A Project Report

*on*

# Design Project (Mobile application)

*by*

Pooja Kanojia (AU18B1008)

*Under the guidance of*

Atul Choudhary


School of Computer Engineering

Avantika University, Ujjain

**2019-2020**

# Contents

# Abstract

The domain targeted was online food donation platforms. These platforms are used by various professionals, NGOs, Animal Shelters, Old age homes, college mess for collection and donation of food which gets wasted instead at Marriages, parties, college fests and other events. These platforms provide a food donation service to make but does not help the donor connect to his/her required food collector after making the donation. Hence, finding the right food collector in time after making the donations is important so that a food donor keeps donating more, both the sides are benefited and users are engaged with the application. The solution helps a donor make online food donations and connect them with the food collector/organization as soon he/she accepts that donation. In this way the solution helps to reduce the food wastage. A food collector gets the freedom to either accept or decline a donation and a food donor can track his/her donations with the live stats provided within the application.

# Chapter 1

# Introduction

## 1.1 Motivation

The major motivation for choosing the above domain was the time, energy which gets wasted on finding the right food collector to whom you wish to donate food as a donor. The solution helps reduce the time for finding a user. This platform helps to make a community of helpers who are donating edible wasted food instead of throwing it. It helps both the categories of users and helps spread among users about food wastage.

## 1.2 Problem Statement

To identify a problem and design a mobile application for the same.

## 1.3 Aim

To create a platform to help users donate food and collect food according to their category (user - type).

## 1.4 Objective

- Food donors should get their suitable food collectors at one platform.
- Both donors and collectors should get benefited.
- To reduce the time of finding an individual/organization to collect donated food.
- To provide real time food donation/wastage stats in order to spread awareness.

# Chapter 2

# Background Study

## 2.1 Background Theory

I followed a process during the making of the android application. It started with identifying the problem statement, target users, identifying the target domain. The database designing was done by identifying the entities, attributes and their relationship. After this, the wireframing was done using Adobe XD and then the layouts were coded in XML (Extensible markup language). At this step, we also decide the activities and fragments for our project. We learnt and used XML, different functions for styling a layout, different views like constraint, relative and to establish connection between different activities in Android. XML is a markup language for making the layouts of the mobile application. Functions were made in Java to add validations to the registration and login forms. The menu was made using the item function. Animations were included by including Lottie dependency and files. Charts and stats were made using the MPAndroidchart library. PHP was used to fetch, store, display data to and from the database. PHP was used since it is easy to use and open source. MySQL database was used. The localhost server (XAMPP) was used for the mobile app. To connect our mobile app to the database, the .mysqli_connect() function in which we passed the hostname, username, password and database name. Session handling is implemented in the project by using the concept of shared preferences. The session is maintained by checking token key of the logged in user. Token key is a unique 32 digit id/key which is unique to a user.

## 2.2 Existing Systems

1. No food waste - India
2. Zero Percent – Chicago based
3. Waste no food – Waste no Food California based non-profit

## 2.3 Proposed Solution

A mobile application which will help users make food donations and post their donations to automatically to all the food collectors/organizations registered with the app. These Food collectors will be present on the same platform on which the donations are made. Compulsory registration for a food donor and food collector since we need their personal details to ensure authenticity of the platform. A person can either be a donor/collector but not both. A donor can post more than one donation also he/she can track the quantity of food donated with the help of live stats visible. An interviewee can accept or reject more than one donation but not the same survey more than once. Donor gets notified about all the accepted donations in the notifications section.

## 2.4 Unique Selling Points (USP)

1. A donor posting a donation can find their food collectors on the same platform.
2. Track your food donations and wastage with real time statistics.
3. Collector gets the choice to accept or reject a donation.

# Chapter 3

# Software Requirement Specification

## 3.1 Software Requirements

1. XAMPP Sever (version 5.6.14)
2. MySQL database (version 5.7)
3. Visual Studio Code (editor for coding version 1.40)
4. Draw.io (Online Platform for making ER and Schema diagrams)
5. Adobe XD (for wireframing and making layouts)

Operating system (Windows/linux)

Android Studio (version 4.0)

Android Virtual Device (for testing of the application)

## 3.2 Hardware Requirements

For xampp (512MB ram)

The default databases like information_schema, test, mysql, spin are of the sizes 0.185 MB, 0.015 MB, 1.179 MB, 145.8 MB

Data cable and an android phone for testing on real device.

# Chapter 4

# Assumptions

1. A food donor cannot be a food collector and vice versa.

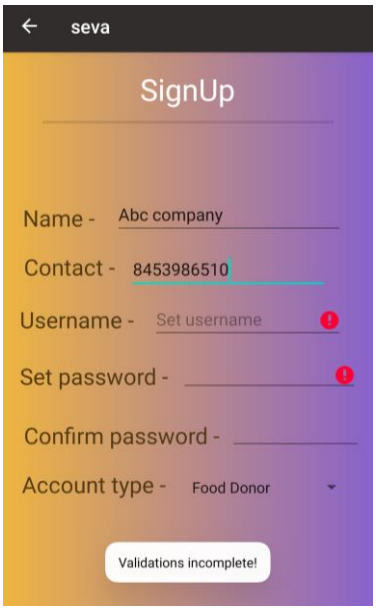
# Chapter 5

# System Functional Requirement

To make the mobile application functional I created different Java functions in android and used inbuilt functions of php for the sever side.

User defined functions were made in Java to validate signup and sign in forms. Conditions for empty fields, contact number, regex for password were validated.

```
public boolean validatePassword() {
    String set_pw = set_password.getText().toString();
    String conf_pw = confirm_password.getText().toString();
    if (set_pw.isEmpty()) {
        set_password.setError("password field required");
        return false;
    } else if (!PASSWORD_PATTERN.matcher(set_pw).matches()) {
        set_password.setError("password too weak");
        Toast toast = Toast.makeText(getBaseContext(),
                text: "atleast one uppercase, lowercase, specia
                Toast.LENGTH_LONG);
        toast.show();
        return false;
    } else if (conf_pw.isEmpty()) {
        confirm_password.setError("confirm your password");
        return false;
    } else if (!(conf_pw.equals(set_pw))) {
        confirm_password.setError("password does not match");
        return false;
    }
    return true;
```




Used new mysqli() function of php to connect the applicaton to the database. In this function we pass the hostname, username, password and database name as parameters. $response array takes error or message or data as its parameters.

```
<?php
$db = new mysqli('localhost','root','','seva');//making a new connection to the database

if($db->connect_errno){
    $response = [ //making an array
            'error' => true,
            'message' => $db->connect_error
    ];

    echo json_encode($response);
    return;
}
```

I used query() function of php to execute our sql  query.

I used fetch_assoc() function and while loop to fetch data from the database in the form of array and store it in the $response array.

```
$response = [];
if(!empty($data['token'])) //only token needed from the user
{
    global $db;
    $sql = "select user_id from users where token_key = '$data[token]'";
    $result = $db->query($sql);
    if ($result->num_rows > 0){
        $id = $result->fetch_assoc()['user_id'];
```

```
while($row = $result->fetch_assoc())
{
    $row['image_name'] = 'http://'.$_SERVER['HTTP_HOST']."/mobile/seva/images/".$row['image_name'];
    $response[] = $row;

}
```

$req_obj helps select the type of request we want to send in the API service(POST/GET)
$req_action is the request we want to send. header() function is set to json content type to get the response as a JSON object. Json_encode() function to convert the function into a json format

```
$req_obj = ($_SERVER['REQUEST_METHOD'] == "POST")?$_POST:$_GET; //checking if request object is post or get
$req_action = (isset($req_obj['req'])) ? $req_obj['req'] : "default";
header("Content-Type:application/json"); //setting the content type as json
require 'db.php';
```

```
 }
 echo json_encode($response);
```

Picasso library used to convert the url from the database to the image (JPG/JPEG) format for displaying it. Retrofit library is used to implement the API services (connection between the front and back end). Lottie is used for animations. MPAndroid chart is used for creating charts and displaying stats in the application.

```
implementation 'com.squareup.retrofit2:retrofit:2.2.0'
implementation 'com.squareup.retrofit2:converter-gson:2.2.0'
implementation 'com.squareup.picasso:picasso:2.5.2'
implementation 'com.google.firebase:firebase-analytics:17.2.2'
implementation 'com.airbnb.android:lottie:3.4.0'
implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
```

# Chapter 6

# Entity-Relationship Design

## 7.1 Identify Entities of the above problem

- Users
- Donations

## 7.2 Identify the attributes of the above entities

### 7.2.1 Users:

- user_id
- name
- contact
- username
- password
- token_key
- user_type

### 7.2.2 Donations:

- donation_id
- id
- image_name
- donation_quantity
- donation_brief
- collectorID
- donation_date

## 7.3 Identify the Primary and Foreign keys of all entities identified

7.3.1 **Users:** user_id

7.3.2 **Donations:** donation_id, id, collectorID

## 7.4 Identify the attribute types:

7.4.1 **Users:**

- user_id (single valued)
- name (single valued)
- contact (single valued)
- password (single valued)
- token_key (single valued)
- user_type (single valued)

**7.4.2 Donations:**

- donation_id (single valued)
- id (single valued)
- donation_quantity (single valued)
- donation_brief (single valued)
- collectorID (single valued)
- image_name (single valued)
- donation_date (single valued)

## 7.5 Identify the Relationships and relationships attributes between the entities:

Table 7.5.1. Relation and relationship attributes between the entities

| S.no | Relationship name | Entities name having relationship among them | Attributes |
|---|---|---|---|
| **1.** | donates | User and Donations | |
| **2.** | accepts | Users and Donations | |

## 7.6 Identify the constraints on the entities:

| S. No | Relationship Name | Cardinality | Reason |
|---|---|---|---|
| **1.** | donates | 1 : N | Since 1 donor can donate N number of donations. |
| **2.** | accepts | 1: N | Since 1 collector can accept N number of donations |

# Chapter 7

# Implementation Details

I made the registration layout with spinner options like Food donor and Food collector to select from while one registers for the platform. The following user is then stored in the database with his/her user_type category. The login process is done by checking the user_type from the users table, its token and applying conditional statements accordingly for directing the user to the donor/collector landing page accordingly.

I created two different interfaces. The token_key of the user is stored in shared preferences to pass the information of that user into the other layouts by using the getApplicationContext method. If a user revisits the app again without logging out and if the token_key, user_type of the user matches with the token key in the database then there is no need for the user to again log in.

When a new user registers on the registration page there are validations applied on all the fields. The password must have an uppercase, lowercase, numeric, special character. This was done using regex. Validations were applied using Java. On click of submit button we call the function in which all the conditional statements are written.

For connecting the database with the front end, we make APIs and integrate it by adding them to the api service class, a result and a URL file is then made accordingly. For uploading a food image (file), we need an external read only permission. Onclick of the upload button, if the image pick code is available then (permissions granted) then the file manager is opened from which we can select the image. On click of submit button (new donation form) the form gets submitted within the fragment and the fragment gets replaced by passing the index of the other fragment.

In the notifications section on the donor side I have used a query to fetch all the donations drom the database whose food_status is yes for that particular food donor. The cards shown in the layout are displayed using an ArrayList adapter and a recycler view. In the array list I fetched the details of those donations (accepted donations) and then displayed using the adapter. In the waste food tracker section, I have used the MPchart dependency to display the pie and bar chart according to real time data from the database. In the logout section, we end the session and redirect the user to the homepage of the application.

On the collector side interface, a food collector gets to see all the new donations made by various food donors sorted in a date wise order. The see more button directs the collector to the activity where he/she can view the complete details of a food donor. At this point, the collector gets a choice to either accept or decline a food donation. If the donation is declined, the food donation gets temporarily removed from his/her arrayList. If a donation is accepted, the food staus in the database is set to yes and the details of the collector ia notified to the donor whose donation it was. Hence, a cycle is maintained between the food donor and collector.

# Chapter 8

# Results


Fig.1. Splash screen for the mobile app


Fig.2. Home screen for the app

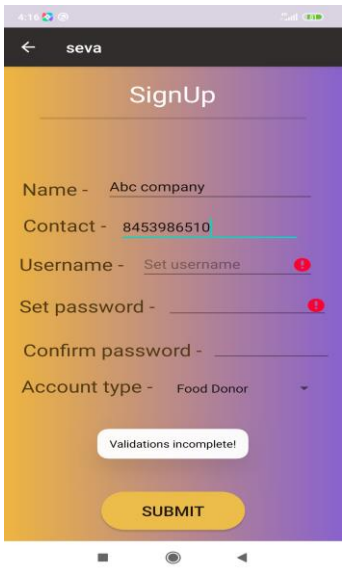



Fig.3. Sign up screen for the app

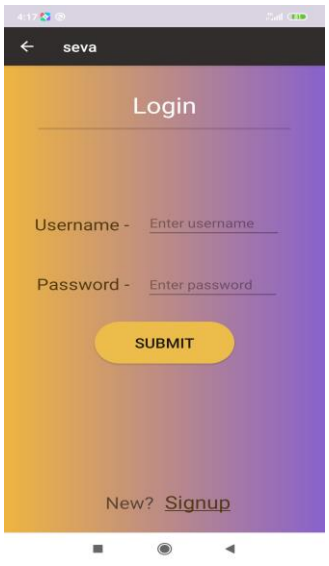



Fig.4. Login screen for the app

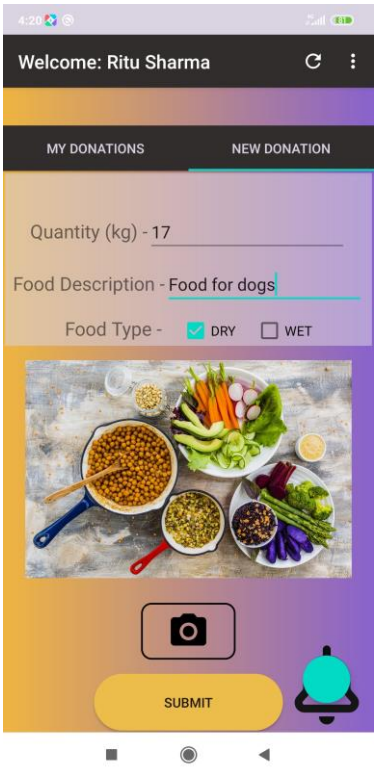



Fig.5. Make a new donation (Donor side)

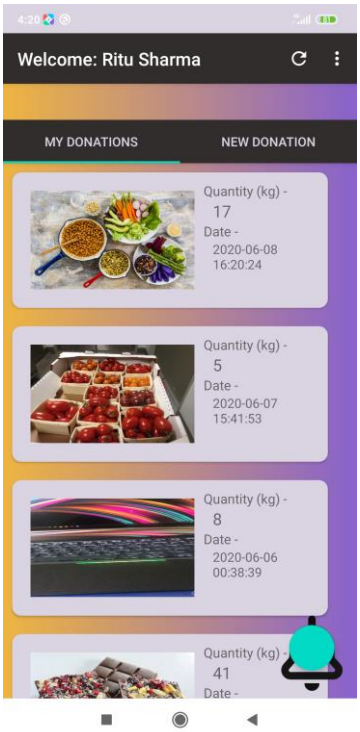



Fig.6. My donations list get updated

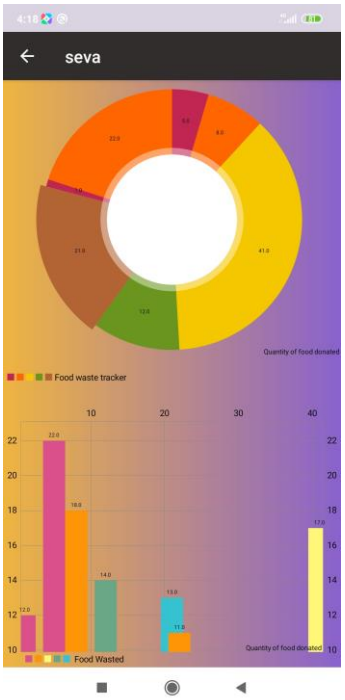



Fig.7. Displaying real time stats (food quantity)

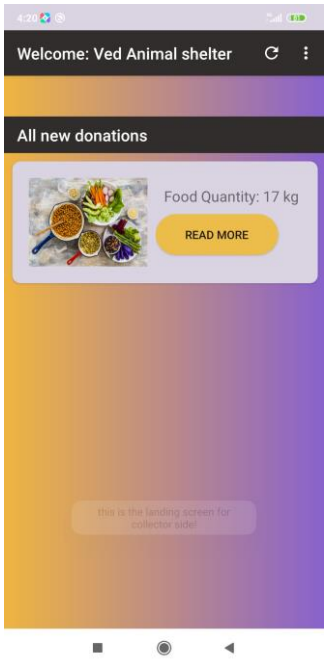



Fig.8. Collector side new donations

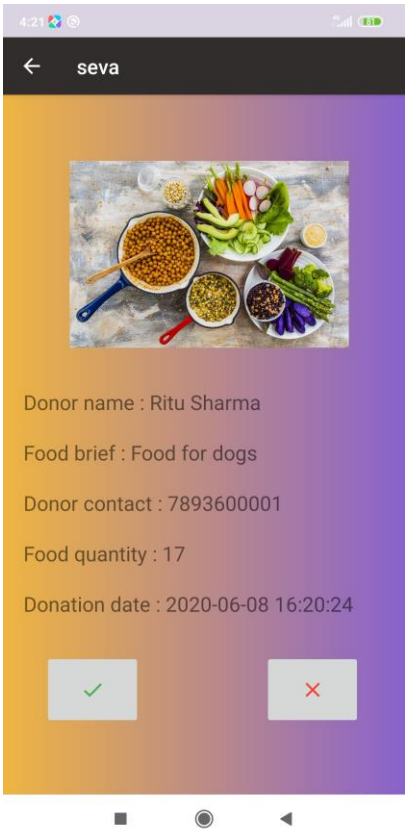



Fig.9. Show more option (collector side)

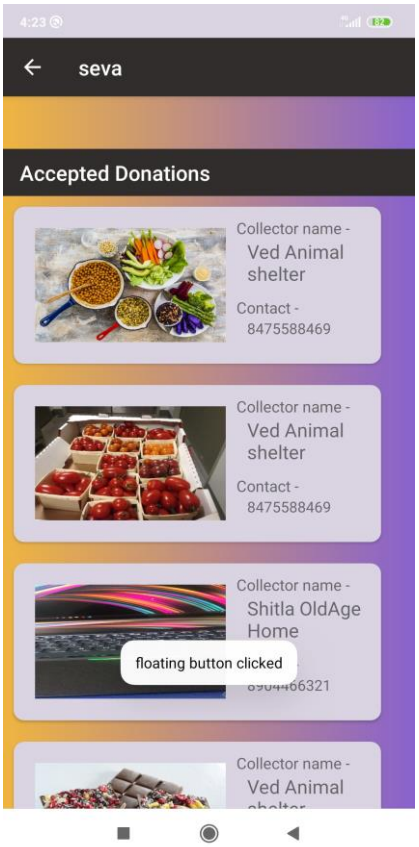



Fig.10. Collector accepts donation (donor side)

# Chapter 9

# Conclusion

Currently the system has two interfaces for two different users Food donor and Food collector. Under my donations in the donor interface the cards are displayed with the logged in user's record of past donations (image, quantity and date). While, in the new donation sections he/she can make a new donation which gets displayed later in "my donations" section. Hence, data transaction is maintained. Under the "notifications section", the user gets a result when his/her is donation is accepted by a food collector. In the menu section in the action bar, donor gets the option to see his/her donations in the form of live statistics (pie and bar graph). The Collector interface of the application consists of a new donation section in which a collector gets to see all the new donations made by any food donor. After selecting the see more, the collector gets the choice to accept/decline the donation. Improvements in the UI, more statistics and an admin section can be added in the future.