CS 481: Mobile Programing

Final Project: Alarm Clock App

Group 3

Prepared By:

Brian Ramirez, Rachel Arellano, Eric Lozano, Eduardo Terrazas

Code Located on GitHub [cs481csusm/finalproject-group_3]

Purpose

The purpose of this report is to provide an overview of what has been accomplished towards the completion of our project and what if any roadblocks have been encountered.

Application Overview

When the application opens it will prompt the user to login or create an account. After logging into the application the user will be taken to the main navigation page. From the main navigation page users will be able to navigate between several different fragments. These fragments will host different functionalities such as a timer, stopwatch, pomodoro timer, list view and a clock.

Feature Goals

- Navigation
- Fragments
- Firebase
- Room
- Unit Testing
- ViewModel and Live Data
- RecyclerView
- Permission
- Material Design
- Notification

Current Progress

- Completed integration of the Firebase component via login and signup functionalities
- Completed integration of the Navigation component, allowing users to toggle between several fragments
- In-progress
    - Fragments
    - Room
    - Unit Testing
    - ViewModel and Live Data
    - Recycler View
    - Permission
    - Material Design
    - Notifications

Division of Labor

Eric Lozano	Room, unit testing, Firebase
Eduardo Terrazas	Permission, Clock Fragment, Notifications
Brian Ramirez	Room, Material Design, RecyclerView
Rachel Arellano	Pomodoro Fragment, LiveData View Model

Current Features:

1.1. Navigation

```
private fun botNav()

    val clockFragment = ClockFragment()
    val stopwatchFragment = StopWatchFragment()
    val timerFragment = TimerFragment()
    val pomodoroFragment = PomodoroFragment()

    bottom_nav.setOnItemSelectedListener{
            item->
        when(item.itemId){
            R.id.ic_clock -> {
                changeFragment(clockFragment)
                true
            }
            R.id.ic_stop_watch ->{
                changeFragment(stopwatchFragment)
                true
            }
            R.id.ic_timer ->{
                changeFragment(timerFragment)
                true
            }
            R.id.ic_pomodoro ->{
                changeFragment(pomodoroFragment)
                true
            }
            else -> false
        }
    }


private fun changeFragment(fragment: Fragment)

    supportFragmentManager.beginTransaction().apply {
        replace(R.id.fContainer,fragment)
        commit()
    }
```

Figure 1.1.1

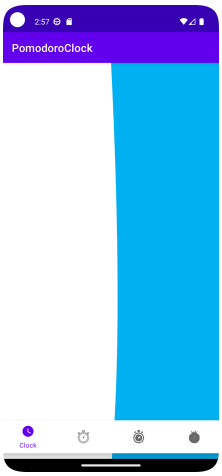

Figure 1.1.2

For navigation we chose to do a bottom nav menu. Figure 1.1.1 shows the kotlin code that allows the program to switch between fragments depending on which icon is pressed on the bottom menu. Figure 1.1.2 shows the in-app design of how the bottom nav will look. We plan to make each icon in the menu its own clock. For example, there is a button for alarms, one for timers, and etc.

1.2. Fragments

Each clock type will be its own fragment that will be displayed in a container view. For example, in figure 1.1.2 if the user were to click the first button for the alarm clock then the app will display a clock so that the user can set a time.

1.3. Firebase

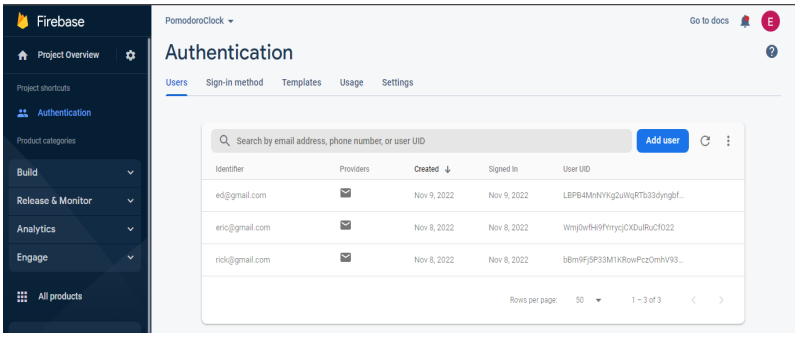

Figure 1.3.1

```
private fun loginUser()
{
    var email = viewBinding.LogEmail.text.toString()
    var password = viewBinding.LogPassword.text.toString()

    if(email.isNotEmpty() && password.isNotEmpty())
    {
        user.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(CreateActivity()){ task->
                if(task.isSuccessful) {
                    Toast.makeText( context: this,  text: "User Signed in Successfully.", Toast.LENGTH_SHORT).show()
                    startActivity(Intent( packageContext: this, HolderActivity::class.java))
                }
                else{
                    Toast.makeText( context: this,task.exception!!.message,Toast.LENGTH_SHORT).show()
                }
            }
    }
    else{
        Toast.makeText( context: this,  text: "Email and Password are incorrect.", Toast.LENGTH_SHORT).show()
    }
}
```

Figure 1.3.2

Figure1.3.1 shows the emails that we have registered so far in our Firebase project through the built-in authentication. As for the code, we use viewBinding so that we do not have to use findViewById calls. Then we check if both of the editText fields are not empty, if not empty then we create a user object with the given email and password. Else, it will just throw an error.

1.4. Room(local)

Room database is in the process of being developed. It will store specific times like alarm times of the user. It will hold the users based on the fire authentication code given by firebase.

1.5. Unit Testing

Unit testing will be used with a room database much like the in class example. To see if we are adding or removing correctly into the database.

1.6. ViewModel and LiveData

Viewmodel Live data will be implemented by adding a counter to see how many current users are creating accounts for the app. As well as view model live data will be used on the pomodoro if the user presses the start pomodoro button it will count up just so we know how many users are actually using that specific clock feature.

1.7. Recycler View

Recyclerview will read from the room database and display the alarm times users have saved in their accounts. The pomodoro break times/active times and cool down times. Eventually the choosing of alarm sounds will be implemented so storing like 5 variants of alarm names will be used.

1.8. Permission

Permissions will be asking the user if we can get their general location so the clock can be set to their time zone automatically. Permission to vibrate when the alarm goes off.

1.9. Material Design

Material design as you can see with the background in our mock ups we have already started to implement backgrounds and buttons. Towards the design aspect of our layouts. Will try to add clock design in the fragments and count down text boxes to make it seem more sleek and professional. Floating action buttons will be added for signout feature purposes.

1.10. Notifications

Notifications will be used when the pomodoro clock is about to finish its countdown to push the users towards those final seconds. Notification will be used the alarm time is reached for testing purposes.