INF 551 Project Final Report

Hyun Jun Choi {choi797@usc.edu}
Wanjin Li {wanjinli@usc.edu} Taoran Ju {taoranju@usc.edu}

1. Overview

Our idea is to develop an Android application called **DiSCover**. It is designed for the University of Southern California's (USC) community, including students, faculty and alumni, in order to recommend good resources in the USC campus and allow them to share their extraordinary experiences with each other. Students can post photos, pictures, choose categories, rate an item and write several texts to describe it, and other students can ask a question and chat with the author through the application (app). We used Firebase as the database to store all the data and resources and implemented the app via the MVC structure.

2. Architecture

2.1. Components

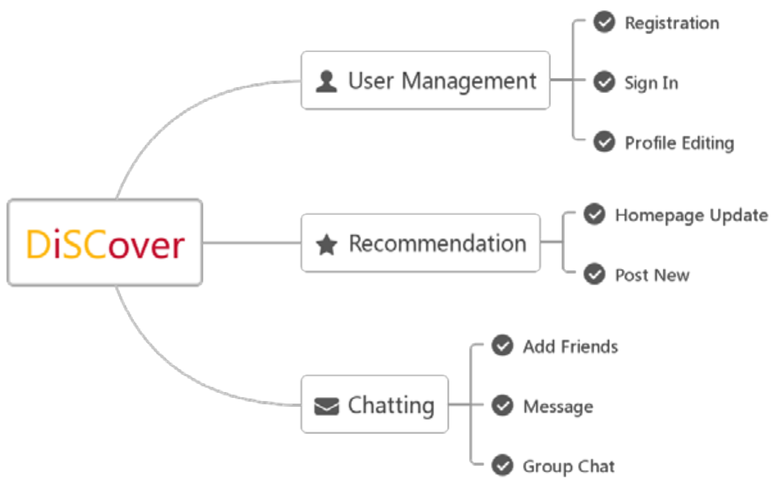


Fig. 1: Major components in DiSCover

DiSCover has three major components: user management, recommendation posts and chatting. User management components mainly implement the user registration, sign in and profile editing function. The recommendation post component supports the user in submitting new posts and includes many details; a user can update all the recommendation posts on the homepage in real-time. The chatting component realizes the communication between users, the ability to add friends, send and receive messages and group chatting.

2.2. User Interface

We used Java to develop an Android app and chose API at level 26, which is the newest edition. When designing the UI, we used the representative USC style. We also imported some external libraries, such as Picasso, Butterknife, and Firebaseui, to make the user interface more friendly and better looking.

2.3. Firebase

2.3.1. Database

The real-time database in Firebase is a NoSQL, JSON, and cloud-hosted database, and we used it to store the user information and post records. Within the users' group, it generates a unique UID for every user when they register for the first time. For each user, we store the URL of the user's profile image in both storage and the username.

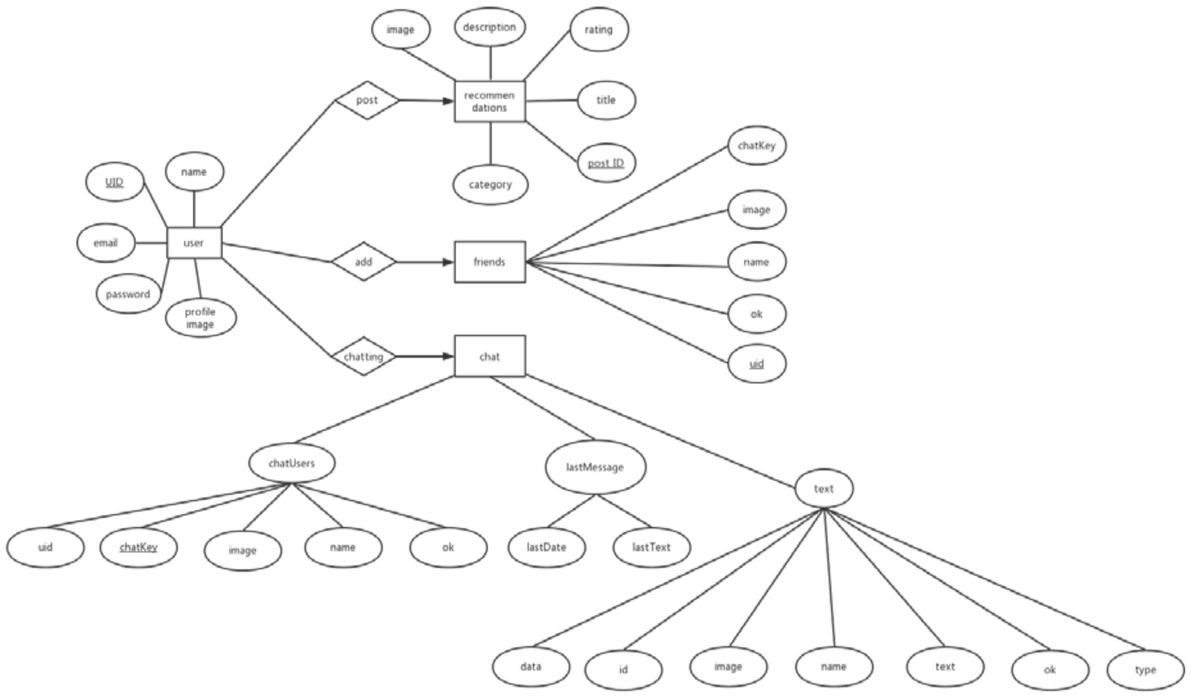
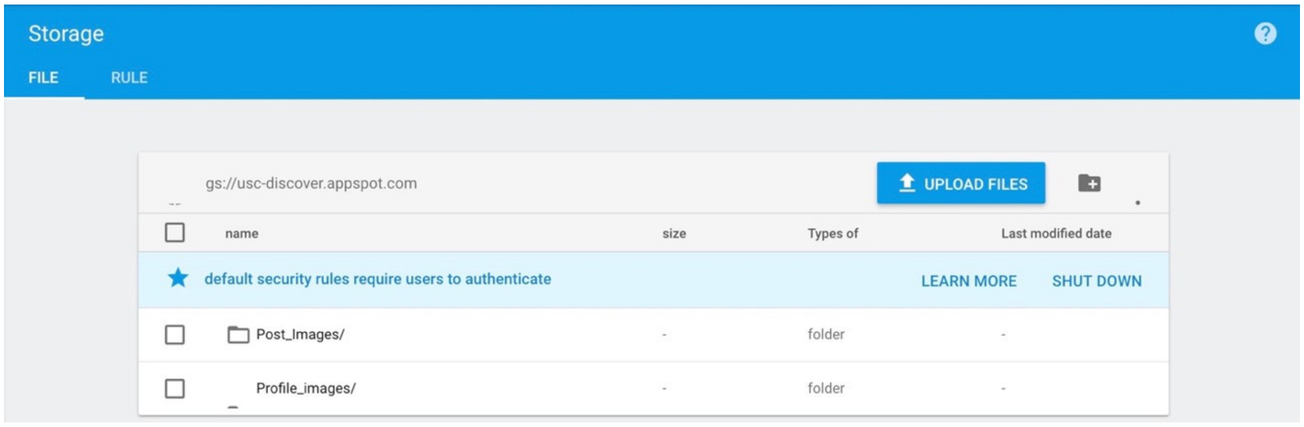


Fig. 2: ER diagram

2.3.2. Storage

Storage in Firebase is designed for keeping media resources like a picture, audio and videos. We use storage to store all the image files. The ‘Profile_images’ file folder contains the photos of a user’s profile and the file folder also contains the pictures for the user to upload when posting recommendations. Each image in storage has a unique URL address which can be stored in the database to later infer to this file.



gs://usc-discover.appspot.com

default security rules require users to authenticate

name	size	Types of	Last modified date
Post_Images/	-	folder	-
Profile_images/	-	folder	-

Fig. 3: Storage in Firebase

3. Implementation

3.1. Login

In this page, a user can log in to their own account, which includes email and password. After entering this information, the user then clicks the ‘LOGIN’ button and the words ‘Starting sign in…’ will appear on the screen. After login, the application will send the user’s email and password to Firebase and this will check whether the user’s information exists in the database. If the information is wrong, the screen will display ‘Error Login’.

If a user has set up their profile information, the interface will be switched to the main one where the user can see information posted by other users. In addition, the user can also log in by using a Google account. After clicking the ‘Google sign in’ button, the user chooses his or her Google account to log in to.

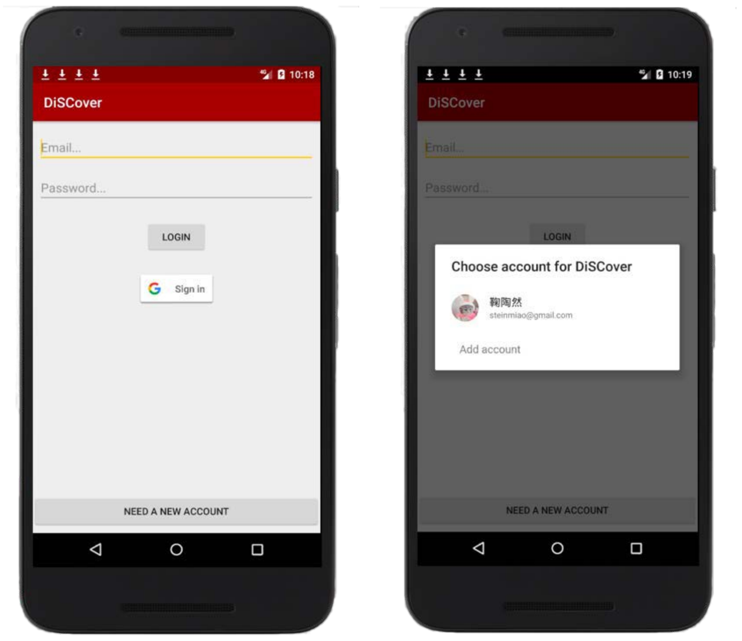


Fig. 4: Log in page / Google log in page

3.2. Set-Up

If a user has not set up their profile after registering an account or logging in, the interface will be switched to 'set up interface'. People can click the profile image and choose photos from the gallery. After choosing a photo, the user can adjust the size and direction of the photo and click the 'CROP' button. The user can then enter his or her display name. By clicking 'FINISH SETUP', the user will see the main interface. When a user wants to change the profile photo, the user can click the button on the top right of the main interface, and click the 'settings' button. This will allow the information to be changed or updated.

After set up, the application will send the image and name to the Firebase and the information of this user will be updated.

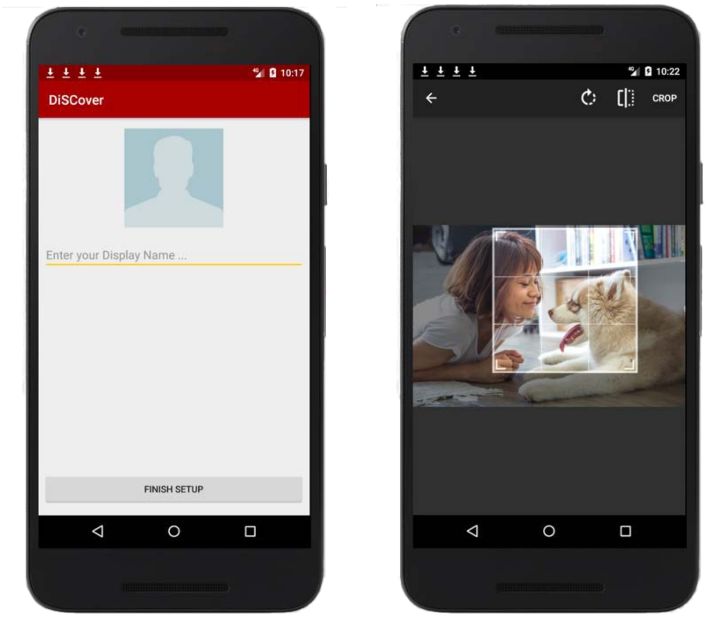
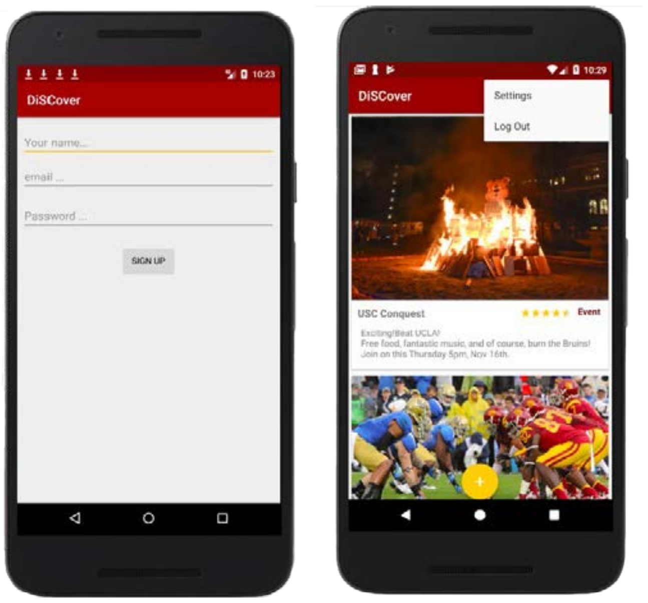


Fig. 5: Set up page / Choose a photo for profile

3.3. Register

If a user does not have an account in our application, the user can click ‘NEED A NEW ACCOUNT’ on the log in page. The application will send the user information to the Firebase and the database will store the user’s name, email and create a unique UID for him or her.

After registering and setting up the account, the user can enter into the main page. If a user wants to log out, the user just needs to click the button on the top right. When the button ‘Log Out’ is clicked the user will go back to the sign-in page. At the backend, Firebase will sign out too.



Exciting!Beat UCLA!
Free food, fantastic music, and of course, burn the Bruins!
Join on this Thursday 5pm, Nov 16th.

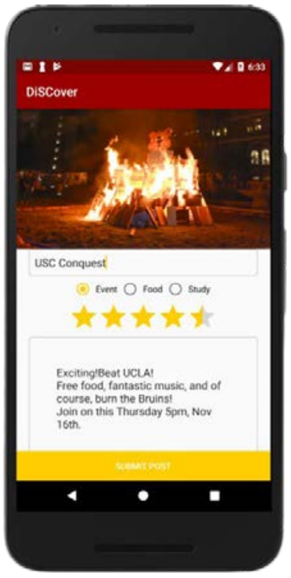
Fig. 6: Register page / Settings and log out function

3.4. Post Recommendation

3.4.1 Post Contents

In the posting page, users are able to create and post their own recommendations. For each post, a user can attach a picture, add a title, choose the category for the post, rate and recommend, and write some text description about the recommendation.

Specifically, we define three categories of post: food, event, and study. By using a RadioGroup widget to store them, the user can only select one category out of the three options. For the rating function, the full score is five stars and the step size is 0.5.

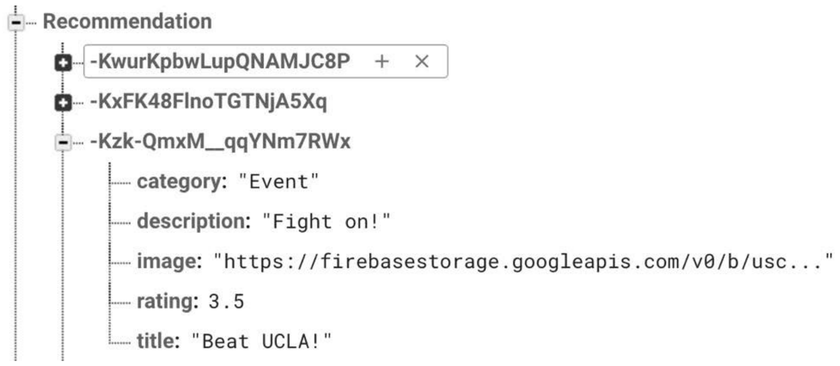


Exciting!Beat UCLA!
Free food, fantastic music, and of course, burn the Bruins!
Join on this Thursday 5pm, Nov 16th.

Fig. 7: Post Recommendation Page

By clicking the ‘Submit Post’ button, this recommendation can be uploaded to the list of all posts, and the page will jump back to the home page where the user can see this post and all other posts.

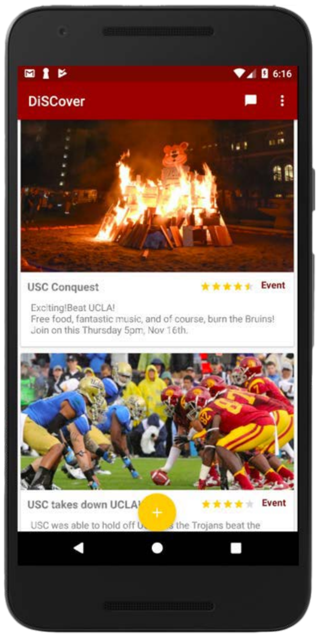
Once the button has been clicked, all the contents will be uploaded to Firebase automatically. It will generate a unique post ID below the branch of ‘Recommendation’. Inside the branch of this post ID, it stores the keys and values of title, description, category, rating, and the URL of the image (the image will have been uploaded to the storage in Firebase at the same time).



```
Recommendation
    -KwurKpbwLupQNAMJC8P
    -KxFK48FlnoTGTNjA5Xq
    -Kzk-QmxM__qqYNm7RWx
        category: "Event"
        description: "Fight on!"
        image: "https://firebasestorage.googleapis.com/v0/b/usc..."
        rating: 3.5
        title: "Beat UCLA!"
```

Fig. 8: Each post record in database

3.4.1. Homepage Display



6:16
DiSCover
USC Conquest
Event
Exciting!Beat UCLA!
Free food, fantastic music, and of course, burn the Bruins!
Join on this Thursday 5pm, Nov 16th.
USC takes down UCLA!
Event
USC was able to hold off U the Trojans beat the

Fig. 9: Homepage

The homepage is the first page for a user who has completed the sign in/sign up process or has already signed in. Once you have jumped to this page, the app will request all the branches of ‘Recommendations’ in Firebase and the user will be able to see all the recommendations. In each recommendation post it will show the title of the post, the post’s category, the rating of this subject, the text description and the picture. If any new recommendation has been uploaded, the user is able to see the update in real-time. Specifically, the user can re-rate the items on the homepage by simply dragging along the star bar and the new rating score can be updated to Firebase synchronously.

In this page, the user can also click the ‘+’ button at the bottom of the page to enter the post’s new recommendation page, click the message button to enter the chatting page or click menu to edit the user’s profile or log out of the account.

3.5. Chatting

3.5.1. Search Friends

You need to use a friend’s email address to find him or her. For example, if you want to find test7@mail.com, just type it into the yellow line and click the search button. By using the email address, it can find your friends in Firebase.

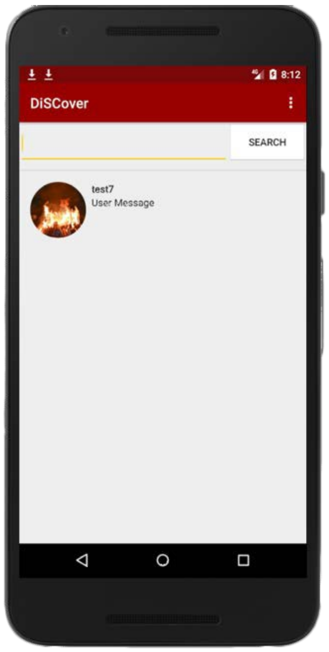
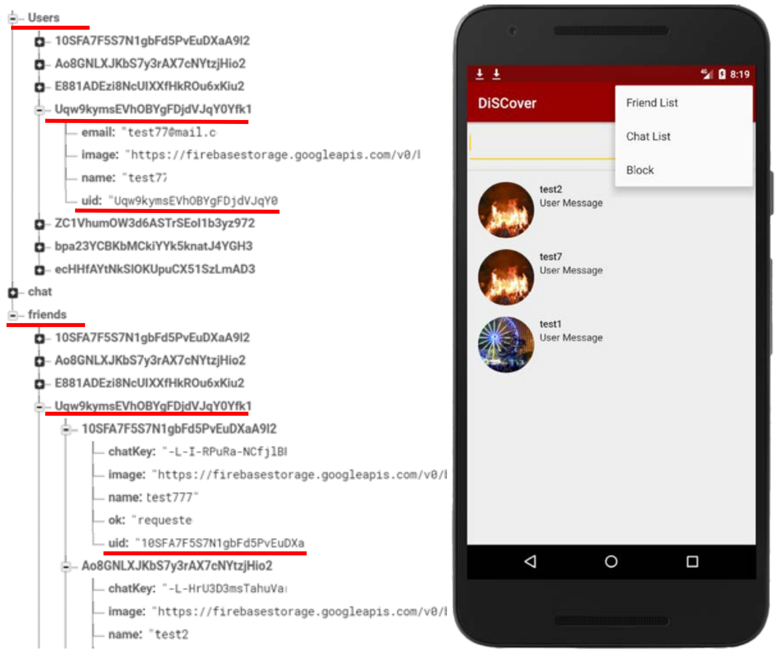


Fig. 10: Search friend

3.5.2. Friends List

If you are already registered when you click the chatting function, it will show your friends. When you want to see your friends list, just click the button with the three dots and then you will be able to see the window where there are three optioins. When you click the friends list, you can check your friends list and when you click on a friend, your friend’s UID, chatKey, image, and name are stored under your UID in the friend’s field.

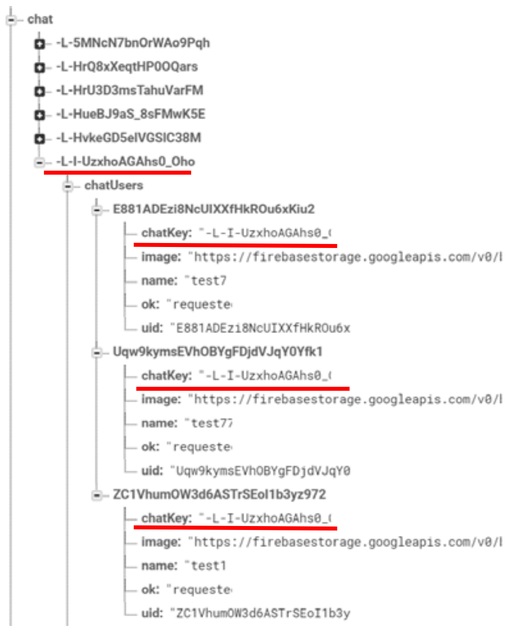


```
Users
    10SFA7F5S7N1gbFd5PvEuDXaA9I2
    Ao8GNLXJKbS7y3rAX7cNYtzjHio2
    E881ADEzi8NcUIXXfHkROu6xKiu2
    Uqw9kymsEVhOBYgFDjdVJqY0Yfk1
        email: "test77@mail.c
        image: "https://firebasestorage.googleapis.com/v0/I
        name: "test77
        uid: "Uqw9kymsEVhOBYgFDjdVJqY0
    ZC1VhumOW3d6ASTrSEol1b3yz972
    bpa23YCBKbMCkiYYk5knatJ4YGH3
    ecHHfAYtNkSIOKUpuCX51SzLmAD3
chat
friends
    10SFA7F5S7N1gbFd5PvEuDXaA9I2
    Ao8GNLXJKbS7y3rAX7cNYtzjHio2
    E881ADEzi8NcUIXXfHkROu6xKiu2
    Uqw9kymsEVhOBYgFDjdVJqY0Yfk1
        10SFA7F5S7N1gbFd5PvEuDXaA9I2
            chatKey: "-L-I-RPuRa-NCfjlBI
            image: "https://firebasestorage.googleapis.com/v0/I
            name: test777"
            ok: "requeste
            uid: "10SFA7F5S7N1gbFd5PvEuDXa
        Ao8GNLXJKbS7y3rAX7cNYtzjHio2
            chatKey: "-L-HrU3D3msTahuVa
            image: "https://firebasestorage.googleapis.com/v0/I
            name: "test2
```

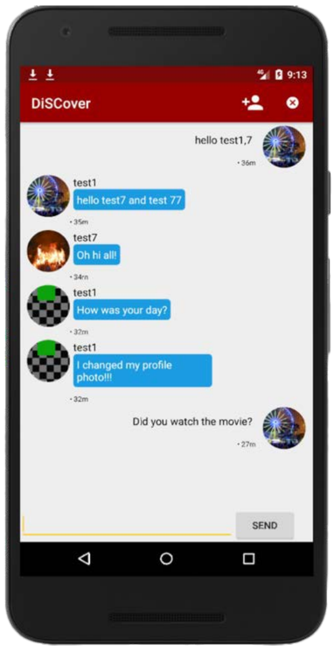
Fig. 11: Registering friends and friend list

3.5.3. Chatting with Friends

After you add your friends to your friends list, you and your friend can enjoy chatting. When you create a new chatting group, each chatting group has its unique chatKey. People who participate in the same chatting group have the same chatKey.



```
chat
  -L-5MNcN7bnOrWAo9Pqh
  -L-HrQ8xXeqtHP0OQars
  -L-HrU3D3msTahuVarFM
  -L-HueBJ9aS_8sFMwK5E
  -L-HvkeGD5elVGSIC38M
  -L-I-UzxhoAGAhs0_Oho
    chatUsers
      E881ADEzi8NcUIXXfHkROu6xKiu2
        chatKey: "-L-I-UzxhoAGAhs0_(
        image: "https://firebasestorage.googleapis.com/v0/l
        name: "test7
        ok: "requeste
        uid: "E881ADEzi8NcUIXXfHkROu6x
      Uqw9kymsEVhOBYgFDjdVJqY0Yfk1
        chatKey: "-L-I-UzxhoAGAhs0_(
        image: "https://firebasestorage.googleapis.com/v0/l
        name: "test77
        ok: "requeste
        uid: "Uqw9kymsEVhOBYgFDjdVJqY0
      ZC1VhumOW3d6ASTrSEol1b3yz972
        chatKey: "-L-I-UzxhoAGAhs0_(
        image: "https://firebasestorage.googleapis.com/v0/l
        name: "test1
        ok: "requeste
        uid: "ZC1VhumOW3d6ASTrSEoI1b3y
```



DiSCover

hello test1,7 · 36m

test1: hello test7 and test 77 · 35m

test7: Oh hi all! · 34m

test1: How was your day? · 32m

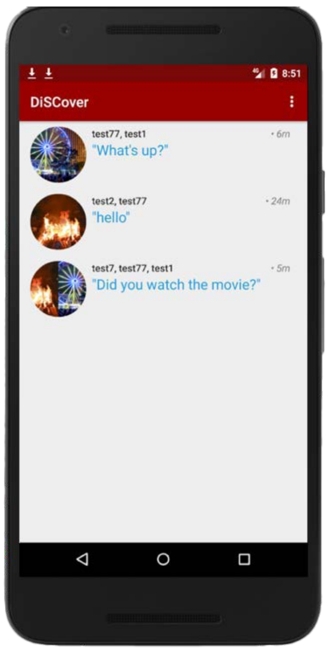
test1: I changed my profile photo!!! · 32m

Did you watch the movie? · 27m

Fig. 12: Chatting

3.5.4. Chat List

When you want to check the chat list, just click the chat list in the button with the three dots (Fig. 11). By using your UID and chatKey, it shows you what chatting groups you have.



DiSCover

test77, test1 · 6m
"What's up?"

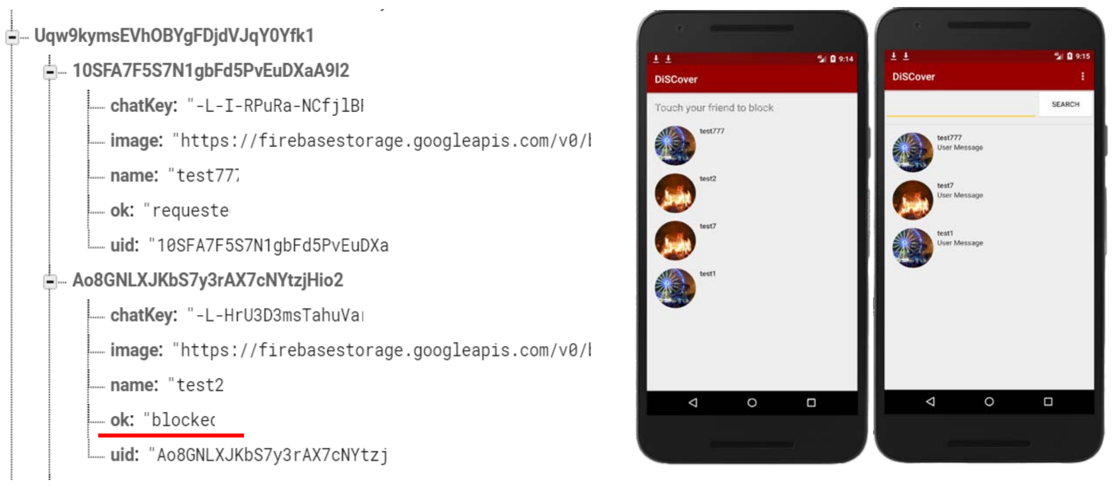
test2, test77 · 24m
"hello"

test7, test77, test1 · 5m
"Did you watch the movie?"

Fig. 13: Chat list

3.5.5. Block

If you want to block a friend, just click the block button in the button with the three dots (Fig. 11). For example, if the user at test77@mail.com blocks the user at test2@mail.com, the ok field stores "blocked". Therefore, some friends who have a 'block' value cannot chat with you.



```
Uqw9kymsEVhOBYgFDjdVJqY0Yfk1
    10SFA7F5S7N1gbFd5PvEuDXaA9I2
        chatKey: "-L-I-RPuRa-NCfj1BI
        image: "https://firebasestorage.googleapis.com/v0/I
        name: "test77
        ok: "requeste
        uid: "10SFA7F5S7N1gbFd5PvEuDXa
    Ao8GNLXJKbS7y3rAX7cNYtzjHio2
        chatKey: "-L-HrU3D3msTahuVa
        image: "https://firebasestorage.googleapis.com/v0/I
        name: "test2
        ok: "blocke
        uid: "Ao8GNLXJKbS7y3rAX7cNYtzj
```

Fig. 14: Block

4. Code Documentation

4.1. Sign In / Sign Up

In the 'LoginAcitivity' class, we have six main functions. The first one is *onCreate()*, which mainly checks the activity of buttons. The functions of *signIn()* and *onActivityResult()* are realizing Google's log in. The *firebaseAuthWithGoogle()* can get an ID token from the *GoogleSignInAccount* object and exchange it for a Firebase credential and authenticate it with Firebase using Firebase's credentials after a user successfully signs in. The *checkLogin()* can check whether the user enters the complete information. The function *checkUserExist()* is used to check if a user's information exists in the database; if a user exists and has set up a profile, the page will be switched to the main page, or alternatively the set up page if a user has never set up a profile.

In the 'RegisterActivity' class, we have three main functions. The first one is *onCreate()*, which mainly checks the activity of buttons. The *onActivityResult()* function is used to obtain and adjust the image from the gallery, while the *startSetupAccount()* can send the updated profile image and display name to Firebase. When a user sets up a profile successfully, the page will be switched to the main page.

In the ‘RegisterActivity’ class, we have two main functions. The first one is *onCreate()*, which mainly checks the activity of buttons and *startRegister()* can obtain the user’s information and send it to Firebase. When registering the account successfully, the page will be switched to set up page if a user never set up a profile or a main page.

4.2. Post Recommendation

4.2.1. Post Contents

The post recommendation function is implemented by class ‘PostActivity’ that extends ‘AppCompatActivity’ with three functions. Firstly, the *onCreate()* function connects with the layout file *activity_post.xml* and obtains contents from widgets such as *EditText()* and *RatingBar()*. In obtaining the image from a local image gallery, it sets the request code for the *startActivityForResult()* method inside the widget’s Image Button in *OnClickListener* as a GALLERY_REQUEST. By overriding the protect method *startActivityForResult()*, when it receives both the request codes GALLERY_REQUEST and RESULT_OK, it will generate the URL for the picture and set it to the Image Button.

Once the ‘Submit Post’ button is clicked, the button’s *OnClickListener* in *onCreate()* function will start the *startPosting()* function, which is the second function in the ‘PostActivity’ class. The *startPosting()* generates a variable of ‘StorageReferences’ to upload the image into the ‘Post_Images’ folder in ‘Storage’. It then generates another variable of the ‘DatabaseReference’, using the *push()* method to create a unique random post ID for it and set the values below their corresponding keys. Most values obtained from widgets can be transferred as string-type, except rating the content is a float-type and the image is the URL in ‘Storage’.

4.2.2. Homepage

The main function for the homepage is displaying and updating all the recommendation posts, and it is also the entrance to other pages. For each individual recommendation post in the homepage, its layout is written in the layout file *post_row.xml*. Inside the ‘Recommendation’ class, all of its private attribute names are the same as with the key names in the ‘Recommendation’ branch of the database. It then generates the ‘Getter and Setter’ for each of these attributes.

The homepage is mainly implemented in the ‘MainActivity’ class. For its layout, the *RecyclerView* widget can be used to list all the posts from the database. We generate a class that extends the *RecyclerView.ViewHolder* called the ‘RecommendationViewHolder’, which is used to set the corresponding contents to the widgets. The *onStart()* function creates an adaption of the *FirebaseRecyclerAdapter* which can download the contents from Firebase one by one and send them to the 'ViewHolder'. We also set a S*etOnRatingBarChangeListener* for the 'RatingBar' in the 'ViewHolder', so that once the user re-rates the score, the database can update the new score below this piece of recommendation. Finally, using the *setAdapter()* method the 'RecyclerView' in the homepage can list all the posts from Firebase in real-time. Here we use the Picasso library (vision is picasso:2.5.2) to get the image from the URL.

4.3. Chatting

4.3.1. Architecture Design

To build a real-time chat system, the Google Firebase server was used to store user information and chat data. By using the Firebase server, there was no need to use a 'request-response' to the server from the android client. An android application receives the data stored in the Firebase server dynamically. The dynamic data is sorted by user ID and is only shown to the related user. The chatting architecture, therefore, is much simpler than using a socket server.

4.3.2. Chat Design

To build the chat function in this application, the decision was made to show two different lists to the users. The first one is the friends list of the user and the second one is the existing chat list. The user can create a new chat group by selecting a friend from the friends list or join an existing chat group by selecting a chat group in the chat list.

4.3.2.1. Friend Activity

In friend activity, the user can add friends freely by searching for an email address. In the activity, the user can start a new chat, or block his or her own friend. Once activity has started, the 'ShowFriendList' method will be called automatically to show the friend list.

(1) Firebase Server Handling

To show the friend list, a new firebase data field called 'friends' was created. Each friend field will be updated under the key value, which is user ID. By this, the user can receive their own friends list from a server directly.

(2) Menu List

- **Action Friend**
  Show friend list in a recycler view
- **Action Chat**
  Show Chat list in a recycler view

(3) Add Friend to Server Data

To add a friend to the server, the user should type the email in the edit text. The ‘SearchPeople’ method will be called to find a user from the database. Value event listener will respond to every user in the Firebase database. The user information was saved to the array list first. If there is no matching email in the array list, a toast will show to let the user know that there is no user with that written email address. If there is the same email in the array list, the ‘CheckFriend’ method will be called to check whether that written email is already added to the friends list of the user. If not, the ‘ShowSearchUser’ method will be called to show the searched user profile. To show a user image, the glide library is used. Once the user selects the ‘Add to friends list’ button, the ‘ShowFriendList’ method will be called again.

(4) Image Library Comparison

For this project, Picasso and Glide library were used to handle images from the Firebase server. Glide was much faster to show an image. Even though Glide requires more memory, it is better to use the Glide library to handle images from the next project.

(5) Show Friend List Method

This method queries all the friend fields from Firebase as sorted by the user’s own ID. The queried data will be added to the friend field array list through the ‘Friend field’ class. The *swapData* from the friend recycler view adapter will be called to refresh the recycler view item list. By building MVC, Firebase could handle the data very simply. The ‘ShowFriendList’ method repeatedly called by the adding friend function. The method clears the ‘FriendDataArrayList’ at first so as not to show the duplicated list in the recycler view. If a user selects a friend in a list, the application starts a chat activity related to the selected friend.

(6) Show Chat List Method

The main function is similar to the ‘ShowFriendList’ method. The biggest difference is that chat data is stored and passed to the adapter as a list. This method queries all the chat list data from Firebase. Each chat data can have multiple user data since this application provides group chat functions. Each user's information is composed of name and ID, and images are added to a separated list. The 'ChatListData' model is called to store received data from Firebase and this list will be handled in an adapter to show item. This function will be explained in the adapter sector. If a user selects a chat list, the application starts a chat activity related to the selected chat group.

4.3.2.2. Friend Adapter

This handles data received from Firebase to show as a list in the recycler view. The Glide library is used to show user image, and the 'CircleCropTransform' function is applied to show the profile image in a circle form. The 'SwapData' method replaces the existing list with new data and calls the 'NotifyDataSetChanged' in an adapter so that the recycler view item can be refreshed. The 'OnClick' method passes the 'AdapterPosition' to the friend activity so that a proper friend field can be obtained from the array list by the 'AdapterPosition'.

4.3.2.3. Chat List Adapter

The chat list adapter handles the list data since each chat list has a different number of users. By using a switch, the chat list adapter shows a different view in an item. The Glide library is used to get a bitmap from the Firebase image. Once the bitmap image is ready, combined image methods will be called to combine multiple images. This Glide library is used to show the user image, and the 'CircleCropTransform' function is applied to show the profile image in a circle format. When the 'SwapData' method is called from the friend activity, the 'SwapData' method replaces the existing list with new data and calls the 'NotifyDataSetChanged' in the adapter, so that the recycler view item can be refreshed. The 'OnClick' method passes the 'AdapterPosition' to friend activity so that proper chat data can be obtained from the array list by this 'AdapterPosition'.

4.3.2.4. Block Activity

Block Activity shows the current friend list in a recycler view. If a user selects a friend in the list, the 'OnClick' method will be called to update the Firebase data field 'ok' to the set value 'blocked'. When inviting a friend to a chat group or chatting with a friend, the application checks whether a user gets blocked by a friend or not.

4.3.2.5. Block Adapter

Block Adapter is built to handle data in the recycler view in Block Activity. The ‘OnClick’ method passes the proper position to Block Activity so that Block Activity can handle proper friend fields from the data.

4.3.2.6. Chat Activity

When ‘ChatActivity’ is called, the chat key passes from the previous activity. Chat Activity receives related chat data from Firebase with chatKey.

(1) Started from Friends List

Chat Activity will check the user and friend fields have been updated to the related Firebase database. If not, Chat Activity will update related user information to Firebase, so that this chat group can be shown in a chat list. Firebase’s ‘AddChildEventListener’ will receive added data dynamically from the Firebase database and update the recycler view. So, the list will look like a real-time chat view. When the send button is clicked, Chat Activity updates the date and chatting text information to Firebase’s database ‘LastMessage’. So, the user can check each last message from the chat list.

(2) Started from Chat List

Firebase’s ‘AddChildEventListener’ will receive added data dynamically from the Firebase database and update the recycler view. Chat groups in the chat list already have related user information. Chat Activity will not update the user information to Firebase.

(3) Menu List

- **Action ADD**
  Start ‘ChatInviteActivity’
- **Action Exit**
  Call exit method

(4) Exit

The user can exit the current chat group. If the rest of the number of users is lower than two, Chat Activity will remove the related chat data from Firebase. If not, the exit message will be updated to data, so that the rest of the users in the chat group can recognise the user’s exit.

4.3.2.7. Chat Invite Activity

Chat Invite Activity will show a current friend to be invited in the recycler view. When a user selects a friend in a list, Chat Invite Activity queries the friend field from Firebase to check whether the user gets blocked by that friend. If a user is not blocked, the 'IinviteToChat' method will be called.

(1) 'InviteToChat' Method

This Method will check whether an invited friend is already in a current chat user list. If not, the invited friend field will be added to an array list, and the 'GetExistingUsers' method will be called.

(2) 'GetExistingUsers' Method

If there is a new member, it is designed to open a new chat group with a new member. Therefore, this method will obtain all the user information that is joining the current chat group and update it to a new chat group. When this method finishes, activity will finish with the resulting data including a new chatKey, so that Chat Activity will load new chat data from Firebase for a new chat group.

5. Discussions on Cloud Database

Cloud databases have many advantages. Firstly, they have a better price-performance ratio. Costs can be trimmed and spending on hardware, software licensing, personnel and annual maintenance can be cut. If your database is not significant, you can even use the cloud database reef charge. For example, we use Firebase in data management. Certain services are provided to us without charge up to the fee threshold. Secondly, the cloud database is more efficient. You can use any computer or mobile device to access the databases, reducing the use of resources overall. Thirdly, the cloud database has been configured for automated backups and maintenance. Users do not need to maintain databases, reducing the costs of maintenance.

However, there are still some disadvantages. Firstly, our data is accessed on the Internet, which may be leaked. Therefore, data security is a big challenge. In addition, although the cloud database can satisfy most normal demands, when we explore our own applications, the cloud database cannot provide customized services.

6. Team Information

6.1. Responsibility

NAME	WORKLOAD
Wanjin Li	Post / Homepage
Taoran Ju	Login / Create Account
Hyun Jun Choi	Chatting