**MINISTRY OF EDUCATION AND TRAINING**

**HO CHI MINH CITY OPEN UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**

---□ & □---

A blue book with a ring around it

Description automatically generated

**REPORT**

**COURSE: PROGRAMMING ON MOBILE DEVICES**

Topic:

**ADVERTISING APPLICATION, ONLINE BOOK SELLING**

**Student implementation: Hồ Ngọc Giang – 2051012024**

**Trần Duy Khang -2051012044**

**Lê Phi Long - 2051012056**

**Nguyễn Thị Ý Nhi - 2051012082**

**Trần Khải Quân – 2051012096**

**Phan Thị Hồng Quế - 2051012097**

**Instructor: ThS. Lê Ngọc Hiếu**

**Ho Chi Minh City, August 2023**

**LIST OF GROUP MEMBERS AND WORK ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **WORK DISTRIBUTION TABLE** | | |
| **First and last name** | **MSSV** | **Assignment** |
| 1. Hồ Ngọc Giang | 2051012024 | Build databases and design APIs |
| 2. Trần Duy Khang | 2051012044 | Handling the function of adding products to the cart, designing the catalog interface |
| 3. Lê Phi Long | 2051012056 | Design interface for registration, login, homepage, categories, handling API calls to display products, authors, promotions and search functions |
| 4. Nguyễn Thị Ý Nhi | 2051012082 | Handles registration, login, personal profile page functions |
| 5. Trần Khải Quân | 2051012096 | Handling the function of adding products to the cart, designing the catalog interface |
| 6. Phan Thị Hồng Quế | 2051012097 | Handles registration, login, personal profile page functions |
| **\*Note:** Besides the main assigned tasks, team members still fully participate and contribute ideas. | | |

Table of contents

[1. Business Process Description 4](#_Toc144219855)

[1.1. Registration 4](#_Toc144219856)

[1.2. Login 4](#_Toc144219857)

[1.3. Book Search 4](#_Toc144219858)

[1.4. Display Advertising Slides 4](#_Toc144219859)

[1.5. Display Prominent Author Pages and Highlighted Books 4](#_Toc144219860)

[1.6. Display Discounts, Percentage-Based Discounts 5](#_Toc144219861)

[1.7. Filter Books by Category in the Catalog 5](#_Toc144219862)

[1.8. Customize Personal Profile Information 5](#_Toc144219863)

[1.9. Adding Books to the Shopping Cart 5](#_Toc144219864)

[2. Technical Documentation: 6](#_Toc144219865)

[2.1. Database and API Design 6](#_Toc144219866)

[2.2. Registration and Login Functionality 11](#_Toc144219867)

[2.3. Displaying Product Advertisement Slides 18](#_Toc144219868)

[2.4. Function to display authors, books, promotions 21](#_Toc144219869)

[2.5. Product search function 25](#_Toc144219870)

[2.6. Function to sort books by genre in the category 28](#_Toc144219871)

[2.7. Customizing User Profile Information Feature 31](#_Toc144219872)

[2.8. Function to add books to cart 33](#_Toc144219873)

[3. Instructions for Use: 35](#_Toc144219874)

[3.1. Registration and Login 35](#_Toc144219875)

[3.2. Edit Profile 35](#_Toc144219876)

[3.3. Select Products and Add to Cart 36](#_Toc144219877)

[3.4. Manage Cart 36](#_Toc144219878)

[3.5. Checkout 36](#_Toc144219879)

## 1. Business Process Description

### 1.1. Registration

Register is the process that allows users to create a personal account on the application in order to use the features and services of the application. This business process includes:

**Entering personal information:** On the Registration screen, users will be required to provide necessary information to create an account. Information includes: Account name, email address, password, re-enter password.

**Registration confirmation:** Users select the "Register" button after reviewing the information and ensuring that all information is accurate.

**Successful account creation:** After users confirm and successfully create an account, the application will provide a notification and redirect to the Login screen for users to log in with the newly created account.

**Error notifications (if applicable):** In case users do not enter complete or valid information, the application will display an error notification and ask users to make adjustments.

### 1.2. Login

Login is the process that allows users to access their personal accounts by entering the login information registered earlier. This business process includes:

**Entering login information:** Users will be required to enter login information to access their accounts. This information includes: email address and password of the registered account.

**Login confirmation:** Users select the "Login" button after entering the correct login information.

**Information verification:** The application will verify the login information and password for accuracy. If the information does not match, users will receive an error notification.

**Successful login:** In case of correct information, users will be redirected to the main screen of the book-selling app.

### 1.3. Book Search

Users can use this feature to search for the names of books they desire, enabling quick and time-saving searches.

### 1.4. Display Advertising Slides

Advertising slides will be displayed on the homepage to promote various events and highlighted books.

### 1.5. Display Prominent Author Pages and Highlighted Books

Display the names of prominent authors and highlighted books on the homepage to help customers see them when they start using the app.

### 1.6. Display Discounts, Percentage-Based Discounts

Each book will come with an accompanying discount price, displayed next to the original price of the book.

### 1.7. Filter Books by Category in the Catalog

The catalog consists of primary classifications of books based on criteria such as genre, author, price, illustrated images, etc. This section's tasks include:

**Displaying the list of books:** The catalog will display a list of different types of books or specific topics for customers to search and browse.

**Search bar:** Users can use the search bar to quickly find books they desire, saving time.

**Selecting a specific category:** When users choose a specific category, they will be directed to a page listing the books belonging to that category.

### 1.8. Customize Personal Profile Information

Edit Profile is the process that allows users to update and adjust their personal information on their account. This section's tasks include:

**Displaying information:** Before editing, the user's current personal information will be displayed, including information provided during registration, with empty fields for missing information.

**Editing information:** Users can adjust the following personal information: Username, change password, change phone number, change email, gender, date of birth.

**Confirming changes:** When users have accurately adjusted their personal information, they press the "Save" button, and the system will save the edited information.

### 1.9. Adding Books to the Shopping Cart

The shopping cart is where users can view the books they have selected to purchase and the total amount they need to pay. This section's tasks include:

**Adding and removing products:** Users can add books to the shopping cart by clicking the "Buy" button on the product page. They can also remove products from the cart if needed.

**Calculating the total amount:** The system will calculate the total amount based on the quantity and price of each book in the shopping cart.

**Checking information:** Users can review the products in the shopping cart and the total amount before proceeding to payment.

**Payment:** After reviewing the information, users can proceed with the payment.

Both the "category" and "cart" sections aim to create a convenient, intuitive, and enjoyable shopping experience for customers, from browsing books to managing ord

## 2. Technical Documentation:

### 2.1. Database and API Design

#### **2.1.1. Overview of Spring Boot**

In this report, we will learn about building an API using Spring Boot and MySQL. Spring Boot is a widely used Java application development framework for building web services. MySQL is a popular database management system used for storing data.

##### **2.1.1.1. Spring MVC Model**

- Consists of 3 main components:

• Controller: Responsible for navigating user requests, containing business logic, handling requests from the client, storing data into the database, web service, etc.

• Model: Contains logic tasks, processing methods, database retrieval, processing functions, classes, etc.

• View: Displays interactive information to users, collecting forms or HTML files, CSS, templates, etc.

A diagram of a database

Description automatically generated

*Hình 1.1 Mô hình MVC*

##### **2.1.1.2 Introduction to Spring Boot**

- Spring Boot is a project situated within the IO Execution layer of the Spring MVC Framework.

- Spring Boot was created to make programming and application development easier. It supports embedded containers, allowing web applications to run independently without the need for deployment on external web servers.

- Therefore, Spring Boot includes both the Spring Framework (Spring MVC) and an Embedded HTTP server (Tomcat, servlet, etc.). The advantage of Spring Boot is that it eliminates the need for XML configuration, minimizing time and enhancing productivity.

##### **2.1.1.3 Initializing a Spring Boot Project:**

- To get started, you need to initialize a Spring Boot project. You can use Spring Initializr or similar tools to create a basic Spring Boot project.

##### **2.1.1.4 Configuring MySQL Database Connection:**

In the application.properties or application.yml file, you need to provide the connection information for the MySQL database, including URL, username, and password.

Example configuration in the application.properties file:

A screen shot of a computer

Description automatically generated

##### **2.1.1.5 Creating Entity and Repository:**

Use annotations like @Entity, @Table, @Id, @Column to define Entity objects in your application. These are Java classes that represent tables in the database.

Use @Repository to annotate Repository interfaces, which contain methods for interacting with the database.

A screen shot of a computer code

Description automatically generatedExample:

Image 1.5.1: Create @Entity

Image 1.5.2: Create @Repository

##### **2.1.1.6 Creating Services:**

Create Service classes to handle business logic and invoke methods from the Repository to interact with the database.

A screenshot of a computer program

Description automatically generatedExample:

Image 1.6: Book’s Class Service

##### **2.1.1.7 Using CRUD Methods:**

- Spring Boot provides built-in CRUD methods through JpaRepository. You can use methods like save(), findById(), findAll(), deleteById(), etc., to interact with the database.

A computer screen shot of text

Description automatically generated-Example:

Image 1.7: Controller utilizing CRUD methods

##### **2.1.1.8. API Methods Used for the Large Project:**

A computer screen shot of text

Description automatically generatedGet the list of all books: https://bookapi.up.railway.app/api/book/getAll

A computer screen with text

Description automatically generated• Search for a book by Id: https://book-api.up.railway.app/api/book/2

A screen shot of a computer program

Description automatically generated• Get all categories: https://book-api.up.railway.app/api/category/getAll

A screen shot of a computer code

Description automatically generated• Search by category: https://bookapi.up.railway.app/api/book/getByCategory/3222

#### **2.1.2. Overview of MySQL Workbench**

##### **2.1.2.1. Introduction:**

MySQL Workbench is an open-source database management tool developed by Oracle. It provides a graphical interface for designing databases, managing and interacting with MySQL databases. This report will introduce MySQL Workbench and its key features.

##### **2.1.2.2. Key Features of MySQL Workbench:**

- MySQL Workbench offers an Entity-Relationship (ER) design tool that allows you to create, edit, and manage database structures. You can create tables, relationships, indexes, and foreign key constraints in a visual, intuitive environment.

- MySQL Workbench enables you to execute SQL queries and interact with MySQL databases. You can write and run queries, insert, edit, and delete data, and view query results in a user-friendly graphical interface.

- Workbench provides tools for managing users, roles, and permissions within the MySQL database. You can create new users, grant and revoke roles, set access privileges for each user, and control data access.

- MySQL Workbench allows you to perform database backup and recovery. You can create backups of the entire database or specific tables, and restore data from the created backups.

- Workbench offers tools to view and manage the status of the MySQL system, including information about the MySQL version, system resources, processes, and current connections.

##### **A screenshot of a computer Description automatically generated2.1.2.3. ERD Model Used in the System:**

Image 2.3: ERD Model

#### **2.1.3. Deploying the Application with Railway**

##### **2.1.3.1. Introduction**

Railway is a cloud-based application deployment platform that helps developers build, deploy, and manage applications with ease. This report will outline the process of deploying an application using Railway and the benefits of using this platform.

##### **2.1.3.2. Deployment Process with Railway**

• Register and Create a Project:

- Register an account on Railway (https://railway.app/) if not already done.

- Create a new project on Railway to initiate the deployment process.

• Connect Project to Repository:

- Link the project to your repository on GitHub or GitLab.

- Allow Railway to access the repository to automate deployment from source code.

• Set Up Environments:

- Create a new environment within the project and name it.

- Configure necessary environment variables, such as database configurations, secrets, etc.

• Configure railway.yaml File:

Create or edit the railway.yaml file in your repository.

- Define the required steps to deploy the application, including dependencies installation, building, running commands, etc. Use YAML syntax to describe these steps.

• Deploy the Application:

- Navigate to the project page on Railway.

- Select the environment you created and click the "Deploy" button to initiate the deployment process.

- Railway will automatically fetch source code from the repository and deploy the application based on the configuration in railway.yaml.

• Monitor Deployment Progress:

- Railway provides an interface for you to monitor the deployment progress and view the ongoing steps.

- You can view errors, messages, and logs from the deployment process to identify and resolve issues (if any).

• Test the Application:

- Once the deployment process is complete, you can access your application using the link provided by Railway.

##### **2.1.3.3. Benefits of Using Railway**

* Easy Deployment: Railway offers a simple and automated deployment process from repositories, reducing effort and time required to deploy applications.
* Environment Management: Railway allows you to create and manage different environments for your application, facilitating testing, development, and easy deployment.
* Seamless Integration: Railway integrates well with popular repositories like GitHub and GitLab, enabling you to automate deployment processes from source code.
* Environment Variables Management: Railway allows you to manage environment variables, including database configurations, secrets, and other values, ensuring security and flexibility in application deployment.
* Monitoring and Debugging: The Railway platform provides an interface to monitor deployment progress and view messages, logs, and errors. This aids in easy issue identification and quick fixes.

##### **2.1.3.4. Conclusion:**

- Railway is a powerful and convenient cloud-based application deployment platform. Throughout this report, we have presented the process of deploying applications using Railway and the benefits of using this platform. By utilizing Railway, you can easily deploy applications, automate processes, and efficiently manage environments.

### 2.2. Registration and Login Functionality

#### **2.2.1. Introduction to Firebase**

Firebase is a cloud-based database service coupled with Google's powerful server infrastructure. Its primary function is to simplify database operations for users when programming applications. It achieves this through providing simple APIs for application development. The goal is to increase user numbers and generate more revenue.

Notably, it also offers versatile and strong security services. Firebase supports both Android and iOS platforms. It's no surprise that many developers choose Firebase as their primary platform to build applications for millions of users worldwide.

Advantages of using Firebase: Easy account creation and use; Rapid development speed; Multiple services in a single platform; Provided by Google; Focus on developing user interfaces; Firebase eliminates the need for a server.

Firebase offers a wide array of services, but for this project, we will only use the following three services for handling registration and login data:

- Authentication: This service manages user accounts simply and securely. Authentication offers multiple authentication methods such as email and password, Google, and Facebook.

- Realtime Database: This service stores and synchronizes user data in real-time. It supports Android, iOS, Web, C++, Unity, and Xamarin. Users can easily store and retrieve data from the server.

- Cloud Storage: This service is capable of storing and sharing user-generated content such as images, audio, and video with powerful, straightforward, and cost-effective memory, built to Google's scale.

#### **2.2.2. Connecting to Firebase**

Connecting Firebase in Android Studio

+ Open Firebase in Android Studio: Tool -> Firebase

+ Connect to Firebase Analytics

+ Add Analytics to the app

+ Add Firebase Authentication SDK to the app

+ Add Realtime Database to the app

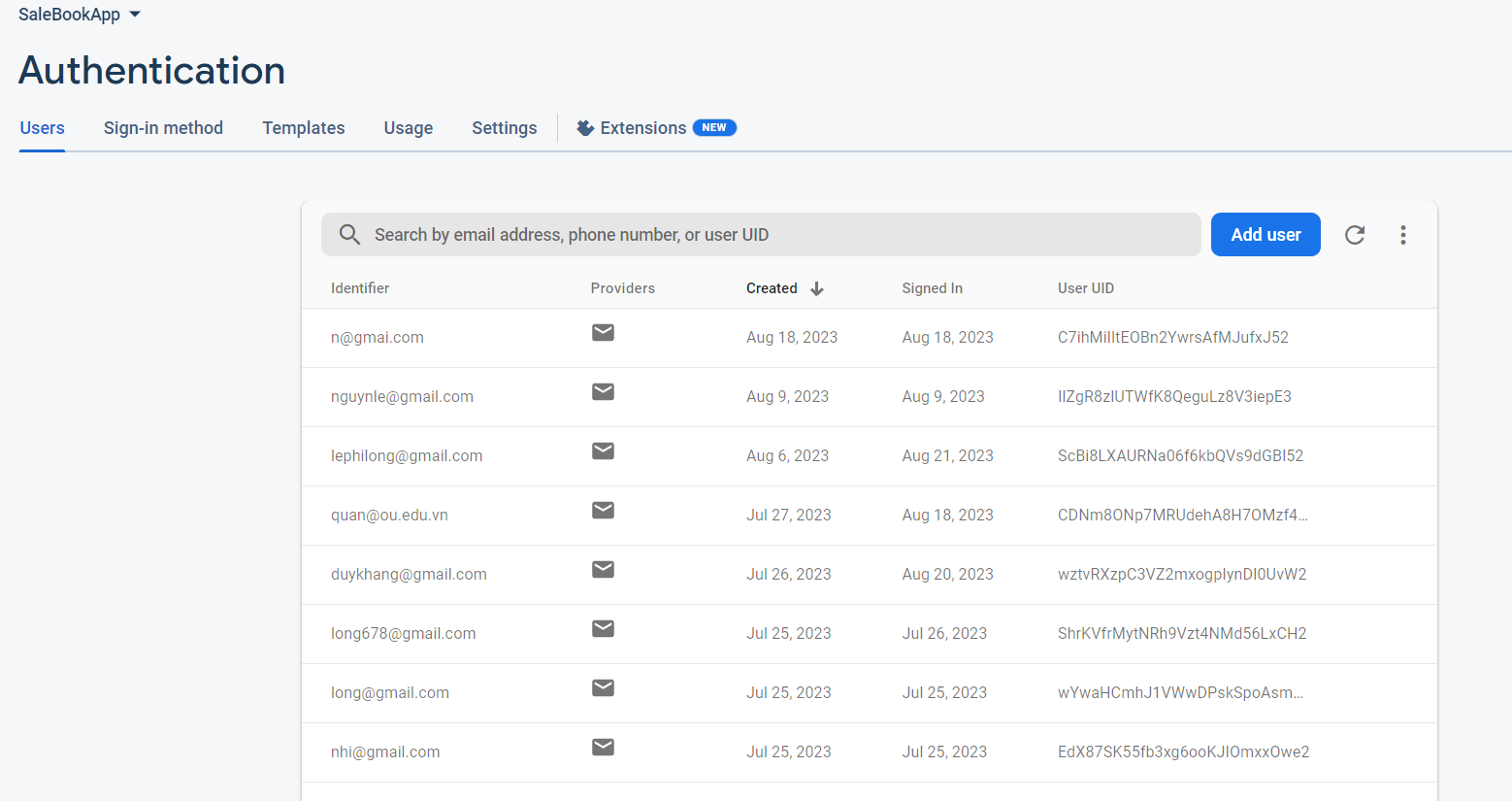
+ Add Cloud Storage to the app

In the build.gradle(Module:app), add these SDKs:

A screen shot of a computer

Description automatically generated

Connecting Firebase on Google:

Authentication, Realtime Database and Storage have been initialized on Google Firebase and connected to the project. Next are the Firebase interfaces on Google.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#### **2.2.3. Using Firebase for Data Processing**

* 1. **Registration:**
* Declaration: private FirebaseAuth firebaseAuthThis variable is used to reference an object of the FirebaseAuth class, a part of Firebase Authentication.
* Inside onCreate(), Firebase Auth needs to be initialized to use the firebaseAuth variable:
* Declaration: private ProgressDialog progressDialog This variable is used to create a progress dialog to inform users about ongoing processes in the app. In this case, it will show a waiting dialog when performing user authentication-related tasks, such as login or registration. This helps users know that the app is performing a task and is not frozen or unresponsive.
* Inside onCreate(), set up the progress dialog:A screen shot of a computer code

  Description automatically generated

The provided code snippet will create a progress dialog with the title "please wait" and prevent users from closing the dialog by touching outside of it. The purpose of this is typically to inform users that the application is performing a certain operation, and they need to wait until the task is completed.

* Write a function to create a new account “ createAccount()”:

A screen shot of a computer program

Description automatically generated

This function contains the logic to create a user account using Firebase Authentication. It ensures that users are informed about the ongoing account creation process by displaying a progress dialog and provides relevant feedback. Upon successful completion, the process will call the 'updateUserInfo()' method to update user information on Firebase. In case of failure, the dialog will be closed using 'progressDialog.dismiss()'. An error message (based on the thrown 'Exception') will be displayed to users through a 'Toast'.

Write the function to update information on Firebase 'updateUserInfo()':A computer screen shot of a program code

Description automatically generated

A computer screen shot of a program

Description automatically generated

This function performs the task of saving newly created user information into the Firebase Realtime Database. It displays a progress dialog with the message 'Saving user info...', creates a 'timestamp' variable to obtain the current time in milliseconds. It retrieves the user's 'uid' using 'firebaseAuth.getUid()', which is how you identify the current user in Firebase. It creates a 'HashMap' object to package the user's information and populates the information using 'HashMap.put()'. It obtains a reference to the "Users" node in the Firebase Realtime Database. It uses 'ref.child(uid).setValue(hashMap)' to set the value of the node under 'uid' with the data in the 'HashMap'. If the update is successful, the 'onSuccess' method is executed, the success progress dialog is closed, and the user is navigated to the profile screen. Conversely, if the update fails, the 'onFailure' method is executed, the progress dialog is closed, and an error message is displayed to the user.

* 1. **Login:**
* Declare 'private FirebaseAuth firebaseAuth' and 'private ProgressDialog progressDialog;' and initialize Firebase Auth, and set up the progress dialog similarly to the code in the registration section.
* Write the user login function 'loginUser()':

A screen shot of a computer code

Description automatically generated

This function executes the process of user login using Firebase Authentication. It ensures that users are aware that the app is performing the login process by displaying a progress dialog and provides appropriate feedback to the user. It utilizes 'firebaseAuth.signInWithEmailAndPassword(userName, passWord)', which was initialized earlier, to perform the login using the registered email and password. If the login is successful, the 'onSuccess' method is executed, and the 'checkUser()' method is called to verify if the user has registered personal information or not. Conversely, if the login fails, the 'onFailure' method is executed, the progress dialog is closed, and an error message is displayed to the user.

* Write the function to check whether the customer's account exists on Firebase or not, 'checkUser()':

A screen shot of a computer program

Description automatically generated

This function performs the verification of whether the entered login information already exists in the Firebase data. It sets the progress dialog message as "checking User...".

It retrieves the current 'FirebaseUser' object through 'firebaseAuth.getCurrentUser()', allowing you to authenticate the logged-in user.

It obtains a reference to the 'Users' node on the Firebase Realtime Database using 'FirebaseDatabase.getInstance().getReference("Users")'.

It uses 'firebaseUser.getUid()' to get the UID of the current user, enabling you to identify the corresponding node for the user's information in the Firebase Realtime Database. It employs 'addListenerForSingleValueEvent' to listen for a one-time data change at the corresponding user node. When there's a data change, the 'onDataChange' method is called. In this case, you retrieve the value of the "userType" field from the 'snapshot' to authenticate the user type. If 'userType' is "user," it means that the logged-in user is a regular user, and they will be directed to the 'MainActivity' and the current screen will be finished. The 'onCancelled' method is executed if an error occurs during the data listening process.

### 2.3. Displaying Product Advertisement Slides

#### **2.3.1. Introduction to the Glide library**

Glide is a powerful and versatile image processing library developed for the Android platform. This library helps manage and load images efficiently in your Android app. Below are some important points about Glide:

1. Efficient image loading: Glide is designed to load images efficiently, making your app run smoothly and saving memory.
2. Caching: Glide offers a robust caching system to store loaded images. This reduces network loads and speeds up image display when users revisit them.
3. Loading from various sources: Glide allows you to load images from multiple sources, including URLs, local resources, and even from raw data.
4. Customization and image manipulation: Glide lets you perform various image manipulations such as resizing, cropping, rotating, and applying image filters before displaying them.
5. Easy integration: Glide integrates seamlessly into your Android project using Gradle or Maven. Once integrated, you can start using it immediately.
6. GIF support: Glide supports displaying GIFs and allows you to load and view these animated images in your app.
7. Multithreading support: Glide operates on background threads, preventing your app from hanging when loading images on the main thread.
8. Support for various image formats: Glide supports multiple image formats such as JPEG, PNG, WebP, GIF, and many others.

Using Glide makes managing and displaying images in your Android app easier and more efficient, allowing you to focus on developing your app's core features without worrying about image processing.



#### **2.3.2. Handling functions**

A screen shot of a computer program

Description automatically generated

1. ActionViewFlipper(): This method is defined within a class or fragment in an Android app. It performs the task of creating and managing a ViewFlipper to display advertisements.
2. List<String> quangCao: This is a list that contains URLs of advertisement images. These URLs will be utilized to load images from the internet and display them within the ViewFlipper.
3. Inside a for loop, a new ImageView is generated for each URL of advertisement images in the quangCao list. Subsequently, the Glide library is employed to load the image from the URL and set it into the ImageView. The ImageView is configured to adjust the aspect ratio and dimensions of the image to fit within the ImageView.
4. bannerAd: This is a ViewFlipper, an Android user interface component employed to transition between child views (in this scenario, advertisement images). The ImageViews created in the preceding step are added to the ViewFlipper using bannerAd.addView(imageView).
5. bannerAd.setFlipInterval(3000): Sets the interval duration between image transitions in the ViewFlipper to 3 seconds (3000 milliseconds).
6. bannerAd.setAutoStart(true): Configures the ViewFlipper to automatically switch between images when the application is launched.
7. Animation slide\_in and Animation slide\_out: These are two animations (effects) defined within the app's resource documents (R.anim.slide\_in\_right and R.anim.slide\_out\_right). These animations are employed to implement transition effects when the ViewFlipper switches between images. setInAnimation and setOutAnimation are used to assign animations to the ViewFlipper.

In conclusion, this code generates and manages a ViewFlipper within an Android application, using Glide to load and exhibit advertisement images from a URL list, and establishing animations for the ViewFlipper to achieve transition effects between the images.

A screen shot of a computer screen

Description automatically generated

* This is an animation set in Android:
* The animation comprises translation from the initial position (-100%) to the final position (0%) horizontally and a fade-in effect from the initial opacity (0.0) to the final opacity (1.0).
* The overall execution time of this animation set is 2 seconds.

A screen shot of a computer program

Description automatically generated

* This is an animation set in Android:
* The animation involves translation from the initial position (0%) to the final position (100%) horizontally and a fade-out effect from the initial opacity (1.0) to the final opacity (0.0).
* The overall execution time of this animation set is 2 seconds.

Both pieces of code can be used to create motion and fading effects when displaying or hiding an element within the user interface of an Android app.

### 2.4. Function to display authors, books, promotions

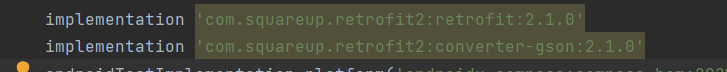
#### **2.4.1. Introducing Retrofit**

Retrofit is a popular open-source library for Android app development used for working with RESTful web services. It's developed by Square, Inc. and provides a convenient and powerful way to perform HTTP requests and interact with APIs from remote servers. Here are some key points about Retrofit:

1. **Simple Usage:** Retrofit simplifies communication with RESTful web services. It uses annotations to define HTTP requests and relies on interfaces to describe how your app will interact with the API.
2. **Automatic JSON Conversion:** Retrofit automatically converts JSON data from requests and responses into Java objects, making it easy to work with data from the server.
3. **Flexible Customization:** You can customize Retrofit to add headers, authentication information, error handling, and many other options for your HTTP requests.
4. **Multithreading Support:** Retrofit integrates well with other libraries like RxJava or Kotlin Coroutines to facilitate multithreaded communication.
5. **Caching:** Retrofit supports caching based on the Cache-Control header in server responses.
6. **Automatic Logging:** It provides built-in integration for logging requests and responses, helping you monitor the communication process with the server.
7. **Support for Various Formats:** Retrofit supports not only JSON but also other data formats such as XML, Protocol Buffers, and Scalars.
8. **Time and Effort Savings:** Retrofit saves time and effort in performing HTTP requests, allowing you to focus on developing the core features of your app.

In summary, Retrofit is a powerful and popular Android development library for communicating with RESTful web services. It simplifies the process of working with APIs and enables the creation of Android apps capable of efficient interaction with remote servers.

#### **2.4.2. Using Retrofit to Make API Calls in Android Studio**



A screen shot of a computer program

Description automatically generated

1. **Gson gson = new GsonBuilder().setDateFormat("dd-MM-yyyy").create();:** This section creates a Gson object, a tool used to convert JSON data into Java objects and vice versa. The configuration setDateFormat("dd-MM-yyyy") is used to format dates during conversion.
2. **SachAPI apiInterface = (SachAPI) new Retrofit.Builder()...create(SachAPI.class);:** This section creates a Retrofit object, a library used to send HTTP requests to a remote server and receive responses. Details:

* .baseUrl("https://book-api.up.railway.app/"): This is the base address of the API server, where API requests will be sent.
* .addConverterFactory(GsonConverterFactory.create(gson)): Retrofit needs to know how to convert JSON data into Java objects and vice versa. We use GsonConverterFactory and provide the previously created Gson object.
* .build(): Constructs a configured Retrofit object.
* .create(SachAPI.class): Creates an object that implements the SachAPI interface. This allows using methods from the interface to send API requests.

1. **@GET("api/book/getAll"):** This is an annotation that marks the getList() method. It indicates an HTTP GET request that will be sent to the path "api/book/getAll" on the server.
2. **Call<List<Sach>> getList();:** This is a method within the SachAPI interface. It returns a Call object, which Retrofit uses to execute HTTP requests. This method is expected to return a list of Sach objects from the remote server's response.

In summary, this code defines an API interface using Retrofit to send HTTP requests to a remote server and receive responses. The responses from the server are then converted into Java objects using Gson for easier handling in Android applications.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

* **getListTacGia():** Sends an API request to fetch a list of authors.
* In the case of a successful response (onResponse), the list is obtained from the response and displayed by attaching it to an adapter (tacGiaAdapter) and setting it in a RecyclerView (rcvTacGia).
* In the case of a failure response (onFailure), displays a "Failure" message using a Toast.

A screen shot of a computer program

Description automatically generated

* **getListPopularBook():** Sends an API request to fetch a list of popular books.
* In the case of a successful response (onResponse), the list is obtained from the response and displayed by attaching it to an adapter (bookAdapter) and setting it in a GridView (gvPopularBook).
* In the case of a failure response (onFailure), displays a "Failure" message using a Toast.

Both methods share the same structure and goal of fetching data from the API and displaying it in the user interface. The crucial part is using Retrofit to execute requests and handle responses reliably.

### 2.5. Product search function

#### **2.5.1. Handle function**

- In SachAdapter

A screen shot of a computer program

Description automatically generated

- In SearchActivity

A computer screen shot of a program code

Description automatically generated

* This method takes a **text** string as input to search within the **listSach** (book list).
* It creates an empty **filteredList** to store the found book items.
* It then iterates through each **item** in the **listSach.**
* For each **item**, it checks whether the **text** string appears in the item's title (obtained using **getTitle()**), by comparing the lowercase version of the string with the lowercase version of the **item**'s title.
* If the **text** appears in the title of the **item**, it adds that item to the **filteredlist**.
* After iterating, if the **filteredlist** is empty, it means no search results were found, so it displays the message "No Data Found." Otherwise, it calls the **filterlist()** method of the adapter to update the displayed data on the GridView (**gvBook**) with the contents of the **filteredlist**.

A computer screen shot of text

Description automatically generated

1. **@Override public boolean onCreateOptionsMenu(Menu menu):** This method is overridden to create a menu for your activity. It takes a Menu object as a parameter and returns true to display the menu or false to hide it.
2. **MenuInflater inflater = getMenuInflater();:** Used to get a MenuInflater object, which helps you inflate menus from resources.
3. **inflater.inflate(R.menu.search\_menu, menu);:** This section inflates the menu from resources with the ID search\_menu and adds it to the activity's menu.
4. **MenuItem searchItem = menu.findItem(R.id.actionSearch);:** Finds a menu item with the ID actionSearch in the inflated menu.
5. **SearchView searchView = (SearchView) searchItem.getActionView();:** Retrieves the search view from the found menu item. This allows interaction with the search bar.
6. **searchView.setOnQueryTextListener (new SearchView.OnQueryTextListener() { ... }):** In this case, it uses OnQueryTextListener to handle events when the user enters or changes text in the search bar.

* **onQueryTextSubmit(String query):** This method is called when the user presses Enter or the search button on the keyboard after entering text. In this case, it does nothing and returns false to indicate that the event has been handled.
* **onQueryTextChange(String newText):** This method is called when the user changes the content in the search bar. In this case, it calls the filter(newText) method to search and filter data based on the newly entered text in the search bar and returns false to indicate that the event has been handled.

In summary, this code allows you to add a search bar to the menu of an Android activity and implements search events to perform search and filtering tasks based on the data within the app.

### 2.6. Function to sort books by genre in the category

#### **2.6.1. Handle function**

A screen shot of a computer program

Description automatically generated

1. **Gson Configuration**:
   * **Gson gson = new GsonBuilder().setDateFormat("dd-MM-yyyy").create();**:This snippet creates a Gson object to convert JSON data into Java objects and vice versa. The important feature here is that the date format is configured as "dd-MM-yyyy".
2. **Tạo Retrofit API Interface**:
   * **DanhMucAPI apiInterface = new Retrofit.Builder()...create(DanhMucAPI.class);**: This snippet creates a Retrofit object to interact with the remote web service. Specifically:
     + **.baseUrl("https://book-api.up.railway.app/")**: The origin address of the API server, where API requests will be sent
     + **.addConverterFactory(GsonConverterFactory.create(gson))**: Retrofit needs to know how to convert JSON data to Java objects and vice versa. Here, we use GsonConverterFactory and provide the Gson object created above.
     + **.build()**:Build a configured Retrofit object
     + **.create(DanhMucAPI.class)**: Create an object that implements (implements) the **DanhMucAPI** interface. This allows the methods in this interface to be used to send API requests
3. **API methods**:
   * The methods defined in this interface represent specific API requests related to different book categories. For example:
     + **@GET("api/book/getByCategory/3222")**:This is a method to get books in the "Thiếu Nhi" category. Code 3222 can represent the category code of "Thiếu Nhi" books on the API server.
     + Similarly, other methods represent different categories of books like "Kinh Doanh", "Khoa Học", "Văn Hóa", "Lịch Sử", and "Giáo Dục".

This code allows making API requests easily through the methods defined in the DanhMucAPI interface, and uses Retrofit to make these requests and get the response from the remote server in the form of a list of books. belonging to different categories.

- In CategoryFragment

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

Description automatically generated

- Above is how to call 2 APIs for 2 different book categories, and the other categories are similar.

1. **context = CategoryFragment.this.getContext();:** This part assigns the context variable with the current Fragment's getContext(). The context variable will be used later to display Toast messages and initialize Intents.
2. **View view = inflater.inflate(R.layout.fragment\_category, container, false);:** This part inflates the user interface from the fragment\_category.xml file into the view variable. This is the interface for this Fragment.
3. **gvBook = view.findViewById(R.id.gv\_sach);:** Finds and stores a reference to the GridView with ID gv\_sach in the user interface. This GridView is used to display the list of books.
4. **Toast.makeText(context, "Giáo dục", Toast.LENGTH\_SHORT).show();:** Displays a short notification using Toast to indicate the "Education" category.
5. **getGiaoDuc();:** Calls the getGiaoDuc() method to retrieve the list of books belonging to the "Education" category and displays them in the GridView.
6. The code snippets from tvSearch = view.findViewById(R.id.tv\_timkiem); to lnrLichSu = view.findViewById(R.id.lnr\_lichsu); are responsible for finding and storing references to various user interface elements. These elements could be buttons or layouts that users can interact with to select different book categories.
7. Each user interface element is associated with an OnClickListener event, so when users click on them, the corresponding methods (getGiaoDuc(), getThieuNhi(), etc.) are called to retrieve and display the respective list of books in the GridView. Additionally, Toast is used to display the selected category's name.
8. Finally, the onCreateView method returns the view variable, representing the user interface of this Fragment. This interface contains the GridView and other interactive elements that allow users to view books in different categories.

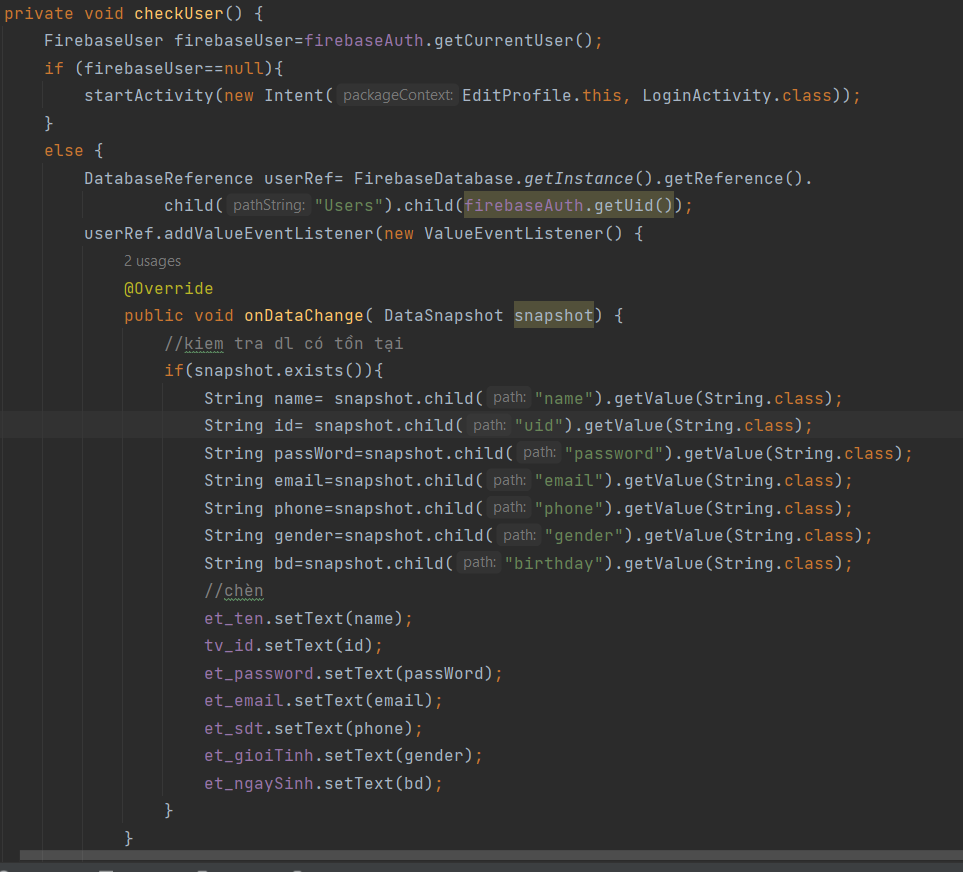
### 2.7. Customizing User Profile Information Feature

#### **2.7.1. Handle function**

- Declaration: private FirebaseAuth firebaseAuth: This variable is used to reference an object of the 'FirebaseAuth' class, a part of Firebase Authentication.

- In onCreate(), Firebase auth must be initialized to use the firebaseAuth variable: firebaseAuth = FirebaseAuth.getInstance();

- Writing the function to check the user account "checkUser":



This function uses Firebase Authentication and Firebase Realtime Database in the Android application to check the user's login status and display their personal information in the EditProfile screen.

FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();: This code retrieves the current user's login information from Firebase Authentication.

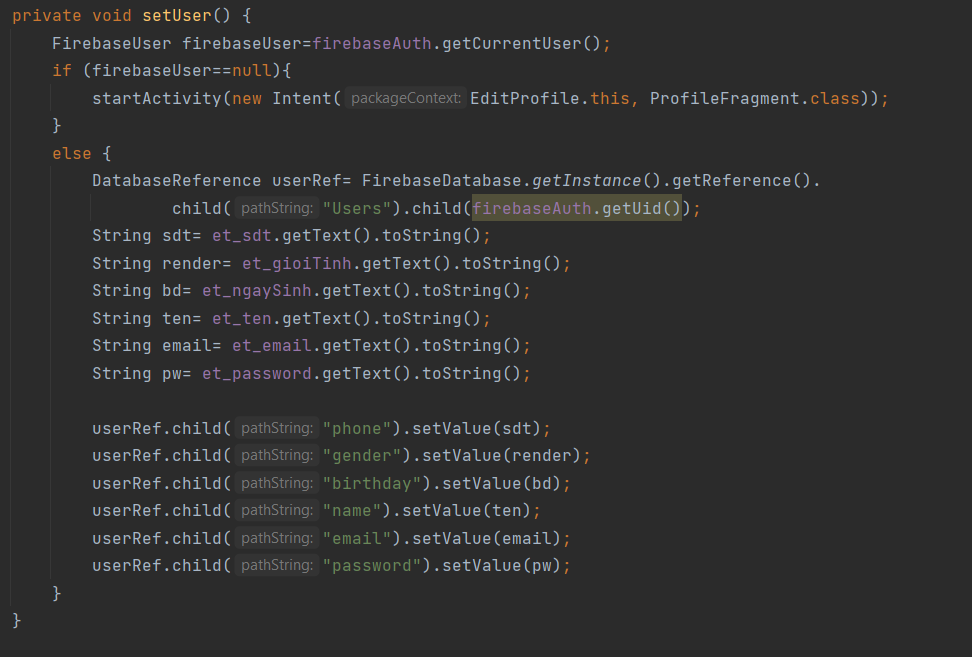
startActivity(new Intent(EditProfile.this, LoginActivity.class));: If the user is not logged in, the screen will redirect to the login screen (LoginActivity).

DatabaseReference\_userRef = FirebaseDatabase.getInstance().getReference().child("Users").child(firebaseAuth.getUid());: This code constructs a DatabaseReference object to access the "Users" node in the Firebase Realtime Database and retrieve user data based on the user's UID.

public void onDataChange(DataSnapshot snapshot) {...}: This method is called when the data changes on userRef. In this method, you check whether user data exists in the database through snapshot.exists().

Extracting information from snapshot: If user data exists, you extract information such as name, ID, password, email, phone number, gender, and date of birth from the snapshot and display them in input fields (EditText) or other interface components.

- Writing the function to update information to Firebase setUser():



FirebaseUser firebaseUser = firebaseAuth.getCurrentUser();: This code retrieves the current user's login information from Firebase Authentication.

Checking the user's login status: If the user is not logged in (firebaseUser is null), the source code will redirect the user to the profile screen (ProfileFragment). However, in this case, it seems a bit unusual to redirect to a Fragment instead of an Activity. Typically, we would redirect to an Activity for login.

Extracting information from input fields: This code retrieves information from input fields (such as phone number, gender, date of birth, name, email, password) that the user has filled in to update their personal information.

DatabaseReference\_userRef = FirebaseDatabase.getInstance().getReference().child("Users").child(firebaseAuth.getUid());: This code constructs a DatabaseReference object to access the "Users" node in the Firebase Realtime Database and retrieve user data based on the user's UID.

Updating information to Firebase Realtime Database: Use child() to specify the child node to update, for example, "phone", "gender", "birthday", etc. Then use setValue() to assign new values to these child nodes, updating the user's personal information.

### 2.8. Function to add books to cart

#### **2.8.1. Handing function**

Using setOnClickListener in SachAdapter to capture the "Buy" button press event.

A screen shot of a computer

Description automatically generated

Initialize an Intent to retrieve data and use the putExtra of the Intent to retrieve data (image, title, price) of any book when clicked.

Call startActivity to send the data.

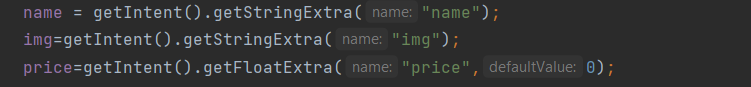
.

A screen shot of a computer code

Description automatically generated

Then create the activity\_add\_cart (Cart) to receive the data.

Use getIntent().getStringExtra() to retrieve string-type data (including title and image) and getIntent().getFloatExtra() to retrieve floating-point data (price).



Create a CartAdapter adapter to create products in the cart with the data just retrieved.

Finally, add the Adapter to the pre-created list interface (GridView) to place the products.



Additionally, on the Cart page, there are buttons like "Pay" and "Delete" on each product using setOnClickListener and using list.remove() and list.removeAll() to remove from the cart.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

## 3. **Instructions for Use:**

### 3.1. Registration and Login

Start the book-selling application on your mobile device.

The app will display the login screen. Choose the "Register" option. Enter necessary personal information such as username, email address, password (first entry), and password (second entry).

Confirm the entered information to ensure accuracy. Complete the registration process by clicking the "Register" button.

The app will display the login screen again. Users input their login information, including the registered email address and password.

Click the "Login" button to authenticate the information and access the personal account.

### 3.2. Edit Profile

In the application, find and select the profile icon interface "Profile." Press "Edit" to begin editing the personal profile.

View personal information such as name, email, password, phone number, and birthdate. If there is inaccurate information or if users want to change other details, they can input or edit the required information.

After editing the information, click the "Save" button to update the new personal information.

### 3.3. Select Products and Add to Cart

Customers browse the book-selling website and search for desired books using the search function or by browsing categories.

When customers find the desired product, they can view images and details to ensure they have selected the correct book.

Each product includes pricing details, images, and titles for customer reference and selection.

Customers can choose the books they want to purchase and then click the "Buy" button to add the product to the cart.

The system will confirm the addition of the product to the cart and display a confirmation message.

### 3.4. Manage Cart

After adding products to the cart, customers can click the cart icon next to the categories icon to view detailed information about their cart.

In the cart page, customers can view all added products, images, prices, and total amounts for each product.

They can also update the cart or remove products from the cart if needed.

### 3.5. Checkout

After managing the cart and confirming the desired purchases, customers can click the "Checkout" button to proceed to checkout.

They can also review the order and total amount before confirming payment.

Upon confirmation, the system will process the order, deduct money from the customer's account, or use the selected payment method.