

GARBAGE COLLECTION APPLICATION

COMPUTER SCIENCE PROJECT I

COM 311

A PROJECT SUBMITTED TO:

THE DEPARTMENT OF

STATISTICS AND COMPUTER SCIENCE

MOI UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

BY:

|  |  |  |  |
| --- | --- | --- | --- |
| *Reg. No* | *Name* | *Phone* | *Signature* |
| *C/SC/457/14* | *MUKELE DANIEL* | *0727587728* |  |

DECLARATION:

I declare that this report is my original work and has not been presented for a degree award in any other university. No part of this work may be reproduced without prior written permission of the authors and moi university.

**Approval page**

**Supervisor’s Declaration**

This project has been submitted for reviews with my approval as the university supervisor.

Signature…………………………………… Date………………………………………..

Name……………………………………………………………………………………….

DEPARTMENT OF STATISTICS AND COMPUTER SCIENCE

MOI UNIVERSITY.

DEDICATION:

I dedicate this project to the entire Moi fraternity, my fellow comrades and my family at large.

Thanks for your support through out the development process.

**ABSTRACT**

This project addresses the study and development of a Garbage collection Application on

County Garbage Management System to enable residents, both the county administrators and the citizens to make use of the application. Traditionally, garbage management from the authorized bodies has been manually where county administrators has just been seen it then collecting it unlike in most interior parts of the county where county administrators has not been accessing the issue of garbage management has not possible.

The Garbage collector App will cater for garbage collection in every part of the county more so in major county towns like Nairobi where the issue seems to be dominating. This Mobile project addresses the problems faced by the residents from different parts where garbage is not collected and also where it takes time for it be collected because of manual collection of garbage. The project studies some issues on implementation and also recommendations on how Garbage collection Application can take place effectively.

This project recommends on a record keeping system based on individual app usage rate system. This project also recommends a Decision Support System to deal with Administrators requirement whereby it provides reliable choices to administrator to make decision. This project includes the development of a Garbage collection mobile App to support the

project objective. This Mobile Application will assist in future development that would support a

fully integrated system that links county’s cleanliness monitoring and the entire country. Borrowing from other counties measures of cleanliness will be taken into consideration.

ACKNOWLEDGEMENT.

This documentation of my project had been successfully completed. A lot of time and effort was spent to ensure that this project meets the objectives that were stated. I wish to express my sincere gratitude to Mr. Kiprop for his guidance and patience in supervising

the writing of this documentation of my project. His advice and valuable suggestions are the source of encouragement throughout the completion of my project.

Last, but not least, I wish to thank my beloved friends who have been

supportive during the writing of my project. Am very grateful for their encouragement and moral support during the progress of the project.

**Table of Contents**

**1. Introduction and Background**

1.1 Statement of Problem Area

1.2 Background

1.3 Brief Project Description

1.4 Purpose/Objectives/justification of Project

**2. Literature Review**

2.1 Previous and Current Work, Methods and Procedures

**3. System Functional Specification**

3.1 Functions Performed

3.2 User Interface Design

3.3 System Data Base/File Structure Preview

3.4 External and Internal Limitations and Restrictions

3.5 User Interface Specification

3.5.1 User Screens/Dialog

3.5.2 Report Formats/Sample Data

3.5.3 On­line Help Material (if applicable)

3.5.4 Error Conditions and System Messages

3.5.6 Control Functions

**4. System Performance Requirements**

4.1 Efficiency

* + Reliability
  + Security

4.3.1 Hardware Security

4.3.2 Software Security

4.3.3 Data Security

4.4 Maintainability

4.5 Modifiability

4.6 Portability

**5. System Design Overview**

5.1 System Data Flow Diagrams

5.2 System Architecture and Structure

5.3 System Data Dictionary

5.4 Description of System Operation

5.5 Equipment Configuration

5.6 Implementation Languages

5.7 Required Support Software

**6. System Data Structure Specifications**

6.1 User Input Specification

6.1.1 Identification of Input Data

6.1.2 Source of Input Data

6.1.3 Input Medium and/or Device

6.1.4 Data Format/Syntax

6.2 User Output Specification

6.2.1 Identification of Output Data

6.2.2 Output Medium and/or Device

6.2.3 Output Format/Syntax

6.2.4 Output Interpretation

6.3 System Data Base/File Structure Specification

6.3.1 Identification of Data Base/Files

6.3.2 (Sub)systems Accessing the Data Base

6.3.3 Logical File Structure

6.3.4 Physical File Structure

6.3.5 Data Base Management Subsystems Used

7. **Module Design specifications**

7.1

7.1.1 Module Functional specification

* + - * + Functions Performed
        + Module Interface Specifications
        + Module Limitations and Restrictions

7.1.2 Module operational Specification

* + Locally Declared Data Specifications
  + Algorithm Specification
  + Description of Module Operation

**8. System Verification**

8.1 Items/Functions to be Tested

8.2 Description of Test Cases

8.3 Justification of Test Cases

8.4 Test Run Procedures and Results

8.5 Discussion of Test Results

**9. Conclusions**

9.1 Summary

9.2 Problems Encountered and Solved

9.3 Suggestions for Better Approaches Problem /Future Extensions to Project

**10. Bibliography**

**11. Appendices**

**12. Program Listings**

* **User Manual**

**INTRODUCTION AND BACKGROUND.**

**1.1 Statement of problem area**

At present, the public can only know about the importance of having and maintaining cleanliness through conventional media means such as radio, newspaper or television advertisements. There is no information regarding the garbage collection programs available on any of the portal.

The current system that is using by the garbage collection is manual system. With the manual system, there are problems in managing garbage. The garbage might management information might not be safely kept and might be missing collection records due to human error. Beside that, errors might occur when the administrator more than one record for same collection place.

There is no centralized database of garbage collection in our counties. So, it becomes really tedious for a person to search and find concerned authority in case of need of garbage collection. The only option is to manually search and wait if the county will come and collect. If a county makes collection in different places, no previous records can be traced except if the administrator brings along the collection information. Hence, the collection is considered to be a first-timer if they make collection in a new place or when new administrator is in the office it become tedious for him or her to trace collection grounds.

Without an automated garbage management system, there are also problems in keeping track of the actual amount of each and every collection made in the entire county. In addition, there is also no alert available when the quantity of garbage increases in mostly affected areas is above its par level of collection.

**1.2 Background**

The percentage of people disposing garbage is increasing day by day due to awareness to garbage collection sites for counties. The garbage disposed in different places to be managed thoroughly so that there will be no negative effect to the residents in the area of disposal site. The current system of garbage collection is mostly manual. In order to arrange for collection the information, have to follow the record as it is arranged by county’s management office and allocate collection arrangement as per the information sent by user of the system in different places of the entire county. Since there are different collection areas, the collection criteria method causes problems like sending collectors in wrong places. This is not only time wasting but also causes mix ups information .Which is why there is need for an automated system that minimize if not eliminate most of the problems of the manual system.

**1.3 Brief project description**

This is a Garbage application system that is used as a means of curbing issues of uncollected garbage in different parts of the county. Each place has its own collection period which runs every time a person from a certain county takes a garbage photo and provides required information for collection.

The system keeps the record of all the users, administrator, collection status, collection area and collected . This project intends to computerize the garbage management system in a county in order to improve the information management efficiency due to the grown size of records of data for garbage.

**1.4 Purpose/objectives/justification of the project**

The purpose of the garbage management project is to simplify and automate the process of searching for uncollected garbage in the county for maintaining cleanliness and also ensure there is reduced contamination of airborne diseases like cholera which is very dangerous to humans living in certain highly populated places and maintain the records of collection places, users and system administrators**.**

**The goals and objectives of the Garbage Collection Application project are as follows:**

1. To provide a means for collection of uncollected garbage.

2. To allow any county citizens to be able to use report a garbage only if he/she has smart phone.

3. To provide an efficient information keeping for garbage after collection.

4. To improve the cleanliness in the county for instance the air pollution.

5. To provide synchronized and centralized garbage collection database.

6. To provide immediate storage and retrieval of data and information about garbage in different areas.

**2. Literature Review**

**2.1 previous and current work, methods and procedures**

There is one system that has been selected as benchmark for the development of Garbage Collection Application. The Smart Garbage Monitoring System for Waste Management.

The website for The Smart Garbage Monitoring System for Waste Management is a website that provides the facility for the citizen to register himself or herself as a system user. Only citizens in Malaysia can register to the system. Piles of rubbish are one of the major problems faced by most people in Malaysia especially those living in flats. The Smart Garbage Monitoring System for Waste Management measures waste level in garbage bin real-time and to alert the municipality, in particular cases, via SMS. It supposes to generate and send the warning messages to the municipality via SMS when the waste bin is full or almost full, so the garbage can be collected immediately.

However, this Smart Garbage Monitoring System for Waste Management does not provide any easy usage of the system. Therefore, they cannot know how is collection statuses in case of any fault. As for residents, they cannot know what’s happening in case a fault occurs. Without having this function to access administrator in case of fault in the system, the resident cannot monitor his or her health condition in term of garbage collection . This will make the citizens(users) become unaware of why the garbage is still pending uncollected.

**3. System Functional Specification**

**3.1 Functions Performed**

Citizens(users) registration-The system allows county residents to register as the users and saves the information in county database.

* Home Page- has some guidelines on the work of the county responsibilities and how the county does it .
* Accounts- User can update their profile by updating their phone numbers, email and full names in case there was an error in the names during registration. In the account, one can send a place of garbage requesting its collection.
* Report garbage- this is the activity where the a logged in user can open a camera and take a photo of the garbage and save it in the gallery for upload to county databases also the user can view whether the photo is saved in the gallery.
* Choose image- this activity gives for user to upload a garbage photo taken in the database
* Administrator Module- The admin has a lot of roles in our Garbage Collection Application. The admin manages the requests by approving them so the garbage can be collected. The admin is responsible for deleting collected garbage photo after collection has been done. The admin can see the information posted by the user of the system.

**3.2 User Interface Design**

Being a Garbage Collection System, I choose my primary color to be green as a symbolic representation of environment due to vegetation. The user interface is easy to use for users and administrators. The following are the User interface components of my system:

* Welcome Activity- this is a xml provision which contains the buttons which on pressed open the pages as per the name of the button. Its offers quick and easy to use navigation through the various sections of my mobile application. The welcome activity appears when the app icon is clicked. The button in the welcome activity is labeled according to the modules they are linked to so it makes it more easier to navigate through my mobile application .The following are the navigation buttons in my system;
* Proceed- on click it opens the Report garbage activity where there is camera.
* Sign-up Activity – it registers the user.
* Choose image – on click it opens the images then you the image for garbage to be uploaded.
* Upload – on click it upload the selected image in the database .
* Big star- it displays selected.
* Camera-on click it opens the phone camera so that photo can be taken
* Gallery – on click it opens the gallery to confirm the image.
* Report Garbage- contains/holds the contents of the Camera, gallery, big star and proceed button.
* Main2Activity*-* it is constant in all the buttons of the mobile application such buttons includes upload, choose image and admin.
* Admin – on click it registers administrator of the system.

**3.3 System Database/File Structure Preview**

Am using Fire-base Database for my system. The database has the following key tables:

* Administrator- contains administrator details like email and his log in password.
* User Register- contains all details of the user, which he/she enters during the registration process. These details are; full names, email, phone number, place of garbage and password.
* Uploads- stores the uploaded image and the image name.

File Structure Preview- i have structured the system files in the following manner.

* Administrator- contains some of java the files that represents actions that the administrator can perform. These actions are; managing users, managing uploaded images(collection status either collected or uncollected) and managing the collections in different places.
* Users- has the java file for the users actions, the user is responsible for uploading garbage to the system.
* Welcome page- it is an include folder which contains xml and java files necessary user actions from proceed button there is user registration.
* Report Garbage Activity - contains the java files necessary for the user actions. The actions performed by a user include; taking photo with a camera, view taken from the gallery.

**3.4 External and Internal Limitations and Restrictions**

* It is difficult to know whether information given by the users on registration is correct.
* Power blackouts system out of use.
* Low Internet connectivity also slows down the working of the my Garbage collection Application

**3.5 User Interface Specification.**

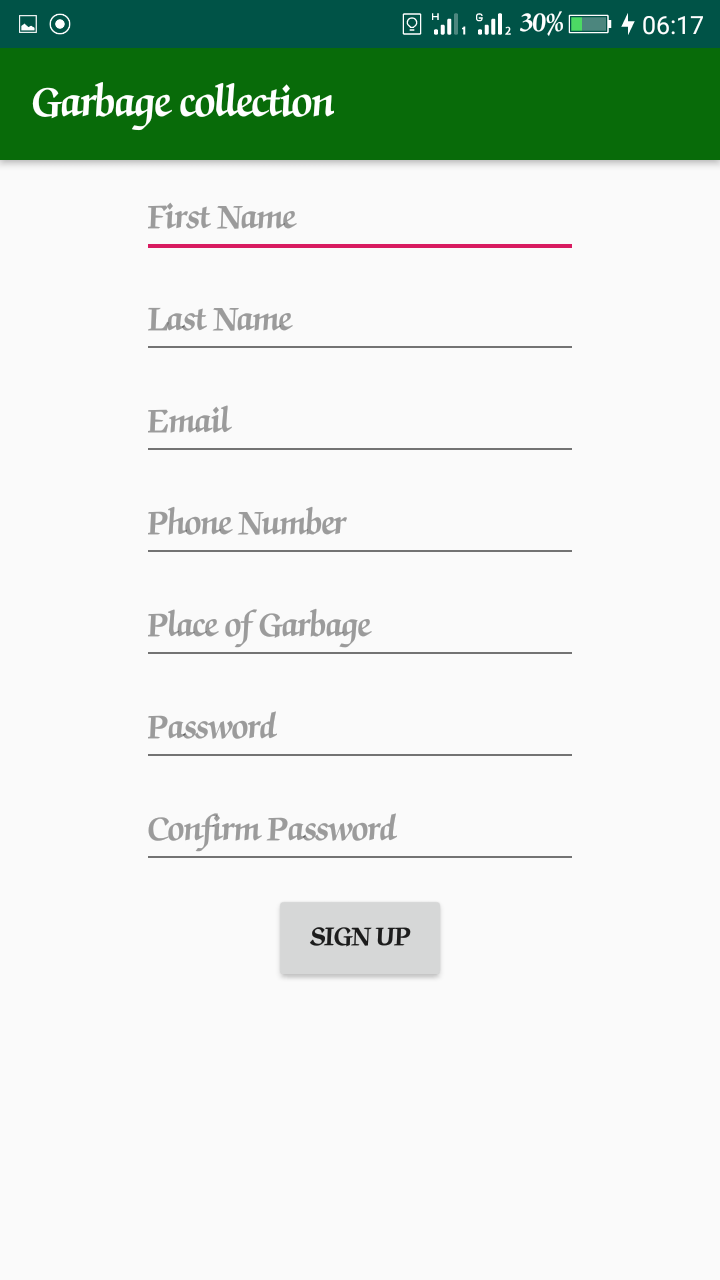
In this system i communicate with the users using the user interfaces. Some of the interfaces are:

* Welcome Page- displays the general information of the Garbage collection system
* Sign-up Activity Page- displays registration form to users
* Report garbage page- where the user can take photo and view taken photo in gallery.
* Main Activity2 page- displays buttons for upload and image selection for the user.

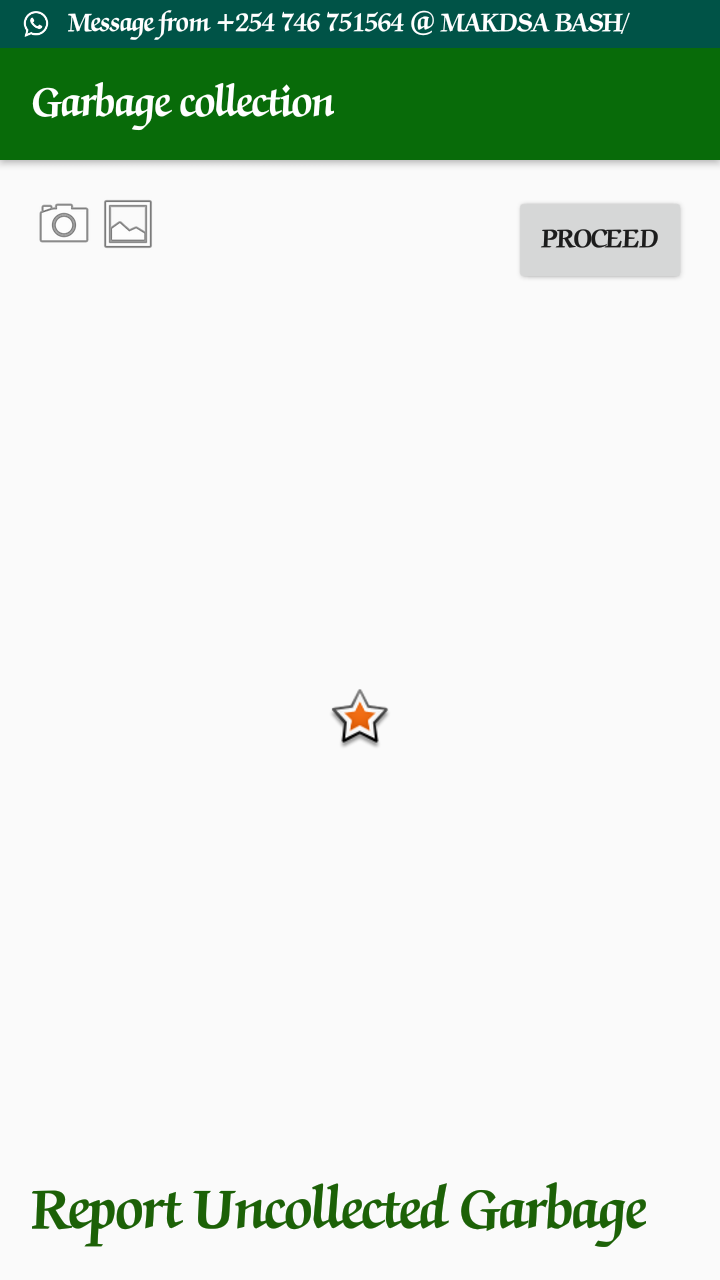
**3.5.1 User Screens/Dialog**

The following are the user screens for the Garbage collection System

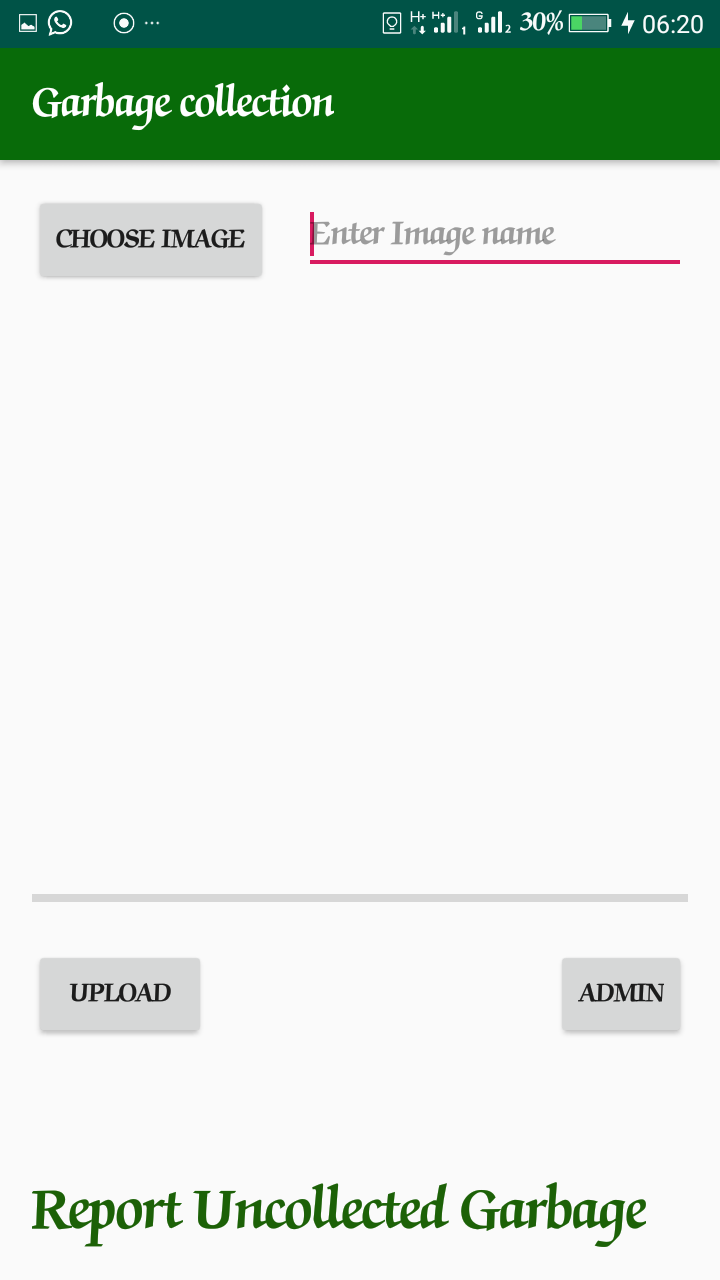
1. **Garbage Collection Welcome Page**

**b) Garbage Collection User Register Page**

**c) Garbage Collection Report Garbage Page**

****

**d) Garbage Collection Main2Activity Page**

****

**3.5.2 Report Formats/Sample Data**

My system just like any other systems generates reports. The reports include: garbage photos, number of garbage photos and collection reports. The amount of photos is displayed in admin page. If garbage is reported from a certain area, the photo is immediately uploaded. The number of uploaded photos is a report which is visible to the garbage collection administrator. It shows all garbage photo requests, the person requesting for collection, phone number, email, place of garbage and the required date. From this the administrator can confirm the request of collection. The administrator can prove the requests and the garbage can be collected as first as possible if all the conditions are taken in measure.

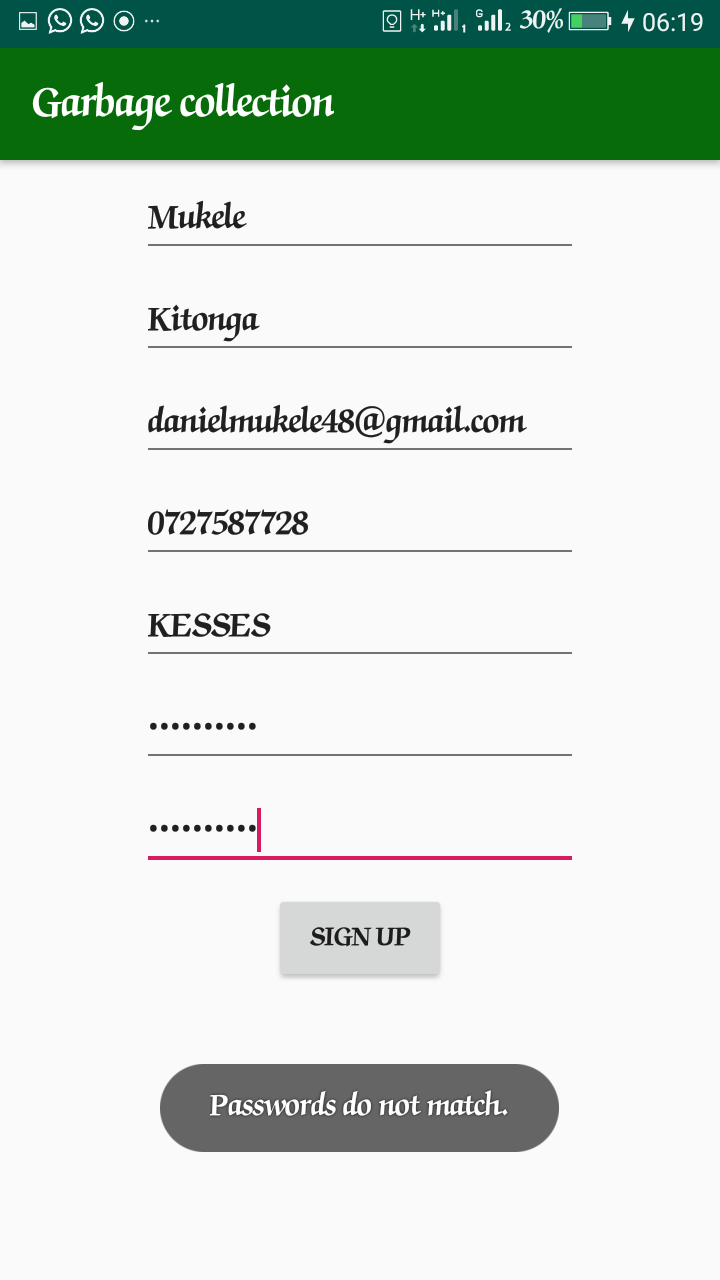
**3.5.3 Online Help Material**

For my Garbage Collection System i have designed it in such a way that it is easy to use. Any person who has the idea of smart phone usage can use the mobile application very easily without any complication if there is sufficient or fast internet connectivity.

**3.5.4 Error Conditions and System Messages**

The following are the examples of the error conditions in my system:

|  |  |
| --- | --- |
| Condition | Message |
| Wrong user registration details | Error while registering |
| Email and mobile already in the database | Email or Mobile Number already exists. Please Try another one |
| Firebase app is not initialized | After using different types of dependencies in android studio |
| Wrong email and password | Email or Password does not exists |
| Passwords didn’t match | Passwords does not Match. Please Try Again |

The following is a screenshot for the error condition

following are the examples of the system messages

|  |  |
| --- | --- |
| Condition | Message |
| User creates a new account | Successful registered |
| User upload photo | Photo uploaded successfully |

**3.5.5 Control Functions**

The main control functions in my mobile application are performed by the user and the admin

* Admin Control Functions

1. approving collected garbage
2. approving uncollected garbage
3. Managing users

* User Control Function
* Report garbage
* Register for system’s usage
* Resend collection information if it wasn’t collected

**4. System Performance Requirements**

**4.1 Efficiency**

Compared to the manual garbage collection system where all the files are kept at one place and the records are taken manually my system is more efficient given that I am keeping my records electronically and are at one database.

My system reduces a lot of paper work in the garbage collection exercise since the user has to input the details in my system.

Immediate response for garbage collection requests is very possible given that the requests go directly to the garbage collection admin and he approves the requests within no time. The manual one takes time to approve the requests.

**4.2 Reliability**

Our Garbage Collection system is reliable, that is it can be online for a long time without needing any additional maintenance. However the app’s maintenance needs to be done manually.

**4.2.1 Description of Reliability Measures**

In order to achieve full reliability of my system, i have taken the following measures:

**Consistency**- The mobile application has consistency in the user interface and user inputs. This consistency ensures users develop quick familiarity with the application so as to solve their problems more efficiently.

**Accuracy** – The Garbage Collection System ensures that correct user information is taken during the user registration process.

**Availability** – My Garbage Collection System is online for 24 hours therefore making it reliable for all the times even at the late night hours

**4.3 Security**

**4.3.1 Hardware Security**

For hardware security i have the following measures in place for my system

* Secure server place-my server for all the Garbage sites in the county will be at secure locked place where only limited persons have the key to the server room

**4.3.2 Software Security**

I have put measures to ensure the security of my Garbage collection System. These include

* Authentication -Users have to enter their email addresses and password in order to access their accounts. This controls the number of malicious users of the system.
* Password hashing- This ensures that even the database administrator does not have access to user passwords so only the user knows his/her password.
* Administrator privileges- limits information access to malicious persons
* Anti-virus- we have anti-virus programmers to prevent virus and malware attacks

**4.3.3 Data Security**

The measures put in place to ensure the security of data in the system include:

* Data encryption- Only specific people can decrypt the data making it reach only the intended persons.
* Data access Control- specific information is only available to specific people. This limits the number of people who knows the system.
* Passwords- we are using passwords which are hard to guess to prevent unauthorized access to the system
* Keeping all my software up to date.

**4.4 Maintainability**

My Garbage Collection System is easy to maintain and the maintenance process takes one in a while.

**4.5 Modifiability**

My Garbage Collection System code base can be easily modified without breaking any of the core functionalities of the system.

The user interface styling is modular therefore components can be swapped without much overhead.For modules as they are independent i can easily modify the user’s module without affecting the admin module and vice versa.

**4.6 Portability**

The system can run only run any Android platform since it is a mobile based application. It can run on most modern Android versions like Android 8.1. Older Android versions are supported through fall back code.

**5. System Design Overview**

**5.1 System Data Flow Diagrams**

This is the System Data Flow for our Blood Bank System.

1. User Data Flow Diagrams

Users

Account

b) Admin Data Flow Diagram

Admin

Account

**5.2 System Architecture and Structure**

The software system is designed around simplicity. The system core functions are to take photo, view in gallery, upload photo and create records. But these functions are accessible depending on the type of the user of the system. Normal users can only create accounts, take photo, view photo and upload the photo. The administrator can delete other administrators, delete photos, delete users and administer for collection.. The system also has a front end that serves to present organized data to users and administrators.

The structure of the application system is such that it include:

* Authentication subsystem
* Report Garbage subsystem
* Garbage subsystem
* Admin panel Subsystem

**5.3 System Data Dictionary**

Admin

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Links to** | **Comments** | **MIME** |
| admin\_id | int(10) | No |  |  |  |  |
| Email | varchar(20) | Yes | *NULL* |  |  |  |
| Password | varchar(20) | Yes | *NULL* |  |  |  |

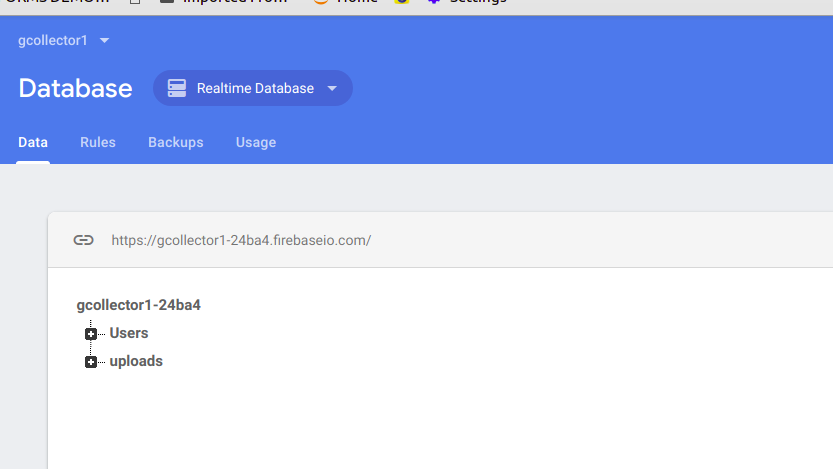
* 1. Users

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Links to** | **Comments** | **MIME** |
| First name | varchar(30) | No |  |  |  |  |
| Last name | varchar(30) | No |  |  |  |  |
| Email | varchar(10) | No |  |  |  |  |
| Phone number | int(10) | No |  |  |  |  |
| Password | Varchar (6) | No |  |  |  |  |
| Mobile | int(15) | No |  |  |  |  |
| Email | varchar(100) | No |  |  |  |  |
| Password | varchar(500) | No |  |  |  |  |

* 1. Uploads

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column** | **Type** | **Null** | **Default** | **Links to** | **Comments** | **MIME** |
| Image name | varchar(10) | No |  |  |  |  |
| Image url | varchar(10) | No |  |  |  |  |
| Image | picture | No |  |  |  |  |
| Date/time | Date | No |  |  |  |  |

This screenshot for the data dictionary



**5.4 Description of System Operation**

My Garbage Collect system operates in the following manner. Normally the user registers with the system by giving the following details: First Name, Second Name, Email, Mobile Number, Place of garbage, Password and password confirmation. This information is stored in my database. The users cannot access their accounts once they have uploaded garbage photo to be collected. Once the user wants to use the application again he/she will be prompted to register the garbage again. After uploading the garbage photo for collection, the garbage collector Administrator goes ahead to approve this requests by going to collect the garbage as per the request details. The administrator also can send emails or sms messages to users to request for more information. For every collection, there must be the administrator prove which prevents users from requesting for collection of garbage which is not there.

**5.5 Equipment Configuration**

In order for my system to work properly, the only equipment configuration that needs to be done is to have internet access in the mobile phones,laptops and desktop computers since our system cannot work off line. This configuration is very easy to achieve and anyone can be able to configuration his/her device to access internet.

**5.6 Implementation Languages**

I have used the following programming languages to come up with the Garbage Collection Mobile Application the front end and back end of my system;

**Front End**

* XML – Defines the structure of the App and how it will appear on android phone.

**Back End**

* JAVA - it is a functionalities based and server side language that is supported by most web hosting companies it is also easy to learn and implement.
* Json- for validations
* Fire base - Am using fire-base database to store user and administrator data and garbage data. Fire-base is a very secure database and that’s why I preferred to use it. Its also real-time database.

**5.7 Required Support Software**

* Android Phone – for enabling the user to access the software where one can click the icon to access my mobile application.
* Browser- this is to enable the administrator access my real-time database.

**6. System Data Structure Specifications**

**6.1 User Input Specification**

**6.1.1 Identification of Input Data**

Input data for the system in sub-classes is in different categories;

* Registration input data for Users-it is a form and also has place holders
* Photo input data - This is data about important information that is input to be viewed on the admin page.

**6.1.2 Source of Input Data (NOT input device)**

Sources of the input data are 3; the admin(s) and the user(s) .

* Admin

Login Input Data

- An admin provides login credentials at the admin login portal.

* User

Login Input Data

- A user provides login credentials at the user’s login portal.

Photo Input Data

-A user provides photo input data of the garbage to be collected.

**6.1.3 Input Medium and/or Device**

- The primary input medium is a keyboard and the device can be any device which is android.

**6.1.4 Data Format/Syntax**

* Registration input data

User Registration

Any user can register through the user registration form available in the app by clicking signupActivity.

- The form has seven form input fields which must be filled, First Name (characters for the first name), Last Name (characters for the last name), E-mail (valid email address), Phone Number(integers),Place of Garbage(characters for the place of garbage) Password (alphanumeric characters) and Confirm Password (the same as input to Password).

Admin Registration

- Admin can register through the admin registration form available by clicking Admin button.

- The form has two form input fields which must be filled, E-mail (valid email address), Password (alphanumeric characters) and Confirm Password. (The same as input to Password)

**6.2 User Output Specification**

**6.2.1 Identification of Output Data**

Output data is obtained after input data is provided and thus the subclasses are also 7;

* Registration Output Data

This is data that is output after the Sign Up button is clicked during account registration.

* ReportGarbage Output Data

This is data that is output after the ReportGarbage button is clicked.

**6.2.2 Output Medium and/or Device**

- The primary output medium is a screen and the device can be any device which can access the World Wide Web.

**6.2.3 Output Format/Syntax**

In every output subclass there are two kinds of output; standard output (successful form submission) and standard error (unsuccessful form submission).

**User Registration**

Standard Output

No output but redirected to registered account.

Standard Error

If any fields are unfilled a tool tip pops up at the field with the message Please fill out this field.

If an account with similar credentials exists, Sorry! Email already exists. Please try another one.

* Login Output Data

Standard Output

All login portals redirect the user their matching account, but there is no output.

Standard Error

If user does not provide an email in correct format, Please include ‘@’ in your email address.’<Email characters>’ is missing a ‘@’.

If user credentials are wrong, Email or Password does not exist.

If any field is unfilled a tool tip pops up at the field with the message Please fill out this field.

**6.2.4 Output Interpretation (meaning of output)**

**Registration Output Data**

Successfully registered. Click here to login and successfully registered. Users may now login alert messages mean account registration was successful.

Sorry! Email already exists. Please try another one means that an account with the same credentials exists so you must have already registered.

Please fill out this field means a form field that must be filled was not filled.

**Login Output Data**

Please include ‘@’ in your email address.’<email\_characters>’ is missing an ‘@’ means that the format of the email address entered was incorrect.

Email or Password does not exist means login credentials are wrong.

Please fill out this field means a field that must be filled was not filled.

**6.3 System Data Base/File Structure Specification**

**6.3.1 Identification of Data Base/Files**

The Garbage collection mobile app uses Firebase database system and only has a single database named gcollector1 with 3 tables in it;

Users-Admin data

Users- users data

uploads – uploaded data

**6.3.3 Logical File Structure**

The logical file structure of the database is managed by the google storage engine.

Record Format

The record/row format used in the tables in the database is the compact row and column format.

File organization

Logical file organization is done through per-table table spaces. A table space is a logical group of one or more files in a database.

Access Methods

In databases using fire-base storage engine data is accessed from table asynchronously (where possible) or sequentially.

**6.3.4 Physical File Structure**

Storage Device

Physical files are stored in a directory name gcollector1 in the physical storage/ hard drive of the computer hosting the fire-base real-time database website.

File organization

The physical files that collective form a logical database are stored in a folder named gcollector1. The physical structure of the gcollector1 database is shown in the tree below;

|---Firebase-----gcollector1-----|  
 |---Users

|---uploads

|---collector

7. **Module Design specifications**

**7.1 Admin module**

**7.1.1 Module Functional specification**

**Functions Performed**

* Registration

The admin module registers the accounts of other admin accounts.

* Management

The admin module manages user uploads, user accounts and the collection of garbage. Management here means collecting, pending or deleting them.

**7.2 User module**

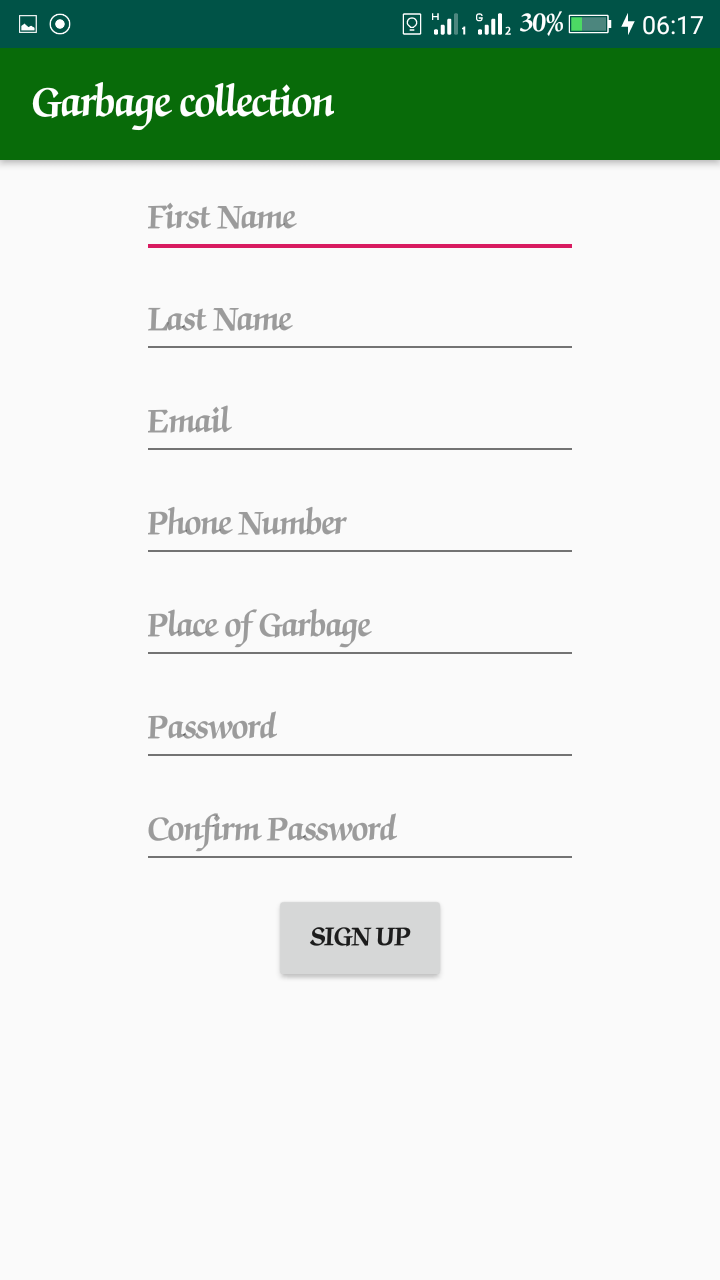
**7.2.1 Module Functional specification**

Doctor module can add a new donor/recipient. Functions Performed

* garbage photo uploading

User module performs garbage photo uploading by giving information about the garbage to be collected and the place of collection.

**Register**

****

* Addition

Request Addition

First Name, last name, mobile number, email, place of garbage,password and confirm password all the fields are needed for inputs in the form.

Output

Its output is an array of data fetched from the requests table and presented on a table.

**Description of Module Operation**

After the user has input a valid email address and password and clicked the Sign in button they will be logged into their account. Otherwise an alert will pop up with the message Email or Password does not exist.

In the logged in account there are several buttons on the next activities.

The buttons are;

Camera-The user can click on the camera icon to take picture of the garbage to be reported and upload it giving the required details.

Gallery-This is icon where photos are stored after they are taken by a camera if the photo was taken in absence of internet the user after getting internet access can retrieve the photo from the gallery and update it.

Proceed – This button gives room for the previous activities.

**8. System Verification**

Software testing method is carried out to verify that the software package functions as mentioned in the software requirement document. The core purpose for testing is to detect the software function issues so that they are corrected and perfected.

**8.1 Unit Testing**

**Unit Testing** is a level of software **testing** where individual units/ components of a software are tested. The purpose is to validate that each **unit** of the software performs as designed. A **unit** is the smallest testable part of software. It usually has one or a few inputs and usually a single output.

The terms used in this section:

Test Scenario - is any functionality that can be tested.

Test Case - a case that sets a precedent for other cases involving the same question of law.

Precondition - a condition that must be fulfilled before other things can happen or be done

Test Step – The procedures that are outlined to cover for the unit testing procedure.

Test Data - is **data** which has been specifically identified for use in **tests**, typically of a computer program

Expected Result - The final result that the tester is expected to get on the test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Scenario | Test Case | Precondition | Test Step | Test Data | Expected Result |
| Check login functionality for user. | Check response on entering valid user email and password | Internet connection must be available | Launch the android phone.  Click on register tab.  Enter valid user email address and Password.  Click on the register button. | User email address: | Login Must be Successful |

**Discussion of Test Results**

From the above test the result must be successful only if the valid credentials are used during the verification process in the system.

If the wrong credentials are used, the system prompts a message to the user to input the correct details in the system.

**9. Conclusions**

**9.1 Summary**

Technology is introducing new innovations day by day, thus reducing the time required to do things. The Garbage Collection system can be used to reduce the time required to collect garbage in major towns.

The mobile application provides a way of communication between the citizens and the county administration.

The database is a vital aspect of the system. The database of the garbage collection must be checked for consistency on regular basis for smooth working of the system.

The new automated garbage collection system is highly reliable, easy, fast and consistent and will play a crucial role for reliable service for residents and for the management

**9.2 Problems encountered and solved**

1 .Scope changes.

I had to modify some parts of the project as to make it better and this caused it to drift. Ultimately, am able to adjust, put in some effort and beat the deadline.

2. Inadequate resources

As there was no one in the group with expertise in the health sector i had to visit some areas in the county and consult county authority on the mobile application system. This was time consuming and needed funds. I solved this by raising funds.

**9.3 Suggestions for better approaches and future extensions to the project**

1. Application of robotics in garbage collection.

**10. BIBLIOGRAPHY**

List the books and the links used in the project report.

[A](http://www.bloodbank.appslab.co.ke/admin)ndroid online tutorials book

**11. APPENDIX**

Appendix I

**Glossary**

Admin – Administrator

System menu- The overview User interphase if the system.

UI - user inter phase

URL – Uniform resource locator.

Appendix II

Financial Statement for setting up the system.

Appendix III

|  |  |
| --- | --- |
| **Comodity** | **Cost (Kshs)** |
| Computer(s) | 60,000 + |
| Internet connection per month subscription | 2,000 |
| Stationary (notebook & pens) | 5 |
|  | 3,500 |
| Anti virus | 3,500 |
| Other expenses | 500 |
| **TOTAL** | **Ksh. 70,000** |

**Memorandum of agreement with partners and software users**

Appendix IV

Specific study and evaluation of reports

The system is designed to help manage the garbage collection procedure.

|  |  |
| --- | --- |
| **Comodity** | **Cost (Kshs)** |
| Computer(s) | 60,000 + |
| Internet connection per month subscription | 2,000 |
| Stationary (notebook & pens) | 500 |
| Android phone | 3,500 |
| Anti virus | 3,500 |
| Other expenses | 500 |
| **TOTAL** | **Ksh. 70,000** |

**12. Program Listings**

**Admin module**

**//Activity login\_user.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context=".login\_user">

<EditText

android:id="@+id/email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email" />

<EditText

android:id="@+id/password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password" />

<EditText

android:id="@+id/passconfirm"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="confirm password" />

<Button

android:id="@+id/login"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login" />

<Button

android:id="@+id/registration"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Registration" />

</LinearLayout>

**//Login\_user.java**

**package com.root.gcolletctor;**

import android.content.Intent;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

public class login\_user extends AppCompatActivity {

private EditText mEmail, mPassword, mPlace;

private Button mLogin, mRegistration;

private FirebaseAuth mAuth;

private FirebaseAuth.AuthStateListener firebaseAuthListener;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_login\_user*);

mAuth = FirebaseAuth.*getInstance*();

firebaseAuthListener = new FirebaseAuth.AuthStateListener() {

@Override

public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {

FirebaseUser user = FirebaseAuth.*getInstance*().getCurrentUser();

if (user != null) {

Intent intent = new Intent(login\_user.this, Adminn.class);

startActivity(intent);

finish();

return;

}

}

};

mEmail = (EditText) findViewById(R.id.*email*);

mPassword = (EditText) findViewById(R.id.*password*);

mPlace = (EditText) findViewById(R.id.*passconfirm*);

mLogin = (Button) findViewById(R.id.*login*);

mRegistration = (Button) findViewById(R.id.*registration*);

mRegistration.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

final String email = mEmail.getText().toString();

final String password = mPassword.getText().toString();

mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(login\_user.this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (!task.isSuccessful()) {

Toast.*makeText*(login\_user.this, "sign-up error", Toast.*LENGTH\_SHORT*).show();

} else {

String user\_id = mAuth.getCurrentUser().getUid();

DatabaseReference current\_user\_db = FirebaseDatabase.*getInstance*().getReference().child("Users").child("collectors").child(user\_id);

current\_user\_db.setValue(true);

}

}

});

}

});

mLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

final String email = mEmail.getText().toString();

final String password = mPassword.getText().toString();

final String place = mPlace.getText().toString();

mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(login\_user.this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (!task.isSuccessful()) {

Toast.*makeText*(login\_user.this, "sign-in error", Toast.*LENGTH\_SHORT*).show();

}

}

});

}

});

}

@Override

protected void onStart() {

super.onStart();

mAuth.addAuthStateListener(firebaseAuthListener);

}

@Override

protected void onStop() {

super.onStop();

mAuth.removeAuthStateListener(firebaseAuthListener);

}

}

**package com.root.gcolletctor;**

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.View;

import android.widget.ProgressBar;

import android.widget.Toast;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;

import java.util.List;

public class Adminn extends AppCompatActivity implements ImageAdapter.OnItemClickListener {

private RecyclerView mRecyclerView;

private ImageAdapter mAdapter;

private ProgressBar mProgressCircle;

private DatabaseReference mDatabaseRef;

private List<Upload> mUploads;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_adminn);

mRecyclerView = findViewById(R.id.recycler\_view);

mRecyclerView.setHasFixedSize(true);

mRecyclerView.setLayoutManager(new LinearLayoutManager(this));

mProgressCircle = findViewById(R.id.progress\_circle);

mUploads = new ArrayList<>();

mDatabaseRef = FirebaseDatabase.getInstance().getReference("uploads");

mDatabaseRef.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {

Upload upload = postSnapshot.getValue(Upload.class);

mUploads.add(upload);

}

mAdapter = new ImageAdapter(Adminn.this, mUploads);

mRecyclerView.setAdapter(mAdapter);

mAdapter.setOnItemClickListener(Adminn.this);

mProgressCircle.setVisibility(View.INVISIBLE);

}

@Override

public void onCancelled(DatabaseError databaseError) {

Toast.makeText(Adminn.this, databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

mProgressCircle.setVisibility(View.INVISIBLE);

}

});

}

@Override

public void onItemClick(int position) {

Toast.makeText(this, "Normal click at position:" + position, Toast.LENGTH\_SHORT).show();

}

@Override

public void onWhatEverClick(int position) {

Toast.makeText(this, "Whatever click at position:" + position, Toast.LENGTH\_SHORT).show();

}

@Override

public void onDeleteClick(int position) {

Toast.makeText(this, "Delete click at position:" + position, Toast.LENGTH\_SHORT).show();

}

}

**//admin.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".Adminn">

<ProgressBar

android:id="@+id/progress\_circle"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true" />

<android.support.v7.widget.RecyclerView

android:id="@+id/recycler\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</RelativeLayout>

**USER MODULE**

**//WelcomeActivity.java**

**package com.root.gcolletctor;**

import android.app.ProgressDialog;

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.ProgressBar;

public class WelcomeActivity extends AppCompatActivity {

private static ProgressDialog *progressDialog*;

private static Button *buttonProceed*;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_welcome*);

//get the button ui id

*buttonProceed* = (Button) findViewById(R.id.*button\_proceed*);

//create a progress dialog

*progressDialog* = new ProgressDialog(this);

//handling the button click event

*buttonProceed*.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

*progressDialog*.setMessage("..........");

Intent mainActivity = new Intent(getApplicationContext(), signUpActivity.class);

*progressDialog*.dismiss();

startActivity(mainActivity);

}

});

}

}

**// WelcomeActivity.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@drawable/backg"

android:gravity="center"

android:orientation="vertical"

tools:context=".WelcomeActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="10dp"

android:text="@string/help\_keep\_your\_county\_clean"

android:textAlignment="center"

android:textColor="#1c6107"

android:textSize="30sp"

android:textStyle="bold" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="@drawable/item\_bg"

android:gravity="center"

android:orientation="vertical">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/what\_we\_do"

android:textColor="#f4f4f7"

android:textSize="20sp"

android:textStyle="bold" />

<TextView

android:id="@+id/textView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/garbage\_collection"

android:textColor="#ffffff"

android:textStyle="italic" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/waste\_disposal"

android:textColor="#ffffff"

android:textStyle="italic" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/cleaning\_public\_resources"

android:textColor="#ffffff"

android:textStyle="italic" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/monitoring\_cleanliness\_in\_the\_county"

android:textColor="#ffffff"

android:textStyle="italic" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="50dp"

android:layout\_marginLeft="50dp"

android:layout\_marginTop="10dp"

android:text="@string/private\_cleaning\_tenders"

android:textColor="#ffffff"

android:textStyle="italic" />

<Button

android:id="@+id/button\_proceed"

android:layout\_width="200dp"

android:layout\_height="wrap\_content"

android:layout\_marginTop="40dp"

android:background="@drawable/app\_icon\_shapes"

android:padding="10dp"

android:text="proceed"

android:textColor="#ffffff"

android:textStyle="bold" />

</LinearLayout>

</LinearLayout>

**//Report garbage.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingLeft="16dp"

android:paddingTop="16dp"

android:paddingRight="16dp"

android:paddingBottom="16dp"

app:layout\_behavior="android.support.design.widget.AppBarLayout$ScrollingView"

tools:context=".ReportGarbage">

<ImageView

android:id="@+id/ivCamera"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentStart="true"

android:layout\_alignParentLeft="true"

android:layout\_alignParentTop="true"

android:src="@android:drawable/ic\_menu\_camera" />

<ImageView

android:id="@+id/ivGallery"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_toEndOf="@id/ivCamera"

android:layout\_toRightOf="@+id/ivCamera"

android:src="@android:drawable/ic\_menu\_gallery" />

<Button

android:id="@+id/ivUpload"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_alignParentEnd="true"

android:layout\_alignParentRight="true"

android:text="proceed" />

<ImageView

android:id="@+id/ivImage"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:src="@android:drawable/star\_big\_on" />

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentBottom="true"

android:text="Report Uncollected Garbage"

android:textColor="#1c6107"

android:textSize="30sp"

android:textStyle="bold" />

</RelativeLayout>

**//Reportgarbage.java**

**package com.root.gcolletctor;**

import android.app.Activity;

import android.content.ContentResolver;

import android.content.DialogInterface;

import android.content.Intent;

import android.graphics.Bitmap;

import android.icu.text.Collator;

import android.icu.text.DateIntervalFormat;

import android.net.Uri;

import android.os.Handler;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.webkit.MimeTypeMap;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.ImageView;

import android.widget.ProgressBar;

import android.widget.Toast;

import com.kosalgeek.android.photoutil.CameraPhoto;

import com.kosalgeek.android.photoutil.GalleryPhoto;

import com.kosalgeek.android.photoutil.ImageLoader;

import java.io.FileNotFoundException;

import java.io.IOException;

public class ReportGarbage extends AppCompatActivity {

private final String TAG = this.getClass().getName();

ImageView ivCamera, ivGallery, ivImage;

Button ivUpload;

CameraPhoto cameraPhoto;

GalleryPhoto galleryPhoto;

final int CAMERA\_REQUEST = 13323;

final int GALLERY\_REQUEST = 22131;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_report\_garbage*);

cameraPhoto = new CameraPhoto(getApplicationContext());

galleryPhoto = new GalleryPhoto(getApplicationContext());

ivCamera = (ImageView) findViewById(R.id.*ivCamera*);

ivImage = (ImageView) findViewById(R.id.*ivImage*);

ivGallery = (ImageView) findViewById(R.id.*ivGallery*);

ivUpload = (Button) findViewById(R.id.*ivUpload*);

ivUpload.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent mainActivity = new Intent(getApplicationContext(), Main2Activity.class);

startActivity(mainActivity);

}

});

ivCamera.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try {

startActivityForResult(cameraPhoto.takePhotoIntent(), CAMERA\_REQUEST);

cameraPhoto.addToGallery();

} catch (IOException e) {

Toast.*makeText*(getApplicationContext(),

"Something wrong while taking the photo", Toast.*LENGTH\_SHORT*).show();

}

}

});

ivGallery.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startActivityForResult(galleryPhoto.openGalleryIntent(), GALLERY\_REQUEST);

}

});

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (resultCode == *RESULT\_OK*) {

if (requestCode == CAMERA\_REQUEST) {

String photoPath = cameraPhoto.getPhotoPath();

try {

Bitmap bitmap = ImageLoader.*init*().from(photoPath).requestSize(512, 512).getBitmap();

ivImage.setImageBitmap(bitmap);

} catch (FileNotFoundException e) {

Toast.*makeText*(getApplicationContext(),

"Something wrong while loading the photo", Toast.*LENGTH\_SHORT*).show();

}

} else if (requestCode == GALLERY\_REQUEST) {

Uri uri = data.getData();

galleryPhoto.setPhotoUri(uri);

String photoPath = galleryPhoto.getPath();

try {

Bitmap bitmap = ImageLoader.*init*().from(photoPath).requestSize(512, 512).getBitmap();

ivImage.setImageBitmap(bitmap);

} catch (FileNotFoundException e) {

Toast.*makeText*(getApplicationContext(),

"Something wrong while choosing the photo", Toast.*LENGTH\_SHORT*).show();

}

}

}

}

}

**//Main2Activity.java**

**package com.root.gcolletctor;**

import android.content.ContentResolver;

import android.content.Intent;

import android.net.Uri;

import android.os.Handler;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.webkit.MimeTypeMap;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.ProgressBar;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener;

import com.google.firebase.FirebaseApp;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.OnProgressListener;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.StorageTask;

import com.google.firebase.storage.UploadTask;

import com.squareup.picasso.Picasso;

public class Main2Activity extends AppCompatActivity {

private static final int *PICK\_IMAGE\_REQUEST* = 1;

private Button mButtonChooseImage;

private Button mButtonUpload;

private Button mTextViewShowUploads;

private StorageReference mStorageRef;

private DatabaseReference mDatabaseRef;

private StorageTask mUploadTask;

private EditText mEditTextFileName;

private ImageView mImageView;

private ProgressBar mProgressBar;

private Uri mImageUri;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main2*);

mEditTextFileName = (EditText) findViewById(R.id.*edit\_text\_file\_name*);

mButtonChooseImage = (Button) findViewById(R.id.*button\_choose\_image*);

mButtonUpload = (Button) findViewById(R.id.*button\_upload*);

mImageView = (ImageView) findViewById(R.id.*image\_view*);

mProgressBar = (ProgressBar) findViewById(R.id.*progress\_bar*);

mTextViewShowUploads = (Button) findViewById(R.id.*text\_view\_show\_uploads*);

mTextViewShowUploads.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent main2Activity = new Intent(getApplicationContext(), login\_user.class);

startActivity(main2Activity);

}

});

FirebaseApp.*initializeApp*(Main2Activity.this);

mStorageRef = FirebaseStorage.*getInstance*().getReference("uploads");

mDatabaseRef = FirebaseDatabase.*getInstance*().getReference("uploads");

mButtonChooseImage.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

openFileChooser();

}

});

mButtonUpload.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (mUploadTask != null && mUploadTask.isInProgress()) {

Toast.*makeText*(Main2Activity.this, "Upload in progress", Toast.*LENGTH\_SHORT*).show();

} else {

uploadFile();

}

}

});

mTextViewShowUploads.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

openImagesActivity();

}

});

}

public void openFileChooser() {

Intent intent = new Intent();

intent.setType("image/\*");

intent.setAction(Intent.*ACTION\_GET\_CONTENT*);

startActivityForResult(intent, *PICK\_IMAGE\_REQUEST*);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == *PICK\_IMAGE\_REQUEST* && resultCode == *RESULT\_OK*

&& data != null && data.getData() != null) {

mImageUri = data.getData();

Picasso.*with*(this).load(mImageUri).into(mImageView);

}

}

private String getFileExtension(Uri uri) {

ContentResolver cR = getContentResolver();

MimeTypeMap mime = MimeTypeMap.*getSingleton*();

return mime.getExtensionFromMimeType(cR.getType(uri));

}

private void uploadFile() {

if (mImageUri != null) {

StorageReference fileReference = mStorageRef.child(System.*currentTimeMillis*()

+ "." + getFileExtension(mImageUri));

mUploadTask = fileReference.putFile(mImageUri)

.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {

@Override

public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

Handler handler = new Handler();

handler.postDelayed(new Runnable() {

@Override

public void run() {

mProgressBar.setProgress(0);

}

}, 500);

Toast.*makeText*(Main2Activity.this, "Upload successful", Toast.*LENGTH\_LONG*).show();

Upload upload = new Upload(mEditTextFileName.getText().toString().trim(),

mStorageRef.getDownloadUrl().toString());

String uploadId = mDatabaseRef.push().getKey();

mDatabaseRef.child(uploadId).setValue(upload);

}

})

.addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.*makeText*(Main2Activity.this, e.getMessage(), Toast.*LENGTH\_SHORT*).show();

}

})

.addOnProgressListener(new OnProgressListener<UploadTask.TaskSnapshot>() {

@Override

public void onProgress(UploadTask.TaskSnapshot taskSnapshot) {

double progress = (100.0 \* taskSnapshot.getBytesTransferred() / taskSnapshot.getTotalByteCount());

mProgressBar.setProgress((int) progress);

}

});

} else {

Toast.*makeText*(this, "No image selected", Toast.*LENGTH\_SHORT*).show();

}

}

private void openImagesActivity() {

Intent intent = new Intent(Main2Activity.this, Adminn.class);

startActivity(intent);

}

}

**//Main2Activity.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingLeft="16dp"

android:paddingTop="16dp"

android:paddingRight="16dp"

android:paddingBottom="16dp"

app:layout\_behavior="android.support.design.widget.AppBarLayout$ScrollingView"

tools:context=".ReportGarbage">

<Button

android:id="@+id/button\_choose\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Choose image" />

<Button

android:id="@+id/button\_upload"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_marginBottom="103dp"

android:text="UPLOAD" />

<Button

android:id="@+id/text\_view\_show\_uploads"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignBaseline="@+id/button\_upload"

android:layout\_alignBottom="@+id/button\_upload"

android:layout\_marginLeft="173dp"

android:layout\_marginBottom="2dp"

android:layout\_toRightOf="@+id/button\_upload"

android:text="Admin"

android:textSize="14sp" />

<EditText

android:id="@+id/edit\_text\_file\_name"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginStart="16dp"

android:layout\_marginLeft="16dp"

android:layout\_toRightOf="@+id/button\_choose\_image"

android:hint="Enter Image name" />

<ImageView

android:id="@+id/image\_view"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_above="@+id/progress\_bar"

android:layout\_below="@+id/edit\_text\_file\_name"

android:layout\_marginTop="16dp" />

<ProgressBar

android:id="@+id/progress\_bar"

style="@style/Widget.AppCompat.ProgressBar.Horizontal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_above="@+id/button\_upload"

android:layout\_marginBottom="16dp" />

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentBottom="true"

android:text="Report Uncollected Garbage"

android:textColor="#1c6107"

android:textSize="30sp"

android:textStyle="bold" />

</RelativeLayout>

**//Activitymain.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context=".MainActivity">

</LinearLayout>

**//MainActivity.java**

**package com.root.gcolletctor;**

import android.content.Intent;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.*menu*, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.*menu\_add\_new*:

Intent reportIntent = new Intent(getApplicationContext(), ReportGarbage.class);

startActivity(reportIntent);

break;

}

return super.onOptionsItemSelected(item);

}

}

**//Uploads.java**

**package com.root.gcolletctor;**

public class Upload {

private String mName;

private String mImageUrl;

public Upload() {

//empty constructor needed

}

public Upload(String name, String imageUrl) {

if (name.trim().equals("")) {

name = "No name";

}

mName = name;

mImageUrl = imageUrl;

}

public String getName() {

return mName;

}

public void setName(String name) {

mName = name;

}

public String getmImageUrl() {

return mImageUrl;

}

public void setImageUrl(String imageUrl) {

mImageUrl = imageUrl;

}

}

**//SignUpActivity.java**

**package com.root.gcolletctor;**

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

public class signUpActivity extends AppCompatActivity {

TextView editTextSUFirstName;

TextView editTextSULastName;

TextView editTextSUEmail;

TextView editTextSUPhone;

TextView editTextSUaddress;

TextView editTextSUPassword;

TextView editTextSUConfPassword;

Button buttonSUSignUp;

FirebaseAuth authRef = FirebaseAuth.*getInstance*();

DatabaseReference rootRef = FirebaseDatabase.*getInstance*().getReference();

DatabaseReference usersRef = rootRef.child("Users");

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_sign\_up*);

editTextSUFirstName = (TextView) findViewById(R.id.*editTextSUFirstName*);

editTextSULastName = (TextView) findViewById(R.id.*editTextSULastName*);

editTextSUEmail = (TextView) findViewById(R.id.*editTextSUEmail*);

editTextSUPhone = (TextView) findViewById(R.id.*editTextSUPhone*);

editTextSUaddress = (TextView) findViewById(R.id.*editTextSUaddress*);

editTextSUPassword = (TextView) findViewById(R.id.*editTextSUPassword*);

editTextSUConfPassword = (TextView) findViewById(R.id.*editTextSUConfPassword*);

buttonSUSignUp = (Button) findViewById(R.id.*buttonSUSignUp*);

}

@Override

protected void onStart() {

super.onStart();

// Login

buttonSUSignUp.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

signUp();

}

;

});

}

private void signUp() {

final String email = editTextSUEmail.getText().toString().trim();

final String firstName = editTextSUFirstName.getText().toString().trim();

final String lastName = editTextSULastName.getText().toString().trim();

final String phoneNumber = editTextSUPhone.getText().toString().trim();

final String addressCode = editTextSUaddress.getText().toString().trim();

final String password = editTextSUPassword.getText().toString().trim();

final String confPassword = editTextSUConfPassword.getText().toString().trim();

// Validate all fields

if (firstName.isEmpty() || lastName.isEmpty() || phoneNumber.isEmpty() || addressCode.isEmpty()) {

Toast.*makeText*(signUpActivity.this, "Please fill all fields.", Toast.*LENGTH\_SHORT*).show();

} else if (!password.equals(confPassword)) {

Toast.*makeText*(signUpActivity.this, "Passwords do not match.", Toast.*LENGTH\_SHORT*).show();

} else {

// Create user

authRef.createUserWithEmailAndPassword(email, password).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if (task.isSuccessful()) {

Toast.*makeText*(signUpActivity.this, "Sign up successful!", Toast.*LENGTH\_LONG*).show();

FirebaseUser user = task.getResult().getUser();

String UID = user.getUid();

User newUser = new User(firstName, lastName, email, phoneNumber, addressCode);

usersRef.child(UID).setValue(newUser);

finish();

startActivity(new Intent(signUpActivity.this, ReportGarbage.class));

} else {

try {

throw task.getException();

} catch (Exception e) {

Toast.*makeText*(signUpActivity.this, task.getException().getMessage(), Toast.*LENGTH\_LONG*).show();

}

}

}

});

}

}

}

**//ActivitySignup.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".signUpActivity">

<android.support.constraint.ConstraintLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<!-- First NAME \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUFirstName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="First Name"

android:inputType="textCapWords"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<!-- LAST NAME \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSULastName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Last Name"

android:inputType="textCapWords"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUFirstName" />

<!-- EMAIL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUEmail"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Email"

android:inputType="textEmailAddress"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSULastName" />

<!-- PHONE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUPhone"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Phone Number"

android:inputType="phone"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUEmail" />

<!-- PLACE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUaddress"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Place of Garbage"

android:inputType="textCapCharacters"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUPhone" />

<!-- PASSWORD \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUPassword"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Password"

android:inputType="textPassword"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUaddress" />

<!-- PASSWORD CONFIRMATION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<EditText

android:id="@+id/editTextSUConfPassword"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:ems="10"

android:hint="Confirm Password"

android:inputType="textPassword"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUPassword" />

<!-- BUTTONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<Button

android:id="@+id/buttonSUSignUp"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

android:layout\_marginTop="8dp"

android:layout\_marginRight="8dp"

android:text="Sign Up"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/editTextSUConfPassword" />

</android.support.constraint.ConstraintLayout>

</ScrollView>

**//Image Adapter.java**

**package com.root.gcolletctor;**

import android.content.Context;

import android.support.v7.widget.RecyclerView;

import android.view.ContextMenu;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import com.bumptech.glide.Glide;

import com.bumptech.glide.request.RequestOptions;

import com.squareup.picasso.Picasso;

import java.util.List;

public class ImageAdapter extends RecyclerView.Adapter<ImageAdapter.ImageViewHolder> {

private Context mContext;

private List<Upload> mUploads;

private OnItemClickListener mListener;

public ImageAdapter(Context context, List<Upload> uploads) {

mContext = context;

mUploads = uploads;

}

@Override

public ImageViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

View v = LayoutInflater.*from*(mContext).inflate(R.layout.*image\_item*, parent, false);

return new ImageViewHolder(v);

}

@Override

public void onBindViewHolder(ImageViewHolder holder, int position) {

Upload uploadCurrent = mUploads.get(position);

holder.textViewName.setText(uploadCurrent.getName());

RequestOptions options = new RequestOptions();

options.fitCenter();

Glide.*with*(mContext)

// .load(uploadCurrent.setImageUrl());

.load(uploadCurrent.getmImageUrl());

// .apply(options)

// // .placeholder(R.mipmap.ic\_launcher)

// //.fit()

// // .centerCrop()

// .into(holder.imageView);

}

@Override

public int getItemCount() {

return mUploads.size();

}

public class ImageViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener,

View.OnCreateContextMenuListener, MenuItem.OnMenuItemClickListener {

public TextView textViewName;

public ImageView imageView;

public ImageViewHolder(View itemView) {

super(itemView);

textViewName = itemView.findViewById(R.id.*text\_view\_name*);

imageView = itemView.findViewById(R.id.*image\_view\_upload*);

itemView.setOnClickListener(this);

itemView.setOnCreateContextMenuListener(this);

}

@Override

public void onClick(View v) {

if (mListener != null) {

int position = getAdapterPosition();

if (position != RecyclerView.*NO\_POSITION*) {

mListener.onItemClick(position);

}

}

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {

menu.setHeaderTitle("Select Action");

MenuItem doWhatever = menu.add(Menu.*NONE*, 1, 1, "COLLECTED");

MenuItem delete = menu.add(Menu.*NONE*, 2, 2, "UNCOLLECTED");

doWhatever.setOnMenuItemClickListener(this);

delete.setOnMenuItemClickListener(this);

}

@Override

public boolean onMenuItemClick(MenuItem item) {

if (mListener != null) {

int position = getAdapterPosition();

if (position != RecyclerView.*NO\_POSITION*) {

switch (item.getItemId()) {

case 1:

mListener.onWhatEverClick(position);

return true;

case 2:

mListener.onDeleteClick(position);

return true;

}

}

}

return false;

}

}

public interface OnItemClickListener {

void onItemClick(int position);

void onWhatEverClick(int position);

void onDeleteClick(int position);

}

public void setOnItemClickListener(OnItemClickListener listener) {

mListener = listener;

}

}

**//user.java**

**package com.root.gcolletctor;**

public class User {

private String firstName;

private String lastName;

private String email;

private String phoneNumber;

private String addressCode;

public User() {

// Default constructor required for calls to DataSnapshot.getValue(User.class)

firstName = "";

lastName = "";

email = "";

phoneNumber = "";

addressCode = "";

}

public User(String firstName, String lastName, String email, String phoneNumber, String addressCode) {

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.phoneNumber = phoneNumber;

this.addressCode = addressCode;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public String getEmail() {

return email;

}

public String getPhoneNumber() {

return phoneNumber;

}

public String getaddressCode() {

return addressCode;

}

}

**//image\_item.xml**

**<?xml version="1.0" encoding="utf-8"?>**

<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="8dp">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/text\_view\_name"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="8dp"

android:layout\_marginBottom="8dp"

android:text="Name"

android:textColor="@android:color/black"

android:textSize="20sp" />

<ImageView

android:id="@+id/image\_view\_upload"

android:layout\_width="match\_parent"

android:layout\_height="200dp" />

</LinearLayout>

</android.support.v7.widget.CardView>

**13. USER MANUAL**

**GENERAL INFORMATION**

Garbage collection system is a mobile based application that creates a link between a citizen and a county garbage management.

**System Overview.**

Garbage collection Management System is a secure Mobile based platform design to bridge the gap between the residents and the county authority. The application has a database where all the garbage data is saved and also can be retrieved by the administrator.

**Organization of the Manual**

The user’s manual consists of five sections: General Information, System Summary, Getting Started,

Using the System, and Reporting.

General Information section explains in general terms the system and the purpose for which it is

Intended.

System Summary section provides a general overview of the system. The summary outlines the uses of

The system’s hardware and software requirements, system’s configuration, user access levels and

System’s behavior in case of any contingencies.

Getting Started section explains how to get the garbage collection System and run it in your android phone. The section presents briefly system menu.

Using The System section provides a detailed description of system functions.

Reporting section describes in what way information collected by the application are presented and how to access the information.

**SYSTEM SUMMARY**

Garbage collection system is a mobile based application that creates a link between a citizens and county authority under strict monitoring of the county garbage system administrator. The System is mobile based so meaning it will run in a android phones.

**SYSTEM CONFIGURATION**

Garbage collection Systems operates on mobile devices with Android operating system. There are no additional system configuration as it is straight forward and ready to use.

This user manual is designed to help the user of garbage collection system to be able to operate the system with a lot of ease, help the system administrator setup the system for a respective areas,

**USER ACCESS LEVELS**

The resident is able to request for collection, the request is first approved by the administrator then the collection is done from the residents.

**GETTING STARTED**.

This section provides a general walk-through of the system from initiation through exit. The logical arrangement of the information shall enable the functional personnel to understand the sequence and flow of the system. Use screen prints to depict examples of text under each heading.

**LOGING ON**

The system has 2 authentication levels, and they are the normal user-(resident/citizen) and the administrator.

Each user requires a password and email to access the system.

**SYSTEM MENU**

Here’s is the welcome screen to the system when the link is first visited by the normal user. It has the description of the whole system, this is what it does, the impact it has.



**USING THE SYSTEM ONLINE**

The User

The user can:

* Take garbage photo
* Send requests for garbage collection.
* Update their their personal information.

The Administrator

The administrator can:

* Manage users
* Manage requests
* Manage garbage from different places