A&J's Clockwork

Android Alarm System Report

By Juan Rodriguez, Johnson Dinh, And Adrian Debraga

Humber College
CENG-319-ONC

A&J's ClockWork

1. Abstract

A&J's Clockwork is an android based alarm mobile application where users are able to set and customize alarms of their choice for daily use. The user will be able to create multiple alarm profiles along with being able to set timers and stopwatches. When connected to its corresponding hardware component, the app furthers its capabilities by allowing users the ability to read local temperatures via the sensor included. The app is designed to give users ease of access to anything time related in a simple and clean formfactor.

Students can benefit greatly from this product for those that struggle waking up at certain times of day and require an external source for assistance such as the app. It can also be used to set reminders for certain time periods such as deadlines for assignments etc. The temperature readings are an added bonus as it readily displays the current temperature in real time without having the need to open other respective applications for similar purposes.

In order for the app to function as intended, it is linked up with a database known as Firebase. This allows user information such as their own personal accounts, alarm templates and settings to be stored in a cloud which is ultimately located in Firebase.

2. Introduction

The app consists of many components to accomplish the tasks we set out to implement and create. These range from the functionality of the alarm, clock, and timer, to the inclusion of database readings when dealing with temperature. Database authentication is also a crucial step as this allows users to save their desired settings under their own ID which would tie to the database attached to the app. Once everything is implemented and connected to one another, development of the app can move onto different aspects not yet incorporated such as UI improvements and hardware involvement.

3. Design Specification

The way we would develop the app is by using the Android studio platform and create separate files where our code would be placed. Our focus at the start of the project was to create screen layouts for each individual section so that we can picture where certain data would be displayed and formatted. We designed the user interface to be clean and simple so that it would feel inviting for the user rather than being complex and intimidating. Having the 3 main features of

the app (Clock, Alarm, Timer) on the homepage via tab fragments would make it quick and easy for the user to traverse without having to flip through various menus and options.
Once the layouts were created, functionality for each part was able to commence. The data that the app uses such as different time zones, alarms, and stopwatch count would be provided by user input so that they could choose whatever they desire. In terms of the temperature readings, we were not able to provide its actual functionality in its current state due to not developing the integration yet. In the meantime, we planned to show a demo of how stock temperature values would be stored and communicated to the database. The way we would manage the MVC architecture is by focusing on the model first and have it functioning properly before moving on as the model is the foundation of the whole project. If something were to go wrong in the model, it could affect most aspects down the line making it difficult to solve later on thereby making it top priority during development.

4. Data Structures Implemented

The data structure used in the APP, formats and organize this in turn leads to a consistent accurate information that helps maintain database integrity. This data structures takes the user ID from the database, and corresponds with the temperature reading and timestamp. As the user sends temperature readings using the function setTemperature to the database, the timestamp is manually implemented with it, that itself uses the function setTimestamp, but instead the data structure encrypts the timestamp and it is saved as a string in the database.

Once the user wants to read from the database, the data structure uses getTimestamp, this will then decrypted the timestamp and display the proper timestamp in day, months, year, and time. Following that the previous temperature reading will be called from the database using getTemperature, and display its reading to the user. At the moment, the current data structure only works with the user input readings, and not the actual sensor readings. Future plans to improve the structure is to have the temperature sensor implemented into the data structure and also have the a proper history of temperature readings, rather than the last reading.

5. Database Design

The overall design of the database allows the user to write and read from the database, so long as they are logged in. Without doing so, the offline user will not be able have any connections related with the database and will only have access to non-database related content. In order for the user to have communications with the database, it is required for the user to register on the register page. The register page only requires an email and password. When the user is registered, they will be assigned with a User ID, every newly created account will have a different UID, this is also the primary key. The email and password will be stored in the database authentication section, this in turn will allow the user to have access to read and write from the

database. A specific date just after the user created an account is made and as well as their last sign in, this is so the database can keep of the user.

This real time database uses the data structure to help properly structure. The UID is used in conjunction to help store the proper readings that the user requires. When the user writes to the database, the database will have the UID as well as the temperature readings and a timestamp that was saved. Although the user does not input the timestamp manually, it is automatically generated, at the same time the temperature reading was saved. While the user is reading from the database, that user will be met with the previous temperature readings and the time it was saved, this is used with the authentication the user logged in with, UID.

6. Hardware Inclusion

The hardware portion is very similar to a standard digital clock. With this hardware the plan is to implement the temperature sensor, display screen and have communications with the app using bluetooth. The hardware component will have access to the alarm, and temperature readings. The idea of it is when the user sets an alarm on the app, the hardware will recognize it and begin a countdown to when the alarm will go off with a corresponding ringtone. The hardware will be connected with a raspberry pi 3 model B, along with the temperature sensor and display screen. The display screen will show the user the current time, alarm and temperature. The temperature sensor (HTU21D-F) will provide accurate measurement readings of humidity and temperature, from which it is displayed on the hardware and the app. The hardware will constantly update the temperature according to the local area and will be recorded in the database.

7. Test Cases

<u>Authentication:</u>

This is important as it can give the user access to the database. The first thing they will see is if they can login, for new users they would have to register. If the user attempts to login without registering, a message will appear noting that the account is not registered. The register page consist of a email and password. When the user inputs a incorrect email address such as @email.com, it would not work due to the authentication recognizing that this is not a proper email address. Therefore, the user would have to use either gmail, hotmail, yahoo, etc. Once everything is met, the account is now registered and saved onto the firebase authentication. They can now log in, if any mistakes are made a warning will appear prompting the user that there is a mistake. A successful login message will appear, if they meet all the necessary credentials and they will move on the to the clock screen.

Firebase Read/Write:
In order to read and write to the database, the user first has to be logged in. If the user is in offline mode they will not be able to read or write to the database. The database is structured so that it will always use the UID as the primary key and then the temperature and timestamp under it. So if a new user saves a temperature, a new structure of their UID, temperature readings, and timestamp will be displayed on the database. This in turn can always have accurate reads of timestamp and temperature for different selected users.

Clock Screen Test:
This fragment is shown right after someone logs in, or go in offline mode. You can access this page anytime by clicking the Tab that has a label "Clock" on it. To test the app there are textViews displaying the current time, and the time of the selected time zone. Select a time zone by pressing the spinner, it will show a list of all the available time zones in standard time. Once you pick a time zone, the textView for the selected time should change to the appropriate time. Changing to landscape mode changes the layout design to fit appropriately.

Alarm Screen Test:
You can get to this Fragment by clicking the second tab labeled "Alarm". At first it should show the text "No Alarm Set" clicking the set alarm button will open up a TimePickerDialog that, on default, should be on the exact time you click the button, you can set the time by picking the hour then selecting the minutes. Or you can change to a different mode to change the alarm manually. After selecting the time, the text view should show "alarm set for ___" and will send a notification once it reaches the time selected. To cancel the alarm, click the cancel button and the text should change to "alarm canceled". Changing to landscape mode will retain an appropriate layout design.

Stopwatch Screen Test:
The stopwatch can be accessed by clicking the third tab Fragment labeled Stopwatch. In this screen it displays a chronometer for minutes and seconds. If you press the start button it will start to count up. If you press the pause button it will pause the stopwatch at the time, pressing start again will resume the stopwatch. Reset button will reset the time displayed back to 0. landscape has an appropriate design layout.

8. Android Components

Many android components and libraries are used during development of the app, each with their own specific purpose.

Major methods used for general overall use:

- **Intent (this, MainActivity.class)** - this method is often called when switching between screens (Often used with buttons or TextView links)
- **findViewById (int id)** - this method is used frequently to locate and interact with views found from layout resource files that are attached to the current activity
- **toastmakeTest(applicationContext, text, duration) -** this method is used often to send feedback to the user when they do a certain task like logging in.

Main methods for user authentication:

- **signInWithEmailAndPassword -** this method is called when the user gains access to their respective account once the proper email and password are entered
- **getCurrentUser() -** this method retrieves the data from the current user logged into the server, thereby accumulating further input they enter and save to the database

Main methods of Clock Fragment:

- **SimpleDateFormat** - allows you to set and use your own custom date format. Allows you to set the timezone format.
- **Calendar** - It's a abstract class with methods that allow you to set calendar fields such as Year. Month, Day of month, Hour.
- **TimeZone** - represents a timezone offset, you can use getavailableIds to set the spinner widget to display all the time zones.

Main methods of Alarm Fragment:

- **TimePicker** - A widget that allows you to pick a date and time 24h AM/PM, using TimerPickerDialog that displays a dialog to the user to set the time.
- **AlarmManager** - This allows you to schedule your application to be run at some point in the future, using intents to broadcast it. It retains the application when the device is asleep.
- **NotificationManager -** Class to notify the user of events that happen. This is how you tell the user that an alarm has been triggered in the background. You can set vibration,

lights, sound, descriptions and the icon. You can also put default intensity settings for vibration or sounds.

Main methods of Stopwatch Fragment:

- **Chronometer** - A class that implements a timer. You can set when the timer should start counting and also change it to a countdown timer.
- **SystemClock** - has access to all timekeeping facilities. The use of this method was to set the chronometer to 0 when you start it, so it started on the system clock time.

9. Team Contribution

Juan Rodriguez - My contributions toward this project are:

- Concept
- SRS (Software Requirement Specification)
- Creation of Powerpoints for Presentations
- Intent functionality for each page of the app
- Creation of Database
- Database Authentication for Login and Registration
- Logout/Quit Functionality
- Layout Designs and Functionality for Settings Page and About Us Screen
- French language Implementation
- Offline Mode
- Final Report (Abstract, Introduction, Design Specification, Part of Android Components, References)

Johnson Dinh - My contributions toward this project are:

- Concept
- SRS (Software Requirement Specification)
- Gantt Chart (Project Schedule)
- App Icon
- Launcher Screen
- Login/ Registration Page Layout
- Design the data structure
- Read/Write Temperature Readings To Database
- Layout Designs
- Debugging
- Final Report (Data Structure, Database Design, Hardware Inclusion, Test Cases)

Adrian Debraga - My contributions toward this project are:

- Mock UI Design
- Home Screen
- Alarm Screen
- Stopwatch / Alarm Screen
- Tabbed Fragment Layout Design
- Full Functionality Of Clock, Alarm and Timer Screen
- Final Report (Test cases, Android Components, Conclusion)

10. Conclusion

To conclude, each layout and functionality of the activities/fragments are working accordingly. The database is up and running and is linked to the app allowing for the user authentication process and saving settings and different alarms. Some features in the future are: linking different accounts with different alarms or settings, connecting our app to our hardware, allowing for more alarms to be set, and also some bug fixes we are still working on.

11. References

AustinCENG. (2018, November 01). AustinCENG/Lecture6_FirebaseAPP. Retrieved from https://github.com/AustinCENG/Lecture6_FirebaseAPP

Add Android App Links | Android Developers. (n.d.). Retrieved from https://developer.android.com/studio/write/app-link-indexing

Build a UI with Layout Editor | Android Developers. (n.d.). Retrieved from https://developer.android.com/studio/write/layout-editor

Create app icons with Image Asset Studio | Android Developers. (n.d.). Retrieved from https://developer.android.com/studio/write/image-asset-studio

Connect to Firebase | Android Developers. (n.d.). Retrieved from https://developer.android.com/studio/write/firebase

Localize the UI with Translations Editor | Android Developers. (n.d.). Retrieved from https://developer.android.com/studio/write/translations-editor

Structure Your Database | Firebase Realtime Database | Firebase. (n.d.). Retrieved from https://firebase.google.com/docs/database/android/structure-data