National College of Ireland

<Software Project>

<Business Information System>

<Academic Year 2020>

<ZIJIAN ZHAO>

<X16112962>

<X16112962@STUDENT.NCIRL.IE>

<Let2Wish(flea market app)>

Technical Report

Contents

Executive Summary

This report mainly introduces the requirements, design and implementation of a second-hand trading platform system, so that readers can have an overall understanding of the system structure and implementation. In the requirements section, the report focuses on the functional requirements of the system, and classifies the requirements according to the division of user roles. Next, according to the requirements and functional requirements, the development framework and background data model of the system are designed, and the data model is briefly introduced. In the implementation stage, we focus on the core function modules of the system, and give a detailed description and explanation of the completed function points. Through the implementation of this system, the author is familiar with the main process and method of development based on firebase and Android, which is of great benefit to the future software development work.

1.0 Introduction

1.1. Background

In recent years, with the great enrichment of people's material life, the market scale of second-hand goods or second-hand goods has grown explosively, At the same time, the purchasing power of students is also gradually improving, and the waste of campus resources is becoming increasingly prominent. In order to more reasonably match and use the resources, the campus second-hand market has also emerged. How to effectively use second-hand goods has become a serious problem that needs to be solved urgently. The traditional second-hand trading platform needs to provide a trading place for trading, and needs to post advertisements as the middle of the transaction. The uncertainty and instability of time and place increase the difficulty of trading, and the flow of information lags behind. However, with the rapid development of the Internet, especially the rapid development of mobile Internet represented by smart phones provides new possibilities for commodity trading. Compared with the traditional commodity trading mode, the Internet information dissemination speed is fast, easy to obtain, abundant resources, flexible and simple operation, which can avoid many disadvantages of traditional commodity trading and provide commodity exchange More convenient and efficient way to promote the rapid development of the second-hand market to maximize the use of resources.

1.2. Aims

The research goal of this project is to build a second-hand commodity trading platform. The system is aimed at the campus student user groups. Students can publish second-hand goods, auction, comment and donate through the platform. The establishment of a campus second-hand trading market can not only facilitate the students in the school, but also create a campus culture atmosphere of diligence, thrift and simple life. It also exerts the students' ability of independent entrepreneurship, practice and innovation, and enhances their economic consciousness and survival consciousness.

1.3. Technology

1.3.1 Android

Due to the huge user base of Android Market, the project decided to develop based on Android system and build with Android studio compiler. Android is an open-source mobile terminal system, and the latest version is Android 11. Android development community is rich in learning materials, and there are many third-party libraries that can be integrated into their own projects, such as butterknife, glide and other excellent open-source libraries.

1.3.2 Jetpack

Jetpack is a suite provided by Google for rapid development of android app. The kit includes a number of common libraries for app development, which can help developers follow best practices, reduce repetitive code writing, and let developers focus on writing app business code. Here are some common jetpack Libraries

activity *	Access composable APIs built on top of Activity.
appcompat *	Allows access to new APIs on older API versions of the platform (many using Material Design).
camera *	Build mobile camera apps.
compose *	Define your UI programmatically with composable functions that describe its shape and data dependencies.
databinding *	Bind UI components in your layouts to data sources in your app using a declarative format.
fragment *	Segment your app into multiple, independent screens that are hosted within an Activity.
hilt *	Extend the functionality of Dagger Hilt to enable dependency injection of certain classes from the androidX libraries.
lifecycle *	Build lifecycle-aware components that can adjust behavior based on the current lifecycle state of an activity or fragment.
Material Design Components *	Modular and customizable Material Design UI components for Android.

navigation *	Build and structure your in-app UI, handle deep links, and navigate between screens.
paging *	Load data in pages, and present it in a RecyclerView.
room *	Create, store, and manage persistent data backed by a SQLite database.
test *	Testing in Android.
work *	Schedule and execute deferrable, constraint-based background tasks.
ads	Get an advertising ID with or without Play Services.
annotation	Expose metadata that helps tools and other developers understand your app's code.
arch.core	Helper for other arch dependencies, including JUnit test rules that can be used with LiveData.

1.3.3 Firebase

The background of this project is based on many services provided by firebase. Firebase is a Google cloud platform designed for developers to provide various tools to solve infrastructure problems. Firebase is a platform for building mobile applications, providing real-time data storage and synchronization, user authentication and other functions. In our project, we mainly use the following functional components:

(1) Authentication

Firebase Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end identity solution, supporting email and password accounts, phone auth, and Google, Twitter, Facebook, and GitHub login, and more. Based on the authentication service, we can complete the functions of APP registration, login and password reset.

(2) Cloud Storage

Cloud Storage is designed to help you quickly and easily store and serve user-generated content, such as photos and videos.

In our app, it is inevitable that many images need to be stored and accessed, such as pictures of goods and images of users. Therefore, we choose this service as the container for image access.

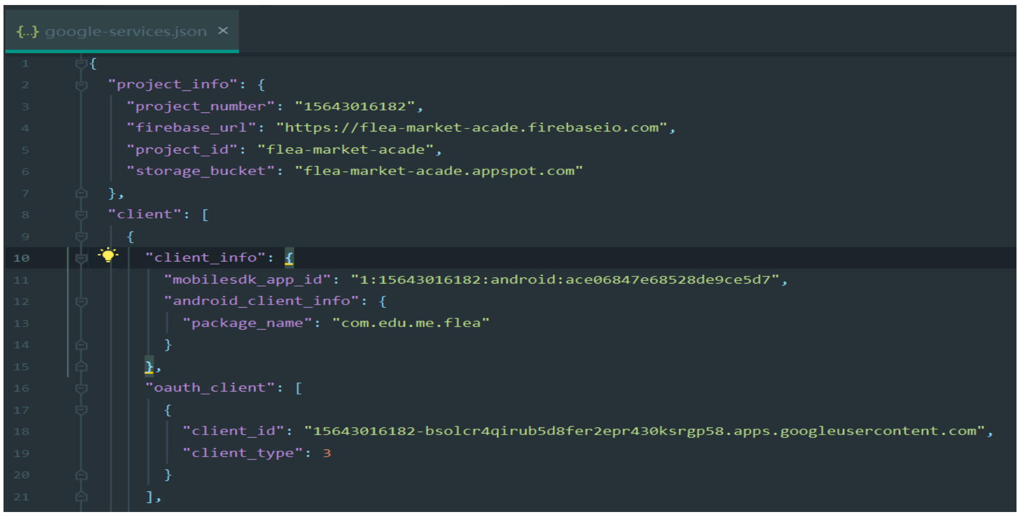
(3)RealTime Database

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in realtime. Based on realtime database, we can build the internal message mechanism of our app, which allows users to chat and communicate freely. The front-end app does not need to know how the specific message is delivered. We only need to publish the message content to the real-time database, and other listeners can get the message content. This greatly facilitates developers to build their own message system.

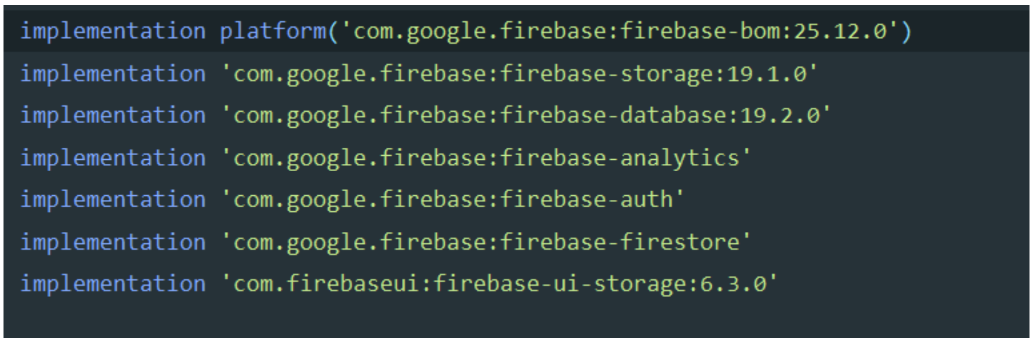
（4）Cloud FireStore

Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps - at global scale. Based on the fact that firestore can access the background data generated by app, firestore can save the front-end data in the form of collection, and query according to specified conditions when necessary. The user information and product information that needs to be saved in our app need the support of cloud firestore.

When using firebase, it is necessary to integrate the Google service files provided by firebase platform in Android studio, and add dependencies on various components of firebase in the gradle configuration file of project app.


google-services.json

```
{
  "project_info": {
    "project_number": "15643016182",
    "firebase_url": "https://flea-market-acade.firebaseio.com",
    "project_id": "flea-market-acade",
    "storage_bucket": "flea-market-acade.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:15643016182:android:ace06847e68528de9ce5d7",
        "android_client_info": {
          "package_name": "com.edu.me.flea"
        }
      },
      "oauth_client": [
        {
          "client_id": "15643016182-bsolcr4qirub5d8fer2epr430ksrgp58.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
```

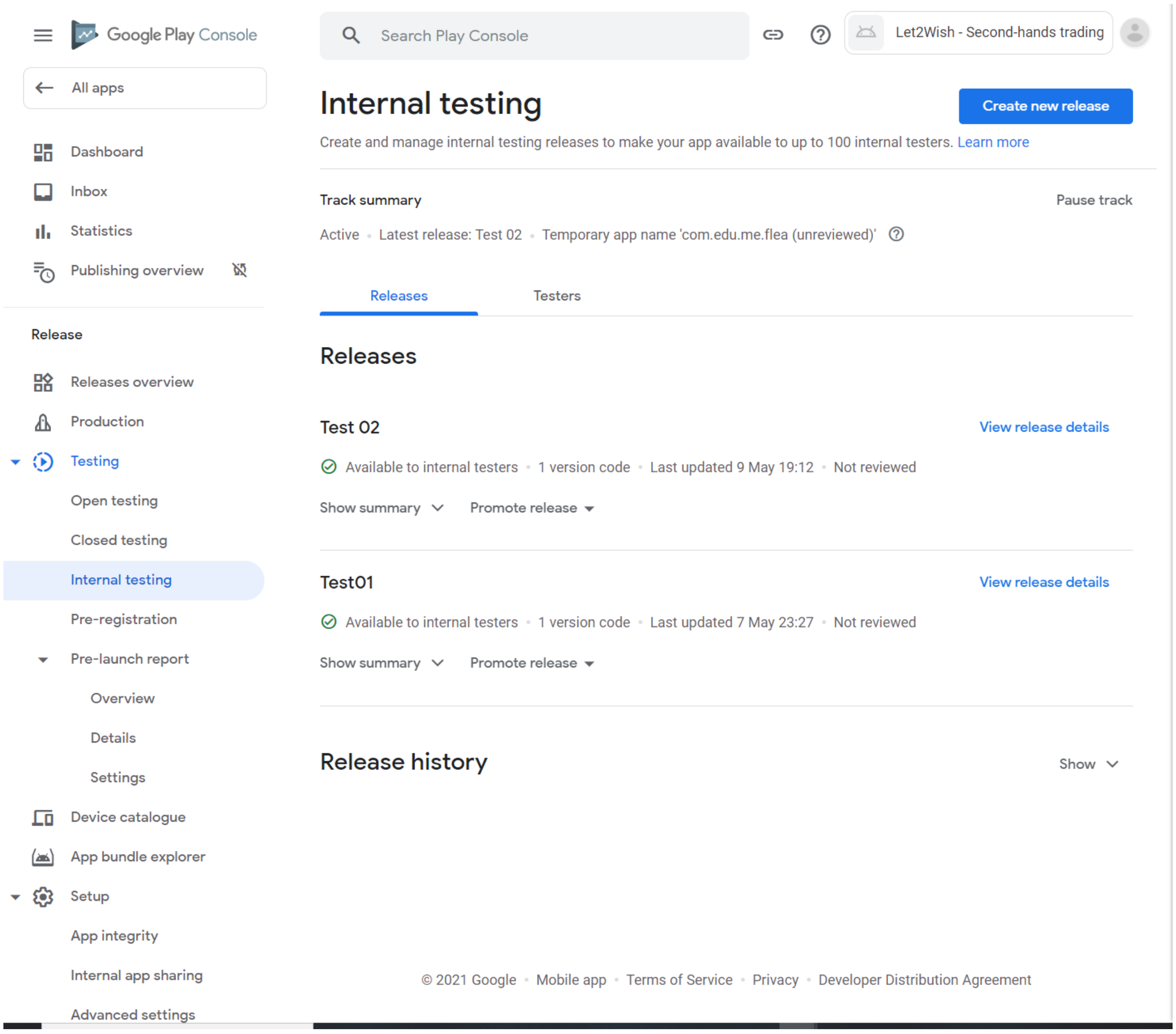


```
implementation platform('com.google.firebase:firebase-bom:25.12.0')
implementation 'com.google.firebase:firebase-storage:19.1.0'
implementation 'com.google.firebase:firebase-database:19.2.0'
implementation 'com.google.firebase:firebase-analytics'
implementation 'com.google.firebase:firebase-auth'
implementation 'com.google.firebase:firebase-firestore'
implementation 'com.firebaseui:firebase-ui-storage:6.3.0'
```

1.3.4 Google Play Console.

The Google play console is the platform for publish the apps. With the console, I can try to make the app update, iteration, and control the several tracks.

The console allowed the developer to test their own app before make the app production. So I created the test Releases.



Internal testing

Create and manage internal testing releases to make your app available to up to 100 internal testers. Learn more

Track summary

Active • Latest release: Test 02 • Temporary app name 'com.edu.me.flea (unreviewed)'

Releases

Test 02
Available to internal testers • 1 version code • Last updated 9 May 19:12 • Not reviewed

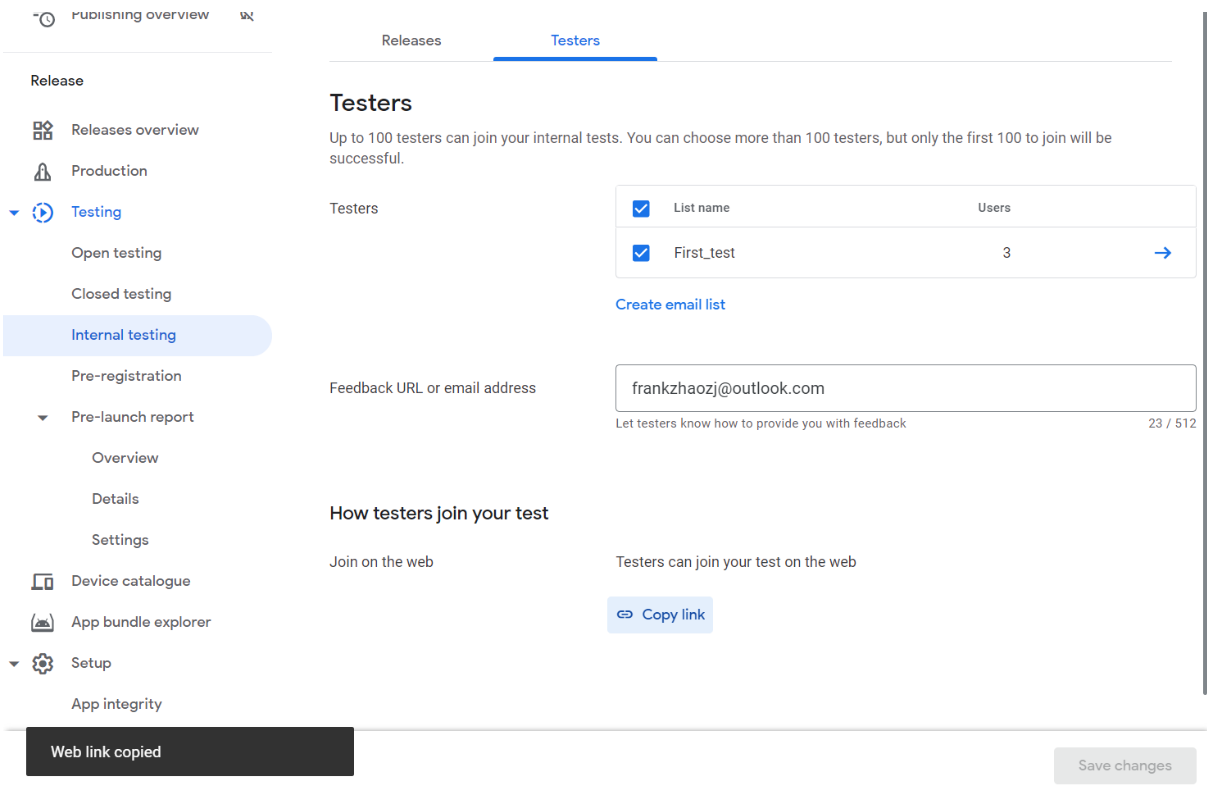
Test01
Available to internal testers • 1 version code • Last updated 7 May 23:27 • Not reviewed

Release history

© 2021 Google • Mobile app • Terms of Service • Privacy • Developer Distribution Agreement

After removing the debugging code, adding the keystore path. The APK pack being signed as release version. I also tried to generate the install pack, which is suggested by Android studio as its download size for user is smaller than APK version. Two types of package are both being uploaded to Google Play console and all passed with the system check.

I also invited a few users to test. The result shows the app can be using totally fine on API 26, API 28, API 30 with the actual phone. Rather than just on virtual devices.



Testers

Up to 100 testers can join your internal tests. You can choose more than 100 testers, but only the first 100 to join will be successful.

Testers

	List name	Users
✓	First_test	3

Feedback URL or email address: frankzhaozj@outlook.com

Let testers know how to provide you with feedback 23 / 512

How testers join your test

Join on the web: Testers can join your test on the web

Web link copied

1.4. Structure

The first chapter mainly discusses the background and goal of the project, and introduces the technology of system application

The second chapter mainly introduces the requirements of the system, including functional requirements, user requirements and data requirements. It also introduces the overall framework design of the system, including the design part of the data model, and then discusses the realization of specific functions.

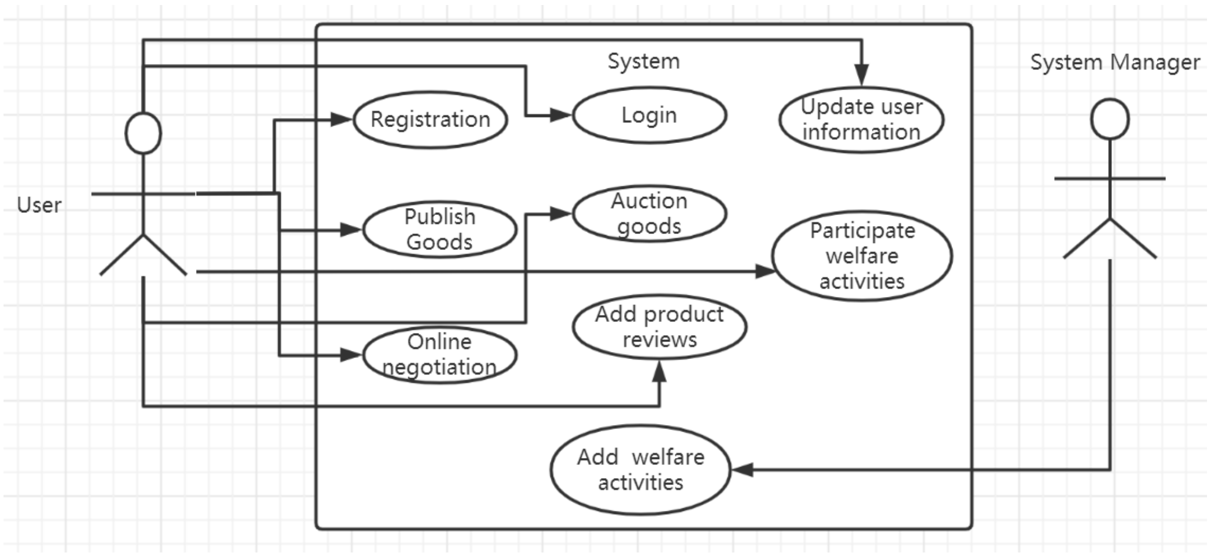
2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram

According to the requirements of the system, the use case diagram can be designed according to the function modules as follows:



2.1.1.2. Requirements

According to the system use case diagram, the main design and completion of the app the following functional requirements.

Requirement 1:	User registration
Description & Priority	App provides users with consistent cloud servers, and users should be able to access their specific data sets among different devices. Therefore, the app provides the registration function. The registration function module can generate a certain user group for the system, and some subsequent functions depend on the registered users.

Requirement 2:	User Login
Description & Priority	In order to restrict different users from using different functions, the system distinguishes users into two categories. One is the logged in user and the other is the non logged in user. The logged in user can use all the same functions, and the non logged in user can only view the system's published product list and public welfare list.

Requirement 3:	Update User Information
Description & Priority	User information is an important information that the system needs to store and maintain. Depending on the user's basic information, the system can provide more personalized services, such as providing different product recommendations according to gender and hobbies. Therefore, the login users should provide the function of perfecting and updating the user information.

Requirement 4:	Publish Goods
Description & Priority	Publishing product information is the core function of this app. Users who log in can publish their products for sale. On the product publishing page, users should be able to edit product information. The edited information should be composed of title, product description and pictures, price and some related tags. After editing, users choose to publish and store the published product information in firebase so that other users can view it by querying the list of published products. For the users who have not logged in, they should be guided to the login page to log in.

Requirement 5:	Auction Goods
Description & Priority	Similar to the function of publishing goods, we provide the function of commodity auction. Users who have logged in can auction their favorite items. In addition to the title, product description and pictures, price and some related tags, there is also an auction deadline. The published auction products can also be viewed by other users in the home page list. For those who are not logged in, they should be guided to the login page to log in.

Requirement 6:	Online Negotiation
Description & Priority	Users can view the list of products published in the system in the home page, and click an item in the list to view the detailed content of the products. For the logged in users, if they can choose to conduct online price negotiation according to the purchase intention, the buyer and the seller can communicate with each other on the negotiation page, and the offline transaction can be conducted after reaching an agreement. The unregistered users need to guide them to the login page for login

Requirement 7:	Add Product Reviews
Description & Priority	Users can view the list of products published in the system on the home page, click an item in the list to view the detailed content of the product, and on the product details page, users can click the Comment button to view the comments of the product. If the user

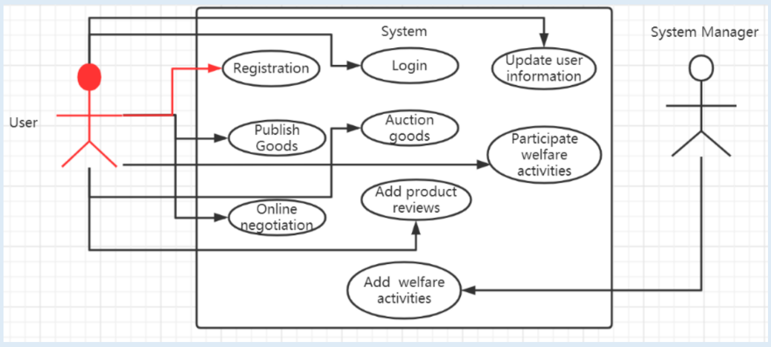
	has logged in, he can also post comments under the product. If he / she is not logged in, if he / she wants to post a comment, he / she should go to the login page to log in. The function of adding comments is to provide users with more reference information about products. For users, more product information can help them decide whether the product is good or bad.

Requirement 8:	Add Welfare Activities
Description & Priority	Our app focuses on public welfare. In the public welfare module, we view the public welfare activities provided by the system. These public welfare activities are added and maintained by the system administrator. If the system administrator adds a public welfare activity to the database table in the background of firebase, other users can view the public welfare activity information in the public welfare list. As an important module of the system, public welfare activities provide users with a more meaningful platform. It is beneficial for the platform itself and society to contribute second-hand goods to others.

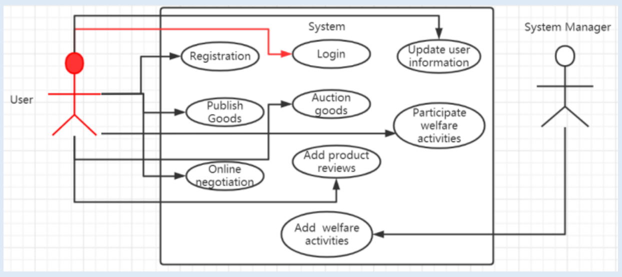
Requirement 9:	Participate welfare Activities
Description & Priority	Users who have logged in can participate in the public welfare activities in the system. The information of public welfare activities should include the title of the activity, the description of the activity, the progress of the activity, the number of participants and the goal of the activity. If the current activity progress has reached 100%, then the public welfare activity has met the requirements, and the user can not participate in it, otherwise the user can participate in the activity The user of the activity can input the amount of contribution (for example, the goal of a public welfare activity is to collect 100 books, and the user's input of 1 means the public welfare book). The system will record the user's contribution amount. In other subsequent functions, the contribution amount can be used as a reference value for whether the user is reliable.

2.1.1.3. Use Case

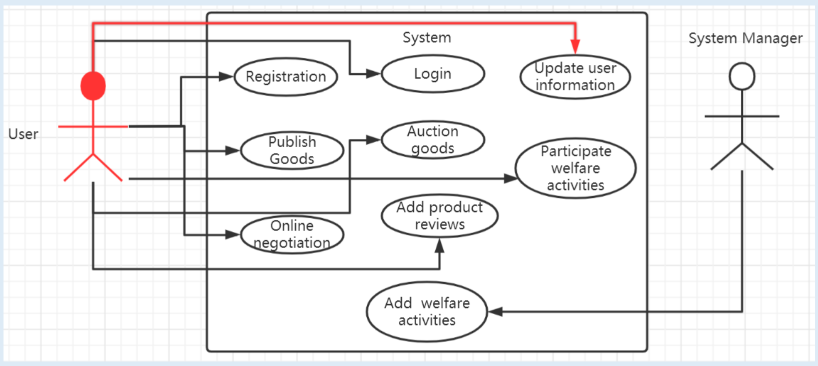
In the previous section, we described the main functional modules that the system needs to implement, and this part continues to describe the main use cases in detail.

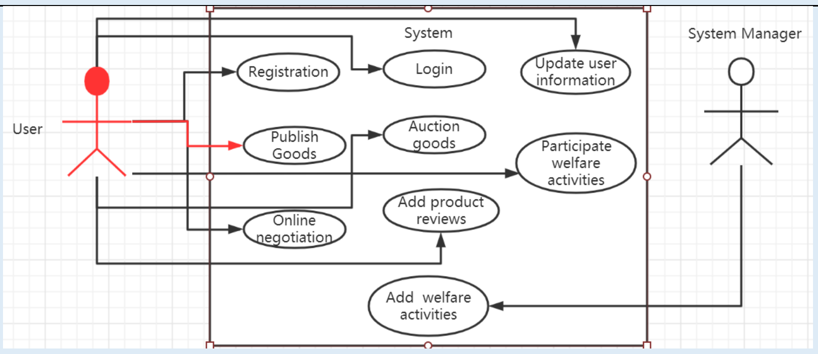
User case Id 1	
Scope	System
Description	User Registration
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User not logged in
Activation	After the user chooses to register
Main -Flow	1. Enter your nickname, email and password, and click Register 2.The user did not enter a nickname, email or password, and click Register （SEE E1） 3.Enter the existing email or nickname and click Register（SEE E2） 4.User input email format is not correct, click Register （SEE E3）
Exception-Flow	E1: input user information exception The system prompts for user information E2： User already register exception Prompt user to register E3： email format error The user is prompted to enter an incorrect mailbox format
Termination	1. Successfully registered, jump to home page

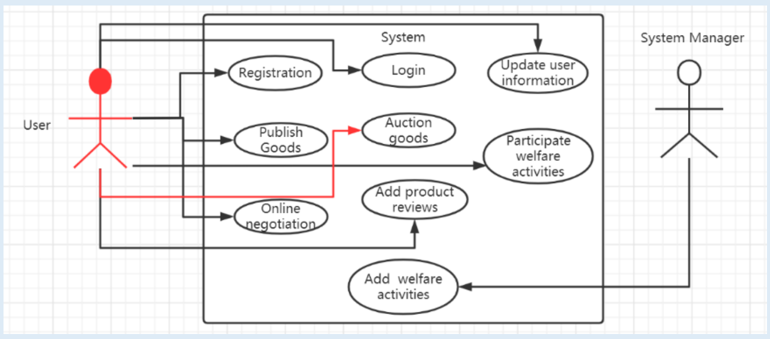
	2. The user closed the registration page
Post Condition	The user has entered the correct registration information

User case Id 2	
Scope	System
Description	User Login
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User not logged in
Activation	After the user clicks the login button
Main -Flow	1. User input email and password, click login 2.User did not enter email or password, click login （SEE E1） 3.The user input the wrong email or password, and click login（SEE E2） 4.User input email format is not correct, click login （SEE E3）
Exception-Flow	E1: input user information exception The system prompts for user information E2： User login information exception Prompt user login account or password error E3： email format error The user is prompted to enter an incorrect mailbox format
Termination	1.Login successfully, jump to home page

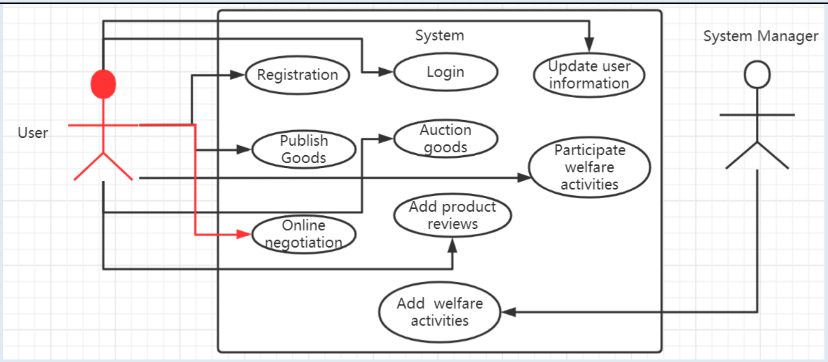
	2.The user closed the login page and returned to the home page
Post Condition	The user has entered the correct login account and password

User case Id 3	
Scope	System
Description	Update User Information
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User logged in
Activation	The user clicks the update button on the user information editing page
Main -Flow	1. The user modified the user information and click Update 2. Users delete the original information, click Update （E1）
Exception-Flow	E1: input user information exception The system prompts the user for new information
Termination	Modify successfully, jump to personal page
Post Condition	User changed user information

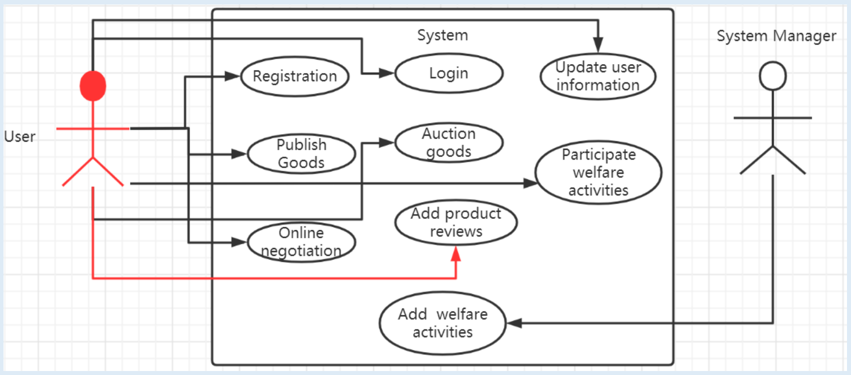
User case Id 4	
Scope	System
Description	Publish Goods
User Case Diagram	 System User System Manager Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User logged in
Activation	The user clicks the Publish button after editing the product page
Main -Flow	1. The user enters the title, description, image selection, price and tag, and then click publish 2. User selects product picture （A1） 3. User did not input product information, click publish （E1） 4. User input correct product information, click publish, system error （E2） 5. Jump to refresh home page after product release
Alternate-Flow	A1: Select multiple pictures Users can select multiple product pictures
Exception-Flow	E1: input user information exception The system prompts the user to enter the corresponding product E2： server exception Prompt system abnormal information
Termination	Post successfully, jump to the front page of the product
Post Condition	The user has entered the correct product information

User case Id 5	
Scope	System
Description	Auction Goods
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User logged in
Activation	The user clicks the Publish button after editing the auction product page
Main -Flow	1. After the user enters the title, description, picture, price, tag and deadline, click publish 2. User selects product picture （A1） 3. User did not input product information, click publish （E1） 4. The user entered an expired deadline （E2） 5. User input correct product information, click publish, system error （E3） 6. Jump to refresh home page after product release
Alternate-Flow	A1: Select multiple pictures Users can select multiple product pictures
Exception-Flow	E1: input user information exception The system prompts the user to enter the corresponding product E2： invalid datetime

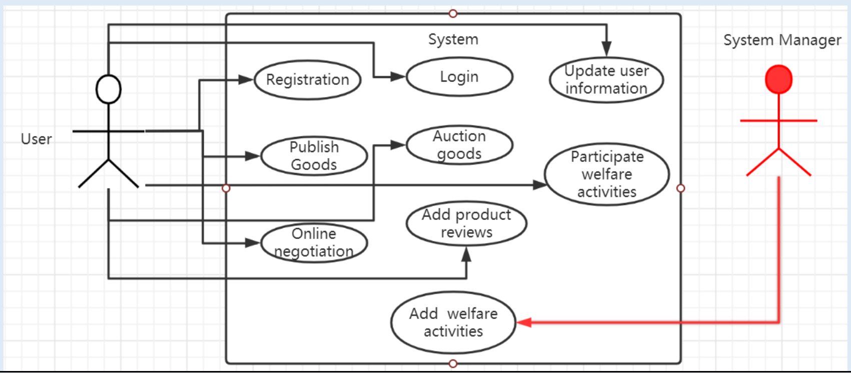
	Prompt the user for the correct deadline E3： server exception Prompt system abnormal information
Termination	Post successfully, jump to the front page of the product
Post Condition	The user has entered the correct information about the auction item

User case Id 6	
Scope	System
Description	Online Negotiation
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	The user has logged in, but the user is not viewing the product details published by himself
Activation	Users view the product details page and click the I want button
Main -Flow	1. The user clicks the I want button 2. Enter the online chat page, enter the chat content, and click send 3. Send the chat content to the firebase server for storage, and refresh the chat interface 4. If the other party replies to the message, the interface will automatically swipe and display the message content 5. If the other party is not in the chat page, a notice will pop up to prompt the user, and the latest chat record will be displayed in

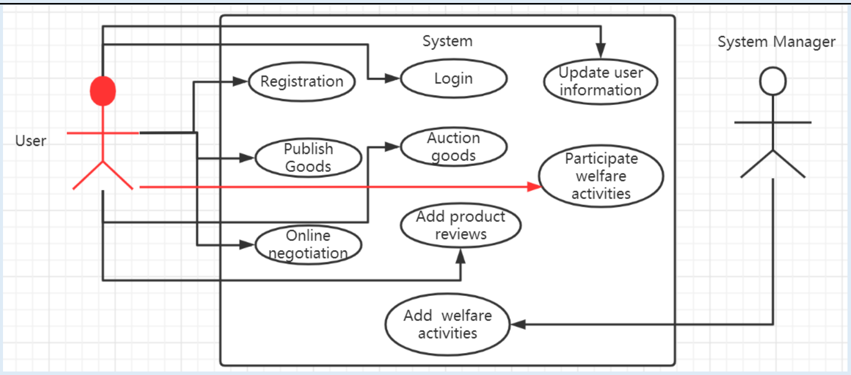
	the message list. The user can click the record to enter the chat page. 5. If no content is entered, click send (E1)
Exception-Flow	E1: input chat information exception Prompt user for chat content
Termination	User launches chat page
Post Condition	Other users have published products

User case Id 7	
Scope	System
Description	Add Product Reviews
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	The user has logged in, but the user is not viewing the product details published by himself
Activation	Users view the product details page and click the Comment button
Main -Flow	1. The user clicks on the home page to jump to the product details page 2. The user clicks the Comment button on the product details page 3. Pop up the comment pop-up box, enter the comment content, and click send

	3. Send the comments to the firebase server for storage, and refresh the comment interface 4. If other users comment on the product, you can see it in the comment pop-up list 5. If no content is entered, click send (E1)
Exception-Flow	E1: input chat information exception Prompt users to enter comments
Termination	User closes comment pop-up
Post Condition	Other users have published products

User case Id 8	
Scope	System
Description	Add Welfare Activities
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	The public collection table is created in firestore
Activation	Click add document in the public welfare table directory
Main -Flow	1. The system administrator clicks add document 2. The system administrator enters each field of the document, including title, description, duetime, destination, progress, current, count, etc. after entering, click Submit. 3. Users can open the app and find the latest added public welfare activities under the public welfare section.
Termination	Cancel add

Post Condition	The corresponding project has been created in firebase platform

User case Id 9	
Scope	System
Description	Participate welfare Activities
User Case Diagram	 System System Manager User Registration Login Update user information Publish Goods Auction goods Participate welfare activities Add product reviews Online negotiation Add welfare activities
Precondition	User logged in
Activation	The user clicks on an activity of the public welfare list under the public welfare section of the home page
Main -Flow	1. Users enter the details page of public welfare activities 2. The user clicks the contribution button to pop up the input pop-up box. The user enters the contribution value in the pop-up box and then clicks OK 3. Refresh the progress of the public welfare activity on the details page to show that you have participated in the public welfare activity 4. When the user enters the wrong contribution value, such as (exceeding the target value of public welfare), the system makes an error (E1)
Exception-Flow	E1: invalid amount exception The system prompts the user to input abnormal contribution value

Termination	Close the pop-up box for input contribution value
Post Condition	The progress of the public welfare activity is not completed

2.1.2. Data Requirements

According to the objectives and functions of the system, we need to put forward the data requirements. Any functional modules and requirements depend on the data to complete. We will discuss the data requirements scenario, data source, data display and so on.

2.1.2.1 Data demand scenario

(1) This system is an online second-hand commodity trading platform. The platform relies on the registered users to trade, so the user data should be one of the core data of the system.

(2) The system can publish commodity information, users can view all kinds of published goods, so the system needs to maintain commodity data, commodity data is the basis of core business.

(3) This system has message mechanism, users can chat and communicate with each other, so the system needs to maintain the message data between users, message data is an important component of the core business.

(4) The system has public welfare and advertising module, public welfare and advertising belong to the means of commercial publicity and promotion, and are the auxiliary data of the system. This part of data is the data source of business expansion.

2.1.2.2. Data source

Data generation

The system is mainly generated by users and system administrators.

Data storage

The data is mainly stored on the firebase cloud platform

2.1.2.3. Data display

The internal data of the system is divided into display part and non-display part. The non-display part is generally used for business logic processing, such as user password. The main parts that can be displayed are as follows:

(1) User information

The system can display user's non sensitive information, such as user's nickname, signature, avatar, etc.

(2) Commodity information

In order to facilitate users to use, it can display the information of products currently stored in the system, display the basic information of commodities in the form of pagination on the home page, and users can view the detailed information of a specific commodity.

(3) Public information

Users can view the public welfare information entered by the system administrator in the public welfare section. The public welfare information is displayed in a list. Users can click the list to view the specific details.

(4) Other statistics

Users can see some statistical information on the home page, such as the products published by users, the public welfare users participate in, etc.

2.1.3. User Requirements

2.1.3.1 Target groups

The main target group is college students, and the secondary target group is users who are willing to look for second-hand goods in the network.

2.1.3.2 Division of user roles

Generally speaking, users can be divided into two categories: users with uncertain shopping intention and users with strong motivation. The former may not register as system users, but just view the existing products in the system at will, while the latter will register as users of the system and will specially search for their favorite products. In order to better understand the user requirements, we can also subdivide the system users at different levels

1. Low-income users

2. Favorite users

3. Online shopping users

4. Environmental protection and low carbon users

5. Foreigner Student

2.1.3.3 User portrait

For different users, we design different user profiles to discuss the user requirements model of the system.

Name	Jack
Role	Low-income users

Description	Jack is an office worker, but his salary is not sufficient and his life is in short supply. He often buys some second-hand goods. He thinks that this can relieve the pressure of life, so he often patronizes some second-hand markets, and he does buy a lot of goods with high-cost performance.
Goal and Tasks:	Be able to look for some good second-hand goods

Name	Lucy
Role	collector
Description	Lucy is a collector. He likes some items with a sense of age, so she likes to hang out in the antique market. However, there are few antique markets around her. So, she often goes to the Internet to look for some antique websites. When she meets her favorite antiques, she will ask the merchants about the price. In order to be safe, she often chooses to trade offline.
Goal and Tasks:	Be able to negotiate prices online and trade offline

Name	Mary
Role	Online shopping enthusiasts
Description	Mary is an online shopping enthusiast. She likes to buy some new things on the e-commerce platform, but after using it for a period of time, she is no longer keen. She gradually accumulates a lot of things that are no longer used in her home. She is very melancholy about why she often likes the new and dislikes the old. She wants to find a place to deal with these things.
Goal and Tasks:	Idle goods can be sold on the platform

Name	Kite
Role	Environmental protection and low carbon users
Description	Kate is a staunch environmentalist. He often castigates the waste behavior around him. He is very disappointed with the environmental problems caused by waste. He feels that many items in daily use do not maximize their value

Goal and Tasks:	It can release some public welfare activities, can appeal to people to protect the environment and maximize the value of idle goods

Name	Luke
Role	Foreigner Student
Description	Luke is a foreigner student from Europe. He would spend 1 year in Ireland and live in a rented apartment. He needs some stuff for daily use. He also has to prepare for leaving. Superfluous new stuff does not apply for his short-term staying time.
Goal and Tasks:	Platform provide the service of buy and sell the staff with a simple flow.

2.1.4. Environmental Requirements

hardware environment

This system depends on Android mobile phone devices, and the device system should be Android version 5.0 or above(recommended). Downward compatibility till API 26.

software environment

Mobile app and firebase cloud service, mobile app needs network support.

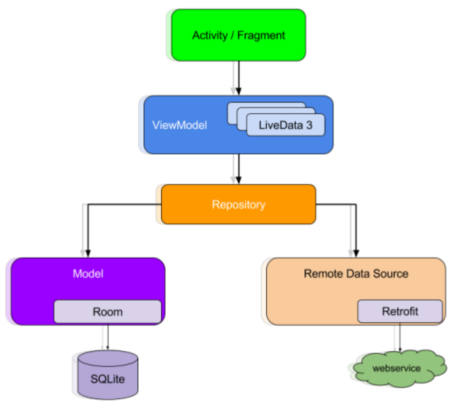
2.1.5. Usability Requirements

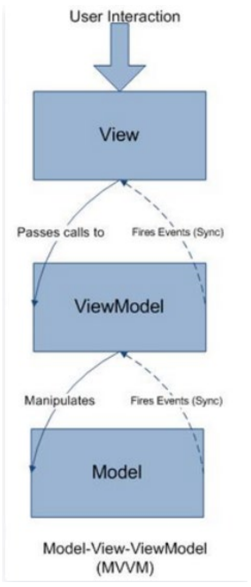
The system may be shut down for some reasons, such as operation failure, hardware failure, software failure, planned downtime, etc. in order to give users a good experience, the system requires that the number of failures should not be higher than 1 / per month, and the failure time should not be more than 5min / per month.

2.2 Design & Architecture

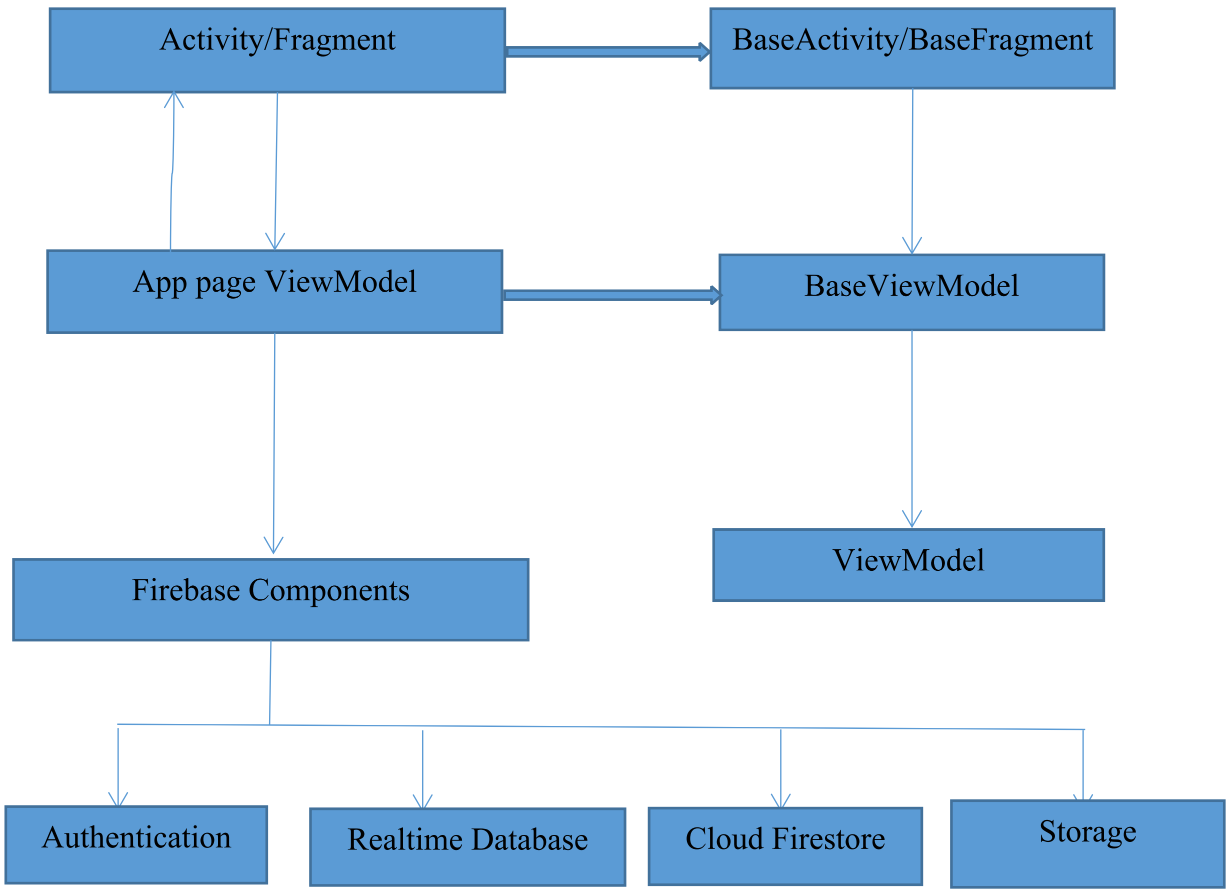
2.2.1 System Architecture

The app is developed based on the MVVM architecture. The typical MVVM architecture is shown in the following figure:





Activity and fragment are views in MVVM. When receiving data from ViewModel, activity / fragment is responsible for refreshing the page according to the data. As the connection layer between activity / fragment and model, ViewModel is responsible for the logical processing of business. Generally, livedata is used in combination with ViewModel layer. When the data is processed, the activity / fragment is informed to refresh the page. In order to further improve the development efficiency, the original MVVM is further encapsulated, and the generic is applied to the base class object baseactivity and fragment of activity / fragment. The generic can provide the ViewModel needed by the page, and further abstract and encapsulate the ViewModel as baseviewmodel. In our system, the model layer is actually made up of many service components provided by firebase. So the development architecture of our app is shown below：



2.2.2 System Model

Although firebase provides us with many consistent service components to facilitate the access of background data, we still need to design a data structure to meet our business needs. The data in firebase is managed by NoSQL database, which is different from the traditional relational database. The data of NoSQL in firebase is the JSON tree. We can access the data of its nodes in the form of JSON. Therefore, according to the functional requirements, we need to design the following JSON collection tree:

(1) User Collection

The user table is used to store the registered user information in the cloud firestore of firebase

Collection	Document	Filed	Type	description
user	user ID	id	String	Id of user
		email	String	Email of user

		nickname	String	Name of user

(2) Goods Collection

Goods collection is used to record the details of goods published or auctioned and stored in firebase's cloud firestore

Collection	Document	Filed	Type	description
goods	goods id	id	String	Id of goods
		title	String	Title of goods
		content	String	Description of goods
		price	String	Price of goods
		tags	String	Tags of goods
		creatorId	String	Id of creator this record
		creatorName	String	Name of creator
		hotDegree	Number	How many users want this goods
		checkCount	Number	Check Count of goods
		images	Array	Images of goods
		dueTime	Number	This filed is for auction goods,it’s the due time of goods.
		createTime	Number	TimeStamp of creating this goods

(3) Snapshot collection

Snapshot collection is used to save the snapshot of goods data and stored in the cloud firestore of firebase

Collection	Document	Filed	Type	description

snapshot	snapshot id	id	String	Id of snapshot
		detailId	String	Id of goods
		title	String	Title of goods
		price	String	Price of goods
		tags	String	Tags of goods
		creatorId	String	Id of creator this record
		creatorName	String	Name of creator
		hotDegree	Number	How many users want this goods
		checkCount	Number	Check Count of goods
		cover	String	A cover image of goods
		dueTime	Number	This filed is for auction goods,it's the due time of goods.
		createTime	Number	TimeStamp of creating this goods

The snapshot collection is the snapshot data of the goods collection. The detailid in the snapshot record corresponds to the ID of the goods record. Therefore, the specific goods record can be found through this field.

(4) Comment Collection

The comment collection is used to save the user's comment data for the product, which is stored in the cloud firestore of firebase

Collection	Document	Collection	Document	Filed	Type	description
comment	goods id	comment	comment id	id	String	Id of comment
				content	String	Content of comment
				creatorId	String	Id of

						creator this record
				creatorName	String	Name of creator
				createTime	Number	TimeStamp of creating this comment

In the comment collection, there are two levels of structure. In the first level, you can see that the comments are saved according to the goods ID. the second level is that all comments under each product are a collection. Generally, it is not suitable to construct the depth of the tree too deep when designing a JSON tree, but this is to improve the efficiency of reading data. We adopt a compromise strategy between them.

(5) Welfare Collection

The welfare collection is used to store the details of public welfare activities in the cloud firestore of firebase

Collection	Document	Filed	Type	description
welfare	welfare id	id	String	Id of welfare activity
		title	String	title of welfare activity
		description	String	description of welfare activity
		cover	String	A cover image of activity
		current	Number	Current value of welfare activity
		destination	Number	Destination value of welfare activity
		progress	Number	progress of welfare activity
		hotDegree	Number	How many users participate this

				welfare activity
		createTime	Number	Timestamp of creating this activity

(6) Messages Collection

The messages collection is used to store the chat content of users and stored in the realtime database of firebase.

Collection	Document	Document	Filed	Type	description
Messages	Id of room	Ids of messages	msg	String	Content of message
			msgType	Number	type of message
			creatorId	String	Id of creator
			creatorName	String	Name of creator
			timeStamp	Number	Timestamp of creating this message

The room ID of messages is composed of the ID of the chat user. For example, if the ID of user A is abc and the ID of user B is 123, then the room ID is abc-123. This is to enable users A and B to quickly find historical chat content through this room ID in messages.

(7) Chatlist Collection

Chatlist stores the chat list of users and records the latest chat content, which is stored in the realtime database of firebase.

Collection	Document	Document	Filed	Type	description
Chatlist	Id of user	Id of room	lastMessage	String	content of last message
			userId	String	Id of peer
			nickName	String	Name of peer
			goodsId	String	Id of goods of chatting

			cover	String	Cover image of goods
			title	String	Title of goods
			timeStamp	Number	Timestamp of creating this message

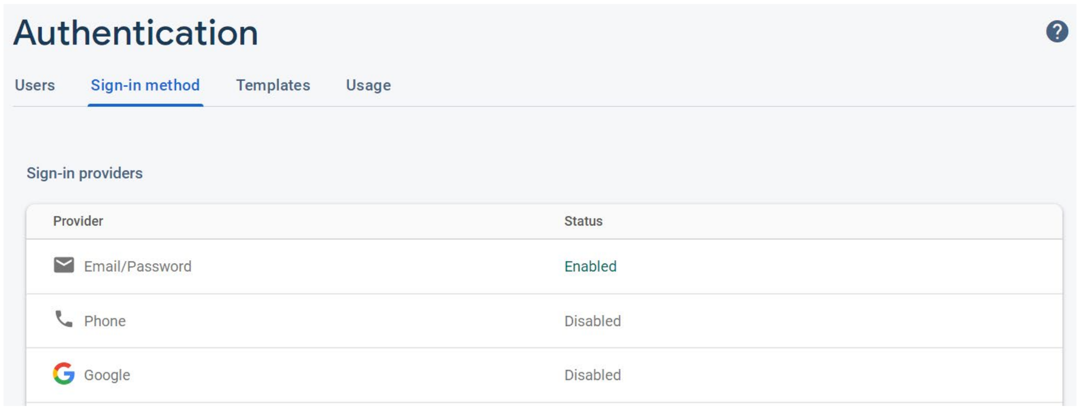
Chatlist collection stores the chat list with the user's ID as the node, so that users can get the chat list according to their own ID. for each chat user, the recent chat content is saved by the room ID. The chat list only records the last message content, because from the chat list to the chat page, we can search all the historical chat records from the message collection according to the room ID. At the same time, some product related information will be saved in each chat record of chatlist. This is to display the product information to the top of the chat interface when entering the chat interface from the chat list. This is to show users which product they are currently discussing.

2.3 Implementation

This chapter will introduce the details of the app implementation, we mainly discuss the implementation process of the core function modules of the system.

2.3.1 user registration and login

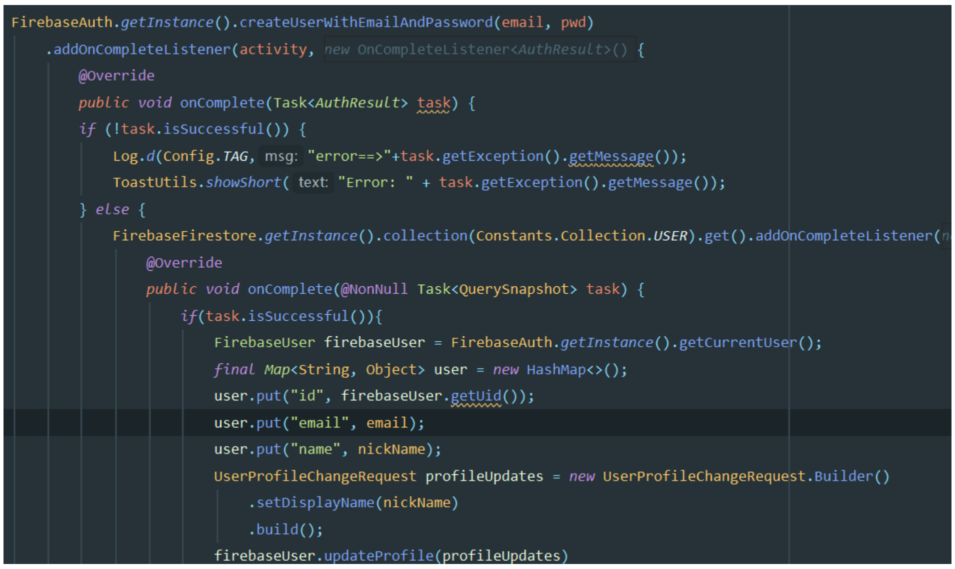
The login and registration function of the system depends on the authentication component of firebase. Before using the firebase component, we need to create a project in the firebase console and enable the email password authentication mode of authentication.



Sign-in providers

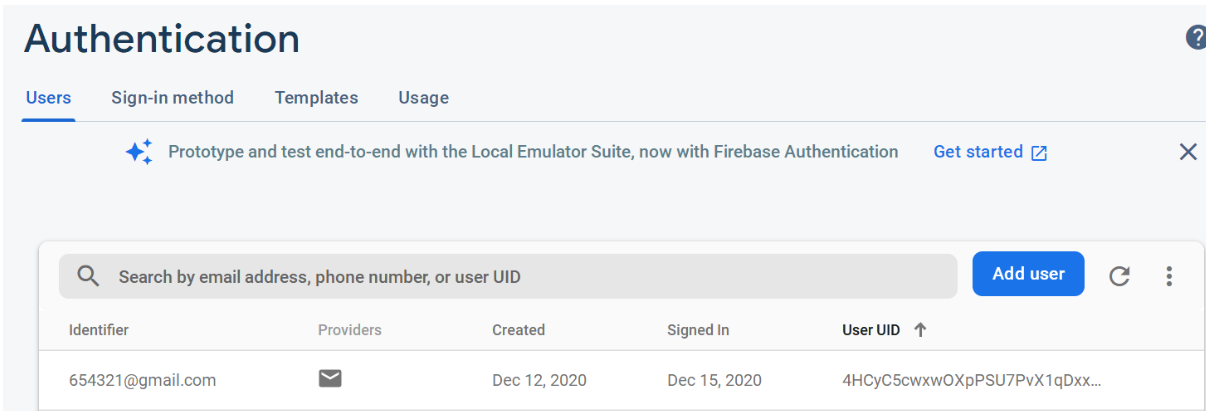
Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled

On the registration page, we need to get the user's nickname, email address and password, and then register the user through the method createUserWithEmailAndPassword of FirebaseAuth. After registration, the registered user information needs to be stored through FirebaseFireStore. At the same time, the display name in FirebaseUser is the nickname used for registration.



```
FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, pwd)
    .addOnCompleteListener(activity, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(Task<AuthResult> task) {
        if (!task.isSuccessful()) {
            Log.d(Config.TAG, msg: "error==>"+task.getException().getMessage());
            ToastUtils.showShort( text: "Error: " + task.getException().getMessage());
        } else {
            FirebaseFirestore.getInstance().collection(Constants.Collection.USER).get().addOnCompleteListener(
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if(task.isSuccessful()){
                        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
                        final Map<String, Object> user = new HashMap<>();
                        user.put("id", firebaseUser.getUid());
                        user.put("email", email);
                        user.put("name", nickName);
                        UserProfileChangeRequest profileUpdates = new UserProfileChangeRequest.Builder()
                            .setDisplayName(nickName)
                            .build();
                        firebaseUser.updateProfile(profileUpdates)
```

After successful registration, we can also see our registered account information in the firebase console

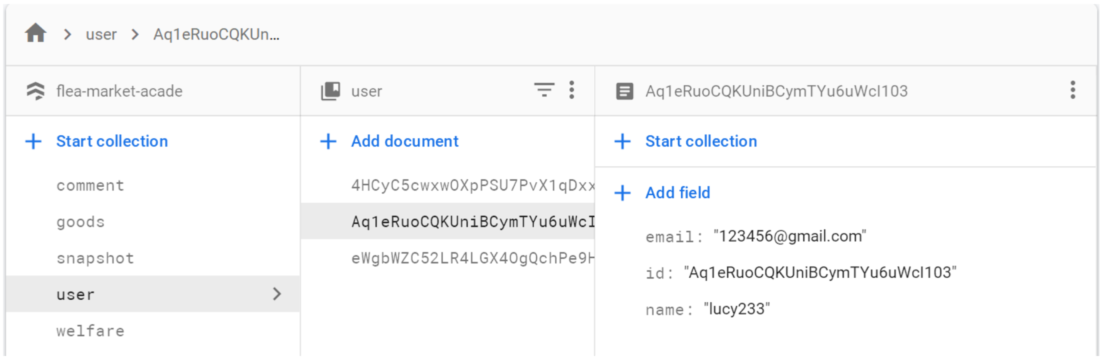


Authentication

Users
Sign-in method
Templates
Usage
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication
Get started
Search by email address, phone number, or user UID
Add user

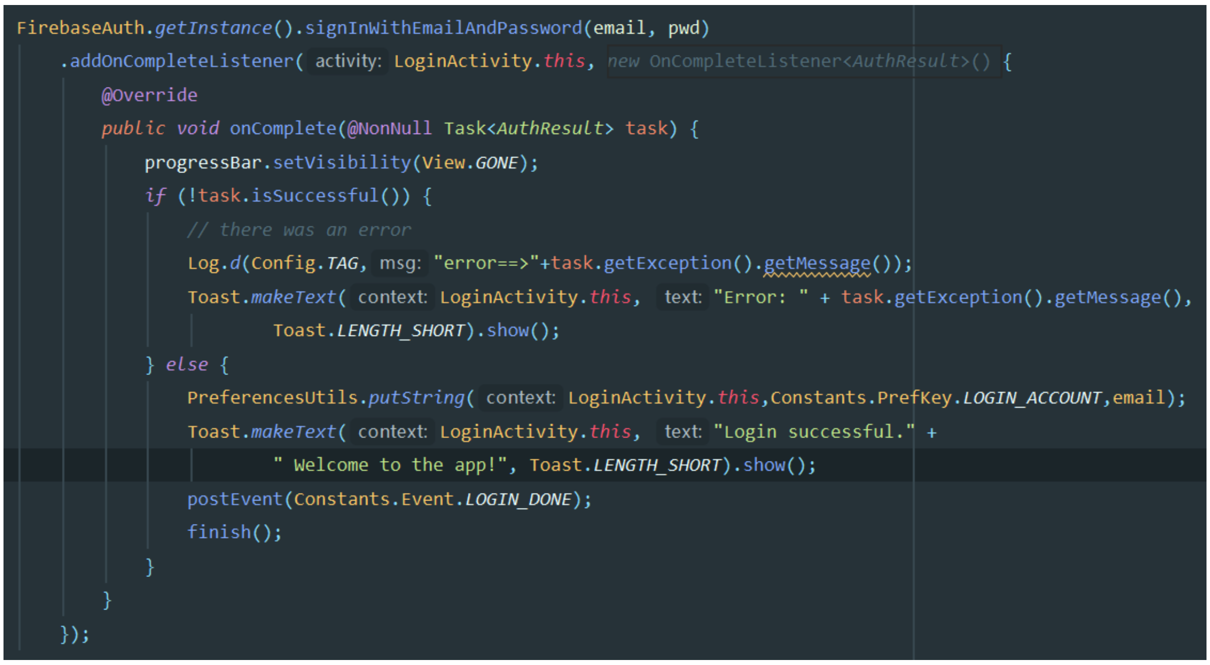
Identifier	Providers	Created	Signed In	User UID ↑
654321@gmail.com	✉	Dec 12, 2020	Dec 15, 2020	4HCyC5cwxwOXpPSU7PvX1qDxx...

You can also see the documentation we created for this user in firestore.



email: "123456@gmail.com"
id: "Aq1eRuoCQKUniBCymTYu6uWcl103"
name: "lucy233"

After registration, we can use the registered account and password to login in the login module. Login is done through the signInWithEmailAndPassword method of FirebaseAuth.

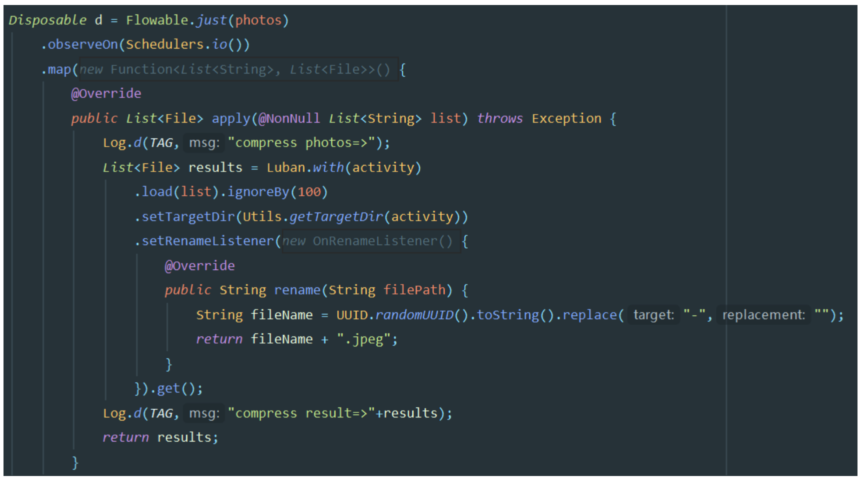


```
FirebaseAuth.getInstance().signInWithEmailAndPassword(email, pwd)
    .addOnCompleteListener( activity: LoginActivity.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            progressBar.setVisibility(View.GONE);
            if (!task.isSuccessful()) {
                // there was an error
                Log.d(Config.TAG, msg: "error==>"+task.getException().getMessage());
                Toast.makeText( context: LoginActivity.this,  text: "Error: " + task.getException().getMessage(),
                        Toast.LENGTH_SHORT).show();
            } else {
                PreferencesUtils.putString( context: LoginActivity.this,Constants.PrefKey.LOGIN_ACCOUNT,email);
                Toast.makeText( context: LoginActivity.this,  text: "Login successful." +
                        " Welcome to the app!", Toast.LENGTH_SHORT).show();
                postEvent(Constants.Event.LOGIN_DONE);
                finish();
            }
        }
    });
```

If the login is successful, the login page will be closed and the event notification of successful login will be sent.

2.3.2 publish goods

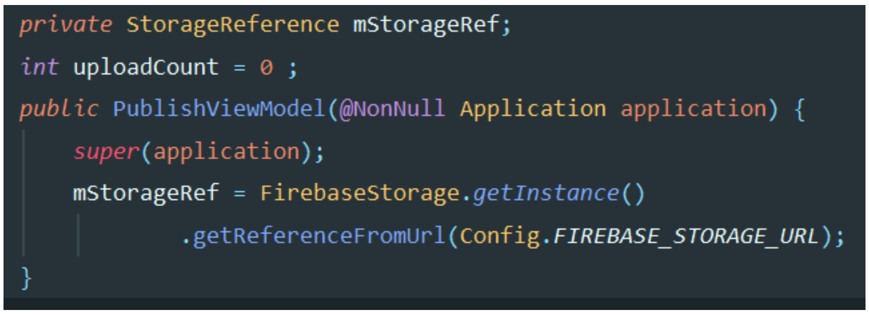
After editing the product information on the product editing page, click publish product to save the product information. This process includes several steps. First, the published product image needs to be saved in firebase storage In order to save bandwidth and ensure the efficiency of upload, the first step is to compress the image. During the compression process, we rename the compressed cache file. Each cache file is specified by a random UUID to ensure its uniqueness. The name of the uploaded file is consistent with the UUID. Later, we can see that the network access address of the image can be obtained through



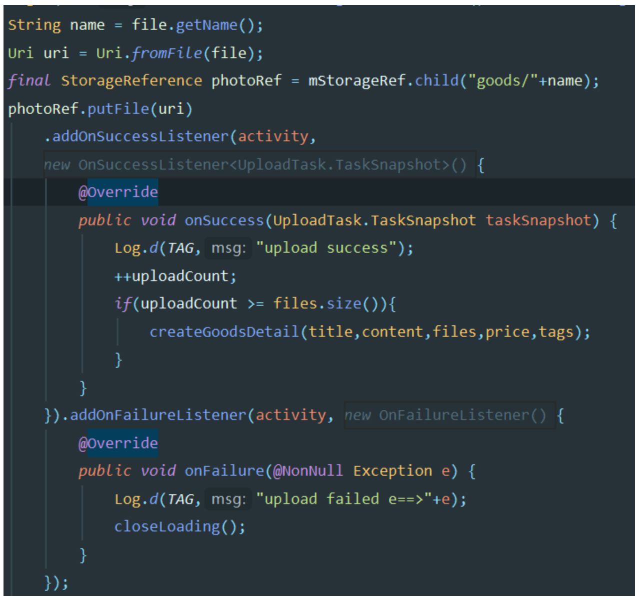
```
Disposable d = Flowable.just(photos)
    .observeOn(Schedulers.io())
    .map(new Function<List<String>, List<File>>() {
        @Override
        public List<File> apply(@NonNull List<String> list) throws Exception {
            Log.d(TAG, msg: "compress photos=>");
            List<File> results = Luban.with(activity)
                .load(list).ignoreBy(100)
                .setTargetDir(Utils.getTargetDir(activity))
                .setRenameListener(new OnRenameListener() {
                    @Override
                    public String rename(String filePath) {
                        String fileName = UUID.randomUUID().toString().replace( target: "-", replacement: "");
                        return fileName + ".jpeg";
                    }
                }).get();
            Log.d(TAG, msg: "compress result=>"+results);
            return results;
        }
```

this UUID.

After compressing the images, the next step is to upload the images to firebase storage. In the storage, we classify the images through the directory, and the images here are uploaded to the goods directory. First of all, we get the StorageReference object through the method getReferenceFromUrl of FirebaseStorage, and then get the corresponding storage reference under the goods node through this object. Then, we can save the compressed cache image in the goods node through the PutFile method.

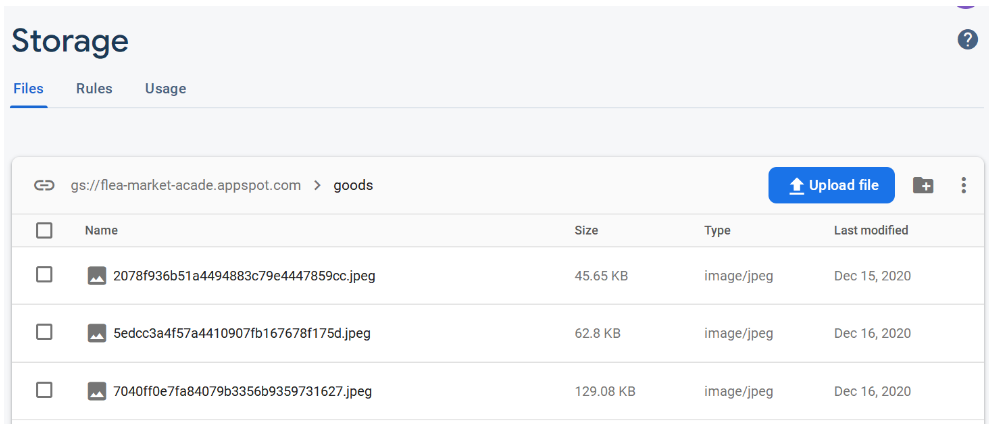


```java
private StorageReference mStorageRef;
int uploadCount = 0 ;
public PublishViewModel(@NonNull Application application) {
    super(application);
    mStorageRef = FirebaseStorage.getInstance()
            .getReferenceFromUrl(Config.FIREBASE_STORAGE_URL);
}
```



```java
String name = file.getName();
Uri uri = Uri.fromFile(file);
final StorageReference photoRef = mStorageRef.child("goods/"+name);
photoRef.putFile(uri)
    .addOnSuccessListener(activity,
    new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            Log.d(TAG, msg: "upload success");
            ++uploadCount;
            if(uploadCount >= files.size()){
                createGoodsDetail(title,content,files,price,tags);
            }
        }
    }).addOnFailureListener(activity, new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Log.d(TAG, msg: "upload failed e==>"+e);
            closeLoading();
        }
    });
```

After uploading the images, we can see the uploaded images in the goods directory of the firebase console storage node



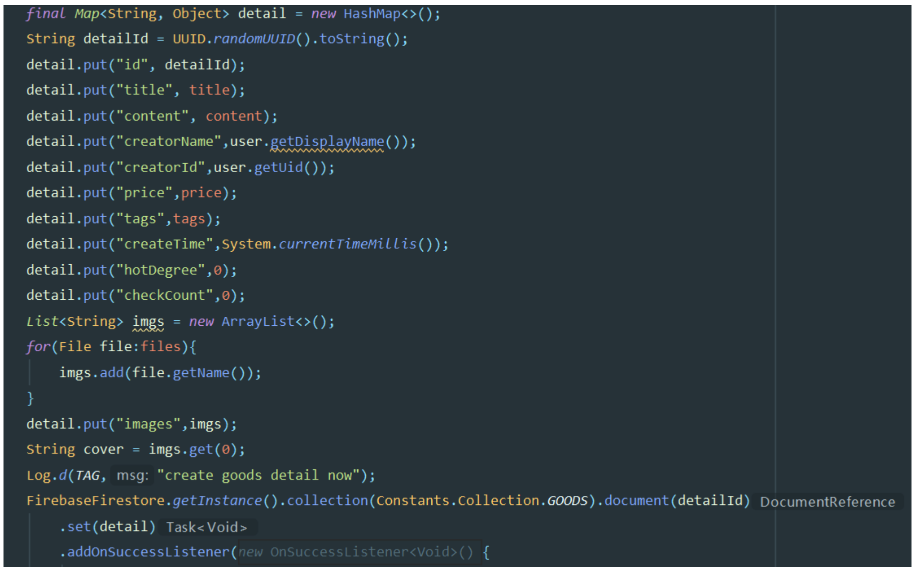
Storage

Files Rules Usage

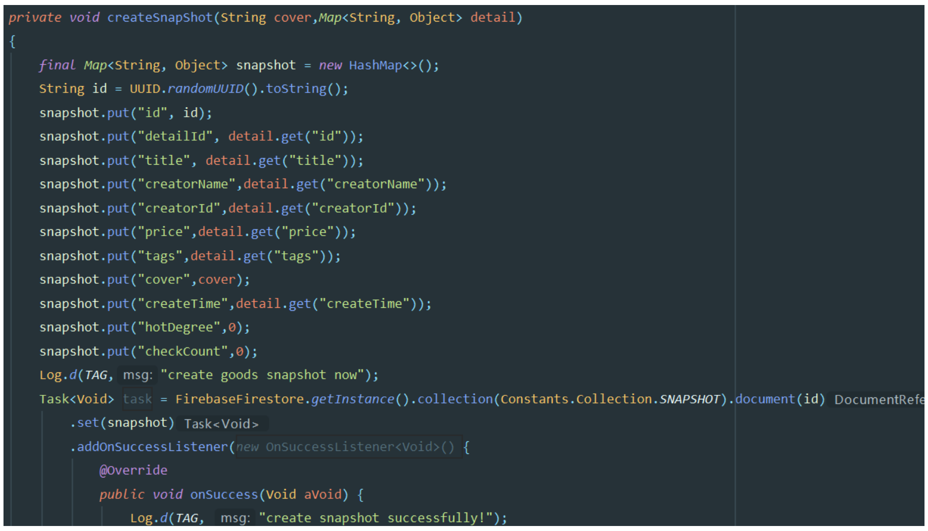
gs://flea-market-acade.appspot.com > goods

Name	Size	Type	Last modified
2078f936b51a4494883c79e4447859cc.jpeg	45.65 KB	image/jpeg	Dec 15, 2020
5edcc3a4f57a4410907fb167678f175d.jpeg	62.8 KB	image/jpeg	Dec 16, 2020
7040ff0e7fa84079b3356b9359731627.jpeg	129.08 KB	image/jpeg	Dec 16, 2020

First of all, store the product name, price information in the list, including the product name, the price information, the product name, the price information, and so on. At the same time, we also save a snapshot record for the product information, which includes the snapshot ID, commodity ID, title, price and heat value. Through the product ID of the snapshot record, we can find the detailed record of the product. The reason for creating a snapshot of published products is to speed up the data transmission when loading products on the home page, and to speed up the display of product list, so that users will have a better use experience.



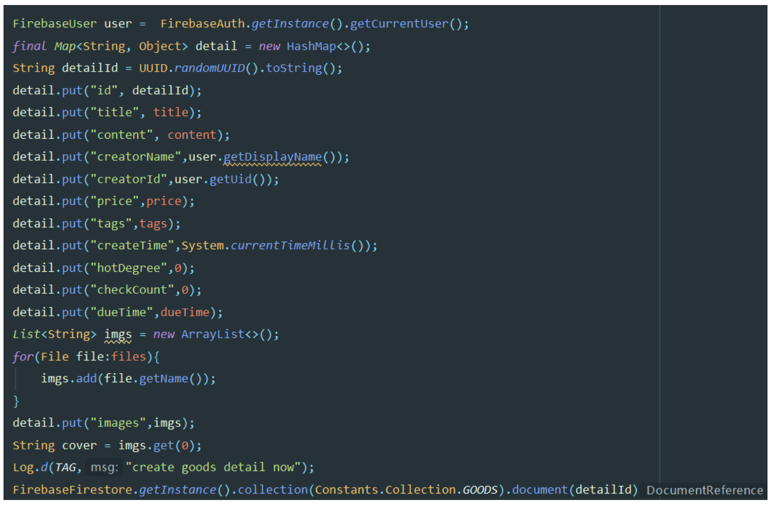
```
final Map<String, Object> detail = new HashMap<>();
String detailId = UUID.randomUUID().toString();
detail.put("id", detailId);
detail.put("title", title);
detail.put("content", content);
detail.put("creatorName",user.getDisplayName());
detail.put("creatorId",user.getUid());
detail.put("price",price);
detail.put("tags",tags);
detail.put("createTime",System.currentTimeMillis());
detail.put("hotDegree",0);
detail.put("checkCount",0);
List<String> imgs = new ArrayList<>();
for(File file:files){
    imgs.add(file.getName());
}
detail.put("images",imgs);
String cover = imgs.get(0);
Log.d(TAG, msg: "create goods detail now");
FirebaseFirestore.getInstance().collection(Constants.Collection.GOODS).document(detailId) DocumentReference
    .set(detail) Task<Void>
    .addOnSuccessListener(new OnSuccessListener<Void>() {
```



```
private void createSnapShot(String cover,Map<String, Object> detail)
{
    final Map<String, Object> snapshot = new HashMap<>();
    String id = UUID.randomUUID().toString();
    snapshot.put("id", id);
    snapshot.put("detailId", detail.get("id"));
    snapshot.put("title", detail.get("title"));
    snapshot.put("creatorName",detail.get("creatorName"));
    snapshot.put("creatorId",detail.get("creatorId"));
    snapshot.put("price",detail.get("price"));
    snapshot.put("tags",detail.get("tags"));
    snapshot.put("cover",cover);
    snapshot.put("createTime",detail.get("createTime"));
    snapshot.put("hotDegree",0);
    snapshot.put("checkCount",0);
    Log.d(TAG, msg: "create goods snapshot now");
    Task<Void> task = FirebaseFirestore.getInstance().collection(Constants.Collection.SNAPSHOT).document(id) DocumentRefe
        .set(snapshot) Task<Void>
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Log.d(TAG, msg: "create snapshot successfully!");
```

2.3.3 Auction Goods

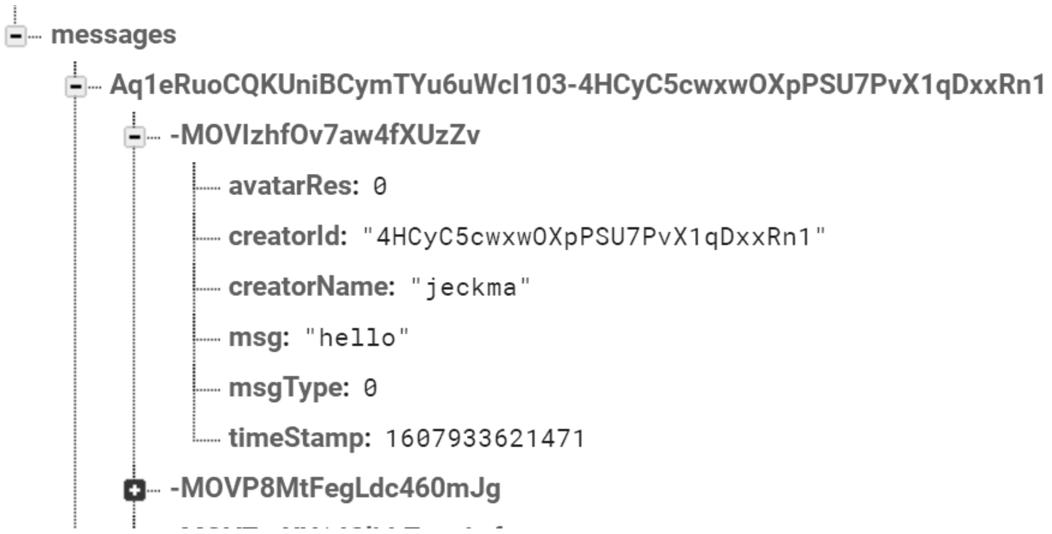
The function of auction goods is similar to that of publishing goods, except that when auctioning goods, we will add an additional field: deadline. If the date exceeds the deadline, the auction will be finished, and the highest price is the current transaction price. Therefore, when users participate in the auction of goods, they can enter their own bid in the product page after online negotiation, and the price will be recorded in the background. After the deadline, the input prices of all users will be compared and the winning price will be calculated.



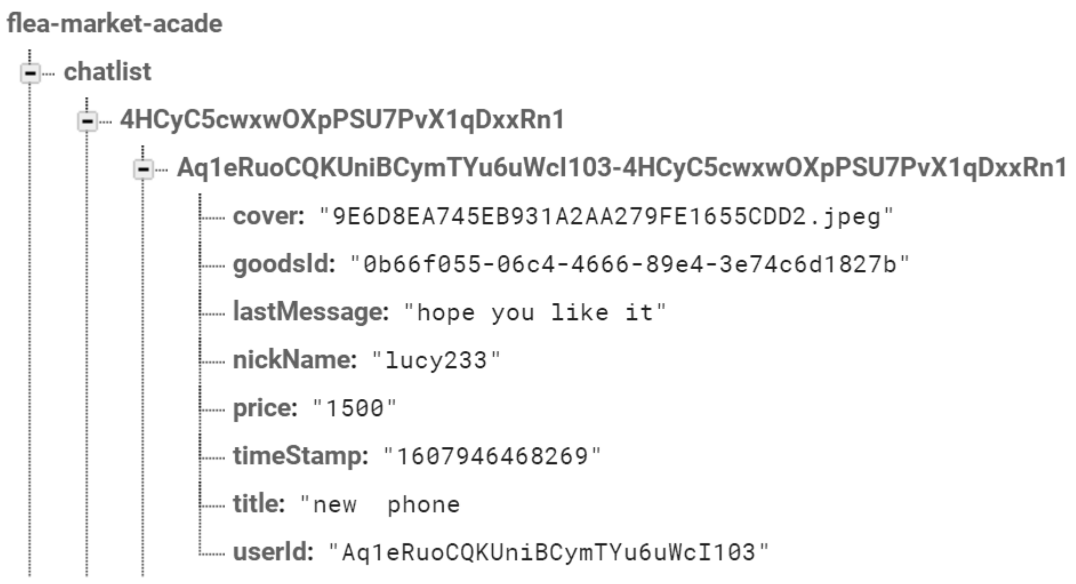
```java
FirebaseUser user =  FirebaseAuth.getInstance().getCurrentUser();
final Map<String, Object> detail = new HashMap<>();
String detailId = UUID.randomUUID().toString();
detail.put("id", detailId);
detail.put("title", title);
detail.put("content", content);
detail.put("creatorName",user.getDisplayName());
detail.put("creatorId",user.getUid());
detail.put("price",price);
detail.put("tags",tags);
detail.put("createTime",System.currentTimeMillis());
detail.put("hotDegree",0);
detail.put("checkCount",0);
detail.put("dueTime",dueTime);
List<String> imgs = new ArrayList<>();
for(File file:files){
    imgs.add(file.getName());
}
detail.put("images",imgs);
String cover = imgs.get(0);
Log.d(TAG, msg: "create goods detail now");
FirebaseFirestore.getInstance().collection(Constants.Collection.GOODS).document(detailId) DocumentReference
```

2.3.4 Online Negotiation

The online negotiation function depends on the message mechanism of the system, which is implemented based on the realtime database of firebase. For any two registered users, there will be two unique ID values (for example, the ID value of user A is abc, The ID value of user B is efg). Through these two ID values, a new unique ID value (abc-efg) is formed. This new ID value can be used as the room ID of the two users chatting. Then the information sent by these two users can be added to the document corresponding to the ID value. In firebase's realtime database, we design two tables: messages and chatlist. Messages table is used to save chat information of any two users, and chatlist saves chat list of users.



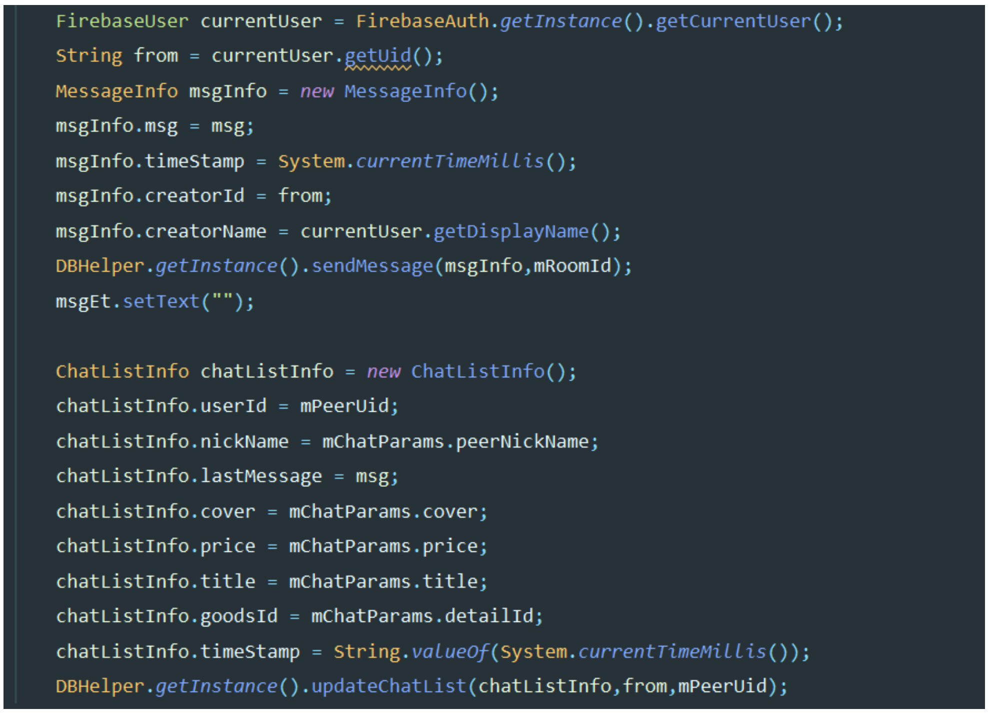
```
messages
    Aq1eRuoCQKUniBCymTYu6uWcl103-4HCyC5cwxwOXpPSU7PvX1qDxxRn1
        -MOVlzhfOv7aw4fXUzZv
            avatarRes: 0
            creatorId: "4HCyC5cwxwOXpPSU7PvX1qDxxRn1"
            creatorName: "jeckma"
            msg: "hello"
            msgType: 0
            timeStamp: 1607933621471
        -MOVP8MtFegLdc460mJg
```



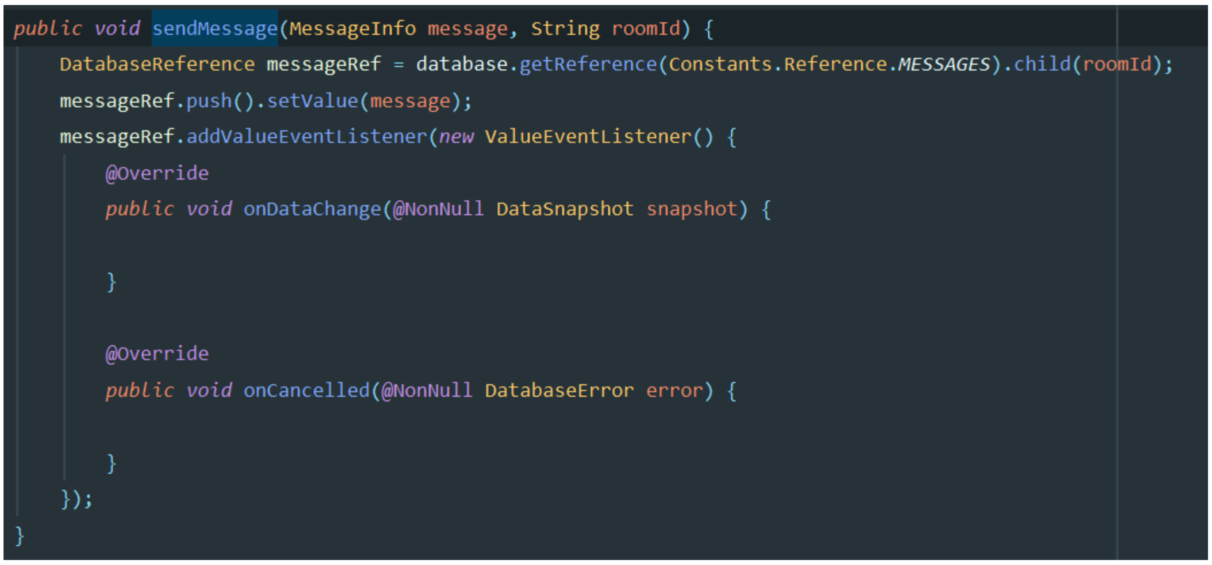
```
flea-market-acade
  chatlist
    4HCyC5cwxwOXpPSU7PvX1qDxxRn1
      Aq1eRuoCQKUniBCymTYu6uWcI103-4HCyC5cwxwOXpPSU7PvX1qDxxRn1
        cover: "9E6D8EA745EB931A2AA279FE1655CDD2.jpeg"
        goodsId: "0b66f055-06c4-4666-89e4-3e74c6d1827b"
        lastMessage: "hope you like it"
        nickName: "lucy233"
        price: "1500"
        timeStamp: "1607946468269"
        title: "new  phone
        userId: "Aq1eRuoCQKUniBCymTYu6uWcI103"
```

In the above figure, I can distinguish that in messages, we are the ID of two users as the access node of messages. The ID value is spliced from the ID of two users. Each message has its own generated ID, which we don't need to care about. The content of the message includes the sender ID, sender name, message content, message type and sending time.

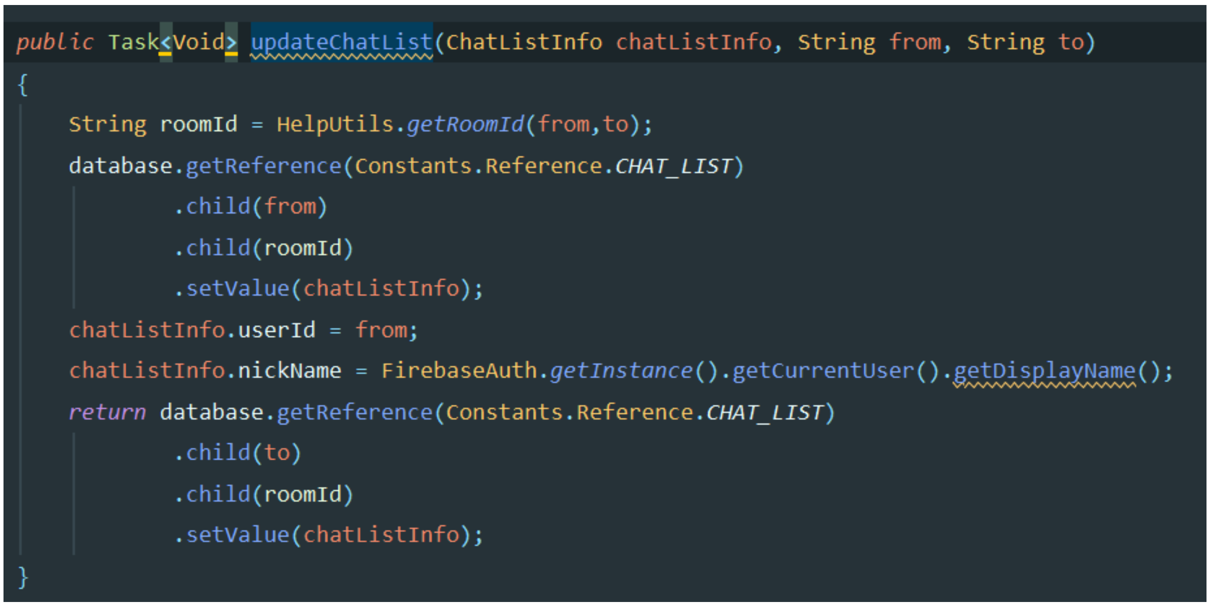
In chatlist, we design to use the user ID as the access node. There may be multiple room ID under the user ID, which represents the multiple users who have chat history with the current user. However, there will be only one up-to-date historical chat information under each room. These records include the cover, title, ID and price of the product, the latest message content, the nickname and user ID of the other party, creation time, etc.



```
    FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
    String from = currentUser.getUid();
    MessageInfo msgInfo = new MessageInfo();
    msgInfo.msg = msg;
    msgInfo.timeStamp = System.currentTimeMillis();
    msgInfo.creatorId = from;
    msgInfo.creatorName = currentUser.getDisplayName();
    DBHelper.getInstance().sendMessage(msgInfo,mRoomId);
    msgEt.setText("");

    ChatListInfo chatListInfo = new ChatListInfo();
    chatListInfo.userId = mPeerUid;
    chatListInfo.nickName = mChatParams.peerNickName;
    chatListInfo.lastMessage = msg;
    chatListInfo.cover = mChatParams.cover;
    chatListInfo.price = mChatParams.price;
    chatListInfo.title = mChatParams.title;
    chatListInfo.goodsId = mChatParams.detailId;
    chatListInfo.timeStamp = String.valueOf(System.currentTimeMillis());
    DBHelper.getInstance().updateChatList(chatListInfo,from,mPeerUid);
```

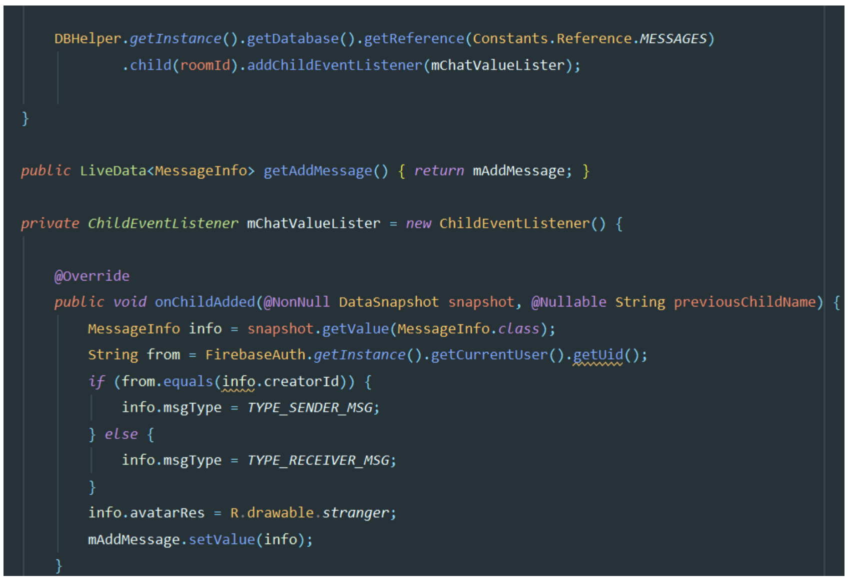


```
public void sendMessage(MessageInfo message, String roomId) {
    DatabaseReference messageRef = database.getReference(Constants.Reference.MESSAGES).child(roomId);
    messageRef.push().setValue(message);
    messageRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {

        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```



```
public Task<Void> updateChatList(ChatListInfo chatListInfo, String from, String to)
{
    String roomId = HelpUtils.getRoomId(from,to);
    database.getReference(Constants.Reference.CHAT_LIST)
            .child(from)
            .child(roomId)
            .setValue(chatListInfo);
    chatListInfo.userId = from;
    chatListInfo.nickName = FirebaseAuth.getInstance().getCurrentUser().getDisplayName();
    return database.getReference(Constants.Reference.CHAT_LIST)
            .child(to)
            .child(roomId)
            .setValue(chatListInfo);
}
```

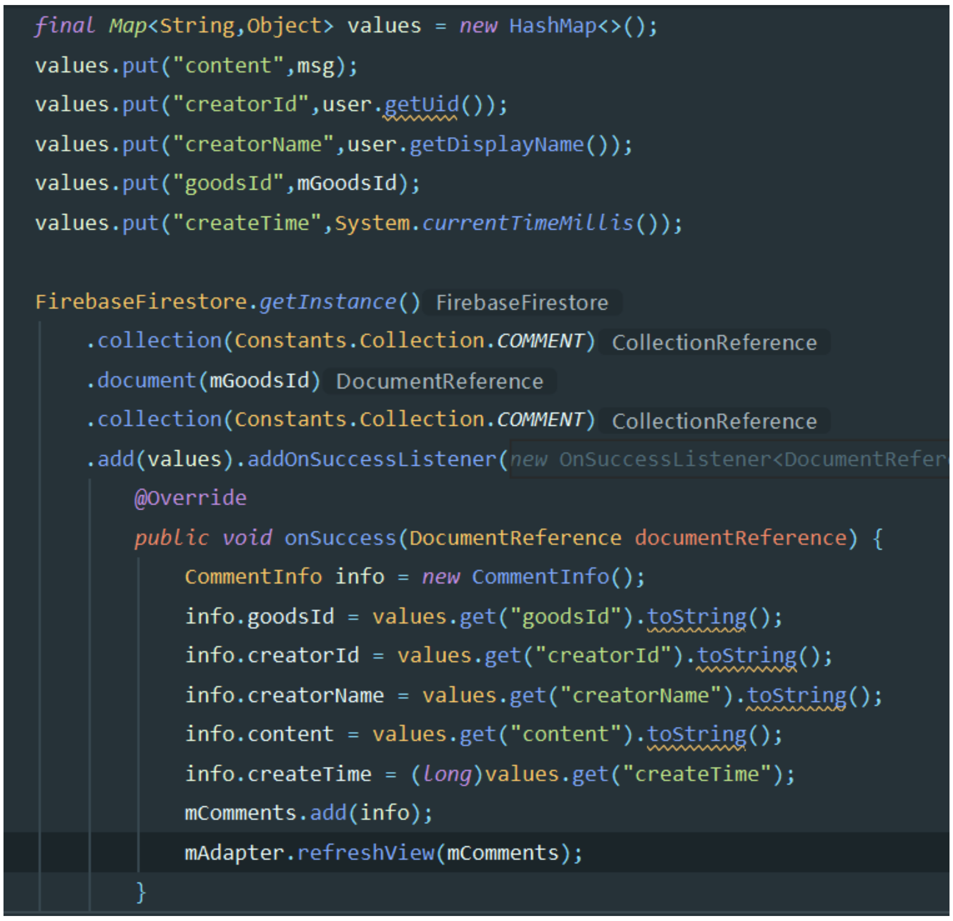
It should be noted that if any user sends a message, for example, user a sends the message hello to user B, we will add the latest message record under the chatlist of the two users (the latest message record will be updated as hello at the same time under the A and B user nodes in the chatlist table). If the user is on the chat page, the message needs to be monitored. The monitoring is based on the front room ID is used as a node. When the message under the node changes (delete, change or add), we can listen to it to refresh the page. The room ID is calculated by our current user ID and the other user ID.

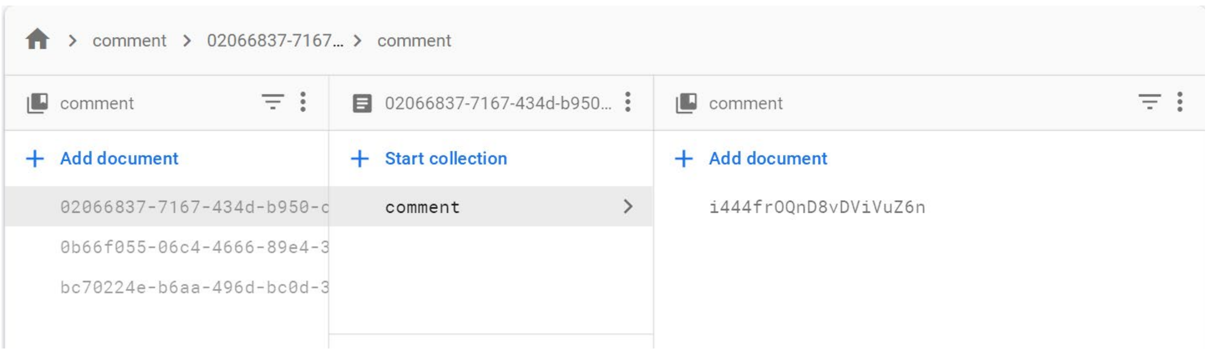


```
    DBHelper.getInstance().getDatabase().getReference(Constants.Reference.MESSAGES)
            .child(roomId).addChildEventListener(mChatValueLister);

}

public LiveData<MessageInfo> getAddMessage() { return mAddMessage; }

private ChildEventListener mChatValueLister = new ChildEventListener() {

    @Override
    public void onChildAdded(@NonNull DataSnapshot snapshot, @Nullable String previousChildName) {
        MessageInfo info = snapshot.getValue(MessageInfo.class);
        String from = FirebaseAuth.getInstance().getCurrentUser().getUid();
        if (from.equals(info.creatorId)) {
            info.msgType = TYPE_SENDER_MSG;
        } else {
            info.msgType = TYPE_RECEIVER_MSG;
        }
        info.avatarRes = R.drawable.stranger;
        mAddMessage.setValue(info);
    }
```

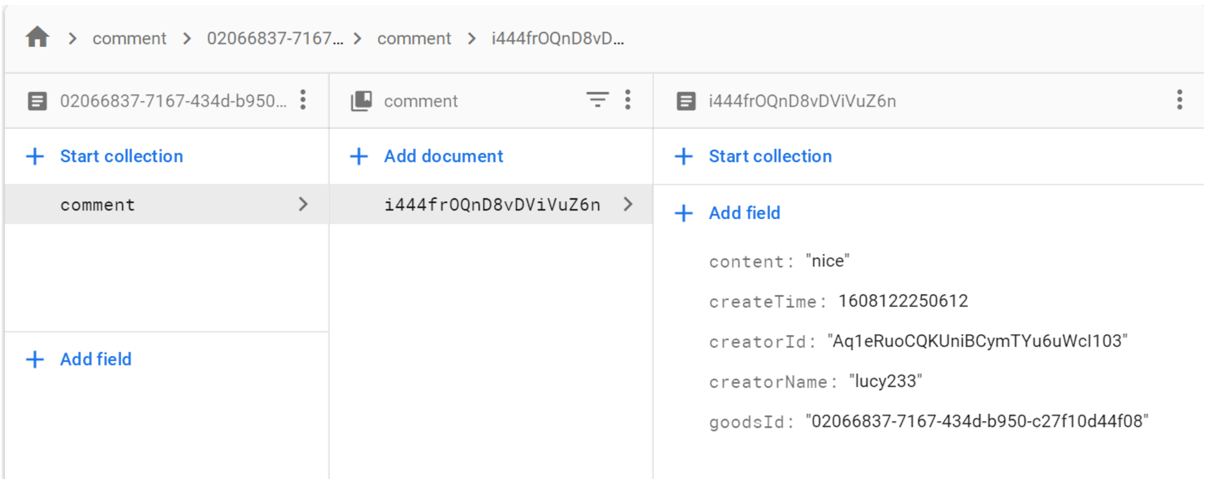
2.3.5 Add Product Reviews

The user who has logged in can comment on the product when viewing the product details. The comment content of the product is saved in the comment table. Each comment record includes the comment content, reviewer's ID, nickname and comment time.


```
final Map<String,Object> values = new HashMap<>();
values.put("content",msg);
values.put("creatorId",user.getUid());
values.put("creatorName",user.getDisplayName());
values.put("goodsId",mGoodsId);
values.put("createTime",System.currentTimeMillis());

FirebaseFirestore.getInstance() FirebaseFirestore
    .collection(Constants.Collection.COMMENT) CollectionReference
    .document(mGoodsId) DocumentReference
    .collection(Constants.Collection.COMMENT) CollectionReference
    .add(values).addOnSuccessListener(new OnSuccessListener<DocumentRefer
        @Override
        public void onSuccess(DocumentReference documentReference) {
            CommentInfo info = new CommentInfo();
            info.goodsId = values.get("goodsId").toString();
            info.creatorId = values.get("creatorId").toString();
            info.creatorName = values.get("creatorName").toString();
            info.content = values.get("content").toString();
            info.createTime = (long)values.get("createTime");
            mComments.add(info);
            mAdapter.refreshView(mComments);
        }
```

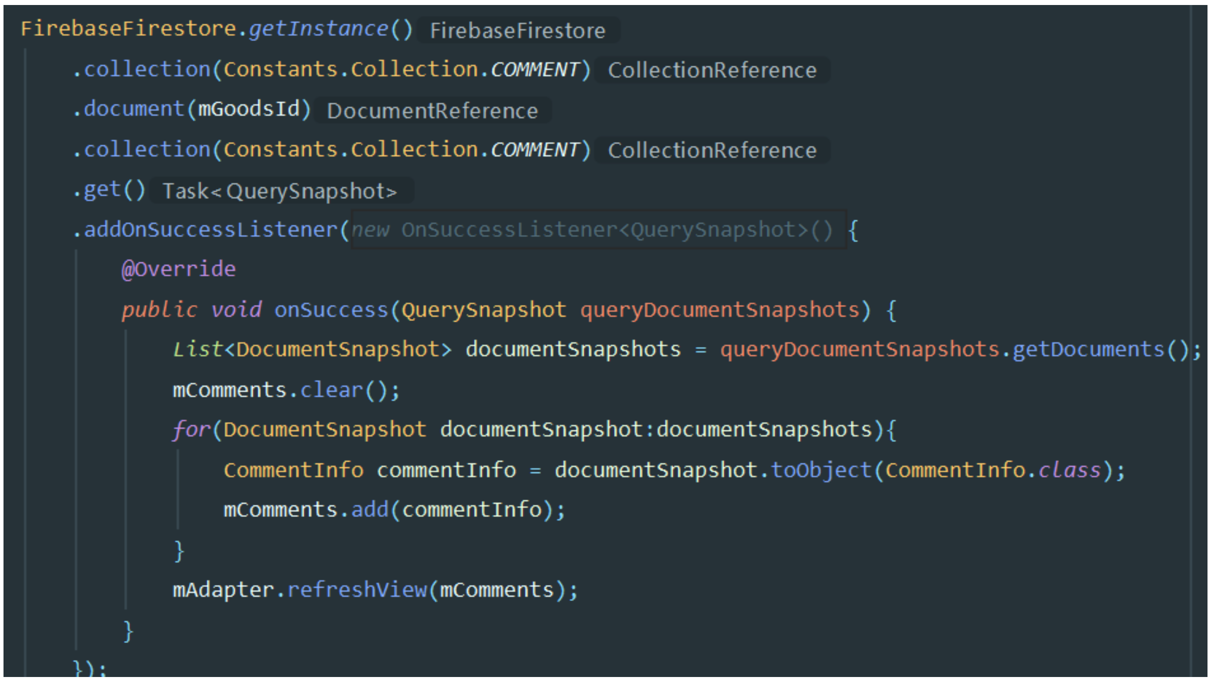

comment > 02066837-7167... > comment

comment	02066837-7167-434d-b950...	comment
+ Add document	+ Start collection	+ Add document
02066837-7167-434d-b950-c	comment >	i444frOQnD8vDViVuZ6n
0b66f055-06c4-4666-89e4-3		
bc70224e-b6aa-496d-bc0d-3		


comment > 02066837-7167... > comment > i444frOQnD8vD...

02066837-7167-434d-b950...	comment	i444frOQnD8vDViVuZ6n
+ Start collection	+ Add document	+ Start collection
comment >	i444frOQnD8vDViVuZ6n >	+ Add field
+ Add field		content: "nice"
		createTime: 1608122250612
		creatorId: "Aq1eRuoCQKUniBCymTYu6uWcl103"
		creatorName: "lucy233"
		goodsId: "02066837-7167-434d-b950-c27f10d44f08"

Different from the structure of the message table, the comment table is accessed with the commodity ID as the node. This is to quickly query the comment content of the current product. A comment node is created under the commodity ID node to save all the comment records of the product.

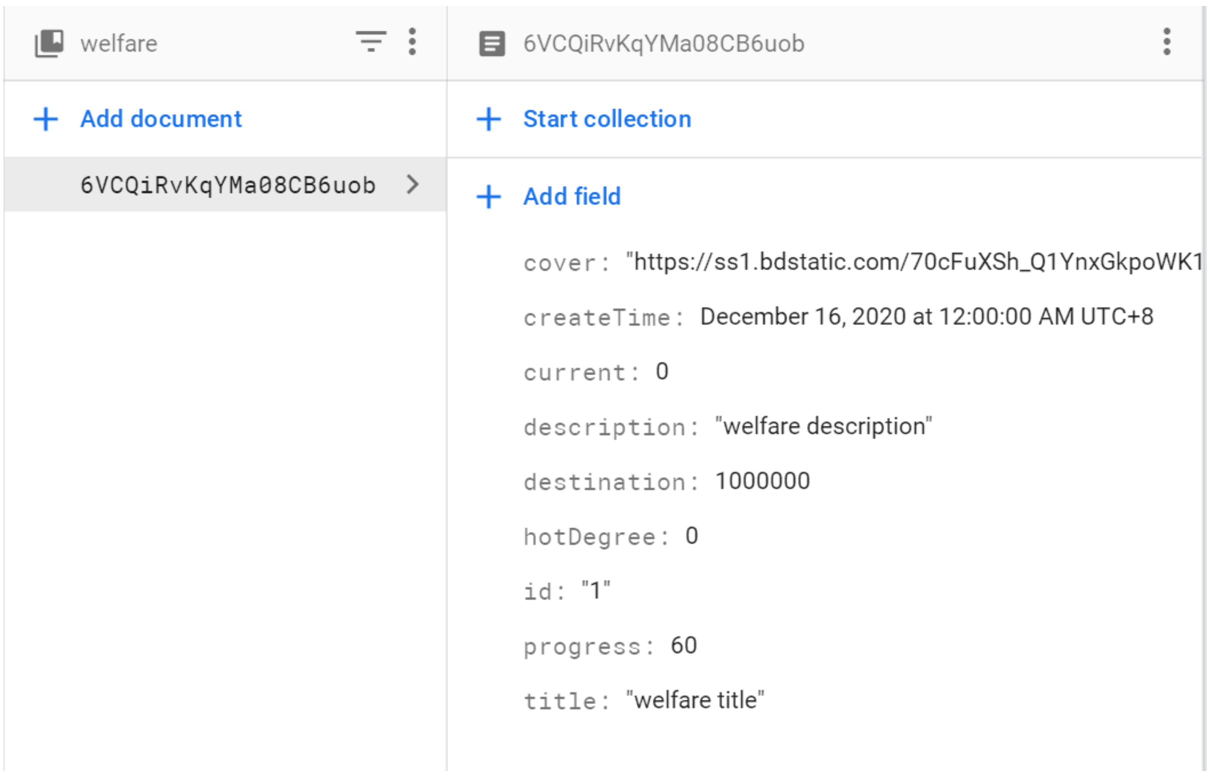


```
FirebaseFirestore.getInstance() FirebaseFirestore
    .collection(Constants.Collection.COMMENT) CollectionReference
    .document(mGoodsId) DocumentReference
    .collection(Constants.Collection.COMMENT) CollectionReference
    .get() Task<QuerySnapshot>
    .addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
        @Override
        public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
            List<DocumentSnapshot> documentSnapshots = queryDocumentSnapshots.getDocuments();
            mComments.clear();
            for(DocumentSnapshot documentSnapshot:documentSnapshots){
                CommentInfo commentInfo = documentSnapshot.toObject(CommentInfo.class);
                mComments.add(commentInfo);
            }
            mAdapter.refreshView(mComments);
        }
    });
```

The Goods ID is used as the node to load the comment list of the product.

2.3.6 Add Welfare Activities

Adding public welfare activities is completed by the system administrator in the firebase console. The system administrator can log in to firebase and find the welfare collection in the firestore development component. In the welfare collection, you can select add document to add a public welfare activity. The fields of public welfare activity include activity ID, public welfare activity cover, title, description, goal and progress, as well as heat value and creation time. After adding and saving, users can see the items of the activity in the public welfare activities section.


welfare

Add document

6VCQiRvKqYMa08CB6uob

6VCQiRvKqYMa08CB6uob

Start collection

Add field

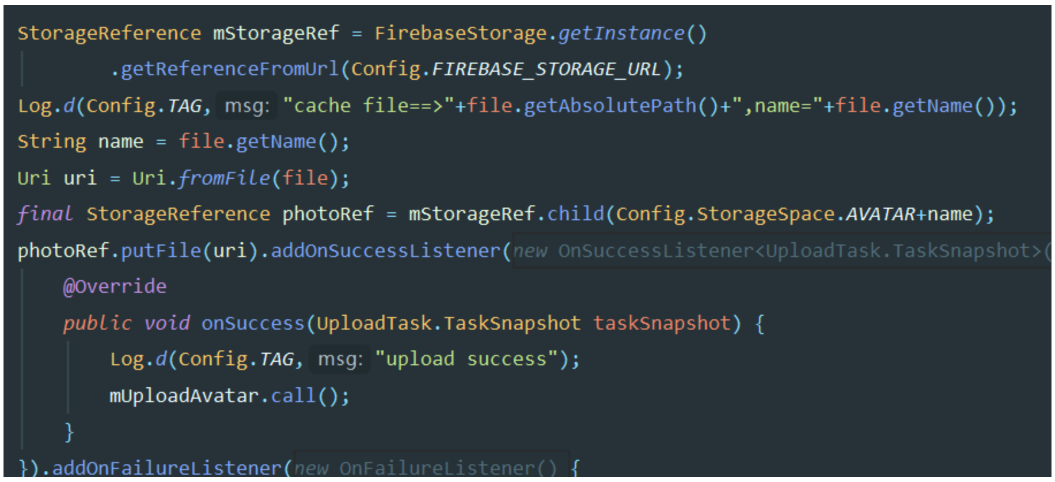
```
cover: "https://ss1.bdstatic.com/70cFuXSh_Q1YnxGkpoWK1
createTime: December 16, 2020 at 12:00:00 AM UTC+8
current: 0
description: "welfare description"
destination: 1000000
hotDegree: 0
id: "1"
progress: 60
title: "welfare title"
```

2.3.7 Participate welfare Activities

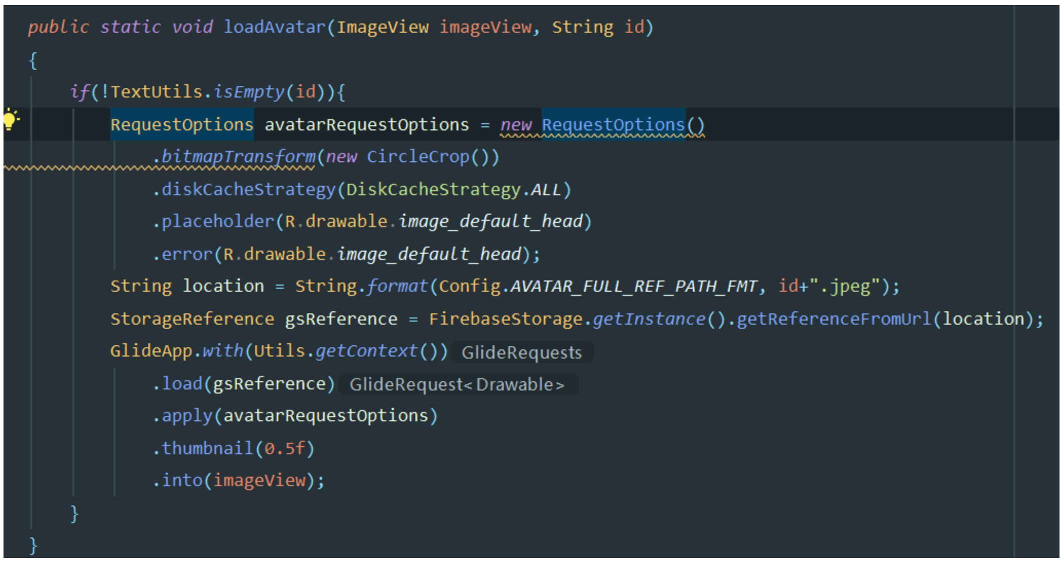
If a public welfare activity is still in progress, the user can choose to participate in the public welfare activity. The process of participating in the public welfare activity is very simple. Users need to input their own contribution value, and then update the public welfare progress. In the records of public welfare activities, the current field represents the current contribution value that has been reached, and destination is the total required contribution value. When current is greater than or equal to destination, it means that the public welfare activity has been completed. Users who participate in public welfare activities will be saved in a contribution table. In the contribution table, the ID of public welfare activities is used as the node to save all the contributor information.

2.3.8 Update User Information

Users can update their own user information. The logged in users can choose to edit their personal information. In the editing page, they can modify their avatars, add signature information, and add interests and hobbies. These can be saved in the user table. We need to pay attention to the update of the user's Avatar. The user's Avatar needs to be uploaded to the server after the user selects it. Similarly, before uploading, we need to compress and rename the image to the current user's ID value, so that we can access the user's Avatar through the user's ID value.



```
StorageReference mStorageRef = FirebaseStorage.getInstance()
        .getReferenceFromUrl(Config.FIREBASE_STORAGE_URL);
Log.d(Config.TAG, msg: "cache file==>"+file.getAbsolutePath()+",name="+file.getName());
String name = file.getName();
Uri uri = Uri.fromFile(file);
final StorageReference photoRef = mStorageRef.child(Config.StorageSpace.AVATAR+name);
photoRef.putFile(uri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>(
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        Log.d(Config.TAG, msg: "upload success");
        mUploadAvatar.call();
    }
}).addOnFailureListener(new OnFailureListener() {
```

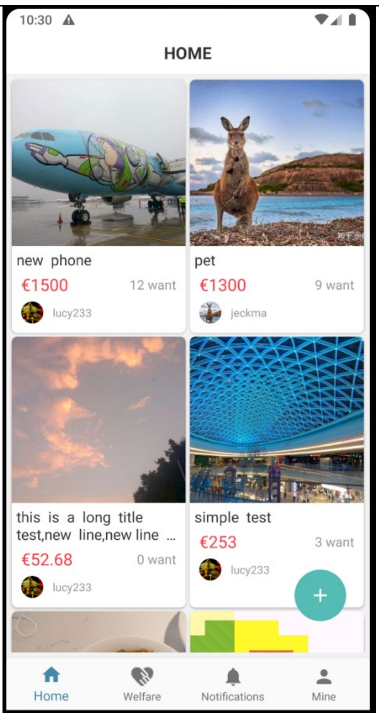
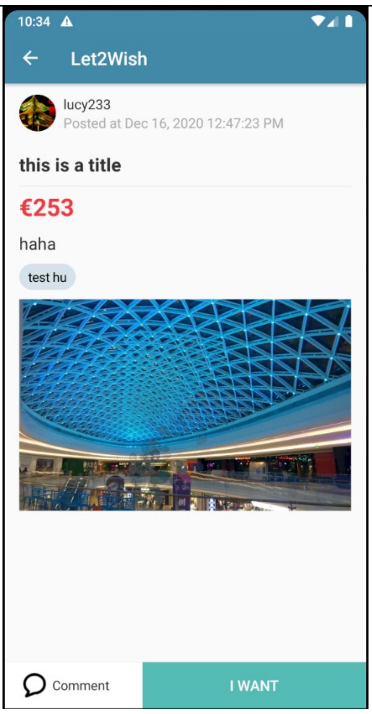


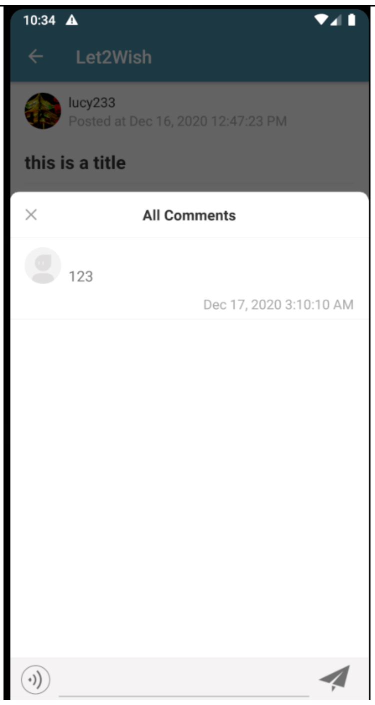
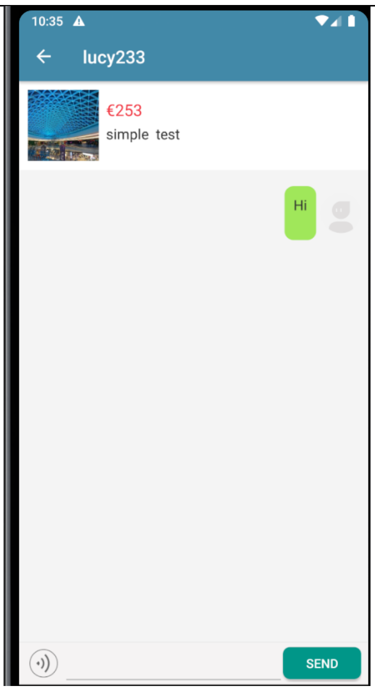
```
public static void loadAvatar(ImageView imageView, String id)
{
    if(!TextUtils.isEmpty(id)){
        RequestOptions avatarRequestOptions = new RequestOptions()
            .bitmapTransform(new CircleCrop())
            .diskCacheStrategy(DiskCacheStrategy.ALL)
            .placeholder(R.drawable.image_default_head)
            .error(R.drawable.image_default_head);
        String location = String.format(Config.AVATAR_FULL_REF_PATH_FMT, id+".jpeg");
        StorageReference gsReference = FirebaseStorage.getInstance().getReferenceFromUrl(location);
        GlideApp.with(Utils.getContext()) GlideRequests
            .load(gsReference) GlideRequest<Drawable>
            .apply(avatarRequestOptions)
            .thumbnail(0.5f)
            .into(imageView);
    }
}
```

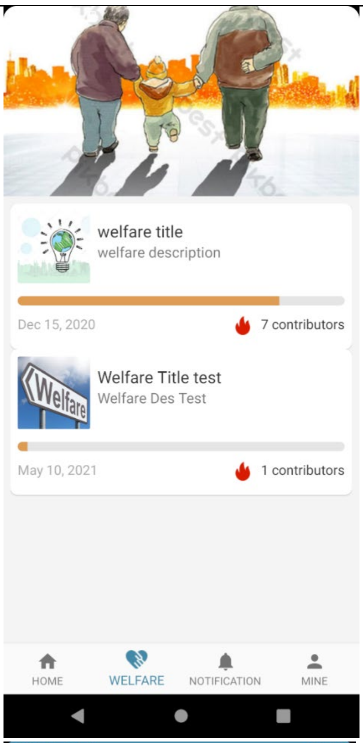
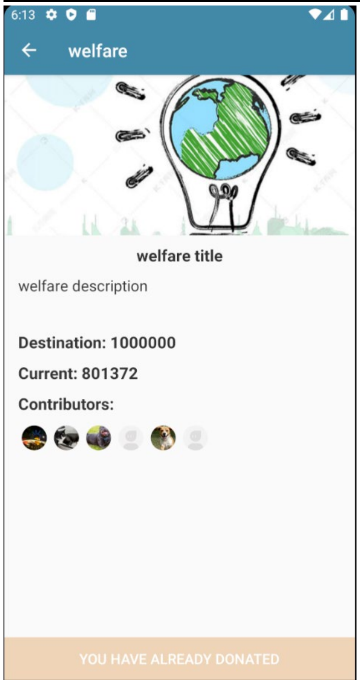
The user's Avatar is saved under avatars. We can get the location information of the avatar in the storage by splicing the access prefix and user ID of the storage, and then load the image of the location through glideapp.

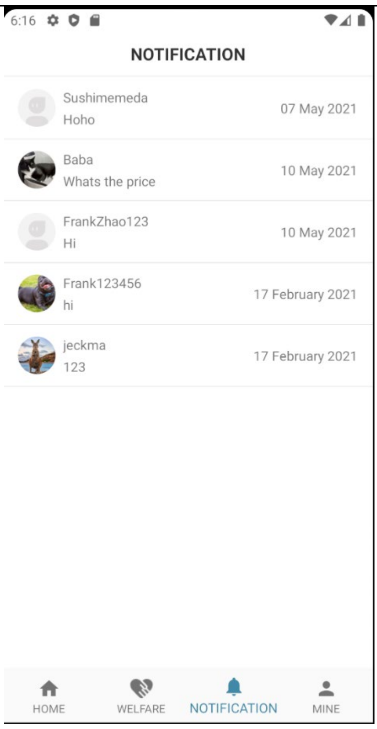
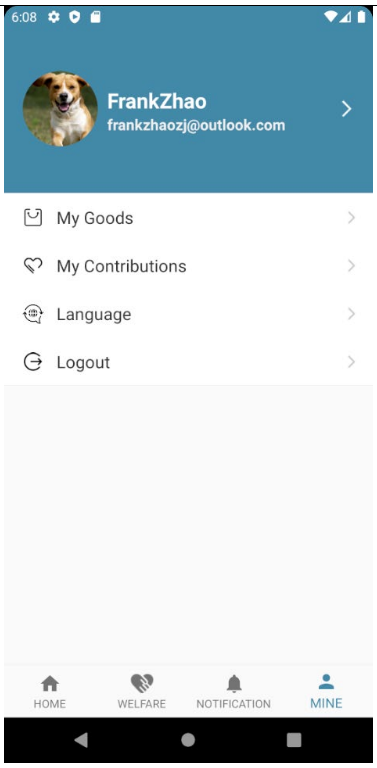
```
public static final String AVATAR_FULL_REF_PATH_FMT = "gs://flea-market-acade.appspot.com/avatars/%s";
```

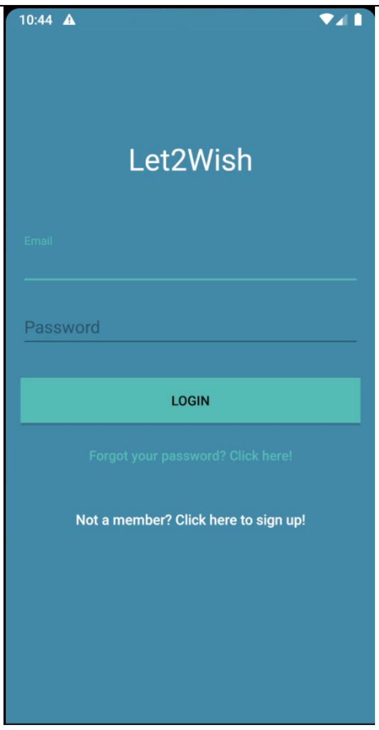
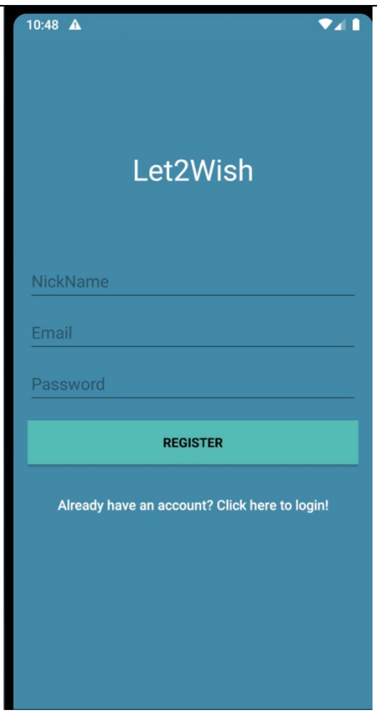
2.4 Graphical User Interface (GUI)

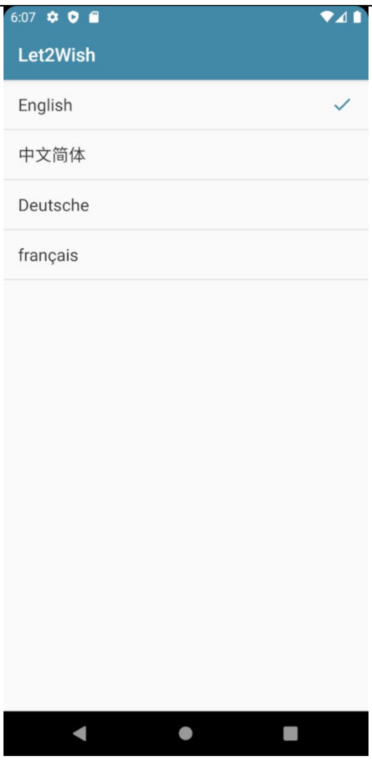
2.2.1	Home Page	 10:30 HOME new phone €1500 12 want lucy233 pet €1300 9 want jeckma this is a long title test,new line,new line ... €52.68 0 want lucy233 simple test €253 3 want lucy233 Home Welfare Notifications Mine	The user would see the list of goods on the home page. The brief info about goods would be listed.
2.2.2	Goods Detail	 10:34 Let2Wish lucy233 Posted at Dec 16, 2020 12:47:23 PM this is a title €253 haha test hu Comment I WANT	On the detail page of goods. The user would see the full detail of the goods, the user who posted, the time of posted, price and other relevant information.

2.2.3	Comments Page	 10:34 Let2Wish lucy233 Posted at Dec 16, 2020 12:47:23 PM this is a title All Comments 123 Dec 17, 2020 3:10:10 AM	The user would leave the comment on the goods detail page.
2.2.4	Chat Page	 10:35 lucy233 €253 simple test Hi SEND	The user would send message to the event creator. The event creator can reply the message from the notification page.

2.2.5	Welfare/Donation	 welfare title welfare description Dec 15, 2020 7 contributors Welfare Title test Welfare Des Test May 10, 2021 1 contributors  welfare welfare title welfare description **Destination: 1000000** **Current: 801372** **Contributors:** 	The user may donate amount of money. s

2.2.6	Notification Page	 6:16 NOTIFICATION Sushimemeda Hoho 07 May 2021 Baba Whats the price 10 May 2021 FrankZhao123 Hi 10 May 2021 Frank123456 hi 17 February 2021 jeckma 123 17 February 2021	The user would check the notification message from notification page.
2.2.7	Profile Page	 6:08 FrankZhao frankzhaozj@outlook.com My Goods My Contributions Language Logout HOME WELFARE NOTIFICATION MINE	On profile page. The user would modify the avatar, check the goods list create by them. Check contribution to welfare. Change language. Or logout.

2.2.8	Initial Page	 10:44 Let2Wish Email Password LOGIN Forgot your password? Click here! Not a member? Click here to sign up!	On this page. The new user may register an account or log in with existing account. The user may also need to restore the password if needed.
2.2.9	Register Page	 10:48 Let2Wish NickName Email Password REGISTER Already have an account? Click here to login!	The user can sign up from this page. They need to type in the Nickname, Email, Password.

2.2.10	Multi-Language	 6:07 Let2Wish English ✓ 中文简体 Deutsche français	The user can change the language at Language page.

2.5 Testing

In order to ensure that the App function meets the preset functional requirements, we need to create some test cases based on user use cases according to the requirements. These test cases cover the basic functions of app, and the test results are as follows:

ID	Test Case Description	Expect	Result
Test Case 1	Check user can register success using valid account and password	Register success and navigate user to login page	pass
Test Case 2	Check user register using invalid account and password	Register failed and display error message	pass
Test Case 3	Check user can login with Valid account and password	Login success and navigate user to home page	Pass
Test Case 4	Check user login with invalid account and password	Login failed and display error message	pass
Test Case 5	Check user enter home page and display the list of second-hand goods published by users	Enter home page and display goods list	Pass

Test Case 6	Check user can publish second-hand goods information	User enter the commodity information editing page to publish, If the release is successful, it will be displayed in the home page list	pass
Test Case 7	Check user can publish goods to be auctioned	User enter the commodity information editing page to publish, If the release is successful, it will be displayed in the home page list	pass
Test Case 8	Check user can view the published products	In my home page, click my goods to see the products that users have published.	pass
Test Case 9	Check user can delete the published products	In my goods List,Long press a commodity information record to delete.	pass
Test Case 10	Check users can view the product details published by other users	Click a commodity in the list of commodities on the main page to enter the commodity details page, and display the detailed information of the goods,including the publisher information,commodity information, etc.	pass
Test Case 11	Check user can comment on published products	User can click the Comment button, enter the comment content in the pop-up box, and click publish to see it in the comment list	pass
Test Case 12	Check user wants to buy the specified product.	Click the "I want" button to further inquire about the details of the product, as well as the price and logistics negotiation with the seller who publishes the product.	pass
Test Case 13	Check user can bid on the goods at auction	On the product details page, click the auction button and enter the price acceptable to the user in the pop-up box to participate in the auction.	pass

Test Case 14	Check user can view the price list of auction goods	On the item details page, view other users' bids for the item.	pass
Test Case 15	Check User can view the message information of other users on the notification page.	User can see the latest chat list on the notification page, and click any one to see the history chat record	pass
Test Case 16	Check user can view the public welfare list in the public welfare page	User can see the public welfare list published by the system	pass
Test Case 17	Check user can view a public welfare activity in the public welfare list to view specific details	User can view public welfare activities to see the specific details	pass
Test Case 18	Check user can participate in the public welfare activities released by the system, and those who have participated can no longer participate	User can click the donate button to participate in public welfare activities	pass
Test Case 19	Check user can view the public welfare activities they participate in	User can see the list of public welfare they participate in on their personal page, and can click to view the details	pass
Test Case 20	Check user can update and modify their personal information on the personal information page.	Click the nickname of the personal page to enter the personal information page to view the current name	pass

2.6 Evaluation

Based on the above test cases, the basic functions of app are basically covered and tested. The passing of these functional tests also means that the data services provided by firebase in the background have undergone basic verification. Of course, these test cases do not completely cover all the functions of the app, and some potential bugs may not be found for the time being. This needs to be tracked and improved continuously. In addition, we can rely on the test lab provided by firebase to track the existing problems of the app. Based on the test lab, you can see which devices the app crashed on. Based on the details of the code log and the operation screen shots and videos, you can easily locate problems, which is very helpful for the subsequent optimization of the app.

3.0 Conclusions

So far, the development of the app has been completed, and the basic functions have passed the test, basically realizing the research content of this topic. In the project development, we use Android and firebase platform for development. Firebase platform provides stable data and services for app development, such as data file storage, real-time message notification, login, registration and authentication, which enables us to quickly realize the basic functions of the app to meet the needs of users. Android platform has a large number of users, and the open community makes it very easy for us to build our own app to complete the core functions of the project. However, due to limited conditions and resources, the functions of the app are not perfect. Although firebase provides many system background services, it is not omnipotent. If the app needs to add online payment function in the later stage, the system needs to develop its own background services, which is also a great challenge for me. At the same time, a lot of work needs to be done in terms of APP user experience.

4.0 Further Development or Research

Although the current functions of APP have met the most basic user needs, as a perfect system and considering the future commercialization, there are still some functions that need to be further completed and improved in the future, including the following aspects:

(1) Considering the perfect upgrade service for the app, it can remind users to upgrade the version after the application has added new functions, so as to continuously iterate the product functions of the app.

(2) In order to enrich the product functions, we develop our own back office services and provide users with online payment functions in combination with firebase services. Users can directly conduct online transactions in the app after selecting products.

(3) Classification and search for platform products is convenient for users to find products they are interested in. Background services can also develop recommendation system to bring better user experience.

(4) Short video product introduction is supported. Users can shoot videos of products they sell and upload them to the platform for more intuitive introduction.

(5) Improve the information of platform users, increase credit rating for users, buyers can score and evaluate sellers, and support sellers to authenticate personal information to increase the authenticity of platform users.

5.0 References

Platform Architecture | Android Developers.2021.Available at:

https://developer.android.com/guide/platform [Accessed:2 March 2021]

Understanding Navigator | Material Design.2021.Available at:

https://material.io/design/navigation/understanding-navigation.html[Accessed:5 March 2021]

Android Jetpack | Android Developers.2021.Available at:

https://developer.android.com/jetpack[Accessed:10 March 2021]

Structure Your Database | Firebase Realtime Database.2021.Available at

https://firebase.google.com/docs/database/web/structure-data [Accessed:11 March 2021]

Installation & Setup on Android | Firebase Realtime Database.2021.Available at

https://firebase.google.com/docs/database/android/start [Accessed:16 March 2021]

Get Started with Firebase Authentication on Android.2021.Available at

https://firebase.google.com/docs/auth/android/start [Accessed:22 March 2021]

Get Started with Cloud Firestore | FireBase.2021.Available at

https://firebase.google.com/docs/firestore/quickstart [Accessed:3 April 2021]

Get Started with Cloud Storage on Android.2021.Available at

https://firebase.google.com/docs/storage/android/start [Accessed:10 April 2021]

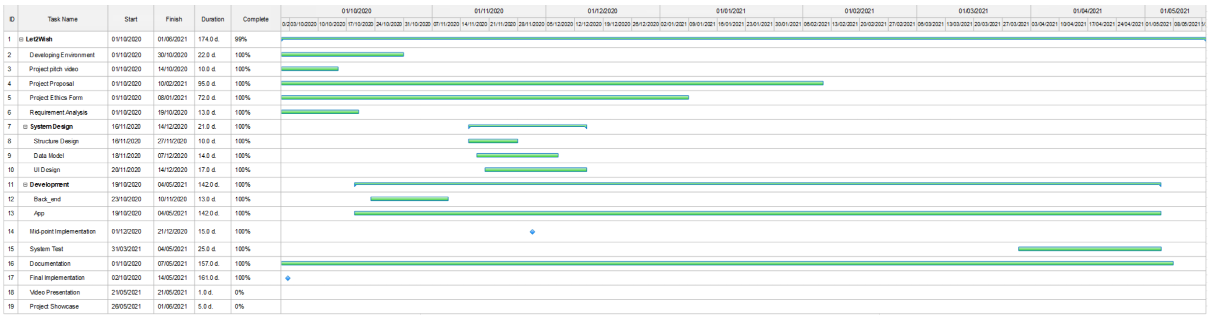
Hongbeom.[2021].Create Android app with MVVM pattern simply using Android Architecture Component.Available at:

https://medium.com/hongbeomi-dev/create-android-app-with-mvvm-pattern-simply-using-android-architecture-component-529d983eaabe.[Accessed:12 March 2021]

6.0 Appendices

This section should contain information that is supplementary to the main body of the report.

6.1. Project Plan



ID	Task Name	Start	Finish	Duration	Complete
1	⊟ **Let2Wish**	01/10/2020	01/06/2021	174.0 d.	99%
2	Developing Environment	01/10/2020	30/10/2020	22.0 d.	100%
3	Project pitch video	01/10/2020	14/10/2020	10.0 d.	100%
4	Project Proposal	01/10/2020	10/02/2021	95.0 d.	100%
5	Project Ethics Form	01/10/2020	08/01/2021	72.0 d.	100%
6	Requirement Analysis	01/10/2020	19/10/2020	13.0 d.	100%
7	⊟ **System Design**	16/11/2020	14/12/2020	21.0 d.	100%
8	Structure Design	16/11/2020	27/11/2020	10.0 d.	100%
9	Data Model	18/11/2020	07/12/2020	14.0 d.	100%
10	UI Design	20/11/2020	14/12/2020	17.0 d.	100%
11	⊟ **Development**	19/10/2020	04/05/2021	142.0 d.	100%
12	Back_end	23/10/2020	10/11/2020	13.0 d.	100%
13	App	19/10/2020	04/05/2021	142.0 d.	100%
14	Mid-point Implementation	01/12/2020	21/12/2020	15.0 d.	100%
15	System Test	31/03/2021	04/05/2021	25.0 d.	100%
16	Documentation	01/10/2020	07/05/2021	157.0 d.	100%
17	Final Implementation	02/10/2020	14/05/2021	161.0 d.	100%
18	Video Presentation	21/05/2021	21/05/2021	1.0 d.	0%
19	Project Showcase	26/05/2021	01/06/2021	5.0 d.	0%

6.1. Ethics Approval Application (only if required)

6.2. Reflective Journals

1st Month

For this month. I started with preparing the ideas for the project. I started with the idea of develop a blog app as I have 2 years' experience about it. After checking the previous project. I realized that the codes in that project is not that formally. It is kindly complicated and insufficient clear to use them as references. So, I decided to give up it and chose a new topic.

During that situation. I talked with other people and did some search from online sources. I finally chose to make a second-hand trading website. The inspiration is coming from Facebook Market and XianYu (Most famous second-hand trading company in China).

As I have experience about the developing the blog app. It strikes me that I might keep using Firebase as the data storage. In addition to this, I also know some codes about the uploading, refreshing, logging, register, and other functions. This may help me to avoid some risk during develop the new project.

During the process of making the video recording for pitch. I start to plan the process of project. I would start with designing the UI. Then comes with the function.

At current situation. I'm working for the project proposal. I'm also trying to get contact with the supervisor for the further instructions.

2nd Month

For this month. I was working for the chat feature. I spend a lot of time to find the relative documentation that may help me to find the right solution.

Prototype of the app has been created. The color design, structure of the design, development environment has been totally set up.

Profile page has been created. I’m planning to add more features. Including multi language support, edit/delete post, Bio profile, Customer photo, etc.

I also spend a few days for working on the documentation.

3rd Month

For this month. I was working for the mid-term documentation and coding. For the documentation. I finished the technical report, including Introduction, system, Implementation, GUI. And also amended the proposal draft.

Besides it. I also finished a presentation. Including PowerPoint slides. Presentation speech draft. A dozen of video clips. A few editing work to combine the video presentation.

For the development of the project. I have tried to implement a few features to run. Including register page, logging page, reset password. Publish goods and relative fragment. I also finished the draft of goods detail. The further plan relate to this feature would be auto-generate for print screen. The hardest part of developing till now is the chat function. I spend a few days trying to get inspiration from research. And finally found the way to implement it.

From the suggestion from my friend. I’m also trying to improve the simplification the app loading usage. With the database design. The snapshot is being used to improve the customer experience.

For the further plan. The coding part should be the very important thing for the next two months.

4th Month

For this month. I added delete feature. And also deployed the welfare feature to the back-end. The administrator (Me) can create the welfare page.

I spend a few days for writing the report and other documentation.

Fixed a few bugs relate to message features, the previous error may cause the app shut down or not appear the new message. By modify the manifest file and gradle file, the error disappeared.

Start to do the research of the difference between debug version and release version.

5th Month

For this month. I fixed the multi-language support feature. By get different xml file, the string can be converted to different language. I also adjusted a few features to make the app get more pleasantly.

Logo has been created. And came with the different resolution for multiple use.

I started to prepare for the test. Also, I tried to do the research on how to publish the app to the Google Play store.

6th Month

For this month. I fixed the refresh feature by uploaded the Gradle file. I also started to create the developer account of Google Play store. The first Internal testing has been set up. I had created the keystore. The app is signed and, install pack is being generated.

The pack is uploaded to the Google play store. A few warning and errors relate to debugging has appeared. They went gone after I removed the debugging code.

Start to updating with more language support.

7th Month

For this month. I did several test for the app. I tried on different virtual device with different API. As the Google storage policy being changed after API 28. I faced some compatibility issue. After modified the Application list file. The issue disappeared.

As this is my first time trying to release the production app. I'm still working on Google Play Console. Trying to release the app ASAP.

Calendar has been created by me. As the language setting issue happened. I created the second one.

6.3. Other materials used

6.3.1. Proposal

National College of Ireland

Project Proposal

< Software Project>

<Trading Platform(undetermined)>

<Specialisation>

<Academic Year 2020>

<ZIJIAN ZHAO>

<X16112962>

<x16112962@student.ncirl.ie>

Contents

1.0 Objectives

The research goal of this project is to build a second-hand commodity trading platform. The system is aimed at the campus student user groups. Students can publish second-hand goods, auction, comment and donate through the platform. The establishment of a campus second-hand trading market can not only facilitate the students in the school, but also create a campus culture atmosphere of diligence, thrift and simple life. It also exerts the students' ability of independent entrepreneurship, practice and innovation, and enhances their economic consciousness and survival consciousness.

2.0 Background

In recent years, with the great enrichment of people's material life, the market scale of second-hand goods or second-hand goods has grown explosively, At the same time, the purchasing power of students is also gradually improving, and the waste of campus resources is becoming increasingly prominent. In order to more reasonably match and use the resources, the campus second-hand market has also emerged. How to effectively use second-hand goods has become a serious problem that needs to be solved urgently. The traditional second-hand trading platform needs to provide a trading place for trading, and needs to post advertisements as the middle of the transaction. The uncertainty and instability of time and place increase the difficulty of trading, and the flow of information lags behind. However, with the rapid development of the Internet, especially the rapid development of mobile Internet represented by smart phones provides new possibilities for commodity trading. Compared with the traditional commodity trading mode, the Internet information dissemination speed is fast, easy to obtain, abundant resources, flexible and simple operation, which can avoid many disadvantages of traditional commodity trading and provide commodity exchange More convenient and efficient way to promote the rapid development of the second-hand market to maximize the use of resources.

3.0 Technical Approach

Every year, a group of new students come to the campus to start a new study life. At the same time, a large number of graduates leave the campus to enter the society. Between the two groups, one side is the demand side of learning and living supplies, and the other side is the supplier of learning and daily necessities, because the vast majority of graduates are not willing to take away daily necessities and learn at the same time The books in their childhood are either thrown away or sold as waste products, while the junior students rush to the bookstore to buy new books. Therefore, I think building a second-hand trading platform among students can better solve this kind of demand problem.

In order to ensure the fairness, transparency and legitimacy of the platform, the following principles and methods must be followed in the process of solving such requirements:

1) The platform should provide convenient services for users to collect necessary user information, such as gender, grades, email, etc.

2) The user's sensitive information should be treated by necessary technical means to prevent illegal access and disclosure

3) If necessary, do questionnaire survey or interview with users to optimize the platform function and provide the entrance for feedback and suggestions

4) The platform should prohibit the trading of prohibited goods, such as violence tools, alcohol, drugs, etc., prohibit the release of bad information, and punish those who violate the rules

5) The commodity information released by the platform should be related to the life and study of students, and irrelevant articles should not be released

Considering the usage scenarios and habits of student users, mobile Internet and mobile applications are the network and software that most students are more easily exposed to. The platform will rely on the development and deployment, and also rely on the third-party cloud server to provide data support. The platform will be a typical C / S architecture system.

4.0 Special Resources Required

1) Hardware:

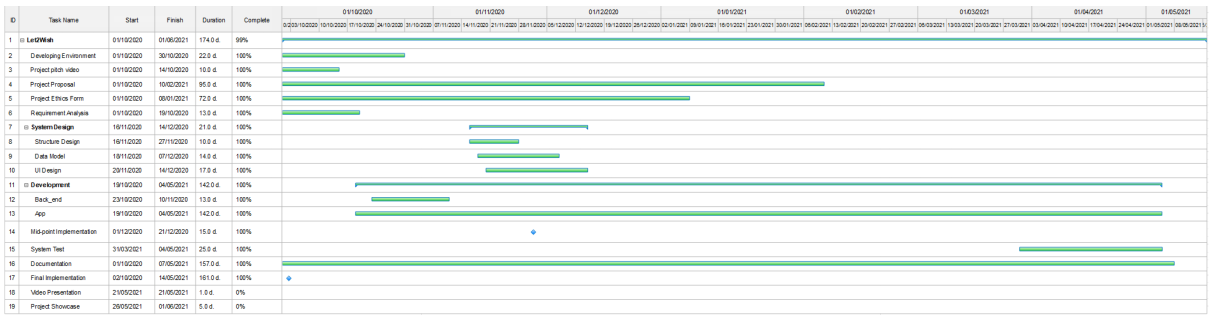
Android mobile phone, PC

2) Books:

《Android 9 Development Cookbook》Rick Boyer

《Android Programming》Bill Phillips, Brian Hardy

5.0 Project Plan



ID	Task Name	Start	Finish	Duration	Complete
1	⊟ **Let2Wish**	01/10/2020	01/06/2021	174.0 d.	99%
2	Developing Environment	01/10/2020	30/10/2020	22.0 d.	100%
3	Project pitch video	01/10/2020	14/10/2020	10.0 d.	100%
4	Project Proposal	01/10/2020	10/02/2021	95.0 d.	100%
5	Project Ethics Form	01/10/2020	08/01/2021	72.0 d.	100%
6	Requirement Analysis	01/10/2020	19/10/2020	13.0 d.	100%
7	⊟ **System Design**	16/11/2020	14/12/2020	21.0 d.	100%
8	Structure Design	16/11/2020	27/11/2020	10.0 d.	100%
9	Data Model	18/11/2020	07/12/2020	14.0 d.	100%
10	UI Design	20/11/2020	14/12/2020	17.0 d.	100%
11	⊟ **Development**	19/10/2020	04/05/2021	142.0 d.	100%
12	Back_end	23/10/2020	10/11/2020	13.0 d.	100%
13	App	19/10/2020	04/05/2021	142.0 d.	100%
14	Mid-point Implementation	01/12/2020	21/12/2020	15.0 d.	100%
15	System Test	31/03/2021	04/05/2021	25.0 d.	100%
16	Documentation	01/10/2020	07/05/2021	157.0 d.	100%
17	Final Implementation	02/10/2020	14/05/2021	161.0 d.	100%
18	Video Presentation	21/05/2021	21/05/2021	1.0 d.	0%
19	Project Showcase	26/05/2021	01/06/2021	5.0 d.	0%

6.0 Technical Details

The front end of the system is an Android mobile app, which is developed by kotlin + Java, and the overall architecture is developed by Google's jetpack. Jetpack is a suite composed of multiple libraries, which can help developers follow best practices, reduce template code, and write code that can run consistently in various Android versions and devices, so that developers can concentrate on writing important code. The main libraries are as follows:

activity *	Access composable APIs built on top of Activity.
appcompat *	Allows access to new APIs on older API versions of the platform (many using Material Design).
camera *	Build mobile camera apps.
compose *	Define your UI programmatically with composable functions that describe its shape and data dependencies.
databinding *	Bind UI components in your layouts to data sources in your app using a declarative format.
fragment *	Segment your app into multiple, independent screens that are hosted within an Activity.
hilt *	Extend the functionality of Dagger Hilt to enable dependency injection of certain classes from the androidx libraries.
lifecycle *	Build lifecycle-aware components that can adjust behavior based on the current lifecycle state of an activity or fragment.
Material Design Components *	Modular and customizable Material Design UI components for Android.

navigation *	Build and structure your in-app UI, handle deep links, and navigate between screens.
paging *	Load data in pages, and present it in a RecyclerView.
room *	Create, store, and manage persistent data backed by a SQLite database.
test *	Testing in Android.
work *	Schedule and execute deferrable, constraint-based background tasks.
ads	Get an advertising ID with or without Play Services.
annotation	Expose metadata that helps tools and other developers understand your app's code.
arch.core	Helper for other arch dependencies, including JUnit test rules that can be used with LiveData.

The system back-end is developed using firebase provided by Google. Firebase is designed by Google cloud platform for developers, providing various tools to solve infrastructure problems. Firebase provides foundational work and tools that allow developers to focus on developing quality applications and expanding the user base. We will use the authentication, cloud storage, cloud firestore and real-time database packages in the back-end development.

（1）Authentication

Manage your users in a simple and secure way. Firebase Auth offers multiple methods to authenticate, including email and password, third-party providers like Google or Facebook, and using your existing account system directly. Build your own interface, or take advantage of our open source, fully customizable UI.

（2）Cloud Storage

Store and share user-generated content like images, audio, and video with powerful, simple, and cost-effective object storage built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

（3）Cloud Firestore

Extend your app with custom backend code without needing to manage and scale your own servers. Functions can be triggered by events, which are emitted by Firebase products, Google Cloud services, or third parties, using webhooks.

（4）Realtime Database

Realtime Database is Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime. We recommend Cloud Firestore instead of Realtime Database for most developers starting a new project.

7.0 Evaluation

The system test includes two aspects: front-end test and back-end test. For the front-end test, function test and compatibility test are required. The test method is mainly black box test, which mainly verifies whether the page logic is correct, whether the data returned by the interface is correct, and whether there are compatibility problems on different versions of the system. The back-end test is mainly to test the interface data, and it is necessary to write test cases to verify the correctness of the data.