



VAASAN AMMATTIKORKEAKOULU  
UNIVERSITY OF APPLIED SCIENCES

Jiayi Li

# TIME RECORDER ANDROID APPLICATION

Technology and Communication  
2021

## **ACKNOWLEDGEMENTS**

I have got a lot of encouragement and support during the composition of my thesis.

To my supervisor, Ms. Pirjo Prosi, I would like to offer my sincerest thanks, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I gratefully acknowledge the effort that I received from Dr. Seppo Mäkinen and Dr. Ghodrat Moghadampour before. There is no doubt that any attempt at any level cannot be satisfactorily completed without the help of you.

In addition, I am deeply indebted to my parents for all their great support, not only on the financial level but also on the mental during the study period. I thank you for your wise counsel and sympathetic ear. You have always been there for me, giving me all support I need and unconditionally love me. I am grateful for you.

## ABSTRACT

Author	Jiayi Li
Title	Time Recorder Android Application
Year	2021
Language	English
Pages	89
Name of Supervisor	Prosi Pirjo

---

This thesis project aims to develop a time tracking system which allows users to record their working time at anytime and anywhere. In addition, they can also search the history records and send the result to their email if they want. Because of the portability of smartphone and the compatibility of Android operating system, the Android mobile operating system is chosen as a design and implementation platform of the project.

Then, in the phase of implementation, the core technologies such as the implementation of the GUI design, network communication and data storage have been studied. In details, the primary and advanced components of Android have been combined to compete the client user interface, with SharedPreferences, SQLite and MySQL database used for client data storage by situation. Volley framework was used to interact with HTTP communication between the client and server sides with PHP and Golang used for server side development. Finally, the functional requirements of system demand analysis and performance requirement were completed.

The last but not the least, the program was tested in functional modules. The tests showed that the system achieved the requirement of comprehensive functions and great performance. With the diligent work of the program developer, this mobile application can satisfy the practical requirements and improve the efficient of recording time for daily use.

---

Keywords                      Android, SQLite, PHP, MySQL, HTTP,JSON, Java

## CONTENTS

ACKNOWLEDGEMENTS .....	2
1 INTRODUCTION .....	13
2 RELEVANT TECHNOLOGIES AND TOOLS .....	15
2.1 Application Development Environment.....	15
2.1.1 Operating System .....	15
2.1.2 Java JDK .....	16
2.1.3 Android SDK.....	16
2.1.4 Android Studio .....	16
2.2 Application Technologies .....	17
2.2.1 Android .....	17
2.2.2 XAMPP .....	18
2.2.3 MySQL Database .....	21
2.2.4 Volley.....	21
2.2.5 PHP .....	22
2.2.6 JSON .....	23
2.2.7 Go Language .....	23
3 APPLICATION DESCRIPTION.....	25
3.1 Requirements Analysis of Mobile Tracking System .....	25
3.2 Use Case Diagram .....	27
3.3 Sequence Diagram.....	28
3.3.1 Sequence Diagram for User Registration .....	28
3.3.2 Sequence Diagram for User Login.....	29
3.3.3 Sequence Diagram for User Add Work Case .....	30
3.3.4 Sequence Diagram for User Edit Category.....	31
3.3.5 Sequence Diagram for User Check History and Send Result to Email .....	32
3.3.6 Sequence Diagram for User Log Out.....	33
4 APPLICATION DESIGN.....	35
4.1 System Architecture .....	35
4.2 Design of Communication .....	36
4.3 User Interface Design.....	38

4.3.1	Activity .....	38
4.3.2	Android UI Components .....	39
4.3.3	Fragment .....	48
4.3.4	Graphical User Interface Design.....	54
4.4	Data Storage Design .....	60
4.4.1	SharedPreferences.....	60
4.4.2	SQLite database .....	60
4.4.3	MySQL database.....	61
5	IMPLEMENTATION .....	64
5.1	Implementation of Registration.....	64
5.2	Implementation of Login .....	66
5.3	Implementation of Home Page.....	69
5.4	Implementation of Edit Page .....	72
5.5	Implementation of History Page .....	73
5.6	Implementation of Navigation Drawer .....	78
6	TESTING .....	81
7	CONCLUSION .....	82
8	FUTURE IMPROVEMENT .....	84
	REFERENCES .....	85

## APPENDICES

## **ABBREVIATIONS**

<b>CRUD</b>	Create, Read, Update, and Delete
<b>C/S</b>	Client/Server
<b>ER diagram</b>	Entity Relationship diagram
<b>FTP</b>	File Transfer Protocol
<b>GPS</b>	Global Positioning System
<b>GUI</b>	Graphical User Interface
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>ID</b>	Identity Document
<b>IDE</b>	Integrated Development Environment
<b>I/O</b>	Input/Output
<b>JDK</b>	Java SE Development Kit
<b>JSON</b>	JavaScript Object Notation
<b>JVM</b>	Java Virtual Machine
<b>OS</b>	Operating System
<b>QFD</b>	Quality Function Deployment

<b>SDK</b>	Software Development Kit
<b>UI</b>	User Interface
<b>URL</b>	Uniform Resource Locator
<b>XML</b>	Extensible Markup Language

## LIST OF FIGURES AND TABLES

Figure 1. Android architecture.....	18
Figure 2. XAMPP Control Panel .....	19
Figure 3. XAMPP dashboard .....	19
Figure 4. MySQL service running .....	20
Figure 5. phpMyAdmin page with database tables .....	20
Figure 6. Use Case Diagram .....	27
Figure 7. Sequence Diagram of User Register .....	29
Figure 8. Sequence Diagram of User Login.....	30
Figure 9. Sequence Diagram of User Add Working Case.....	31
Figure 10. Sequence Diagram of User Edit Category.....	32
Figure 11. Sequence Diagram of User Check History and Send Result to Email.....	33
Figure 12. Sequence Diagram of User Logout.....	34
Figure 13. System Architecture.....	35
Figure 14. The lifecycle of Activity .....	39
Figure 15. Determinate and Indeterminate ProgressBar .....	40
Figure 16. Spinner items.....	41
Figure 17. TimePicker Dialog.....	42
Figure 18. DatePicker Dialog.....	43



Figure 19. Chronometer.....	43
Figure 20. simple AlertDialog.....	44
Figure 21. Toast message .....	44
Figure 22. Sliding Navigation Drawer .....	47
Figure 23. Two separate versions of the same screen, each with a different screen size.....	49
Figure 24. The lifecycle of Fragment .....	50
Figure 25. Comparison of Fragment and Activity lifecycle .....	51
Figure 26. Register Page.....	55
Figure 27. Login Page .....	56
Figure 28. Home Page.....	57
Figure 29. Edit Category Page.....	58
Figure 30. History Page .....	59
Figure 31. Email with result table.....	59
Figure 32. ER diagram.....	62
Figure 33. loginregister database .....	63
Table 1. Quality Function Deployment for Teacher.....	25
Table 2. Non-function Requirement for System Development.....	26
Table 3. Development environment of Android client .....	64
Table 4. Test Case table.....	87

Code Snippet 1. StringRequest example .....	37
Code Snippet 2. TimePicker code .....	42
Code Snippet 3. DatePicker code.....	42
Code Snippet 4. ListView adapter.....	45
Code Snippet 5. ListViewAdapter class .....	46
Code Snippet 6. FragmentTransaction .....	52
Code Snippet 7. MainFragment class.....	53
Code Snippet 8. MainActivity class.....	54
Code Snippet 9. StringRequest object in SignUpActivity .....	66
Code Snippet 10. register.php file.....	66
Code Snippet 11. StringRequest object in LoginActivity .....	68
Code Snippet 12. Login.php file.....	69
Code Snippet 13. Handler object in MianActivity .....	70
Code Snippet 14. addworkcase.php file .....	71
Code Snippet 15. ChangeFragment method in MainActivity.....	72
Code Snippet 16. SQLiteHelper class .....	73
Code Snippet 17. StringRequest object of searching history in MainActivity .....	75
Code Snippet 18. WorkinghourAdapter class .....	76
Code Snippet 19. Search history function in Golang .....	76

Code Snippet 20. StringRequest object of send email in MainActivity .....	77
Code Snippet 21. Send email function in Golang .....	78
Code Snippet 22. Navigation methods .....	79
Code Snippet 23. onClick properties of Navigation items .....	80

## **LIST OF APPENDICES**

**APPENDIX 1.** Testing case table

# 1 INTRODUCTION

This thesis focused on the development of an Android time recording application to help lecturers keep track of their working hours more efficiently.

Advances in mobile technology have improved people's lifestyles over the last decade, making their lives more convenient. Meanwhile, people's reliance on mobile phones allows mobile applications to thrive, promoting the rapid development of mobile applications. For instance, people get used to replying to emails while waiting for a plane at the airport. Young people have long been used to chatting with friends on social media while riding the bus. Due to portability, people have become accustomed to using smartphones to deal with daily affairs. It is widely acknowledged that the mobile internet era has arrived.

At present, the major platforms are primarily the Android and iOS operating systems. The Android phones have gradually overcome their lagging issues thanks to continuous improvements in smartphone hardware configuration and operating system optimization. Furthermore, Android-based mobile devices are more popular than iOS-based mobile devices due to Google's openness and tolerance for Android. Android devices offer a wider range of services and applications, as well as a greater number of free downloads. Mobile phones which run the Android OS hold 83.8 percentage shares of the market in 2021 /1/. Since the emergence and breakthrough of new technologies, developers have begun to try to apply new technologies in variety application scenarios.

There are several time management applications on the market, such as TimeRecorder and Hours. The time management applications record the time that customers spend and summarize the trend of time usage. The objective of this thesis project was to develop a time tracking application which records teachers' working hours, summarizing and analyzing the time spent by the lecturers. Finally, a full-time report will be sent to lecturers'

email, therefore the teachers can better stay within the given hourly resources.

The research purpose of this topic is to explore the entire development process of this Android application including the design of user interface, back-end logic, network communication protocol and data storage from scratch. In the implementation of user interfaces, the key components, such as different layouts, navigation drawer and fragment were studied. For the network communication, the Volley framework was chosen, which can deal with the HTTP request successfully. In the data storage part, suitable storage solution was made according to the requirement of different business scenarios. For example, MySQL database was used to store the user's registration information and working records data. The SQLite database was used to store working categories value and SharedPreferences to store the user's id and email; when the user logs in and reuses these data when saving the working records and sending email.

This thesis is structured as follows. Chapter 2 gives the theoretical research about technologies. Chapter 3 describes the application description. Chapter 4 describes the application design work. Chapter 5 provides the implementation work. Chapter 6 presents the test results of the project. Chapters 7,8 concludes the overall work and future improvement.

## **2 RELEVANT TECHNOLOGIES AND TOOLS**

The related technology and tools used in software development are described in this section. The tools used in the application development environment are introduced first, followed by the technologies used in application implementation and testing.

### **2.1 Application Development Environment**

In order to start developing an Android application, the appropriate development environment needs to be set up firstly.

#### **2.1.1 Operating System**

First of all, an operating system (OS) is required because it manages different resources of the computer. A user interface is also established by the OS and OS can execute for multiple applications. The Microsoft Windows, Linux, Apple macOS, Android and Apple's iOS are the most common operating systems. However, the hardware needs to be appropriately chosen in order to avoid some unexpected problems when developing the Android applications. For instance, the powerful M1 chip is not perfect for developing Android applications yet. It is not possible to install Intel HAXM for Android Emulator, yet even though there is an Android Emulator provider for M1. The emulator is still in the preview stage /2/. The appropriate OS for Android application development are:

- Microsoft Windows 8 or later version
- Linux (includes the GNU C Library) 2.7 or later version
- Mac OS X 10.5.8 or later version (Intel chip)

### **2.1.2 Java JDK**

Java Development Kit (JDK) is a software development kit which contains essential tools to write Java programs. The JDK includes a compiler to convert the Java code into bytecode. All JDK versions comes bundled with the Java Runtime Environment (JRE). The JRE is a piece of software which contains class libraries, loader class, Java Virtual Machine (JVM) and other supporting files. The JVM that has many libraries, tools and frameworks offers a platform-independent way of executing the Java source code. The JVM comes with a Just-in-Time compiler which converts the source code into machine language. Java JDK 5 and JRE 6 are the minimum required versions for Android application development.

### **2.1.3 Android SDK**

The Android software development kit (SDK) contains a comprehensive set of software development tools, libraries, sample code and documentation. The corresponding SDK that comes with the Google newly released update of Android is included in the Android Studio Integrated Development Environment (IDE). The Android SDK provides all needed tools for developing programs from scratch, debugging and testing on virtual devices.

### **2.1.4 Android Studio**

Android Studio that is an official IDE for Android development was announced at the Google I/O conference in 2013. Google and JetBrains developed the Android Studio specifically for Android development. The IDE can be downloaded from the official developer's website. Android Studio consists of all necessary tools for building Android applications.



## **2.2 Application Technologies**

### **2.2.1 Android**

Android is a Linux-based operating system. It was announced by Google on 5 November 2007 with the Smartphone operating system. The platform consists of operating system, middleware, user interface and application software. It is regarded as the first fully open and comprehensive mobile terminal software.

It is comparatively easier to build and release applications on Android due to Google's open source and openness policies than on iOS. Developers can call the hardware devices such as mobile phones, GPS, gyroscope, camera and so on according to their own application needs, and can also access local contacts, calendars and other information. The development of applications on Android does not require Google authentication, so the whole application market of Android is flourishing.

Android can seamlessly integrate with Google's map service, email system and search service, and some of them have even been embedded into the Android system. Android was released in 2007 and the latest version is Android 8.0.

From top to bottom, the Android device architecture is separated into five layers: the application layer, application framework layer, system runtime layer, hardware abstraction layer, and Linux kernel layer. Figure 1 presents the system architecture of Android.



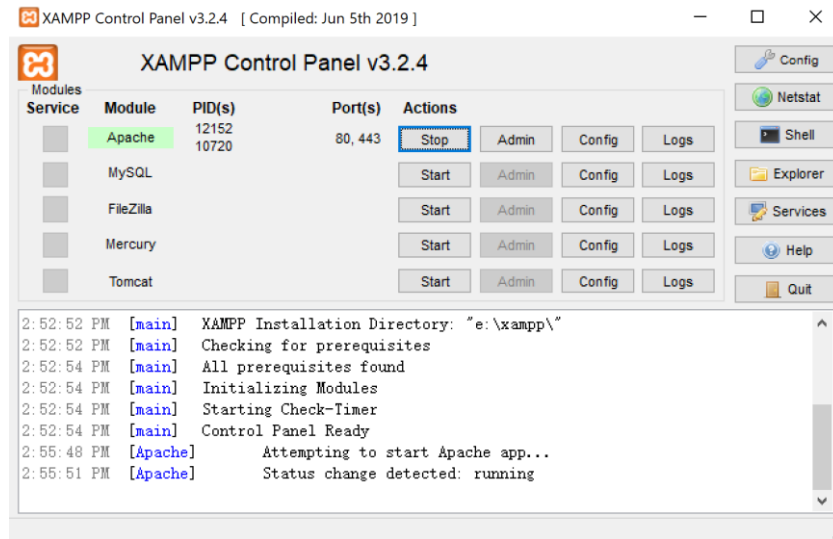
**Figure 1.** Android architecture /3/

## 2.2.2 XAMPP

XAMPP (Apache-MySQL-PHP-PERL) is a powerful station-building integration package whose original name was LAMPP, but the new version was changed to XAMPP to prevent confusion. It works on a variety of operating systems, including Windows, Linux, Mac OS X, Solaris, and others.

Installing Apache servers is not easy, as many users have discovered from their own experiences. Adding MySQL, PHP, and Perl to the combination is even more difficult. XAMPP was developed as an easy-to-install Apache distribution with MySQL, PHP, and Perl to prevent these complicated steps.

After downloading XAMPP, we can launch the XAMPP Control Panel, as shown in Figure 2:



**Figure 2.** XAMPP Control Panel

When starting the Apache program, the word 'Apache' turns green. Then it can open the browser to reach localhost (or 127.0.0.1 IP address), and the view shown in Figure 3 indicates that the server environment was successfully installed.



**Figure 3.** XAMPP dashboard

Next, XAMPP can be used to test the MySQL service. By clicking 'Admin' after the MySQL service has started, as shown in Figure 4 and the phpMyAdmin page shown in Figure 5 will come up:

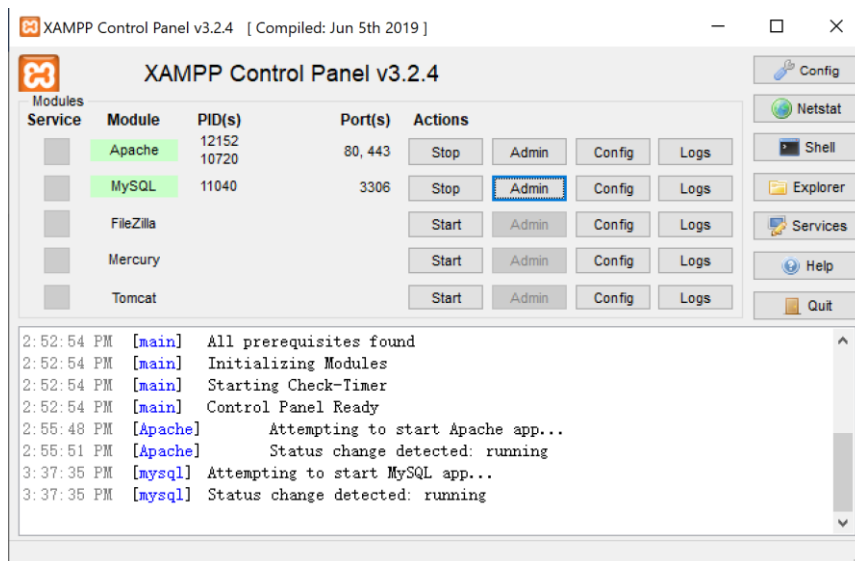


Figure 4. MySQL service running

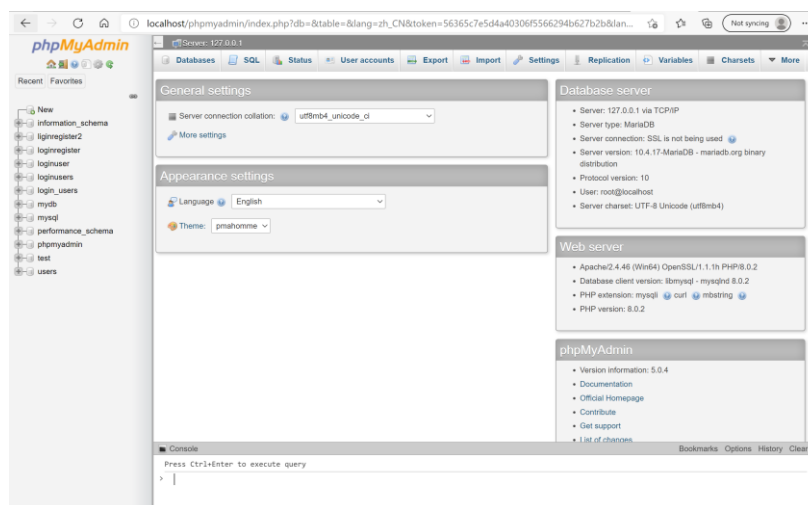


Figure 5. phpMyAdmin page with database tables

### **2.2.3 MySQL Database**

Database refers to a data set that is stored together in a certain way, can be shared with multiple users, has as little redundancy as possible, and is independent of the application program. It can be regarded as an electronic filing cabinet - the place where electronic files are stored. Users can add, query, update and remove data on the file, as well as other operations. A warehouse used to access and manage certain data.

MySQL is a relational database management system developed by MySQL AB company in Sweden, and currently belongs to Oracle company. MySQL is a relational database management system. Rather than storing all data in a massive warehouse, the relational database saves it in separate tables, which improves speed and flexibility.

Since MySQL uses a C/S architecture, there are two programs in operation. The MySQL server program, also known as the `mysqld` program, is one of them. It is responsible for monitoring and handling service requests from network clients and operates on the database server. According to these requests, it accesses the contents of the database, and then sends the relevant information back to the client [4]. The other program is the MySQL client program, which is responsible for connecting to the database server and issuing commands to tell the server what it wants to operate.

### **2.2.4 Volley**

We almost always have network technologies to use when building Android applications, and most applications will transmit and receive network data through the HTTP protocol. The two main HTTP communication methods supported by the Android system are `URLConnection` and `HttpClient`. These two classes can be found and used frequently in almost every project code.

The use of `HttpURLConnection` and `HttpClient`, on the other hand, is complicated. It is possible to have to write plenty of repetitive code if it is not properly encapsulated. As a result, several Android network communication systems, such as `AsyncHttpClient`, have been developed. In the internal, `AsyncHttpClient` encapsulates all HTTP communication information. By simply calling a few lines of code, it can complete the communication operation. Another example is `Universal-Image-Loader`, which simplifies the process of viewing network pictures on the interface. Developers will not have to worry about how to get pictures from the internet, how to start threads, how to recycle picture tools, or any other information. `Universal-Image-Loader` has done an excellent job.

The Android development team recognizes the need to simplify HTTP communication, so at the 2013 Google I/O conference, they introduced Volley, a new network communication system [5]. Volley combines the benefits of both `AsyncHttpClient` and `Universal-Image-Loader`. It can not only interact with HTTP as efficiently as `AsyncHttpClient`, but it can also load images over the network as efficiently as `Universal-Image-Loader`. Volley has made significant performance improvements in addition to being quick and easy to use. Its purpose in design is to be well-suited to network activity with little data but regular contact. Volley's efficiency would be low for network operations which involve large amounts of data, such as uploading files.

### **2.2.5 PHP**

PHP is a server-side HTML scripting language. It is a versatile scripting language widely used in open source, especially suitable for web development and can embed HTML. Its syntax is close to C, Java and Perl, and it is easy to learn. The language allows web developers to write dynamic and interactive web pages quickly.

## 2.2.6 JSON

JSON is a data structure that can be used instead of XML. It is more lightweight than XML and has the same definition potential as XML. Because of its compactness, data transmission flow in the network would be reduced. While JSON is just a string, the elements are labelled with symbols.

{ }: Objects are indicated by curly brackets.

[ ]: Square brackets are used to represent an array.

"": A property or value is included inside the double quotation marks.

: The colon means that the value of the latter is equal to the value of the former (this value can be a string, a number, or another array or object)

So {"name": "Michael"} can be understood as an object containing the name of Michael.

And [{"name": "Michael"}, {"name": "Jerry"}] represents an array containing two objects.

Of course, it can also use {"name": ["Michael", "Jerry"]} to simplify the above part. This is an object that has a list of names.

## 2.2.7 Go Language

Go (also known as Golang) is a programming language developed by Google, it is strongly typed, compiled, parallel and has garbage collection function. The Go language is specially optimized for the programming of multiprocessor system applications. The program compiled by go language is comparable to the speed of C or C++ code, and it is more secure and supports parallel processes. The Go language is mainly used for server-side development and is suitable for the development of "large-scale soft-

ware". It has a long development cycle, supports cloud computing, and combines the efficiency of traditional compiler language with the ease of use and expressiveness of script language. /6/



### 3 APPLICATION DESCRIPTION

This chapter first describes the requirements analysis process of the mobile tracking system, then analyses the system's function and performance requirements from the system development. After obtaining the overall architecture scheme of the system illustrated by Use Case Diagram, each function module of the system will be presented by Sequence Diagram.

#### 3.1 Requirements Analysis of Mobile Tracking System

The requirements analysis of the system is an important step before the system development. The further implementation and testing can be done smoothly only by clarifying the system criteria priorities and rendering a reasonable and feasible architecture scheme. Quality Function Deployment (QFD) is a process and collection of tools for defining consumer requirements and converting them into comprehensive engineering specifications and plans for manufacturing the products that meet those requirements. The Quality Function Deployment (QFD) process is perhaps the most powerful methodology for catching and listening to the “voice of the customer” [7]. Table 1 below declares the function requirements expected from the users by priority.

**Table 1.** Quality Function Deployment for The User

Must have Requirement with priority level 1
<ul style="list-style-type: none"><li>• The application must have authentication pages.</li><li>• The user must be able to login the system by providing email and password.</li><li>• The user must be able to logout the system from the home page.</li><li>• The application must have time tracking system, for example, chronometer.</li><li>• The user must be able to select date, working types and record working hours from home page.</li><li>• The user must be able to add/delete working categories.</li><li>• The corresponding list must be able to be updated with newest value.</li><li>• The user must be able to retrieve history records and send result to email.</li></ul>

<ul style="list-style-type: none"> <li>• The application must be able to calculate total hours of retrieving list.</li> </ul>
Should have Requirement with priority level 2
<ul style="list-style-type: none"> <li>• The user should be able to see response result if authentication fails.</li> <li>• The user should be able to reset chronometer for error operation.</li> <li>• The user should be warned if no records satisfying the filters conditions.</li> <li>• The application should be user-friendly.</li> </ul>
Nice to have Requirement with priority level 3
<ul style="list-style-type: none"> <li>• The user can see the summary of history record in pie chart.</li> <li>• The user can reset the password if forgetting password.</li> </ul>

**Table 2.** Non-functional Requirements for System Development

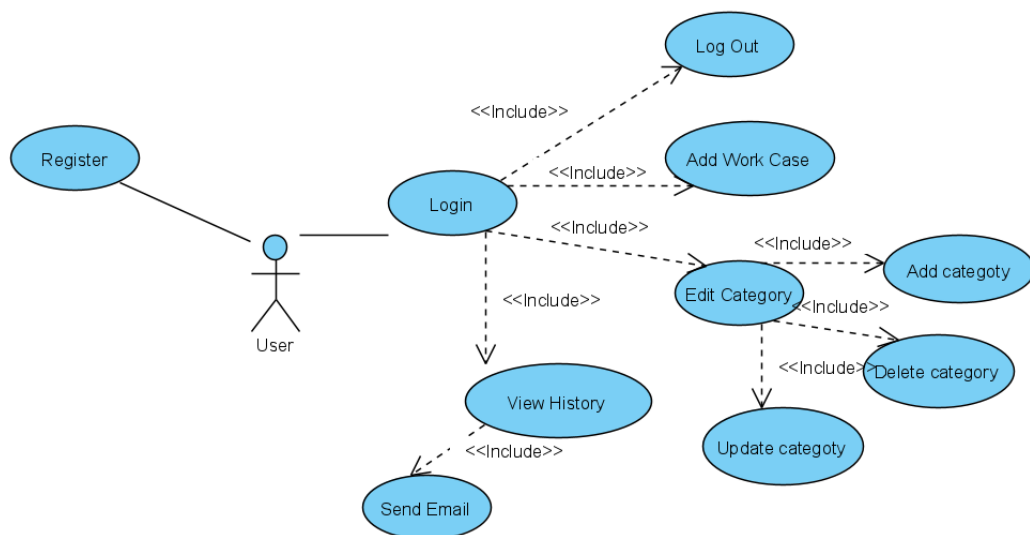
Requirement	Description
Safety	Only registered users can access to the program. Users need to login to the program with registered information.
Usability	The user interface is fancy, so the new users can operate system without doubt.
Functionality	The program can provide users with various operations and accurate result.
Maintainability	The system should be able to operate continuously and stably after development and testing

Table 2 states the non-functional requirements which are requested to achieve for system development. The system must first be smooth and feasible, then maintain safe running, and eventually, in the latter stages of the project, simple maintenance.

## 3.2 Use Case Diagram

Diagrams for case use are often used to define functions and the connections between roles and use cases. They describe who will be using the system and what they will be able to do with it. A use case diagram contains multiple model elements, such as systems, actors, and use cases, and shows various relationships between these elements, such as generalization, association, and dependency. It displays to an external user a functional model diagram of the system. /8/

The actors in this application are users. Users need to first create accounts on the Registration page before they can login to the program with their registered information. After signing into the program, users can use the chronometer to record working hours, edit working categories, access history records using the filters feature, and submit a list of results to users' email addresses. They can also log out of the system at any time. The use case diagram is presented in Figure 6.



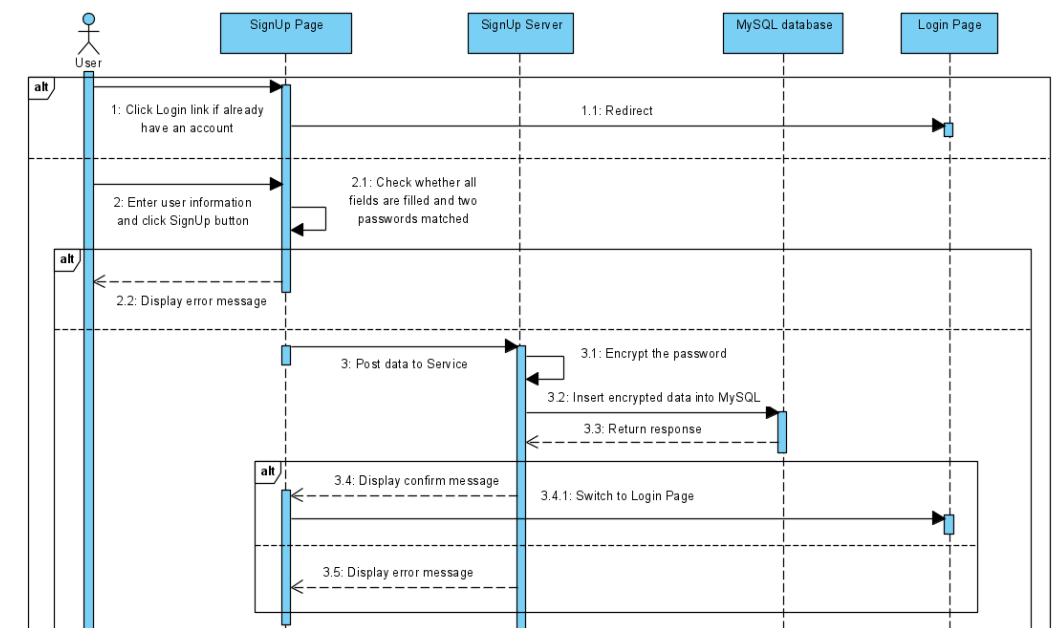
**Figure 6.** Use Case Diagram

### **3.3 Sequence Diagram**

A sequence diagram is a diagram that depicts the relationship between objects in sequential order. The sequence diagrams show the interaction between objects as well as the order in which messages are exchanged between them. The sequence diagram's modeling elements primarily include: actor, lifeline, message, and so on. In this section, there are several sequence diagrams that demonstrate the key use case of running this android program in order to get a better understanding of the role of each part of each Activity.

#### **3.3.1 Sequence Diagram for User Registration**

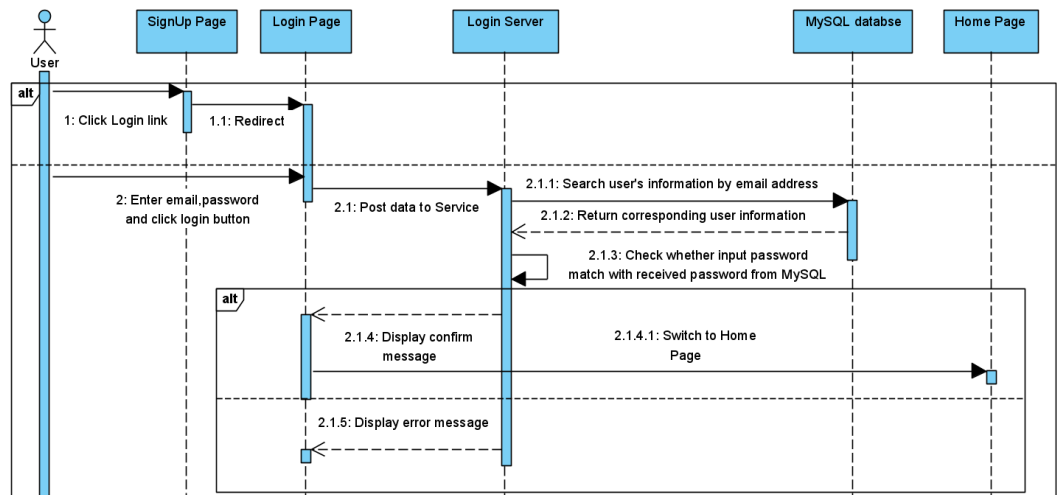
The user register page is the first page that appears after starting the program. The user should fill in his or her details to the specified blank fields on this page, such as the full name, email address, password, and confirm password. After clicking the "SignUp" button, the user may send the details. If any blank fields are not filled or the password does not fit the confirm password, the device will show an error message. While all information is filled out correctly, the device will transfer the data to the server side using the Volley framework, and the server will encrypt the password into a hash model to accomplish the purpose of securing the user's privacy. The server side would then use PHP scripts and INSERT INTO statement of SQL to insert the data into the MySQL database. The register page will show the message "Register Success!" if the procedure for entering user data into the database is correct. Otherwise, the recipient will get an error message that says "Register Error!". In addition, if the user already has an account, there is a link below the submit button that the user may choose to press. As a result, the registration page will redirect to the login page, allowing users to log in directly. Figure 7 shows the sequence diagram for this whole process.



**Figure 7.** Sequence Diagram of User Register

### 3.3.2 Sequence Diagram for User Login

After creating an account, the user can access the application using his or her email address and password. First, the user must fill in the blank fields with the registered information and press the Login button; then, using the Volley framework, the filled data will be posted to the server side. After that, the Login server can use the SELECT statement of SQL query with email data as a condition parameter to retrieve the user's registered password value from the MySQL database. The Login server would then need to verify that the registered password value matches the password entered by the user on the Login page. If these two values are the same, this page will show the confirmation message "Login Success!" before returning to the home page. Otherwise, an error message would be shown on the Login page. Figure 8 shows the sequence diagram for this whole process.

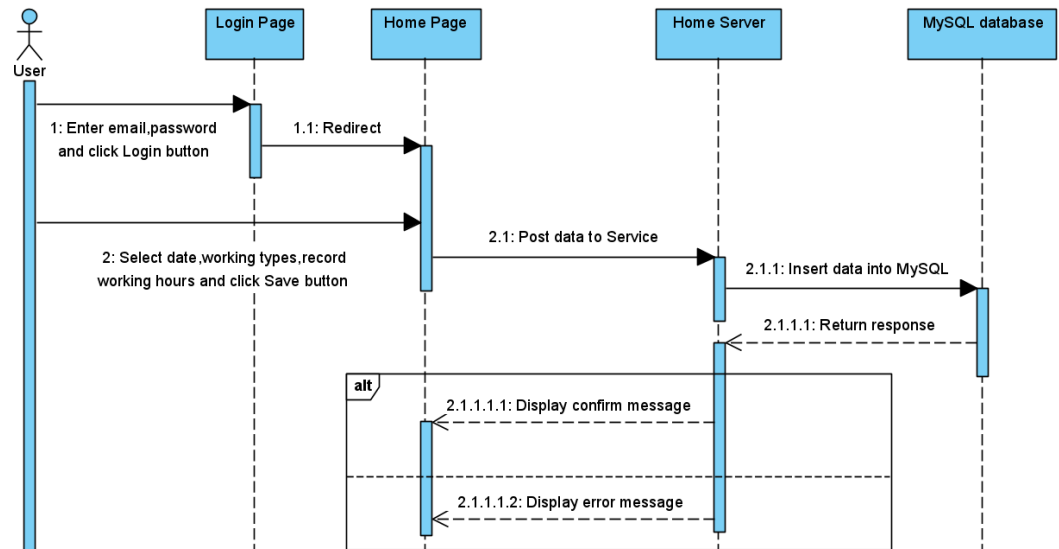


**Figure 8.** Sequence Diagram of User Login

### 3.3.3 Sequence Diagram for User Add Work Case

This key responsibility of this application is to track of teachers' working hours. After logging into the program, the user will be taken to the home screen, which has a sliding menu in the top corner. On this home page, the user can choose the working date, the working types from two drop-down lists, and the working hours from three measurement methods. The first method involves pressing the chronometer's start button before he or she begins working and then pressing the stop button when the job is completed. The second method is to enter the start and end times. The last method allows the user to input the working hours value directly into the result viewing space of the first two methods, which is clearly more convenient than the other methods. Once the user has completed all the fields and clicked the Save button, the device will send all the information to the server side. The data would then be inserted into the MySQL database using PHP scripts and INSERT INTO statement of SQL by the server. If the operation is successful, the server will receive the MySQL confirm response and display the message "Data added success!" on the home page. Otherwise, the server will receive an error response from the database and display the error

message instead. Figure 9 shows the sequence diagram for this whole process.

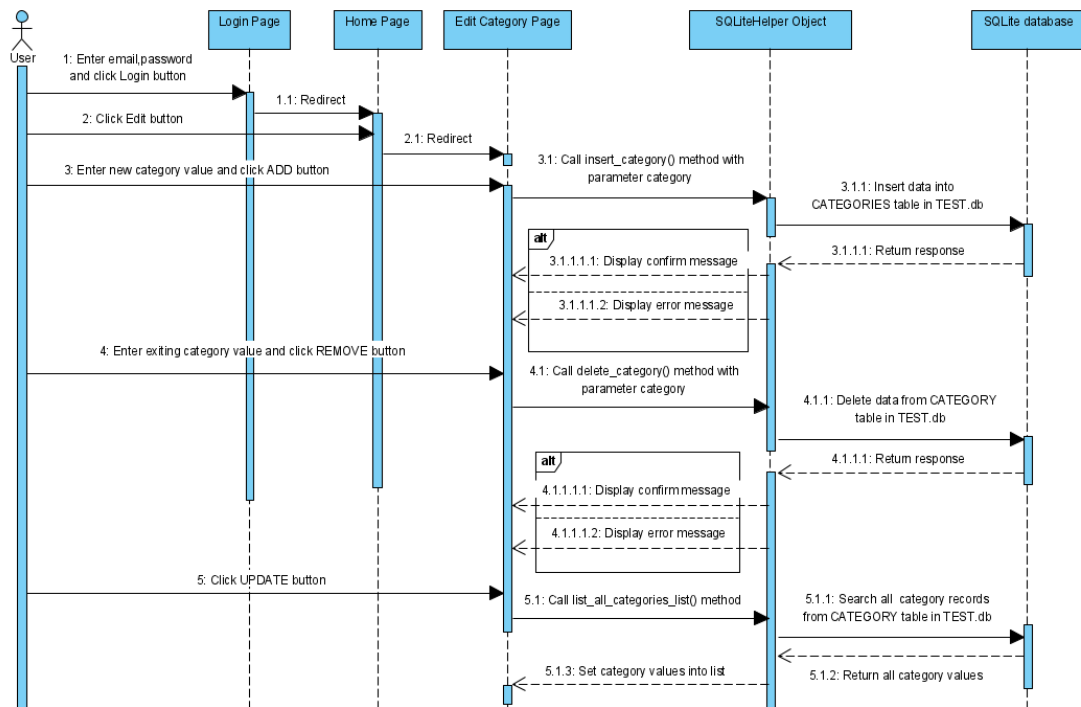


**Figure 9.** Sequence Diagram of User Add Working Case

### 3.3.4 Sequence Diagram for User Edit Category

To make the process of recording time more flexible, the user can add or remove values from a single drop-down list showing the working type that appears on the home page. After pressing the Edit button next to the list, the user would be taken to the edit page. On this page, the user can fill in the blank space with the category value he or she wishes to add to the list, then press the ADD button. The value will be entered into the CATEGORY table of the SQLite database. By entering the category name and clicking the REMOVE tab, the user may also delete category values that are no longer in the list. Finally, the user must press the UPDATE button to display the updated values in the list. In order to achieve that, the SELECT statement of a SQL query will be used in the SQLiteHelper Object to search and retrieve all the category values in the CATEGORY to the ArrayList. The ob-

ject would then use ArrayAdapter to set all the group values into the displaying list. As a result, the spinner on the home page and the list that appears on the edit page will both simultaneously refresh with the most recent array values. Figure 10 shows the sequence diagram for this whole process.



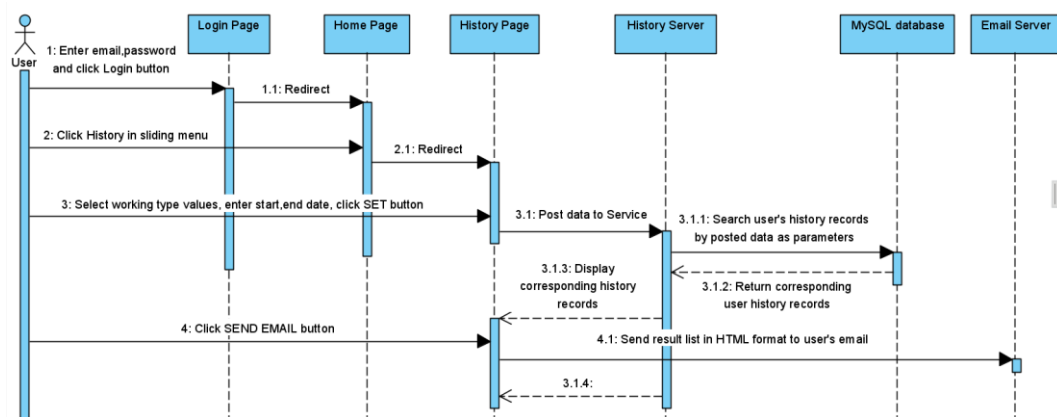
**Figure 10.** Sequence Diagram of User Edit Category

### 3.3.5 Sequence Diagram for User Check History and Send Result to Email

There is a sliding menu in the top left corner that appears on the home screen. Three options are included in this menu, home, history and logout. If the user wishes to review the record history, then the user is sent to the History page by clicking on the History icon. On this page, the user can choose the two values of the working types and set the start/end date that he or she wants to see. When he or she has configured all the filters, the data will be posted to server side using Volley framework after the user



clicks the SET button. The server would then search for the user's history records in the MySQL database using the SELECT statement of a SQL query with posted data as parameters to retrieve the user's history records. Finally, on the History page, all recovered records will be put in the ListView below the SET button. As a result, the user will view the history records based on the given period and different categories, with the total working hours shown below the column. If the user wishes to send the result report by email, the SEND EMAIL button is pressed after receiving the result records. The Email Server will directly set the result record value into a table written in HTML format as the email content. Following that, the user will receive an email with the same table result that appears on the history page. Figure 11 shows the sequence diagram for this whole process.

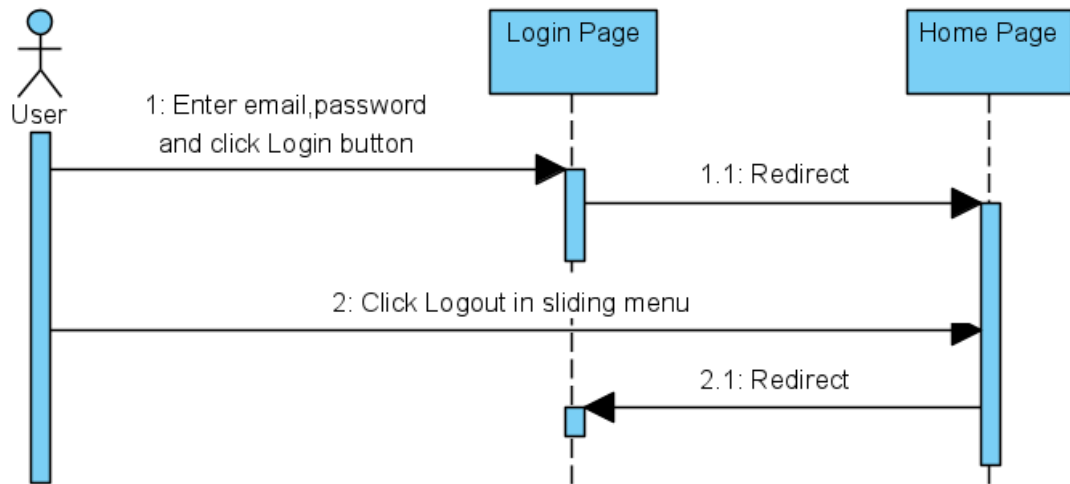


**Figure 11.** Sequence Diagram of User Check History and Send Result to Email

### 3.3.6 Sequence Diagram for User Log Out

If a person has logged into the program, he or she can also log out. After entering the login information and successfully reaching the home page, the user can press the Logout value in the sliding menu as previously mentioned, and a dialog with the warning message "Are you sure you want to

log out?" appears. If you are unsure about that, click the NO button. In comparison, if the user clicks the YES button, the application page will be redirected to the Login page. Figure 12 shows the sequence diagram for this whole process.



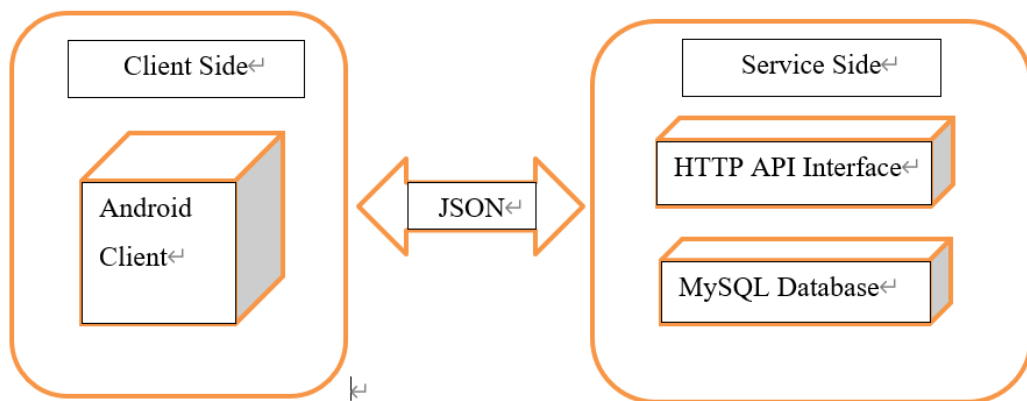
**Figure 12.** Sequence Diagram of User Logout

## 4 APPLICATION DESIGN

This section mainly introduces the system architecture, design of communication protocol, the user interface design, database design.

### 4.1 System Architecture

There are two sections of this system, the first section is the client side which displays the user interface of the application. The other section is the server side which provide server of storing data. For the server side, the main body uses Apache + MySQL + PHP to construct the application software for the corresponding server architecture of the HTTP request module. The server solution includes the Apache architecture, which is used to create the main structure of the server, and MySQL, which is used to store the back-end data as seen in Figure 13. The development of the client side is based on the Android operation system and the server side is deployed on XAMPP.



**Figure 13.** System Architecture

The client uses the HTTP protocol to reach the server, which then executes the requested process and returns the corresponding data. The JSON data format is used when exchanging data between the client and the server.

## 4.2 Design of Communication

The communication protocol serves as the connection between the client and the server in a C/S architecture system. The `HttpURLConnection` provided by JDK (java development kit), `HttpClient` supported by Apache, and some mainstream open-source frameworks are the main ways to achieve network communication on the Android platform.

In this program, the Volley framework was used in the HTTP network communication of the Android client module's "register and login," "add work case," "check history," and "send email" functions.

Volley is quite simple to use. First, it will submit an HTTP request and get an HTTP response to the most fundamental HTTP correspondence. To start, it first needs to obtain a `RequestQueue` object that can be obtained with the following method:

```
RequestQueue mQueue =Volley.newRequestQueue(context);
```

The `RequestQueue` object obtained here will cache all HTTP requests and then submit these requests simultaneously according to an algorithm. Because `RequestQueue` is well suited to high concurrency, in any activity that needs to communicate with the network basically, one `RequestQueue` object is created. /9/

Next, it is needed to create a `StringRequest` object to submit an HTTP request as seen in Code Snippet 1:

```

StringRequest stringRequest = new StringRequest(Request.Method.POST,
url, new Response.Listener<String>() {
    @Override
    public void onResponse(String response){
        Log.e("TAG", response);
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e("TAG", error.getMessage(), error);
        }
    }) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> map = new HashMap<>();
        map.put("params1", value1);
        map.put("params2", value2);
        return map;
    }
};

```

### Code Snippet 1. StringRequest example

StringRequest requires four parameters to be passed to its constructor. The first parameter is the HTTP function, the second parameter is the URL address of the target server, the third parameter is the callback for an effective response from the server, and the fourth parameter is the callback for a failure response from the server. Volley can attempt to obtain the POST parameters by calling the getParams() method in Request, the parent class of StringRequest.

Finally, as shown below, add this StringRequest object to RequestQueue:

```
mQueue.add(stringRequest);
```

Furthermore, since Volley would need to connect to the Internet, the Internet permission must be added to the AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET" />
```

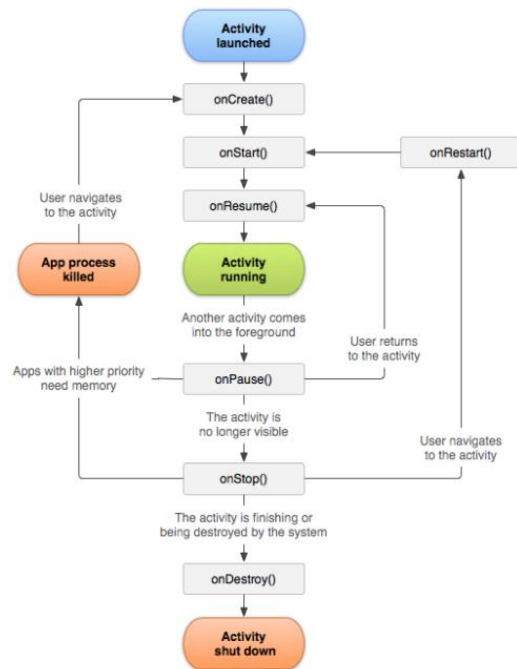
## 4.3 User Interface Design

### 4.3.1 Activity

Activity is one of the four major components of Android. It is a visual interface that gives users a window to process instructions. We must call the `setContentView` method after we have generated the Activity to complete the presentation of the interface and provide users with an integrated entrance. Activity is the most common used component in development since almost everything that can be seen in the Android APP depends on Activity.

Activity is managed by the Activity task stack. If we now open an Activity A, the Activity B will be put on top of the stack and labelled as running state when we open a new Activity B. At the same time, Activity A is tucked under the stack and goes into the background. If Activity B, which has just been opened, is destroyed, Activity A will return to the top of the list.

Figure 14 illustrates the lifecycle of Activity, the rectangles represent the methods that Activity needs to callback between states. The colored ovals represent the mainly states that Activity is in.



**Figure 14.** The lifecycle of Activity /10/

### 4.3.2 Android UI Components

Some Android common layouts were used in the project. There are five common layouts in the system SDK in Android. All layouts inherit ViewGroup, namely LinearLayout, RelativeLayout, FrameLayout, AbsoluteLayout, TableLayout.

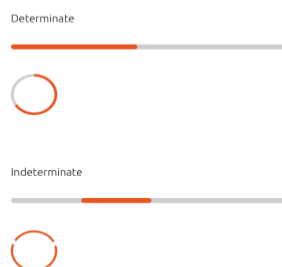
LinearLayout is a linear layout control, a subclass of ViewGroup, that sorts the child views according to the value of the android:orientation attribute and can arrange them vertically or horizontally. Each LinearLayout subview will appear on the screen in the order that they appear in the XML.

RelativeLayout is a relative layout in which the component location is determined based on its relative position. A component needs to depend on another control or a parent component. In actual layouts, this is one of the

most widely used layout methods. It is more flexible, and has many attributes, but it is still more complex to use. (For example, to build a text view with the attribute `toLeftOf="@id/my button"` to put a text view on the left side of a button.)

FrameLayout is one of the simplest layouts in Android, which simply opens a blank space on the phone. All components will be located on the upper left corner of this field when adding components to it. Only the top component can be shown at the same time if all the components are the same size. Of course, the alignment can be defined by adding the `layout_gravity` attribute to the component.

Some Android primary components were used in the project. In Android, the ProgressBar is most widely used. There are two forms of ProgressBars: determinate and indeterminate. What is determinate is that the progress can be seen clearly, and what is indeterminate is that it is not clear, and it is not sure how long an operation will take to complete. In real enterprise development, a determinate progress bar is generally used to indicate the progress of downloading files, and an indeterminate progress bar is used to indicate that the network is being accessed. Figure 15 below shows these two types of progress bars.

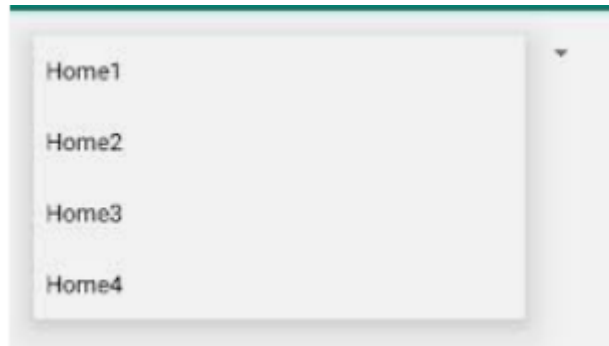


**Figure 15.** Determinate and Indeterminate ProgressBar

Spinner is a drop-down control of the Android system. A collection list occurs when the spinner is pressed, and it looks like a button. A spinner collection list is shown in Figure 16. The XML `android:entries` attribute can be



used to specify a spinner option, or you can use a data adapter to programmatically load items.

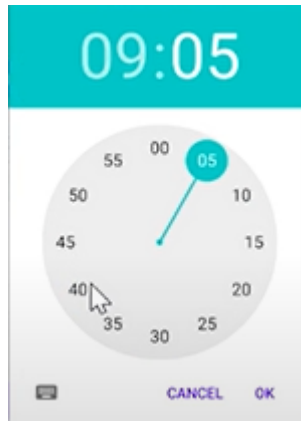


**Figure 16.** Spinner items

TimePicker Dialog is the time control in Android applications appearing as a pop-up dialog box to let user choose time. It needs to implement the OnTimeSetListener interface as seen in Code Snippet 2. The result time can be set in specific format by calling SimpleDateFormat object and the result is as Figure 17 shows.

```
final Calendar calendar = Calendar.getInstance();
    TimePickerDialog.OnTimeSetListener timeSetListener = new
TimePickerDialog.OnTimeSetListener() {
    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int
minute) {
        calendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
        calendar.set(Calendar.MINUTE, minute);
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat(
mat("HH:mm"));
        time.setText(simpleDateFormat.format(calendar.get-
Time()));
    }
};
    new TimePickerDialog(getActivity(), timeSetListener, calen-
dar.get(Calendar.HOUR_OF_DAY), calendar.get(Calendar.MINUTE),
true).show();
```

## Code Snippet 2. TimePicker code



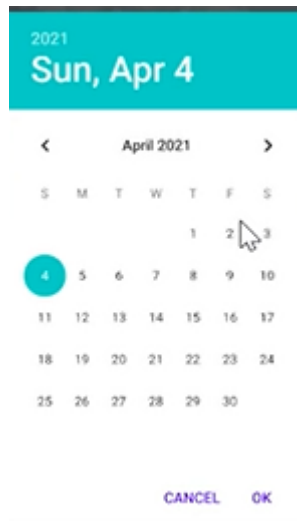
**Figure 17.** TimePicker Dialog

In Android applications, date controls include DatePicker and DatePickerDialog, which are essentially the same. The use of DatePickerDialog is a little more complicated. It appears as a pop-up dialog box and needs to implement the OnDateSetListener interface (mainly the onDateSet method).

```
Calendar calendar = Calendar.getInstance();
    final int year = calendar.get(Calendar.YEAR);
    final int month = calendar.get(Calendar.MONTH);
    final int day = calendar.get(Calendar.DAY_OF_MONTH);
    DatePickerDialog datePickerDialog = new DatePickerDialog(Main-
Activity.this, new DatePickerDialog.OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int
month, int day) {
            month = month + 1;
            String date = year + "-" + month + "-" + day;
            editText.setText(date);
        }
    }, year, month, day);
```

## Code Snippet 3. DatePicker code

In this way, when the date in DatePickerDialog is changed, the date in edit-Text changes accordingly. The result is seen in Figure 18:



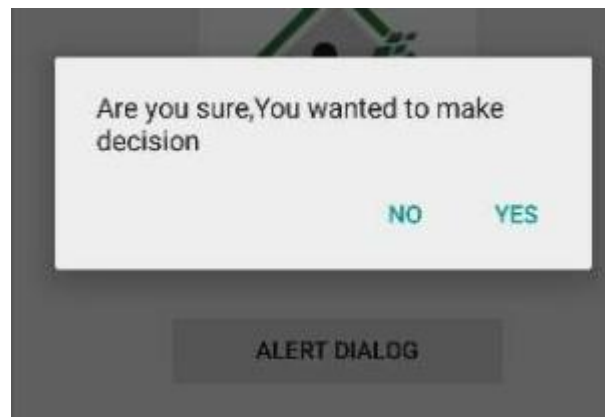
**Figure 18.** DatePicker Dialog

Chronometer can be used as a timer to see the time passage rather than the phase rise. It is extremely helpful when recording the time, a user takes to carry out an action, or when limiting the time in the game is required. Here, the Chronometer object format property can be used to configure the text around the viewing time. The time and additional text only appears in after calling the start() method To interrupt the timer, the stop() method may be called. The setBase() method is used to set the starting point of the timer. When the timer is set to 0, the timer will start from the last time the phone was restarted. Figure 19 below presents the effect picture of Chronometer.



**Figure 19.** Chronometer

When developing for Android, it is common to bring up several dialog boxes on the Android app, such as to ask the user a question or to give the user a choice (such as deleting dialog box, warning dialog box). The AlertDialog dialog box implements these features with the title, content, image, and two-button listen events set. Figure 20 depicts the running influence.

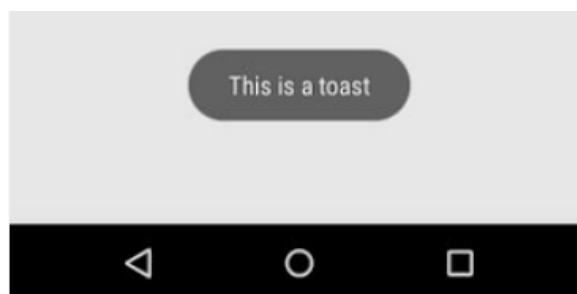


**Figure 20.** Simple AlertDialog

Toast is a prompt method in the Android system, displaying a paragraph to prompt the user. The function of the Toast is to inform the user of what is going on right now. It cannot interfere with the user's operation, so the user will only passively acknowledge it. Toast is very easy to use; the official SDK has contributed to encapsulating it. The easiest way to use it is to call the static method `makeToast` of the `Toast` class and enter three parameters: context, prompt content, and display duration:

```
Toast.makeText(this, "This is a toast", Toast.LENGTH_LONG).show();
```

The `makeText` method returns the `Toast` object that appears while the `show` method is called. Figure 21 illustrates the operational effect.



**Figure 21.** Toast message

Some Android advanced components were used in the project. In Android development, ListView is a very common component, which displays the specific class content in the form of a list and can be displayed adaptively according to the length of the data.

The display of the list requires three elements:

- ListView: This is the view that is used to display the list.
- Adapter: A data mapping intermediary that connects the data to the ListView.
- Data source: Each row of data in the data source corresponds to a row of View in the ListView.

Using a pre-made adapter class is the simplest way to connect data to a view. SimpleAdapter can be used to map static data stored in an array. SimpleCursorAdapter makes it simple to bind data from a query. Both adapters need several parameters to map the underlying data to each item's display structure. A simple example is seen in Code Snippet 4:

```
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, android.R.layout.simple_list_item1, cursor, new String[] {TITLE}, new int[] {android.R.id.text1});
setListAdapter(adapter);
```

#### **Code Snippet 4.** ListView adapter

The code creates a SimpleCursorAdapter that can bind data to the built-in simple\_list\_item1 layout. This layout is used to represent a single line of text in the ListView. All data is mapped to the ID of the view in the layout based on the marked column of the database. Once the adapter is created, it will be bound to the AdapterView, which will display the data to the user.

When there are just a few views, a simple adapter will suffice, but for more complex data, a custom adapter is required. To create a custom adapter, it is needed to override the Adapter class and implement the getView method.

We create a ListViewAdapter class, derived from BaseAdapter, and we override four methods as seen in Code Snippet 5.

```
public class ListViewAdapter extends BaseAdapter{
    private List<String> data;
    private LayoutInflater inflater;

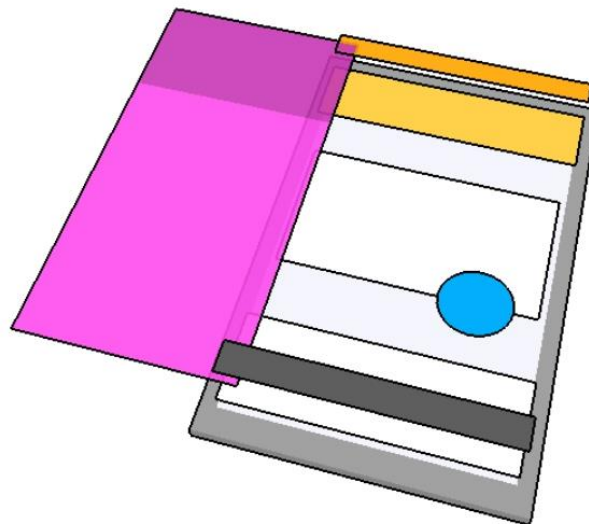
    public ListViewAdapter(Context context,List<String> data){
        Inflater=LayoutInflater.from(context);
        this.data=data;
    }
    @Override
    public int getCount(){
        return data.size();
    }
    @Override
    public Object getItem(int position){
        return data.get(position);
    }
    @Override
    public long getItemId(int position){
        return position;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent){
        ViewHolder viewHolder;
        if(convertView==null){
            viewHolder=new ViewHolder();
            convertView = inflater.inflate(android.R.layout.simple_list_item_1,
parent, false);
            viewHolder.text1 = (TextView) convertView.findViewById(an-
droid.R.id.text1);
            convertView.setTag(viewHolder);
        }else{
            viewHolder=(ViewHolder) convertView.getTag():
        }
        viewHolder.text1.setText(data.get(position));
        return convertView;
    }
    private class ViewHolder{
        private TextView text1;
    }
}
```

#### Code Snippet 5. ListViewAdapter class

When the ListView begins to draw, the system first calls the getCount() function to determine the listView's length based on the return value, and then calls getView() to draw each line one by one based on this length. The writing of this adapter is a relatively standard fixed writing so far. The

ViewHolder class is used in the getView method. This is due to ListView's RecycleBin mechanism, which reuses the ItemView while the list is scrolled. The advantage of this is that no matter how far down the list scrolls, only a screen of View can be generated.

In Android applications, more and more developers will put their menu interfaces in a list, and then users can slide to the right (or left) to see all the application features. Navigation Drawer is a panel on the left edge of the screen to display application navigation items. The navigation drawer is not visible much of the time, but it can appear in two situations: one is swiping right from the left side of the screen, and the other is clicking the application icon in the toolbar. When using a navigation drawer, DrawerLayout is used as the user interface's root view. Two subviews need to be placed under the DrawerLayout view. One is used to show the navigation drawer and the other is used to show the main content of the screen (when the navigation drawer is hidden).



**Figure 22.** Sliding Navigation Drawer /11/

In the program, the click event of the navigation item is actually the click event of different TextView including Home, History and Logout. It is needed to change the main content according to the clicked items. The view that

displays the main content will generally be an Activity at runtime. Therefore just need to switch the current Activity to the corresponding Activity to achieve the purpose of page changing.

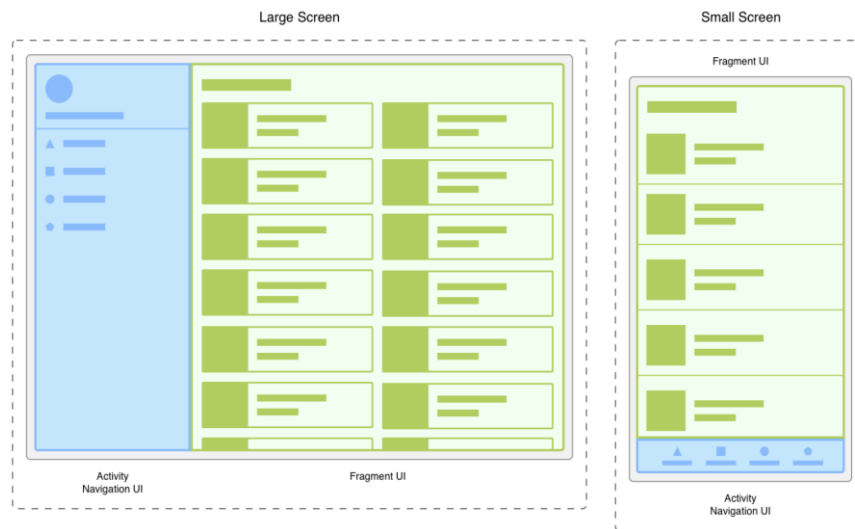
### **4.3.3 Fragment**

Fragment is an Android 3.0 API that is mostly visible in the more dynamic and versatile UI architecture of large screen devices (such as tablet). Within an activity, a fragment describes an action or a modular portion of the user interface.

Fragment has its own lifecycle and layout management capabilities. It cannot exist on its own; it needs to be hosted by an activity or another fragment. Many fragments may be combined in a single operation to create a multi-pane user interface with the following benefits:



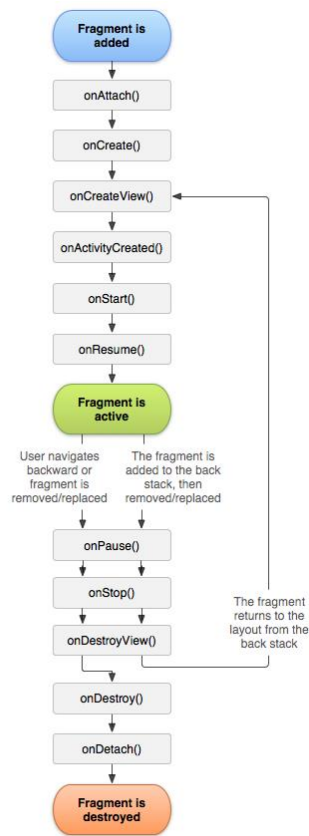
- Deal with UI issues that appear on various screens, such as the mobile phone and tablet adaptation issue shown in Figure 23.



**Figure 23.** Two separate versions of the same screen, each with a different screen size /12/

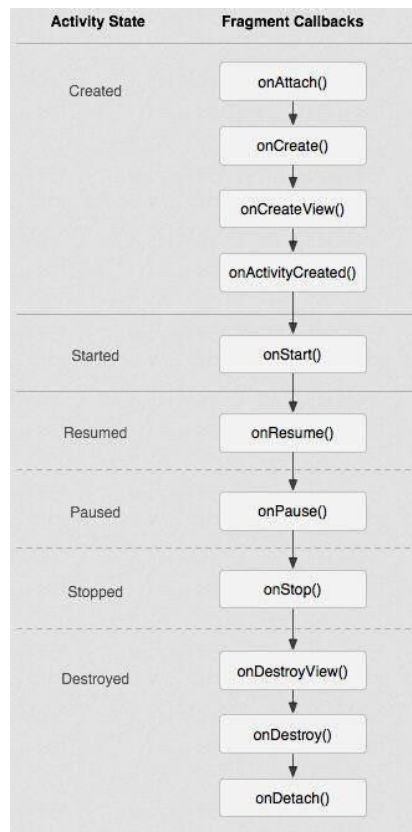
- Make activity more modular, and many business logics can be processed in the corresponding fragment, requiring only the activity reveal and hide the fragment.
- Different activities may make use of fragment. An activity can, of course, load multiple fragments.

Figure 24 depicts the fragment lifecycle. Readers who are familiar with activity components will note that the fragment and activity have a lot in common in terms of lifecycle.



**Figure 24.** The lifecycle of Fragment /12/

Figure 25 shows another picture that compares fragments to activity. The activity State method is on the left, and the lifecycle callback methods of fragment is on the right.



**Figure 25.** Comparison of Fragment and Activity lifecycle

As shown in Figure 25, fragment has the following methods more than activity:

`onAttach()`: When a fragment is first attached, the managing activity calls it.

`onCreateView()`: `onCreate()` is called, followed by `onCreateView()`. `onCreateView()` is where you set up your user interface.

`onActivityCreated()`: the `onCreate` method of the activity bound to fragment has been executed.

`onDestroyView()`: destroy views related to fragment.

`onDetach()`: unbind activity.

In an activity, `FragmentManager` is used to handle fragments. It is possible to use a fragment in an activity to obtain a `FragmentManager` event. `getFragmentManager()` is a method that returns a fragment manager.

The `getSupportFragmentManager()` method is used to get the corresponding `FragmentManager` while using the support extension package. It should be remembered that the fragment in the SDK and the fragment in the support extension package are two separate classes, as are these two `FragmentManagers`. They cannot be mixed together.

If the prompt parameter types do not fit when used, search to see if the import fragment and the fragment manager are in the same package. They can be modified to the same package path if they are not compatible.

`FragmentTransaction` has direct access to `Fragment`. The `FragmentManager` often calls the `beginTransaction()` method to start a transaction. They are normally accompanied by a sequence of acts, such as adding, replacing, or doing something else. The `FragmentTransaction.commit()` method must be called after the operation on `Fragment` is completed to commit the transaction. /13/

```
fm.beginTransaction()  
    .add(R.id.fragment_list_container, mFragment  
    .commit()
```

#### **Code Snippet 6.** `FragmentTransaction`

The common methods used by `FragmentTransaction` are listed as below:

- `add()`: Add a `Fragment` to Activity.
- `remove()`: Remove a `Fragment` from Activity.
- `replace()`: Replaces the current `Fragment` by the new `Fragment`.
- `hide()`: Hide `Fragment`.
- `show()`: Display `Fragment`.
- `commit()`: Commit transaction.

Communication between Fragment and Activity takes place as follows:

An internal callback interface is defined in the fragment, then the activity which contains the fragment implements it so that the fragment can call the callback method to send data to the activity. To help readers understand the communication between Fragment and Activity, Code Snippets 7 and 8 have been attached.

```
public class MainFragment extends Fragment{
    public FragmentListener mListener;

    //MainFragment public interface
    public static interface FragmentListener{

        //jump to page 5
        void toH5Page();

        //display message
        void showMsg(String str);
    }
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        //
        if(activity instanceof FragmentListener){

            mListener = ((FragmentListener)activity);

        }
    }
    ...

    mButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            msgListener.showMsg("Hello pass data to Activity to display.");
        }
    });
}
```

**Code Snippet 7.** MainFragment class

```
...
public class MainActivity extends FragmentActivity implements
FragmentListener{
```

```
@override
public void toH5Page(){...}

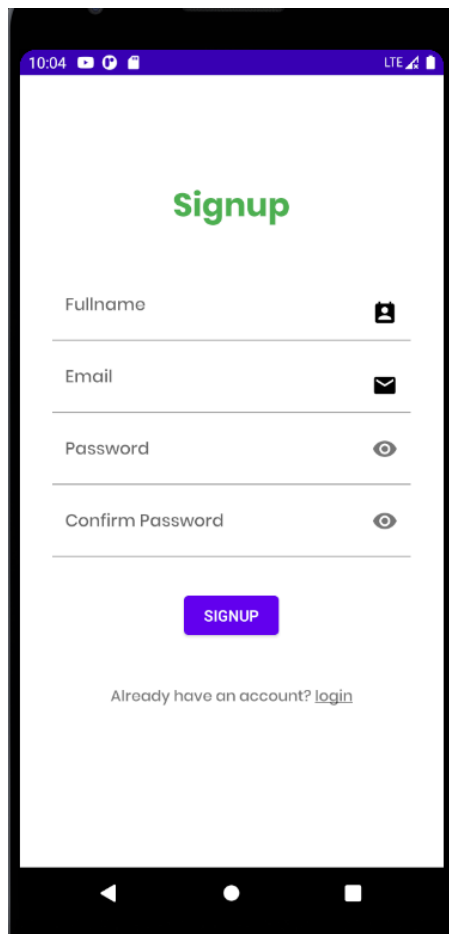
@override
public void showMsg(String str){...}
    Toast.makeText(MainActivity.this, str, Toast.Lenfth_SHORT).show();
}
}
```

**Code Snippet 8.** MainActivity class

#### 4.3.4 Graphical User Interface Design

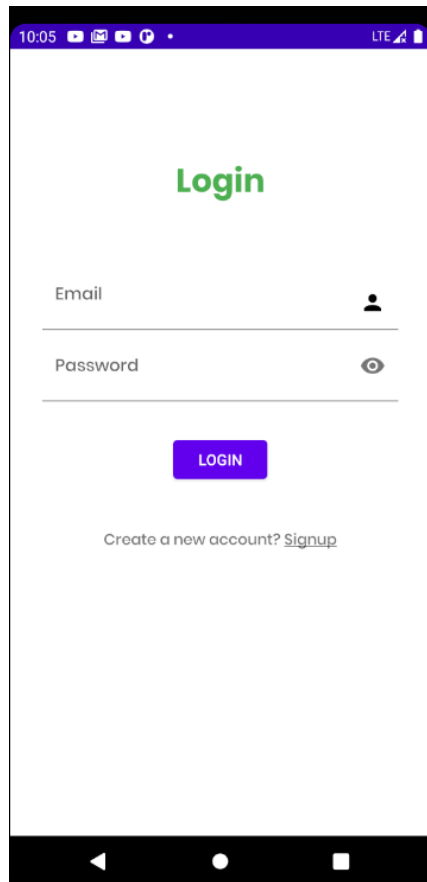
The user interfaces of this Android application are designed using XML which is a markup language for front-end design. All XML files are stored in the layout directory of this application package, there are five pages of this program including register page, login page, home page, edit category page and history page.

The user register page allows users to register an account for this program by providing user's personal information including full name, email address, password and confirm password. After entering all the information, the data will be sent to database by clicking SIGNUP button. Besides, there is another link below the button which can take the user to the Login page if the user already has an account. Figure 26 illustrates the user register page.



**Figure 26.** Register Page

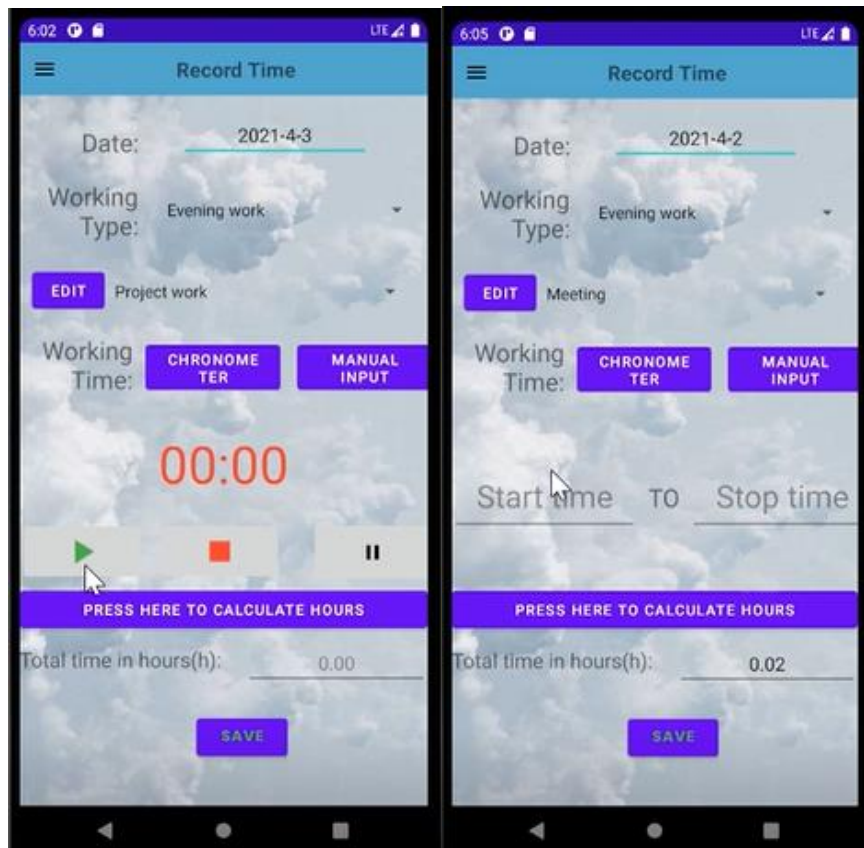
The login page allows users to fill in the user's information with email address and password. After entering the user's information and clicking the LOGIN button, the system will check whether the offered data match to the user's saved data in the database. If it is correct, the user will be taken to the home page immediately. There is another link below the LOGIN button which will take user to the register page in case the user does not have registered. Figure 27 below illustrates the login page.



**Figure 27.** Login Page

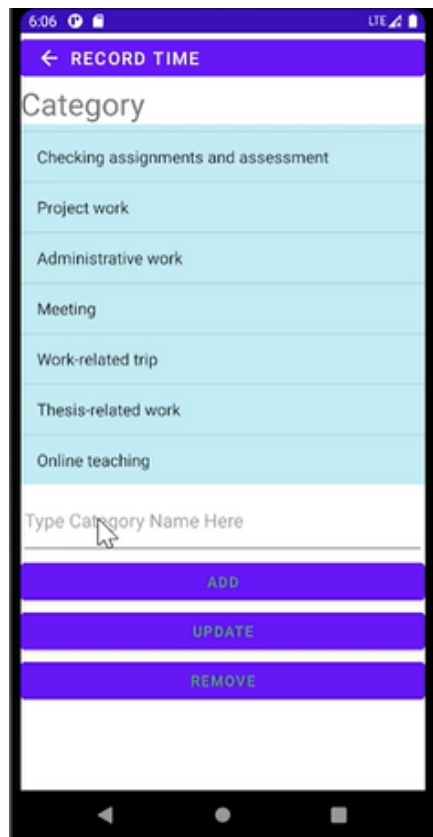
The user home page allows users to operate chronometer or input start time and end time manually with other data for instance date and working types together to record working information. There are two methods that the user can use to record working hours by clicking corresponding buttons without switching to another page. The technology behind it is using `FragmentManager` to replace the first chronometer fragment with manual input fragment and vice versa. Above the total time, there is a button which must be clicked after the user have finished recording time. As the result the value of working hours in each fragment will be sent to the home activity and will be inserted into MySQL database with other required data in this page after clicking the `SAVE` button. Figure 28 below illustrates the teacher home page.





**Figure 28.** Home Page

The edit category page allows users to add or delete working categories to the display list as they want by entering category name and clicking corresponding button. This page contains a ListView which is used to displaying all the working categories one by one on each row and it has capability to scroll down. Users can also come back to the home page by clicking button on the top of the page after finishing the editing. The spinner on the home page will receive the newest data list and update immediately. Figure 29 below illustrates edit category page.




**Figure 29.** Edit Category Page

Finally, the history page allows users to view the qualified records based on the selected working categories and given periods. These records will be displayed on the ListView below the SET button. Furthermore, the program will automatically calculate total hours of every working hour in the list and display the value of it below the ListView. For an addition features, users can send the qualified records to their email writing with HTML language. Figures 30 and 31 illustrate history page and received email with the same record list.



Figure 30. History Page

**Work** ☆  
 From: XX官方 <282065023@qq.com>   
 Date: Sunday, Apr 4, 2021 10:56 PM  
 To: Crush <386443153@qq.com>

Date	Working Type1	Working Type2	Working Hours
2021-04-04	Zoom Teaching	Time and place bound	2.19
2021-03-30	Zoom Teaching	Time and place bound	2.30
		Total:	4.49

Figure 31. Email with result table

## **4.4 Data Storage Design**

Often when developing an application on product level we need to store those data on the mobile device locally, including user details after logging in. The advantage is that the user details can be accessed if the mobile phone is not running the application. Local data management in Android primarily comprises SharedPreferences, SQLite database, file storage, ContentProvider storage data and network storage data. /14/

Different functional requirements should be based on the characteristics of storage methods to design different storage solutions, so as to optimize use of storage space and data resources to achieve the storage requirements of the system under the premise of rational use of system hardware resources.

### **4.4.1 SharedPreferences**

SharedPreferences is stored in an XML file as a key-value pair. Other applications do not have operation permission in usual conditions, so it is relatively safe. When users uninstall an application or clear application data in the device settings, they can, of course, delete SharedPreferences file. SharedPreferences can store some simple data, such as user information after login.

Compared with SQLite database, the SharedPreferences object removes certain operations, such as creating database, creating table and writing an SQL statement, which is simpler and straightforward. However, only five simple data types can be stored such as String, int, Boolean, float, long and no conditional queries can be made. /15/

### **4.4.2 SQLite database**

SQLite is a lightweight database that uses relatively few storages. In embedded systems, a few hundred kilobytes of memory can be sufficient. /16/

It supports windows, Linux, Unix, and other popular operating systems, and it can be used with a variety of programming languages, including C#, PHP, Java, and so on.

SQLite has the following advantages:

- 1) Lightweight: SQLite is separate from C / S mode database software. There is no database client or server because it is an in-process database engine. It have only one of its dynamic libraries to carry using SQLite to experience all its features.
- 2) No need to "Install": The main engine of SQLite itself does not rely on any applications from third parties, and it does not need to "Install" to use it, which is a bit like green software.
- 3) Single file: all the information in the database (such as tables, views, etc.) are contained in one file. This file can be copied freely to other directories or machines. It is convenient to transfer data between different activities or even different applications.

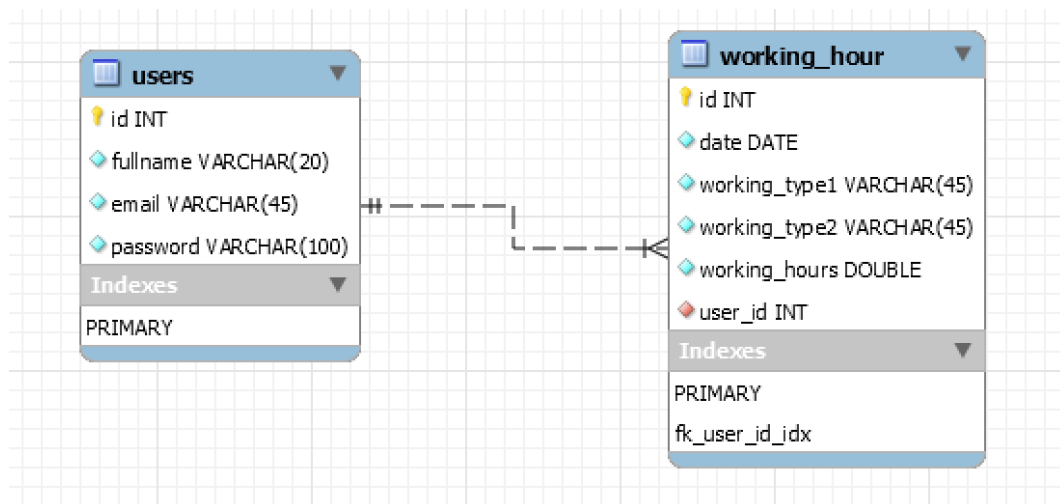
The application of SQLite database in this project is to store the working types of teachers, so as to facilitate the addition and deletion of working categories in the future.

#### **4.4.3 MySQL database**

MySQL Workbench is the design tool for the MySQL database. Database architects, program developers, and system planners may use MySQL Workbench to visualize SQL development, database modeling, and database management.

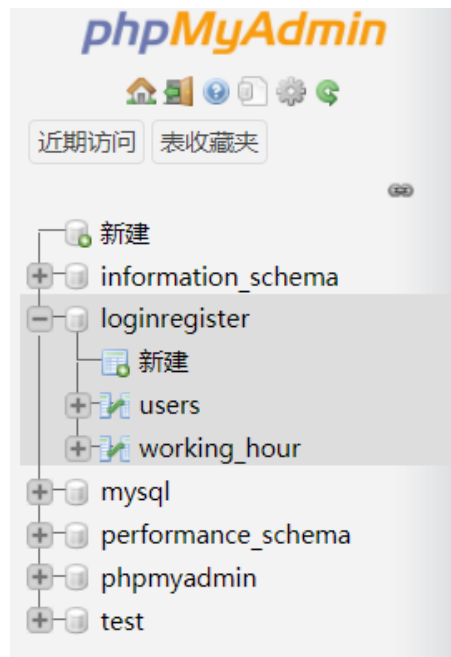
The ER diagram is used to describe a conceptual schema of static data structures. It will combine three basic concepts for example entities, relationships and attributes to summarize the basic structure of data. There are three types of relationships between entities for instance one-to-one, one-to-many and many-to-many. /17/

In this project, two database tables are needed to store user registration information and records of user working hours separately. Figure 32 below illustrates that users table with primary key ID has one-to-many relationship with table working\_hour with foreign key user\_id, which is the same with ID in users table which means that one user could have zero to many records of working hours.



**Figure 32.** ER diagram

After forwarding the ER diagram to the MySQL database in server side. Figure 33 below illustrates these two tables in loginregister database which is stored in MySQL database management system—phpMyAdmin.



**Figure 33.** loginregister database

## 5 IMPLEMENTATION

This chapter will clarify how to transform the system requirements analysis and feature design from the previous chapters into a practical project. Firstly, the main function modules of Android client are introduced, including registering and login, adding work cases, editing work category, viewing history records and sending e-mail. Then the navigation drawer in the system implementation process is chosen to be explained.

The development environment of Android client of mobile tracking system is shown in Table 3. Explanation of proper nouns: IDE is the integrated development environment, JDK is the software development kit of Java language, and Android SDK is the Android Software Development Kit.

**Table 3.** Development environment of Android client

Operating system	Development language	IDE	Android SDK
Window 10	Java	Android Studio	10.0 version

### 5.1 Implementation of Registration

First, in Code Snippet 9, a `StringRequest` object is created to submit an HTTP POST request to the URL link and Volley will obtain the POST parameters including full name, email address and password by calling the `getParams()` method in `Request`. There are two internal interfaces, `Listener` and `ErrorListener`, which can represent the callback after successful request and failed request respectively. The `JSONObject` object will take response for receiving HTTP response from the callback of `onResponse()` method in JSON format. If the value of 'sucess' defined as "success" value in `JSONObject` response is equals to 1 which means the result of insert three



parameters into the database which was written in register.php file is successful as seen in Code Snippet 10, the user will receive the confirm message “Register Success!” and move to the login page immediately. Finally, a RequestQueue object is created to cache all HTTP requests and then submit these requests simultaneously. And the StringRequest object will be added to RequestQueue.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST,
    URL_REGIST, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try {
                JSONObject jsonObject = new JSONObject(response);
                String success = jsonObject.getString("success");
                if (success.equals("1")) {
                    Toast.makeText(getApplicationContext(), "Register Success!",
                        Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                    startActivity(intent);
                }
            } catch (JSONException e) {
                e.printStackTrace();
                Toast.makeText(getApplicationContext(), "Register Error!" +
                    e.toString(), Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.GONE);
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(getApplicationContext(), "Register Error!" +
                    error.toString(), Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.GONE);
            }
        }
    ){
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("fullname", fullname);
            params.put("email", email);
            params.put("password", password);
            return params;
        }
    };
RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
```

```
requestQueue.add(stringRequest);
```

### Code Snippet 9. StringRequest object in SignUpActivity

```
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST'){
    $fullname = $_POST['fullname'];
    $email = $_POST['email'];
    $password = $_POST['password'];
    $password = password_hash($password, PASSWORD_DEFAULT);
    require_once 'connect.php';

    $sql = "INSERT INTO users (fullname, email, password) VALUES ('$fullname', '$email', '$password')";

    if ( mysqli_query($conn, $sql) ) {
        $result["success"] = "1";
        $result["message"] = "success";

        echo json_encode($result);
        mysqli_close($conn);
    } else {
        $result["success"] = "0";
        $result["message"] = "error";

        echo json_encode($result);
    }
    mysqli_close($conn);
}
?>
```

### Code Snippet 10. register.php file

## 5.2 Implementation of Login

The implementation of login is quite similar with registration with difference in POST parameters and one more JSONArray object. The Volley will obtain the POST parameters including email address and password by calling the `getParams()` method in Request. In `onResponse()` method, JSONArray object is created to represent the “login” value in JSONObject object which

receives the HTTP response from server side. If the provided data matched with value in MySQL database the user will also get a confirm message “Login Success!” and be taken to the home page which is shown in Code Snippet 11. In addition, user’s id number and email address will be stored using SharedPreferences after login successfully everytime in order to distinguish users with their user id in adding work cases and acquiesce in the email is the receiving email in the future operation. In details, the user’s id and email are obtained from JSONObject object when the system receives the HTTP response from the server side and then the value is saved in the form of XML files in the system by using SharedPreferences. When these data are to be used later, the system can fetch the value from SharedPreferences.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST,
        URL_LOGIN, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response);
                    String success = jsonObject.getString("success");
                    JSONArray jsonArray = jsonObject.getJSONArray("login");
                    if (success.equals("1")) {
                        for (int i = 0; i < jsonArray.length(); i++) {
                            JSONObject object = jsonObject.getJSONObject(i);
                            String name = object.getString("fullname");
                            String email = object.getString("email");
                            int id = object.getInt("id");
                            String id1 = String.valueOf(id);
                            Toast.makeText(getApplicationContext(), "Login Success!",
                                Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                            startActivity(intent);
                            SharedPreferences sharedPreferences = getSharedPreferences("user", Context.MODE_PRIVATE);
                            SharedPreferences.Editor editor = sharedPreferences.edit();
                            editor.putString("userid", id1);
                            editor.putString("useremail", email);
                            editor.commit();
                            progressBar.setVisibility(View.GONE);
                        }
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        });
```

```

        progressBar.setVisibility(View.GONE);
        Toast.makeText(getApplicationContext(), "Error" + e.toString(),
Toast.LENGTH_SHORT).show();
    }
}
},
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            progressBar.setVisibility(View.GONE);
            Toast.makeText(getApplicationContext(), "Error" + er-
ror.toString(), Toast.LENGTH_SHORT).show();
        }
    }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("email", email);
            params.put("password", password);
            return params;
        }
    };
    RequestQueue requestQueue = Volley.newRequestQueue(getApplication-
Context());
    requestQueue.add(stringRequest);

```

**Code Snippet 11.** StringRequest object in LoginActivity

```

<?php

if ($_SERVER['REQUEST_METHOD']=='POST') {

    $email = $_POST['email'];
    $password = $_POST['password'];

    require_once 'connect.php';

    $sql = "SELECT * FROM users WHERE email='$email' ";
    $response = mysqli_query($conn, $sql);
    $result = array();
    $result['login'] = array();

    if ( mysqli_num_rows($response) === 1 ) {
        $row = mysqli_fetch_assoc($response);
        if ( password_verify($password, $row['password']) ) {

            $index['fullname'] = $row['fullname'];
            $index['email'] = $row['email'];
            $index['id'] = $row['id'];

```

```

        array_push($result['login'], $index);

        $result['success'] = "1";
        $result['message'] = "success";
        echo json_encode($result);

        mysqli_close($conn);
    } else {

        $result['success'] = "0";
        $result['message'] = "error";
        echo json_encode($result);

        mysqli_close($conn);
    }
}
?>

```

**Code Snippet 12.** Login.php file

### 5.3 Implementation of Home Page

In the home page, user can enter the date, select two working types from spinners and operate the chronometer or input time manually. These are inserted with user's id number to the working\_hour table in loginregister database of MySQL. It is needed to use Handler and post a Runnable then put the PutData code within the run method. To begin, two arrays are created, one for the parameter's field name and the other for the data. These arrays, the URL and the request method are passed as arguments when creating the PutData object. When calling startFetch() to start the process, it will get a boolean value back. Using onComplete(), which returns a boolean value, to determine when the process is finished and call getResult() to obtain the result value as seen in Code Snippet 13.

```

Handler handler = new Handler(Looper.getMainLooper());
handler.post(new Runnable() {
    @Override
    public void run() {
        String[] field = new String[5];
        field[0] = "date";
        field[1] = "working_type1";
        field[2] = "working_type2";
        field[3] = "working_hours";
    }
});

```

```

        field[4] = "user_id";
        //Creating array for data
        String[] data = new String[5];
        data[0] = WorkhourdateHolder;
        data[1] = Workhourtype1Holder;
        data[2] = Workhourtype2Holder;
        data[3] = WorkhourhoursHolder;
        data[4] = WorkhouruseridHolder;
        PutData putData = new PutData(URL_Work,"POST", field, data);
        if (putData.startPut()) {
            if (putData.onComplete()) {
                String result = putData.getResult();
                if (result.equals("Data added Success")) {
                    Toast.makeText(getApplicationContext(), result,
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(getApplicationContext(), result,
Toast.LENGTH_SHORT).show();
                }
                //End ProgressBar (Set visibility to GONE)
            }
        }
        //End Write and Read data with URL
    }
});

```

### Code Snippet 13. Handler object in HomeActivity

```

<?php
require "DBAdapter.php";

class DataBase
{
    public $connect;
    public $data;
    private $sql;
    protected $servername;
    protected $username;
    protected $password;
    protected $databasename;

    public function __construct()
    {
        $this->connect = null;
        $this->data = null;
        $this->sql = null;
        $dbc = new DataBaseConfig();
        $this->servername = $dbc->servername;
        $this->username = $dbc->username;
        $this->password = $dbc->password;
        $this->databasename = $dbc->databasename;
    }
}

```

```

function dbConnect()
{
    $this->connect = mysqli_connect($this->servername, $this->username, $this->password,
    $this->databasename);
    return $this->connect;
}

function prepareData($data)
{
    return mysqli_real_escape_string($this->connect, stripslashes(htmlspecialchars($data)));
}

function dataadd($stable, $date, $working_type1, $working_type2, $working_hours,
$user_id)
{
    $date = $this->prepareData($date);
    $working_type1 = $this->prepareData($working_type1);
    $working_type2 = $this->prepareData($working_type2);
    $working_hours = $this->prepareData($working_hours);
    $user_id = $this->prepareData($user_id);
    $this->sql = "INSERT INTO " . $stable . " (date, working_type1, working_type2, work-
ing_hours, user_id) VALUES (" . $date . ", " . $working_type1 . ", " . $working_type2 . ", " .
$working_hours . ", " . $user_id . ")";
    if (mysqli_query($this->connect, $this->sql)) {
        return true;
    } else return false;
}
}
?>

```

#### Code Snippet 14. addworkcase.php file

In Code Snippet 15, the ChangeFragment() method is used to replace the current fragment with the other fragment in FrameLayout by clicking the CHRONOMETER button or MANUAL INPUT button. And in the xml file, the onclick property of buttons should be defined as “ChangeFragment” also.

```

public void ChangeFragment(View view) {
    Fragment fragment;

    if (view == findViewById(R.id.button_fragment1)) {
        fragment = new FragmentOne();
        FragmentManager fm = getFragmentManager();
        FragmentTransaction ft = fm.beginTransaction();
        ft.replace(R.id.fragment_place, fragment);
        ft.commit();
    }

    if (view == findViewById(R.id.button_fragment2)) {
        fragment = new FragmentTwo();
    }
}

```

```

        FragmentManager fm = getFragmentManager();
        FragmentTransaction ft = fm.beginTransaction();
        ft.replace(R.id.fragment_place, fragment);
        ft.commit();
    }
}

```

**Code Snippet 15.** ChangeFragment method in MainActivity

## 5.4 Implementation of Edit Page

The SQLiteHelper class inherits the SQLiteOpenHelper class and overrides onCreate() and onUpgrade() methods. In more details, the CATEGORIES table in TEST.db is created in onCreate() method with two columns which are integer id as primary key and text category respectively. Next, the insert\_category() and delete\_category() methods are used to add and delete category value from the CATEGORIES table with the same parameter which can be obtained from user entering in edit page. The last list\_all\_categories\_list() method is used to search all the data in CATEGORIES table to be included into ListView because the result will show in it. In order to read records from the table, the Cursor class is used. The method.rawQuery() is called to execute the select query for selecting all the categories. Then, the moveToNext() method is used to go for each raw which means the cursor will go from the first raw of the result to the end raw of the result. Furthermore, the result data will be mapped to ListViewq through adapter as seen in Code Snippet 16. All these methods can be called in EditActivity.java, so the client can CRUD the TEST database easily.

```

public class SQLiteHelper extends SQLiteOpenHelper {
    private ArrayAdapter<Category> adapter;

    public SQLiteHelper(@Nullable Context context, @Nullable String name,
        @Nullable SQLiteDatabase.CursorFactory factory, int version) {
        super(context, "TEST.db", factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE CATEGORIES(ID INTEGER PRIMARY KEY AUTOINCREMENT,
CATEGORY TEXT);");
    }
    @Override

```



```

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS CATEGORIES;");
    onCreate(db);
}
public void insert_category(String category) {
    ContentValues contentValues = new ContentValues();
    contentValues.put("CATEGORY", category);
    this.getWritableDatabase().insertOrThrow("CATEGORIES", "", contentValues);
}
public void delete_category(String category) {
    this.getWritableDatabase().delete("CATEGORIES", "CATEGORY='" + category + "'", null);
}
public void list_all_categories_list(Context context, ListView listview) {
    Cursor cursor = this.getReadableDatabase().rawQuery("SELECT * FROM CATEGORIES",
null);
    final ArrayList<Category> categories = new ArrayList<>();
    Category c;

    while (cursor.moveToNext()) {
        c = new Category();
        c.setId(cursor.getInt(0));
        c.setCategory(cursor.getString(1));
        categories.add(c);
    }
    adapter = new ArrayAdapter(context, android.R.layout.simple_list_item_1, categories);
    listview.setAdapter(adapter);
}
}

```

**Code Snippet 16.** SQLiteHelper class

## 5.5 Implementation of History Page

It is needed to get user id number which was stored by SharedPreferences to be post with other parameters using Volley framework. First, the SharedPreferences object is created and the getString() method is called to obtain the user\_id value which was stored when user logging in. Then the WorkinghourAdapter object is created in WorkinghourAdapter.java to set the ArrayAdapter to the ListView. After that, the StringRequest object is created to submit an HTTP POST request with five parameters which are obtained from user selected date and working categories with user's id as shown is Code Snippet 17. In the onResponse() method, the JSONArray object is used to receive HTTP response and JSONObject will parse JSONArray. The corresponding values including date, work\_type1, work\_type2

and work\_hours will be mapped in to the ListView through getView() method in WorkinghourAdapter class which inherits the ArrayAdapter class as shown in Code Snippet 18.

```

SharedPreferences sharedPreferences = getSharedPreferences("user", Context.MODE_PRIVATE);
    user_id = sharedPreferences.getString("userid", null);
    WorkinghourAdapter workinghourAdapter = new WorkinghourAdapter(ReportActivity.this, R.layout.list_item);
    listView.setAdapter(workinghourAdapter);
    StringRequest stringRequest = new StringRequest(Request.Method.POST,
        DATAURL, new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            try {
                double total = 0;
                JSONArray jsonArray = new JSONArray(response);
                for (int i = 0; i < jsonArray.length(); i++) {
                    JSONObject object = jsonArray.getJSONObject(i);
                    String date = object.getString("date");
                    String work_type1 = object.getString("working_type1");
                    String work_type2 = object.getString("working_type2");
                    double work_hours = object.getDouble("working_hours");
                    total += work_hours;
                    Workinghour wh = new Workinghour(date, work_type1, work_type2,
work_hours);
                    workinghourAdapter.add(wh);
                }
                textViewtotalhour.setText(String.valueOf(total));
            } catch (JSONException e) {
                Toast.makeText(getApplicationContext(), "Error" + e.toString(),
Toast.LENGTH_SHORT).show();
            }
        }
    },
    new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            // progressBar.setVisibility(View.GONE);
            Toast.makeText(getApplicationContext(), "Error" + error.toString(),
Toast.LENGTH_SHORT).show();
        }
    }) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        Map<String, String> params = new HashMap<>();
        params.put("worktype1", worktype1);
        params.put("worktype2", worktype2);
        params.put("startdate", startdate);
        params.put("enddate", enddate);
        params.put("user_id", user_id);
    }
}

```

```

        return params;
    }
};
RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
requestQueue.add(stringRequest);

```

### Code Snippet 17. StringRequest object of searching history in MainActivity

```

public class WorkinghourAdapter extends ArrayAdapter {
    List list = new ArrayList();
    public WorkinghourAdapter(@NonNull Context context, int resource) {
        super(context, resource);
    }
    public void add(Workinghour object) {
        super.add(object);
        list.add(object);
    }
    @Override
    public int getCount() {
        return list.size();
    }
    @Override
    public Object getItem(int position) {
        return list.get(position);
    }
    @Override
    public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
        View row;
        row = convertView;
        WorkhourHolder workHolder;
        if (row == null) {
            LayoutInflater inflater = (LayoutInflater) this.getContext().getSystemService
                (Context.LAYOUT_INFLATER_SERVICE);
            row = inflater.inflate(R.layout.list_item, parent, false);
            workHolder = new WorkhourHolder();
            workHolder.tx_date = row.findViewById(R.id.tvdate);
            workHolder.tx_wt1 = row.findViewById(R.id.tvwt1);
            workHolder.tx_wt2 = row.findViewById(R.id.tvwt2);
            workHolder.tx_wh = row.findViewById(R.id.tvwhs);
            row.setTag(workHolder);
        } else {
            workHolder = (WorkhourHolder) row.getTag();
        }
        Workinghour workinghour = (Workinghour) this.getItem(position);
        workHolder.tx_date.setText(workinghour.getDate());
        workHolder.tx_wt1.setText(workinghour.getWorktype1());
        workHolder.tx_wt2.setText(workinghour.getWorktype2());
        workHolder.tx_wh.setText(String.valueOf(workinghour.getWorkhour()));
        return row;
    }
}

```

```

static class WorkhourHolder {
    TextView tx_date, tx_wt1, tx_wt2, tx_wh;
}
}

```

### Code Snippet 18. WorkinghourAdapter class

A Golang script serves as intermediary between Android client side and MySQL database. The posted parameters will be used as condition value of select query and the query result will be written into JSON format then will be sent back to the client side in HTTP response as shown in Code Snippet 19.

```

http.HandleFunc("/hello", func(w http.ResponseWriter, r *http.Request) {
    r.ParseForm()
    worktype1:=r.Form["worktype1"]
    worktype2:=r.Form["worktype2"]
    startdate:=r.Form["startdate"]
    enddate:=r.Form["enddate"]
    user_id:=r.Form["user_id"]

    sql:=fmt.Sprintf(" select * from working_hour where date between '%v'
and '%v' and working_type1='%v' and working_type2='%v' and user_id=%v order by
date",startdate,enddate,worktype1,worktype2,user_id)

    sql=strings.ReplaceAll(sql,"[", "")
    sql=strings.ReplaceAll(sql,"]", "")
    fmt.Println(sql)
    var workingHours []WorkingHour
    err:=Db.Select(&workingHours,sql)
    if err != nil {
        fmt.Println("exec failed, ", err)
        return
    }
    json,_:=json.Marshal(workingHours)
    io.WriteString(w, string(json))
})

```

### Code Snippet 19. Search history function in Golang

Code Snippet 20 shows how the StringRequest object of the Volley framework obtains the POST parameters including worktype1, worktype2, startdate, enddate, user\_id and mail by calling the getParams() method in Request. And these value will be posted to the server side which is written with Golang script.

```

SharedPreferences sharedPreferences =
    getSharedPreferences("user", Context.MODE_PRIVATE);
user_id = sharedPreferences.getString("userid", null);
user_email = sharedPreferences.getString("useremail", null);
StringRequest stringRequest1 = new StringRequest(Request.Method.POST,
    MAILURL, new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
    }
    },
    new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {

        Toast.makeText(getApplicationContext(), "Error" + error.toString(),
Toast.LENGTH_SHORT).show();
    }
    }) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
    Map<String, String> params = new HashMap<>();
    params.put("worktype1", worktype1);
    params.put("worktype2", worktype2);
    params.put("startdate", startdate);
    params.put("enddate", enddate);
    params.put("user_id", user_id);
    params.put("mail", user_email);
    return params;
    }
    };
RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
requestQueue.add(stringRequest1);

```

### Code Snippet 20. StringRequest object of send email in MainActivity

In the golang file below, the posted data will be used as condition value of select query to search out corresponding records in loginregister table and the query result will be set into the email content table which is written in HTML format. Then the server side will sent the email to user with the same record result with records in the ListView in history page.

```

http.HandleFunc("/mail", func(w http.ResponseWriter, r *http.Request){
    r.ParseForm()
    worktype1:=r.Form["worktype1"]
    worktype2:=r.Form["worktype2"]
    startdate:=r.Form["startdate"]
    enddate:=r.Form["enddate"]
    user_id:=r.Form["user_id"]
    toMail:=fmt.Sprintf("%s", r.Form["mail"] )

```

```

        sql:=fmt.Sprintf(" select * from working_hour where date between '%v'
and '%v' and working_type1='%v' and working_type2='%v'
and user_id=%v",startdate,enddate,worktype1,worktype2,user_id)

        sql=strings.ReplaceAll(sql,"[","")
        sql=strings.ReplaceAll(sql,"]","")
        toMail=strings.ReplaceAll(toMail,"[","")
        toMail=strings.ReplaceAll(toMail,"]","")
        fmt.Println(sql)
        var workingHours []WorkingHour
        err:=Db.Select(&workingHours,sql)

        mailContent:=""
        var total float32
        for _,v:=range workingHours{
            mailCon-
tent+=fmt.Sprintf("<tr><td>%s</td><td>%s</td><td>%s</td><td>%.2f</td></tr>",v.Date,v.W
orkingType1,v.WorkingType2,v.WorkingHours)
            total+=v.WorkingHours
        }
        mailContent+=fmt.Sprintf("<tr><td></td><td></td><td><td>To-
tal:</td><td>%.2f</td></tr>",total)
        mailBody:=fmt.Sprintf("<table border=\"1\" style=\"border-collapse:
collapse;\"><tr><th>Date</th><th>Working Type1</th>
<th>Working Type2</th><th>Working Hours</th></tr>%s</ta-
ble>",mailContent)
        if err != nil {
            fmt.Println("exec failed, ", err)
        }
        mailTo := []string{}
        mailTo=append(mailTo, toMail)
        SendMail(mailTo,"Work",mailBody)

        json,_:=json.Marshal(workingHours)
        io.WriteString(w, string(json))
    })

```

**Code Snippet 21.** Send email function in Golang

## 5.6 Implementation of Navigation Drawer

In Code Snippet 22, the openDrawer() method is called when user click the menu icon in home page and the slid navigation drawer will appear with Home, History and Logout options in this menu. When user click the Logout textview, there will be an AlertDialog appear with "Logout" as its title and ask the user's intention to logout. If user choose yes, the application will

close, but if the answer is no, it will stay on the original page. The `redirectToActivity()` method allows user switch to another page. If these two activities are in the same application, the `TaskId` of two activities will be the same. If not, the new `Task` will be created.

```
public static void openDrawer(DrawerLayout drawerLayout) {
    drawerLayout.openDrawer(GravityCompat.START);
}

public static void logout(final Activity activity) {
    AlertDialog.Builder builder = new AlertDialog.Builder(activity);
    builder.setTitle("Logout");
    builder.setMessage("Are you sure you want to logout?");
    builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            activity.finishAffinity();
            System.exit(0);
        }
    });
    builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    builder.show();
}

public static void redirectActivity(Activity activity, Class aClass) {
    Intent intent = new Intent(activity, aClass);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    activity.startActivity(intent);
}
```

### Code Snippet 22. Navigation methods

The following methods can be called in both `MainActivity.java` and `reportActivity.java` as the result, user can click the menu icon in both home page and history page to have slide navigation of this application.

```
public void ClickMenu(View view) {
    openDrawer(drawerLayout);
}

public void ClickHome(View view) {
    recreate();
}

public void ClickHistory(View view) {
```

```
    redirectActivity(this, ReportActivity.class);
}

public void ClickLogout(View view) {
    logout(this);
}
```

**Code Snippet 23.** onClick properties of Navigation items



## 6 TESTING

This chapter will test and analyze the system based on the system implementation. The function test will list the test cases of the system, and then give the effect of the actual implementation of the system.

Functional testing is also called black box testing. Test cases are constructed beginning from the interface and judging the difference between the real outcomes and the predicted results, without considering the system's internal implementation. Functional checking is used to ensure that the project's operations have been completed and the system's input and performance are as anticipated. /18/

This project has been tested by the client side of the mobile phone by Huawei Mate 30 and the test template and the results of the testing are given in the Table 4 (Appendix 1.).

The function test verifies that the system has implemented all the project requirement analyses' functions.

## 7 CONCLUSION

This thesis project was designed to develop a mobile work time tracking system for teachers based on the Android platform with multiple functions for instance authentication, record working hours, edit work categories, search history records and send email. Teachers can analyze and manage working hours more rationally and efficiently.

This paper expounds the complete practice process of the system according to the background of the project, related technologies and tools, requirement analysis, system design, project implementation and system testing process.

During the implementation of this program, some studies have been found out through implementing different methodologies to solve data storage and network communication issues. For instance, Android developers can make suitable storage solution according to the requirement of different business scenarios. SharedPeferences is suitable to store some simple data with Key/Value property. In fact, the most used part of SharedPreferences in Android is also used to save the configuration information in the device. The SQLite database, a standard database that comes with Android is suitable to store some lightweight information such as chatting records of social media applications. Because of its built-in feature, there is no need to construct client and server sides of database when using SQLite. As the result, the processing speed of SQLite is faster comparing with MySQL and PostgreSQL.

The most important and most time-consuming part of developing this program was building a network communication system between the client side and server side. Usually when writing the business logic of network communication, there will be a large section of HttpURLConnection logic. However, the Volley framework can simplify the writing code and realize complex com-

munication content by only a few lines of code because it encapsulates classes commonly used for network communication, such as `HttpURLConnection`. Besides, the Volley framework is suitable for scenarios where network communication is frequent but the amount of data is small and it can greatly improve the development efficiency.

In addition, when creating the interface for the server side to receive HTTP requests with posted parameters from client side, the PHP scripting language and the Go programming language were used to deal with requests of authentication and searching history records respectively. It is found that the amount of code writing with PHP is more than that writing with Golang to generate the same functionality interface. Furthermore, Golang serves more requests than almost any other language including PHP [19]. Therefore, Go is the most extensible programming language and it will grow as the business grows to accommodate the increasing load of the application effectively.

In summary, this project has realized a stable, functional and interactive mobile tracking system through the Android development technology, Volley network communication technology and mobile time recording theory practice.

## **8 FUTURE IMPROVEMENT**

The authentication function of this project can be implemented by a Firebase console, in which customers can sign in with their email and password and the registration information will be stored in Firebase instead of MySQL database. In this modern authentication method, developers can save effort without creating a server side to deal with HTTP request from the client and generating a database and table to store posted data. Firebase will help to solve authentication issue and resetting the password requirement effectively through adding Firebase Authentication to the application.

For the functional module of this program, the function of summarizing history records of work by providing a graphic analysis, such as pie chart can be added to the system. Therefore, the customers can observe their working records more intuitive based on the graphical chart in weeks, months or years.

The target customers of this mobile application can include not only the teachers but also the state employees. Depending on the position and the value of working hours offered by this program, people can be paid with relative salaries by companies also with integrity operation if this application becomes popular in companies.

## REFERENCES

/1/ Smartphone Market Share. Accessed 14.05.2021.

<https://www.idc.com/promo/smartphone-market-share>.

/2/ First View for Android Development on Apple M1 Chip Device 2021. Accessed 18.04.2021. <https://medium.com/mobile-app-development-publication/first-view-for-android-development-on-apple-m1-chip-device-e9d3d52b27aa>.

/3/ Platform Architecture. Accessed 19.04.2021. <https://developer.android.com/guide/platform>.

/4/ mysqld – The MySQL Server. Accessed 19.04.2021.

<https://dev.mysql.com/doc/refman/8.0/en/mysqld.html>

/5/ Wickham, M. 2018. Practical Android: 14 Complete Projects on Advanced Techniques and Approaches. Texas: Apress.

/6/ Donovan, A and Kernighan, B. 2015. The Go Programming Language. Boston: Addison-Wesley Professional.

/7/ Quality Function Deployment (QFD). Accessed 23.04.2021. <https://quality-one.com/qfd/>.

/8/ What is Use Case Diagram. Accessed 23.04.2021. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.

/9/ Set up RequestQueue. Accessed 25.04.2021. <https://developer.android.com/training/volley/requestqueue>.

/10/ Activity. Accessed 25.04.2021. <https://developer.android.com/reference/android/app/Activity>.

/11/ Mew, K. 2016. Android Design Patterns and Best Practice. Birmingham: Packt Publishing.

/12/ Fragments. Accessed 28.04.2021.

<https://developer.android.com/guide/fragments>.

/13/ Fragments 2021. Accessed 29.04.2021. <https://medium.com/mobile-development-group/fragments-6ff6307583a4>.

/14/ Storage Options. Accessed 05.05.2021.

<https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/data/data-storage.html>.

/15/ SharedPreferences. Accessed 07.05.2021. <https://developer.android.com/reference/android/content/SharedPreferences>.

/16/ SQLiteDatabase. Accessed 09.05.2021. <https://sqlite.org/about.html>.

/17/ What is Entity Relationship Diagram (ERD)? Accessed 09.05.2021. <https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/>.

/18/ Bångerijs, S and Fröberg, F. 2016. Functional testing of an Android application, pp. 3-5. Accessed 19.05.2021.

<http://www.diva-portal.org/smash/get/diva2:1034457/FULLTEXT01.pdf>.

/19/ Handling 1 Million Requests per Minute with Go 2015. Accessed 19.05.2021.

<http://marcio.io/2015/07/handling-1-million-requests-per-minute-with-go-lang/>

## APPENDIX 1

**Table 4.** Testing case table

S/N	Test Case Description	Steps	Expected System Response	Pass/Fail
1.	Check whether new user can create an account successfully.	<ol style="list-style-type: none"> <li>1. Access to the register page and fill in the user register information.</li> <li>2. Click the SIGNUP button.</li> </ol>	The application displays the confirm message and redirects to the login page.	Pass
2	Check whether user who already registered can register again.	<ol style="list-style-type: none"> <li>1. Access to the register page and fill in the existed register information.</li> <li>2. Click the SIGNUP button.</li> </ol>	The application displays the error message. Refresh the register page.	Pass
3	Check for empty inputs on register page.	<ol style="list-style-type: none"> <li>1. Access to the register page and leave the fields empty.</li> <li>2. Click the SIGNUP button.</li> </ol>	The application displays the error message.	Pass
4	Check for clickable login link on register page.	<ol style="list-style-type: none"> <li>1. Access to the register page and click the login link.</li> </ol>	The application redirects to the login page.	Pass
5.	Check whether user can login successfully who has an account of the application.	<ol style="list-style-type: none"> <li>1. Access to the login page and fill in the user registered information.</li> <li>2. Click LOGIN button.</li> </ol>	The application displays the confirm message and redirects to the home page.	Pass
6	Check for incorrect inputs on login page.	<ol style="list-style-type: none"> <li>1. Access to the login page and fill in the incorrect information.</li> <li>2. Click LOGIN button.</li> </ol>	The application displays the error message. Refresh the login page.	Pass
7	Check for empty inputs on register page.	<ol style="list-style-type: none"> <li>1. Access to the login page and leave the fields empty.</li> <li>2. Click the LOGIN button.</li> </ol>	The application displays the error message.	Pass

8	Check for clickable sign-up link on login page.	<ol style="list-style-type: none"> <li>1. Access to the login page and click the signup link.</li> </ol>	The application redirects to the register page.	Pass
9	Check whether can log-out.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Click Logout option in navigation menu.</li> <li>3. Click YES when the remind dialog appear.</li> </ol>	The application exits.	Pass
10	Add work case.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Select date, work type1, work type2, and click start button of chronometer when the work starts and click stop button when the work ends or input the start time and stop time manually.</li> <li>3. Click the convert button to calculate total hours.</li> <li>4. Click SAVE button to insert all value to the MySQL database.</li> </ol>	The application displays the confirm message and the working_hour table in loginregister database has the new record stored in.	Pass
11	Check for empty inputs on home page.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Leave the required fields empty and click SAVE button.</li> </ol>	The application displays error message.	Pass
12	Edit work category.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Click Edit button next to the spinner.</li> <li>3. Access to the edit page.</li> </ol>	The application displays the confirm message after user adding or deleting the work category and displays the new data array in the list. Also the spinner on the	Pass



		<ol style="list-style-type: none"> <li>4. Enter the category name and click ADD or REMOVE button.</li> <li>5. Click UPDATE button to view new value list.</li> </ol>	home page displays the newest data list.	
13	View history records.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Click History in navigation drawer and access to the history page.</li> <li>3. Choose two work types from spinners and enter start date and end date, then click SET button.</li> </ol>	The result records display on the ListView and the sum of working hours for all records shows in the end of the list.	Pass
14	Check for nonexistent records.	<ol style="list-style-type: none"> <li>1. Login to the application and access to the home page.</li> <li>2. Click History in navigation drawer and access to the history page.</li> <li>3. Choose two work types and enter period which do not have corresponding records in database.</li> </ol>	The application displays error message.	Pass
15	Send record result to user's email.	<ol style="list-style-type: none"> <li>1. Repeat the operation of No.13 and click SEND EMAIL button.</li> </ol>	User will receive the email with table of the same record result shown on history page.	Pass