Team Name: Mooncake 9.0

Project Name: BU Global Music Festival App

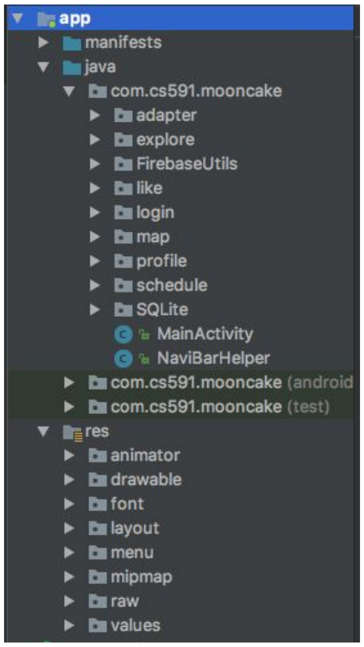
Team Members: Qifan He, Mingyang Yan (Mindy), Lin Li, Zhuoshu Yang (Allen), Yuchen Zhang (Dennis)

Technical Report

Introduction:

Our team worked with BU Art Initiative, which is the institution holding the annual BU Global Music Festival in October 5th and 6th. There will be famous musicians from all around the world performing extraordinary music. Our app makes convenience for people who want to explore, schedule and enjoy the Music Festival. Users don't need to worry about the tedious process of signup and login. By logging in with third-party account (Google, Facebook, and Twitter) or even logging in as anonymous user, people can get all information they need for the festival, which was proudly delivered by Firebase(real-time) and SQlite(local) database. With our fancy UI, users can easily track various events in the festival, get to know those awesome artists in the Explore page, and save their favorite performances in the Liked page. In our Schedule Page, when a simple "add event" button is clicked, Google and Android calendar on user's device will be synchronized as well. Furthermore, the built-in google map and user-friendly indoor maps navigate people to the event location precisely.

Overall Design



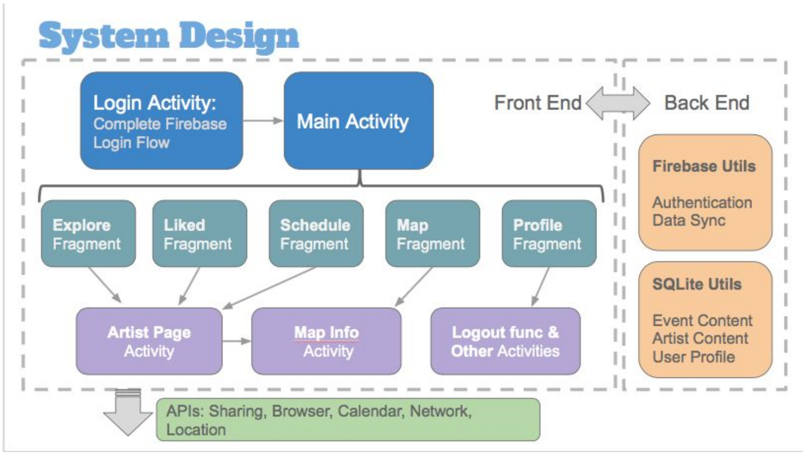
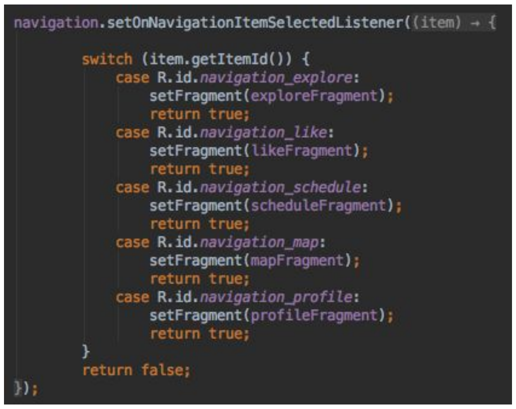


Fig 1. Structure of the App Fig 2. System Design

Description: The main structure of our app is shown in Fig 1. Explore, Liked, Schedule, Map, and Profile pages are implemented as fragments. By utilizing BottomNavigationView, fragments are changed using switch cases, see Fig 3.



```
navigation.setOnNavigationItemSelectedListener((item) → {

        switch (item.getItemId()) {
            case R.id.navigation_explore:
                setFragment(exploreFragment);
                return true;
            case R.id.navigation_like:
                setFragment(likeFragment);
                return true;
            case R.id.navigation_schedule:
                setFragment(scheduleFragment);
                return true;
            case R.id.navigation_map:
                setFragment(mapFragment);
                return true;
            case R.id.navigation_profile:
                setFragment(profileFragment);
                return true;
        }
        return false;
});
```

Fig 3. Set Fragments

Sections below describe in detail how each page and each function in it is implemented.

Database

Firebase Models

FirebaseInitialize.java

Handle Firebase connection initialization; Set up authentication state listener.

FirebaseProfile.java

Create profile on Firebase; Fetch current user's profile from Firebase; Set up real-time database connection and data changed listener; Provide interface to notify MainActivity on data changed.

SQLite Models

DataBaseUtil.java

Copy pre-built database to application's database folder at the first time a user launches the app.

MySQLiteHelper.java

A helper class to handle read/write queries on SQLite database.

SingleArtist.java | SingleEvent.java | SingleUser.java

A human-readable representation of content containers. Used as tokens among front-end, MySQLiteHelper and Firebase models.

Login

Firebase Login Flow

Fig l1. Standard Firebase Authentication Flow

The login flow is completed by LoginActivity.java. A credential listener will be set up when one of four login methods is selected. Once the onSuccess() callback function is called, the app will use the credential complete the authentication process and start MainActivity. At meanwhile, a authentication state listener is set up and put in the onStart() function in the app’s launch activity.

Explore:

Explore session includes a fragment to display artist and event previews and an artist information activity to show details.

Fragment Views Hierarchy:

Fig. e1

The layout of explore fragment contains a nested RecyclerView: A vertical main RecyclerView is the top layer. Each holder of the main RecyclerView contains either a vertical RecyclerView or a horizontal RecyclerView, depends on which type of view the user selected.

MainAdaptor.java

Generate the main RecyclerView and populate each holder with horizontal RecyclerView by default.

HorizontalAdaptor.java

Generate the nested horizontal RecyclerView and populate each holder with a single card view.

VerticalAdaptor.java

Generate the nested vertical RecyclerView and populate each holder with a single card view.

Artist Page Activity Views Hierarchy:

The four buttons in each detailed page allows user to add to schedule, like, share, and show location in map.

Fig. e2

ArtistActivity.java

Populate the page dynamically:

// Generate a view using a layout by code

```
View view1 = LayoutInflator.from(context).inflate(R.layout.view1, null);

View view2 = LayoutInflator.from(context).inflate(R.layout.view2, null);
```

// Add views to a linear-layout in any order

```
linearLayout.addView(view1);
linearLayout.addView(view2);
```

<u>Liked Page:</u>

Like java resources: LikeAdapter.java (Adapter for liked page), LikeFragment.java (Main functions compiled here) Like xml resources: fragment_like.xml (layout for liked page), rv_like_items.xml (layout for cardview in liked page)

Description: Below are the general layouts of liked page, see Fig l1&l2&l3. Fig l4 shows how to add contents to each itemView. Fig l5 shows how to set adapter for Liked Page. Details are explained below in the images. The Layout is embedded with one RecyclerView and there is a list of CardViews in the RecyclerView. There is a TextView in the FrameLayout that shows a message when the there is no liked/saved event in the Liked Page. In each CardView, we get the event's name and image from database using helper function getName() and getPic() (Fig l4). When the user want to unlike an event by clicking on the heart on the lower right corner of each CardView, the dialog box is created using AlertDialog.Builder (Fig. l6). The functionality of "Yes" and "No" buttons are set using setPositiveButton & setNegativeButton. When clicked on Yes, the corresponding event will be removed from the Liked Page. In the meantime, function likeChangedHandler() in the LikeFragment.java notifies MainActivity.java through an Interface, and then MainActivity.java handles the data updates.

Fig l1. Layout for Like page

Fig l2. Layout for CardView in RecyclerView **Fig l3. Liked Page**

Fig l4. Set Content for Each CardView in LikeAdapter

Fig l5. Set Adapter for Liked Page in LikeFragment.java

Fig l6. Set Alert Dialog Box

<u>Schedule Page:</u>

Schedule java resources: scheduleAdapter.java (Adapters for schedule page), ScheduleFragment.java (Main functions compiled here)

Schedule xml resources: fragment_schedule.xml (layout for schedule page), rv_schedule_items.xml (layout for cardview in schedule page)

General Description of Schedule Page

General Layout:

Fig s1. AllSchedule Page .

Fig s2. MySchedule Page

Fig s3. Layout for Schedule page

Fig s4. Layout for CardView in RecyclerView

Fig s5. ScheduleLayoutSetup1

Fig s6. ScheduleLayoutSetup2

Fig s7. Set Adapter for AllSchedule Page

Fig s8. Set Adapter for MySchedule Page

Description: Here are the general layouts of schedule page. Fig s1 is AllSchedule Layout. Fig s2 is MySchedule Layout. Fig s3 and Fig s4 are the xml source code of Schedule page. Fig s5 and Fig s6 are the source code that we use to set value to Objects in xml. Fig s7 and Fig s8 are to set adapters for AllSchedule page and MySchedule page. Details are explained below the images. On the upper right hand corner, there is a button to switch between AllSchedule Page and MySchedule Page. The Layout is embedded with two RecyclerViews which is orientated by Date of the events. In Fig 7s and 8s, when we set adapters, we use singleEvent.getDate() with a switch case to determine which events go to specific RecyclerView. In each CardView, we will get the event's information from database, shown in Fig s5 and s6, and codes are commented. The plus sign and checked sign shows the status of events, whether it has been added to MySchedule or not. All information are extracted and added to database. The user can also add events to schedule in EventActivity and ArtistActivity. They will sync with Calendar and Schedule Page because our database will be updated by calling singleUser.addScheduled() with unique eventID. Everything shows in Schedule Page is extracted from our database by using singleUser.getScheduled().

Further Implementation in Schedule Page

Fig s9. Add events to Calendars

Fig s10. Delete events from Calendars

Fig s11. Add event to Calendar source code

Fig s12. Delete events from Calendar source code

Fig s13. Unadd event page changes

Description: With the checked icon in Fig s1 and s2, the event has been added into MySchedule Page, and also by granting the permission to Calendar, calling AddToCalendarHandler with a unique eventID in Fig s11 will sync the schedule to Android Calendar and Google Calendar. If the user does not add an event in their own schedule, the icon will appear as plus sign as shown in Fig s1. When user unadded an event, the event will disappear in the MySchedule Page and

Calendars. Also, the image icon will change back to plus sign simultaneously in AllSchedule, EventActivity and ArtistActivity shown in Fig s13.

Map:

General Description of Map Page:

CustomInfoWindowAdapter.java (Custom Info Window)

MapActivity(Explore button to map)

MapFragment(Map Functionality)

ShowInfo(Indoor Map Activity)

Fig1 **Fig2** **Fig3**

As you can see in the Fig1, we have the button on the top right corner of the screen for current location. When you click it, the map will move and there will be a blue point showing the location of where you at. Because the festival has only two place for performance, we only put two marker on the map, which is GSU and CAS building. When you click the GSU marker, there will be a customized info window pop out (Fig2).Also, there will be a toolbar on the bottom right of the map. It is designed for the navigation. The left one will show you the routine and the right one will show you the place in google map.

For implementation, we first have to check if the user has the correct version of Google play service by using the isServiceOK() function. Also, we have the onMapReady() function, which is a call back function and will present the map for the user. The getDeviceLocation() will allow us get the current location of user and getLocationPermission() will ask the user for location permission. Rather than using the gps and location service all the time, which consumes battery badly, we override the activity lifecycle in order to save energy for users when they are not focusing on the map.

Rather than simply putting two button on the map, we care about the user experience and we want to deliver more straightforward and user-friendly solution. We customize the info window so users can click it and it will lead them to the indoor map. There are two buttons for user to select, which is the floor plan of basement and 2nd floor. By using the third party api, the indoor map is actually a photo view, which can be zoom in and zoom out. We put four icons washroom, stair, entrance and elevator in Fig3 so users can easily zoom in the map and find what they want.

Fig4

In the artist page, when you click the location button, it will start an intent and pass it unique ID to map activity. And we will get the unique ID and by searching this ID in SQLite, we get the building info and put a info window to indicate that is place where users should go. We reuse the code we already have for the main map so we don't have to write it twice.

<u>Profile:</u>

AboutPage.java(Call for activity_about_page.xml)

CustomAdapter.java(Adapters for profile page)

ProfileFragment.java(Main functions compiled here)

fragment_profile.xml(layout for profile page)

activity_about_page.xml(layout for about page)

General Description of Schedule Page:

Description: After the user login into our APP, whatever which authentication he/she uses, such as Google, Facebook or Twitter, all his/her information will be stored on the Firebase, so I used a function called DowloadImage, to input the url of user’s portrait and output a Bitmap of user’s photo to store into the SQLite database. Then when user open the profile page, the functions to set users' portrait and name will be activated, which can get users portrait and name on social media. In addition, there are also five main functions in profile part, which are "website, invite friend, feedback, about and ticket", to let users visit BU Art Initiative website, invite friend to download our app, to do a survey about his/her experiment on each event, check the information about music festival,and to buy ticket online. Finally, there is also a Logout Button, that user can click to logout and jump back to login page to start over.

Fig p1. overall design

General Layout:

Fig p2. Guest Page **Fig p3. User Page**

Fig p4. Overall Profile XML

Code for profile:

Fig p5 Overall Profile JAVA

Fig p6 Overall Profile JAVA

Code for AboutPage:

Fig p7 Aboutpage Code

Our profile page is using ListView, so when the user click on About, it will trigger the case 3, and this code will lead the user to AboutPage.class in below.

Fig p8 Aboutpage XML

Description: The DowloadImage function is used to input the url of user's portrait and output a Bitmap of user's photo that I can store into SQLite. By calling this Function, I also used a helper function called getHttpConnection. This function takes the url of the photo and get HTTP connection first, then transfer into a stream, finally send the stream back to DowloadImage and return a Bitmap.

Fig p9 Downloadimage code

Fig p10 getHTTP Code

To set user's portrait and name. I stored the Bitmap into our SQLite by using the command that Qifan designed "myDb.addProfile(singleUser)", and then I use "userPic.setImageBitmap" and "username.setText" in profile fragment to display user's photo and name.

Fig p11 add to database JAVA

Fig p12 Set Portrait and name JAVA

In addition, there are also condition check, such as, if user login with Guest option, the UI will display the default portrait and name. Then I'm going to introduce the five main functions in profile.

Five main Functions:

since Lin used a ListView to set each category, so I used a switch to handle different cases.

Fig p13 Five Main functions JAVA

Fig p14 Five Main functions Code

Finally, there is also a Logout Button, that user can click to logout and jump back to login page to start everything again.

Fig p15 Logout Activity

Other Works done:

Advertisement

Description: There is advertisement banner on the bottom of main page in Fig a1, so I write the function in MainActivity. Since the product owner wants to promote BU colleagues and the funders instead of commercial use, I just use a Handler to make an adView with clickable imageview in Fig a2. When user clicks on that, it will lead them to website of specific banner. The ads will be refresh every 30 seconds by the commend handler.postDelayed(this, 30000). If user finds it is disturbing, they can close the ads when they click on close button. The source code can be found on bottom of Fig a2.

Fig a1. Ads in App

Fig a2. Ads source code

Animations

Description: We have implemented animation in Event, Schedule and Profile pages. We use a fade in animation when returning to previous page in Fig anim1 and Fig anim2. For Schedule and Profile pages, we use the same animations, hightlighting the CardView that user selected on and sliding in from left shown in Fig anim3, Fig anim4 and Fig anim5.

Fig anim1. Animation in Event Page

Fig anim3. Animation in Schedule and Profile Page

Fig anim2. Animation show in Event Page

Fig anim4. Animation show in Schedule and Profile (Same Animation)

Fig anim5. Highlighting Animation in xml

Asking Permission

Description: The permission will be explained in Fig permission1 and asked in Fig permisison2 when the app first installed in user's phone. The user can choose Allow or Deny. If user choose

Allow, it will simply grant permission in Fig permission5. Otherwise, they denied. Later, when they enter in the application, the permission will be asked again in Fig permission3. This time, the app will allow user to deny it with not asking again. If they choose denying without asking again, they can turn on the permission in their phones' settings. They can choose Deny again and the same process will be applied, simply just choosing Allow.

Fig permission1. Explanation

Fig permission2. Ask permission for first time

Fig permission3. Ask after first time deny

Fig permission4. Ask permission

Fig permission5. Add Permission and Granted Permission

Share Functionality

Description: This is implemented in the ShareUtil.java which will used by both event share and inviting friends. It sends text/url intent to only apps shown on Fig Share by checking the package names of those apps.There will be an eventID passed to ShareUtil.java from the explore page so that it can share the message with the corresponding name. As for inviting friends, the message with the url of our app in the google play store will be sent.