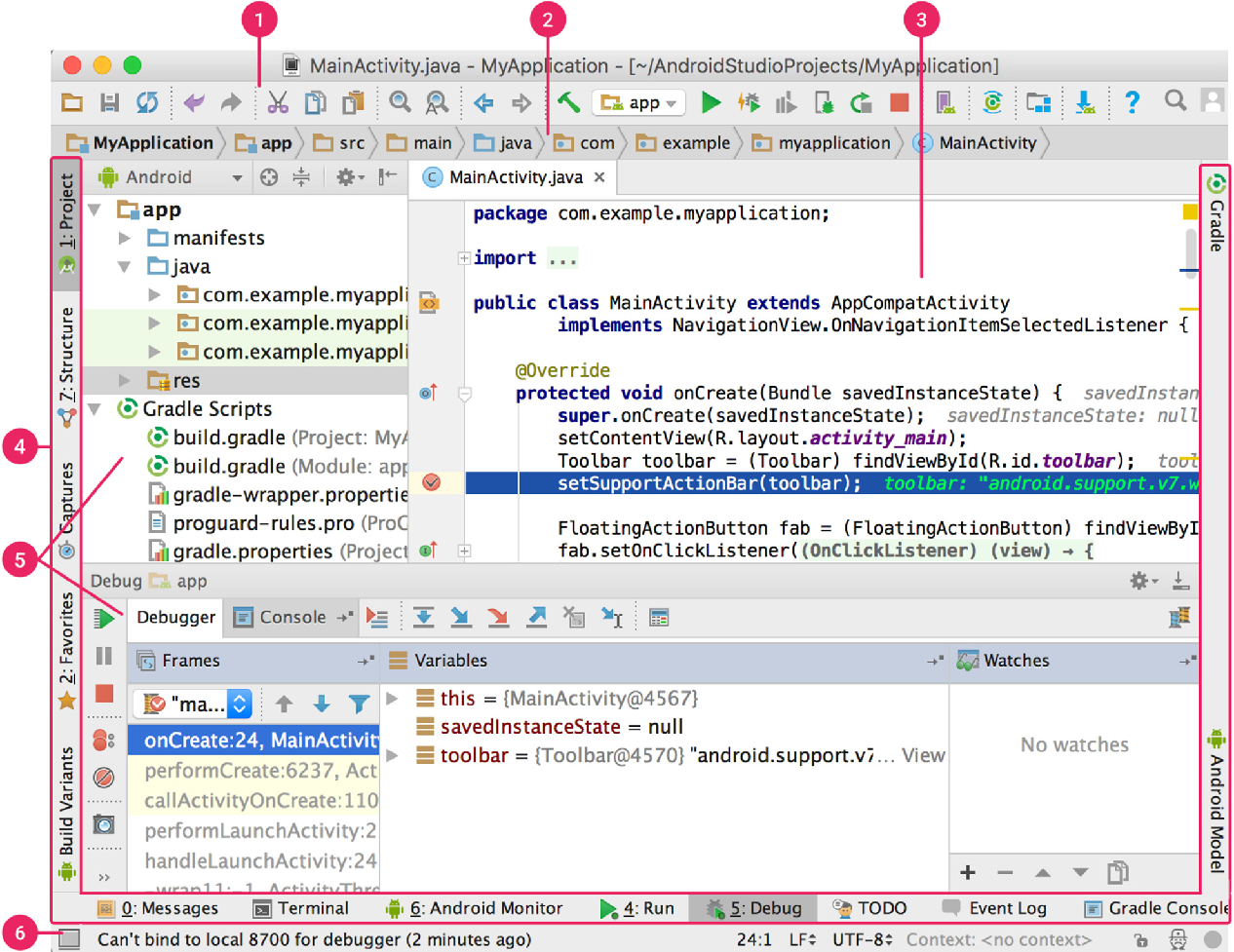
You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.



**Figure 2.** The project files in Problems view, showing a layout file with a problem.

**The user interface**

The Android Studio main window is made up of several logical areas identified in figure 3.



**Figure 3.**The Android Studio main window.

1. The **navigation bar** helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
2. The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
3. The **editor window** is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.
5. The **tool windows** give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
6. The **status bar** displays the status of your project and the IDE itself, as well as any warnings or messages.
7. You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

**Tool windows**

* Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window. To expand or collapse a tool window, click the tool’s name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
* To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
* To show or hide the entire tool window bar, click the window icon  in the bottom left-hand corner of the Android Studio window.
* To locate a specific tool window, hover over the window icon and select the tool window from the menu.
* You can also use keyboard shortcuts to open tool windows. Table 1 lists the shortcuts for the most common windows

|  |  |  |
| --- | --- | --- |
| **TOOLS WINDOW** | **Windows and Linux** | **Mac** |
| Project | Alt+1 | Command+1 |
| Version Control | Alt+9 | Command+9 |
| Run | Shift+F10 | Control+R |
| Debug | Shift+F9 | Control+D |
| Logcat | Alt+6 | Command+6 |
| Return to Editor | Esc | Esc |
| Hide All Tool Windows | Control+Shift+F12 | Command+Shift+F12 |

**Table 1.Keyboard shortcuts for some useful tool windows.**

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables *Distraction Free Mode*. To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use Speed Search to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

For more tips, see Keyboard shortcuts.

**Code completion**

Android Studio has three types of code completion, which you can access using keyboard shortcuts.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **DESCRIPTION** | **WINDOWS AND LINUX** | **MAC** |
| Basic Completion | Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members. | Control+Space | Control+Space |
| Smart Completion | Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice in a row, you see more results, including chains. | Control+Shift+Space | Control+Shift+ Space |
| Statement Completion | Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc. | Control+Shift+Enter | Shift+Command+ Enter |

**Table 2.** Keyboard shortcuts for code completion.

You can also perform quick fixes and show intention actions by pressing **Alt+ Enter**.

For more information about code completion, see Code Completion.

**Find sample code**

The Code Sample Browser in Android Studio helps you find high-quality, Google-provided Android code samples based on the currently highlighted symbol in your project. For more information, see Find sample code.

**Navigation**

Here are some tips to help you move around Android Studio.

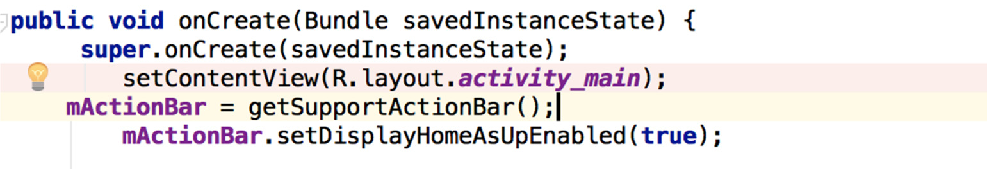
Switch between your recently accessed files using the *Recent Files* action.

* Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
* View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
* Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N** (**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.
* Navigate to a file or folder using the *Navigate to File* action. Bring up the Navigate to File action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
* Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the Navigate to Symbol action by pressing **Control+Shift+Alt+N** (**Command+Option+O** on a Mac).
* Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7** (**Option+F7** on a Mac).

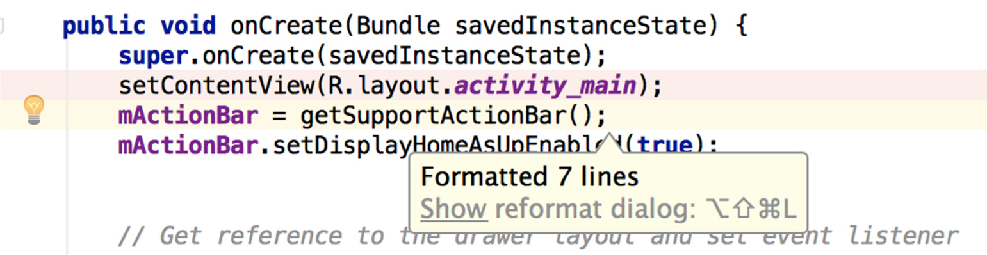
**Style and Formatting**

As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings. You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines. To customize your code style settings, click **File > Settings > Editor > Code Style** (**AndroidStudio > Preferences > Editor > Code Style** on a Mac.)

Although the IDE automatically applies formatting as you work, you can also explicitly call the Reformat Codeaction by pressing **Control+Alt+L** (**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I**(**Control+Option+I** on a Mac).



**Figure 4.** Code before formatting.



**Figure 5.**Code after formatting.

**Version control basics**

Android Studio supports a variety of version control systems (VCS’s), including Git, GitHub, CVS, Mercurial, Subversion, and Google Cloud Source Repositories.

After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

* From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
* From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.
* The VCS menu now displays a number of version control options based on the system you selected.

Note: You can also use the File > Settings > Version Control menu option to set up and modify the version control settings.

**GRADLE BUILD SYSTEM**

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plug-in for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

* Customize, configure, and extend the build process.
* Create multiple APKs for your app, with different features using the same project and modules.
* Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named build.gradle. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plug-in for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

To learn more about the build system and how to configure, see Configure your build.

**Build variants**

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of your

app, or if you want to distribute multiple APKs for different device configurations on Google Play.

For more information about configuring build variants, see Configure build variants.

**Multiple APK support**

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

For more information about multiple APK support, read Build multiple APKs.

**Resource shrinking**

Resource shrinking in Android Studio automatically removes unused resources from your packaged app and library dependencies. For example, if your application is using Google Playservicesto access Google Drive functionality, and you are not currently usingGoogle Sign-In,then resource shrinking can remove the various drawable assets for the SignInButton buttons.

**Note:** Resource shrinking works in conjunction with code shrinking tools, such as ProGuard.

For more information on shrinking code and resources, see Shrink your code and resources.

**Managing dependencies**

Dependencies for your project are specified by name in the build.gradle file. Gradle takes care of finding your dependencies and making them available in your build. You can declare module dependencies, remote binary dependencies, and local binary dependencies inyour build.gradle file. Android Studio configures projects to use the Maven Central Repository by default. (This configuration is included in the top-level build file for the project.) For more information about configuring dependencies, read Add build dependencies.

**Debug and profile tools**

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

**Inline debugging**

Use inline debugging to enhance your code walk-through in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

* Inline variable values.
* Referring objects that reference a selected object
* Method returns values.
* Lambda and operator expressions.
* Tooltip values



**Figure 6.** An inline variable value.

To enable inline debugging, in the **Debug** window, click **Settings** and select the checkbox for **Show Values Inline**.

**Performance profilers**

Android Studio provides performance profilers so you can more easily track your app’s memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyse network requests. With your app running on a device or emulator, open the **AndroidProfiler** tab.

For more information about performance profilers, see Performance profiling tools.

**Heap dump**

When you’re profiling memory usage in Android Studio, you can simultaneously initiate garbage collection and dump the Java heap to a heap snapshot in an Android-specific HPROF binary format file. The HPROF viewer displays classes, instances of each class, and a reference tree to help you track memory usage and find memory leaks.

For more information about working with heap dumps, see Inspect the heap and allocations.

**Memory Profiler**

You can use Memory Profiler to track memory allocation and watch where objects are being allocated when you perform certain actions. Knowing these allocations enables you to optimize your app’s performance and memory use by adjusting the method calls related to those actions.

For information about tracking and analysing allocations, see Inspect the heap and allocations.

**Data file access**

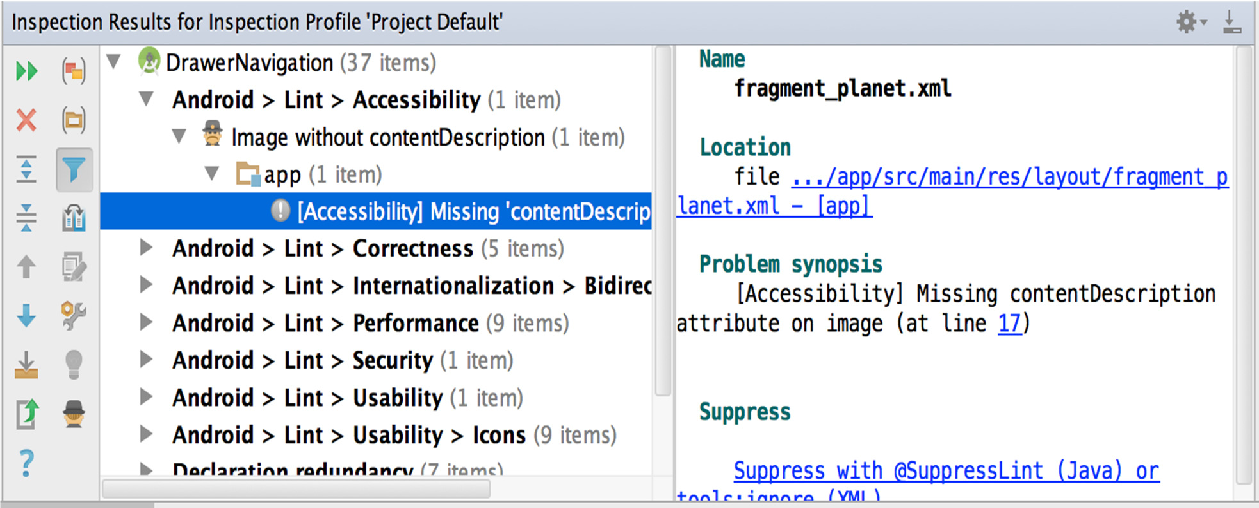
The Android SDK tools, such as Systrace, and logcat, generate performance and debugging data for detailed app analysis.

To view the available generated data files, open the Captures tool window. In the list of the generated files, double-click a file to view the data. Right-click any. hprof files to convert them to the standard investigate your RAM usage file format.

**Code inspections**

Whenever you compile your program, Android Studio automatically runs configured Lint and other IDE inspections to help you easily identify and correct problems with the structural quality of your code.

The Lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.



**Figure 7.** The results of a Lint inspection in Android Studio.

In addition to Lint checks, Android Studio also performs IntelliJ code inspections and validates annotations to streamline your coding workflow.

**Annotations in Android Studio**

Android Studio supports annotations for variables, parameters, and return values to help you catch bugs, such as null pointer exceptions and resource type conflicts. The Android SDK Manager packages the Support-Annotations library in the Android Support Repository for use with Android Studio. Android Studio validates the configured annotations during code inspection.

**6. SOFTWARE DESIGN**

**6.1. ACTIVITY FLOW DIAGRAM**

**6.1.1. WELCOME SCREEN**

WELCOME SCREEN

LOGIN

SIGNUP

**6.1.2. LOGIN SCREEN**

USER

FIRESTORE AUTHENTICATION

**6.1.3. SIGN UP SCREEN**

USER

FIREBASE AUTHENTICATION

**6.1.4. HOME SCREEN**

FIRESTORE DATABASE

CATEGORIES

FEATURED

MENU

BESTSELLER

SEARCH

SEE ALL

LOG OUT

Phone Cases

CART

Air Pods

Speakers

**6.1.5. GENRE SCREEN**

FIRESTORE DATABASE

CATEGORY

**6.1.6. ALL PRODUCTS SCREEN**

FIRESTORE DATABASE

**6.1.7. PRODUCT DETAIL SCREEN**

FIRESTORE DATABASE

**6.1.8. CART SCREEN**

FIRESTORE DATABASE

**6.1.8. ADDRESS SCREEN**

FIRESTORE DATABASE

ADD ADDRESS

CONTINUE TO PAYMENT

**7.** **SOFTWARE TESTING AND IMPLEMENTATION**

**Software Testing** is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* Meets the requirements that guided its design and development,
* Responds correctly to all kinds of inputs,
* Performs its functions within an acceptable time,
* It is sufficiently usable,
* Can be installed and run in its intended environments, and
* Achieves the general result its stakeholders desire.

**TESTING APPROACH**

### **1. Static vs. Dynamic Testing**

### There are many approaches available in software testing. [Reviews](https://en.wikipedia.org/wiki/Code_review), [walkthroughs](https://en.wikipedia.org/wiki/Software_walkthrough), or [inspections](https://en.wikipedia.org/wiki/Software_inspection) are referred to as static testing, whereas executing programmed code with a given set of [test cases](https://en.wikipedia.org/wiki/Test_case) is referred to as [dynamic testing](https://en.wikipedia.org/wiki/Dynamic_testing)**.**

Static testing is often implicit, like proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as [static program analysis](https://en.wikipedia.org/wiki/Static_program_analysis). Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete [functions](https://en.wikipedia.org/wiki/Function_(computer_science)) or modules. Typical techniques for these are either using [stubs](https://en.wikipedia.org/wiki/Method_stub)/drivers or execution from a [debugger](https://en.wikipedia.org/wiki/Debugger) environment. Static testing involves [verification](https://en.wikipedia.org/wiki/Software_verification), whereas dynamic testing also involves [validation](https://en.wikipedia.org/wiki/Software_validation).

**2. Exploratory Approach**

Exploratory testing is an approach to software testing that is concisely described as simultaneous learning, [test design](https://en.wikipedia.org/wiki/Test_design) and test execution. [Cem Kaner](https://en.wikipedia.org/wiki/Cem_Kaner), who coined the term in 1984, defines exploratory testing as "a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the quality of his/her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project.”

**2.1. The ”Box” approach**

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that the tester takes when designing test cases. A hybrid approach called grey-box testing may also be applied to software testing methodology. With the concept of grey-box testing which develops tests from specific design elements gaining prominence, this "arbitrary distinction" between black- and white-box testing has faded somewhat.

**2.2. White-Box Testing**

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., [in-circuit testing](https://en.wikipedia.org/wiki/In-circuit_test) (ICT).

While white-box testing can be applied at the [unit](https://en.wikipedia.org/wiki/Unit_testing), [integration](https://en.wikipedia.org/wiki/Integration_testing), and [system](https://en.wikipedia.org/wiki/System_testing) levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

**Techniques used in white-box testing include:**

* [API testing](https://en.wikipedia.org/wiki/API_testing) – testing of the application using public and private [APIs](https://en.wikipedia.org/wiki/Application_programming_interfaces) (application programming interfaces)
* [Code coverage](https://en.wikipedia.org/wiki/Code_coverage) – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* [Fault injection](https://en.wikipedia.org/wiki/Fault_injection) methods – intentionally introducing faults to gauge the efficacy of testing strategies
* [Mutation testing](https://en.wikipedia.org/wiki/Mutation_testing) methods
* [Static testing](https://en.wikipedia.org/wiki/Static_testing) methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important [function points](https://en.wikipedia.org/wiki/Function_points) have been tested. Code coverage as a [software metric](https://en.wikipedia.org/wiki/Software_metric) can be reported as a percentage for:

* Function coverage, which reports on functions executed
* Statement coverage, which reports on the number of lines executed to complete the test
* Decision coverage, which reports on whether both the True and the False branch of a given test has been executed

100% statement coverage ensures that all code paths or branches (in terms of [control flow](https://en.wikipedia.org/wiki/Control_flow)) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly. Pseudo-tested functions and methods are those that are covered but not specified (it is possible to remove their body without breaking any test case).

**Black Box Testing**

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: [equivalence partitioning](https://en.wikipedia.org/wiki/Equivalence_partitioning), [boundary value analysis](https://en.wikipedia.org/wiki/Boundary_value_analysis), [all-pairs testing](https://en.wikipedia.org/wiki/All-pairs_testing), [state transition tables](https://en.wikipedia.org/wiki/State_transition_table), [decision table](https://en.wikipedia.org/wiki/Decision_table) testing, [fuzz testing](https://en.wikipedia.org/wiki/Fuzz_testing), [model-based testing](https://en.wikipedia.org/wiki/Model-based_testing), [use case](https://en.wikipedia.org/wiki/Use_case) testing, [exploratory testing](https://en.wikipedia.org/wiki/Exploratory_testing), and specification-based testing to Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough [test cases](https://en.wikipedia.org/wiki/Test_case) to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be [functional](https://en.wikipedia.org/wiki/Functional_testing) or [non-functional](https://en.wikipedia.org/wiki/Non-functional_testing), though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases check something that could have been tested by only one test case or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing:

* Unit,
* Integration,
* System,
* Acceptance.

It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

**Component Interface Testing**

Component interface testing is a variation of [black-box testing](https://en.wikipedia.org/wiki/Black-box_testing), with the focus on the data values beyond just the related actions of a subsystem component. The practice of component interface testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units. The data being passed can be considered as "message packets" and the range or data types can be checked, for data generated from one unit, and tested for validity before being passed into another unit. One option for interface testing is to keep a separate log file of data items being passed, often with a timestamp logged to allow analysis of thousands of cases of data passed between units for days or weeks. Tests can include checking the handling of some extreme data values while other interface variables are passed as normal values. Unusual data values in an interface can help explain unexpected performance in the next unit.

**Visual Testing**

The aim of visual testing is to provide developers with the ability to examine what was happening at the point of software failure by presenting the data in such a way that the developer can easily find the information she or he requires, and the information is expressed clearly.

At the core of visual testing is the idea that showing someone a problem (or a test failure), rather than just describing it, greatly increases clarity and understanding. Visual testing, therefore, requires the recording of the entire test process – capturing everything that occurs on the test system in video format. Output videos are supplemented by real-time tester input via picture-in-a-picture webcam and audio commentary from microphones.

Visual testing provides a number of advantages. The quality of communication is increased drastically because testers can show the problem (and the events leading up to it) to the developer as opposed to just describing it and the need to replicate test failures will cease to exist in many cases. The developer will have all the evidence she or he requires of a test failure and can instead focus on the cause of the fault and how it should be fixed.

[Ad hoc testing](https://en.wikipedia.org/wiki/Ad_hoc_testing) and [exploratory testing](https://en.wikipedia.org/wiki/Exploratory_testing) are important methodologies for checking software integrity, because they require less preparation time to implement, while the important bugs can be found quickly. In ad hoc testing, where testing takes place in an improvised, impromptu way, the ability of the tester(s) to base testing off documented methods and then improvise variations of those tests can result in more rigorous examination of defect fixes. However, unless strict documentation of the procedures is maintained, one of the limits of ad hoc testing is lack of repeatability.

**Grey Box Testing**

Grey-box testing (American spelling: grey box testing) involves having knowledge of internal data structures and algorithms for purposes of designing tests while executing those tests at the user, or black-box level. The tester will often have access to both "the source code and the executable binary." Grey-box testing may also include [reverse engineering](https://en.wikipedia.org/wiki/Reverse_coding) (using dynamic code analysis) to determine, for instance, boundary values or error messages. Manipulating input data and formatting output do not qualify as grey-box, as the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting [integration testing](https://en.wikipedia.org/wiki/Integration_testing) between two modules of code written by two different developers, where only the interfaces are exposed for the test.

By knowing the underlying concepts of how the software works, the tester makes better-informed testing choices while testing the software from outside. Typically, a grey-box tester will be permitted to set up an isolated testing environment with activities such as seeding a [database](https://en.wikipedia.org/wiki/Database). The tester can observe the state of the product being tested after performing certain actions such as executing [SQL](https://en.wikipedia.org/wiki/SQL) statements against the database and then executing queries to ensure that the expected changes have been reflected. Grey-box testing implements intelligent test scenarios, based on limited information. This will particularly apply to data type handling, [exception handling](https://en.wikipedia.org/wiki/Exception_handling), and so on.

**TESTING LEVELS**

Broadly speaking, there are at least three levels of testing: unit testing, integration testing, and system testing. However, a fourth level, acceptance testing, may be included by developers. This may be in the form of operational acceptance testing or be simple end-user (beta) testing, testing to ensure the software meets functional expectations. Tests are frequently grouped into one of these levels by where they are added in the software development process, or by the level of specificity of the test.

**Unit Testing**

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch [corner cases](https://en.wikipedia.org/wiki/Corner_case) or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development life cycle. Unit testing aims to eliminate construction errors before code is promoted to additional testing; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

Depending on the organization's expectations for software development, unit testing might include [static code analysis](https://en.wikipedia.org/wiki/Static_code_analysis), [data-flow analysis](https://en.wikipedia.org/wiki/Data-flow_analysis), metrics analysis, peer code reviews, [code coverage](https://en.wikipedia.org/wiki/Code_coverage) analysis and other software testing practices.

**Integration Testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

**System Testing**

System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a login interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

**Operational Acceptance Testing**

Operational acceptance is used to conduct operational readiness (pre-release) of a product, service or system as part of a [quality management system](https://en.wikipedia.org/wiki/Quality_management_system). OAT is a common type of non-functional software testing, used mainly in [software development](https://en.wikipedia.org/wiki/Software_development) and [software maintenance](https://en.wikipedia.org/wiki/Software_maintenance) projects. This type of testing focuses on the [operational readiness](https://en.wikipedia.org/w/index.php?title=Operational_readiness&action=edit&redlink=1) of the system to be supported, or to become part of the production environment. Hence, it is also known as operational readiness testing (ORT) or [Operations readiness and assurance](https://en.wikipedia.org/wiki/Operations_readiness_and_assurance) (OR&A) testing. [Functional testing](https://en.wikipedia.org/wiki/Functional_testing) within OAT is limited to those tests that are required to verify the *non-functional* aspects of the system.

In addition, the software testing should ensure that the portability of the system, as well as working as expected, does not also damage or partially corrupt its operating environment or cause other processes within that environment to become inoperative.

**TESTING TYPES, TECHNIQUES AND TACTICS**

**Installation Testing**

Most software systems have installation procedures that are needed before they can be used for their main purpose. Testing these procedures to achieve an installed software system that may be used is known as installation testing.

**Compatibility Testing**

A common cause of software failure (real or perceived) is a lack of its [compatibility](https://en.wikipedia.org/wiki/Computer_compatibility) with other [application software](https://en.wikipedia.org/wiki/Application_software), [operating systems](https://en.wikipedia.org/wiki/Operating_system) (or operating system [versions](https://en.wikipedia.org/wiki/Software_versioning), old or new), or target environments that differ greatly from the original (such as a [terminal](https://en.wikipedia.org/wiki/Computer_terminal) or [GUI](https://en.wikipedia.org/wiki/GUI) application intended to be run on the [desktop](https://en.wikipedia.org/wiki/Desktop_metaphor) now being required to become a [Web application](https://en.wikipedia.org/wiki/Web_application), which must render in a [Web browser](https://en.wikipedia.org/wiki/Web_browser)). For example, in the case of a lack of [backward compatibility](https://en.wikipedia.org/wiki/Backward_compatibility), this can occur because the programmers develop and test software only on the latest version of the target environment, which not all users may be running. These results in the unintended consequence that the latest work may not function on earlier versions of the target environment or on older hardware that earlier version of the target environment was capable of using. Sometimes such issues can be fixed by proactively [abstracting](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) operating system functionality into a separate program [module](https://en.wikipedia.org/wiki/Modular_programming) or [library](https://en.wikipedia.org/wiki/Library_(computing)).

**Smoke and Sanity Testing**

[Sanity testing](https://en.wikipedia.org/wiki/Sanity_testing) determines whether it is reasonable to proceed with further testing.

[Smoke testing](https://en.wikipedia.org/wiki/Smoke_testing_(software)) consists of minimal attempts to operate the software, designed to determine whether there are any basic problems that will prevent it from working at all. Such tests can be used as [build verification test](https://en.wikipedia.org/wiki/Build_verification_test).

**Regression Testing**

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover [software regressions](https://en.wikipedia.org/wiki/Software_regression), as degraded or lost features, including old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly, stops working as intended. Typically, regressions occur as an [unintended consequence](https://en.wikipedia.org/wiki/Unintended_consequence) of program changes, when the newly developed part of the software collides with the previously existing code. Regression testing is typically the largest test effort in commercial software development, due to checking numerous details in prior software features, and even new software can be developed while using some old test cases to test parts of the new design to ensure prior functionality is still supported.

Common methods of regression testing include re-running previous sets of test cases and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the [risk](https://en.wikipedia.org/wiki/Risk_management) of the added features. They can either be complete, for changes added late in the release or deemed to be risky, or be very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk. In regression testing, it is important to have strong assertions on the existing behaviour. For this, it is possible to generate and add new assertions in existing test cases, this is known as automatic test amplification.

**Acceptance Testing**

Acceptance testing can mean one of two things:

1. A [smoke test](https://en.wikipedia.org/wiki/Smoke_testing_(software)) is used as a build acceptance test prior to further testing, e.g., before [integration](https://en.wikipedia.org/wiki/Integration_testing) or [regression](https://en.wikipedia.org/wiki/Regression_testing).
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as [user acceptance testing](https://en.wikipedia.org/wiki/User_acceptance_testing) (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.

**Alpha Testing**

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing before the software goes to beta testing.

**Beta Testing**

Beta testing comes after alpha testing and can be considered a form of external [user acceptance testing](https://en.wikipedia.org/wiki/User_acceptance_testing). Versions of the software, known as [beta versions](https://en.wikipedia.org/wiki/Beta_version), are released to a limited audience outside of the programming team known as beta testers. The software is released to groups of people so that further testing can ensure the product has few faults or [bugs](https://en.wikipedia.org/wiki/Computer_bug). Beta versions can be made available to the open public to increase the [feedback](https://en.wikipedia.org/wiki/Feedback#In_organizations) field to a maximal number of future users and to deliver value earlier, for an extended or even indefinite period of time ([perpetual beta](https://en.wikipedia.org/wiki/Perpetual_beta)).

* + - * 1. **8**. **APPENDICES**

**8.1 SOURCE CODE**

**activity\_splash**:

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:background="@color/red"  
 android:gravity="center"  
 tools:context=".activity.SplashActivity">  
  
 <ImageView  
 android:layout\_width="245dp"  
 android:layout\_height="295dp"  
 android:padding="3dp"  
 android:paddingEnd="1dp"  
 android:src="@drawable/mobhub \_logo">  
  
 </ImageView>  
</LinearLayout>

**SplashActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
  
import com.example.spellbound.R;  
  
public class SplashActivity extends AppCompatActivity {  
  
 static final int *TIME\_LIMIT*=2000;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_splash*);  
 new Handler().postDelayed(()->{  
 Intent intent=new Intent(SplashActivity.this, WelcomeActivity.class);  
 startActivity(intent);  
 finish();  
 },*TIME\_LIMIT*);  
 }  
}

**activity\_welcome:**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:orientation="vertical"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/welcome\_pg\_2"  
 android:gravity="center"  
 tools:context=".activity.WelcomeActivity">  
  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="serif"  
 android:text="Welcome To SpellBound"  
 android:textColor="@color/white"  
 android:textSize="24sp"  
 android:textStyle="bold" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:fontFamily="serif"  
 android:text="Get the Best Offers"  
 android:textColor="@color/white"  
 android:textSize="20sp"  
 android:textStyle="bold" />  
  
 <ImageView  
 android:layout\_width="200dp"  
 android:layout\_height="300dp"  
 android:layout\_marginTop="40dp"  
 android:layout\_marginBottom="40dp"  
 android:background="@drawable/welcome\_img\_bg"  
 android:src="@drawable/spellbound\_logo" />  
  
 <Button  
 android:id="@+id/btn\_login"  
 android:layout\_width="340dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginEnd="20dp"  
 android:backgroundTint="@color/dark\_blue"  
 android:textColor="@color/white"  
 android:fontFamily="serif"  
 android:textStyle="bold"  
 android:text="LOGIN"  
 android:textSize="14sp"  
 app:cornerRadius="2dp" />  
  
  
 <Button  
 android:id="@+id/btn\_sign\_up"  
 android:layout\_width="340dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginEnd="20dp"  
 android:fontFamily="serif"  
 android:textStyle="bold"  
 android:textColor="@color/white"  
 android:backgroundTint="@color/dark\_blue"  
 android:text="SIGN UP"  
 android:textSize="14sp"  
 app:cornerRadius="2dp" />  
  
</LinearLayout>

**WelcomeActivity:**

package com.example.mobhub.activity;  
  
import static com.example.spellbound.UtilClass.*screenNavigation*;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.Button;  
  
import com.example.spellbound.R;  
import com.google.firebase.auth.FirebaseAuth;  
  
public class WelcomeActivity extends AppCompatActivity {  
  
 Button loginBtn,signUpBtn;  
 FirebaseAuth firebaseauth;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_welcome*);  
 firebaseauth = FirebaseAuth.*getInstance*();  
  
 loginBtn=findViewById(R.id.*btn\_login*);  
 signUpBtn=findViewById(R.id.*btn\_sign\_up*);  
 navigationOfViews();  
 }  
  
 private void navigationOfViews() {  
 loginBtn.setOnClickListener(v->{  
 *screenNavigation*(WelcomeActivity.this, LoginActivity.class);  
 finish();  
 });  
 signUpBtn.setOnClickListener(v->{  
 *screenNavigation*(WelcomeActivity.this, SignUpActivity.class);  
 finish();  
 });  
 }  
 protected void onStart() {  
 super.onStart();  
 if(firebaseauth.getCurrentUser()!=null) {  
 *screenNavigation*(WelcomeActivity.this, HomeActivity.class);  
 finish();  
 }  
 }  
}

**activity\_login**:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/login\_page"  
 tools:context=".activity.LoginActivity">  
  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline17"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_begin="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_end="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_end="20dp" />  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="352dp"  
 android:layout\_height="68dp"  
 android:layout\_marginTop="8dp"  
 android:minHeight="?attr/actionBarSize"  
 android:theme="?attr/actionBarTheme"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline"  
 app:layout\_constraintHorizontal\_bias="0.45"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline17"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline2" />  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="52dp"  
 android:text="Login Screen"  
 android:fontFamily="serif"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:textSize="28sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/toolbar"  
 app:layout\_constraintStart\_toStartOf="@+id/toolbar"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout"  
 android:layout\_width="311dp"  
 android:layout\_height="64dp"  
 android:layout\_marginTop="44dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView"  
 app:layout\_constraintStart\_toStartOf="@+id/textView"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_email"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email"  
 android:inputType="textEmailAddress" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout2"  
 android:layout\_width="311dp"  
 android:layout\_height="64dp"  
 android:layout\_marginTop="44dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout"  
 app:passwordToggleEnabled="true">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_password"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:inputType="textPassword" />  
 </com.google.android.material.textfield.TextInputLayout>

**Login\_Activity:**

package com.example.mobhub.activity;  
  
import static com.example.spellbound.UtilClass.*screenNavigation*;  
import static com.example.spellbound.UtilClass.*snackBar*;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Toast;  
  
import com.example.spellbound.databinding.ActivityLoginBinding;  
import com.google.firebase.auth.FirebaseAuth;  
  
import java.util.Objects;  
  
public class LoginActivity extends AppCompatActivity {  
 ActivityLoginBinding binding;  
 FirebaseAuth firebaseauth;  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityLoginBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
 firebaseauth=FirebaseAuth.*getInstance*();  
 navigationOfView();  
 }  
 private void navigationOfView() {  
 binding.btnLogin.setOnClickListener(v -> {  
 checkLoginValidation(v);  
 });  
 binding.tvSignUp.setOnClickListener(v -> {  
 *screenNavigation*(LoginActivity.this, SignUpActivity.class);  
 finish();  
 });  
 }  
  
 private void checkLoginValidation(View view) {  
 String userEmail= Objects.*requireNonNull*(binding.etEmail.getText()).toString();  
 String userPassword= Objects.*requireNonNull*(binding.etPassword.getText()).toString();  
 if(!userEmail.isEmpty() && !userPassword.isEmpty())  
 {  
 firebaseauth.signInWithEmailAndPassword(userEmail,userPassword).addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 Toast.*makeText*(this, "Logged In Successfully", Toast.*LENGTH\_SHORT*).show();  
 *screenNavigation*(LoginActivity.this, HomeActivity.class);  
 finish();  
 } else if(userPassword.length()<6){  
 *snackBar*(view,"Password must be at least 6 characters");  
 }  
 else {  
  
 Toast.*makeText*(this, "Failed to Login", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 else {  
 Toast.*makeText*(this, "Please fill Email and Password", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
}

**activity\_signup:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/sign\_up\_page"  
 tools:context=".activity.SignUpActivity">  
  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_begin="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline7"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_end="20dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline8"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_end="20dp" />  
  
 <androidx.appcompat.widget.Toolbar  
 android:id="@+id/toolbar2"  
 android:layout\_width="368dp"  
 android:layout\_height="76dp"  
 android:minHeight="?attr/actionBarSize"  
 android:theme="?attr/actionBarTheme"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline8"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline5"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline6" />  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:text="Sign Up"  
 android:textColor="@color/black"  
 android:textSize="28sp"  
 android:fontFamily="serif"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/toolbar2"  
 app:layout\_constraintHorizontal\_bias="0.496"  
 app:layout\_constraintStart\_toStartOf="@+id/toolbar2"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar2" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout3"  
 android:layout\_width="310dp"  
 android:layout\_height="69dp"  
 android:layout\_marginTop="32dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView4"  
 app:layout\_constraintStart\_toStartOf="@+id/textView4"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView4">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Name"  
 android:inputType="textPersonName" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout4"  
 android:layout\_width="308dp"  
 android:layout\_height="63dp"  
 android:layout\_marginTop="32dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout3"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout3"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout3">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_email"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Email"  
 android:inputType="textEmailAddress" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout5"  
 android:layout\_width="302dp"  
 android:layout\_height="60dp"  
 android:layout\_marginTop="32dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout4"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout4"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout4"  
 app:passwordToggleEnabled="true">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_password"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:inputType="textPassword" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <Button  
 android:id="@+id/btn\_sign\_up"  
 android:layout\_width="321dp"  
 android:layout\_height="54dp"  
 android:layout\_marginTop="36dp"  
 android:backgroundTint="@color/sage"  
 android:text="SIGN UP"  
 android:fontFamily="serif"  
 android:textStyle="bold"  
 android:textSize="16sp"  
 app:cornerRadius="2dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout5"  
 app:layout\_constraintHorizontal\_bias="0.571"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout5"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout5" />  
  
 <TextView  
 android:id="@+id/textView5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginTop="28dp"  
 android:fontFamily="serif"  
 android:text="Already have an Account?"  
 android:textColor="@color/black"  
 android:textSize="17sp"  
 app:layout\_constraintStart\_toStartOf="@+id/btn\_sign\_up"  
 app:layout\_constraintTop\_toBottomOf="@+id/btn\_sign\_up" />

<TextView  
 android:id="@+id/tv\_login"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="28dp"  
 android:text="Log In"  
 android:textColor="@color/black"  
 android:textSize="17sp"  
 android:fontFamily="serif"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline8"  
 app:layout\_constraintHorizontal\_bias="0.295"  
 app:layout\_constraintStart\_toEndOf="@+id/textView5"  
 app:layout\_constraintTop\_toBottomOf="@+id/btn\_sign\_up" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**SignupActivity:**

package com.example.spellbound.activity;  
  
import static com.example.spellbound.UtilClass.*screenNavigation*;  
import static com.example.spellbound.UtilClass.*snackBar*;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Toast;  
  
import com.example.spellbound.databinding.ActivitySignUpBinding;  
import com.google.firebase.auth.FirebaseAuth;  
  
import java.util.Objects;  
  
public class SignUpActivity extends AppCompatActivity {  
  
 ActivitySignUpBinding binding;  
 private FirebaseAuth mAuth;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivitySignUpBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
 // Initialize Firebase Auth  
 mAuth = FirebaseAuth.*getInstance*();  
 navigationOfViews();  
 }  
 private void navigationOfViews() {  
 binding.btnSignUp.setOnClickListener(v -> {  
 signUpValidation(v);  
 });  
 binding.tvLogin.setOnClickListener(v -> {  
 *screenNavigation*(SignUpActivity.this, LoginActivity.class);  
 finish();  
 });  
  
 }  
 private void signUpValidation(View view){  
 String userName= Objects.*requireNonNull*(binding.etName.getText()).toString();  
 String userEmail= Objects.*requireNonNull*(binding.etEmail.getText()).toString();  
 String userPassword= Objects.*requireNonNull*(binding.etPassword.getText()).toString();  
 if(!userName.isEmpty() && !userEmail.isEmpty() && !userPassword.isEmpty())  
 {  
 mAuth.createUserWithEmailAndPassword(userEmail,userPassword).addOnCompleteListener

(task -> {  
 if(task.isSuccessful()) {  
 Toast.*makeText*(this, "Registered Successfully", Toast.*LENGTH\_SHORT*).show();  
 *screenNavigation*(SignUpActivity.this, HomeActivity.class);  
 finish();  
 } else if(userPassword.length()<6){  
 *snackBar*(view,"Password must be at least 6 characters");  
 }  
 else {  
 Toast.*makeText*(this, "Registration Failed", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 else {  
 *snackBar*(view,"Please fill all the details");  
 }  
 }  
}

**rv\_categories\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/cl\_category\_item\_row"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <ImageView  
 android:id="@+id/iv\_category"  
 android:layout\_width="250dp"  
 android:layout\_height="150dp"  
 android:layout\_margin="6dp"  
 android:background="@drawable/sharp\_corners\_views"  
 android:padding="5dp"  
 android:src="@drawable/ic\_launcher\_background"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**Category:**

package com.example.spellbound.model;  
  
import android.os.Parcel;  
import android.os.Parcelable;  
  
import androidx.annotation.NonNull;  
  
  
public class Category {  
 private String categoryImageUrl;  
 private String categoryType;  
  
 public Category() {  
 }  
  
 public String getCategoryImageUrl() {  
  
 return categoryImageUrl;  
 }  
  
 public void setCategoryImageUrl(String categoryImageUrl) {  
 this.categoryImageUrl = categoryImageUrl;  
 }  
  
 public String getCategoryType() {  
  
 return categoryType;  
 }  
  
 public void setCategoryType(String categoryType) {  
  
 this.categoryType = categoryType;  
 }  
  
}

**CategoryAdapter:**

package com.example.spellbound.adapter;  
  
import static com.example.spellbound.UtilClass.*screenNavigationWithDataPassing*;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
  
import androidx.annotation.NonNull;  
import androidx.constraintlayout.widget.ConstraintLayout;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.activity.GenreActivity;  
import com.example.spellbound.model.Category;  
  
import java.util.List;  
  
public class CategoryAdapter extends RecyclerView.Adapter<CategoryAdapter.CategoryViewHolder> {  
  
 Context context;  
 List<Category> categoryList;  
  
 public CategoryAdapter(Context context, List<Category> categoryList) {  
 this.context = context;  
 this.categoryList = categoryList;  
 }  
  
  
 @NonNull  
 @Override  
 public CategoryViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view= LayoutInflater.*from*(context)  
 .inflate(R.layout.*rv\_categories\_item\_row*,parent,false);  
 return new CategoryViewHolder(view);  
  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull CategoryAdapter.CategoryViewHolder holder, int position) {  
 Glide.*with*(context).load(categoryList.get(position).getCategoryImageUrl())  
 .into(holder.categoryImage);  
 holder.categoryImage.setOnClickListener(v -> {  
 *screenNavigationWithDataPassing*(context, GenreActivity.class,"categoryType",  
 categoryList.get(position).getCategoryType());  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
  
 return categoryList.size();  
 }  
  
 static class CategoryViewHolder extends RecyclerView.ViewHolder {  
 *// ConstraintLayout categoryClItemRow;* ImageView categoryImage;  
 public CategoryViewHolder(@NonNull View itemView)  
 {  
 super(itemView);  
 categoryImage=itemView.findViewById(R.id.*iv\_category*);  
 *// categoryClItemRow = itemView.findViewById(R.id.cl\_category\_item\_row);* }  
 }  
}

**rv\_featured\_item\_row:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/sharp\_corners\_views"  
 android:id="@+id/cl\_featured\_item\_row"  
 android:padding="1dp"  
 android:layout\_margin="4dp">  
  
 <ImageView  
 android:id="@+id/iv\_featured"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_marginTop="4dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <TextView  
 android:id="@+id/tv\_featured\_product\_price"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="4dp"  
 android:text="700"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 android:textColor="@color/white"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/iv\_featured" />  
  
 <TextView  
 android:id="@+id/tv\_featured\_product\_name"  
 android:layout\_width="160dp"  
 android:layout\_height="wrap\_content"  
 android:text="Harry Potter"  
 android:textColor="@color/white"  
 android:textSize="16sp"  
 android:maxLines="1"  
 android:ellipsize="end"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_featured\_product\_price"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_featured\_product\_price"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_featured\_product\_price" />  
  
 <TextView  
 android:id="@+id/tv\_featured\_author\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="25dp"  
 android:ellipsize="end"  
 android:maxLines="1"  
 android:text="Sarah J Mass"  
 android:textColor="@color/white"  
 android:textSize="16sp"  
 android:textStyle="italic"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_featured\_product\_name"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_featured\_product\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_featured\_product\_name" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**Featured:**

package com.example.spellbound.model;  
  
import android.os.Parcel;  
import android.os.Parcelable;  
  
import androidx.annotation.NonNull;  
  
import com.example.spellbound.FirestoreProducts;  
  
public class Featured implements Parcelable, FirestoreProducts {  
 private String featuredProductName;  
 private String featuredProductPrice;  
 private String featuredProductImageUrl;  
 private String featuredProductId;  
 private String featuredProductDescription;  
 private float featuredProductRating;  
 private String featuredProductQty;  
 private String featuredAuthorName;  
 private String featuredProductPublished;  
 private String featuredProductFormat;  
 private String featuredProductISBN;  
  
 public Featured() {  
 }  
  
 protected Featured(Parcel in) {  
 featuredProductName = in.readString();  
 featuredProductPrice = in.readString();  
 featuredProductImageUrl = in.readString();  
 featuredProductId = in.readString();  
 featuredProductDescription = in.readString();  
 featuredProductRating = in.readFloat();  
 featuredProductQty = in.readString();  
 featuredAuthorName = in.readString();  
 featuredProductPublished = in.readString();  
 featuredProductFormat = in.readString();  
 featuredProductISBN = in.readString();  
 }  
  
 public static final Creator<Featured> *CREATOR* = new Creator<Featured>() {  
 @Override  
 public Featured createFromParcel(Parcel in) {  
 return new Featured(in);  
 }  
  
 @Override  
 public Featured[] newArray(int size) {  
 return new Featured[size];  
 }  
 };  
  
 public String getFeaturedProductName() {  
 return featuredProductName;  
 }  
  
 public void setFeaturedProductName(String featuredProductName) {  
 this.featuredProductName = featuredProductName;  
 }  
  
 public String getFeaturedProductPrice() {  
 return featuredProductPrice;  
 }  
  
 public void setFeaturedProductPrice(String featuredProductPrice) {  
 this.featuredProductPrice = featuredProductPrice;  
 }  
  
 public String getFeaturedProductImageUrl() {  
 return featuredProductImageUrl;  
 }  
  
 public void setFeaturedProductImageUrl(String featuredProductImageUrl) {  
 this.featuredProductImageUrl = featuredProductImageUrl;  
 }  
  
 public String getFeaturedProductId() {  
 return featuredProductId;  
 }  
  
 public void setFeaturedProductId(String featuredProductId) {  
 this.featuredProductId = featuredProductId;  
 }  
  
 public String getFeaturedProductDescription() {  
 return featuredProductDescription;  
 }  
  
 public void setFeaturedProductDescription(String featuredProductDescription) {  
 this.featuredProductDescription = featuredProductDescription;  
 }  
  
 public float getFeaturedProductRating() {  
 return featuredProductRating;  
 }  
  
 public void setFeaturedProductRating(float featuredProductRating) {  
 this.featuredProductRating = featuredProductRating;  
 }  
  
 public String getFeaturedProductQty() {  
 return featuredProductQty;  
 }  
  
 public void setFeaturedProductQty(String featuredProductQty) {  
 this.featuredProductQty = featuredProductQty;  
 }  
  
 public String getFeaturedAuthorName() {  
 return featuredAuthorName;  
 }  
  
 public void setFeaturedAuthorName(String featuredAuthorName) {  
 this.featuredAuthorName = featuredAuthorName;  
 }  
  
 public String getFeaturedProductPublished() {  
 return featuredProductPublished;  
 }  
  
 public void setFeaturedProductPublished(String featuredProductPublished) {  
 this.featuredProductPublished = featuredProductPublished;  
 }  
  
 public String getFeaturedProductFormat() {  
 return featuredProductFormat;  
 }  
  
 public void setFeaturedProductFormat(String featuredProductFormat) {  
 this.featuredProductFormat = featuredProductFormat;  
 }  
  
 public String getFeaturedProductISBN() {  
 return featuredProductISBN;  
 }  
  
 public void setFeaturedProductISBN(String featuredProductISBN) {  
 this.featuredProductISBN = featuredProductISBN;  
 }  
  
 @Override  
 public int describeContents() {  
 return 0;  
 }  
  
 @Override  
 public void writeToParcel(@NonNull Parcel dest, int flags) {  
 dest.writeString(featuredProductName);  
 dest.writeString(featuredProductPrice);  
 dest.writeString(featuredProductImageUrl);  
 dest.writeString(featuredProductId);  
 dest.writeString(featuredProductDescription);  
 dest.writeFloat(featuredProductRating);  
 dest.writeString(featuredProductQty);  
 dest.writeString(featuredAuthorName);  
 dest.writeString(featuredProductPublished);  
 dest.writeString(featuredProductFormat);  
 dest.writeString(featuredProductISBN);  
 }  
  
 @Override  
 public String getFirestoreProductId() {  
 return featuredProductId;  
 }  
  
 @Override  
 public void setFirestoreProductId(String setId) {  
 this.featuredProductId=setId;  
  
 }  
  
 @Override  
 public String getFirestoreProductPrice() {  
 return featuredProductPrice;  
 }  
  
 @Override  
 public String getFirestoreProductImageUrl() {  
 return featuredProductImageUrl;  
 }  
  
 @Override  
 public String getFirestoreProductName() {  
 return featuredProductName;  
 }  
  
 @Override  
 public String getFirestoreProductQty() {  
 return featuredProductQty;  
 }  
}

**FeaturedAdapter:**

package com.example.spellbound.adapter;  
  
import static com.example.spellbound.UtilClass.*screenNavigationWithDataPassing*;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.constraintlayout.widget.ConstraintLayout;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.activity.ProductDetailActivity;  
import com.example.spellbound.model.Featured;  
  
import java.util.List;  
  
public class FeaturedAdapter extends RecyclerView.Adapter<FeaturedAdapter.FeaturedViewHolder>{  
  
 Context context;  
 List<Featured> featuredList;  
  
 public FeaturedAdapter(Context context, List<Featured> featuredList) {  
 this.context = context;  
 this.featuredList = featuredList;  
 }  
  
 @NonNull  
 @Override  
 public FeaturedAdapter.FeaturedViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view= LayoutInflater.*from*(context)  
 .inflate(R.layout.*rv\_featured\_item\_row*,parent,false);  
 return new FeaturedViewHolder(view);  
  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull FeaturedViewHolder holder, int position) {  
 Glide.*with*(context).load(featuredList.get(position)  
 .getFeaturedProductImageUrl()).into(holder.featuredImage);  
 String strPrice = featuredList.get(position).getFeaturedProductPrice();  
 String priceWithSymbol = holder.itemView.getContext().getString(R.string.*rupee\_sign*, strPrice);  
 *// holder.featuredProductPrice.setText(featuredList.get(position).getFeaturedProductPrice());* holder.featuredProductPrice.setText(priceWithSymbol);  
 holder.featuredProductName.setText(featuredList.get(position).getFeaturedProductName());  
 holder.featuredAuthorName.setText(featuredList.get(position).getFeaturedAuthorName());  
 holder.featuredClItemRow.setOnClickListener(v -> {  
 *screenNavigationWithDataPassing*(context, ProductDetailActivity.class,  
 "productDetails",featuredList.get(position));  
 });  
  
 }  
  
 @Override  
 public int getItemCount() {  
  
 return featuredList.size();  
 }  
  
 static class FeaturedViewHolder extends RecyclerView.ViewHolder {  
 ConstraintLayout featuredClItemRow;  
 ImageView featuredImage;  
 TextView featuredProductPrice,featuredProductName,featuredAuthorName;  
 public FeaturedViewHolder(@NonNull View itemView) {  
 super(itemView);  
 featuredClItemRow = itemView.findViewById(R.id.*cl\_featured\_item\_row*);  
 featuredImage = itemView.findViewById(R.id.*iv\_featured*);  
 featuredProductPrice = itemView.findViewById(R.id.*tv\_featured\_product\_price*);  
 featuredProductName = itemView.findViewById(R.id.*tv\_featured\_product\_name*);  
 featuredAuthorName=itemView.findViewById(R.id.*tv\_featured\_author\_name*);  
 }  
 }  
}

**rv\_bestseller\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/sharp\_corners\_views"  
 android:id="@+id/cl\_best\_seller\_item\_row"  
 android:padding="1dp"  
 android:layout\_margin="4dp">  
  
 <ImageView  
 android:id="@+id/iv\_bestseller"  
 android:layout\_width="200dp"  
 android:layout\_height="200dp"  
 android:layout\_marginTop="4dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <TextView  
 android:id="@+id/tv\_bestseller\_product\_price"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="4dp"  
 android:text="700"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 android:textColor="@color/white"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/iv\_bestseller" />  
  
 <TextView  
 android:id="@+id/tv\_bestseller\_product\_name"  
 android:layout\_width="160dp"  
 android:layout\_height="wrap\_content"  
 android:text="Harry Potter"  
 android:textColor="@color/white"  
 android:textSize="16sp"  
 android:maxLines="1"  
 android:ellipsize="end"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_bestseller\_product\_price"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_bestseller\_product\_price"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_bestseller\_product\_price" />  
  
 <TextView  
 android:id="@+id/tv\_bestseller\_author\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="25dp"  
 android:ellipsize="end"  
 android:maxLines="1"  
 android:text="Sarah J Mass"  
 android:textColor="@color/white"  
 android:textSize="16sp"  
 android:textStyle="italic"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_bestseller\_product\_name"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_bestseller\_product\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_bestseller\_product\_name" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**BestSeller:**

package com.example.spellbound.model;  
  
import android.os.Parcel;  
import android.os.Parcelable;  
  
import androidx.annotation.NonNull;  
  
import com.example.spellbound.FirestoreProducts;  
  
public class BestSeller implements Parcelable, FirestoreProducts {  
 private String bestSellerProductName;  
 private String bestSellerAuthorName;  
 private String bestSellerProductId;  
 private String bestSellerProductPrice;  
 private String bestSellerProductImageUrl;  
 private String bestSellerProductDescription;  
 private float bestSellerProductRating;  
 private String bestSellerProductQty;  
 private String bestSellerProductPublished;  
 private String bestSellerProductFormat;  
 private String bestSellerProductISBN;  
  
 public BestSeller() {  
 }  
  
 protected BestSeller(Parcel in) {  
 bestSellerProductName = in.readString();  
 bestSellerAuthorName = in.readString();  
 bestSellerProductId = in.readString();  
 bestSellerProductPrice = in.readString();  
 bestSellerProductImageUrl = in.readString();  
 bestSellerProductDescription = in.readString();  
 bestSellerProductRating = in.readFloat();  
 bestSellerProductQty = in.readString();  
 bestSellerProductPublished = in.readString();  
 bestSellerProductFormat = in.readString();  
 bestSellerProductISBN = in.readString();  
 }  
  
 public static final Creator<BestSeller> *CREATOR* = new Creator<BestSeller>() {  
 @Override  
 public BestSeller createFromParcel(Parcel in) {  
 return new BestSeller(in);  
 }  
  
 @Override  
 public BestSeller[] newArray(int size) {  
 return new BestSeller[size];  
 }  
 };  
  
 public String getBestSellerProductName() {  
 return bestSellerProductName;  
 }  
  
 public void setBestSellerProductName(String bestSellerProductName) {  
 this.bestSellerProductName = bestSellerProductName;  
 }  
  
 public String getBestSellerAuthorName() {  
 return bestSellerAuthorName;  
 }  
  
 public void setBestSellerAuthorName(String bestSellerAuthorName) {  
 this.bestSellerAuthorName = bestSellerAuthorName;  
 }  
  
 public String getBestSellerProductId() {  
 return bestSellerProductId;  
 }  
  
 public void setBestSellerProductId(String bestSellerProductId) {  
 this.bestSellerProductId = bestSellerProductId;  
 }  
  
 public String getBestSellerProductPrice() {  
 return bestSellerProductPrice;  
 }  
  
 public void setBestSellerProductPrice(String bestSellerProductPrice) {  
 this.bestSellerProductPrice = bestSellerProductPrice;  
 }  
  
 public String getBestSellerProductImageUrl() {  
 return bestSellerProductImageUrl;  
 }  
  
 public void setBestSellerProductImageUrl(String bestSellerProductImageUrl) {  
 this.bestSellerProductImageUrl = bestSellerProductImageUrl;  
 }  
  
 public String getBestSellerProductDescription() {  
 return bestSellerProductDescription;  
 }  
  
 public void setBestSellerProductDescription(String bestSellerProductDescription) {  
 this.bestSellerProductDescription = bestSellerProductDescription;  
 }  
  
 public float getBestSellerProductRating() {  
 return bestSellerProductRating;  
 }  
  
 public void setBestSellerProductRating(float bestSellerProductRating) {  
 this.bestSellerProductRating = bestSellerProductRating;  
 }  
  
 public String getBestSellerProductQty() {  
 return bestSellerProductQty;  
 }  
  
 public void setBestSellerProductQty(String bestSellerProductQty) {  
 this.bestSellerProductQty = bestSellerProductQty;  
 }  
  
 public String getBestSellerProductPublished() {  
 return bestSellerProductPublished;  
 }  
  
 public void setBestSellerProductPublished(String bestSellerProductPublished) {  
 this.bestSellerProductPublished = bestSellerProductPublished;  
 }  
  
 public String getBestSellerProductFormat() {  
 return bestSellerProductFormat;  
 }  
  
 public void setBestSellerProductFormat(String bestSellerProductFormat) {  
 this.bestSellerProductFormat = bestSellerProductFormat;  
 }  
  
 public String getBestSellerProductISBN() {  
 return bestSellerProductISBN;  
 }  
  
 public void setBestSellerProductISBN(String bestSellerProductISBN) {  
 this.bestSellerProductISBN = bestSellerProductISBN;  
 }  
  
 @Override  
 public int describeContents() {  
 return 0;  
 }  
  
 @Override  
 public void writeToParcel(@NonNull Parcel dest, int flags) {  
 dest.writeString(bestSellerProductName);  
 dest.writeString(bestSellerAuthorName);  
 dest.writeString(bestSellerProductId);  
 dest.writeString(bestSellerProductPrice);  
 dest.writeString(bestSellerProductImageUrl);  
 dest.writeString(bestSellerProductDescription);  
 dest.writeFloat(bestSellerProductRating);  
 dest.writeString(bestSellerProductQty);  
 dest.writeString(bestSellerProductPublished);  
 dest.writeString(bestSellerProductFormat);  
 dest.writeString(bestSellerProductISBN);  
 }  
  
 @Override  
 public String getFirestoreProductId() {  
 return bestSellerProductId;  
 }  
  
 @Override  
 public void setFirestoreProductId(String setId) {  
 this.bestSellerProductId=setId;  
  
 }  
  
 @Override  
 public String getFirestoreProductPrice() {  
 return bestSellerProductPrice;  
 }  
  
 @Override  
 public String getFirestoreProductImageUrl() {  
 return bestSellerProductImageUrl;  
 }  
  
 @Override  
 public String getFirestoreProductName() {  
 return bestSellerProductName;  
 }  
  
 @Override  
 public String getFirestoreProductQty() {  
 return bestSellerProductQty;  
 }  
}

**BestSellerAdapter:**

package com.example.spellbound.adapter;  
  
import static com.example.spellbound.UtilClass.*screenNavigationWithDataPassing*;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.constraintlayout.widget.ConstraintLayout;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.activity.ProductDetailActivity;  
import com.example.spellbound.model.BestSeller;  
  
import java.util.List;  
  
public class BestSellerAdapter extends RecyclerView.Adapter<BestSellerAdapter.BestSellerViewHolder>{  
  
 Context context;  
 List<BestSeller> bestSellerList;  
  
 public BestSellerAdapter(Context context, List<BestSeller> bestSellerList) {  
 this.context = context;  
 this.bestSellerList = bestSellerList;  
 }  
  
 @NonNull  
 @Override  
 public BestSellerViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view= LayoutInflater.*from*(context)  
 .inflate(R.layout.*rv\_bestseller\_item\_row*,parent,false);  
 return new BestSellerAdapter.BestSellerViewHolder(view);  
  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull BestSellerViewHolder holder, int position) {  
 Glide.*with*(context).load(bestSellerList.get(position)  
 .getBestSellerProductImageUrl()).into(holder.bestSellerImage);  
 *// holder.bestSellerProductPrice.setText(bestSellerList.get(position).getBestSellerProductPrice());* String strPrice=bestSellerList.get(position).getBestSellerProductPrice();  
 String priceWithSymbol=holder.itemView.getContext().getString(R.string.*rupee\_sign*,strPrice);  
 holder.bestSellerProductPrice.setText(priceWithSymbol);  
 holder.bestSellerProductName.setText(bestSellerList.get(position).getBestSellerProductName());  
 holder.bestSellerAuthorName.setText(bestSellerList.get(position).getBestSellerAuthorName());  
 holder.bestSellerClItemRow.setOnClickListener(v-> {  
 *screenNavigationWithDataPassing*(context, ProductDetailActivity.class,  
 "productDetails",bestSellerList.get(position));  
 });  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return bestSellerList.size();  
 }  
  
 static class BestSellerViewHolder extends RecyclerView.ViewHolder {  
 ConstraintLayout bestSellerClItemRow;  
 ImageView bestSellerImage;  
 TextView bestSellerProductPrice,bestSellerProductName,bestSellerAuthorName;  
 public BestSellerViewHolder(@NonNull View itemView) {  
 super(itemView);  
 bestSellerClItemRow = itemView.findViewById(R.id.*cl\_best\_seller\_item\_row*);  
 bestSellerImage = itemView.findViewById(R.id.*iv\_bestseller*);  
 bestSellerProductPrice = itemView.findViewById(R.id.*tv\_bestseller\_product\_price*);  
 bestSellerProductName = itemView.findViewById(R.id.*tv\_bestseller\_product\_name*);  
 bestSellerAuthorName = itemView.findViewById(R.id.*tv\_bestseller\_author\_name*);  
 }  
 }  
}

**rv\_product\_search\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" >  
  
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/cv\_product\_item\_row"  
 app:cardCornerRadius="8dp"  
 android:backgroundTint="@color/red"  
 app:cardElevation="4dp"  
 android:layout\_margin="6dp">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <ImageView  
 android:id="@+id/iv\_product\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="4dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/tv\_product\_search\_price"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <TextView  
 android:id="@+id/tv\_product\_search\_price"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="10dp"  
 android:text="₹700"  
 android:textColor="@color/white"  
 android:fontFamily="serif"  
 app:layout\_constraintEnd\_toEndOf="@+id/iv\_product\_image"  
 app:layout\_constraintStart\_toStartOf="@+id/iv\_product\_image"  
 app:layout\_constraintTop\_toBottomOf="@+id/iv\_product\_image" />  
  
 <TextView  
 android:id="@+id/tv\_product\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="10dp"  
 android:paddingBottom="6dp"  
 android:text="HarryPotter"  
 android:textColor="@color/white"  
 android:fontFamily="serif"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_product\_search\_price"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_product\_search\_price"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_search\_price" />  
  
 <TextView  
 android:id="@+id/tv\_product\_search\_author\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="10dp"  
 android:paddingBottom="6dp"  
 android:text="Sarah J Mass"  
 android:textColor="@color/white"  
 android:fontFamily="serif"  
 android:textSize="16sp"  
 android:textStyle="italic"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_product\_name"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_product\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_name" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </androidx.cardview.widget.CardView>  
  
  
</FrameLayout>

**AllProducts:**

package com.example.spellbound.model;  
  
import android.os.Parcel;  
import android.os.Parcelable;  
  
import androidx.annotation.NonNull;  
  
import com.example.spellbound.FirestoreProducts;  
  
import java.lang.reflect.Array;  
import java.util.Arrays;  
import java.util.List;  
  
public class AllProducts implements Parcelable, FirestoreProducts {  
 private String productId;  
 private String productName;  
 private String productAuthorName;  
 private String productDescription;  
 private String productPublished;  
 private String productFormat;  
 private String productISBN;  
 private String genreType;  
 private String productImageURL;  
 private int productPrice;  
 private int productQty;  
 private float productRating;  
  
 public AllProducts() {  
 }  
  
 protected AllProducts(Parcel in) {  
 productId = in.readString();  
 productName = in.readString();  
 productAuthorName = in.readString();  
 productDescription = in.readString();  
 productPublished = in.readString();  
 productFormat = in.readString();  
 productISBN = in.readString();  
 genreType = in.readString();  
 productImageURL = in.readString();  
 productPrice = in.readInt();  
 productQty = in.readInt();  
 productRating = in.readFloat();  
 }  
  
 public static final Creator<AllProducts> *CREATOR* = new Creator<AllProducts>() {  
 @Override  
 public AllProducts createFromParcel(Parcel in) {  
 return new AllProducts(in);  
 }  
  
 @Override  
 public AllProducts[] newArray(int size) {  
 return new AllProducts[size];  
 }  
 };  
  
 public String getProductId() {  
 return productId;  
 }  
  
 public void setProductId(String productId) {  
 this.productId = productId;  
 }  
  
 public String getProductName() {  
 return productName;  
 }  
  
 public void setProductName(String productName) {  
 this.productName = productName;  
 }  
  
 public String getProductAuthorName() {  
 return productAuthorName;  
 }  
  
 public void setProductAuthorName(String productAuthorName) {  
 this.productAuthorName = productAuthorName;  
 }  
  
 public String getProductDescription() {  
 return productDescription;  
 }  
  
 public void setProductDescription(String productDescription) {  
 this.productDescription = productDescription;  
 }  
  
 public String getProductPublished() {  
 return productPublished;  
 }  
  
 public void setProductPublished(String productPublished) {  
 this.productPublished = productPublished;  
 }  
  
 public String getProductFormat() {  
 return productFormat;  
 }  
  
 public void setProductFormat(String productFormat) {  
 this.productFormat = productFormat;  
 }  
  
 public String getProductISBN() {  
 return productISBN;  
 }  
  
 public void setProductISBN(String productISBN) {  
 this.productISBN = productISBN;  
 }  
  
 public String getGenreType() {  
 return genreType;  
 }  
  
 public void setGenreType(String genreType) {  
 this.genreType = genreType;  
 }  
  
 public String getProductImageURL() {  
 return productImageURL;  
 }  
  
 public void setProductImageURL(String productImageURL) {  
 this.productImageURL = productImageURL;  
 }  
  
 public int getProductPrice() {  
 return productPrice;  
 }  
  
 public void setProductPrice(int productPrice) {  
 this.productPrice = productPrice;  
 }  
  
 public int getProductQty() {  
 return productQty;  
 }  
  
 public void setProductQty(int productQty) {  
 this.productQty = productQty;  
 }  
  
 public float getProductRating() {  
 return productRating;  
 }  
  
 public void setProductRating(float productRating) {  
 this.productRating = productRating;  
 }  
  
 @Override  
 public int describeContents() {  
 return 0;  
 }  
  
 @Override  
 public void writeToParcel(@NonNull Parcel dest, int flags) {  
 dest.writeString(productId);  
 dest.writeString(productName);  
 dest.writeString(productAuthorName);  
 dest.writeString(productDescription);  
 dest.writeString(productPublished);  
 dest.writeString(productFormat);  
 dest.writeString(productISBN);  
 dest.writeString(genreType);  
 dest.writeString(productImageURL);  
 dest.writeInt(productPrice);  
 dest.writeInt(productQty);  
 dest.writeFloat(productRating);  
 }  
  
 @Override  
 public String getFirestoreProductId() {  
 return productId;  
 }  
  
 @Override  
 public void setFirestoreProductId(String setId) {  
 this.productId=setId;  
  
 }  
  
 @Override  
 public String getFirestoreProductPrice() {  
 return String.*valueOf*(productPrice);  
 }  
  
 @Override  
 public String getFirestoreProductImageUrl() {  
 return productImageURL;  
 }  
  
 @Override  
 public String getFirestoreProductName() {  
 return productName;  
 }  
  
 @Override  
 public String getFirestoreProductQty() {  
 return String.*valueOf*(productQty);  
 }  
}

**AllProductsAdapter:**

package com.example.spellbound.adapter;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.activity.HomeActivity;  
import com.example.spellbound.activity.ProductDetailActivity;  
import com.example.spellbound.databinding.RvProductSearchItemRowBinding;  
import com.example.spellbound.model.AllProducts;  
  
import java.util.List;  
  
public class AllProductsAdapter extends RecyclerView.Adapter<AllProductsAdapter.AllProductsViewHolder> {  
 Context context;  
 List<AllProducts> allProductList;  
  
 public AllProductsAdapter(Context context, List<AllProducts> allProductList) {  
 this.context = context;  
 this.allProductList = allProductList;  
 }  
  
 @NonNull  
 @Override  
 public AllProductsAdapter.AllProductsViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 RvProductSearchItemRowBinding binding = RvProductSearchItemRowBinding.  
 *inflate*(LayoutInflater.*from*(context), parent, false);  
 return new AllProductsViewHolder(binding);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull AllProductsAdapter.AllProductsViewHolder holder, int position) {  
 holder.binding.tvProductName.setText(allProductList.get(position).getProductName());  
 holder.binding.tvProductSearchAuthorName.setText(allProductList.get(position).getProductAuthorName());  
  
 String strPrice = String.*valueOf*(allProductList.get(position).getProductPrice());  
 String priceWithSymbol = holder.itemView.getContext().getString(R.string.*rupee\_sign*, strPrice);  
  
 holder.binding.tvProductSearchPrice.setText(priceWithSymbol);  
  
 if (!(context instanceof HomeActivity)) {  
 Glide.*with*(context).load(allProductList.get(position)  
 .getProductImageURL()).into(holder.binding.ivProductImage);  
 } else {  
 holder.binding.ivProductImage.setVisibility(View.*GONE*);  
 }  
 holder.binding.cvProductItemRow.setOnClickListener(v -> {  
 UtilClass.*screenNavigationWithDataPassing*(context, ProductDetailActivity.class,  
 "productDetails", allProductList.get(position));  
 });  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return allProductList.size();  
  
 }  
  
 static class AllProductsViewHolder extends RecyclerView.ViewHolder {  
 RvProductSearchItemRowBinding binding;  
  
 public AllProductsViewHolder(RvProductSearchItemRowBinding binding) {  
 super(binding.getRoot());  
 this.binding = binding;  
 }  
 }  
}

**activity\_home:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/lion"  
 tools:context=".activity.HomeActivity">  
  
 <include  
 layout="@layout/toolbar\_custom\_layout"  
 android:id="@+id/toolbar\_custom"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <ImageView  
 android:id="@+id/iv\_search\_icon"  
 android:layout\_width="30dp"  
 android:layout\_height="38dp"  
 android:layout\_marginStart="17dp"  
 android:layout\_marginTop="6dp"  
 app:layout\_constraintEnd\_toStartOf="@+id/et\_product\_search\_box"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintHorizontal\_chainStyle="spread\_inside"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom"  
 app:srcCompat="@drawable/search\_icon" />  
  
 <EditText  
 android:id="@+id/et\_product\_search\_box"  
 android:layout\_width="0dp"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="6dp"  
 android:layout\_marginEnd="2dp"  
 android:ems="18"  
 android:inputType="text"  
 android:hint="Search"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toEndOf="@+id/iv\_search\_icon"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_product\_search"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="6dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/home\_container"  
 app:layout\_constraintEnd\_toEndOf="@id/et\_product\_search\_box"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@id/et\_product\_search\_box"  
 app:layout\_constraintTop\_toBottomOf="@id/et\_product\_search\_box" />  
  
 <FrameLayout  
 android:id="@+id/home\_container"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="15dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@id/rv\_product\_search"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@id/rv\_product\_search"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_product\_search">  
  
 </FrameLayout>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**HomeActivity:**

package com.example.spellbound.activity;  
  
import static com.example.spellbound.UtilClass.*screenNavigation*;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
import androidx.fragment.app.FragmentTransaction;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Toast;  
  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.adapter.AllProductsAdapter;  
import com.example.spellbound.databinding.ActivityHomeBinding;  
import com.example.spellbound.fragment.HomeFragment;  
import com.example.spellbound.R;  
import com.example.spellbound.model.AllProducts;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class HomeActivity extends AppCompatActivity {  
  
 private static final String *TAG* = "HomeActivity";  
 boolean isFragmentVisible = false;  
 ActivityHomeBinding binding;  
 Fragment homeFragment;  
 FirebaseAuth firebaseAuth;  
 FirebaseFirestore firebaseFirestore;  
 AllProductsAdapter allProductAdapter;  
 List<AllProducts> allProductList;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityHomeBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 firebaseAuth = FirebaseAuth.*getInstance*();  
 firebaseFirestore=FirebaseFirestore.*getInstance*();  
  
 UtilClass.*setUpCustomToolbar*(HomeActivity.this);  
 homeFragment=new HomeFragment();  
 loadFragment(homeFragment);  
 initRecyclerView();  
 searchFilter();  
  
 }  
 private void initRecyclerView() {  
 allProductList = new ArrayList<>();  
 allProductAdapter = new AllProductsAdapter(this,allProductList);  
 binding.rvProductSearch.setLayoutManager(new LinearLayoutManager(this));  
 binding.rvProductSearch.setAdapter(allProductAdapter);  
  
 setVisibilityVisibleToFragmentHomeContainer();  
 setVisibilityGoneToActivityRvProduct();  
 }  
  
 private void setVisibilityVisibleToFragmentHomeContainer() {  
 binding.homeContainer.setVisibility(View.*VISIBLE*);  
 isFragmentVisible = false;  
 }  
  
 private void setVisibilityGoneToActivityRvProduct() {  
 binding.rvProductSearch.setVisibility(View.*GONE*);  
 isFragmentVisible = true;  
 }  
 private void setVisibilityVisibleToActivityRvProduct() {  
 binding.rvProductSearch.setVisibility(View.*VISIBLE*);  
 isFragmentVisible = false;  
 }  
 private void setVisibilityGoneToFragmentHomeContainer() {  
 binding.homeContainer.setVisibility(View.*GONE*);  
 isFragmentVisible=true;  
 }  
  
  
 private void searchFilter() {  
 binding.etProductSearchBox.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
  
 }  
  
 @Override  
 public void afterTextChanged(Editable editTextUserInput) {  
 String searchQuery = editTextUserInput.toString();  
 if(searchQuery.isEmpty()) {  
 allProductList.clear();  
 allProductAdapter.notifyDataSetChanged();  
 }  
 binding.ivSearchIcon.setOnClickListener(v -> {  
 searchProduct(searchQuery);  
 });  
  
 }  
 });  
 }  
  
 private void searchProduct(String searchQuery) {  
 if(!searchQuery.isEmpty()) {  
 firebaseFirestore.collection("AllProducts")  
 .whereGreaterThanOrEqualTo("productName", searchQuery)  
 .get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 AllProducts allProducts = documentSnapshot.toObject(AllProducts.class);  
 allProductList.add(allProducts);  
 setVisibilityVisibleToActivityRvProduct();  
 setVisibilityGoneToFragmentHomeContainer();  
 allProductAdapter.notifyItemChanged(Objects.*requireNonNull*(documentSnapshot.getData()).size());  
 }  
 }  
 });  
 }  
  
  
 }  
  
 private void loadFragment(Fragment fragment)  
 {  
 FragmentTransaction transaction= getSupportFragmentManager().beginTransaction();  
 transaction.replace(R.id.*home\_container*,fragment,*TAG*);  
 // transaction.addToBackStack(TAG);  
 transaction.commit();  
  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*main\_menu*,menu);  
 return super.onCreateOptionsMenu(menu);  
 }  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
  
 if (item.getItemId() == R.id.*action\_logout*) {  
 firebaseAuth.signOut();  
 Toast.*makeText*(this, "Logged Out", Toast.*LENGTH\_SHORT*).show();  
 *screenNavigation*(HomeActivity.this, WelcomeActivity.class);  
 finish();  
 } else if (item.getItemId()==R.id.*action\_cart*) {  
 *screenNavigation*(this, CartActivity.class);  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

**fragment\_home:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.core.widget.NestedScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:background="@color/lion"  
 tools:context=".fragment.HomeFragment">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 >  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_categories"  
 android:layout\_width="376dp"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="35dp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline10"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline9"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline11" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline9"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_begin="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline10"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_end="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline11"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline12"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_end="10dp" />  
  
 <TextView  
 android:id="@+id/textView7"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="4dp"  
 android:text="Categories"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline9"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline11" />  
  
 <TextView  
 android:id="@+id/tv\_see\_all\_category"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="4dp"  
 android:text="See All"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline10"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline11" />  
  
 <TextView  
 android:id="@+id/textView9"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="4dp"  
 android:text="Featured"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline9"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_categories" />  
  
 <TextView  
 android:id="@+id/tv\_see\_all\_featured"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="4dp"  
 android:text="See All"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline10"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_categories" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_featured"  
 android:layout\_width="376dp"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="35dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/rv\_categories"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="@+id/rv\_categories"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_categories" />  
  
 <TextView  
 android:id="@+id/textView11"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="4dp"  
 android:layout\_marginTop="4dp"  
 android:text="Best Seller"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline9"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_featured" />  
  
 <TextView  
 android:id="@+id/tv\_see\_all\_bestseller"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:layout\_marginEnd="4dp"  
 android:text="See All"  
 android:fontFamily="serif"  
 android:textColor="@color/bistre"  
 android:textStyle="bold"  
 android:textSize="18sp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline10"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_featured" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_bestseller"  
 android:layout\_width="377dp"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="35dp"  
 app:layout\_constraintEnd\_toEndOf="@+id/rv\_featured"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="@+id/rv\_featured"  
 app:layout\_constraintTop\_toBottomOf="@+id/rv\_featured" />  
 </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.core.widget.NestedScrollView>

**HomeFragment:**

package com.example.spellbound.fragment;  
  
import static com.example.spellbound.UtilClass.*screenNavigation*;  
import static com.example.spellbound.UtilClass.*screenNavigationWithDataPassing*;  
  
import android.os.Bundle;  
  
import androidx.annotation.NonNull;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.util.Log;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import com.example.spellbound.activity.AllProductsActivity;  
import com.example.spellbound.activity.GenreActivity;  
import com.example.spellbound.adapter.BestSellerAdapter;  
import com.example.spellbound.adapter.CategoryAdapter;  
import com.example.spellbound.adapter.FeaturedAdapter;  
import com.example.spellbound.databinding.FragmentHomeBinding;  
import com.example.spellbound.model.BestSeller;  
import com.example.spellbound.model.Category;  
import com.example.spellbound.model.Featured;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.firestore.QueryDocumentSnapshot;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class HomeFragment extends Fragment {  
  
 FragmentHomeBinding binding;  
 FirebaseFirestore firebaseFireStore;  
  
 private static final String *TAG* = "HomeFragment";  
 // Category  
 CategoryAdapter categoryAdapter;  
 List<Category> categoryList;  
  
 // Featured  
 FeaturedAdapter featuredAdapter;  
 List<Featured> featuredList;  
  
 // BestSeller  
 BestSellerAdapter bestSellerAdapter;  
 List<BestSeller> bestSellerList;  
  
 public HomeFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 binding=FragmentHomeBinding.*inflate*(inflater,container,false);  
 // Inflate the layout for this fragment  
  
 View view = binding.getRoot();  
 firebaseFireStore = FirebaseFirestore.*getInstance*();  
 initRecyclerView();  
  
 navigationOfViews();  
  
 getCategoryDataFromFireStore();  
 getFeaturedDataFromFireStore();  
 getBestSellerDataFromFireStore();  
  
 return view;  
 }  
  
 private void navigationOfViews() {  
 binding.tvSeeAllCategory.setOnClickListener(v -> {  
 *screenNavigation*(getContext(), GenreActivity.class);  
 });  
 binding.tvSeeAllFeatured.setOnClickListener(v -> {  
 *screenNavigation*(getContext(), AllProductsActivity.class);  
 });  
 binding.tvSeeAllBestseller.setOnClickListener(v -> {  
 *screenNavigation*(getContext(), AllProductsActivity.class);  
 });  
  
 }  
  
 private void getBestSellerDataFromFireStore() {  
  
 firebaseFireStore.collection("BestSeller").get().addOnCompleteListener(task->{  
 if(task.isSuccessful()) {  
 for(QueryDocumentSnapshot document : task.getResult()) {  
 BestSeller bestSeller=document.toObject(BestSeller.class);  
 bestSellerList.add(bestSeller);  
 bestSellerAdapter.notifyItemInserted(document.getData().size());  
 Log.*d*(*TAG*,document.getId()+" => "+ document.getData());  
 }  
 } else {  
 Log.*d*(*TAG*,"getBestSellerDataFromFireStore: Error couldn't get the document",  
 task.getException());  
 }  
 });  
 }  
  
 private void getFeaturedDataFromFireStore() {  
 firebaseFireStore.collection("Featured").get().addOnCompleteListener(task->{  
 if(task.isSuccessful()) {  
 for(QueryDocumentSnapshot document : task.getResult()) {  
 Featured featured=document.toObject(Featured.class);  
 featuredList.add(featured);  
 featuredAdapter.notifyItemInserted(document.getData().size());  
 Log.*d*(*TAG*,document.getId()+" => "+ document.getData());  
 }  
 } else {  
 Log.*d*(*TAG*,"getFeaturedDataFromFireStore: Error couldn't get the document",  
 task.getException());  
 }  
 });  
 }  
  
 private void getCategoryDataFromFireStore() {  
 firebaseFireStore.collection("Category").get().addOnCompleteListener(task->{  
 if(task.isSuccessful()) {  
 for(QueryDocumentSnapshot document : task.getResult()) {  
 Category category = document.toObject(Category.class);  
 categoryList.add(category);  
 categoryAdapter.notifyItemInserted(document.getData().size());  
 Log.*d*(*TAG*,document.getId()+" => "+ document.getData());  
 }  
 } else {  
 Log.*d*(*TAG*,"getCategoryDataFromFireStore: Error couldn't get the document",  
 task.getException());  
 }  
 });  
 }  
  
 private void initRecyclerView() {  
 // Category  
 categoryList=new ArrayList<>();  
 categoryAdapter=new CategoryAdapter(getContext(),categoryList);  
 binding.rvCategories.setLayoutManager(new LinearLayoutManager(getContext(),  
 LinearLayoutManager.*HORIZONTAL*,false));  
 binding.rvCategories.setAdapter(categoryAdapter);  
  
 // Featured  
 featuredList=new ArrayList<>();  
 featuredAdapter=new FeaturedAdapter(getContext(),featuredList);  
 binding.rvFeatured.setLayoutManager(new LinearLayoutManager(getContext(),  
 LinearLayoutManager.*HORIZONTAL*,false));  
 binding.rvFeatured.setAdapter(featuredAdapter);  
  
 // BestSeller  
 bestSellerList=new ArrayList<>();  
 bestSellerAdapter=new BestSellerAdapter(getContext(),bestSellerList);  
 binding.rvBestseller.setLayoutManager(new LinearLayoutManager(getContext(),  
 LinearLayoutManager.*HORIZONTAL*,false));  
 binding.rvBestseller.setAdapter(bestSellerAdapter);  
  
 }  
  
}

**rv\_genre\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <ImageView  
 android:id="@+id/iv\_genre"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:layout\_margin="5dp"  
 android:padding="3dp"  
 android:src="@drawable/ic\_launcher\_background"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**Genre:**

package com.example.spellbound.model;  
  
import java.util.List;  
  
public class Genre {  
 private String genreImageUrl;  
  
 private String genreType;  
 private String categoryType;  
  
 public Genre() {  
 }  
  
 public String getGenreImageUrl() {  
 return genreImageUrl;  
 }  
  
 public void setGenreImageUrl(String genreImageUrl) {  
 this.genreImageUrl = genreImageUrl;  
 }  
  
 public String getGenreType() {  
 return genreType;  
 }  
  
 public void setGenreType(String genreType) {  
 this.genreType = genreType;  
 }  
  
 public String getCategoryType() {  
 return categoryType;  
 }  
  
 public void setCategoryType(String categoryType) {  
 this.categoryType = categoryType;  
 }  
}

**GenreAdapter:**

package com.example.spellbound.adapter;  
  
import static com.example.spellbound.UtilClass.*screenNavigationWithDataPassing*;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.view.animation.Animation;  
import android.view.animation.AnimationUtils;  
import android.widget.ImageView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.activity.AllProductsActivity;  
import com.example.spellbound.model.Genre;  
  
import java.util.List;  
  
public class GenreAdapter extends RecyclerView.Adapter<GenreAdapter.GenreViewHolder> {  
  
 Context context;  
 List<Genre> genreList;  
  
 public GenreAdapter(Context context, List<Genre> genreList) {  
 this.context = context;  
 this.genreList = genreList;  
 }  
  
 @NonNull  
 @Override  
 public GenreViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view= LayoutInflater.*from*(context)  
 .inflate(R.layout.*rv\_genre\_item\_row*,parent,false);  
 return new GenreViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull GenreAdapter.GenreViewHolder holder, int position) {  
 Glide.*with*(context).load(genreList.get(position).getGenreImageUrl())  
 .into(holder.genreImage);  
 Animation animation = AnimationUtils.*loadAnimation*(context.getApplicationContext(), R.anim.*move*);  
 holder.genreImage.startAnimation(animation);  
 holder.genreImage.setOnClickListener(v -> {  
 *screenNavigationWithDataPassing*(context, AllProductsActivity.class,"genreType",  
 genreList.get(position).getGenreType());  
 });  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return genreList.size();  
 }  
 static class GenreViewHolder extends RecyclerView.ViewHolder {  
 ImageView genreImage;  
 public GenreViewHolder(@NonNull View itemView)  
 {  
 super(itemView);  
 genreImage=itemView.findViewById(R.id.*iv\_genre*);  
 }  
 }  
}

**activity\_genre:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.core.widget.NestedScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/brown"  
 tools:context=".activity.GenreActivity">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <include  
 layout="@layout/toolbar\_custom\_layout"  
 android:id="@+id/toolbar\_custom"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_genre"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="12dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_genre\_title" />  
  
 <TextView  
 android:id="@+id/tv\_genre\_title"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:text="Genre"  
 android:fontFamily="serif"  
 android:textSize="34sp"  
 android:textColor="@color/peach"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.496"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</androidx.core.widget.NestedScrollView>

**GenreActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.os.Bundle;  
import android.util.Log;  
  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.adapter.GenreAdapter;  
import com.example.spellbound.databinding.ActivityGenreBinding;  
import com.example.spellbound.model.Genre;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class GenreActivity extends AppCompatActivity {  
 private static final String *TAG* = "AllProductsActivity";  
 ActivityGenreBinding binding;  
 GenreAdapter genreAdapter;  
 List<Genre> genreList;  
 FirebaseFirestore firebaseFirestore;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding= ActivityGenreBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 UtilClass.*setUpCustomToolbar*(GenreActivity.this);  
 firebaseFirestore = FirebaseFirestore.*getInstance*();  
 String categoryType = getIntent().getStringExtra("categoryType");  
 intiRecyclerView();  
 getDataFromGenres(categoryType);  
 }  
  
 private void getDataFromGenres(String categoryType) {  
 if (categoryType==null || categoryType.isEmpty()) {  
 binding.tvGenreTitle.setText("All Genres");  
 firebaseFirestore.collection("Genre").get()  
 .addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 Genre genre=documentSnapshot.toObject(Genre.class);  
 genreList.add(genre);  
 genreAdapter.notifyItemInserted(Objects.  
 *requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*,"getDataFromGenres: ", task.getException());  
 }  
 });  
 } else if(categoryType.equals("Fiction")) {  
 binding.tvGenreTitle.setText("Fiction");  
 firebaseFirestore.collection("Genre").whereEqualTo("categoryType","Fiction")  
 .get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 Genre genre = documentSnapshot.toObject(Genre.class);  
 genreList.add(genre);  
 genreAdapter.notifyItemInserted(Objects.  
 *requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*, "getDataFromGenres: ",task.getException());  
 }  
 });  
 } else if(categoryType.equals("Non-Fiction")) {  
 binding.tvGenreTitle.setText("Non-Fiction");  
 firebaseFirestore.collection("Genre").whereEqualTo("categoryType","Non-Fiction")  
 .get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 Genre genre=documentSnapshot.toObject(Genre.class);  
 genreList.add(genre);  
 genreAdapter.notifyItemInserted(Objects.*requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*,"getDataFromAllGenres: ",task.getException());  
 }  
 });  
 }  
 }  
  
 private void intiRecyclerView() {  
 genreList=new ArrayList<>();  
 genreAdapter=new GenreAdapter(GenreActivity.this,genreList);  
 binding.rvGenre.setLayoutManager(new LinearLayoutManager(this));  
 binding.rvGenre.setAdapter(genreAdapter);  
 }

}

**activity\_all\_products:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/lion"  
 tools:context=".activity.AllProductsActivity">  
  
 <include  
 layout="@layout/toolbar\_custom\_layout"  
 android:id="@+id/toolbar\_custom"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_all\_products"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

**AllProductsActivity:**

package com.example.spellbound.activity;  
  
import static com.example.spellbound.UtilClass.*setUpCustomToolbar*;  
import static com.example.spellbound.UtilClass.*showDataBasedOnTheGenreType*;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.GridLayoutManager;  
  
import android.content.Context;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.SearchView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.adapter.AllProductsAdapter;  
import com.example.spellbound.databinding.ActivityAllProductsBinding;  
import com.example.spellbound.model.AllProducts;  
import com.example.spellbound.model.Genre;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
import java.util.Objects;  
  
public class AllProductsActivity extends AppCompatActivity {  
 private static final String *TAG* = "AllProductsActivity";  
 ActivityAllProductsBinding binding;  
 AllProductsAdapter allProductAdapter;  
 List<AllProducts> allProductsList;  
 FirebaseFirestore firebaseFirestore;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding= ActivityAllProductsBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 *setUpCustomToolbar*(AllProductsActivity.this);  
 firebaseFirestore = FirebaseFirestore.*getInstance*();  
 String genreType = getIntent().getStringExtra("genreType");  
 int position=0;  
 intiRecyclerView();  
 getDataFromAllProducts(genreType,position);  
 }  
 private void getDataFromAllProducts(String genreType,int position) {  
 if (genreType==null || (genreType).isEmpty()) {  
 firebaseFirestore.collection("AllProducts").get()  
 .addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 AllProducts allProducts=documentSnapshot.toObject(AllProducts.class);  
 allProductsList.add(allProducts);  
 allProductAdapter.notifyItemInserted(Objects.  
 *requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*,"getDataFromAllProducts: ", task.getException());  
 }  
 });  
 } else if((genreType).contains("Historical-Fiction")) {  
 *showDataBasedOnTheGenreType*("genreType","Historical-Fiction",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Classics")) {  
 *showDataBasedOnTheGenreType*("genreType","Classics",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Fantasy")) {  
 *showDataBasedOnTheGenreType*("genreType","Fantasy",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Horror")) {  
 *showDataBasedOnTheGenreType*("genreType","Horror",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Mystery")) {  
 *showDataBasedOnTheGenreType*("genreType","Mystery",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Romance")) {  
 *showDataBasedOnTheGenreType*("genreType","Romance",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Science-Fiction")) {  
 *showDataBasedOnTheGenreType*("genreType","Science-Fiction",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Business")) {  
 *showDataBasedOnTheGenreType*("genreType","Business",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Psychology")) {  
 *showDataBasedOnTheGenreType*("genreType","Psychology",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("History")) {  
 *showDataBasedOnTheGenreType*("genreType","History",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Memoir")) {  
 *showDataBasedOnTheGenreType*("genreType","Memoir",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Philosophy")) {  
 *showDataBasedOnTheGenreType*("genreType","Philosophy",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Science")) {  
 *showDataBasedOnTheGenreType*("genreType","Science",allProductAdapter,allProductsList);  
 }  
 else if((genreType).contains("Self-Help")) {  
 *showDataBasedOnTheGenreType*("genreType","Self-Help",allProductAdapter,allProductsList);  
 }  
 }  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*search\_menu*,menu);  
 MenuItem menuItem=menu.findItem(R.id.*action\_search*);  
 SearchView searchView = (SearchView) menuItem.getActionView();  
  
 assert searchView!=null;  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 searchProduct(query);  
 return true;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String newText) {  
 return false;  
 }  
 });  
 return super.onCreateOptionsMenu(menu);  
  
 }  
  
 private void searchProduct(String query) {  
 allProductsList.clear();  
 firebaseFirestore.collection("AllProducts").whereGreaterThanOrEqualTo("productName",query)  
 .get().addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 for(DocumentSnapshot documentSnapshot: task.getResult()) {  
 AllProducts allProducts=documentSnapshot.toObject(AllProducts.class);  
 allProductsList.add(allProducts);  
 allProductAdapter.notifyItemRangeChanged(0,  
 Objects.*requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*, "searchProduct: ",task.getException());  
 }  
 });  
 }  
 private void intiRecyclerView() {  
 allProductsList=new ArrayList<>();  
 allProductAdapter=new AllProductsAdapter(AllProductsActivity.this,allProductsList);  
 binding.rvAllProducts.setLayoutManager(new GridLayoutManager(this,2));  
 binding.rvAllProducts.setAdapter(allProductAdapter);  
 }

}

**activity\_product\_detail:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.core.widget.NestedScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/lion"  
 tools:context=".activity.ProductDetailActivity">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline13"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_end="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline14"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_begin="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline15"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_end="10dp" />  
  
 <androidx.constraintlayout.widget.Guideline  
 android:id="@+id/guideline16"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 app:layout\_constraintGuide\_begin="10dp" />  
 <include  
 android:id="@+id/toolbar\_custom"  
 layout="@layout/toolbar\_custom\_layout"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline16" />  
  
 <ImageView  
 android:id="@+id/iv\_product\_item"  
 android:layout\_width="372dp"  
 android:layout\_height="254dp"  
 android:layout\_marginTop="7dp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintHorizontal\_bias="0.494"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <TextView  
 android:id="@+id/tv\_product\_name"  
 android:layout\_width="340dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="5dp"  
 android:text="product name"  
 android:textColor="@color/red"  
 android:fontFamily="serif"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/iv\_divider"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/iv\_divider"  
 app:layout\_constraintTop\_toBottomOf="@+id/iv\_divider" />  
  
 <TextView  
 android:id="@+id/tv\_product\_author\_name"  
 android:layout\_width="340dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="8dp"  
 android:text="author name"  
 android:textColor="@color/red"  
 android:fontFamily="serif"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_product\_name"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_product\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_name" />  
  
 <TextView  
 android:id="@+id/tv\_product\_price"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="8dp"  
 android:text="price"  
 android:textColor="@color/black"  
 android:fontFamily="serif"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_product\_author\_name"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_product\_author\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_author\_name" />  
  
 <RatingBar  
 android:id="@+id/rating"  
 style="?android:attr/ratingBarStyleIndicator"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10dp"  
 android:isIndicator="true"  
 android:numStars="5"  
 android:stepSize="0.2"  
 android:theme="@style/ratingBar"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_product\_price"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_product\_rating"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_price" />  
  
 <View  
 android:id="@+id/iv\_divider"  
 android:layout\_width="391dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="10dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/iv\_product\_item" />  
  
  
 <TextView  
 android:id="@+id/tv\_product\_rating"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="10dp"  
 android:text="4.16"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:fontFamily="serif"  
 android:textSize="20sp"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_product\_price"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_price" />  
  
 <View  
 android:id="@+id/divider"  
 android:layout\_width="391dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="10dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/rating" />  
  
 <TextView  
 android:id="@+id/textView10"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="6dp"  
 android:layout\_marginTop="7dp"  
 android:text="Synopsis"  
 android:textColor="@color/black"  
 android:fontFamily="serif"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider" />  
  
 <TextView  
 android:id="@+id/tv\_product\_description"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginEnd="24dp"  
 android:text="Harry Potter an"  
 android:textColor="@color/black"  
 android:fontFamily="serif"  
 android:textSize="15sp"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView10"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/textView10"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView10" />  
  
 <View  
 android:id="@+id/divider2"  
 android:layout\_width="391dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="10dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_product\_description" />  
  
 <TextView  
 android:id="@+id/textView12"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="8dp"  
 android:text="Product Information"  
 android:textColor="@color/black"  
 android:fontFamily="serif"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider2" />  
  
 <TextView  
 android:id="@+id/textView13"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:text="ISBN 13:"  
 android:textColor="@color/black"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView12"  
 app:layout\_constraintHorizontal\_bias="0.231"  
 app:layout\_constraintStart\_toStartOf="@+id/textView12"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView12" />  
  
 <TextView  
 android:id="@+id/tv\_product\_isbn"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="44dp"  
 android:text="9245645789325"  
 android:textColor="@color/black"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="@+id/textView13"  
 app:layout\_constraintStart\_toEndOf="@+id/textView13"  
 app:layout\_constraintTop\_toTopOf="@+id/textView13"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <TextView  
 android:id="@+id/textView15"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Format :"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 android:textColor="@color/black"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView13"  
 app:layout\_constraintHorizontal\_bias="0.231"  
 app:layout\_constraintStart\_toStartOf="@+id/textView13"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView13" />  
  
 <TextView  
 android:id="@+id/tv\_product\_format"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="40dp"  
 android:text="paperback"  
 android:textColor="@color/black"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="@+id/textView15"  
 app:layout\_constraintStart\_toEndOf="@+id/textView15"  
 app:layout\_constraintTop\_toTopOf="@+id/textView15"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <TextView  
 android:id="@+id/textView17"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="Publisher :"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 android:textColor="@color/black"  
 app:layout\_constraintEnd\_toEndOf="@+id/textView15"  
 app:layout\_constraintHorizontal\_bias="0.231"  
 app:layout\_constraintStart\_toStartOf="@+id/textView15"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView15" />  
  
 <TextView  
 android:id="@+id/tv\_product\_publisher"  
 android:layout\_width="180dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="28dp"  
 android:text="Walker Books"  
 android:textSize="15sp"  
 android:fontFamily="serif"  
 android:textColor="@color/black"  
 app:layout\_constraintBottom\_toBottomOf="@+id/textView17"  
 app:layout\_constraintStart\_toEndOf="@+id/textView17"  
 app:layout\_constraintTop\_toTopOf="@+id/textView17"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <View  
 android:id="@+id/divider3"  
 android:layout\_width="391dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="17dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView17" />  
  
 <View  
 android:id="@+id/divider4"  
 android:layout\_width="391dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="57dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider3" />  
  
 <TextView  
 android:id="@+id/textView19"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="5dp"  
 android:layout\_marginTop="18dp"  
 android:text="Quantity"  
 android:fontFamily="serif"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 android:textColor="@color/black"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider3" />  
  
 <Button  
 android:id="@+id/btn\_decrease\_qty"  
 android:layout\_width="60dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="30dp"  
 android:layout\_marginTop="4dp"  
 android:text="-"  
 android:backgroundTint="@color/bistre"  
 android:fontFamily="serif"  
 app:cornerRadius="12dp"  
 app:layout\_constraintBottom\_toBottomOf="@+id/textView19"  
 app:layout\_constraintStart\_toEndOf="@+id/textView19"  
 app:layout\_constraintTop\_toTopOf="@+id/textView19"  
 app:layout\_constraintVertical\_bias="0.586" />  
  
 <Button  
 android:id="@+id/btn\_product\_qty"  
 android:layout\_width="76dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginTop="4dp"  
 app:cornerRadius="12dp"  
 app:backgroundTint="@null"  
 android:background="@drawable/gradient\_btn"  
 android:text="1"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="@+id/btn\_decrease\_qty"  
 app:layout\_constraintStart\_toEndOf="@+id/btn\_decrease\_qty"  
 app:layout\_constraintTop\_toTopOf="@+id/btn\_decrease\_qty" />  
  
 <Button  
 android:id="@+id/btn\_increase\_qty"  
 android:layout\_width="60dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginTop="4dp"  
 app:cornerRadius="12dp"  
 android:text="+"  
 android:backgroundTint="@color/bistre"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="@+id/btn\_product\_qty"  
 app:layout\_constraintStart\_toEndOf="@+id/btn\_product\_qty"  
 app:layout\_constraintTop\_toTopOf="@+id/btn\_product\_qty"  
 app:layout\_constraintVertical\_bias="1.0" />  
  
 <Button  
 android:id="@+id/btn\_add\_to\_cart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:background="@drawable/gradient\_btn"  
 android:text="ADD TO CART"  
 android:fontFamily="serif"  
 app:backgroundTint="@null"  
 app:cornerRadius="7dp"  
 app:layout\_constraintEnd\_toStartOf="@+id/guideline13"  
 app:layout\_constraintHorizontal\_bias="0.526"  
 app:layout\_constraintStart\_toStartOf="@+id/guideline14"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider4" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
</androidx.core.widget.NestedScrollView>

**ProductDetailActivity:**

package com.example.spellbound.activity;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.Toast;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.databinding.ActivityProductDetailBinding;  
import com.example.spellbound.model.AllProducts;  
import com.example.spellbound.model.BestSeller;  
import com.example.spellbound.model.Featured;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.Objects;  
  
public class ProductDetailActivity extends AppCompatActivity {  
 private static final String *TAG*="ProductDetailsActivity";  
 ActivityProductDetailBinding binding;  
 FirebaseFirestore firebaseFirestore;  
 FirebaseAuth firebaseAuth;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityProductDetailBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 firebaseAuth = FirebaseAuth.*getInstance*();  
 firebaseFirestore = FirebaseFirestore.*getInstance*();  
  
 UtilClass.*setUpCustomToolbar*(ProductDetailActivity.this);  
 getDataFromFeaturedBestSellerAllProductsRecyclerView();  
 showProductQtyCount();  
 }  
  
 private void showProductQtyCount() {  
 binding.btnIncreaseQty.setOnClickListener(v->{  
 String currentValue = binding.btnProductQty.getText().toString();  
 int qtyValue = Integer.*parseInt*(currentValue);  
 qtyValue++;  
 binding.btnProductQty.setText(String.*valueOf*(qtyValue));  
 });  
 binding.btnDecreaseQty.setOnClickListener(v -> {  
 String currentValue = binding.btnProductQty.getText().toString();  
 int qtyValue = Integer.*parseInt*(currentValue);  
 if(qtyValue>1) {  
 qtyValue--;  
 binding.btnProductQty.setText(String.*valueOf*(qtyValue));  
 }  
 });  
  
 }  
  
 private void getDataFromFeaturedBestSellerAllProductsRecyclerView() {  
 Featured featured = null;  
 BestSeller bestSeller = null;  
 AllProducts allProducts = null;  
  
 Object object = getIntent().getParcelableExtra("productDetails");  
 if(object instanceof Featured) {  
 featured = (Featured) object;  
 } else if (object instanceof BestSeller) {  
 bestSeller = (BestSeller) object;  
 } else if (object instanceof AllProducts) {  
 allProducts = (AllProducts) object;  
 }  
 validateObjectContainsNonNullValues(featured,bestSeller,allProducts);  
 navigationOfViews(featured,bestSeller,allProducts);  
 }  
 private void navigationOfViews(Featured featured, BestSeller bestSeller, AllProducts allProducts) {  
 binding.btnAddToCart.setOnClickListener(v -> {  
 String strQuantity = binding.btnProductQty.getText().toString();  
 if (!strQuantity.isEmpty()) {  
 int productQty = Integer.*parseInt*(strQuantity);  
 if(featured!=null) {  
 featured.setFeaturedProductQty(String.*valueOf*(productQty));  
 addProductToCart(featured);  
 } else if (bestSeller!=null) {  
 bestSeller.setBestSellerProductQty(String.*valueOf*(productQty));  
 addProductToCart(bestSeller);  
 } else if (allProducts!=null) {  
 allProducts.setProductQty(productQty);  
 addProductToCart(allProducts);  
 }  
 }  
 });  
 }  
 private void addProductToCart(Object product) {  
 firebaseFirestore.collection("Users")  
 .document(Objects.*requireNonNull*(firebaseAuth.getUid())).collection("Cart")  
 .add(product).addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 Toast.*makeText*(this, "Item Added to Cart", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 Toast.*makeText*(this, "Failed to Add the Item", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 }  
  
 private void validateObjectContainsNonNullValues(Featured featured, BestSeller bestSeller, AllProducts allProducts) {  
 if(featured != null) {  
 getDataFromFeatured(featured);  
 } else if (bestSeller != null) {  
 getDataFromBestSeller(bestSeller);  
 } else if(allProducts != null) {  
 getDataFromAllProducts(allProducts);  
 }  
 }  
  
  
 private void getDataFromAllProducts(AllProducts allProducts) {  
 Glide.*with*(ProductDetailActivity.this).load(allProducts.getProductImageURL())  
 .into(binding.ivProductItem);  
 String price = String.*valueOf*(allProducts.getProductPrice());  
 String priceWithSymbol = getString(R.string.*rupee\_sign*, price);  
 binding.tvProductPrice.setText(priceWithSymbol);  
 binding.tvProductName.setText(allProducts.getProductName());  
 binding.tvProductDescription.setText(allProducts.getProductDescription());  
 float rating = allProducts.getProductRating();  
 binding.rating.setRating(rating);  
 String rating\_txt = String.*valueOf*(allProducts.getProductRating());  
 binding.tvProductRating.setText(rating\_txt);  
 binding.tvProductAuthorName.setText(allProducts.getProductAuthorName());  
 binding.tvProductFormat.setText(allProducts.getProductFormat());  
 binding.tvProductIsbn.setText(allProducts.getProductISBN());  
 binding.tvProductPublisher.setText(allProducts.getProductPublished());  
 }  
  
  
 private void getDataFromBestSeller(BestSeller bestSeller) {  
 Glide.*with*(ProductDetailActivity.this).load(bestSeller.getBestSellerProductImageUrl())  
 .into(binding.ivProductItem);  
 String price = bestSeller.getBestSellerProductPrice();  
 String priceWithSymbol = getString(R.string.*rupee\_sign*, price);  
 binding.tvProductPrice.setText(priceWithSymbol);  
 binding.tvProductName.setText(bestSeller.getBestSellerProductName());  
 binding.tvProductDescription.setText(bestSeller.getBestSellerProductDescription());  
 float rating = bestSeller.getBestSellerProductRating();  
 binding.rating.setRating(rating);  
 String rating\_txt = String.*valueOf*(bestSeller.getBestSellerProductRating());  
 binding.tvProductRating.setText(rating\_txt);  
 binding.tvProductAuthorName.setText(bestSeller.getBestSellerAuthorName());  
 binding.tvProductFormat.setText(bestSeller.getBestSellerProductFormat());  
 binding.tvProductIsbn.setText(bestSeller.getBestSellerProductISBN());  
 binding.tvProductPublisher.setText(bestSeller.getBestSellerProductPublished());  
 }  
  
  
 private void getDataFromFeatured(Featured featured) {  
 Glide.*with*(ProductDetailActivity.this).load(featured.getFeaturedProductImageUrl())  
 .into(binding.ivProductItem);  
 String price = featured.getFeaturedProductPrice();  
 String priceWithSymbol = getString(R.string.*rupee\_sign*, price);  
 binding.tvProductPrice.setText(priceWithSymbol);  
 binding.tvProductName.setText(featured.getFeaturedProductName());  
 binding.tvProductDescription.setText(featured.getFeaturedProductDescription());  
 float rating = featured.getFeaturedProductRating();  
 binding.rating.setRating(rating);  
 String rating\_txt = String.*valueOf*(featured.getFeaturedProductRating());  
 binding.tvProductRating.setText(rating\_txt);  
 binding.tvProductAuthorName.setText(featured.getFeaturedAuthorName());  
 binding.tvProductFormat.setText(featured.getFeaturedProductFormat());  
 binding.tvProductIsbn.setText(featured.getFeaturedProductISBN());  
 binding.tvProductPublisher.setText(featured.getFeaturedProductPublished());  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*cart\_menu*,menu);  
 return super.onCreateOptionsMenu(menu);  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 if(item.getItemId()==R.id.*action\_go\_to\_cart*) {  
 UtilClass.*screenNavigation*(ProductDetailActivity.this, CartActivity.class);  
 }  
 return super.onOptionsItemSelected(item);  
 }

}

**rv\_cart\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="6dp">  
  
 <androidx.cardview.widget.CardView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:backgroundTint="@color/peach"  
 app:cardCornerRadius="17dp"  
 app:cardElevation="4dp">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="12dp">  
  
 <ImageView  
 android:id="@+id/iv\_cart\_item\_product"  
 android:layout\_width="90dp"  
 android:layout\_height="90dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="12dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <TextView  
 android:id="@+id/tv\_cart\_product\_name"  
 android:layout\_width="160dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="10dp"  
 android:ellipsize="end"  
 android:fontFamily="serif"  
 android:text="Product Name:"  
 android:textColor="@color/black"  
 app:layout\_constraintStart\_toEndOf="@+id/iv\_cart\_item\_product"  
 app:layout\_constraintTop\_toTopOf="@+id/iv\_cart\_item\_product" />  
  
 <TextView  
 android:id="@+id/tv\_cart\_product\_qty"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="22dp"  
 android:fontFamily="serif"  
 android:text="Qty"  
 android:textColor="@color/black"  
 app:layout\_constraintBottom\_toBottomOf="@+id/tv\_cart\_product\_name"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_cart\_product\_name"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_cart\_product\_name"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 <TextView  
 android:id="@+id/tv\_cart\_product\_price\_"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="15dp"  
 android:fontFamily="serif"  
 android:text="700"  
 android:textColor="@color/black"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_cart\_product\_name"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_cart\_product\_name"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_cart\_product\_name" />  
  
 <TextView  
 android:id="@+id/tv\_cart\_product\_total"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="12dp"  
 android:fontFamily="serif"  
 android:text="Product Total:"  
 android:textColor="@color/black"  
 app:layout\_constraintEnd\_toEndOf="@+id/tv\_cart\_product\_price\_"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="@+id/tv\_cart\_product\_price\_"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_cart\_product\_price\_" />  
  
 <TextView  
 android:id="@+id/tv\_cart\_product\_item\_remove"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="72dp"  
 android:fontFamily="serif"  
 android:text="Remove"  
 android:textColor="@android:color/holo\_red\_dark"  
 app:layout\_constraintBottom\_toBottomOf="@+id/tv\_cart\_product\_total"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_cart\_product\_total"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_cart\_product\_total"  
 app:layout\_constraintVertical\_bias="0.0" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
 </androidx.cardview.widget.CardView>  
</FrameLayout>

**CartItemAdapter:**

package com.example.spellbound.adapter;  
  
import android.view.LayoutInflater;  
import android.view.ViewGroup;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.bumptech.glide.Glide;  
import com.example.spellbound.FirestoreProducts;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.databinding.RvCartItemRowBinding;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.List;  
import java.util.Objects;  
  
public class CartItemAdapter extends RecyclerView.Adapter<CartItemAdapter.CartItemViewHolder> {  
  
 List<FirestoreProducts> firestoreProductsList;  
 FirebaseAuth firebaseAuth;  
 FirebaseFirestore firebaseFirestore;  
 ItemRemoved itemRemoved;  
 int quantity;  
 public CartItemAdapter(List<FirestoreProducts> firestoreProductsList,  
 FirebaseAuth firebaseAuth, FirebaseFirestore firebaseFirestore,  
 ItemRemoved itemRemoved, int quantity) {  
 this.firestoreProductsList = firestoreProductsList;  
 this.firebaseAuth = firebaseAuth;  
 this.firebaseFirestore = firebaseFirestore;  
 this.itemRemoved = itemRemoved;  
 this.quantity = quantity;  
 }  
  
 @NonNull  
 @Override  
 public CartItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 RvCartItemRowBinding binding= RvCartItemRowBinding.  
 *inflate*(LayoutInflater.*from*(parent.getContext()),parent,false);  
 return new CartItemViewHolder(binding);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull CartItemViewHolder holder, int position) {  
 if(firestoreProductsList!=null && position<firestoreProductsList.size()) {  
 int quantity = Integer.*parseInt*(firestoreProductsList.get(position).getFirestoreProductQty());  
 holder.binding.tvCartProductQty.setText("Qty: "+quantity);  
 }  
 assert firestoreProductsList!=null;  
 FirestoreProducts products=firestoreProductsList.get(position);  
 int productQty = Integer.*parseInt*(products.getFirestoreProductQty());  
 double productPrice = Double.*parseDouble*(products.getFirestoreProductPrice());  
  
 double totalAmountOfProduct = productQty \* productPrice;  
 holder.binding.tvCartProductTotal.setText("Total: "+totalAmountOfProduct);  
  
 holder.binding.tvCartProductName.setText(firestoreProductsList.get(position)  
 .getFirestoreProductName());  
 String price=String.*valueOf*(firestoreProductsList.get(position).getFirestoreProductPrice());  
 String priceWithSymbol = holder.itemView.getContext().getString(R.string.*rupee\_sign*,price);  
 holder.binding.tvCartProductPrice.setText(priceWithSymbol);  
  
 Glide.*with*(holder.itemView.getContext())  
 .load(firestoreProductsList.get(position).getFirestoreProductImageUrl())  
 .into(holder.binding.ivCartItemProduct);  
 holder.binding.tvCartProductItemRemove.setOnClickListener(v -> {  
 firebaseFirestore.collection("Users")  
 .document((Objects.*requireNonNull*(firebaseAuth.getCurrentUser())).getUid()).  
 collection("Cart")  
 .document(firestoreProductsList.get(holder.getAdapterPosition())  
 .getFirestoreProductId()).delete().addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 firestoreProductsList.remove(firestoreProductsList  
 .get(holder.getAdapterPosition()));  
 notifyItemRemoved(holder.getAdapterPosition());  
 itemRemoved.onItemRemoved(firestoreProductsList);  
 UtilClass.*snackBar*(v,"Product Removed");  
 } else {  
 Toast.*makeText*(holder.itemView.getContext(),  
 "Failed to Remove!",Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 });  
  
 }  
  
 @Override  
 public int getItemCount() {  
 return firestoreProductsList.size();  
 }  
  
 static class CartItemViewHolder extends RecyclerView.ViewHolder {  
 RvCartItemRowBinding binding;  
  
 public CartItemViewHolder(@NonNull RvCartItemRowBinding binding) {  
 super(binding.getRoot());  
 this.binding=binding;  
 }  
 }  
 public interface ItemRemoved {  
 void onItemRemoved(List<FirestoreProducts>itemList);  
 }  
}

**activity\_cart:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/bistre"  
 tools:context=".activity.CartActivity">  
  
 <Button  
 android:id="@+id/btn\_order\_now"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_margin="6dp"  
 android:textSize="17sp"  
 android:text="ORDER NOW"  
 android:fontFamily="serif"  
 app:backgroundTint="@null"  
 android:background="@drawable/gradient\_btn"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:cornerRadius="18dp"/>  
  
 <TextView  
 android:id="@+id/tv\_total\_value"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="6dp"  
 android:layout\_marginBottom="24dp"  
 android:fontFamily="serif"  
 android:textColor="@color/peach"  
 android:gravity="center\_horizontal"  
 android:text="Total: 1000"  
 android:textSize="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btn\_order\_now"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_cart"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/tv\_total\_value"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**CartActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import com.example.spellbound.FirestoreProducts;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.adapter.CartItemAdapter;  
import com.example.spellbound.databinding.ActivityCartBinding;  
import com.example.spellbound.model.AllProducts;  
import com.example.spellbound.model.BestSeller;  
import com.example.spellbound.model.Featured;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.DocumentChange;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class CartActivity extends AppCompatActivity {  
 FirestoreProducts fireStoreProducts;  
 FirebaseFirestore firebaseFirestore;  
 FirebaseAuth firebaseAuth;  
 List<FirestoreProducts> firestoreProductsList;  
 CartItemAdapter cartItemAdapter;  
 ActivityCartBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityCartBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 firebaseFirestore = FirebaseFirestore.*getInstance*();  
 firebaseAuth = FirebaseAuth.*getInstance*();  
  
 recyclerViewInit();  
 navigationOfViews();  
 getCartData();  
  
 }  
  
 private void getCartData() {  
 firebaseFirestore.collection("Users").document(Objects.*requireNonNull* (firebaseAuth.getCurrentUser()).getUid()).collection("Cart").get()  
 .addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 if(task.getResult()!=null) {  
 for(DocumentChange documentChange:task.getResult().getDocumentChanges())  
 {  
 String strDocumentId = documentChange.getDocument().getId();  
 if(documentChange.getDocument().contains("bestSellerProductName")) {  
 fireStoreProducts = documentChange.getDocument().toObject(BestSeller.class);  
 fireStoreProducts.setFirestoreProductId(strDocumentId);  
 firestoreProductsList.add(fireStoreProducts);  
 } else if (documentChange.getDocument().contains("featuredProductName")) {  
 fireStoreProducts = documentChange.getDocument().toObject(Featured.class);  
 fireStoreProducts.setFirestoreProductId(strDocumentId);  
 firestoreProductsList.add(fireStoreProducts);  
 } else {  
 fireStoreProducts = documentChange.getDocument().toObject(AllProducts.class);  
 fireStoreProducts.setFirestoreProductId(strDocumentId);  
 firestoreProductsList.add(fireStoreProducts);  
 }  
 }  
 calculateTotalAmountOfProducts(firestoreProductsList);  
 int quantity = getIntent().getIntExtra("productQty",1);  
 cartItemAdapter = new CartItemAdapter(firestoreProductsList,firebaseAuth,  
 firebaseFirestore,itemList ->calculateTotalAmountOfProducts(firestoreProductsList),quantity);  
 binding.rvCart.setAdapter(cartItemAdapter);  
 }  
 } else {  
 Toast.*makeText*(this, "Failed To Load Cart Data", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 private void calculateTotalAmountOfProducts(List<FirestoreProducts> firestoreProductsList) {  
 double totalAmount=0.0;  
 for(FirestoreProducts firestoreProducts: firestoreProductsList) {  
 int productQuantity=Integer.*parseInt*(firestoreProducts.getFirestoreProductQty());  
 double productPrice = Double.*parseDouble*(firestoreProducts.getFirestoreProductPrice());  
 totalAmount += productQuantity\*productPrice;  
 }  
 binding.tvTotalValue.setText("Overall Total Amount: "+ totalAmount);  
 }  
  
 private void navigationOfViews() {  
 binding.btnOrderNow.setOnClickListener(v -> {  
 UtilClass.*screenNavigationWithDataPassingSerialize*(CartActivity.this,  
 SelectAddressActivity.class,"firestoreProductsList", (Serializable) firestoreProductsList);  
 });  
 }  
  
 private void recyclerViewInit() {  
 firestoreProductsList = new ArrayList<>();  
 binding.rvCart.setLayoutManager(new LinearLayoutManager(this));  
 binding.rvCart.setHasFixedSize(true);  
 }  
}

**rv\_select\_address\_item\_row:**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="@drawable/rounded\_corners\_views"  
 app:cornerRadius="8dp"  
 android:id="@+id/single\_address\_item\_row"  
 android:layout\_margin="4dp" >  
  
 <TextView  
 android:id="@+id/tv\_address"  
 android:layout\_width="0dp"  
 android:layout\_height="20dp"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginBottom="20dp"  
 android:ellipsize="end"  
 android:maxLines="1"  
 android:text="TextView"  
 android:textColor="@color/peach"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/rb\_select"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.062"  
 tools:text="Brooklyn,New York" />  
  
 <RadioButton  
 android:id="@+id/rb\_select"  
 android:layout\_width="0dp"  
 android:layout\_height="20dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginBottom="20dp"  
 android:text="Select"  
 android:textColor="@color/peach"  
 android:fontFamily="serif"  
 app:layout\_constraintBottom\_toBottomOf="@+id/tv\_address"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.3"  
 app:layout\_constraintStart\_toEndOf="@+id/tv\_address"  
 app:layout\_constraintTop\_toTopOf="@+id/tv\_address"  
 app:layout\_constraintVertical\_bias="0.0" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**SelectedAddress**:

package com.example.spellbound.model;  
  
public class SelectedAddress {  
 private String address;  
 private boolean isSelected;  
  
 public SelectedAddress() {  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public boolean isSelected() {  
 return isSelected;  
 }  
  
 public void setSelected(boolean selected) {  
 isSelected = selected;  
 }  
}

**SelectedAddressAdapter:**

package com.example.spellbound.adapter;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.RadioButton;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.example.spellbound.R;  
import com.example.spellbound.model.SelectedAddress;  
  
import java.util.List;  
  
public class SelectedAddressAdapter extends RecyclerView.Adapter<SelectedAddressAdapter.SelectedAddressViewHolder> {  
 Context context;  
 RadioButton selectedRadioButton;  
 List<SelectedAddress> addressList;  
  
 public SelectedAddressAdapter(Context context, List<SelectedAddress> addressList) {  
 this.context = context;  
 this.addressList = addressList;  
 }  
  
 @NonNull  
 @Override  
 public SelectedAddressViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 View view = LayoutInflater.*from*(context).inflate(R.layout.*rv\_select\_address\_item\_row*,  
 parent,false);  
 return new SelectedAddressViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull SelectedAddressViewHolder holder, int position) {  
 holder.addressTv.setText(addressList.get(position).getAddress());  
 RadioButton radioButton=holder.selectRbAddress;  
 radioButton.setChecked(addressList.get(position).isSelected());  
 radioButton.setOnClickListener(view -> {  
 for(SelectedAddress selectedAddress:addressList) {  
 selectedAddress.setSelected(false);  
 }  
 validateRadioButton(view,position);  
 });  
 }  
  
 private void validateRadioButton(View view, int position) {  
 addressList.get(position).setSelected(true);  
 if(selectedRadioButton!=null && !view.equals(selectedRadioButton))  
 selectedRadioButton.setChecked(false);  
 selectedRadioButton = (RadioButton) view;  
 selectedRadioButton.setChecked(true);  
 }  
  
 @Override  
 public int getItemCount() {  
 return addressList.size();  
 }  
 static class SelectedAddressViewHolder extends RecyclerView.ViewHolder {  
 TextView addressTv;  
 RadioButton selectRbAddress;  
  
 public SelectedAddressViewHolder(@NonNull View itemView) {  
 super(itemView);  
 addressTv = itemView.findViewById(R.id.*tv\_address*);  
 selectRbAddress = itemView.findViewById(R.id.*rb\_select*);  
 }  
 }  
}

**activity\_select\_address:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/brown"  
 tools:context=".activity.SelectAddressActivity">  
  
 <include  
 android:id="@+id/toolbar\_custom"  
 layout="@layout/toolbar\_custom\_layout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"/>  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="20dp"  
 android:fontFamily="serif"  
 android:text="Address"  
 android:textColor="@color/peach"  
 android:textSize="20sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/toolbar\_custom"  
 app:layout\_goneMarginStart="10dp" />  
  
 <Button  
 android:id="@+id/btn\_add\_address"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="7dp"  
 android:text="Add Address"  
 android:backgroundTint="@color/bistre"  
 android:fontFamily="serif"  
 android:textColor="@color/white"  
 app:cornerRadius="8dp"  
 android:textSize="17sp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btn\_continue\_payment"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/btn\_continue\_payment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginEnd="10dp"  
 android:layout\_marginBottom="26dp"  
 app:backgroundTint="@null"  
 android:text="Continue to Payment"  
 android:textSize="17sp"  
 android:background="@drawable/gradient\_btn"  
 android:fontFamily="serif"  
 app:cornerRadius="8dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="@+id/btn\_add\_address"  
 app:layout\_constraintStart\_toStartOf="@+id/btn\_add\_address" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv\_address"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginEnd="10dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btn\_add\_address"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintVertical\_bias="0.046"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView3" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**SelectAddressActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import android.os.Bundle;  
  
import com.example.spellbound.FirestoreProducts;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.adapter.SelectedAddressAdapter;  
import com.example.spellbound.databinding.ActivitySelectAddressBinding;  
import com.example.spellbound.model.SelectedAddress;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.io.Serializable;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
public class SelectAddressActivity extends AppCompatActivity {  
 private FirebaseAuth firebaseAuth;  
 private FirebaseFirestore firebaseFirestore;  
 private SelectedAddressAdapter selectedAddressAdapter;  
 private List<SelectedAddress> addressList;  
 private List<FirestoreProducts> firestoreProductsList;  
 ActivitySelectAddressBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivitySelectAddressBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 firebaseAuth=FirebaseAuth.*getInstance*();  
 firebaseFirestore=FirebaseFirestore.*getInstance*();  
  
 UtilClass.*setUpCustomToolbar*(SelectAddressActivity.this);  
  
 getProductListFromCartActivity();  
 navigationOfViews();  
 initRecyclerView();  
 getAddressFromFireStore();  
 }  
  
 private void getAddressFromFireStore() {  
 firebaseFirestore.collection("Users").document(Objects  
 .*requireNonNull*(firebaseAuth.getCurrentUser()).getUid()).collection("Address")  
 .get().addOnCompleteListener(task->{  
 if(task.isSuccessful()) {  
 for(DocumentSnapshot documentSnapshot: task.getResult()) {  
 SelectedAddress showAddress=documentSnapshot.toObject(SelectedAddress.class);  
 addressList.add(showAddress);  
 selectedAddressAdapter.notifyItemInserted(Objects.*requireNonNull*(documentSnapshot.getData()).size());  
 }  
 }  
 });  
 }  
  
 private void initRecyclerView() {  
 addressList=new ArrayList<>();  
 binding.rvAddress.setLayoutManager(new LinearLayoutManager(SelectAddressActivity.this));  
 selectedAddressAdapter=new SelectedAddressAdapter(SelectAddressActivity.this,addressList);  
 binding.rvAddress.setAdapter(selectedAddressAdapter);  
 }  
  
 private void navigationOfViews() {  
 binding.btnAddAddress.setOnClickListener(view ->{  
 UtilClass.*screenNavigation*(SelectAddressActivity.this, AddAddressActivity.class);  
 finish();  
 });  
 binding.btnContinuePayment.setOnClickListener(view -> {  
 if(firestoreProductsList!=null && firestoreProductsList.size()>0) {  
 UtilClass.*screenNavigationWithDataPassingSerialize*(SelectAddressActivity.this,  
 PaymentActivity.class,"firestoreProductsList",(Serializable) firestoreProductsList);  
 }  
 });  
 }  
  
 private void getProductListFromCartActivity() {  
 Serializable serializableExtra = getIntent().getSerializableExtra("firestoreProductsList");  
 if(serializableExtra instanceof ArrayList) {  
 firestoreProductsList = (ArrayList<FirestoreProducts>) serializableExtra;  
 }  
 }  
}

**activity\_add\_address:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.core.widget.NestedScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/lion"  
 tools:context=".activity.AddAddressActivity">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" >  
  
 <include  
 android:id="@+id/toolbar\_custom"  
 layout="@layout/toolbar\_custom\_layout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/textView6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="12dp"  
 android:layout\_marginTop="16dp"  
 android:text="Add Address"  
 android:textColor="@color/bistre"  
 android:fontFamily="serif"  
 android:textSize="19sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.147"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout8"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="36dp"  
 android:layout\_marginBottom="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/textInputLayout6"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.494"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView6"  
 app:layout\_constraintVertical\_bias="0.022"  
 app:layout\_constraintVertical\_chainStyle="packed">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_add\_name"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Name"  
 android:fontFamily="serif"  
 android:textColorHint="@color/black"  
 android:textColor="@color/black"/>  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout6"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginBottom="20dp"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintBottom\_toTopOf="@+id/textInputLayout7"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout8"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout8"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout8">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_add\_full\_address"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPostalAddress"  
 android:textColorHint="@color/black"  
 android:textColor="@color/black"  
 android:hint="Full Address" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout7"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginBottom="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/textInputLayout9"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout6"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout6"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout6">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_add\_city"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textColorHint="@color/black"  
 android:textColor="@color/black"  
 android:hint="City" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout9"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:layout\_marginBottom="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/textInputLayout10"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout7"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout7"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout7">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_add\_postal\_code"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="number"  
 android:maxLength="6"  
 android:textColorHint="@color/black"  
 android:textColor="@color/black"  
 android:hint="Postal Code" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/textInputLayout10"  
 style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"  
 android:layout\_width="300dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/btn\_confirm\_address"  
 app:layout\_constraintEnd\_toEndOf="@+id/textInputLayout9"  
 app:layout\_constraintStart\_toStartOf="@+id/textInputLayout9"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout9">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/et\_add\_phone\_number"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="phone"  
 android:maxLength="10"  
 android:textColorHint="@color/black"  
 android:textColor="@color/black"  
 android:hint="Phone Number" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <Button  
 android:id="@+id/btn\_confirm\_address"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="54dp"  
 android:layout\_marginTop="37dp"  
 android:layout\_marginEnd="54dp"  
 android:layout\_marginBottom="128dp"  
 app:backgroundTint="@null"  
 android:textSize="17sp"  
 android:background="@drawable/gradient\_btn"  
 android:fontFamily="serif"  
 android:textColor="@color/white"  
 android:text="Confirm Address"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintTop\_toBottomOf="@+id/textInputLayout10" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</androidx.core.widget.NestedScrollView>

**AddAddressActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.databinding.ActivityAddAddressBinding;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Objects;  
  
public class AddAddressActivity extends AppCompatActivity {  
 private FirebaseAuth firebaseAuth;  
 private FirebaseFirestore firebaseFirestore;  
 ActivityAddAddressBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityAddAddressBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 firebaseAuth=FirebaseAuth.*getInstance*();  
 firebaseFirestore=FirebaseFirestore.*getInstance*();  
 UtilClass.*setUpCustomToolbar*(this);  
  
 navigationOfViews();  
 }  
  
 private void navigationOfViews() {  
 binding.btnConfirmAddress.setOnClickListener(view->{  
 if(!validateAndGetTheAddressFromUser().isEmpty()) {  
 Map<String,String> hashMapAddress = new HashMap<>();  
 String finalAddress = validateAndGetTheAddressFromUser();  
 hashMapAddress.put("address",finalAddress);  
 addAddressToFireStore(hashMapAddress);  
 }  
 });  
 }  
 private void addAddressToFireStore(Map<String, String> hashMapAddress) {  
 if(hashMapAddress.containsKey("address")) {  
 firebaseFirestore.collection("Users")  
 .document(Objects.*requireNonNull*(firebaseAuth.getCurrentUser()).getUid())  
 .collection("Address").add(hashMapAddress)  
 .addOnCompleteListener(task -> {  
 if(task.isSuccessful()) {  
 Toast.*makeText*(this, "Address Added", Toast.*LENGTH\_SHORT*).show();  
 UtilClass.*screenNavigation*(AddAddressActivity.this, SelectAddressActivity.class);  
 finish();  
 }  
 });  
 }  
 }  
  
 private String validateAndGetTheAddressFromUser() {  
 String name = Objects.*requireNonNull*(binding.etAddName.getText()).toString();  
 String address = Objects.*requireNonNull*(binding.etAddFullAddress.getText()).toString();  
 String city = Objects.*requireNonNull*(binding.etAddCity.getText()).toString();  
 String postalCode = Objects.*requireNonNull*(binding.etAddPostalCode.getText()).toString();  
 String phoneNumber = Objects.*requireNonNull*(binding.etAddPhoneNumber.getText()).toString();  
 String finalAddress ="";  
  
 if(name.isEmpty()) {  
 Toast.*makeText*(this, "Fill All Details", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(address.isEmpty()) {  
 Toast.*makeText*(this, "Fill All Details", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(city.isEmpty()) {  
 Toast.*makeText*(this, "Fill All Details", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(postalCode.isEmpty()) {  
 Toast.*makeText*(this, "Fill All Details", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if (phoneNumber.isEmpty()) {  
 Toast.*makeText*(this, "Fill All Details", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(postalCode.length()<6) {  
 Toast.*makeText*(this, "Postal Code is Incorrect", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(phoneNumber.length()<6) {  
 Toast.*makeText*(this, "Phone Number is Incorrect", Toast.*LENGTH\_SHORT*).show();  
 }  
 else {  
 finalAddress=name+", "+address+", "+city+", "+postalCode+", "+phoneNumber;  
 }  
 return finalAddress;  
 }  
}

**activity\_payment:**

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/brown"  
 tools:context=".activity.PaymentActivity">  
  
 <include  
 android:id="@+id/toolbar\_custom"  
 layout="@layout/toolbar\_custom\_layout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <TextView  
 android:id="@+id/textView8"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:text="Payment"  
 android:textColor="@color/peach"  
 android:fontFamily="serif"  
 android:textSize="25sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.087"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/toolbar\_custom" />  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="280dp"  
 android:layout\_marginStart="10dp"  
 android:layout\_marginTop="60dp"  
 android:layout\_marginEnd="10dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView8"  
 app:srcCompat="@drawable/credit\_card" />  
  
 <TextView  
 android:id="@+id/tv\_total"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="56dp"  
 android:layout\_marginTop="40dp"  
 android:text="Total ->"  
 android:textColor="@color/peach"  
 android:fontFamily="serif"  
 android:textSize="19sp"  
 app:layout\_constraintEnd\_toStartOf="@+id/tv\_payment\_total\_value"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.034"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView" />  
  
 <TextView  
 android:id="@+id/tv\_payment\_total\_value"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="40dp"  
 android:layout\_marginEnd="68dp"  
 android:text="2000"  
 android:textSize="19sp"  
 android:textColor="@color/peach"  
 android:fontFamily="serif"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView" />  
  
 <Button  
 android:id="@+id/btn\_pay\_now"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="12dp"  
 android:background="@drawable/gradient\_btn"  
 android:fontFamily="serif"  
 android:text="Pay Now"  
 android:textSize="18sp"  
 app:backgroundTint="@null"  
 app:layout\_constraintEnd\_toEndOf="@+id/divider5"  
 app:layout\_constraintStart\_toStartOf="@+id/divider5"  
 app:layout\_constraintTop\_toBottomOf="@+id/divider5" />  
  
 <View  
 android:id="@+id/divider5"  
 android:layout\_width="409dp"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="190dp"  
 android:layout\_marginBottom="10dp"  
 android:background="?android:attr/listDivider"  
 app:layout\_constraintBottom\_toTopOf="@+id/btn\_pay\_now"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.217"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView"  
 app:layout\_constraintVertical\_bias="1.0" />  
</androidx.constraintlayout.widget.ConstraintLayout>

**activity\_payment:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import com.example.spellbound.FirestoreProducts;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.databinding.ActivityPaymentBinding;  
  
import java.util.List;  
  
public class PaymentActivity extends AppCompatActivity {  
 List<FirestoreProducts> allProductsList=null;  
 ActivityPaymentBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityPaymentBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 UtilClass.*setUpCustomToolbar*(PaymentActivity.this);  
 calculateTotalAmountOfProductsFromSelectedAddressActivity();  
 navigationOfViews();  
 }  
  
 private void navigationOfViews() {  
 binding.btnPayNow.setOnClickListener(view -> {  
 Toast.*makeText*(this, "Transaction Successful", Toast.*LENGTH\_SHORT*).show();  
 });  
 }  
  
 private void calculateTotalAmountOfProductsFromSelectedAddressActivity() {  
 allProductsList=(List<FirestoreProducts>)getIntent().getSerializableExtra("firestoreProductsList");  
  
 double totalAmount=0.0;  
 assert allProductsList!=null;  
 for(FirestoreProducts firestoreProducts:allProductsList) {  
 int productQty = Integer.*parseInt*(firestoreProducts.getFirestoreProductQty());  
 double productPrice=Double.*parseDouble*(firestoreProducts.getFirestoreProductPrice());  
 totalAmount+=productQty\*productPrice;  
 }  
 binding.tvPaymentTotalValue.setText(String.*valueOf*(totalAmount));  
 }  
}

**PaymentActivity:**

package com.example.spellbound.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import com.example.spellbound.FirestoreProducts;  
import com.example.spellbound.R;  
import com.example.spellbound.UtilClass;  
import com.example.spellbound.databinding.ActivityPaymentBinding;  
  
import java.util.List;  
  
public class PaymentActivity extends AppCompatActivity {  
 List<FirestoreProducts> allProductsList=null;  
 ActivityPaymentBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding=ActivityPaymentBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 UtilClass.*setUpCustomToolbar*(PaymentActivity.this);  
 calculateTotalAmountOfProductsFromSelectedAddressActivity();  
 navigationOfViews();  
 }  
  
 private void navigationOfViews() {  
 binding.btnPayNow.setOnClickListener(view -> {  
 Toast.*makeText*(this, "Transaction Successful", Toast.*LENGTH\_SHORT*).show();  
 });  
 }  
  
 private void calculateTotalAmountOfProductsFromSelectedAddressActivity() {  
 allProductsList=(List<FirestoreProducts>)getIntent().getSerializableExtra("firestoreProductsList");  
  
 double totalAmount=0.0;  
 assert allProductsList!=null;  
 for(FirestoreProducts firestoreProducts:allProductsList) {  
 int productQty = Integer.*parseInt*(firestoreProducts.getFirestoreProductQty());  
 double productPrice=Double.*parseDouble*(firestoreProducts.getFirestoreProductPrice());  
 totalAmount+=productQty\*productPrice;  
 }  
 binding.tvPaymentTotalValue.setText(String.*valueOf*(totalAmount));  
 }

**toolbar\_custom\_layout:**

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.appcompat.widget.Toolbar xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:minHeight="?actionBarSize"  
 android:id="@+id/toolbar\_custom">  
  
</androidx.appcompat.widget.Toolbar>

**UtilClass:**

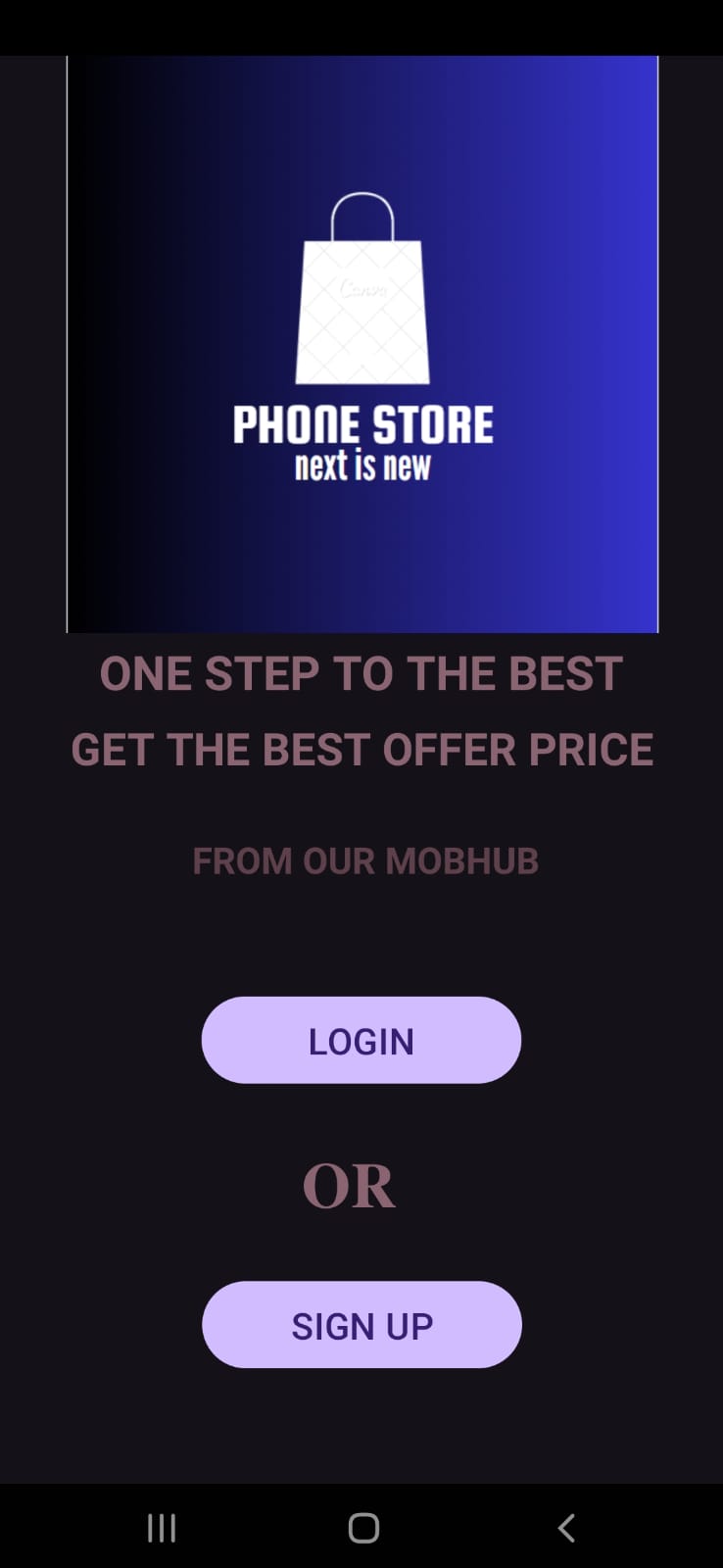
package com.example.spellbound;  
  
import android.content.Context;  
import android.content.Intent;  
import android.os.Parcelable;  
import android.util.Log;  
import android.view.View;  
  
import androidx.appcompat.app.ActionBar;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
  
import com.example.spellbound.adapter.AllProductsAdapter;  
import com.example.spellbound.model.AllProducts;  
import com.google.android.material.snackbar.Snackbar;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
  
import java.io.Serializable;  
import java.util.List;  
import java.util.Objects;  
  
public class UtilClass {  
 private static final String *TAG* = "showDataBasedOnTheGenreType";  
  
  
 public UtilClass() {  
 *//to avoid accidental creation of object* }  
  
 public static void screenNavigation(Context context, Class<?> destination)  
 {  
 Intent intent=new Intent(context,destination);  
 context.startActivity(intent);  
 intent.addFlags(intent.*FLAG\_ACTIVITY\_SINGLE\_TOP*);  
 }  
  
 public static void snackBar(View view, String stringText){  
 Snackbar.*make*(view,stringText,Snackbar.*LENGTH\_LONG*).show();  
 }  
 public static void setUpCustomToolbar(AppCompatActivity activity) {  
 Toolbar toolbar= activity.findViewById(R.id.*toolbar\_custom*);  
 activity.setSupportActionBar(toolbar);  
 activity.setTitle("");  
  
 ActionBar actionBar = activity.getSupportActionBar();  
  
 if(actionBar != null) {  
 actionBar.setDisplayHomeAsUpEnabled(true);  
 }  
 toolbar.setNavigationOnClickListener(v -> {  
 activity.getOnBackPressedDispatcher().onBackPressed();  
 activity.finish();  
 });  
 }  
 public static void screenNavigationWithDataPassing(Context context, Class<?> destination, String key, String value) {  
 Intent intent= new Intent(context, destination);  
 intent.putExtra(key, value);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_SINGLE\_TOP*);  
 context.startActivity(intent);  
 }  
 public static void screenNavigationWithDataPassing(Context context, Class<?> destination, String key, Parcelable value) {  
 Intent intent=new Intent(context, destination);  
 intent.putExtra(key,value);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);  
 context.startActivity(intent);  
 }  
 public static void screenNavigationWithDataPassingSerialize(Context context, Class<?> destination, String key, Serializable value) {  
 Intent intent=new Intent(context,destination);  
 intent.putExtra(key,value);  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);  
 context.startActivity(intent);  
 }  
 public static void showDataBasedOnTheGenreType(String key, String value,  
 AllProductsAdapter allProductAdapter, List<AllProducts> allProductsList) {  
  
 FirebaseFirestore firebaseFirestore;  
 firebaseFirestore = FirebaseFirestore.*getInstance*();  
 firebaseFirestore.collection("AllProducts").whereEqualTo(key,value)  
 .get().addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (DocumentSnapshot documentSnapshot: task.getResult()) {  
 AllProducts allProducts = documentSnapshot.toObject(AllProducts.class);  
 allProductsList.add(allProducts);  
 allProductAdapter.notifyItemInserted(Objects.  
 *requireNonNull*(documentSnapshot.getData()).size());  
 }  
 } else {  
 Log.*e*(*TAG*, "getDataFromAllProducts: ",task.getException());  
 }  
 });  
 }  
}

**FirestoreProducts:**

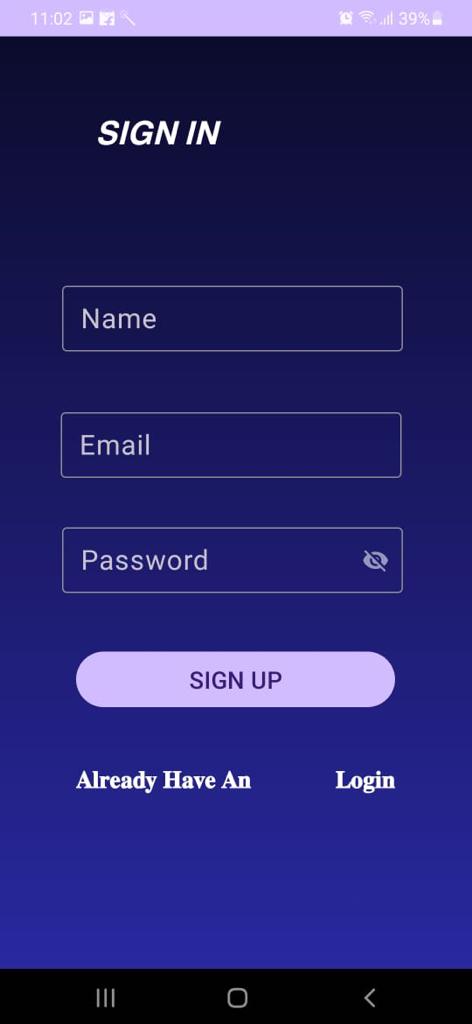
package com.example.spellbound;  
  
import java.io.Serializable;  
  
public interface FirestoreProducts extends Serializable {  
 String getFirestoreProductId();  
 void setFirestoreProductId(String setId);  
 String getFirestoreProductPrice();  
 String getFirestoreProductImageUrl();  
 String getFirestoreProductName();  
 String getFirestoreProductQty();  
  
}

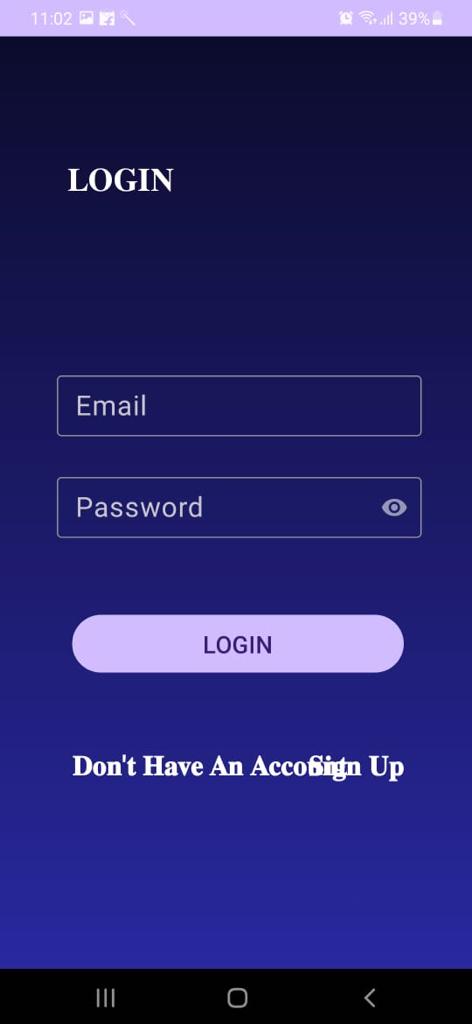
**8.2 SCREENSHOTS**

**SPLASH SCREEN WELCOME SCREEN**

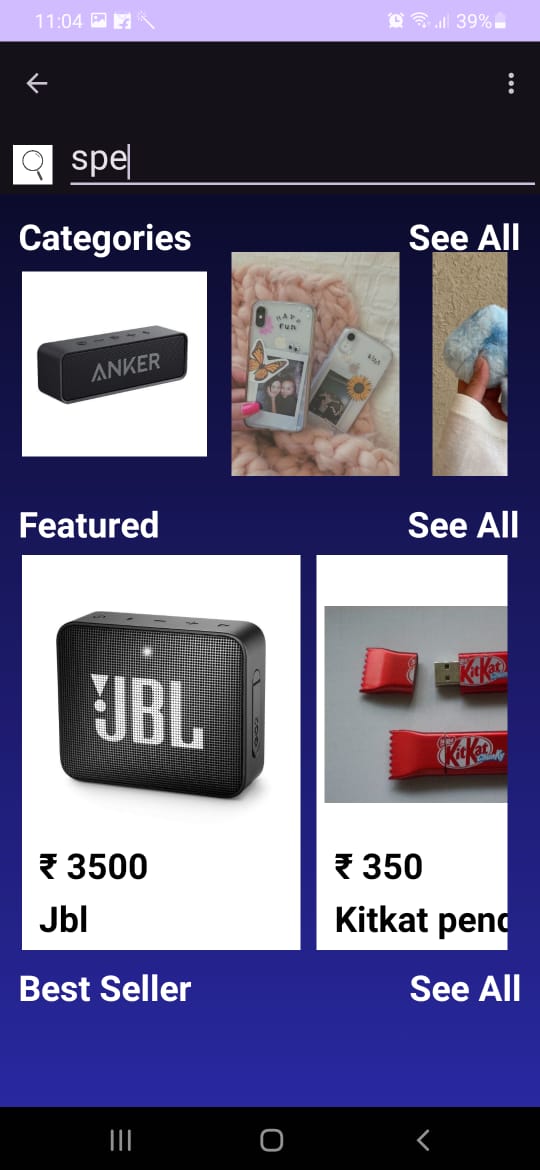
 

**LOGIN SCREEN SIGN UP SCREEN**



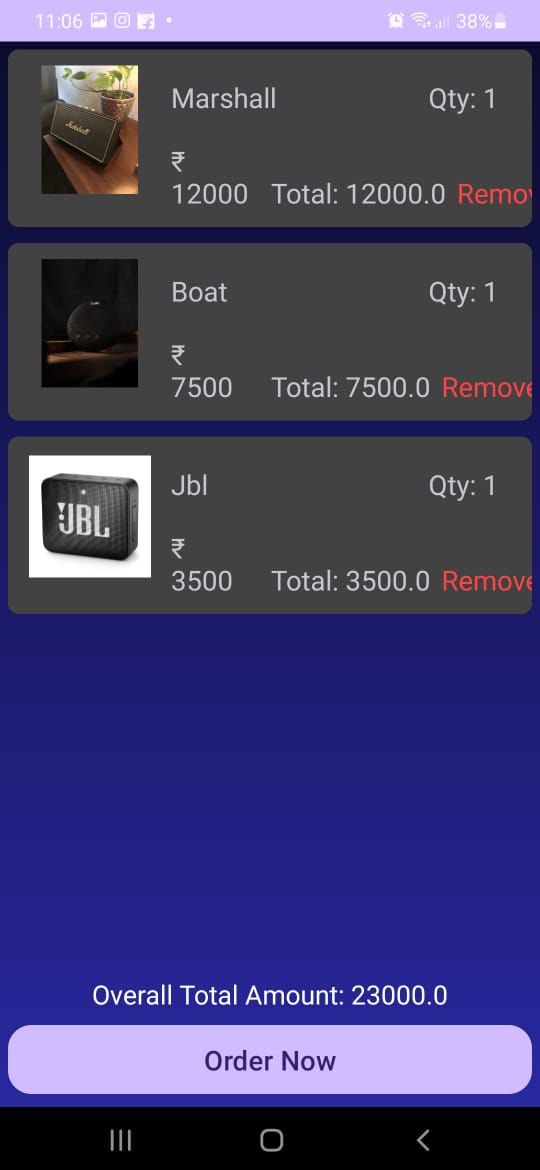


**HOME SCREEN PRODUCT DETAIL SCREEN**



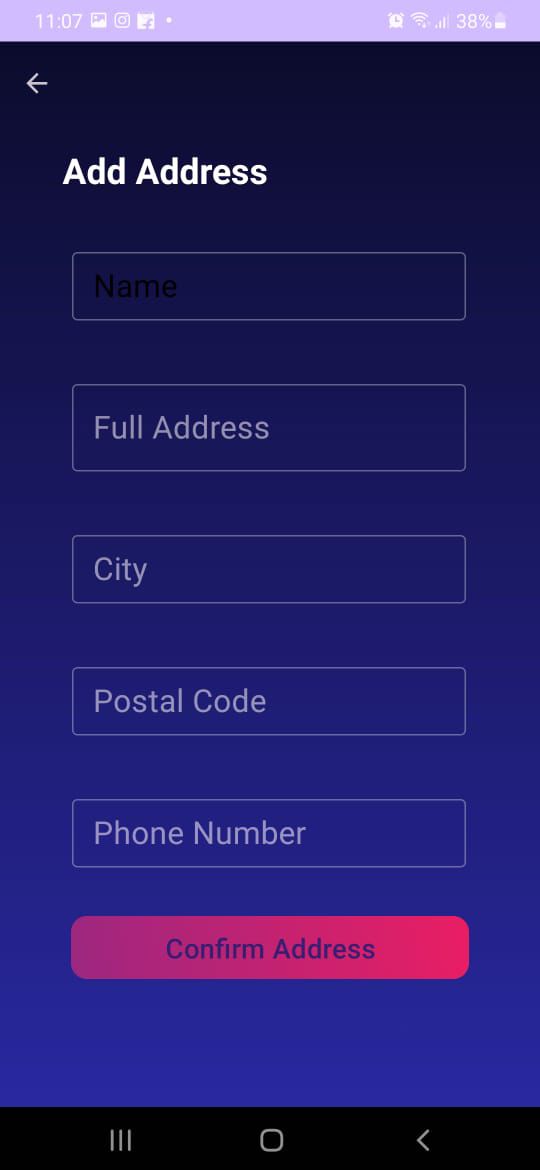
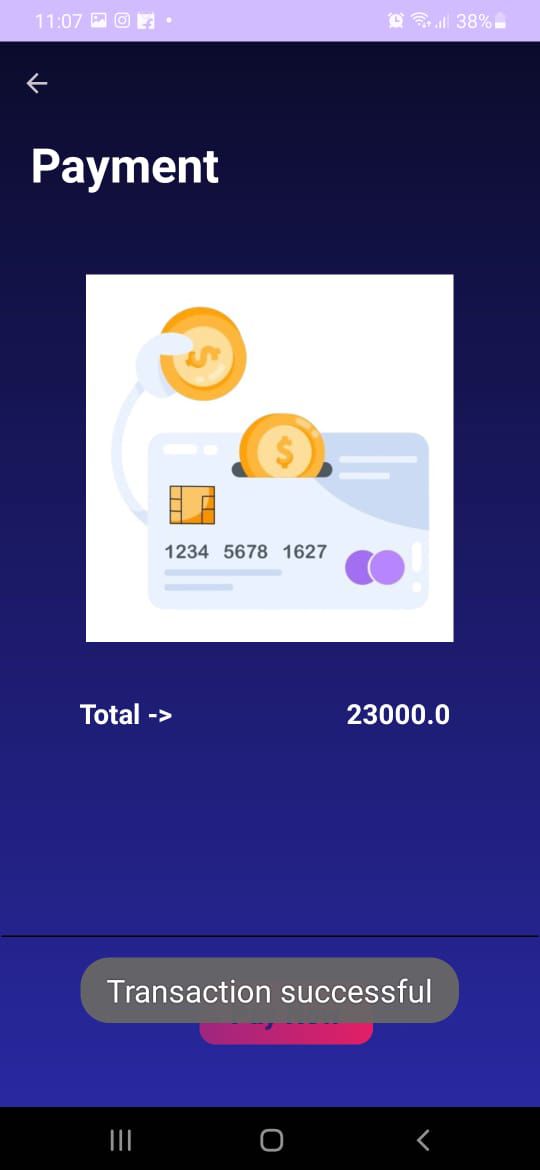


**CART SCREEN SELECT ADDRESS SCREEN**





**ADD ADDRESS SCREEN PAYMENT SCREEN**

**9.CONCLUSION & FUTURE ENHANCEMENT**

**CONCLUSION:**

"In conclusion, the mobile accessories app is designed to enhance the user experience by providing a wide range of accessories for smartphones and other devices. With features such as easy navigation, secure payment options, personalized recommendations, and real-time updates on new products, the app aims to simplify the process of finding and purchasing accessories. By offering convenience, affordability, and quality products, the app not only meets the needs of users but also contributes to the growth of the mobile accessories market. With continued updates and enhancements, we are committed to delivering an exceptional shopping experience for our users."

**FUTURE ENHANCEMENT**

Here are some potential future enhancements for a mobile accessories app:

1.Augmented Reality (AR) Integration: Allow users to visualize how accessories such as phone cases, screen protectors, or earphones would look on their devices using AR technology.

2.Customization Options: Offer personalized accessories by allowing users to choose colors, materials, or add custom designs or engravings.

3.Interactive Tutorials: Include interactive guides or tutorials on how to use and install accessories correctly, enhancing user experience and reducing returns due to installation issues.

4.Social Sharing: Enable users to share their purchases or wishlists on social media platforms, encouraging engagement and word-of-mouth promotion.

5.Subscription Services: Introduce subscription plans for regular accessory updates, discounts, or exclusive offers, providing value to loyal customers.

These enhancements can improve user engagement, satisfaction, and overall competitiveness in the mobile accessories market.

**10.BIBILOGRAPHY & WEBLIOGRAPHY**

**WEBSITES:**

[www.google.com](http://www.google.com)

[www.androidstudio.com](http://www.androidstudio.com)

[www.github.com](http://www.github.com)

[www.goodreads.com](http://www.goodreads.com)