THE UNIVERSITY OF HONG KONG

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Project 2017-2018

Mobile Web Application - Electronic Payment System

Final Report

Supervisor: Dr. W. H. Lam

Name: ZHU Zicong

UID: 3035142132

Submission Date: 2018/3/5

Curriculum: BEng (Computer Engineering)

Summary

Since the last few decades, the smartphone has become necessary for everyone in the daily life, instead of just tools for communications. Due to the rapid development of the mobile system, the mobile applications made people's life more convenient such as map applications and delivering applications. Among the vast quantities of mobile apps, electronic payment applications have become an essential part of the application family. It has become significant for developing and improving electronic payment applications with great qualities.

In this project, a mobile web application for electronic payment will be designed and implemented. This electronic payment application will be focusing on providing users with convenient and safe electronic payment experiences. Multiple kinds of functions will be performed in this application including QR code and face recognition, which have been widely applied in mobile application fields to merely the procedures of payment.

In the design and development of this project, a RESTful structure will be used with android front-end application, Java back-end server, MySQL database and Google Cloud Message pushing service. Furthermore, face recognition parts will be implemented by Tensorflow in Artificial Neural Network field, a method called Eigenface will be referenced.

Convenient and safe electronic payment systems will be developed as the goal of this project. Compared to other existing applications, it is focusing on brand-new system architecture, better payment experiences among users, especially the implementation of face identification payment method. However, there's quite a lot limitation in the final product, like safety method and capacity of the application, which shall be improved for further considerations.

Acknowledgment

Until the current stage of the final report, the project has been fully implemented. Therefore, while writing this report, first, I wish to express my most profound gratitude to my project supervisor, Dr. Lam, who has given me valuable suggestions and advice. Then I would like to show my sincere thanks to the Assistant Lecturer from CAES, Mr. Cezar Cazan, who has taught me with knowledges on technical report writing.

Table of Content

List of Figures & Tables

List of Symbols

App	Application
GCM	Google Clouding Messaging
DFD	Data Flow Diagram
SQL	Structured Query Language
JSON	JavaScript Object Notation
POS	Point of Sale
FYP	Free Yeah Payment
UI	User Interface
IP	Internet Protocol
MAC	Medium Access Control
JDBC	Java DataBase Connectivity
REST	REpresentational State Transfer
URL	Uniform Resource Locator

1. Introduction

1.1. Background

The electronic payment is defined as "Users send payment orders to bank systems directly or indirectly via electronic devices, to achieve currency payment and fund transfer." However, in the recent decades, while the smart mobile phones are being developed rapidly, a significant number of mobile applications for electronic payment are carried out like Alipay, WeChat Pay, and PayPal, which are designed as professional commercial applications and shows substantial influences on users' daily lives.

1.2. Objective

In this project named "Mobile Web Application - Mobile Electronic Payment System," an Android mobile application called "Free Yeah Payment" (denoted as FYP in the following) will be developed.

FYP applications are developed based on ideas of "Better electronic payment experiences on interactions within users." In other words, the bank is not a considerable role in this project; all functions are designed for interactions between users and users.

As a professional application system, it is necessary to implement the following components: mobile application, central server, and back-end database. And platforms, structures, and languages for each part, connections and communication methods between them shall be considered.

Mainly speaking, besides all the essential functions like login and registration, electronic payment methods are the most critical parts to be considered. Three kinds of arrangements for electronic payment will be developed in this project: pay by direct transfer, payment by QR code and payment by Face Recognition. A function of payment by direct transfer is added as a backup choice for users, which can make the whole application more completed and professional. The feature of payment by QR code is implemented as the primary method in FYP, which satisfies the payment needs in different scenarios. In other words, any two users can use QR code in serval ways to achieve the goal of payment. As for the last function of payment by Face Recognition, this is considered as the new feature in electronic payment systems. In this feature, the final goal is to achieve “Payment by scanning faces on portable(mobile) devices.” Until the current moment, there’re still many works to be done in this field; even Alipay does not support face recognition on mobile devices.

1.3. Project Scope and Limitation

This FYP project will be fully developed on a personal computer and a mobile phone, which means the central server and database will be deployed on one single laptop. Therefore, user capacity supported will be a significant limitation in this environment.

Meanwhile, as for the front-end application implemented on the mobile phone, the final product will not help for other operation systems, but only android, even more, it may carry out unpredictable issues like GUI layout missing and function support on mobile phone of another brand different from the testing machine (a Huawei and MI phone are tested in this project).

According to the basic ideas for function design in this project, it is assumed that all the functions are designed for interactions with users. Therefore, as it was mentioned before, the role of the bank is ignored in this project.

As for the safety, some simple security methods are implemented here like user's IP checking and QR code valid number. However, compared to formal commercial electronic applications, security methods implemented in FYP have significant limitations, which worth further improvements.

For the Face Recognition feature which has been implemented already, due to the hardware and Eigenface method used limitations, a limited number of users may be supported. Although the basic idea of face recognition payment has been achieved, even more, issues are worth for further consideration, as the justification of images and a real human, improvement of user capability and handling of wrong identification exception, which shall provide suggestions on formal commercial applications in real life. Details will be discussed in the following parts.

1.4. Project Status

The application for electronic payment system has already been built. According to the following [Table. 1.4.1] with the specific description, functions of the electronic payment system are listed.

Function Name	Description
Login	**Users log in with necessary information**
Registration	**Users register with necessary information**
Payment_byScan	**Users scan a QR Code to pay. Transfer value input is required**
Payment_byQRCode	**Users generate a QR Code which can be scanned by others. Transfer value input is required before generating**
Payment_byAccount	**Users directly type in target and value to transfer**
Gathering_byScan	**Users scan a QR Code for gathering. Transfer value is defined in the QR Code**
Gathering_byQRCode	**Users generate a QR Code which can be scanned by others to transfer**
GatherbyFace	**Users scan a person's face, who has already registered as a member and pre-upload the face images, and then the target user who has been scanned will receive a notification, which allows the target user to make the payment.**

GCM notification	**Users received notifications when transactions happened.**

Table. 1.4.1 Finished Function List

2. Construction - Module Perspective

2.1. Overall System Architecture

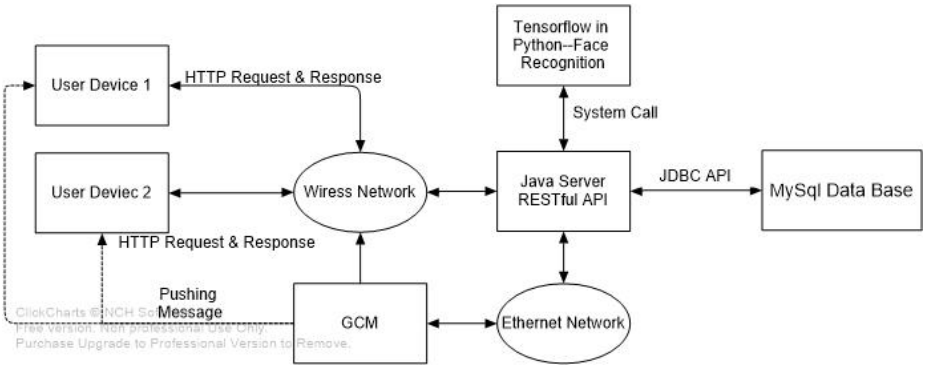


Figure. 2.1.1 Overall System Architecture

[Figure. 2.1.1] shows the overall system architecture of FYP project. There are four main modules in the whole system: Front-end application, primary server, database, and GCM. The front-end application is designed for end-users, who are going to use this electronic payment service. Meanwhile, the front-end application is deployed in users' end-devices, which could be portable. Both central server and database are implemented in one single PC. Therefore, the hardware issue could be a limitation for system capacity and performance. According to [Figure. 2.1.1], the users send HTTP request from the mobile device to the primary server via the wireless network (WIFI mostly). And the central server communicates with MySQL database through JDBC (Java Database Connectivity) API. As for the GCM, the central server will send HTTP request to GCM, which is in the cloud server supported by Google and updated to Firebase Cloud Messaging (FCM), via the Ethernet network. GCM will then push the costumed messages to specific users via the wireless network.

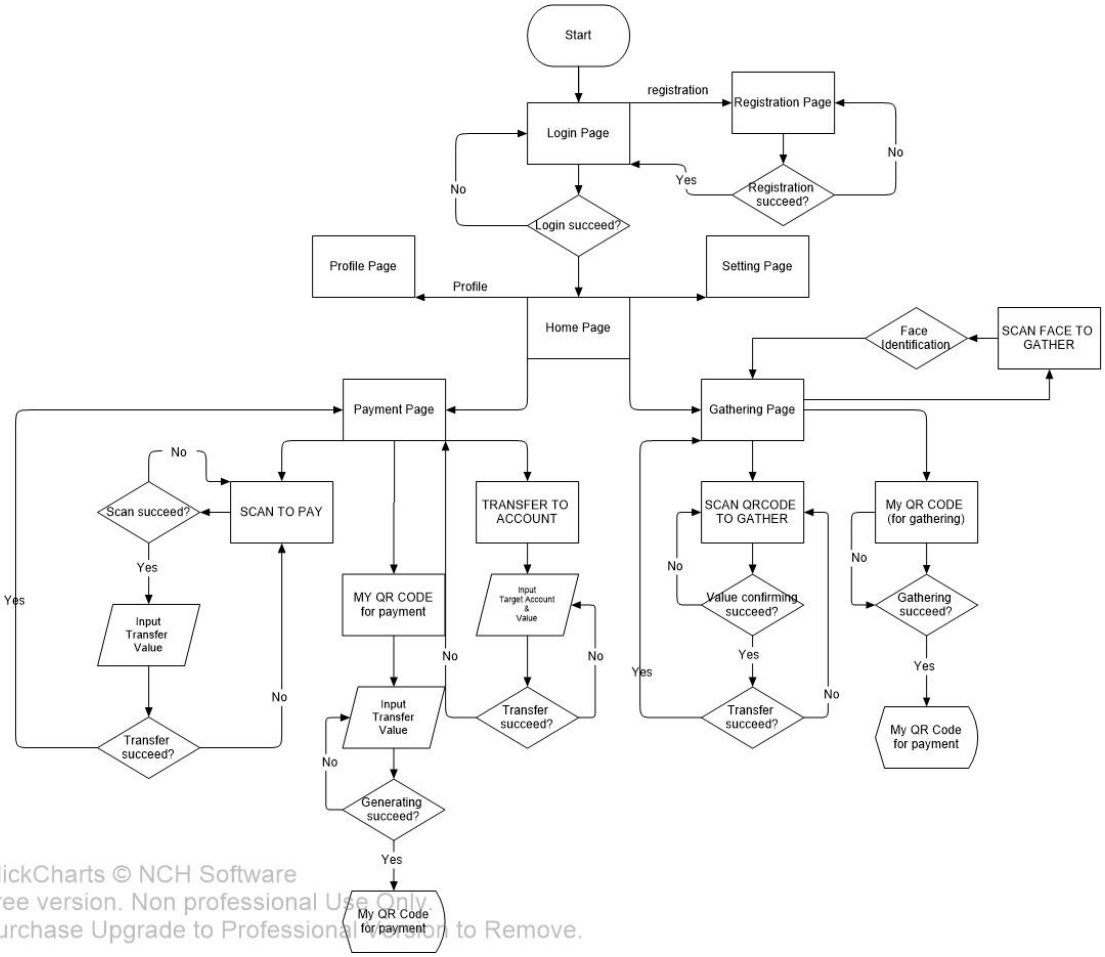
2.2. Front-end Application

2.2.1. Summary of Front-end Application

In this project, the front-end application is developed on Android platform, because Android has currently become the most popular operating system for portable devices. Therefore, the Android Studio IDE is chosen for growing, which is the official IDE tools presented by Google and it is considered to be the most wildly used tools for android programming.

The front-end application takes the responsibility of interaction with different users. Therefore, the principles of UI design shall be taken into consideration like being convenient, precise and user-friendly.

2.2.2. User Interface flow Diagram



ClickCharts © NCH Software
Free version. Non professional Use Only.
Purchase Upgrade to Professional Version to Remove.

Figure. 2.2.2.1 UI Flow Diagram

According to the [Figure. 2.2.2.1], this diagram describes the working flow of the android front-end application and user interface structure. When the user clicks the application icon on mobile devices, the Android app will be launched with the login page.

2.2.3. Login Page

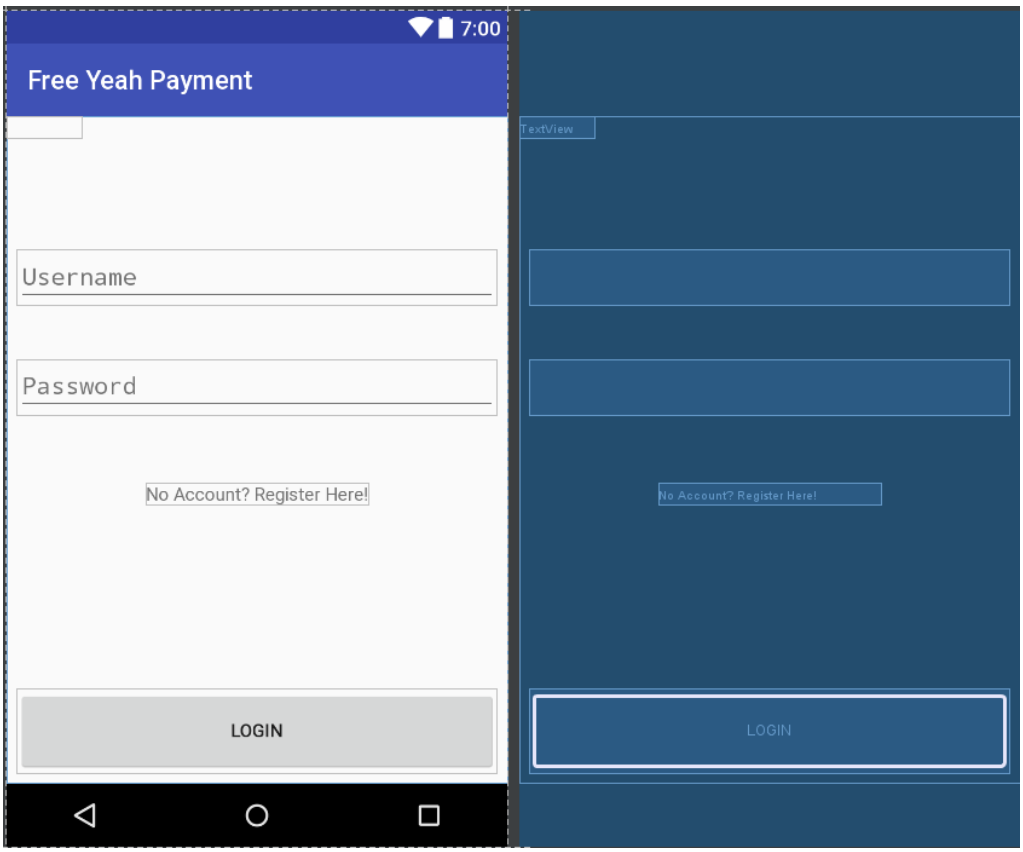


Figure. 2.2.3.1 Login page layout

The uses are asked to log in to start using this application. According to [Figure. 2.2.3.1], two essential attributes are required for login, username, and password. To make this UI simple and easy-to-use, the login button is set at the bottom of the page. However, for a new user, it can jump to the registration page by clicking the text view "No Account? Register Here!".

2.2.4. Registration Page

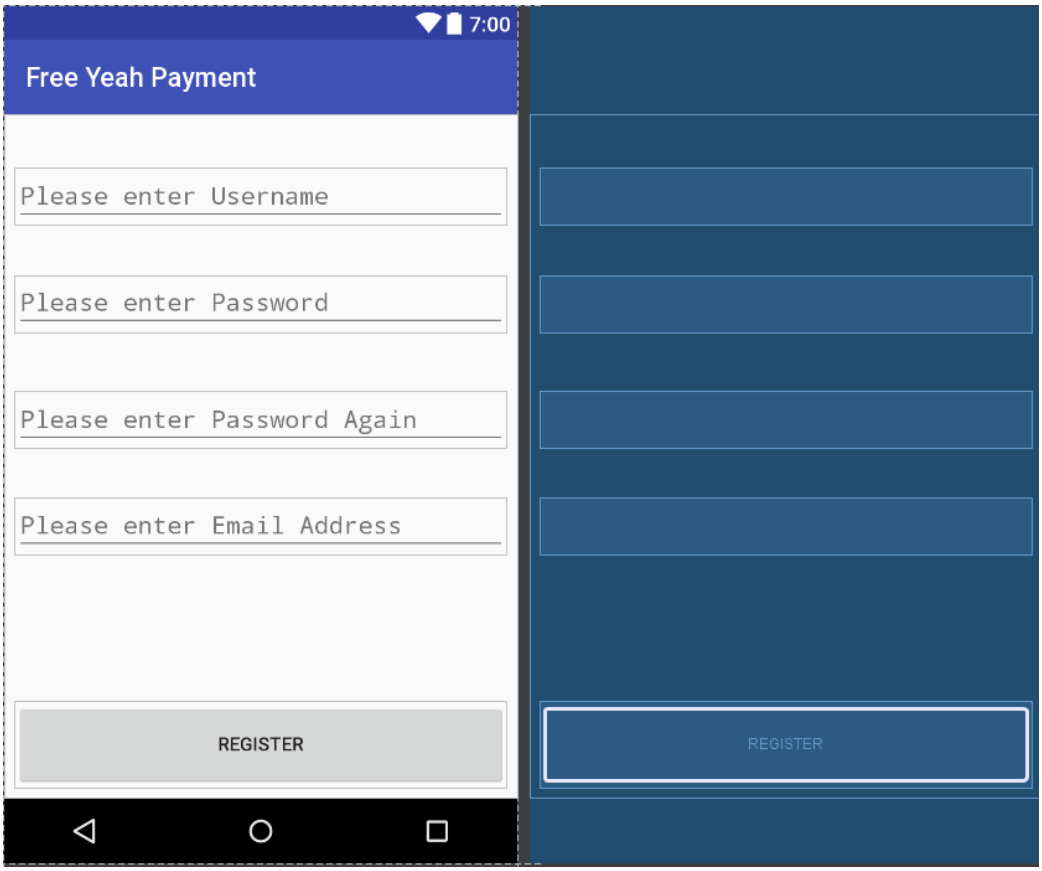


Figure. 2.2.4.1 Registration page layout

In registration page [Figure. 2.2.4.1], the necessary information including username, password, and email address are required for a new account. Once these attributes satisfy the requirements, users can click the register button. If failed, a dialog will show up to inform the user that registration does not succeed. If succeed, the application will jump back to login page.

2.2.5. Payment & Gathering Page

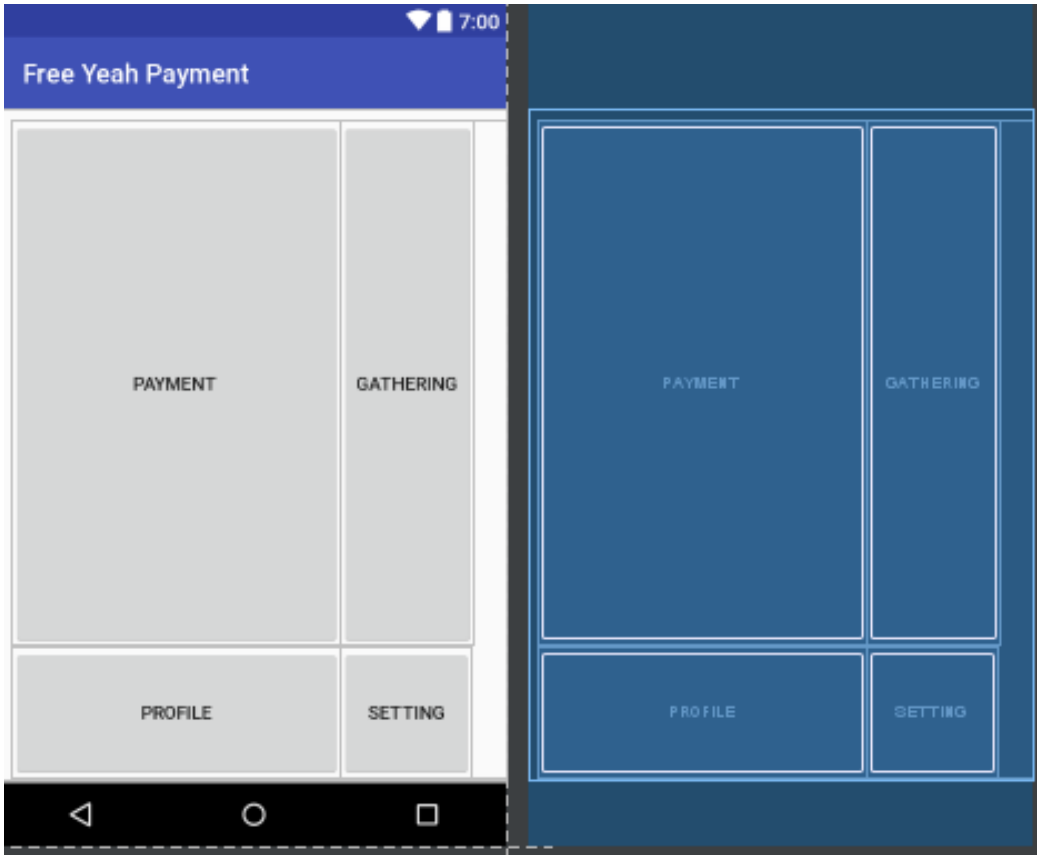


Figure. 2.2.5.1 Homepage layout

After the user successfully login in, the home page is allowed for access. There are only four components (buttons) on the home page, payment, gathering, profile, and setting. This layout is designed to make the users can straightly achieve their goals. Due to the project objectives, it is assumed that users perform payment actions most of the time, secondly the gathering. Therefore, the payment button and gathering button take most of the page space, which makes them visible and easy to click.

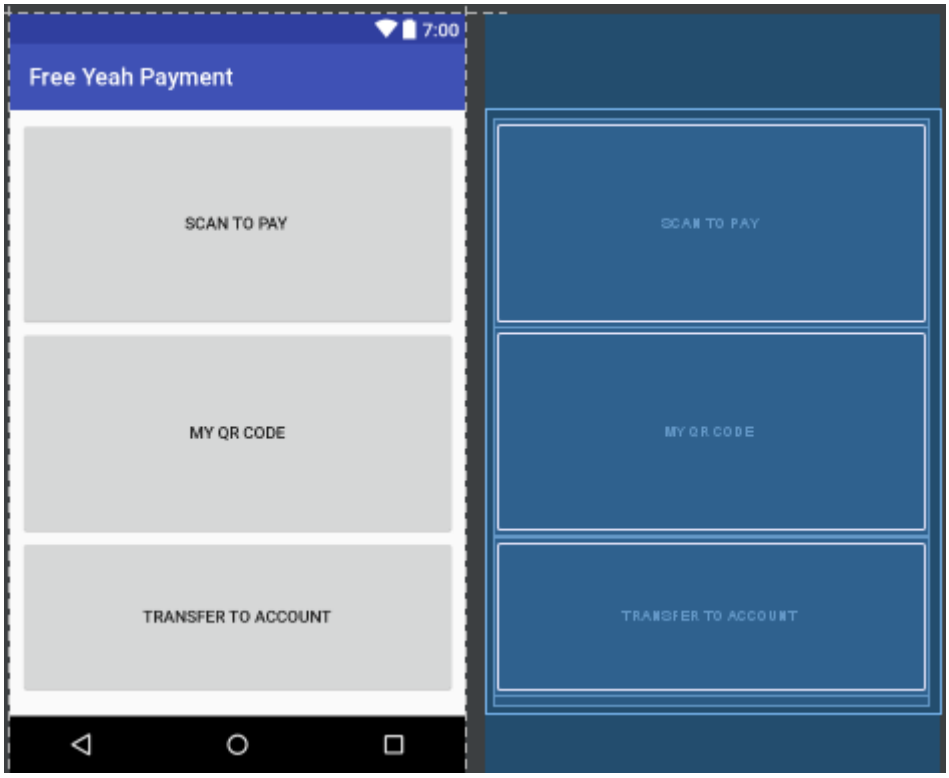


Figure. 2.2.5.2 Payment page layout

[Figure. 2.2.5.2] provides three entries for different payment methods, scan a QR code to pay, generate a QR code to be paid and directly transfer my account. So far as it is, the three buttons take the whole page. Each button is big enough for the user to click. It is similar to the gathering page [Figure. 2.2.5.3].

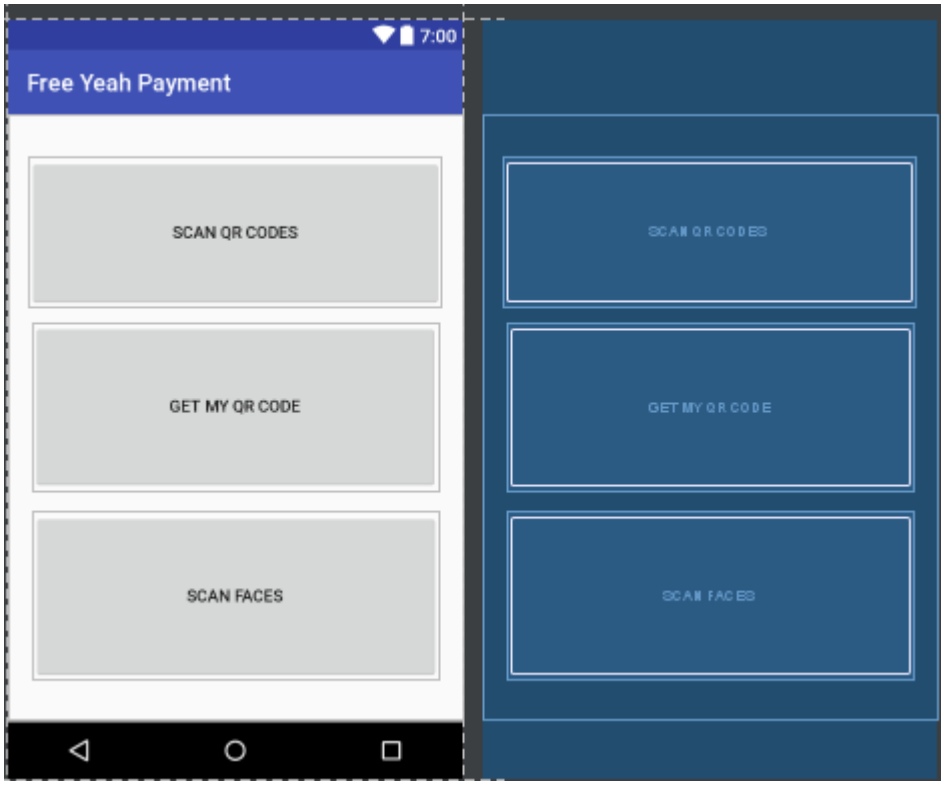


Figure. 2.2.5.3 Gathering page layout

2.2.6. Generate & Scan QR Code

When click “SCAN TO PAY” button in [Figure. 2.2.5.2] or “SCAN QR CODES” in [Figure. 2.2.5.3], the application will call out the camera for scanning QR codes [Figure. 2.2.6.1]. If successful in this stage, a dialog will show up and require the user to input transfer value, which shall be valid in a transaction.

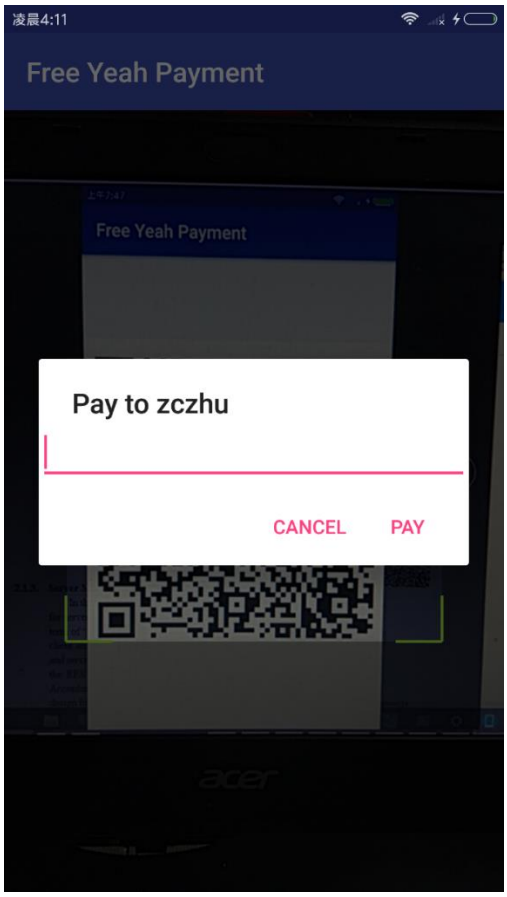


Figure. 2.2.6.1 Pay by scanning QR Code

When clicking "MY QR CODE” button in [Figure. 2.2.2.5] or “GET MY QR CODE” in [Figure. 2.2.6.1], the application will generate the corresponding QR code for the user. In fact, these two types of QR code have different expire period. The QR code for payment can only be scanned and used once, but the QR code for gathering can be accessed without time limitations. The QR code for payment can be considered as a cheque, which contains a value of money inside. Therefore, this QR code shall only be used once. This mechanism is implemented by updating and checking of an attribute of the user called "user_qrValidNum." This "user_qrValidNum" attribute is a randomly generated string, and it is updated to the database when the user requests a QR code for payment. Once the QR code is scanned, and the value is transferred, the server will update the "user_qrValidNum" attribute to the default string. Meanwhile, every moment when another user scanned the QR code to get the value inside, the server will check whether

the "user_qrValidNum" attribute satisfies. In other words, no matter the QR code for payment is used or not, once the owner generates a new one, the old one will be banished.

2.2.7. Scan A Face

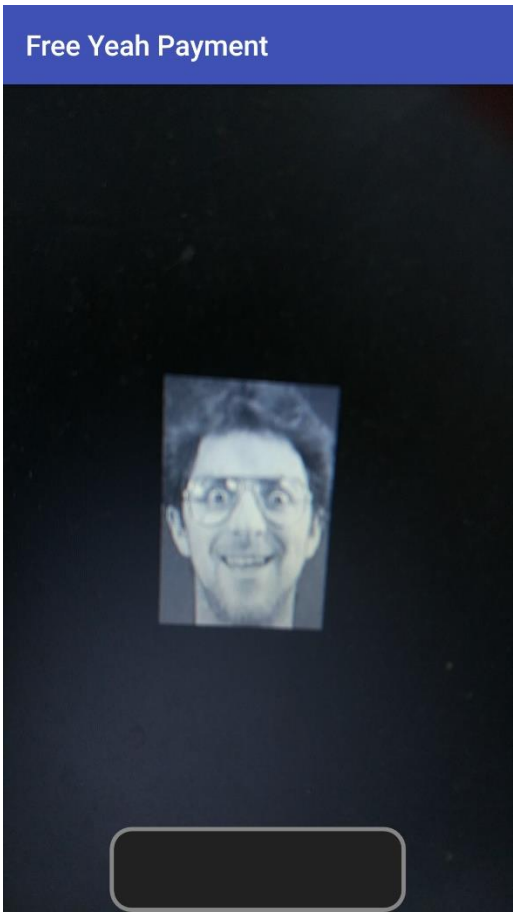


Figure. 2.2.7.1 Pay by scanning QR Code

When the user clicks “SCAN FACES” button in gathering page [Figure. 2.2.5.3], the application will move to page [Figure. 2.2.7.1], which is a costumed camera in fact. The user can use this camera and press the black bar at the bottom to take a photo. This image will be transferred to a 92*112 gray-scale JPEG file, as the HTTP request body with POST method generated by the android client. This HTTP request is sent to ask the server for face identification. If there are no problems so far, the server will send a notification to the target user, which is assumed to be scanned and will pay for the transaction. More technical details will be discussed in the later chapter of Face Identification.

2.2.8. GCM Notification on Application Side

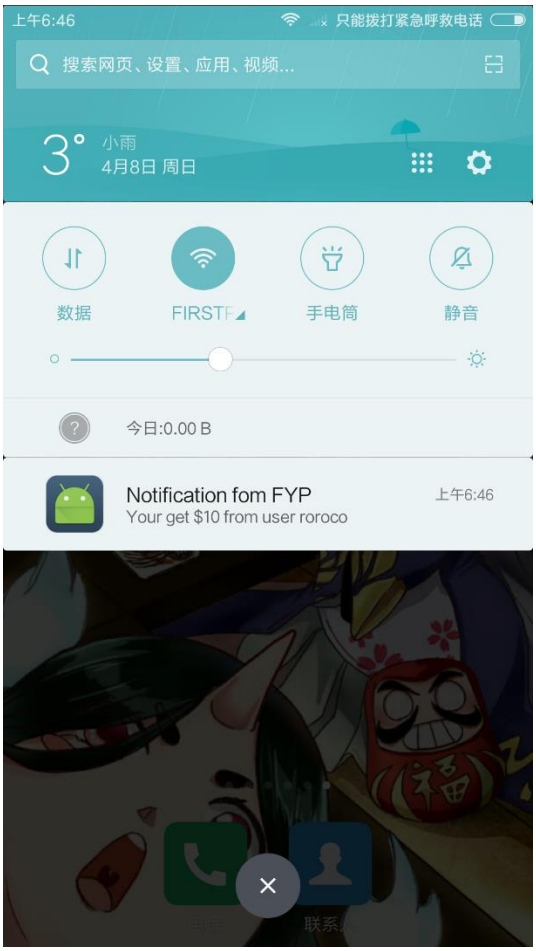


Figure. 2.2.8.1 GCM notification example

As shown in [Figure. 2.2.8.1], the user will receive such notifications when transactions are finished or payment request by face identification. The push time, title, message, icon and click-on event are all costume available. However, due to the primary definition by GCM, there is one limitation of this feature that users cannot receive the notification when the application is currently on.

2.2.9. Technical Implementation of Application

2.2.9.1. Construction Tools

The front-end application is implemented on Android Studio, which is the most widely used IDE for android programming because it is open source and user-friendly for programming, like visual programming for layout and the virtual device simulator. Most of the project files are in Java and XML, called the activity and the layout. In android programming, each application page is generated and run by a Java activity and an XML layout file. In other words, the XML file determines the layout of the page and Java activity defines behaviors and interactions.

2.2.9.2. File Structure

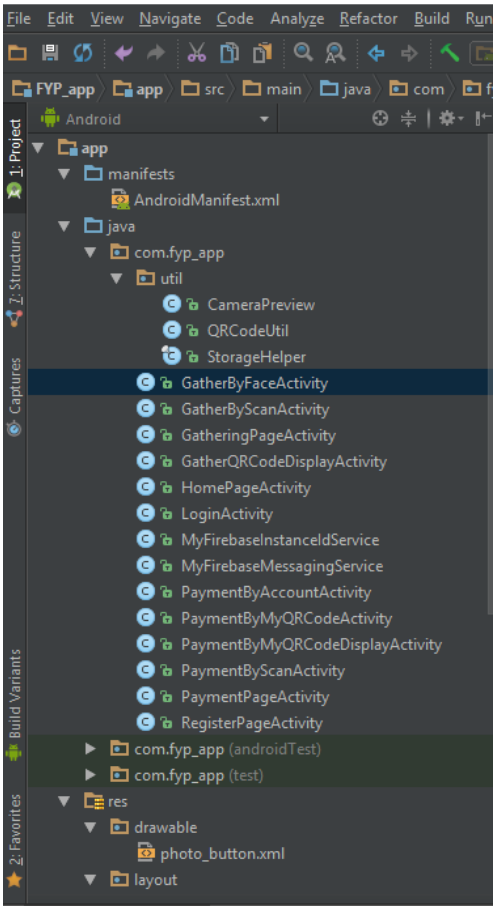


Figure. 2.2.9.2.1 Android file structure_1

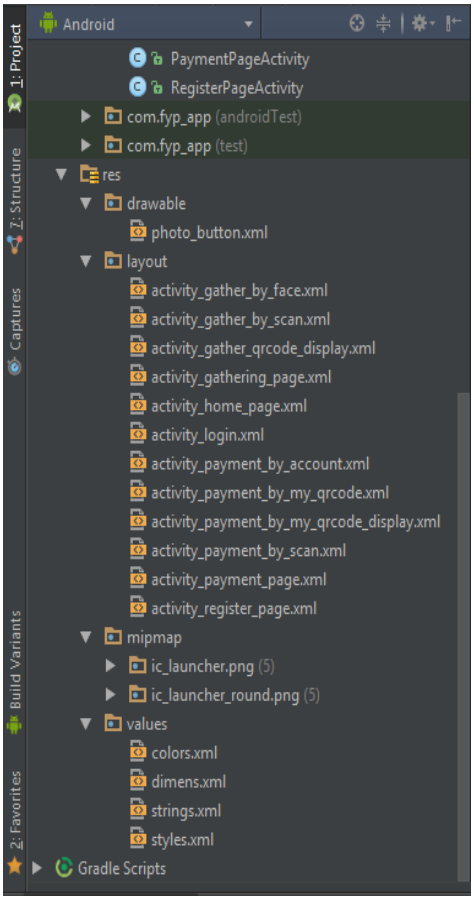


Figure. 2.2.9.2.2 Android file structure_2

- /manifests/AndroidManifest.xml
  (This file defines all the basic information of this project and user permissions allowed)
- /java/com.fyp_app/
  - util/<files> (External and self-defined libraries)
  - <activityFiles>
  - <serviceFiles> (These service files are implemented for the GCM) service.

- /java/com.fyp_app (androidTest)
- /java/com.fyp_app (test)
- /res/drawable/<.xml> (Accessible self-defined image files for icon and button)
- /res/layout/<.xml> (Layout definitions which correspond to the specific <activityFile>)
- /res/mipmap/ (Accessible image files for icon)
- /res/values/ (Accessible pre-defined global values)

2.2.9.3. Communication Method with Server

Because of the server implementation of RESTful API style, which keeps listening, accepting HTTP requests and sending back the corresponding responses. Therefore, whenever the android application is supposed to communicate with the primary server, the application starts a new thread, which generates the HTTP request, sends it out. And then waiting for the response. As is shown in [Figure. 2.2.9.3.1], the HTTP default method is set to be GET, which means parameters are involved in the URL.

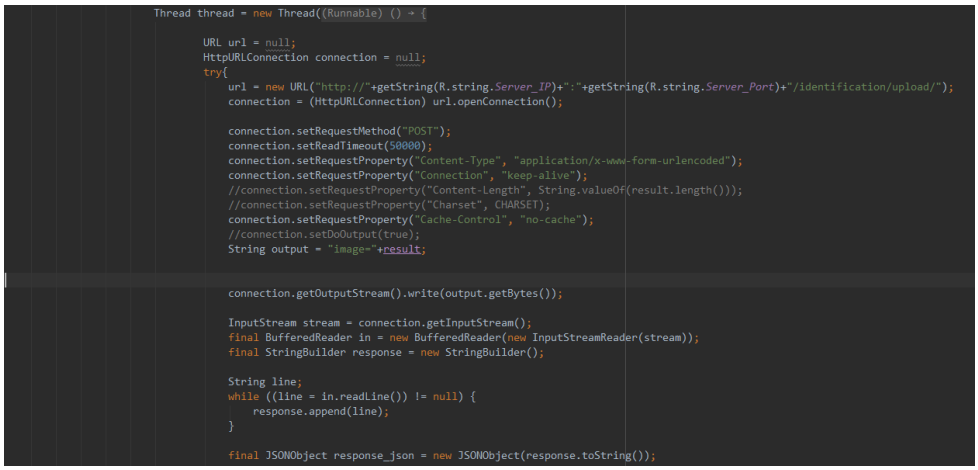
```
//Thread for network
    Thread thread = new Thread(new Runnable() {

        @Override
        public void run() {

        URL url = null;
        HttpURLConnection connection = null;
        try{
            url = new URL("http://"+getString(R.string.Server_IP)+":"+getString(R.string.Server_Port)+"/login/"+login_username+"&"+login_password);
            connection = (HttpURLConnection) url.openConnection();


            InputStream stream = connection.getInputStream();
            final BufferedReader in = new BufferedReader(new InputStreamReader(stream));
            final StringBuilder response = new StringBuilder();

            String line;
            while ((line = in.readLine()) != null) {
                response.append(line);
            }

            final JSONObject response_json = new JSONObject(response.toString());
```

Figure. 2.2.9.3.1 HTTP GET request from the application

This mechanism has benefits on simplifying the communication method, which suits the server requirements best. However, some parameters like password, transfer value, and transfer value shall not be generated into URL directly. It is possible to develop an encryption function for the settings or change the server structure and use HTTPS for the communication.

All the responses from the server are self-defined as JSON format, and the android application can resolve the answer, which determines the next process of the claim.

However, in face identification part, the Android application takes the responsibility of uploading an image to the server to identify. Therefore, in the activity of gathering by face, once the application successful gets the face image and transfers it to the target size in grayscale, the application will encode the image in BITMAP format to string by the BASE64 method. And then transfer the string with HTTP request by POST method [Figure. 2.2.9.3.2].



```
Thread thread = new Thread((Runnable) () -> {

        URL url = null;
        HttpURLConnection connection = null;
        try{
            url = new URL("http://"+getString(R.string.Server_IP)+":"+getString(R.string.Server_Port)+"/identification/upload/");
            connection = (HttpURLConnection) url.openConnection();

            connection.setRequestMethod("POST");
            connection.setReadTimeout(50000);
            connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
            connection.setRequestProperty("Connection", "keep-alive");
            //connection.setRequestProperty("Content-Length", String.valueOf(result.length()));
            //connection.setRequestProperty("Charset", CHARSET);
            connection.setRequestProperty("Cache-Control", "no-cache");
            //connection.setDoOutput(true);
            String output = "image="+result;


            connection.getOutputStream().write(output.getBytes());

            InputStream stream = connection.getInputStream();
            final BufferedReader in = new BufferedReader(new InputStreamReader(stream));
            final StringBuilder response = new StringBuilder();

            String line;
            while ((line = in.readLine()) != null) {
                response.append(line);
            }

            final JSONObject response_json = new JSONObject(response.toString());
```

Figure. 2.2.9.3.2 HTTP POST request from the application

Furthermore, once the server receives the target string, it will decode the string to JPEG format by the BASE64 method. BASE64 encoding and decoding methods are available libraries for Java.

2.3. Server Module

2.3.1. Overview of Server Design

In the design of server module, primarily the RESTful API structure is implemented for server construction. RESTful API has the critical term "REST," which stands for the full term of "Representational State Transfer." REST describes an interaction mode between client and server. Therefore, RESTful APIA means API in REST style structure. Clients and servers communicate with each other through HTTP requests and responses, in other words, the REST structure uses URL for locating resources and HTTP action for behaviors. According to the definition and features of RESTful API, it has been regarded as a suitable design for the web application, cause URL and HTTP have been designed for web technology. However, in FYP project, RESTful API is used as the base server structure, which is not common though, there're reasons for the choice. Firstly, the RESTful API has been well designed and developed already, which means there's no need to define new communication protocol because it uses URL to locate the resources. Secondly, the RESTful API is light-weight and has reasonable performance on the test server machine, which owns only personal level hardware. Lastly, for further consideration, because this API is designed for the web application, therefore, it is assumed that platform transplantation will be more natural. Java is chosen to be the programming language because it is an accessible language for RESTful API server construction.

2.3.2. Working Flow of Server

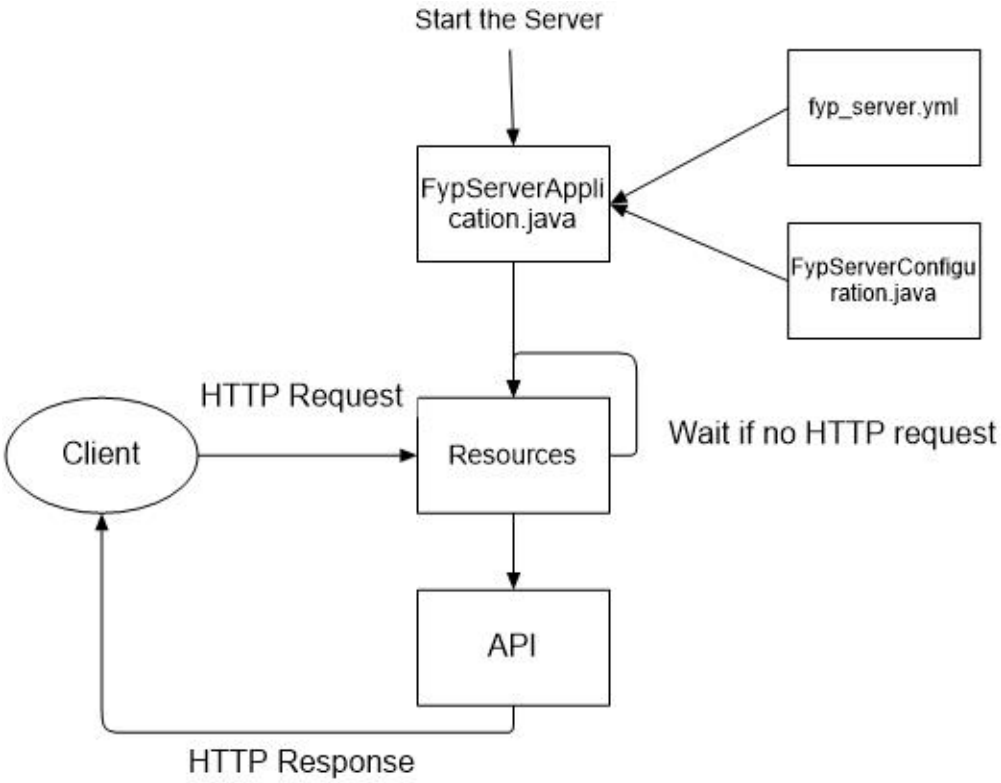


Figure. 2.3.2.1 Flow Diagram of Server

[Figure. 2.3.2.1] is the flow diagram of the server, which shows the working pattern of the server. The followings are the detail description:

- Run file FypServerApplication.java to start the server
- File fyp_server.yml defines the port of the server
- File FYPServerConfiguration.java defines the other initial configurations
- Resource package contains all the resource files (like Login.java). When there is an HTTP request from the client, the server will locate the resource file according to the HTTP request with URL. If there is no request, just waiting for the activation.
- Once the resource file is located, call the API files to response. In other words, API files define the format of the response
- The response is sent back to the client by HTTP

The server design of FYP combines the regular listening pattern and RESTful API together, which is a new design approach for server design. The android applications on user devices send the HTTP request and the server responses with a pre-defined object. However, there're limitations for this server structure. For instance, compared to socket communication method, URL with HTTP actions method may be less efficient in image transfer.

2.3.3. Technical Implementation of Server

2.3.3.1. Construction Tools

The Java server is fully implemented on Eclipse, which is a powerful IDE for Java programming and efficient on project management. Since a RESTful structure was determined as the backbone of the server, a specific version of a RESTful architecture called Dropwizard was implemented.

To have better management on this project, Maven on Eclipse was used to load dependencies and build the project, which is an uncomplicated project management tool. The file structure will be discussed in next section.

Therefore, before server starts, Maven will rebuild the whole project when file structure or dependencies changed. Run the server application file to start the server quickly.

2.3.3.2. File Structure

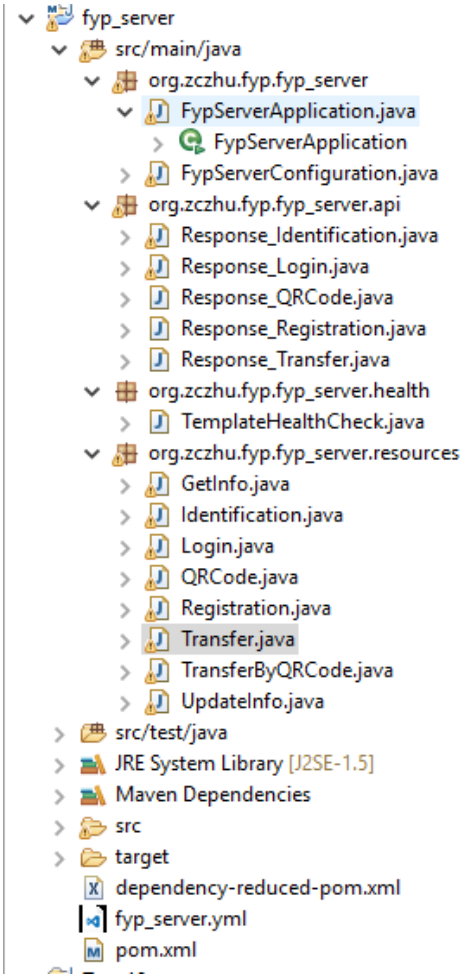


Figure. 2.3.3.2.1 File structure of the server

According to [Figure. 2.3.3.2.1], the following statements describe the file structure and function of this file.

- org.zczhu.fyp.fyp_server
  - FypServerApplication.java
    - Run this file to boot up the whole server system.
    - Creates connection instance of JDBC to the MySQL database.
    - Registers all valid resources and passes the connection instance to them, which allows the resource files to access the database
    - Print out all the available HTTP request formats to the console [Figure. 2.3.3.2.2]
  - FypServerConfiguration.java
    - Defines the default configuration for the server.
    - This file is empty in this project.
- org.zczhu.fyp.fyp_server.api
  - Response_<ResourceName>.java

        - Defines all response object for each corresponding resource.
        - These response objects are returned to clients as JSON objects.
- org.zczhu.fyp.fyp_server.health
    - TemplateHealthCheck.java
        - This is the default health check class which test the server whether is dead or not.
- org.zczhu.fyp.fyp_server.resources
    - <ResourceName>.java
        - Each resource defines a specific function and the corresponding HTTP request format for accessing.
        - Handle the connection instance passed by FypServerApplication.java.
        - Return the corresponding JSON object to the clients as the HTTP response
- pom.xml
    - This file is generated by Maven, which defines all the dependencies and project building information.
- fyp_server.yml
    - This file defines the access ports for the server.
    - Port 8000 for application connectors.
    - Port 8001 for admin connectors. Enter URL http://localhost:8001 to access to the operational menu.

```
GET     /getinfo/test (org.zczhu.fyp.fyp_server.resources.GetInfo)
GET     /getinfo/{username}&{type} (org.zczhu.fyp.fyp_server.resources.GetInfo)
GET     /identification/test (org.zczhu.fyp.fyp_server.resources.Identification)
POST    /identification/upload/ (org.zczhu.fyp.fyp_server.resources.Identification)
GET     /login/{username}&{password} (org.zczhu.fyp.fyp_server.resources.Login)
GET     /qrcode/{username}&{type}&{value}&{fcmToken} (org.zczhu.fyp.fyp_server.resources.QRCode)
GET     /registration/{username}&{password}&{emailaddr} (org.zczhu.fyp.fyp_server.resources.Registration)
GET     /transfer/{fromName}&{toName}&{value} (org.zczhu.fyp.fyp_server.resources.Transfer)
GET     /transferbyqrcode/{fromName}&{toName}&{value}&{type}&{qrValidNum}&{fcmToken} (org.zczhu.fyp.fyp_se
GET     /updateinfo/{username}&{type}&{newdata} (org.zczhu.fyp.fyp_server.resources.UpdateInfo)
```

Figure. 2.3.3.2.2 Available HTTP request formats

2.3.3.3. Communication Method with Database

Because MySQL database was implemented in this project, a library called Java DataBase Connectivity (JDBC) is used for communication between the Java server and the MySQL database.

```
try{
    java.sql.Statement stmt = conn.createStatement();
    java.sql.ResultSet rs = stmt.executeQuery(new String("SELECT * from fyp_user where user_name = \""+username+"\""));
    while(rs.next()){
        if(type.equals("user_name") || type.equals("user_emailAddr") || type.equals("user_bankAccount")){;
            return rs.getString(type);
        }else if (type.equals("user_balance")){
            return rs.getString(type);}}
    rs.close();
    stmt.close();
}catch(SQLException se){
    se.printStackTrace();
}catch(Exception e){
    e.printStackTrace();
}
```

Figure. 2.3.3.3.1 JDBC example

Because connecting and disconnecting in every resource will reduce the performance of the server. Therefore, a connection instance of JDBC is created in FypServerApplication.java and is passed to every resource for utilization. [Figure. 2.3.3.3.1] shows how JDBC was used in the resource, by executing a query via the instance stmt (java.sql.Statement) and storing the query result in instance rs (java.sql.ResultSet). The data can be accessed via the result set rs.

2.4. Database Module

2.4.1. Overview of Database

MySQL database is deployed because MySQL is free for student and has excellent performance in data management. In this project, two tables were created: table fyp_user for user information and table fyp_trans for transaction information.

2.4.2. Table Illustration

2.4.2.1. Table of fyp_user

Field/Attribute	Description
user_id	Primary key, auto increment
user_name	Set to be unique
user_password	Password for login
user_emailAddr	User's email address
user_balance	User's current balance
user_createTime	Timestamp of account creation
user_lastModifiedTime	Timestamp of last modification
user_valid	1 for valid, 0 for invalid
user_bankAccount	User's bank account
user_ip	User's last login IP address
user_qrValidNum	6-bit random string for current QR Code generated by this user
user_qrValue	Value information shall be contained be the current QR Code generated by this user
user_fcmToken	The application token for GCM notification. This attribute will be updated when user login.

Table. 2.4.2.1.1 Attribute description of fyp_user

Field	Type	Null	Key	Default	Extra
user_id	int(5)	NO	PRI	NULL	auto_increment
user_name	varchar(10)	NO	UNI		
user_password	varchar(20)	NO		NULL	
user_emailAddr	varchar(20)	NO			
user_balance	int(10)	NO		0	
user_createTime	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	
user_lastModifiedTime	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	on update CURRENT_TIMESTAMP(6)
user_valid	char(1)	NO		1	
user_bankAccount	varchar(20)	NO			
user_qrExpired	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	
user_ip	varchar(16)	YES		0.0.0.0	
user_qrValidNum	varchar(6)	NO		xxxxxx	
user_qrValue	int(10)	NO		-1	
user_fcmToken	longtext	YES		NULL	

Figure. 2.4.2.1.2 Table structure of fyp_user

The table of fyp_user stores all the information related to the user basically like id, name, password, and balance. However, some special attributes are belonging to specific functions.

For attribute user_ip, it is used to implement a security method. When the user login successfully, the login IP address will be stored into the database, which can guarantee that the transactions are made after the user login. In other words, attribute user_ip can be considered as verification of current session, because every moment when the transaction is going to be made, the server will check the user_ip attribute, make sure the transaction request is from the right user.

For attribute user_qrValidNum, it is used to implement another security method for QR code. According to the function design, the user can generate QR codes for payment and gathering. The QR code for gathering has no limit on usage times. However, the QR code for payment can be scanned and used up for only once. Therefore, every moment when the user sends out the request for generating a QR code for payment with pre-entered value, the server will update the randomly generated user_qrValidNum. And check this attribute when any user is trying to scan the code to receive money from the QR code owner. After the transaction is done, this attribute will be set to the default value.

2.4.2.2. Table of fyp_trans

Field/Attribute	Description
trans_id	**Primary key, auto increment**
trans_fromID	**The user id of payment side**
trans_toID	**The user id of gathering side**
trans_fromName	**User name of payment side**
trans_toName	**User name of gathering side**
trans_fromBalance	**User's balance of payment side**
trans_toBalance	**User's balance of gathering side**
trans_value	**Value of this transaction**
trans_createTime	**Time of transaction creation**

Table. 2.4.2.2.1 Attribute description of fyp_trans

Field	Type	Null	Key	Default	Extra
trans_id	int(20)	NO	PRI	NULL	auto_increment
trans_fromID	int(5)	NO		0	
trans_toID	int(5)	NO		0	
trans_fromName	varchar(10)	NO		NULL	
trans_toName	varchar(10)	NO		NULL	
trans_fromBalance	int(10)	NO		0	
trans_toBalance	int(10)	NO		0	
trans_value	int(10)	NO		0	
trans_createTime	timestamp(6)	NO		CURRENT_TIMESTAMP(6)	

Figure. 2.4.2.2.2 Table structure of fyp_trans

The table of fyp_trans stores information of all successful transactions.

3. Implementation - Function Perspective

3.1. Overview of Function Implementation

Based on the project objectives, this electronic payment system is supposed to be multi-functional. Focusing on electronic payment service, a function list was generated as a reference [Table. 1.4.1]. This chapter mainly discusses the project through a function perspective.

3.2. Login & Register

3.2.1. Login Function

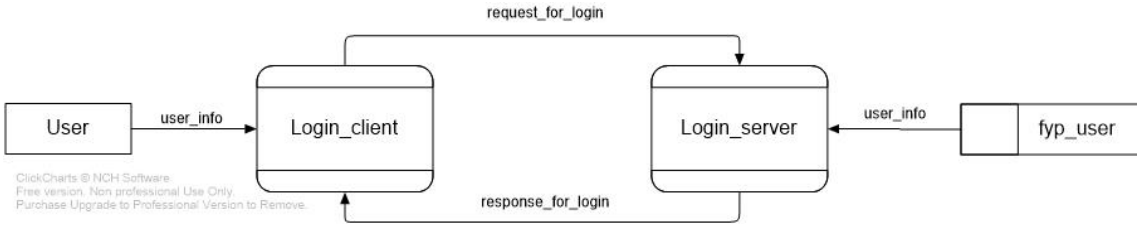


Figure. 3.2.1.1 Data flow diagram of the login

[Figure. 3.2.1.1] shows the data flow of login function. When the user enters the username and password, and then click the login button referred to [Figure. 2.2.3.1], the Android application generates an HTTP request, passes the required information (username and password) to Java server as parameters in URL. Once the server receives the HTTP request for login, it locates to the corresponding resource file based on the URL.

In this process, the server will retrieve the correct password from the database, check whether the password matches and return the corresponding response object to the client.

3.2.2. Register Function

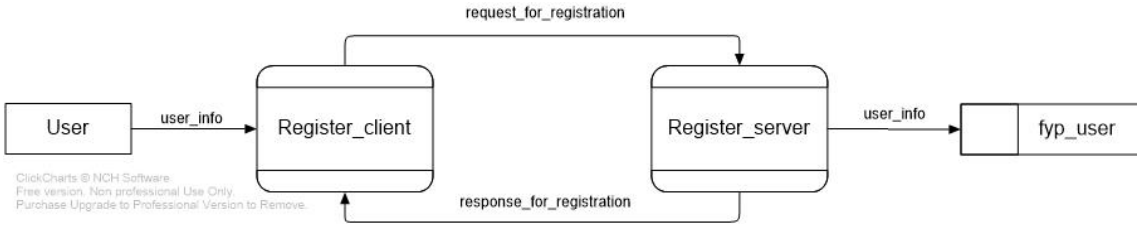


Figure. 3.2.3.1 Data flow diagram of the register

[Figure. 3.2.3.1] shows the data flow of register function. When the user enters the required information and clicks, the register button referred to [Figure. 2.2.4.1], the Android application generates an HTTP request, passes the information required to Java server as parameters in URL. The server will check the validation of register information and return the response to the client.

3.3. Payment by Transfer Function

The function of payment by transfer is designed for a direct transfer method, which requires target username and transfer value to make the transaction. And this function is not supported for gathering.

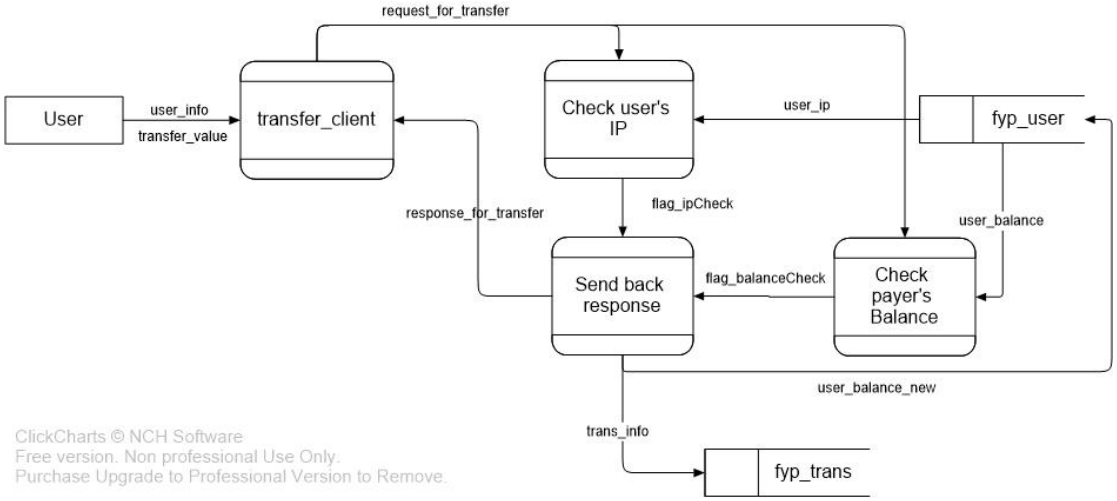


Figure. 3.3.1 Data flow diagram of payment by transfer

[Figure. 3.3.1] shows the data flow of payment by the transfer function. When the android application sends the transfer request to the server, and after the server successfully locates to the resource, the server will begin to check the users' IP address to make sure the request owner has already been login. After that, the server will check the user's balance to make sure this transaction request is valid. Finally, return the response to the client.

3.4. Transaction by QR Code Function

3.4.1. Introduction to QR Code

QR Code stands for Quick Response Code, invented by Denso Cooperation in 1994. It is a 2-dimensional matrix code, which stores data and information. Compared to the traditional barcode, QR code is reliable, safe and better capacity of data. Nowadays, QR code has become a standard technology and shows up in daily life. QR code used in electronic payment application

simplifies the payment procedure.

3.4.2. Flow of Transaction by QR Code

Both of payment and gathering are supported by QR code in this project.

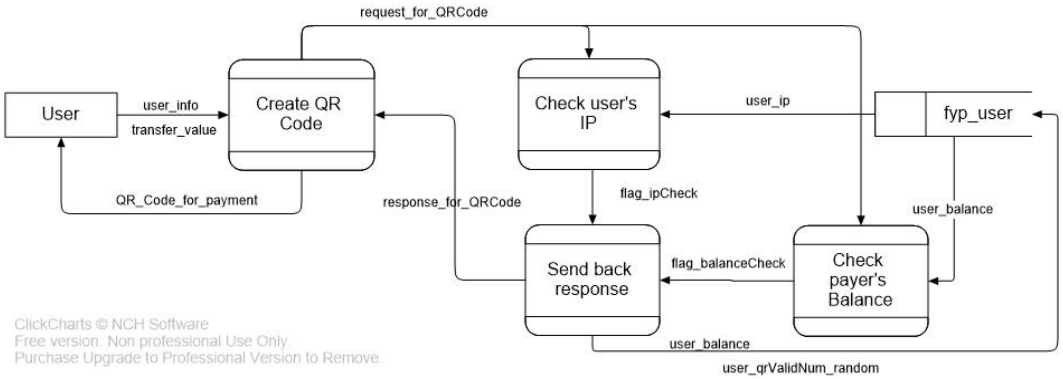


Figure. 3.4.2.1 Data flow diagram of payment by scanning QR codes

When the user clicks the “SCAN TO PAY” button referred to [Figure. 2.2.5.2], the camera for scanning QR codes will be launched. If the user focuses the camera on a QR code, it will automatically scan it and send the corresponding HTTP request to the server.

The QR code carries necessary information like username, QR code type which identifies the QR code is for payment or gathering. Because in this case which scans a QR code to pay to another user, the payment user is required to enter transfer value, which means the QR code only provides the receiver information.

After the user successfully scans and enter the transfer value, like payment by the transfer function, the server will check user's current IP address with login IP address and balance before completing the transaction and respond to the client [Figure. 3.4.2.1].

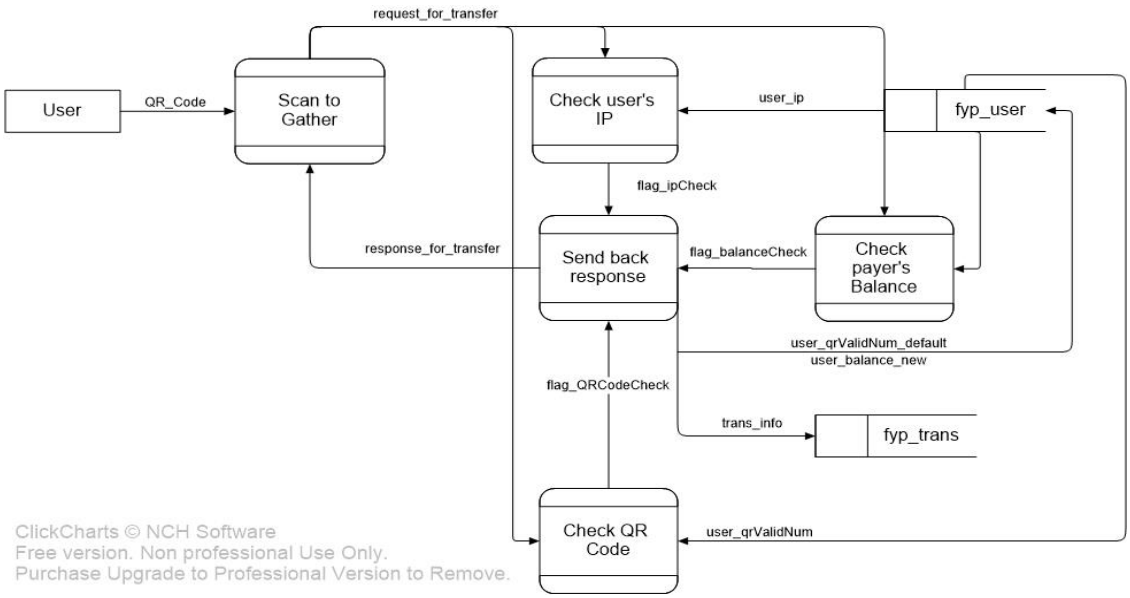


Figure. 3.4.2.2 Data flow diagram of gathering by scanning QR codes

Another transaction function related to QR code is scanning a QR code to receive money. In this case, the QR code performs like the cheque and can only be used once. When the user clicks, the "SCAN QR CODES" button referred to [Figure. 2.2.5.3], the application will launch the QR code camera. Once the user successfully scans the QR code and click confirm button in the dialog, the client will send out the transfer request. There is no need for the user to enter transfer value, cause the allowed transfer value has been carried by QR code. Therefore, an attribute called QR code validation number, which is generated from the QR code will be check in the server process.

The validation number is created when the request of generating QR code is accepted by the server (discussed in next section for generating QR code), and stored in database binding with the QR code owner. Once this QR code is scanned and the transaction is finished, the server will set the QR code validation number of the owner to a default value, which means this QR code has become invalid. Meanwhile, every user can keep only one cheque type QR code valid, and if a new one is generated, the old one will be considered as invalid and lose its value.

3.4.3. Generate & Scan QR Codes

The user can click “MY QR CODE” button in payment page to generate a payment type QR code which requires to the transfer value, and click “GET MY QR CODE” button to generate a gathering type QR code.

In fact, when the user clicks the button to generate a QR code, the client application sends HTTP request for generating QR codes to the server, which will be located to resource file “QRCode.java”. Especially for the payment type QR code, the server will check the transfer value with user’s balance and update the QR code validation number.

The server will return a response for permission to the client. For the payment type QR code, the QR code validation number will also be returned. Based on the response object, the client application will generate the QR code for the user.

Both of the generating and scanning functions are implemented by ZXing library, which is an open source project developed by Google.

3.5. Gathering by Scanning a Face Function

3.5.1. Function Specification

This application supports scanning a registered member’s face to start a transaction. Once the face is scanned, the file will be uploaded to server by an HTTP POST request. If the face identification is successful, the payment user will receive a notification from GCM, and the user can click the notification to start the payment.

3.5.2. Flow of Face Identification Function

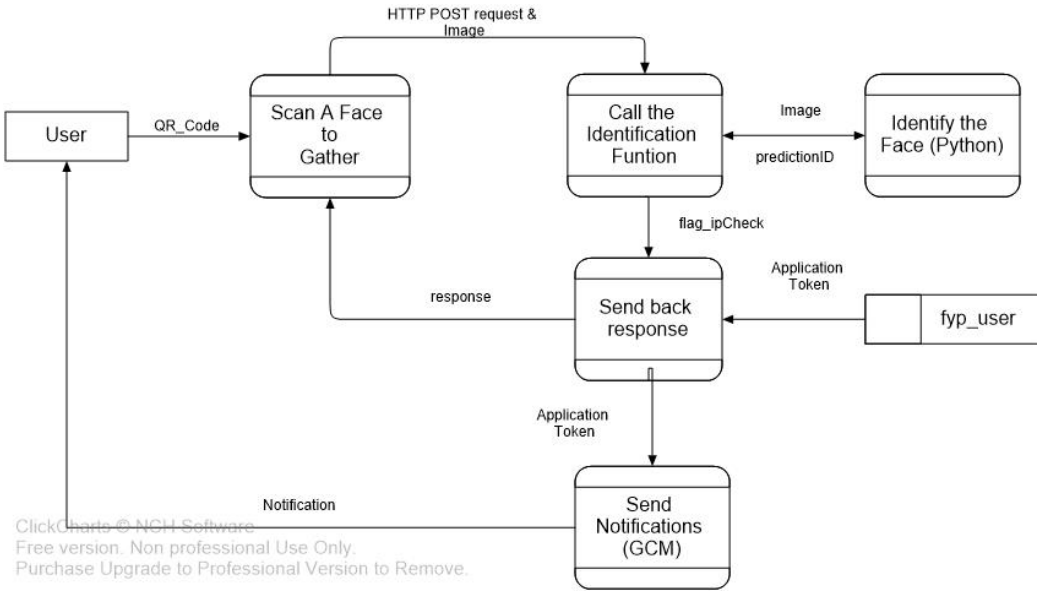


Figure. 3.5.2.1 Data flow diagram of gathering by scanning a face

When the user opens the camera by clicking "SCAN FACES" in gathering page, the user can press the black bar to take a face picture. Meanwhile, this image will be stored in the local memory. Later, the client application will send the HTTP POST request with the face image to the server, and then the server will call the identification file in Python to identify the image. Once the predicated user ID has been returned by the Python identification file, the server retrieves the predicted user's application token from the database, which is used for the GCM server to send a notification to the predicated user.

3.5.3. Theory Explanation of Face Identification Function

In this function, a method for face recognition called "Eigenface" was used for implementation, which is efficient and easy to implement compared to other face identification methods like feature extraction.

The idea of eigenfaces is to represent a collection of faces by principal component analysis (PCA), and generate the feature vector for each face by a reconstructed method [1]. And then use these feature vectors to train an artificial neural network (ANN). As for the prediction, the procedure is to calculate the feature vector from the new image and put it into the neural network model for prediction.

3.5.4. Implementation of Face Identification Function

Based on the method discussed in section 3.5.3, the neural network for face identification was implemented by Tensorflow, which is an open source structure for deep learning. Meanwhile, this part was developed in Python language, because Tensorflow has the excellent capability with Python and Python is power at data processing. The following statements describe how the Tensorflow python files predict the input face images.

- Generate the trained model
    - Read in the images from the face database
    - Calculate the feature vectors from the face collection

    - Define the target labels
        - An array of size of class numbers represents the results. In the matrix, 1 for the correct ID and 0 for the others
        - Define the layers and construct the model
        - Define the layers with random weights and basis by addLayer function
        - Define the network structure with layer number, node number and active function for each layer
        - Define the loss function, as cross-entropy method was used.
        - Define the optimizer, as gradient descent optimizer was used
        - Define the learning rate
    - Run the training session and output the loss
    - Output the test result
    - Save the trained neural network model
- Prediction
    - Read in the target image for identification
    - Calculate the feature vector of the target image
    - Restore the trained model
    - Feed the feature vector to the model
    - Output the prediction result

3.5.5. Re-training of Model

Because the neural network model must be constructed before training with data, it is supposed to be re-trained if new classes join, which will change the model structure. Mainly speaking, in this application, when a new user wants to use the face identification feature, the system shall allow the user to upload the face images and re-train the model to work. In the practical deployment, the server shall re-train the model with new pictures in a reasonable time interval.

3.5.6. Sample Test

3.5.6.1. Assumption

This sample test will test the face identification function which is deployed in this electronic payment system. A face image database of 10 people and ten images for each were used.

According to the model parameters:

- Ten classes and ten images for each
- 60% of images were used for training, and 40 of images are used for testing
- Two network layers
- 50 middle nodes and ten output nodes
- ReLU was used as the active function
- Cross-entropy was used for loss function
- Gradient descent optimizer was used for training
- Learning rate was set to 0.01
- Training iteration number was set to 5000

3.5.6.2. Result of Training

```
0.26331827
0.26330757
0.2632965
0.26328528
0.26327488
The accuracy is
0.975
Saving Completed
PS D:\Files\FYP_Project\Server\identificationFile\Codes>
```

Figure. 3.5.6.2.1 Result of the training example

With environment setting to section 3.5.6.1, the model was trained for multiple times. However, the model was saved, the best result was picked up and shown as [Figure. 3.5.6.2.1].

According to the result, the value of loss function nearly converged to 0.2632, and the final accuracy for test image set was 0.975, which was considered acceptable in a 100-sample test.

3.5.6.3. Sample Result of Face Identification

In this test, the client application took a face photo and transferred to the server for identification. The complete face identification function was tested in this section.

Their photos were taken on a registered member with user ID 1, the following results were got:

- Test Case 1
  - Image:

  - Result:

```
192.168.137.10 - - [09/Apr/2018:04:23:22 +0000] "POST /identification/upload
true
[[6.152719  0.        0.4921527 0.        4.130851  0.3715695 0.
  0.        0.        0.       ]]
1
```

- Test Case 2
  - Image:

  - Result:

```
192.168.137.10 - - [09/Apr/2018:04:24:01 +0000] "POST /identification/upload/
true
[[5.699717   0.05348476 0.6991321  0.         3.6150718  0.00730522
  0.         0.         0.         0.        ]]
1
```

- Test Case 3
  - Image:

  - Result:

```
192.168.137.10 - - [09/Apr/2018:04:26:01 +0000] "POST /identification/upload/ HT
true
[[6.392293   0.26577845 0.94909364 0.         3.6844635  0.
  0.         0.         0.         0.        ]]
1
```

The result contained three parts: "true" mean this identification is successfully made; the matrix showed the final weights of the label, the highest weight represented the predicated ID; the last number indicated the target user ID, which was the predication result.

According to the three test cases, the result was accurate and acceptable, even the target user showed different motions or wore glasses.

3.6. GCM

3.6.1. Introduction to GCM

GCM stands for Google Clouding Messaging, which is used for pushing a message to the end-user device. Because the server is built in request-response mode, therefore, the server cannot send a message to the user if the user doesn't give a request, but some messages are necessary for users like transaction results. GCM takes the responsibility of pushing a message to the end-user when the transaction happened or finished. At this moment, the GCM has been updated to a new system called Firebase Clouding Message which stands for FCM. Both two terms stand for the same meaning.

3.6.2. Flow of GCM Notification

- In android application:
  - Generate the GCM token by a library call
  - Send the GCM token to the server by HTTP request
- In Java server:
  - Retrieve the target GCM token from database
  - Send the HTTP request to GCM cloud server with the token

3.6.3. Implementation of GCM

Besides of using the library in the application, the project was required to register on Firebase [Figure. 3.6.3.1]. The web API key shall be carried by the server, which will be used to send HTTP request to the cloud server for notification.

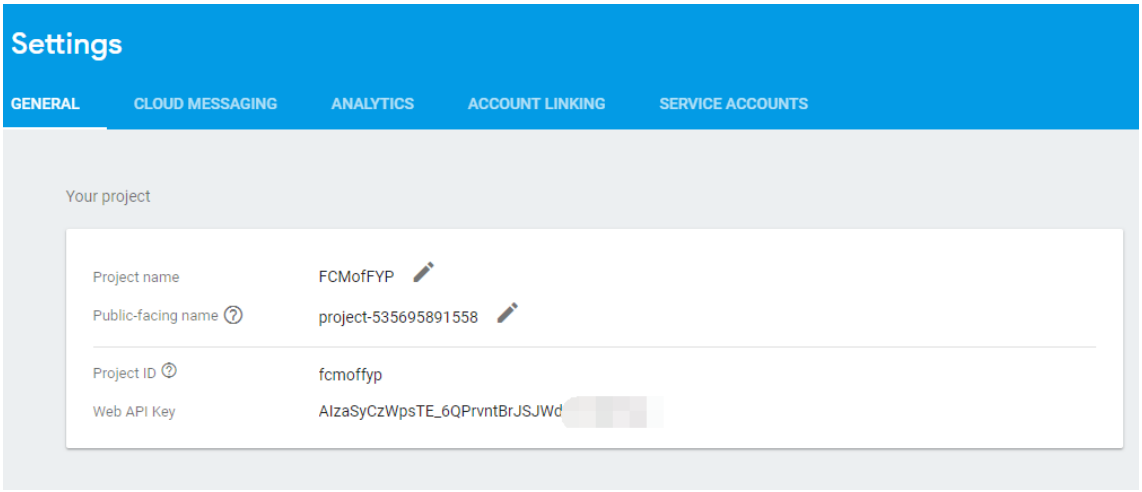


Figure. 3.6.3.1 Firebase page example

4. Limitation & Further Development

4.1. Security

4.1.1. Security of the URL

In this project, HTTP was used, and most of the parameters were transmitted in the unencrypted URL, which means that this method may be under security risks like faking URL. Although the server will check the user's IP address every moment before the transactions are made, it is not exactly safe in some ways.

This part can be improved by updating the communication protocol to more secure ones like HTTPS, or making use device's ID or application ID which can fairly be considered as unique somehow, and checking all the attributes to make sure that the request is sent by the right user.

4.1.2. Security of the QR Code

Although an attribute called QR Code Validation Number was used to prevent some security issues, the QR codes generated by this application can be read by any other applications as a standard QR code.

This issue can be solved by encrypting the data of QR code, and only supports decryptions in the server or in the client application.

4.2. Face Identification

4.2.1. Capacity of the Model

In the test case of face identification function, only ten classes with ten images each

were trained on this model and made reasonable results. However, when changed the test classes for 40, the training time raised up a lot and lost the high accuracy.

It is possible to modify and improve the identification method or justify the model parameters like learning rate.

4.2.2. Incorrect Face Identification Result

If the system gives out incorrect face identification result, the server shall have the ability to handle the mistakes. Although a notification for confirmation will be sent before the transaction, the accuracy shall be even more close to 100%, which makes the face identification function not reliable.

It is possible to make use of the miss-identification data to re-train the model for reducing further mistakes.

4.2.3. Add New Users to the Model

The new user will not be able to upload the face images for training and use the face identification service immediately. Because it is difficult to update the neural network model every moment when new users join.

Therefore, the system can be set to re-train the neural network model once a day or once an hour.

4.2.4. Identify the Unregistered User

Because the output label of the identification model was in same size with number of classes. Therefore, the model will make predictions on any images pushed in, even the images belonging to unregistered users. Furthermore, the model will even make predictions on any image passed to it, no matter it is a face or not.

This issue shall be solved by firstly, adding human face detection feature to the camera, which guarantees the images are human faces. Secondly adding extra classes for unregistered faces may handle the issue.

5. Conclusion

The project “Mobile Web Application – Electronic payment System” focusing on a complete application system design and developing. The final product application of FYP project aims at providing various and comfortable payment experiences, including QR code payment and face recognition payment. Especially the face recognition function, which is assumed to be a new feature applied to portable devices. During the developing progress, different technical methods are implemented like Android and Java for programming, RESTful API structure for system structure and neural network technology for face recognition. The product application works well for all the basic functions and different payment methods.

However, compared to existing commercial applications in the market, there’re limitations of FYP project in fields of UI design and security issue. Furthermore, for the future development, besides UI improvement and security issue, it will be significant to give out better QR code payment method and performance of face recognition function in electronic payment system including capacity and performance.

6. Appendix

6.1. System Requirements

6.1.1. For Mobile Device

- Android 5.0+ is preferred
- Google Play Service is installed and switched on, Ver 12.5.21+ is preferred

6.1.2. For Server

- Windows 7+
- MySQL is installed, and Ver 14.14 is preferred
- Python 3.6 in installed
- Tensorflow is installed, and Ver 1.1 is preferred

7. Reference

[1]. M. Turk and A. Pentland (1991). "Face recognition using eigenfaces" (PDF). Proc. IEEE Conference on Computer Vision and Pattern Recognition. Pp. 586–591.