CS 481: Final Project
Clock App

Implemented by:
Rachel Arellano, Eric Lozano, Brian Ramirez, Eduardo Terrazas

Project Code available on GitHub:
cs481/ finalproject-group_3

Abstract

This paper documents the process of implementing our final project the “Clock App”. At this time the final deliverable of this project is now due, and the term has come to completion. Over the course of the semester, we had very limited knowledge and time to dedicate to the project as we would have liked. However, we believe that our final deliverable is acceptable and that it shows that ultimately, we were able to create an application that shows potential.

Introduction

Clock App is the result of our final project for the course CS 481: Intro to mobile programming. The purpose of this project was to get hands on experience creating a product from scratch with teamwork in mind. This project was intended to help us broaden our experience working with others on a specific task while building out personal portfolios.

Our team decided to re-envision a product that we were all familiar with and that would provide just enough challenge for growth. Thus, we chose to create an alarm clock application called “Clock App”. This application contains all the same features of a traditional alarm clock app such as a timer, stopwatch, and alarm clock. Also, the extended feature of a pomodoro timer. Pomodoro is the technique of increasing productivity by splitting work into intervals separated by short breaks.

For this project we were also allowed to decide which ten features would be implemented into our final product. The features that we chose were the following: Navigation, Firebase, Room, View Model, Live Data, Recycler View, Permissions, Material Design, Notification, Fragments, and Unit Testing. We chose these features based on the likelihood that we would be able to successfully implement them.

Finally, since this was a group assignment, we wanted to divide the tasks among ourselves as evenly as possible. So, we initially made commitments that we believed we could eventually deliver on, however, time being what it is and our experience levels not being even across the board we ultimately may not have been able to keep to these arrangements. Ultimately, we tried our best to create a successful environment and deliver on our application goals but being students still learning and growing ourselves the application still has some rough edges.

Application Description

“Clock App” is an application that provides users with a reliable way of setting reminders along with the extended feature of a pomodoro timer. When the user enters the application, they will be prompted to login or create an account. After successfully logging in to the app the user will be directed to the main activity that hosts all the application fragments. Such as clock, stopwatch, timer, pomodoro and alarm list. Clock allows users to schedule an alarm for a future time. Stopwatch allows users to keep track of the precise time that has elapsed. Timer allows users to set a countdown timer for a specified amount of time. Pomodoro allows users to indicate the number of study and break rounds and their durations. This application uses the following features:

Navigation

Fig. 1

Navigation for this application was implemented using a bottom navigation menu. Users can navigate between timers by selecting the appropriate icon from the bottom navigation menu. The benefit of using a bottom navigation menu for this application is that it allows users to easily navigate and distinguish fragments from one another. This is especially important in an alarm clock app such as ours because of the similarity between all the fragments. This application has five menu icons that represent the corresponding functionalities: clock, stopwatch, timer, pomodoro, and alarm list. The current iteration of this feature is displayed in Figure 1 above.

Fragments

Fragments are used to provide all the application functionalities. They are hosted by a single activity and provide a way of modularity and reusability for an applications user interface. Although our use of fragments was not quite as efficient as they can be it did allow us to separate the bottom navigation menu from the content of the application.

Firebase

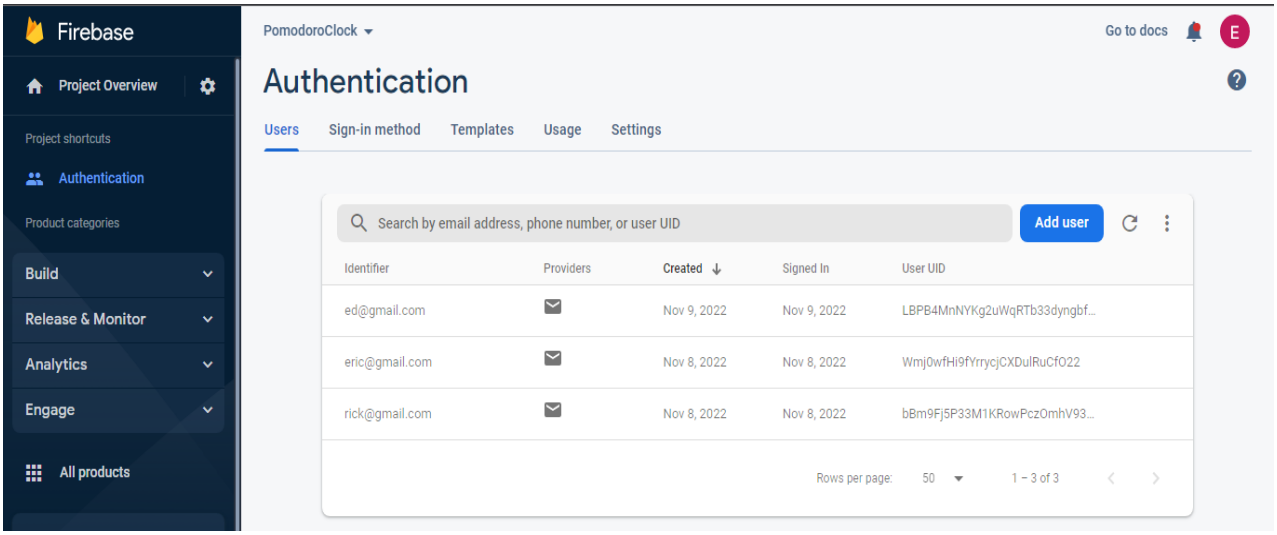


Fig. 2

This application utilizes Firebase Authentication to track users that have signed up for an account. Firebase provides a convenient way of organizing users information as well as some security. Since implementing a user account was not the main aim of the application, we wanted to rely on firebase to facilitate a quick development process while providing useful libraries and easily integrated SDK's. Figure 2 above shows a screen shot of the sample accounts that are stored into firebase after having created through our application.

Room (local)

A Room local database was implemented to store alarms generated by the user. We used Room to create a relational database that is stored locally on the user's device. This allows the data to

persist on the device meaning that it can be accessed even when the device is offline. Having persisting data for this application makes the most sense since alarms still need to reliably go off even when there is no internet.

Unit Testing

Unit testing is a way of testing small chunks of code to ensure that they are performing as expected. Learning to properly perform unit tests is an important part of developing an application. This is because it can help to prevent unintended errors from becoming apart of the final product. Testing should be applied throughout an application and frequently so that the end user has a more reliable product and money is not wasted debugging a deployed application.

View Model & Live Data

View Model & Live data was implemented hand and hand with room database. For example, view model will be able to access the needed information through live data. Live data will then access the created repository with the database information. The repository can insert, delete and read from the database. Essentially the repository is a buffer between the view model and Database keeping data separate from view model. This way we will not accidentally mess with information in the database. Repository is able to do this because we created an alarm dao object which cotains all our queries. Live data just a way view model can detect any changes that have occurred in a database in other words an observable object.

Recycler View

A recycler view will be used to display all the users' alarms saved to the Room database. The recycler view will make displaying user data more efficient since it only creates elements as they are needed. It also helps to reduce the overall power consumption and improves performance by not generating more than it needs at any one time. It also provides a pleasant user interface when searching for previously stored alarms. Recycler view was implemented by creating an Alarms view model object which would be in charge of retrieving the information from the database by using the live data implemented as an observable. Finally, we would just set the alarms by calling our setAlarms function. This would set the alarms on thorugh the recycler view if toggled on.

Permission

Application permissions support a users' privacy by protecting access to restricted data and restricted actions. In relation to our application, we had planned to use permissions to access a users' location to set their time zone automatically. We also wanted to ask the user for permission to vibrate when an alarm rings. Schedule exact alarm was added as a permission in the manifest as it is needed to set schedule an alarm using Alarm manger. Receive boot on completion was used for broadcast receivers since we wanted to execute the alarm when a certain time was reached so we would create an intent to send the required context and info. Th broadcast reciver would then call alarm service which contains the sound for the alarm.

Material Design

Material design is the technique of creating products that provide a friendly user interface and are easily scalable. We wanted to apply this to our application using a cohesive design that performed in obvious ways. Our design approach was to keep the overall look of the app simple but to enhance the user experience by highlighting user actions such as input and buttons. For example, any edit texts in use are highlighted by using a brighter color to indicate its usage. Buttons used throughout the app also indicate usage by transitioning to a brighter color. In the case of the button and the background they were created to be reused easily across the application.

Notifications

Applications use notifications to display a message to the user outside of the user interface. For this application we wanted to use notifications to communicate time sensitive information. 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.

Challenges

One of our first challenges was issues with the fragments, there were times when Android Studio did not like certain lines of code. For example, lines of code that require the activity's context. Room database was also a challenge for us, especially when fetching for some of the data to display for the recycler view. Another challenge was view binding, it was giving us errors so we reverted to just using findViewById calls. Notifications were a challenge to implement because it was just not working in our app.

Limitations

We as students did not yet have the domain knowledge to contribute in a meaningful way until around the course end. This limited the overall time we could dedicate to producing the final deliverable. Especially since we all have different commitments to other courses that require an equal amount of time and effort as this course does. Even so we all put our best foot forward in our attempt to create an application that works in the way that we intended. However, because of our limitations due to time and knowledge our application did not quite meet our desired effectiveness.

There are several features of this application that were not implemented to satisfaction. One such feature was notifications. We had intended to use notifications to update users with time sensitive information regarding their alarm. Such as notifications that a timer has finished its countdown or as for the pomodoro fragment users would be notified that the current interval had been completed. But this feature ultimately was dropped due to the limited time we had to complete it. Another feature that was not implemented was unit testing. In hind site this feature could have been implemented as we wrote the logic behind our timers but due to our unfamiliarity with unit testing and how to use it in an effective way, we again had to drop it. Finally, we attempted to complete the remaining features in a meaningful way however, they may not be as effective as desired we tried to test all the features, but some bugs may have escaped us considering we were working with a limited time frame and resources.

Implemented Features

We were able to implement the majority features such as fragments that contain the following alarm, pomodoro, list of alarm fragments, room database, stop watch, timer watch, firebase authentication, live data, view model, recycler view/adapter, permissions, material design and bottom menu navigation. Firebase was implemented by using activities which would prompt the user to either create and account or log in using an existing account. Firebase would one create the account with the email specified by the user and two it would very the log in by checking if this account exists and if the credentials match. The fragments were implemented by using a bottom navigation. The bottom navigation bar would allow us to rotate through the fragments by clicking on icon on the user wanted to access. The fragments which we used along with the bottom navigation are clock fragment(alarm), timer fragment, stop watch fragment, pomodoro fragment and list of alarms fragment.

Room database was implemented by creating table called Alarms which contained an alarm function which would get passed hours, minutes, days of the week whether it is recurring, title of the event and other variables needed. We would then insert the tableinfo object (insertion occurs in clock fragment) into the database through the use of view model and repository. Live data and view model used to access and retrieve the required information from the created repository which in turn accesses or inserts/updates the database directly. Recycler view was used to display the database alarms which the user has inserted or updated. Material design was implemented through making the time selecting easy to use for the user. For example, we used a time picker to scroll the times for the user when selecting, we have a easy to use buttons the user just clicks a timer starts or stops. We have count downs in the Pomodoro clock where they can stop the clock the click it again and it starts again. They can easily toggle on off alarms in the list of alarms fragment. These were all the features we managed to implement so far.

Conclusion

We started this project with optimistic and ambitious ideas, to implement ten of the concepts we went over in class. However, we did end up running into more challenges than we had expected. Even having to face all these challenges, our group is satisfied with the app that we have made. All of the clocks work and we have implemented the majority of the concepts that we planned in the original proposal, although some changes were made.

Division Of Labor

Team Member	Contributions
Eric Lozano	Room, unit testing, Firebase. Clock fragment, Adapter, Documentation
Eduardo Terrazas	Permission, Stopwatch Fragment, Notifications, Documentation

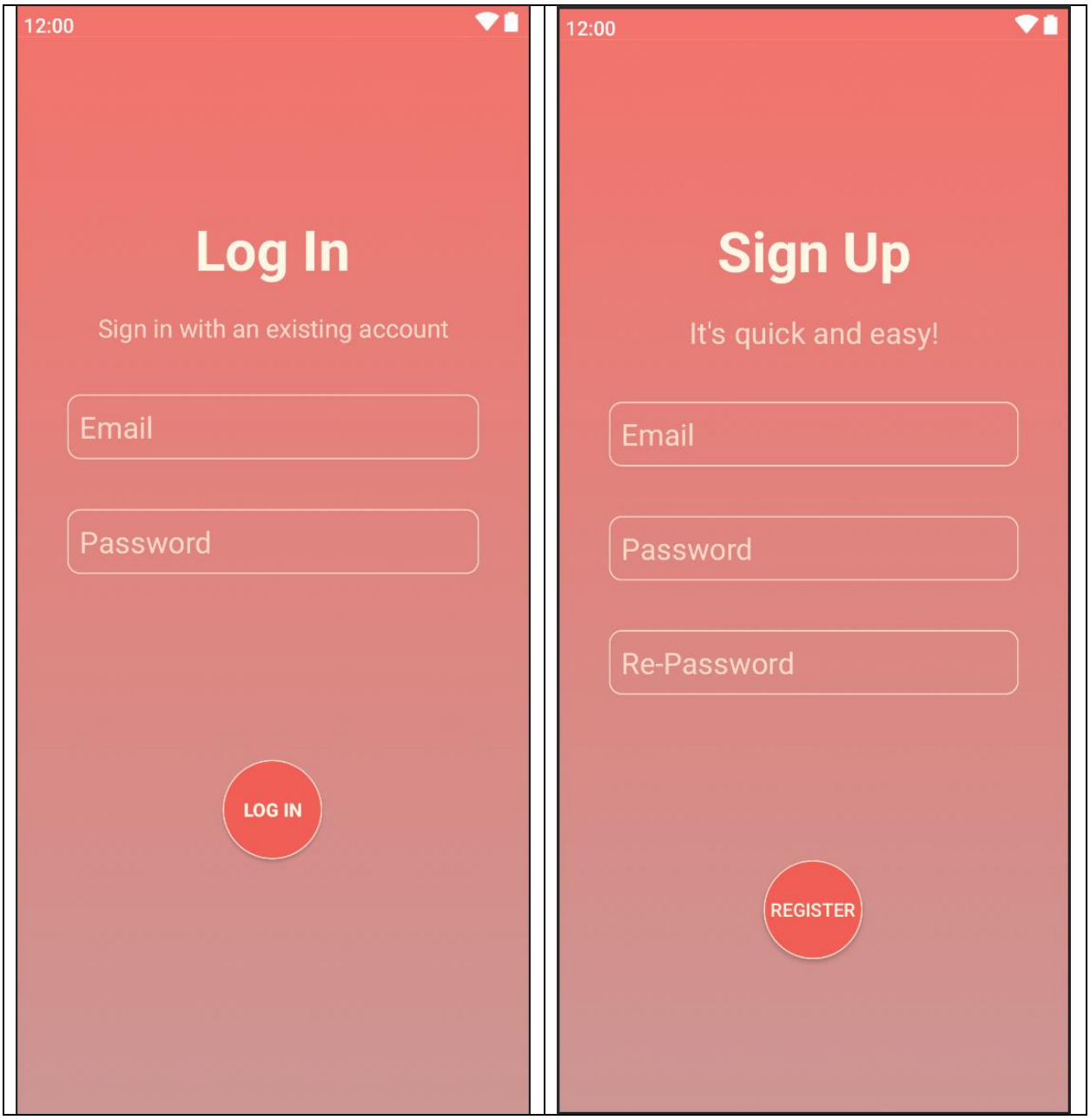
Brian Ramirez	Room, RecyclerView, LiveData ,View Model, Timer Fragment, Documentation
Rachel Arellano	Pomodoro Fragment, Material Design, Documentation

References

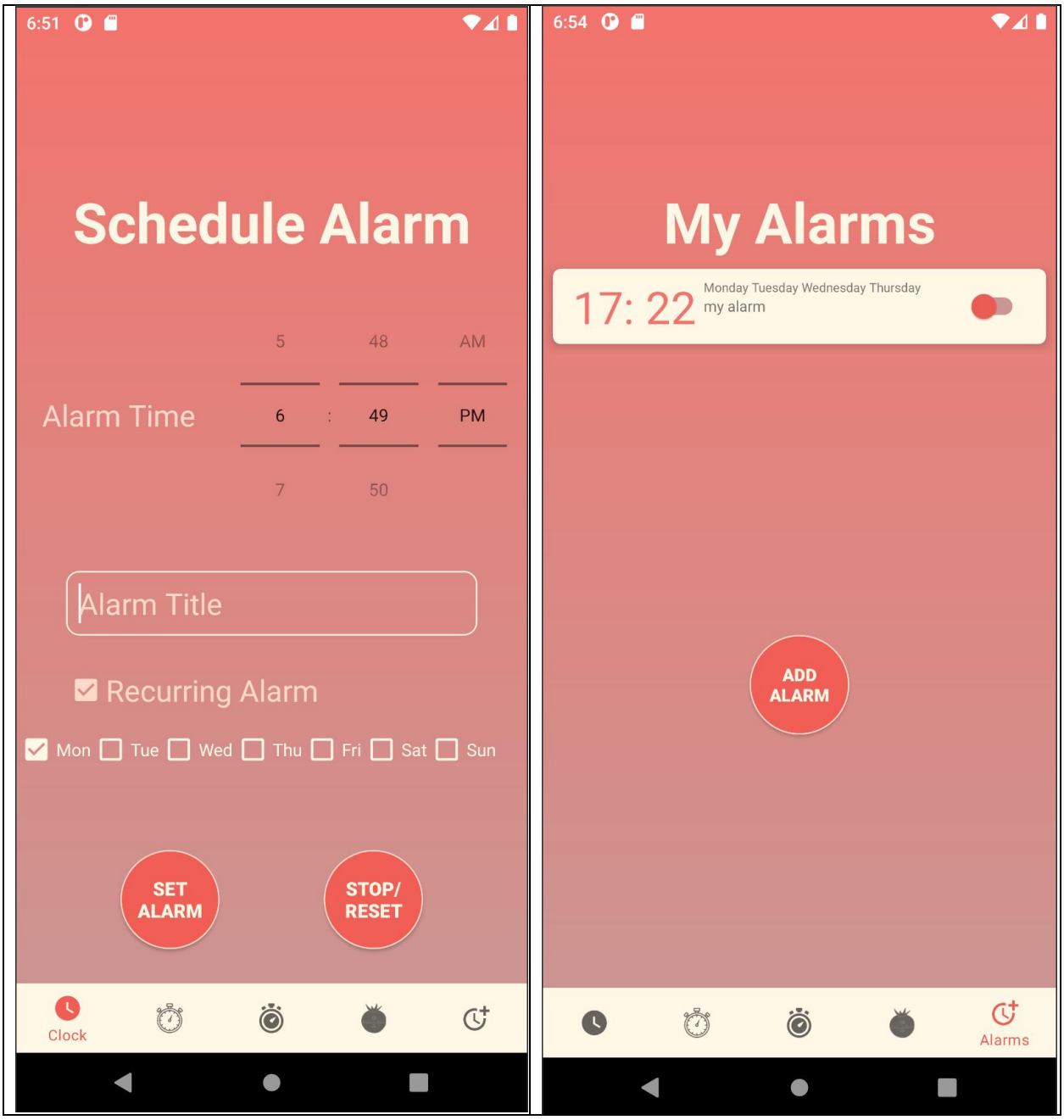
For all things android development related
https://developer.android.com/

Appendix

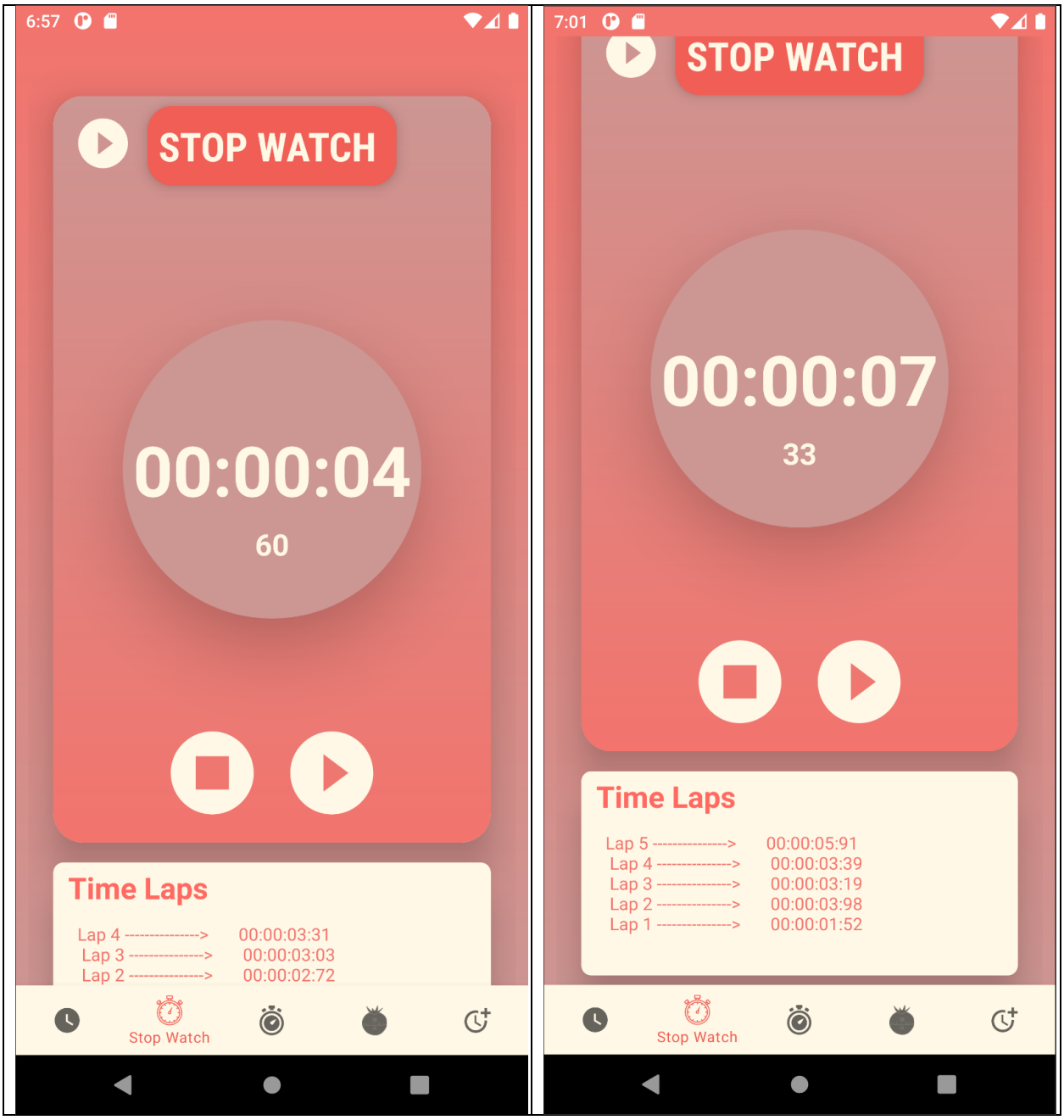
Login & Sign Up pages



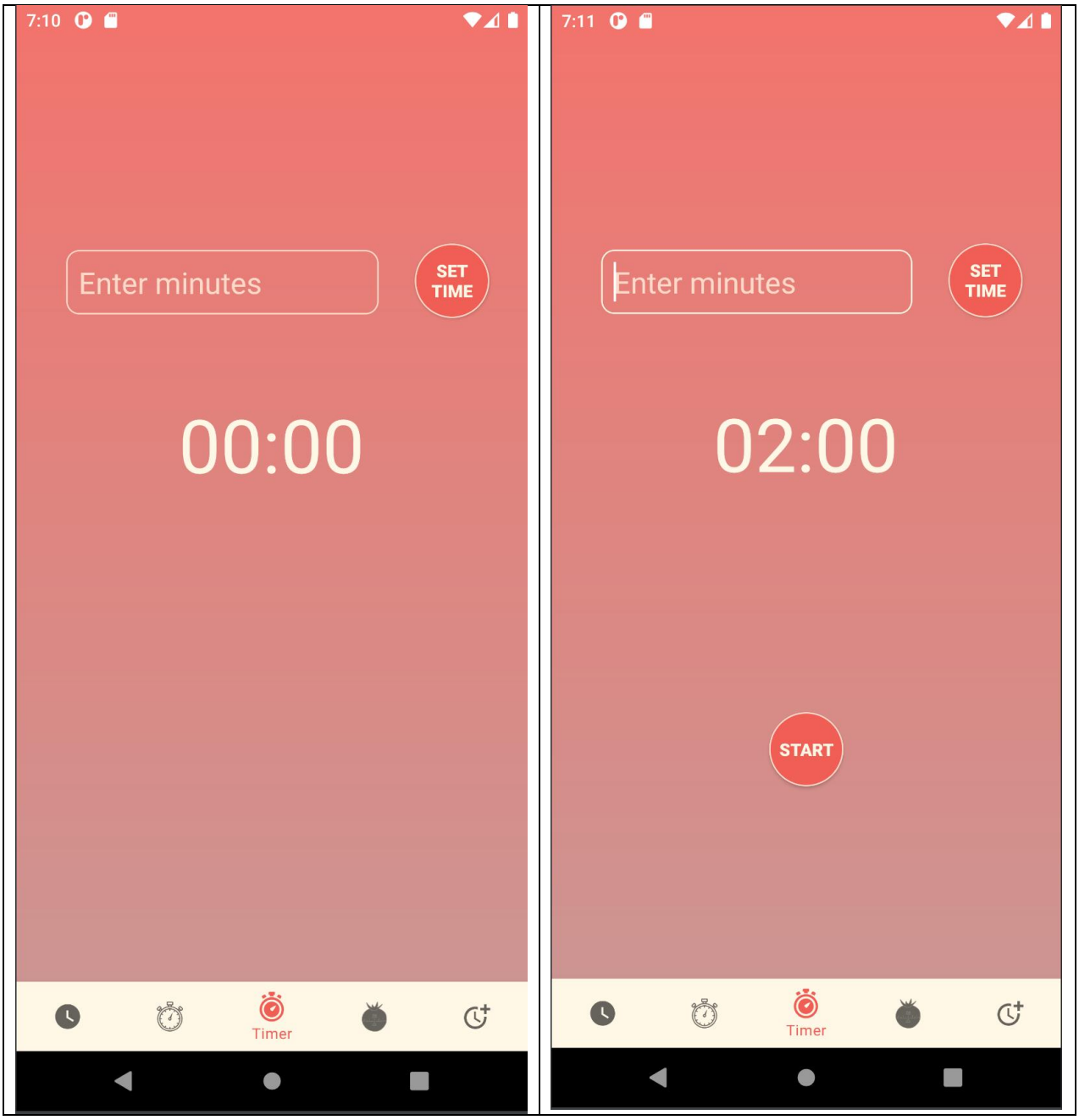
Clock and Alarm List

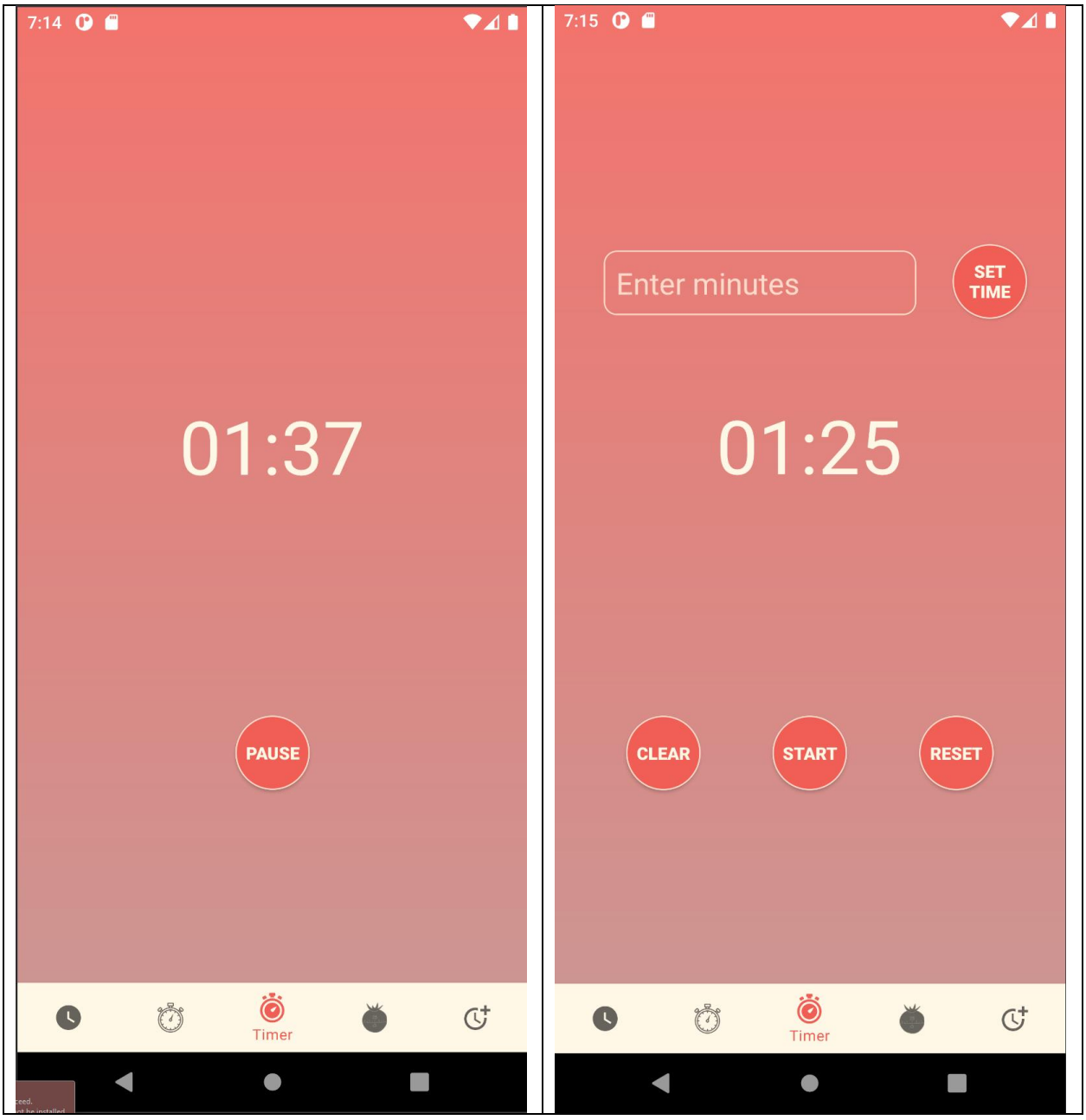

6:51
Schedule Alarm
5
48
AM
Alarm Time
6
:
49
PM
7
50
Alarm Title
Recurring Alarm
Mon
Tue
Wed
Thu
Fri
Sat
Sun
SET ALARM
STOP/ RESET
Clock
6:54
My Alarms
17: 22
Monday Tuesday Wednesday Thursday
my alarm
ADD ALARM
Alarms

Stopwatch

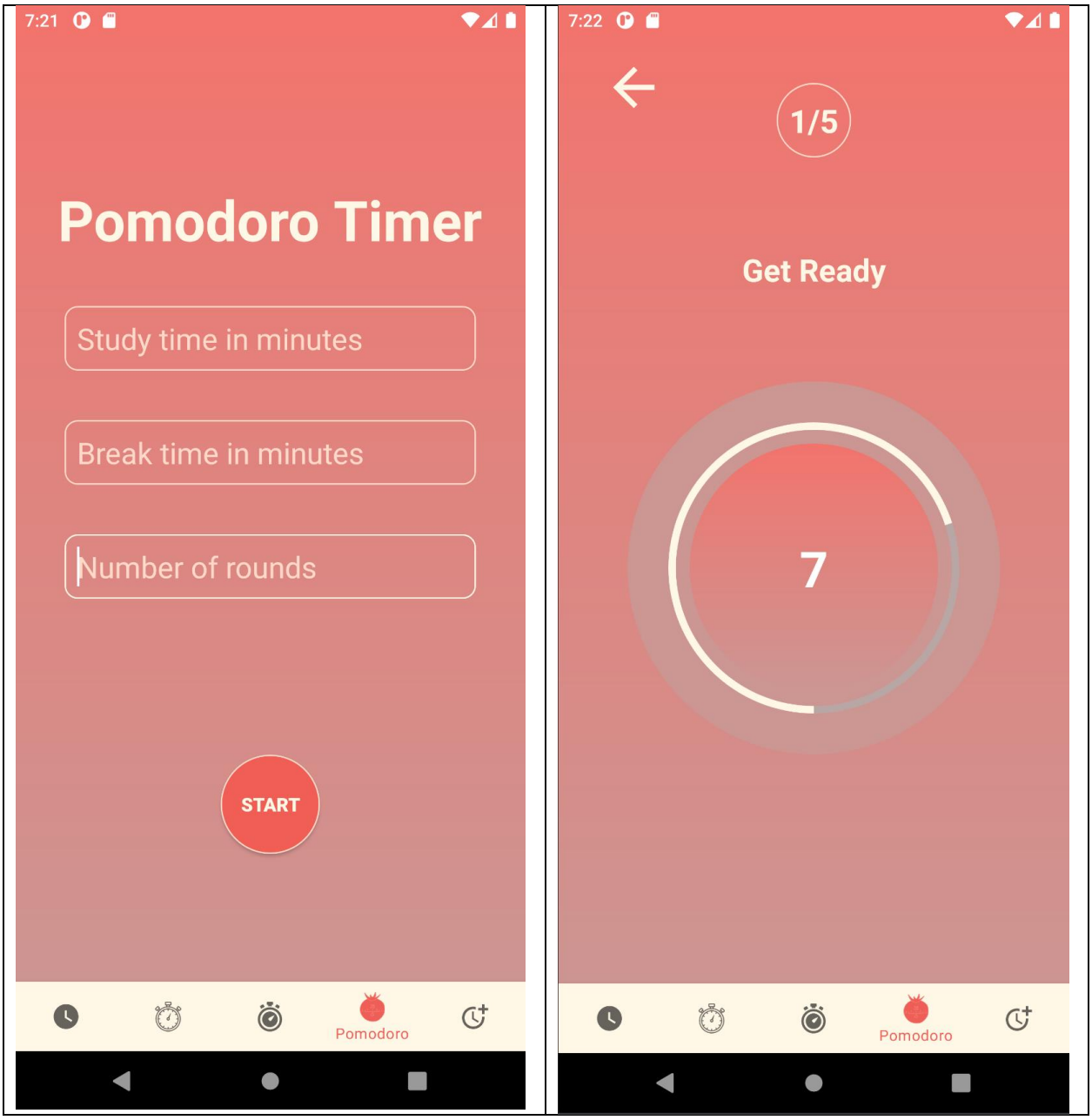

6:57
STOP WATCH
00:00:04
60
Time Laps
Lap 4 ----------> 00:00:03:31
Lap 3 ----------> 00:00:03:03
Lap 2 ----------> 00:00:02:72
Stop Watch
7:01
STOP WATCH
00:00:07
33
Time Laps
Lap 5 ----------> 00:00:05:91
Lap 4 ----------> 00:00:03:39
Lap 3 ----------> 00:00:03:19
Lap 2 ----------> 00:00:03:98
Lap 1 ----------> 00:00:01:52
Stop Watch

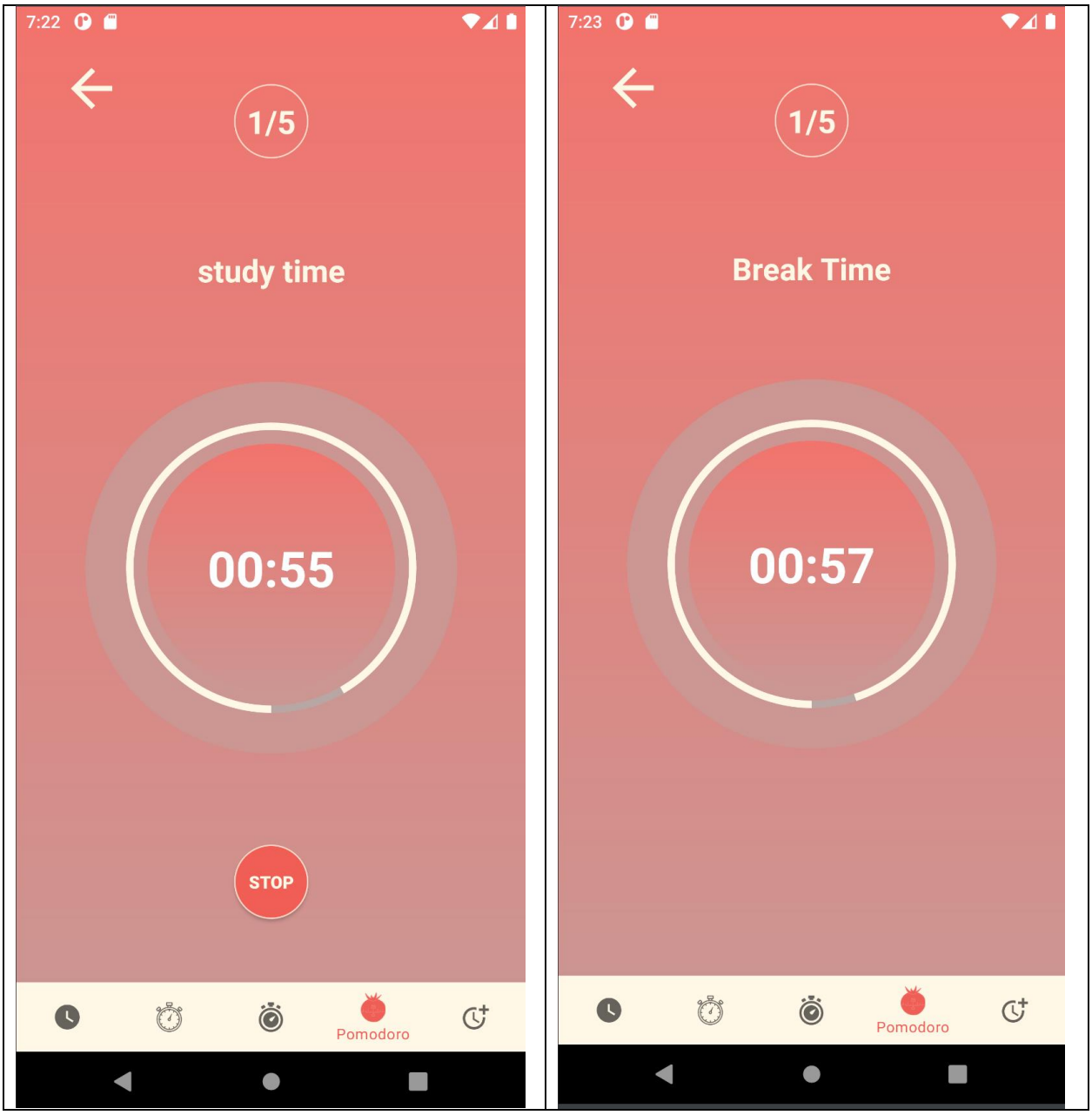
Timer


7:10
Enter minutes
SET TIME
00:00
Timer
7:11
Enter minutes
SET TIME
02:00
START
Timer



Pomodoro Timer


7:21
Pomodoro Timer
Study time in minutes
Break time in minutes
Number of rounds
START
Pomodoro
7:22
1/5
Get Ready
7
Pomodoro


7:22
1/5
study time
00:55
STOP
Pomodoro
7:23
1/5
Break Time
00:57
Pomodoro