

# DOOWA

Wan

Dept. of Computer Software  
Hanyang University  
Kuala Lumpur, Malaysia  
aus.baik@gmail.com

Amin

Dept. of Computer Software  
Hanyang University  
Kuala Lumpur, Malaysia  
amin.sharudin@gmail.com

Megat

Dept. of Information Systems  
Hanyang University  
Johor, Malaysia  
megattmf@gmail.com

**Abstract**—After living in the COVID-19 pandemic for approximately 2 years now, we decided that we want to make something that can alleviate people suffering during this pandemic. We came up with an Android mobile application called ‘DOOWA’. The name DOOWA is derived from the Korean word ‘*deobda*’ which means to help. The main function of our application is to connect individuals or families in the community who required assistance to survive during this pandemic to a donor in the community who are willing to lend a helping hand to reduce their hardship. We are hoping that this app will facilitate and encourage members of the community to help each other during this troubling times.

**Index Terms**—DOOWA, Android Application, Assistance, Donation

## I. INTRODUCTION

### A. Motivation

We focused on people who are financially less fortunate to be seen by the community. We created this mechanism to have a balanced community and people can reach out and seek help from others easily. There are families that are hugely affected, and many suffer from Covid-19. As much as we would like to get rid of any bad situations, it is still unavoidable and usually unexpected. The nature and consequences of these situations can vary significantly and in worst cases can also be life threatening. Therefore, we think that it would be nice to have some mechanism by which we can notify and get notified by certain people about such circumstances and increases the chances of giving and receiving help as soon as possible.

The need for such a mechanism increases even more as in this era of technology, platforms exist to support them. One such platform and a very common one in that a mobile application. Almost everyone today has an access to mobile app as they are easy to use and can be accessed by phone and tablets. Hence, this motivates our team to develop a mobile application for giving and receiving help in the community.

### B. Problem Statement

- Donor receivers are difficult in reaching out for help as they do not have access with other donors physically.

- Some receivers are not able to find the type of donations that they really need. As a result, the donation given is wasted.

- It is hard to contact between donors and receivers as they do not know each other before.

- We acknowledge that people in need can also request help from the government, but often times this help have to go through a long, bureaucratic application process before they are approved.

TABLE I  
ROLE ASSIGNMENTS

Roles	Name	Task Description
User/ Customer	Megat	Act as a beta tester to criticize and provide suggestions from a client point of view. They will provide how the application UI should look like to satisfy the ease of access from a user perspective. They also should test every feature in the app to identify bugs and report to the software developer.
Software Developer	Wan	The software developer should have the general point of view of how the application works overall. They will provide the services for backend server and database as well as providing the general UI of the frontend.
Development Manager	Amin	Development manager will be the main overseer of the project development, and gathers the information from the client side and handles the reports. They also will be the main proofreader for the project documentation to meet the project specification.

### C. Research on any related software

#### 1) KakaoPay

KakaoPay is a mobile payment and digital wallet

service by Kakao based in South Korea that allows users make mobile payments and online transactions easily. It ensures smooth operations where users can make payments or do a money transfer via KakaoPay handily and there is no floating time for recipients to receive the money. It also lets users to invest with a small amount of money, get a loan for a house, and find the perfect insurance partner. Users can also save any credit or debit card information on it so that they can make one-time payment easily without filling in payment information once again with just only one tap. The service also supports contactless payments where users can send an amount of money to anyone they want to. Users can also notify recipients if they have successfully transferred the money and vice versa.

2) *PayPal*

PayPal is one of the world's largest payment services that is secured with advanced technologies. PayPal offers a worldwide payment service and supports Visa, MasterCard and so on. Users can sign up for PayPal account to have an extra level of security and fraud prevention with a quicker payment option and save payment details for future transactions. PayPal also lowest transaction fees for a global transaction, therefore users can freely use any card they prefer to use. Besides, it also offers reward points for each successful transactions that can later benefit users to transfer money wirelessly with even lower transaction fees. PayPal also has its own digital wallet that users can put money in so that users can directly transfer money without entering one's account numbers every single time.

3) *Yogiyo*

Yogiyo is a food delivery service application which enable users to get their food delivered at their doorstep from various restaurants easily. The application connects users to a variety of restaurants from different cuisines such as Western, Korean, Japanese and Chinese. One of the features of the application that we want to emulate is their delivery tracking feature. Through the application users can know the location of the delivery food rider in real time. Rider's location is represented by an icon on the map in the application, and the icon moves in relation to the rider's location. This is the feature that we want to have in our application so that whenever a meeting is set up, one person can know the location of the other party in real time and plan accordingly.

4) *Coupang*

Coupang is an online shopping application based in Seoul, South Korea that sells products from a wide-ranging category including food, clothing, fresh produce, baby products and many more. User can

shop online in the comfort of their own home and have the products delivered on their doorsteps. In the application, users can know the current location of their parcel through an icon. Every time the parcel moves from the seller to the warehouse or currently in delivery, each and every stage of this process is shown to the user so that they can have the assurance that the product that they buy will arrive. We want to do the same thing with our application in case a donor wants to send products through the postal service.

5) *Google/Facebook Account*

Google and Facebook account is something that the majority of people have. We have also seen a lot of application nowadays which requires first time users to make an account if they want to use the services provided by the application. Similar to this, our application will also require first time user to register an account with us. However, instead of filling in their details one by one, we will allow users to use their already existing Google or Facebook account to register on our application. This will ensure smooth registration process and provide a hassle-free service to our users.

6) *Sambal SOS App*

The effects of pandemic have hit Malaysia in many aspects that this has increased the suicide rates at an all-time high. Therefore, Sambal SOS app or know as "White Flag" is an application that is programmed by three Malaysian students where the purpose of the application is to connect Malaysians who are in need with those who can help. Users can also access through web application and log in with Google or Facebook account. This application is also implemented map where users can give their hands easily depending on their location. Donator and recipient can immediately connect on this app to have further details regarding their needs. Recipient can also fill in their details and what type of donations they need so that donator can have detailed information before giving out donation. This application has a basic interface where all generations are able to use the features on the application at ease.

7) *SirenGPS*

SirenGPS integrates emergency management tools with real-time visibility and interoperability for emergency managers, first responders, and stakeholders in your community. This application also has a Siren Alert feature that can send real-time messages to specific groups, locations, buildings or your entire community. Siren Alert not only allows you to inform individuals of a crisis in their immediate vicinity, it lets you warn people as they approach a threat and steer them toward safety. With Siren Alert, you can also enable individuals in your community to respond back in real-time. Our

application plans to integrate the Siren Alert feature and enable anyone to alert any person nearby for donation.

#### 8) *ShareTheMeal*

This application is launched by the United Nations with the aim of feeding the needy especially with foods. This app is available in about 9 languages, including English, Spanish, French, German, Italian, Portuguese, Russian, Korean, and Japanese. Besides, this app gives users a direct channel to fund UN efforts in supporting children in need, on the go, with just a single tap. It only costs 0.50USD to feed one hungry child a day, plus users can track where the meals are distributed and the impact of the donations to the needy. ShareTheMeal application is an extension of the ShareTheMeal non-profit initiative run by the World Food Programme (WFP) which is the world's largest humanitarian agency combating hunger globally and providing food aid to an estimated 80 million people each year. Therefore, this application covers a wide range of location where people in need get to seek help easier.

## II. REQUIREMENT

### A. *Functional Requirements*

- 

#### **Account Creation**

The application should have an account creation system for the users before enabling them to access all the features on the app. On the main screen, there should be a button for sign up option. User will have to enter their email and password and these data will be stored in backend database. In addition, all email address should be unique and users cannot sign up with an already used email address. There should be no specific requirement for password but the application should display the strength of the password inputted by user. Upon completing the registration, a verification email should be sent to the new users' email address. All users will also have unique ID bind to their account.

- **Login**

Upon opening the application, the main screen should display the login menu with blank space for the users to input their email address and password. After the users entered their credentials, the inputted info will be sent to the backend server and it will search for matching email address and password combination in the database. In case of successful login, the application will inform the user that the login was successful and the user will be sent to the main page of the application. In failure case, the user will be rejected by the system and prompted to enter the correct combination. Alternatively, the application will provide the users another option to login via Google, Facebook or Twitter.

- **Chat Bot**

The application should provide a chat bot service to the users which let the users to ask questions and set a reminder. This chat bot should not use an AI approach, but rather a very simple approach where the users just type in the command and arguments and the bot will reply accordingly. The chat bot should have various kind of features and commands that covers all the users' needs.

- **Maps**

The application should have a map as the main interface. To implement this, the Google Maps API services will be used. On this map, it should show all the locations of the other users that are requesting for donation and nearby foodbanks. If the user is a donor, the map should show the location of where the donor is giving out the donation, the time available and the type of donation which will be indicated by an icon on the map. If the user is a receiver, the map will display their location for another donor to come and visit them and the severity of their desperation which will be indicated by color hue of their location. The locations of the two types of users should be indicated by the common map pin with different color each. If the location is an organization type donor, a special pin should be used.

- **Online Banking**

Donation can be made by physically or online transfer for money. The application should provide an online banking service for the users which let online money transfer between the users. Services such as Paypal, FPX banking and KakaoPay should be implemented.

- **Points for Donor**

Setting up points for donor every time they made a donation. This kind of grading system will allow us to identify frequent donors who will be rewarded accordingly with coupons based on their numbers of contribution. Donors will also be put into groups/level according to the points they accumulated.

- **Notifications**

The notification system in the app will pop-up whenever the help that you requested are answered by a donor. This will help the requester to be alert at all time.

- **Donor Rating**

The app will have a rating and review system for its donors and aid receivers. An aid receiver can rate their donor based on a 5-star rating system and also leave a review after they have received the help that they required. This system is put in place in order to prevent scams, and accounts who does not follow our community guidelines.

- **Reminder**

The app will also have a notification system where you can set, to remind you of the date and place of meeting to

receive or deliver your donation. You can set the notification to pop up at 2 hours or 30 minutes before your meeting time.

- **Chat Room between Donor/Receiver**

The chat function between the donor and the receiver is important so that they can get more detailed information on what kind of help that they require. They can also use the chat function to set up time and place to meet if necessary.

- **Donor Tracker (postage/meeting)**

We believe that any form of help is important and should be facilitated. Therefore, in a case where the donor wanted to send the required items through postage, we will have a function where the app tracks the position of your postage. This will help the receiver to identify the location of the items that they so direly need at all times. This function can also be modified to tract the location of individuals on the way to the meeting spot which they have agreed upon beforehand.

- **Set a Face-to-Face Meeting**

This allows donators to have face-to-face meetings with the recipients within a specific time.

- 1) Click the “Time” to pop out the clock to set the meeting time.
- 2) Click the “Date” to pop out the calendar to set the meeting date and day.
- 3) Past dates cannot be clicked as there will be an error message.
- 4) The furthest appointment that can be made is one month ahead. Hence, user cannot make an appointment that is more than one month ahead.
- 5) There are no multiple appointments in one meeting. Therefore, if a date has been chosen for a meeting, user cannot reserve any other dates before cancelling the previous chosen date.
- 6) The dates must be agreed by both donator and the recipient. Click “Accept” button to agree the dates and “Reject” button to inform that one is not available for the chosen date.
- 7) If both parties agreed to the chosen dates, they then move to the confirmation page.

- **Choosing Types of Donation**

This allows donators to categorize their types of donations so that it is easier for recipients to seek for help with a specific type of donation.

- 1) Donators and recipients can choose which type of goods is important to them. This includes money, clothing, foods, baby items, gadgets, home essentials and others.
- 2) If Others is picked, user can state the type of donation that will be give or needed.
- 3) Once the type of donation is picked, then it will only show donator/recipient that have the same type of donation.

- 4) This will help donators and recipients to discover those who have the same interest.

- **Pictures Upload**

This allows users to upload pictures of themselves and the donations for confirmation and proof so that it will make the donation more precise and clearer.

- 1) Click ‘Camera’ button to connect the application to the camera.
- 2) The pictures taken are going to be stored and shared in the user’s profile for others to confirm them.
- 3) Click ‘Trash Bin’ button to delete pictures on the profile and there will be a pop-out whether to confirm the deletion.
- 4) Click ‘Sure’ for confirm deletion, and ‘Cancel’ to cancel the deletion request.

- **Guidelines and Manuals**

There is a guideline check for donators and recipients to follow to prevent any misbehavior and illegal actions throughout the donation process. This helps users to use the application properly.

- 1) Click ‘Guidelines’ to view the guidelines of the application that help users to understand better about the application operation.
- 2) For first-time user, click ‘Agree’ to accept and follow the guidelines given.
- 3) Click ‘Disagree’ if the guidelines if the user does not accept the guidelines of the application.

- **Report Button**

All entries from donor and receivers should comply with the terms and agreements of the application. Hence, for every entry, a report button must be provided. Whenever a user sees any other donor/receiver entry that does not comply with the guidelines, the user can use the report the button to alert the developers. This report feature should provide text box for what kind of terms violated. The report will be sent to the backend database and will be reviewed by the developers. Every report should have a unique ID and the severity point of the violation. The developer then will review the reports starting from the entry with the most severity points.

- **Terms and Conditions**

This will provide legal agreements between us and people who are using our application. Users must agree to abide the terms of the service (application) to use it. This can also act as a disclaimer for users to understand and aware of the terms and conditions that are applied.

Users must acknowledge that the application is collecting names, addresses, credit card information or other personal data from the users. The data given may be used, stored, and shared. Besides, any misbehavior and illegal actions would result in termination of the user accounts.

- 1) Before logging in, user is opted to read and understand all the terms and conditions in the application.
- 2) User must scroll through all the terms and conditions before clicking the 'Agree' button.
- 3) 'Agree' button will pop out if the user has scrolled through the terms and conditions.
- 4) Click 'Disagree' button if user does not accept the terms and conditions.
- 5) If the user clicked 'Disagree' button, user is unable to log in and eventually use the application.

#### B. Non-Functional Requirements

- The application should have dark mode UI.
- The application should ask for satisfaction rating periodically
- The application should be simple enough to be accessible to elder users.
- The application must detect the internet connection before giving access to the users.
- The application should have an option to change language between English and Korean.
- The application must be ready before the final week of class.

### III. DEVELOPMENT ENVIRONMENT

#### A. Choice of Development Platform

##### 1) Platform used

We have decided to use the latest Windows 10 and MacOS as our main development platform for this project. Windows 10 is a good choice for coding because it supports many programs and languages. In addition, it has significantly improved over other versions of Windows and comes with various customization and compatibility options. There are also many advantages to coding on Windows 10 over Mac or Linux. It also provides great security features, easy to upgrade and supports a huge range of programming language such as PHP, Android and XML which are some of the language that will be the main backbone of our program development. On the other hand, MacOS is a Unix-based operation system and is popular choice nowadays for programming. Programmers who work on a lot of back-end web server code often like MacOS for their personal computer, because it's based on Unix and easily runs nearly all Linux software.

##### 2) Programming Language

- Java

We have decided to use Java as our main programming language for our project development; about 70 percent of it. We have chosen Java because first of all, the official language for Android development is Java. Large parts of Android are written in Java and its APIs are designed to be called primarily

from Java. Plus, every beginner programmers, including us already have some experience in creating programs using this language. Moreover, this language is object-oriented which we think is for easier to use for developing an Android application since Java provides a various choices of GUI interface services. Moreover, Java provides a lot of libraries that can be used a lot for our program development such as Listeners, Fragments, Events, and much more. This language will be used for the frontend side of our program which handles the UI action such as the buttons actions after you clicked them or how the input by the users will be handled.

- XML

Extensible Markup Language (XML) is a great choice to be used in developing our application interfaces. We decided to use XML mainly because it is the main language that is being used by the Android Studio for UI designing and various widgets. Moreover, XML is designed to store and transport data and organize its data as a structure. Many UI frameworks use XML as the language to make the UI and once you understand values and attributes, you know enough to look at an XML file and understand the data within.

- PHP

We have decided to use PHP as our main programming language for the backend server to interact with the database. PHP is a popular general-purpose scripting language that is especially suited to web development. The reason why we are choosing PHP for this project is because our backend server will be communicate with the database through the Internet and we need to provide a universal domain which enables our application to fetch data from database while being connected to any type of Internet connection. Moreover, there are a lot of ready made PHP code that provides various services which would be helpful to be integrated into our program.

- SQL

We will use SQL (Struted Query Language) for managing our database. It is a domain-specific language used in programming and designed for managing data that are in our relational database management system. It is particularly useful in handling structured data so this makes it possible to process data such as user accounts, donation requests, list of foodbanks, and map markers.

### 3) Cost Estimation

TABLE II  
COST ESTIMATION

Roles	Description	Cost
Android Studio	Frontend Code Editor	0
Visual Studio Code	Backend Code Editor	0
Github	Remote repository	0
Heroku	Backend Server	0
MySQL	Backend Database	0
Overleaf	Documentation typesetting program	8 USD/month
Google Meet	Video Conference software	0
KakaoTalk	Chat application	0

### 4) Information of Development Environment

- Android Studio



Android Studio provides the fastest tools for building apps on every type of Android device. For frontend, we can create complex layouts in a short amount of time because the layout can be designed by using the drag and drop feature provided by the IDE which will create the code by itself with zero compile error. After designing, the programmer can immediately see the result of the interface and how it will be look on our phone screen. Furthermore, it provides a fast and reliable Android phone simulator that can be used to preview your application on the phone right after the code is compiled. With this, we do not have to build the APK file and install it onto our phones every time we want to test the application. Finally, Android Studio can be run in Windows and MacOS which is perfect for our development environment.

- Visual Code Studio



We decide to use Visual Code Studio as the source code editor for PHP and SQL which will be handled by the backend because they have a built-in support and plugins for both of the programming language. IT also have the auto compile feature which can ease the process of testing our

application without having to compile every time the code is edited.

- Github



Github is one of the most popular website to upload our sources code seamlessly as well as share and develop the program with all the teammates. Moreover, our teammates can easily check the commits done by other members and implement necessary update of the features. Github provides necessary management functions for software development such as basic functions of the Git which includes functional requests, task management, and bug tracking.

- Heroku



The reason why we choose Heroku as our backend server is because it provides a lot of services for a free price such as modules and ClearDB service which will be used as the direct connection between the server Heroku and the MySQL database. Furthermore, our team members are more well versed with the Heroku services which would speed up our application development much faster.

- MySQL



We will use MySQL for data management because as mentioned before, it can be directly connected with our Heroku server. Today, MySQL is the second ranking RDBMS solution in the world, according to DB Engines. Its users include a wide range of websites and applications, including household brands like Spotify, Netflix, Facebook and Booking.com. Finally, the data stored there can be easily viewed in tables.

- Goole Firebase

We are using the Google Firebase as our second database to handle the authentication data if the user decides to



use Google sign in service. By using Firebase, we can extract the data of the users' Google account such as profile picture, name, email address to be used in our application. Furthermore, Firebase also supports cloud storage which will be used to upload images that are submitted by the users whenever they made a submission. After upload, the image link will be extracted and sent to our primary MySQL database.

- XAMPP



Xampp is a web server that hosts PHP websites. Furthermore, XAMPP has a built-in MySQL database setup. It also has a GUI for MySQL. As a result, when you install XAMPP server on your PC, the MySQL database is also installed. Which means you won't need to install a separate MySQL database server. XAMPP is required for PHP websites, but not for JAVA. However, XAMPP is required for MySQL databases.

- Overleaf



We decided to use Overleaf as the platform to complete our project documentation. It provides a collection of useful templates for various use cases that you can simply open in a new project, or download to use offline. Moreover, it also provides the convenience of an easy-to-use LaTeX editor with real-time collaboration and the fully compiled output produced automatically in the background as you type. The downside is the real-time collaboration feature is a premium feature so we will need to pay a few amount for it which is already stated in our cost estimation section (Refer Table II).

- Google Meet

Like every other app development project, we would need a platform for meeting and discussing our development process. Hence, we decided to use Google Meet platform to conduct meetings every week and discuss about our current



development, assignments and future plans.

- KakaoTalk



We decided to use the KakaoTalk chat platform to communicate with each other about meeting time, asking questions and sharing documents on the go.

#### B. Software In Use

##### 1) Google Map API

Our application will have a map as its main backbone which will show up on the very main page after the user has logged in. So, we decided to use the Google Map API service to implement the map features into our application. The reason for this choice is the API key is not only free, but it is also Android Studio friendly; the Android studio provided various widgets and functions for the developers to manipulate the map such as adding markers, getting user current location, getting the distances between two places and much more.

##### 2) Social Media Login API

Our application will provide an alternate way for the users to log in; Google, Facebook and Twitter Login method. This way, users will not have to undergo the hassle of making a new account before using the application. Furthermore, users who logged in by these alternate methods will have their credentials handled by the respective services instead of our own server which grants them more security and safety.

## IV. SPECIFICATIONS

### A. Account Creation and Login

#### 1) Login Page

Whenever a user first launch the application, they will be greeted by the login page of the application (refer Fig. 1). There will be a huge application name "DOOWA" on top of

TABLE III  
ROLE ASSIGNMENTS

Name	Task
Megat	Documentation, Beta Tester, Debugger
Wan	Backend handler, Database, API, Maps implementation
Amin	UI design, Frontend, Layout programming, Translator

the page and under it will be 2 text spaces which will be used to receive input from the user for their username and password. For password textbox, the characters inputted will be privated and changes to dotted characters. Under these 2 text spaces, there will be a button to be clicked after the user filled their login credentials into the mentioned text spaces. Additionally, under the button, there will be 2 more smaller buttons, namely "Forgot password?" and "Register" which carries their specified task that are explained later .

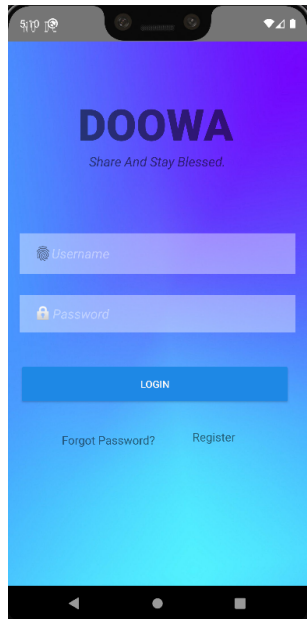


Fig. 1. Login page; first page upon app execution.

- Empty username and password

When the "LOGIN" button is clicked while the username and password are empty, a warning will be displayed onto the respective text box which is currently empty and requires the user to fill up the text box to remove the warning.

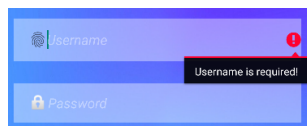


Fig. 2. Username textbox is empty.

- Both fields are filled and LOGIN button is clicked

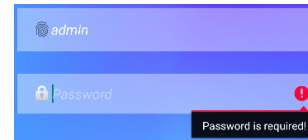


Fig. 3. Password textbox is empty.

When the user inputted both username and password text boxes and clicked the "LOGIN" button, the program will use the "startPut()" method provided by the JAVA Class "PutData" and send the inputted data to our server. Our server will then check whether the combination is correct and then the result will be send back to the frontend side and inform the user whether the username and password combination is valid or not. While all of these process are ongoing, a circular loading bar will be shown at the center of the screen indicating the checking process in ongoing (refer Fig.4) . The loading bar will be gone after the checking process is completed successfully. After that, a message will appear the the bottom of the screen indicating whether the login is successful or not.

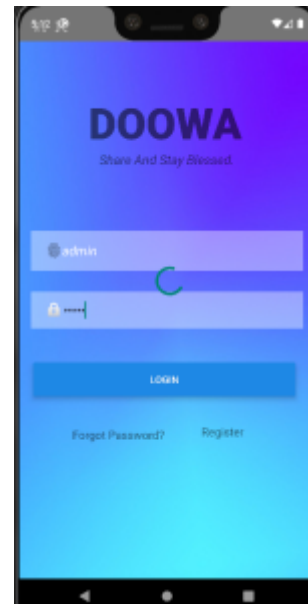


Fig. 4. Loading bar pops up while checking username and password.

- Wrong username and password combination

If the user inputted the wrong combination of username and password, after the LOGIN button is clicked, a message will appear at the bottom of the screen showing "Username or Password wrong" and the application will stay on the login page until the correct combination is inputted.

- Correct username and password combination

If the username and password provided by the user is valid, the user will be able to access the main page of the application.



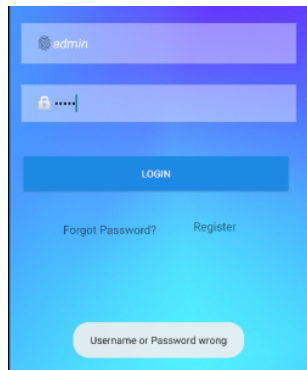


Fig. 5. Wrong username/password.

A message also will be displayed at the bottom of the screen saying "Logged In successfully!".



Fig. 6. Correct username/password.

## 2) User account registration

From the login page, there will be a button called "Register" which allows the user to create a new account for the first time. On the registration page, there will be 5 text fields for the user to input their details which are full name, username, email, password and password confirmation. Under the fields will be a button "Register" which will send all the inputted user details to the server when clicked. Under this button, at the bottom of the screen, there will be a text saying "Already have an account? Log in.". When users click this text, the application will simply send the user back to the previous login page.

### • Full name

Full name can be any text. If this field is blank when the "Register" button is clicked, an error will pop up on the field.

### • Username

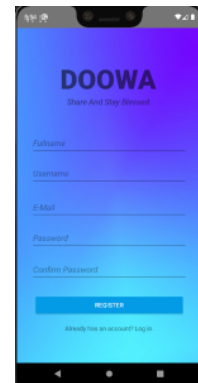


Fig. 7. Registration page.

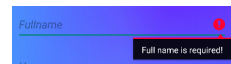


Fig. 8. Blank full name field error.

Username can be any text. If this field is blank when the "Register" button is clicked, an error will pop up on the field.

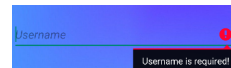


Fig. 9. Blank username field error.

### • Email

Users must provide a valid email in this field. If this field is blank or invalid email address is used, an error will pop up on the field.



Fig. 10. Blank email field error.



Fig. 11. Invalid email error.

### • Password and Confirmation

Users must provide a password that is at least 6 characters long. If this field is blank or invalid password is used, an error will pop up on the field. Same goes to the password confirmation field. If the string entered in both password and password confirmation field are different, an error will pop up on the password confirmation field.

### • Successful Registration

Upon successful registration, user will be sent back to the login page of the application. Additionally, a message saying "User has been successfully created!" will be shown at the bottom

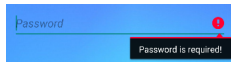


Fig. 12. Blank password field error.

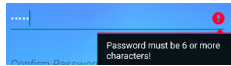


Fig. 13. Invalid password error.

of the screen. Now, user can use the credentials registered just now at the login page.

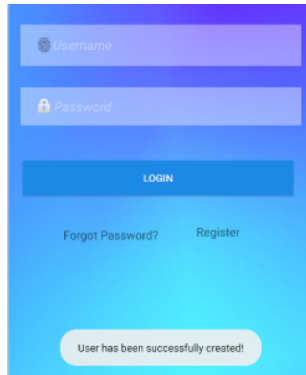


Fig. 14. Successful registration.

#### • Failed Registration

Failed registration can only be occurred whenever a user tries to register using a username or email address that already has been registered. If such occurs, after clicking the "REGISTER" button, a message saying "Username/e-mail address already in use! Please try again." will be displayed at the bottom of the screen.

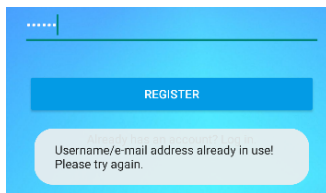


Fig. 15. Failed registration.

### 3) Sign In with Google Authentication

Additionally, users can also sign in into our application by using their existing Google account. Using this, users do not need to undergo the hassle process of creating new account for our application. If users uses this alternate sign in method, the user full name, username, and profile picture will be used instead for the account. With this method, users can feel more secure of their account and they are linked and administrated by the Google itself. To implement this feature, we used the Google Firebase authentication service and implemented the API into our application. Through the Firebase database, we

can identify which users are logged in using the Google sign in method. Finally, users only need to sign in once through the Google authentication and the next time users open the app, they will be automatically signed in.

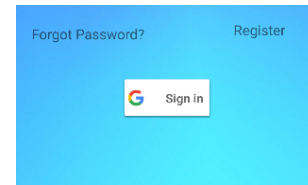


Fig. 16. Google sign in button.

### 4) Password retrieval

Sometimes user might forgot their password from time to time. On the login page, there will be a small clickable text "Forgot Password?" which can be used by the users to reset their password. Users only need to provide their email address which was used for account registration into the field provided. Under the text field, a button "SEND EMAIL VERIFICATION" will be provided and whenever clicked, an email containing the hashed code of the user password will be sent to the inputted email. Failure will occur if the field is blank, invalid email format or the email is not found in the account database.

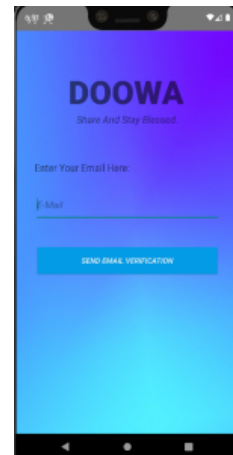


Fig. 17. Password retrieval page.

## B. Map Services

### 1) Implementing Map into Application

Whenever a user logged in into their account, the first thing that should be displayed on the screen is the map which will shows various markers indicating the locations of the people who are requesting donation. Therefore, the most important thing for our application would obviously be implementing map into our application. We decided to use the Google Maps API services because not only it is free, but is is also Android Studio friendly which will be providing

various amount of functions and widgets that can be used to manipulate our map interface.

- Using Google Maps API

First we will go to <https://console.cloud.google.com/> and click "Create Project".

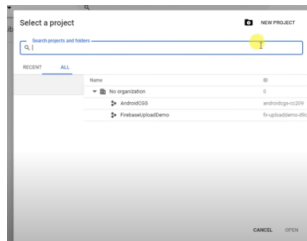


Fig. 18. Create project.

After entering our project name, click the "Create" button and now our project is finally created in the Google console and ready to be provided with various APIs.

Then, on the search bar, search for "Maps SDK for Android"

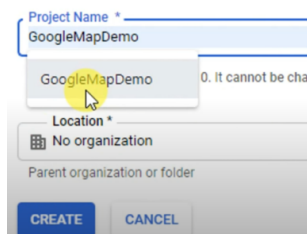


Fig. 19. Enter project name and create.

and click the "Enable" button to enable the API service for our project. Click on the "Credentials" from the tabs above

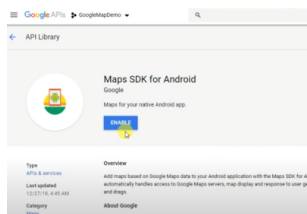


Fig. 20. Enable Maps SDK for Android.

and click on the "API key" to start creating the API key for our Google Maps. Later, API key will be successfully created and we can copy the key to be pasted into the project manifest file.

Finally, we can see the map being displayed in our application as the main display whenever user logs in.

- Map Markers

On the map, there should be multiple markers that are indicating the location of other users which are requesting donations. These markers has different icons according to the donation type. If the donation type is "MONEY", then the

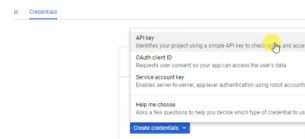


Fig. 21. Create API key.

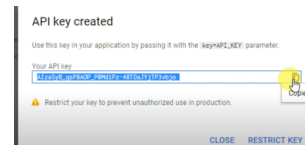


Fig. 22. Copy API key and paste into project manifest file.



Fig. 23. Main map display.

marker should be replaced by money icon. The same rule will be applied for the other donation types. Whenever a marker is clicked, a label showing the details of the request such as the name of the requester, the type of donation needed and full address of the location should be displayed. If multiple markers are showing up at a very close distance, the markers should be grouped as one to make it easier for another user to click the overlapping markers.

- Details upon clicking markers

As mentioned before, upon clicking any markers on the map, the application will bring the users to another page that will display all the details of the request such as the full name of the receiver, type of donation requested, time and date of request, and much more. On the page, there will be a call and message button which can be used by the donor to either call or message the receiver. Whenever these buttons are clicked, a pop up will appear at the bottom of the screen, showing multiple choices of applications which the donor prefer to make a call or send a message. Another button with the icon "navigation" also should be added on the details page. When this button is clicked, the application will launch the Google Maps application to show the location of the marker inside that app which will provide the navigation available to reach the location.

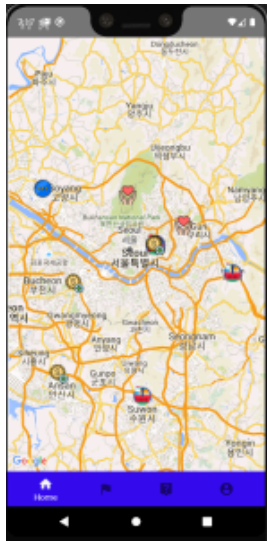


Fig. 24. Multiple markers shown on map.

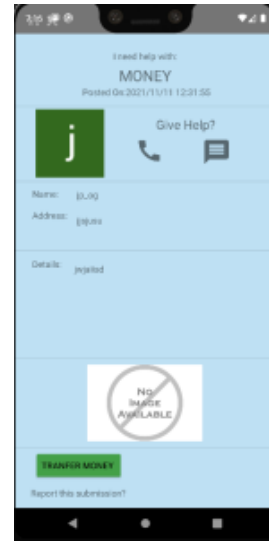


Fig. 25. Details of donation request.

- Marker removed upon completed donation

On the requester side, whenever they are satisfied with the donation arrived, they can click the "Donation completed" button on their donation request page. Clicking this button will cause the marker of the request location to be removed from the map and database.

### C. Details Page

As mentioned before, Whenever the users clicked the markers on the map, the details page of the donation request will appear. This details page will display all the necessary information of the requester such as their account username, profile picture, full name, address, explanation of the donation request, time and date and more.

At the top of the page, the type of donation will be displayed. The type of donation can be either "Money", "Necessities", "Food" or "Others". Underneath it is the profile picture of the account of the requester. Besides it, there are two buttons for call and message. If any users clicked either of the button, the app will direct the user to another application accordingly.

- Call button

Whenever the users clicked this button on the details page, the application will immediately make call to the phone number provided by the requester.

- Message button

Whenever the users clicked this button on the details page, the application will immediately open up messaging app for the user to send message to the requester.

At the center of the details page will be filled by the details of the donation request as mentioned before. Additionally, at the bottom of the screen, the image uploaded by the requester

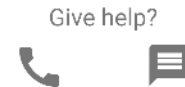


Fig. 26. Buttons for contacting the requester.

will be shown. Moreover, there will also be another button "Transfer Money". With this button, any user will be directed to the banking app which will enable them to transfer money through online banking to the bank account number provided by the requester. Finally, at the very bottom of the page, users can click the "Report this request?" button if the request is inappropriate, suspicious or does not follow the donation request guidelines. Users can state the reason of the report and submit them.

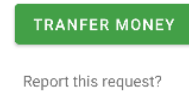


Fig. 27. Buttons for transfer money and report.

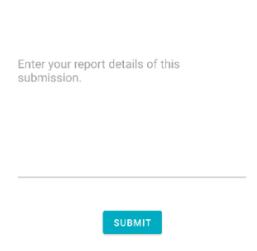


Fig. 28. Submission report page.

#### D. Request Page

Users can request their donation on this page. Users will be asked to fill up few details before proceeding to submit their donation request.

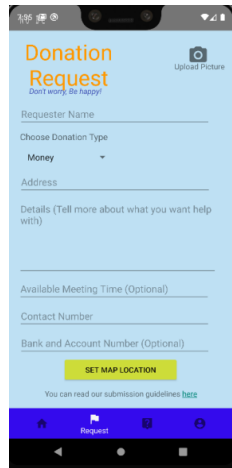


Fig. 29. Donation request page.

- "Donation Request" text
  - This is the title of the page to indicate users that they are accessing the donation request page and its details.
- "Don't worry, Be Happy!" slogan
  - This is the short slogan placed below the title page to motivate and desire users to be not ashamed for not being fortunate.
- Camera icon
  - This lets users to upload an appropriate picture regarding their needs. There is only one picture can be uploaded at a time.
- Upload Picture text
  - This is an indicator for users to notice where to upload picture regarding their needs.
- Requester Name section
  - Users must fill in this section with their correct full name. They may fill in this section with alphabets and numbers. The full name may not be too long exceeding the space provided.
- Choose Donation Type section
  - Users can click this section to have a pop-down menu where they can go more details regarding what type of donation, they are giving by choosing the options such as "money", "necessities", "groceries", and "others". Users can select 'others' if the type of donation is not available among the given options.

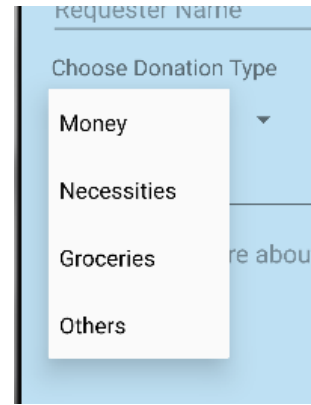


Fig. 30. Donation type choices.

- Address section
  - This lets users to fill in their full address of their home so that people will get the information of where they can provide the donation. In this section, users may fill in with alphabets, numbers, and also special characters.
- Details section
  - Users can add more details regarding their needs such as their necessities, how many family members are in need and the specific things that are needed. This section provides a space for users to add any more details that seem important for people to know that have not been informed in other sections.
- Available Meeting Time section
  - This allows users to inform the time that is most suitable to provide donations or visitation by filling in when is the available time so that people can get aware and plan when the best time is to visit them. This field is optional.
- Contact Number section.
  - This is the section for users to fill in their contact number in case of people want to reach out to them. This section lets user to fill in their contact number with numbers and special characters. It is preferable to have at least one contact number to be filled in.
- Bank and Account Number section
  - Users can fill up this field with the details of their preferred bank account and the number. Using this will let other people to give donation through bank transfer without the hassle of meeting up. This field is optional.
- Set Map Location button
  - Users can press this button to navigate to the real map page where they can indicate the pinpoint on the map of where is exactly the location of their location. This helps to strengthen the address section where it lets people to confirm where the

requester is by looking at the map. Clicking this button will bring the users to the map selection page to pin their selected location for their donation request.

- Guidelines

At the very bottom of the screen, there will be a link that will redirects the user to the guideline page on their default browser. Users can refer to this guidelines and take extra precautions so that the submission are complying with our submission guidelines.

#### E. Food Bank Submission

On this app, users can also open up a donation spot where other people who are in need can come by themselves to receive help needed. For that, user will be able to submit a submission for their open donation on the "Foodbank Submission" page. The food bank submission page is very similar to the donation request page with few different attributes.

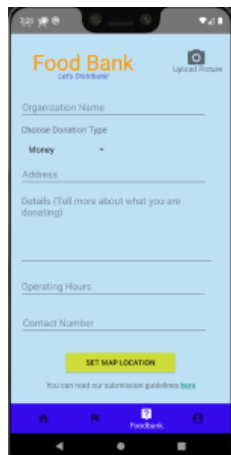


Fig. 31. Main page for food bank submission.

- "Food Bank" text

- This is the title of the page to indicate organization (users) that they are accessing the food bank page and its details.

- "Let's Distribute!" slogan

- This is the short slogan placed below the title page to motivate and desire users to be part of the food bank's activity.

- Camera icon

- This lets organization (users) to upload an appropriate picture regarding their organizations. There is only one picture can be uploaded at a time.

- Upload Picture text

- This is an indicator for users to notice where to upload picture regarding their organizations.

- Organization Name section

- (Organization) Users must fill in this section with their correct organization's name. They may fill in this section with alphabets and numbers. The organization's name may not be too long exceeding the space provided.

- Choose Donation Type section

- Users can click this section to have a pop-down menu where they can go more details regarding what type of donation, they are giving by choosing the options such as money, necessities, groceries, and others. Users can select 'others' if the type of donation is not available among the given options.

- Address section

- This lets users to fill in their full address of the organization so that people will get the information of where they can provide the donation. In this section, users may fill in with alphabets, numbers, and also special characters.

- Details (tell me about what you are donating) section

- Users can add more details regarding their organizations such as their organization landmarks, person in charge and the process of providing donation to this organization. This section provides a space for organizations to add any more details that seem important for people to know that have not been informed in other sections.

- Operating Hours section

- This allows users to inform the time that is most suitable to provide donations by filling in when is the opening time and when is the closing time so that people can get aware and plan when the best time is to visit the organization.

- Contact Number section.

- This is the section for organization to fill in their contact number in case of people want to reach out the organization. This section lets user to fill in their contact number with numbers and special characters. It is preferable to have at least one contact number to be filled in.

- Set Map Location button

- Users can press this button to navigate to the real map page where organization can indicate the pinpoint on the map of where is exactly the location of the organization. This helps to strengthen the address section where it lets people to confirm where the organization is by looking at the map. Clicking this button will bring the users to the map selection page to pin their selected location for their foodbank.

- Guidelines

At the very bottom of the screen, there will be a link that will redirects the user to the guideline page on their default browser. Users can refer to this guidelines and take extra precautions so that the submission are complying with our submission guidelines.



Unlike the Donation Request page, all the fields in the foodbank submission page are mandatory. Failure to such will cause the application to prompt an error to the blank fields until the user inputted the filed.

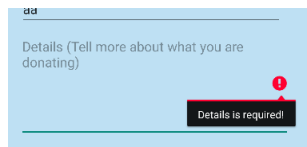


Fig. 32. Error on blank field.

#### F. Select Location Page

This page will appear whenever a user clicked the "Set Map Location" from the donation request page or foodbank submission page. On this page, a map will be executed on the screen and there will be 2 button at the bottom of the screen which are "Set Location" and "Use Current Location". User can use this page to select their desired location for their request or foodbank to appear on the map at the Main page after a successful submission. There will be one draggable marker on the initial execution.



Fig. 33. Location selection page.

Before user are able to reach this page, they must turn on the location and network services on their devices. Failure to do such will causes an error and user will be rejected from entering the page.

##### • Set Location Button

If users wanted to use the draggable marker to mark their desired location, they should use this button. The position of the marker will then be recorded and be added to their submission. After clicking the button, another popup window will appear to get the confirmation of the user if they are really

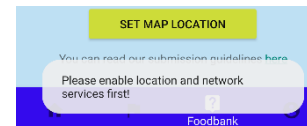


Fig. 34. Error on disabled location service.

satisfied with the location. If they are satisfied, users should click "OK" prompt on the popup window. If not, users can click the "Cancel" button or just click anywhere outside the popup window to return to the location selection page.

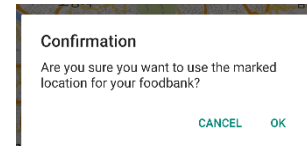


Fig. 35. Confirmation popup window.

##### • Use Current Location

If users prefer to use their own current location, the users can click this button without having to drag the map marker. Using this will cause the donation request or foodbank submission to appear on the map on the exact current location of the user. User also must be aware that their location service should be on before clicking this button. Failure to do such will cause the app to return to the donation request page or foodbank submission age. After clicking the button, another popup window will appear to get the confirmation of the user if they are really satisfied with the location. If they are satisfied, users should click "OK" prompt on the popup window. If not, users can click the "Cancel" button or just click anywhere outside the popup window to return to the location selection page.

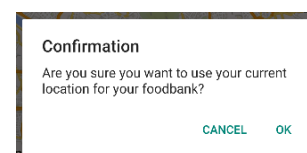


Fig. 36. Confirmation popup window.

If the user successfully chosen the desired location, a message will be shown at the bottom of the page informing that the submission is successfully uploaded and the marker should appear immediately on the map in the Map page.

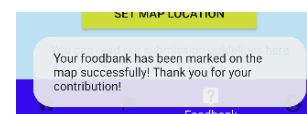


Fig. 37. Successfully submitted the form.

### G. Account Page

The application should have another page dedicated to show all the account information such as user name, full name, user ratings and others services and features.

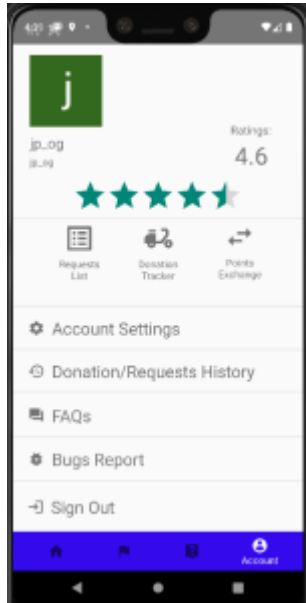


Fig. 38. User account page.

At the top of the page, the user profile picture account will be displayed and their user name and full name will be displayed underneath. On the right side, a numeric display will be shown to indicate the rating of the user. Below the user information is a rating bar that shows accordingly by the rating. If the user signed in using the Google authentication service, the profile picture, user name and full name will be displayed according to the user Google account.

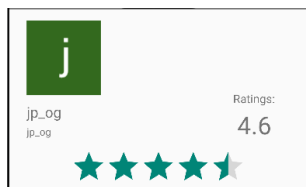


Fig. 39. User details are displayed at top.

Below the details are the 3 additional features that are provided for every user.

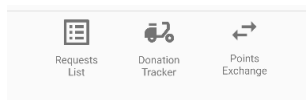


Fig. 40. 3 additional features.

- Request List

This tab will display all the request that has been submitted by the user. User can remove their made requests from here when they are satisfied with all the donation received. Doing such will remove the request from the list and also remove the marker from the map.

- Donation Tracker

This tab is only used for receiver who are going to receive donations via delivery or postage. In this tab, user can see the progress of the postage and their current status.

- Points Exchange

If the users completed any donations, they will receive points and those points can be exchanged to various prizes here. Such prizes are for example discount coupons, cashback, meals and much more.

At the half bottom of the page, we can see few services for the user to manage their account.

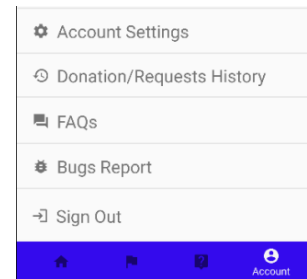


Fig. 41. Bottom half of page.

- Account Settings

As the name implies, this button will let the users to manage their account details such as full name, user name, change password, change profile picture, change app language and much more.

- Donation/Requests History

User can check their completed donations and deleted requests here for future references.

- FAQs

If users has any questions regarding our application features and services, they can refer to the frequently asked questions (FAQs) here.

- Bugs Report

If users encountered any problem or error in our application they can submit a report form here. Additionally, they also can submit suggestion to improve the application.

- Sign Out

Users can sign out from their account by just clicking this button. Upon click, a popup alert window will appear and ask the user if they really wanted to sign out. If "Cancel" is clicked, user will not sign out and stay on the account page. If "OK" button is clicked, user will be sent to the login page and logged out from the application.



Fig. 42. Bugs report and suggestions page.

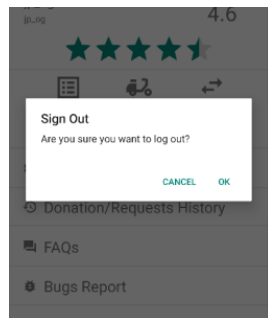


Fig. 43. Sign out confirmation.

## H. Term and Services

These Terms of Use (or “Terms”) govern your use of Doowa, except where we expressly state that separate terms (and not these) apply, and provide information about the Doowa Service (the “Service”), outlined below. When you create an Doowa account or use Doowa, you agree to these terms.

The Doowa Service is provided to you by Doowa, Inc. These Terms of Use therefore constitute an agreement between you and Doowa, Inc.

### • The Doowa Service

We agree to provide you with the Doowa Service. The Service includes all of the Doowa products, features, applications, services, technologies, and software that we provide to advance Doowa’s mission: To bring you closer to the people and things you love. The Service is made up of the following aspects:

- Offering personalized opportunities to create, connect, communicate, discover, and share.

People are different. We want to strengthen your relationships through shared experiences you actually care about. So we build systems that try to understand who and what you and others care about, and use that information to help you create, find, join, and share in experiences that matter to you. Part of that is highlighting content, features, offers, and accounts you might be interested in, and offering ways for you to experience Doowa, based on things you and others do on and off Doowa.

- Fostering a positive, inclusive, and safe environment.

We develop and use tools and offer resources to our community members that help to make their experiences positive and inclusive, including when we think they might need help. We also have teams and systems that work to combat abuse and violations of our Terms and policies, as well as harmful and deceptive behavior. We use all the information we have—including your information—to try to keep our platform secure. We also may share information about misuse or harmful content with other Doowa Companies or law enforcement.

- Developing and using technologies that help us consistently serve our growing community.

Organizing and analyzing information for our growing community is central to our Service. A big part of our Service is creating and using cutting-edge technologies that help us personalize, protect, and improve our Service on an incredibly large scale for a broad global community. Technologies like artificial intelligence and machine learning give us the power to apply complex processes across our Service. Automated technologies also help us ensure the functionality and integrity of our Service.

- Providing consistent and seamless experiences across other Doowa Company Products.

Doowa is part of the Doowa Companies, which share technology, systems, insights, and information—including the information we have about you (learn more in the Data Policy) in order to provide services that are better, safer, and more secure. We also provide ways to interact across the Doowa Company Products that you use, and designed systems to achieve a seamless and consistent experience across the Doowa Company Products.

- Ensuring access to our Service.

To operate our global Service, we must store and transfer data across our systems around the world, including outside of your country of residence. The use of this global infrastructure is necessary and essential to provide our Service. This infrastructure may be owned or operated by Doowa Inc., Doowa Ireland Limited, or their affiliates.

- Connecting you with brands, products, and services in ways you care about.

We use data from Doowa and other Doowa Company Products, as well as from third-party partners, to show you ads, offers, and other sponsored content that we believe will be meaningful to you. And we try to make that content as relevant as all your other experiences on Doowa.

- Research and innovation.

We use the information we have to study our Service and collaborate with others on research to make our Service better and contribute to the well-being of our community.

- How Our Service Is Funded

Instead of paying to use Doowa, by using the Service covered by these Terms, you acknowledge that we can show you ads that businesses and organizations pay us to promote on and off the Doowa Company Products. We use your personal data,

such as information about your activity and interests, to show you ads that are more relevant to you.

We show you relevant and useful ads without telling advertisers who you are. We don't sell your personal data. We allow advertisers to tell us things like their business goal and the kind of audience they want to see their ads. We then show their ad to people who might be interested.

We also provide advertisers with reports about the performance of their ads to help them understand how people are interacting with their content on and off Doowa. For example, we provide general demographic and interest information to advertisers to help them better understand their audience. We don't share information that directly identifies you (information such as your name or email address that by itself can be used to contact you or identifies who you are) unless you give us specific permission. Learn more about how Doowa ads work here.

You may see branded content on Doowa posted by account holders who promote products or services based on a commercial relationship with the business partner mentioned in their content. You can learn more about this here.

- The Data Policy

Providing our Service requires collecting and using your information. The Data Policy explains how we collect, use, and share information across the Doowa Products. It also explains the many ways you can control your information, including in the Doowa Privacy and Security Settings. You must agree to the Data Policy to use Doowa.

- Your Commitments

In return for our commitment to provide the Service, we require you to make the below commitments to us.

- Who Can Use Doowa.

We want our Service to be as open and inclusive as possible, but we also want it to be safe, secure, and in accordance with the law. So, we need you to commit to a few restrictions in order to be part of the Doowa community.

- You must be at least 14 years old. -You must not be prohibited from receiving any aspect of our Service under applicable laws or engaging in payments related Services if you are on an applicable denied party listing. -We must not have previously disabled your account for violation of law or any of our policies. -You must not be a convicted sex offender.

- How You Can't Use Doowa. Providing a safe and open Service for a broad community requires that we all do our part.

- You can't impersonate others or provide inaccurate information.

- You don't have to disclose your identity on Doowa, but you must provide us with accurate and up to date information (including registration information), which may include providing personal data. Also, you may not impersonate someone or something you aren't, and you can't create an account for someone else unless you have their express permission.

- You can't do anything unlawful, misleading, or fraudulent or for an illegal or unauthorized purpose.

- You can't violate (or help or encourage others to violate) these Terms or our policies, including in particular the Doowa Community Guidelines, Doowa Platform Terms and Developer Policies, and Music Guidelines.

- If you post branded content, you must comply with our Branded Content Policies, which require you to use our branded content tool. Learn how to report conduct or content in our Help Center.

- You can't do anything to interfere with or impair the intended operation of the Service.

- This includes misusing any reporting, dispute, or appeals channel, such as by making fraudulent or groundless reports or appeals.

- You can't attempt to create accounts or access or collect information in unauthorized ways.

- This includes creating accounts or collecting information in an automated way without our express permission.

- You can't sell, license, or purchase any account or data obtained from us or our Service. This includes attempts to buy, sell, or transfer any aspect of your account (including your username); solicit, collect, or use login credentials or badges of other users; or request or collect Doowa usernames, passwords, or misappropriate access tokens.

- You can't post someone else's private or confidential information without permission or do anything that violates someone else's rights, including intellectual property rights (e.g., copyright infringement, trademark infringement, counterfeit, or pirated goods).

- You may use someone else's works under exceptions or limitations to copyright and related rights under applicable law. You represent you own or have obtained all necessary rights to the content you post or share. Learn more, including how to report content that you think infringes your intellectual property rights, here.

- You can't modify, translate, create derivative works of, or reverse engineer our products or their components.

- You can't use a domain name or URL in your username without our prior written consent.

- Permissions You Give to Us. As part of our agreement, you also give us permissions that we need to provide the Service.

- We do not claim ownership of your content, but you grant us a license to use it.

You can read more about our Terms and Services and submission guidelines at <https://github.com/jpog99/DOOWA/tree/master/documentations>.

## V. ARCHITECTURE AND DESIGN

### A. Overall Architecture

For our DOOWA application, we have 4 main modules as the framework for the whole architecture. The first part handles the user interface and how user interacts with our application, which is handled in Android Studio as our front end development. This part mainly reacts with the user request such as retrieving all donations and food banks submitted and displayed on the main map, getting all the info and list of users' donation and submission history. For our back end, we implemented 3 core modules for the application to work seamlessly which is the server, Heroku, which is connected to 2 separate databases, PHPMyAdmin and Google Firebase. The first database, which is PHPMyAdmin, will contain the data of the users login authentication (if they are registering through our app) and all the donation and foodbank data such as the location (in longitude and latitude), time submitted, user name, details, image if any, phone number and much more. The second database, Firebase, stores the data of the user login authentication if they logged in using the Google services. Additionally, it is also used to store the image uploaded by the user since the other database is unable to store image. After the image is stored to the Firebase, only the image link will be passed to PHPMyAdmin database according to the which donation that was submitted with the image. Finally, our Heroku server will be the one that handles the retrieval and uploads of the donation request between the user interface and the database using PHP APIs. The communication between all 4 modules is illustrated as image below.

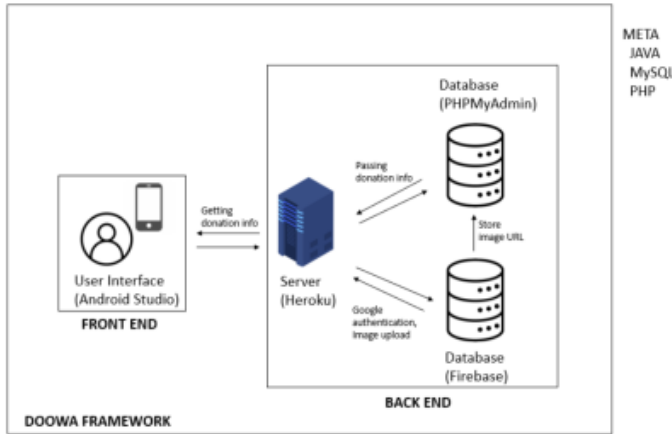


Fig. 44. DOOWA Application Framework.

### B. Directory Organization

Our project repository has various files that are used for building our application. Each files directory and usage are explained as follows.

TABLE IV  
FILE DIRECTORY

Directory	File Name	Module Name
DOOWA/app/src /main/java/com /example/doowa	AccountFragment.java BugReportActivity.java CameraActivity.java DetailsActivity.java FAQActivity.java FoodbankFragment.java ForgotPassActivity.java HistoryListActivity.java HistoryListAdapter.java ListDetailsActivity.java LoginActivity.java MainActivity.java MapFragment.java RegisterActivity.java ReportActivity.java RequestFragment.java Requests.java SetLocationActivity.java SubmissionListActivity.java SubmissionListAdapter.java TrackingActivity.java Upload.java checkSubmissionLocation.java	Android Studio
DOOWA/app/src /main/res/layout/	activity_bug.xml activity_camera.xml activity_check_submission_locati on.xml activity_details.xml activity_faactivity.xml activity_forgot_pass.xml activity_list_details.xml activity_login.xml activity_main.xml activity_register.xml activity_report.xml activity_set_location.xml activity_submission_list.xml activity_tracking.xml adapter.xml fragment_account.xml fragment_foodbank.xml fragment_map.xml fragment_request.xml show_dialog.xml	Android Studio
DOOWA/util/	composer.json DataBase.php deleteRequest.php deleteRequestHistory.php filterRequest.php filterRequestHistory.php insertRequest.php login.php requestDB.php restoreRequest.php signup.php	Server
DOOWA/ docu- mentations/	DOOWA Documentation.tex DOOWA Documentation.pdf README.md	Documentations

### C. AccountFragment.java

```

import logging

class AuthenticationSystem:
    """
    AuthenticationSystem class
    """
    def __init__(self, username, password):
        self.username = username
        self.password = password

    def login(self):
        """
        Login method
        """
        # Simulate login logic
        if self.username == "admin" and self.password == "123456":
            return True
        else:
            return False

    def logout(self):
        """
        Logout method
        """
        # Simulate logout logic
        return True

    def __str__(self):
        return f"AuthenticationSystem(username={self.username}, password={self.password})"

# Example usage
if __name__ == "__main__":
    # Create an instance of AuthenticationSystem
    auth_system = AuthenticationSystem("admin", "123456")

    # Login
    login_result = auth_system.login()
    print(f"Login result: {login_result}")

    # Logout
    logout_result = auth_system.logout()
    print(f"Logout result: {logout_result}")

    # Print the object
    print(auth_system)

```

Fig. 45. AccountFragment.java

1. Purpose: This class serves purpose to handles the account page and how each buttons interacts when pressed. This class extends Fragment class from JAVA library because it acts a a fragment and not an activity in our application. This is because it is one of the 4 main components of the bottom navigation which requires the use of fragment. This class also handles the data of each user and replaces the name and username field on the account page according to the respective users.

2. **Functionality:** As mentioned before, this class will handle the retrieval of the user data from either the MySQL database or Firebase (in case of signed in with Google). After the data is retrieved, this class will change the profile picture, username, full name and ratings field according to each user. This class also handles what each clicks on the button on the account page do such as opening new activity and shows up pop up windows.

3.	Location of Source Code:
DOOWA/app/src/main/java/com/example/doowa/	Account-
Fragment.java	

4. Class components: There are few methods implemented in this class.

- onCreateView()

In this method, the profile picture, full name, username and ratings of a user will be replaced according to the data retrieved from the login info or the Firebase database if the user signed in using Google service. This method also assigns "setOnClickListener" attributes to several text and buttons that are displayed on the account page so that those texts are clickable by users. This method returns View object.

- `onClick()`

This method has switch case which assign what the application do when certain texts are clicked.

- `onChangeLanguageDialog()`

This method shows the pop up window for language choice whenever "Change Language button is clicked"

- `setLocale()` and `loadLocale()`

This method sets application language and restarts the app in that language.

- `signOut()`

This method handles the sign out process for the user.

#### D. BugReportActivity.java

[illegible]

Fig. 46. BugReportActivity.java

1. Purpose: This class is used to let the users send any bugs reports or suggestions for our application improvements.

2. **Functionality:** If a user wanted to submit a bug reports or submissions, they can use the feature implemented from this class and when the submit button is clicked, the submission will be sent to our database.

3.	Location of Source Code:
DOOWA/app/src/main/java/com/example/doowa/BugReportActivity.java	BugReportActivity.java

4. Class components: There are few methods implemented in this class.

- onCreateView()

This method prepares the text box for input and assign the button to be clickable.

- `onClick()`

When the submit button is clicked, this method sends the contents of the text box to database.

## E. CameraActivity.java

```
public class CameraActivity extends AppCompatActivity {

    private static final int PICK_IMAGE_REQUEST = 1;

    private Button btnOpenGallery;
    private Button btnUploadImage;
    private TextView txtImagePreview;
    private EditText edtImageCaption;
    private ProgressBar progressBar;
    private int;

    private Uri imageUri;

    private StorageReference storageRef;
    private DatabaseReference dbReference;

    private DatabaseReference dbReference;

    private void setUpUI() {
        setContentView(R.layout.activity_camera);
        btnOpenGallery = findViewById(R.id.btn_open_gallery);
        btnUploadImage = findViewById(R.id.btn_upload_image);
        txtImagePreview = findViewById(R.id.txt_image_preview);
        edtImageCaption = findViewById(R.id.edt_image_caption);
        progressBar = findViewById(R.id.progress_bar);

        storageRef = FirebaseStorage.getInstance().getReference().child("images");
        dbReference = FirebaseDatabase.getInstance().getReference().child("donations");

        btnOpenGallery.setOnClickListener() {
            // Open gallery to select image
            Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(intent, PICK_IMAGE_REQUEST);
        };

        btnUploadImage.setOnClickListener() {
            // Upload image to Firebase Storage
            if (imageUri != null) {
                StorageReference ref = storageRef.child(System.currentTimeMillis() + ".jpg");
                ref.putFile(imageUri);
                ref.getDownloadUrl().addOnSuccessListener() {
                    // Get download URL
                    Uri downloadUrl = ref.getDownloadUrl();
                    // Save to database
                    dbReference.child(downloadUrl.toString()).setValue(downloadUrl.toString());
                    // Show confirmation window
                    confirmationWindow();
                };
            } else {
                Toast.makeText(this, "No image selected", Toast.LENGTH_SHORT).show();
            }
        };
    }

    @Override
    protected void onActivityResult() {
        if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK) {
            imageUri = Uri.fromParts("content", "com.android.providers.media.documents", "data/" + uri);
        } else {
            // User cancelled
        }
    }
}
```

Fig. 47. CameraActivity.java

1. Purpose: This class is used to let the users upload image they wanted optionally whenever they submit a donation request or foodbank.

2. Functionality: When user open the "Image upload" page, their gallery will be opened and users can choose any picture to upload. After select, the image preview will be shown on the page and when the upload button is clicked, a progress bar will be filled up and user is sent to submission page after successful upload.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/CameraActivity.java

4. Class components: There are few methods implemented in this class.

- onFileChooser()

This method opens the user gallery to choose image to upload.

- getFileExtension()

This method returns the type of file chosen.

- uploadFile()

This method handles the upload process of the image to the Firebase database. Then, the image URL will be sent to the submission page.

- confirmationWindow()

This method will cause a pop up window to appear and ask the user for image upload confirmation.

## F. DetailsActivity.java and ListDetailsActivity.java

1. Purpose: This class handles the details page which shows the details of donations or foodbanks that are appeared on the main map. User will see

```
public class DetailsActivity extends AppCompatActivity {

    private static final int REQUEST_CODE = 1;

    private TextView txtTitle;
    private TextView txtAddress;
    private TextView txtPhone;
    private TextView txtEmail;
    private TextView txtImage;
    private TextView txtDescription;
    private TextView txtStatus;
    private TextView txtDate;
    private TextView txtLocation;
    private TextView txtCategory;
    private TextView txtRating;
    private TextView txtComments;
    private TextView txtActions;

    private Uri imageUri;

    private StorageReference storageRef;
    private DatabaseReference dbReference;

    private void setUpUI() {
        setContentView(R.layout.activity_details);
        txtTitle = findViewById(R.id.txt_title);
        txtAddress = findViewById(R.id.txt_address);
        txtPhone = findViewById(R.id.txt_phone);
        txtEmail = findViewById(R.id.txt_email);
        txtImage = findViewById(R.id.txt_image);
        txtDescription = findViewById(R.id.txt_description);
        txtStatus = findViewById(R.id.txt_status);
        txtDate = findViewById(R.id.txt_date);
        txtLocation = findViewById(R.id.txt_location);
        txtCategory = findViewById(R.id.txt_category);
        txtRating = findViewById(R.id.txt_rating);
        txtComments = findViewById(R.id.txt_comments);
        txtActions = findViewById(R.id.txt_actions);

        storageRef = FirebaseStorage.getInstance().getReference().child("images");
        dbReference = FirebaseDatabase.getInstance().getReference().child("donations");

        txtImage.setOnClickListener() {
            // Open gallery to select image
            Intent intent = new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
            startActivityForResult(intent, REQUEST_CODE);
        };

        txtActions.setOnClickListener() {
            // Call phone number
            Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + txtPhone.getText().toString()));
            startActivity(intent);
        };
    }

    @Override
    protected void onActivityResult() {
        if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
            imageUri = Uri.fromParts("content", "com.android.providers.media.documents", "data/" + uri);
        } else {
            // User cancelled
        }
    }
}
```

Fig. 48. DetailsActivity.java

this page when they clicked the marker on the map.

2. Functionality: When the user clicked on a marker on the map, this class will fetch the data from the database and replaces all the text fields on the details page according to the fetched data. The data is fetched by sending request to server along with the longitude and latitude of the marker.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/DetailsActivity.java DOOWA/app/src/main/java/com/example/doowa/ListDetailsActivity.java

4. Class components: There are few methods implemented in this class.

- loadRequests()

This method retrieves the data from the database to be displayed on the details page.

- makePhoneCall()

This method redirects the user from the application to their phone app and calls the number of the requester immediately.

## G. FAQActivity

1. Purpose: This class will show the frequently asked question text on the FAQ page to the users.

2. Functionality: When the user clicked on the FAQs button on the account page, this class will show the contents of the FAQ page and make the page scrollable.





```

package com.example.doowa;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.example.doowa.databinding.ActivityHistoryListBinding;
import com.example.doowa.model.Submission;
import com.example.doowa.repository.SubmissionRepository;

import java.util.List;

public class HistoryListActivity extends AppCompatActivity {

    private ActivityHistoryListBinding binding;
    private SubmissionRepository repository;
    private RecyclerView recyclerView;
    private LinearLayoutManager layoutManager;
    private List<Submission> submissions;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityHistoryListBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        repository = SubmissionRepository.getInstance();
        layoutManager = new LinearLayoutManager(this);
        recyclerView = binding.recyclerView;
        recyclerView.setLayoutManager(layoutManager);

        repository.getSubmissions().subscribe(
            submissions::addAll,
            () -> {
                Toast.makeText(this, "Data loaded", Toast.LENGTH_SHORT).show();
            }
        );

        binding.button.setOnClickListener(
            () -> {
                repository.deleteSubmission(1);
            }
        );
    }
}

```

Fig. 52. HistoryListActivity.java

```

package com.example.doowa;

import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.example.doowa.databinding.ActivityHistoryListBinding;
import com.example.doowa.model.Submission;
import com.example.doowa.repository.SubmissionRepository;

import java.util.List;

public class HistoryListAdapter extends RecyclerView.Adapter<HistoryListAdapter.ViewHolder> {

    private List<Submission> submissions;
    private SubmissionRepository repository;

    public HistoryListAdapter(List<Submission> submissions, SubmissionRepository repository) {
        this.submissions = submissions;
        this.repository = repository;
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_submission, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        Submission submission = submissions.get(position);
        holder.textView.setText(submission.getTitle());
        holder.textView2.setText(submission.getDescription());
        holder.textView3.setText(submission.getTime());
        holder.button.setOnClickListener(
            () -> {
                repository.deleteSubmission(submission.getId());
            }
        );
    }

    @Override
    public int getItemCount() {
        return submissions.size();
    }
}

```

Fig. 53. HistoryListAdapter.java

#### J. HistoryListActivity.java and SubmissionListActivity.java

1. Purpose: This class handles the retrieval of users' submission from the database and display it on the history submission page.

2. Functionality: First, the class will send the username of the user to the server and the server will select the submission data according to the username. The server will send the list to this class and the list will be displayed.

3. Location of Source Code: - DOOWA/app/src/main/java/com/example/doowa/HistoryListActivity.java  
-DOOWA/app/src/main/java/com/example/doowa/ SubmissionListActivity.java

4. Class components: There are few methods implemented in this class.

- filterRequest()

This method sends the username to server and receive the data of the users' deleted submission.

- loadSubmission()

This method displays the data received from the previous method to the screen.

#### K. HistoryListAdapter.java and SubmissionListAdapter.java

1. Purpose: This class mainly used to let the users interact with the menu for each submission list which is represented by the three dot on the corned in the submission list page (or history page.) such as

see details, check location, or delete the submission.

2. Functionality: The class has few functions which displays the submission type and the time submitted. Then, if the user choose to see the details, then the user will be redirected to the details page. If they decide to check the location, then they will be directed to the map page. Finally, if they choose to delete the submission, the the submission will be sent to the server and the server will delete the submission from the database according to the id.

3. Location of Source Code: - DOOWA/app/src/main/java/com/example/doowa/HistoryListAdapter.java  
-DOOWA/app/src/main/java/com/example/doowa/ SubmissionListAdapter.java

4. Class components: There are few methods implemented in this class.

- onBindViewHolder()

This method handles the choice menu for each submission ion the list and do different things accordingly whenever a user clicked on the chosen menu.

- deleteSubmission()

This method deletes the submission from the database.

- restoreSubmission()

This method lets the user to restore the submission from the history back to the submission list

and will be displayed on the map again.

#### L. LoginActivity.java

A screenshot of the LoginActivity.java file in an IDE. The code shows the class structure, including imports for FirebaseAuth, FirebaseAuthException, and FirebaseFirestore. It includes methods for onCreate, onStart, and onLoginClick, which handle user login logic, including checking credentials against a database and handling Google authentication.

Fig. 54. LoginActivity.java

1. Purpose: This class mainly used to check the user info before logs them in into the main application page.

2. Functionality: The class has few functions which checks if the username and password combination inputted are valid from the database. After the combination is inputted and login button is clicked, the data will be send to the server and checked. If the combination is valid, then the user will be directed to the main page. If not, a failure message pops up. This class also handles the Google authentication services if the user decides to log in using their Google account.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/Login Activity.java

4. Class components: There are few methods implemented in this class.

- createRequest(), signIn(), firebaseAuthWithGoogle()

These methods handles the process of Google authentication if the user signed in using their existing Google account.

- userSignIn()

This method handles the situation if the user signed in using the account registered form the app.

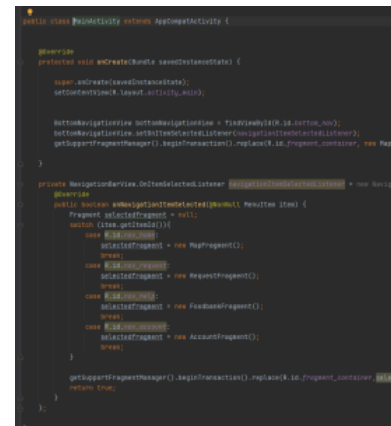
A screenshot of the MainActivity.java file in an IDE. The code shows the class structure, including imports for AppCompatActivity, getSupportFragmentManager, and ViewPager. It includes methods for onCreate, onStart, and onNavigationItemSelected, which handle the main activity logic, including setting up the bottom navigation bar and managing fragments.

Fig. 55. MainActivity.java

#### M. MainActivity.java

1. Purpose: This class is used to process the bottom navigation display on the main page.

2. Functionality: The class basically acts as a fragment container for 4 main pages which are map, request submission, foodbank donation and account.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/Main Activity.java

#### N. MapFragment.java

1. Purpose: This class is used to handle how the maps appears on the main map page and how the submission markers are located on the map.

2. Functionality: The class will first fetch all the submission data from the database and stored in a List object. Then, the latitude and longitude are retrieved from the List and the markers are displayed on that data. The marker icons are also changed according to the donation type of the submission which is retrieved from the List.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/MapFragment.java

4. Class components: There are few methods implemented in this class.

- loadRequest

These methods retrieves all the donation data from the database, extracts the data and marks the submission locations the on the map accordingly.





- When the submit button is clicked, this method sends the contents of the text box to database.

```

        return result;
    }

    String lat;
    String lng;
    String phone, destinationType, time_address, image_name, url, details, details1, meetingTime, type;
    String report, displayName;
    int id;

    public Requester(){}

    public Requester(int id, String lat, String lng, String phone, String destinationType, String address,
        String id = id;
        this.lat = lat;
        this.lng = lng;
        this.phone = phone;
        this.destinationType = destinationType;
        this.address = address;
        this.image = image;
        this.report = report;
        this.details1 = details1;
        this.time = time;
        this.name = name;
        this.graficaImage = graficaImage;
        this.meetingTime = meetingTime;
        this.type = type;
        this.displayName = displayName;
    }

    public String getLat(){}return lat;

    public String getLng(){}return lng;
}

```

1. Purpose: This class is used to store the retrieved submissions (or uploaded image) from the database and create a List object with these types.

3.	Location	of	Source	Code:
	DOOWA/app/src/main/java/com/example/doowa/Requests.java			
	DOOWA/app/src/main/java/com/example/doowa/Upload.java			

1. Purpose: This class is used to let the users set their desired location on the map for their submission. Then, the location is saved and sent to the database.

3. Location of Source Code: DOOWA/app/src/main/java/com/example/doowa/SetLocationActivity.java

- insertRequestDB()



This method prepares the data submitted by the user from the request or foodbank submission page and the marker location. Then, the method will push all these submission details to the database.

- This method is used to fetch the data that are filled by the users from the request or foodbank page.

- This method assigns the current location of the users' device to the double value "latitude" and "longitude" of the class.

- This method checks if the users already enabled network and location services on their devices.

## VI. USE CASES

to be continued...