A

Mini Project Report

On

Sanitization Service Booking App

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

by

Jha Babita Nunu	**(117IT1083B)**
Jha Rohit Santosh	**(118IT3174A)**
Kale Pranali Tanaji	**(118IT3261B)**

Under the guidance of

Prof. Manivannan Panc

Department of Information Technology

Mahatma Gandhi Mission's College of Engineering & Technology

Kamothe, Navi Mumbai – 410 209

University of Mumbai

Academic Year: 2020-21

<u>**CERTIFICATE**</u>

This is to certify that the mini project entitled **“Sanitization Service Booking App”** is a bonafide work of **Jha Babita Nunu (117IT1083B), Jha Rohit Santosh (118IT3174A), Kale Pranali Tanaji(118IT13261A)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Undergraduate”** in **“Information Technology”.**

(Faculty In chagre)

Prof. Manivannan Panc

DECLARATION

We declare that this project report entitled **“Sanitization Service Booking App”** represents our ideas in our own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date:

Place:

(Signature)

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have got this all along the completion of our project work. Whatever we have done is only due to such guidance and assistance and we would not forget to thank them.

It is matter of great pleasure for us to submit the project report on **“Sanitization Service Booking App”**, as a part of our curriculum.

First and foremost, we would like to thank to our director **Dr. Geeta.S.Lathkar**, for giving us an opportunity to do the project work. We would like to thank our H.o.D Prof**. K.V. Raman** and subject incharge **Prof. Manivannan Panc**, for the valuable guidance and advice. He inspired us greatly to work in this project. His willingness to motivate us contributed tremendously to our project.

And last but not the least a special thanks goes to my team members, who helped me to assemble the information and gave suggestions to complete our project.

ABSTRACT

In 2020 maintaining social distancing and hygiene is one of the most important things. Even though maintaining personal hygiene can be easier for a person but maintaining the social standards of hygiene for our surrounding can be much more difficult so to keep our surrounding clean and sanitize and up to the standards of the social norm. we must seek help from the professionals. This Sanitization services booking app will help with that process. The existing system is very time consuming and tiresome. In this application the user will first book a date, after providing that instantly a price quote based on the size and nature of the space will be given, If the user accepts the quote the user can proceed to make the payment after which the process of disinfected and sanitization will be carried out on the fixed date agreed by the user. After the job is done the service personnel will notify the admin and after receiving said services the user can rate the service which will be a great feedback for the company.

Keywords: Sanitize, sanitization, Sanitization service, Sanitization booking

CHAPTER-1
INTRODUCTION

INTRODUCTION

Sanitization service booking application is a mobile phone application on which you can choose services according to your needs and your wants. You can rate the service according to your satisfaction

Currently this can be done through web applications or by going to the service provider which at times can be very time consuming. In the existing system the user first has to find for the vendors providing the specific service after that the user may have to create an account for that specific website after which the user can place the order and do the payment. So to make this process simpler and more convenient we have developed this application. In this app firstly the user needs to log into the application with its login credentials If the user is a existing user otherwise signup into the application. After login the list of services provided by the vendor are present on the home screen. The users can then review the list and then place their order accordingly then the vendor will send the professional service personnel to the address provided by the customer. After the job is done the service personnel will notify the admin and the customer can rate the service as par their satisfaction.

The application has three main roles User, service personnel and admin. First the user login into the application while providing all the required details. The user will have a list to service to choose from along with the price. After the user places the order the admin get a notification then the admin notifies the service personnel and also allocates the service personnel. The vendor login to the application and provide the details about the services and the service personnel

PURPOSE OF THE SYSTEM :

The purpose of developing sanitization service booking app is to create a centralized location where the customer can look for different services and can review the services which will help them in decision making and making it much faster and easier.

In this system there are three main roles the service personnel, the admin and the customer. when customer places the order, the admins get a notification and the admin assign a service personnel which the admin thinks seems perfect for the particular job. After the job is finish the service personnel will notify the admin and customer can rate the service

Objectives:

Making the process of finding and placing a order hustle free.

Eliminate paperwork and save time.

Automatic the entire process.

FEATURES OF THE SYSTEM :

User Friendly : The proposed system is user friendly because the retrieval and storing of data is fast and data is maintained efficiently. The user interface provided in the proposed system is very simple and elegant, which helps user to deal with the system in a very easy and efficient manner. User can place and cancel order with just one click.

Reliable: The system is very reliable.

Range of services: The system provides a range of service which includes disinfectant/sanitization of home, office, car, hospital, clinics, schools. Its also provides a subscription packages based on the potency of the chemical used for disinfectant

Very less paper work: The proposed system requires very less paper work. All the data is feted into the computer immediately and reports can be generated through computers. The work will become very easy because there is no need to keep data on papers. The employer will have

Computer operator control: Computer operator control will be there so no chance of errors. Moreover storing and retrieving of information is easy. So work can be done speedily and in time.

FEASIBILITY STUDY :

Economically Feasibility : Development of this application is highly economically feasible. The only thing to be done is making an environment with an effective supervision. The system being developed is economic with respect to the vendor's point of view. It is cost effective in the sense that has eliminated the paper work completely. The system is also time effective because the calculations are automated which are made at the end of the month or as per the user requirement. The result obtained contains minimum errors and are highly accurate as the data is required.

Technical feasibility : The technical requirement for the system is economic and it does not use any other additional Hardware and software. Technical evaluation must also assess whether the existing systems can be upgraded to use the new technology and whether the organization has the expertise to use it. Users just needs to download the app. The application is taken care by the administrator

Behavioral Feasibility : The system working is quite easy to use and learn due to its simple but attractive interface. User requires no special training for operating the system. Technical performance includes issues such as determining whether the system can provide the right information for the users about the personnel details, and whether the system can be organized so that it always delivers this information at the right place and on time using internet services. Acceptance revolves around the current system and its personnel.

SCOPE OF THE SYSTEM :

The scope of this project is limited to the activities of the operations unit of the users to placing order for home/office/workplace sanitization. The scope of the project is the system on which the software is installed, i.e. the project is developed as a mobile application, and it will work for a particular vendor But later on the project can be modified for adding any other services if the vendor has the intention of doing so. The application can also be modified to using artificial intelligence for measuring the area in terms of square feet through image processing.

CHAPTER 02
SYSTEM ANALYSIS

SYSTEM ANALYSIS

Analysis can be defined as breaking up of any whole so as to find out their nature, function etc. It defines design as to make preliminary sketches of; to sketch a pattern or outline for plan. To plan and carry out especially by artistic arrangement or in a skillful wall. System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and an intellectual orientation. The various tasks in the system analysis include the following :

- Understanding application.
- Planning.
- Scheduling.
- Developing candidate solution.
- Performing trade studies.
- Performing cost benefit analysis.
- Recommending alternative solutions.
- Selling of the system.
- Supervising, installing and maintaining the system.

This system manages the order made by the users and their records. The admin take care for the payment gateway and personnel allocation.

Existing System :

The Existing system is a either manual where the person calls the vendor or through various websites if the vendor is registered. Here the user has go through various websites or directories to find the contact info and the services charge. It will be a tedious job to maintain the record for the vendor. The human effort is more here. The retrieval of the information is not as easy as the records are maintained in the hand written registers. This application requires correct feed on input into the respective field. Suppose the wrong inputs are entered, the application resist to work. so the user find it difficult to use.

Working of Existing System:

In the present system all work is done through phone calls and emails. The users search for the sanitization, disinfectant and etc. companies on google then after going through various websites and reading reviews and cost analysis the users makes a decision. The users then place the order, provides it address and after that the professionals come to the given address and perform their task.

Disadvantages Of Existing System :

Not User Friendly : The existing system is not user friendly because the retrieval of data is very slow and data is not maintained efficiently.

Difficulty in report generating : We require more calculations to generate the report so it is generated at the end of the session.

Manual control : All calculations to generate report is done manually so there is greater chance of errors.

Lots of paper work : Existing system requires lot of paper work. Loss of even a single register/record led to difficult situation because all the papers are needed to generate the reports.

Time consuming : Every work is done manually so we cannot generate report in the middle of the session or as per the requirement because it is very time consuming.

Proposed System :

To overcome the drawbacks of the existing system, the proposed system has been evolved. The existing system has been modified to be implemented into the mobile environment which makes it much more accessible and convenient for the user

Advantages of Proposed System :

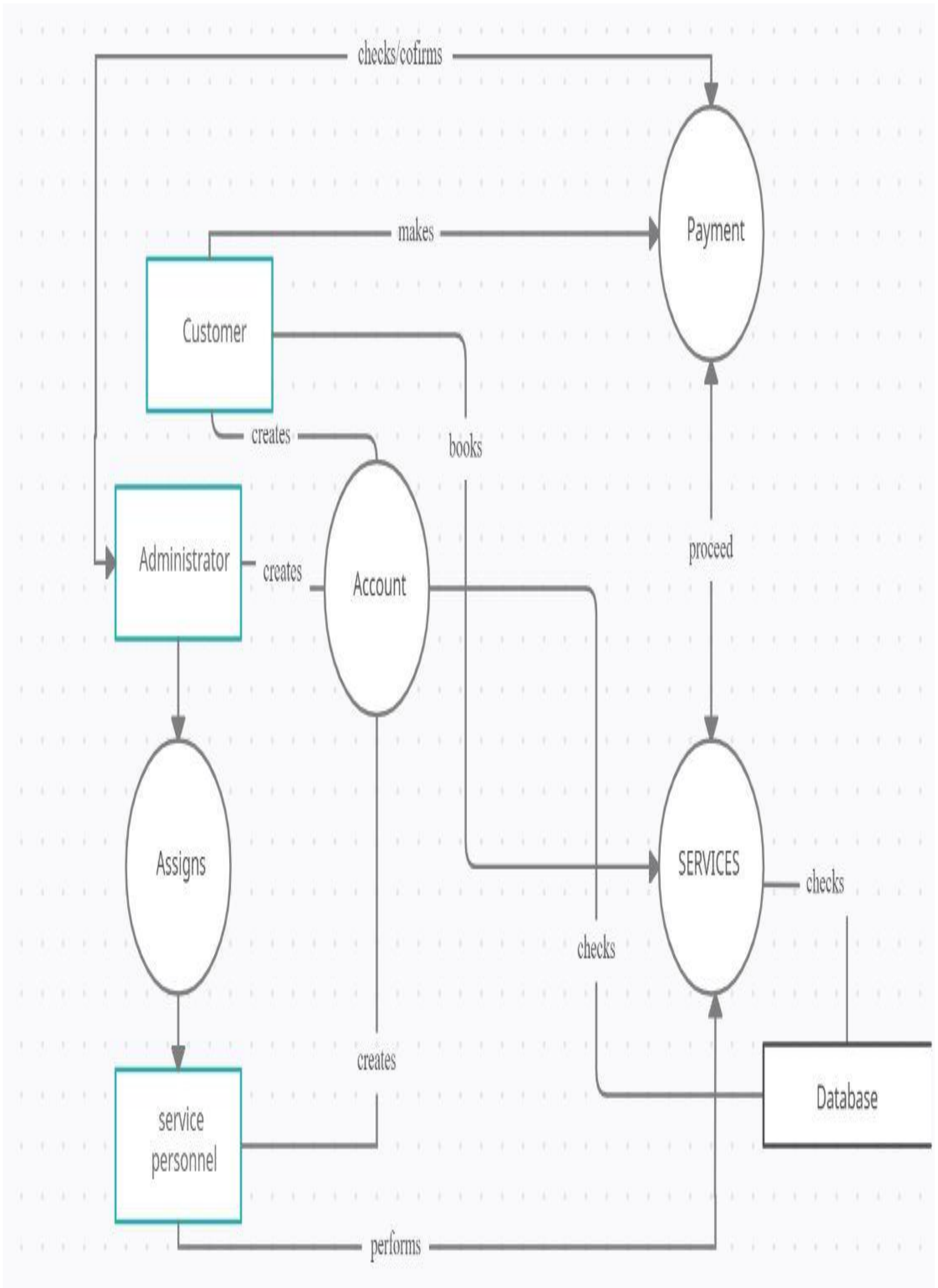
- ➢ It is trouble-free to use.
- ➢ It is a relatively fast approach for the user.
- ➢ It is highly reliable, approximate result from user
- ➢ Best user Interface.
- ➢ Efficient reports.

Overview :

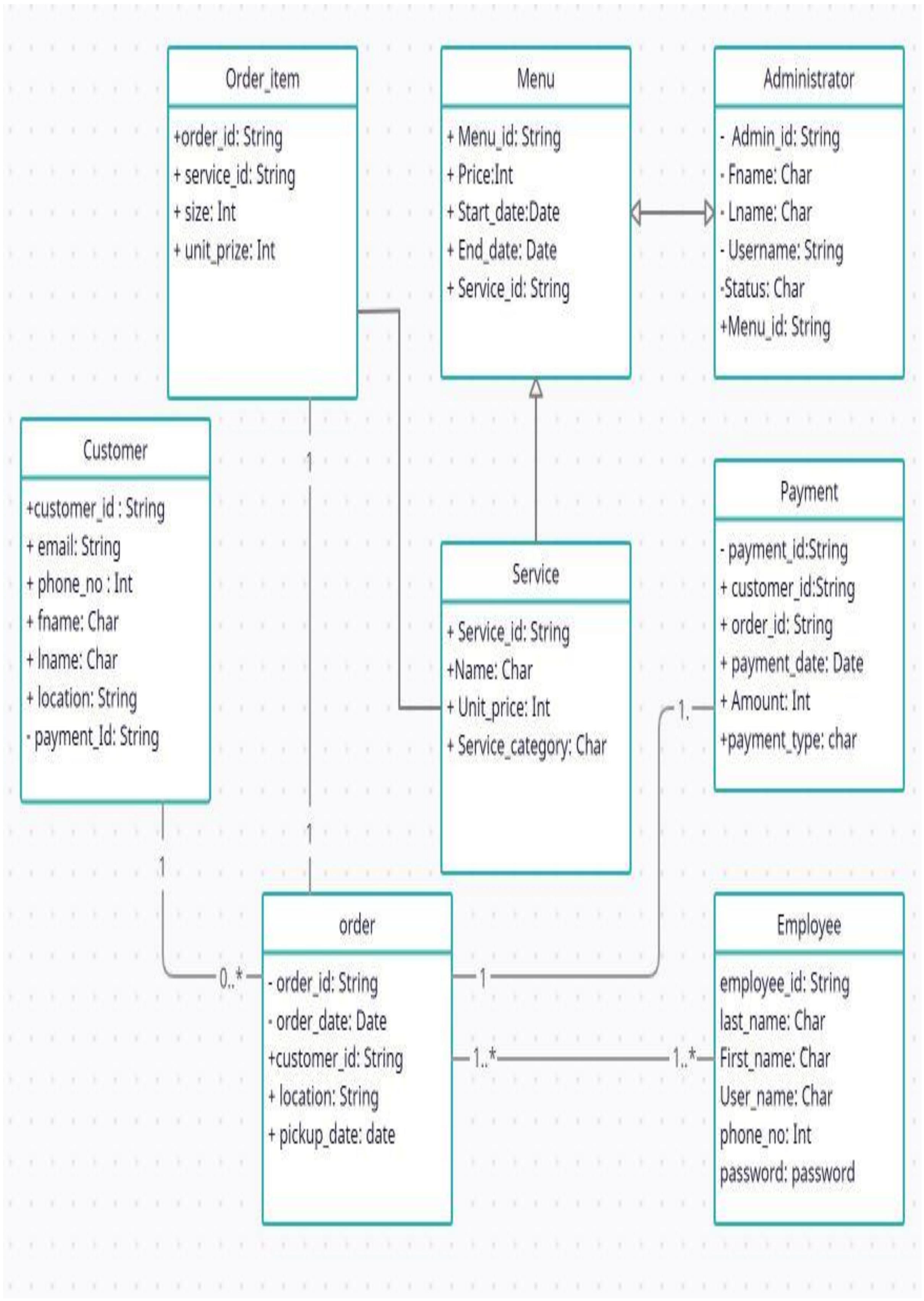
Sanitization service booking app has three main role Admin, user and Service personnel. The vendors will provide the details about the service personnel. The admin will maintain the data and also assign the personnel. The user can search and place their order and provide feedback:

- User will make a appointment and provide the details like Address , Size in square foot and nature of the place(home, office, school, hospital , car etc.).
- User can choose subscription options which are based on the potency of the disinfectant chemicals
- User will do the payment through their preferred medium.
- Admin will confirm the payment and accept the order
- Admin will notify the service personnel regarding the job along with the details
- Service personnel will notify the admin after the job is done
- User can give feedback

Data flow diagram



UML Diagram:


Order_item
+order_id: String
+ service_id: String
+ size: Int
+ unit_prize: Int
Menu
+ Menu_id: String
+ Price:Int
+ Start_date:Date
+ End_date: Date
+ Service_id: String
Administrator
- Admin_id: String
- Fname: Char
- Lname: Char
- Username: String
-Status: Char
+Menu_id: String
Customer
+customer_id : String
+ email: String
+ phone_no : Int
+ fname: Char
+ lname: Char
+ location: String
- payment_Id: String
Payment
- payment_id:String
+ customer_id:String
+ order_id: String
+ payment_date: Date
+ Amount: Int
+payment_type: char
Service
+ Service_id: String
+Name: Char
+ Unit_price: Int
+ Service_category: Char
order
- order_id: String
- order_date: Date
+customer_id: String
+ location: String
+ pickup_date: date
Employee
employee_id: String
last_name: Char
First_name: Char
User_name: Char
phone_no: Int
password: password
1
1
1
0..*
1
1.
1..*
1..*

CHAPTER 03

SYSTEM SPECIFICATION

SYSTEM SPECIFICATION

➢ HARDWARE REQUIREMENTS :

Minimum RAM : 2 GB

Hard Disk : 128 GB

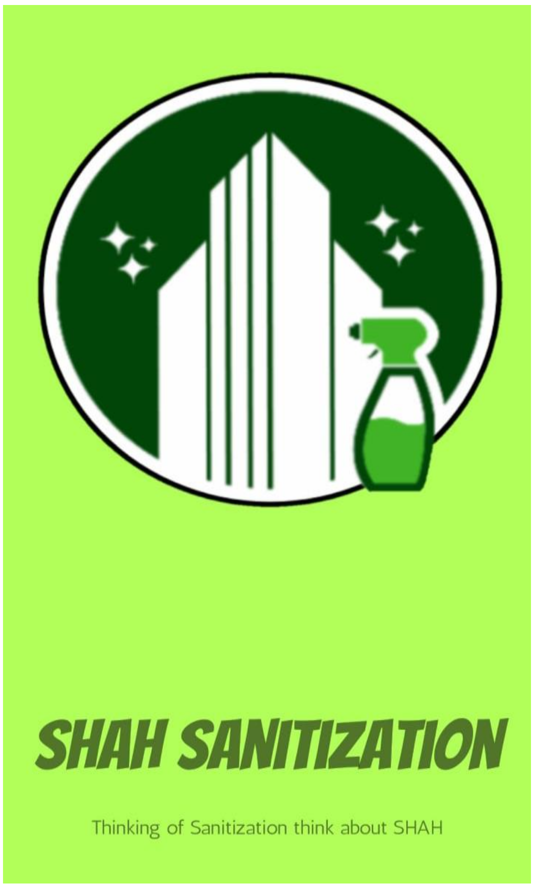
Processor : Microprocessor (2.40 GHz)
Android version : compileSdkVersion 30 buildToolsVersion "30.0.1

➢ SOFTWARE REQUIREMENTS :

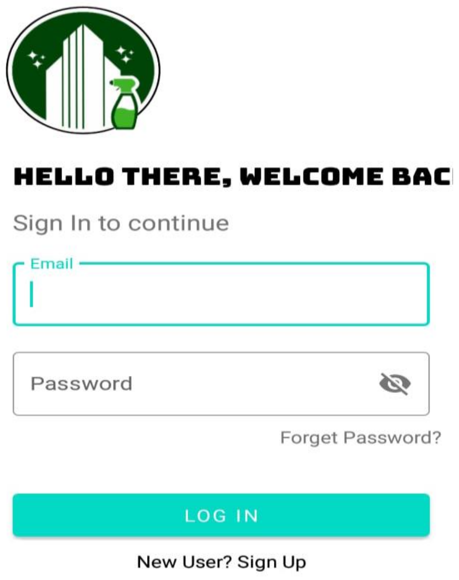
Operating system : Android
Programming language : Java, kotlin
Database : Firebase

CHAPTER.04
SNAPSHOT

Snap Shot:



Login:


HELLO THERE, WELCOME BAC

Sign In to continue

Email

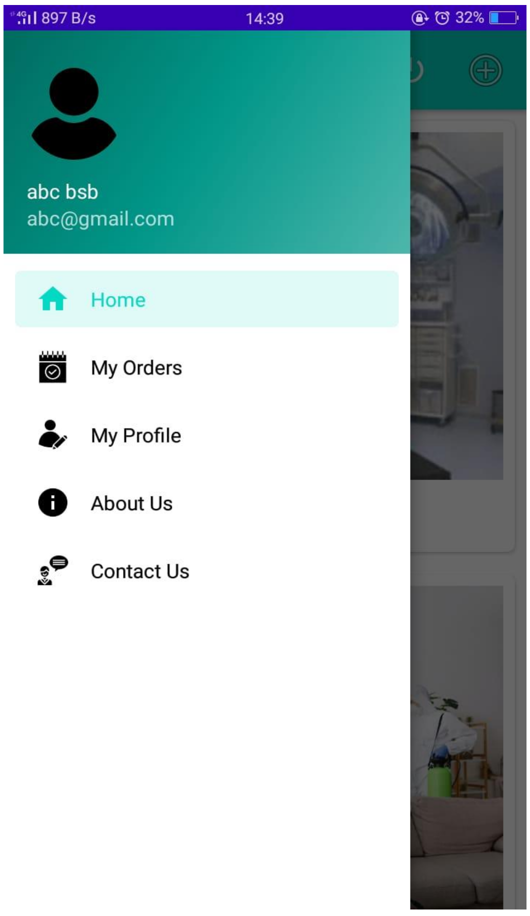
Password

Forget Password?

LOG IN

New User? Sign Up

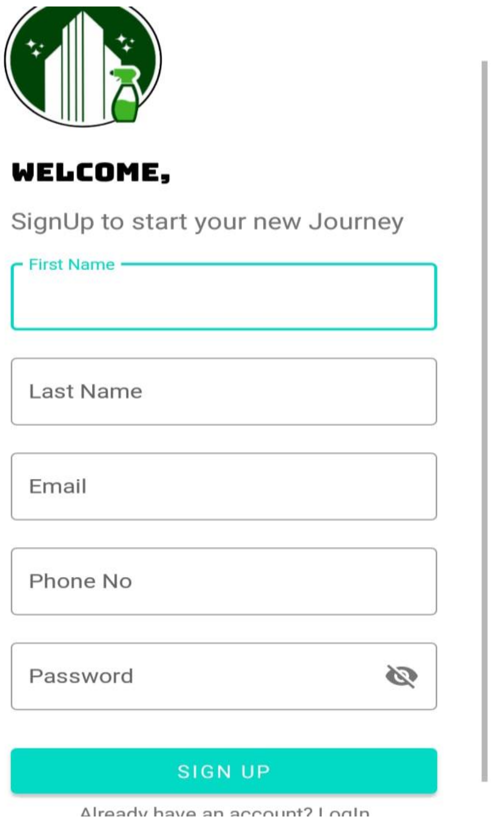
Home:



abc bsb
abc@gmail.com

Home
My Orders
My Profile
About Us
Contact Us

Signup:


WELCOME,

SignUp to start your new Journey

First Name

Last Name

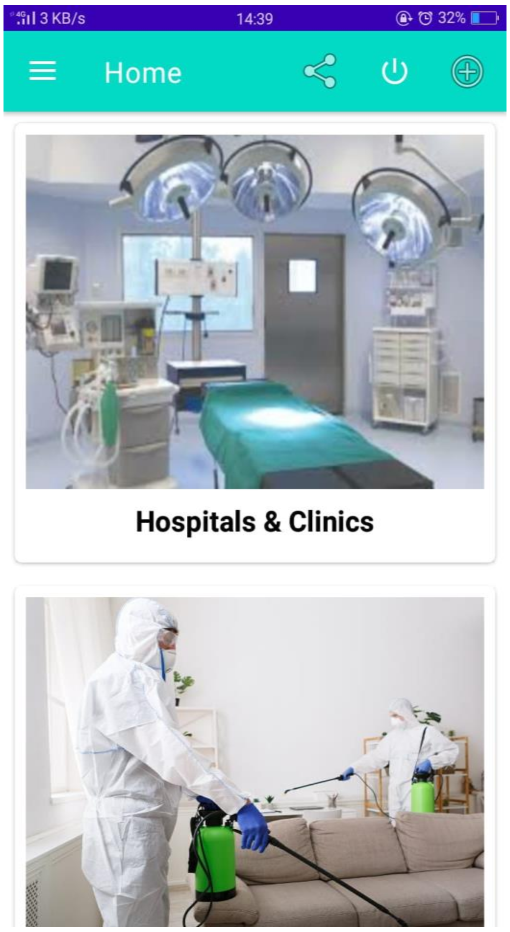
Email

Phone No

Password

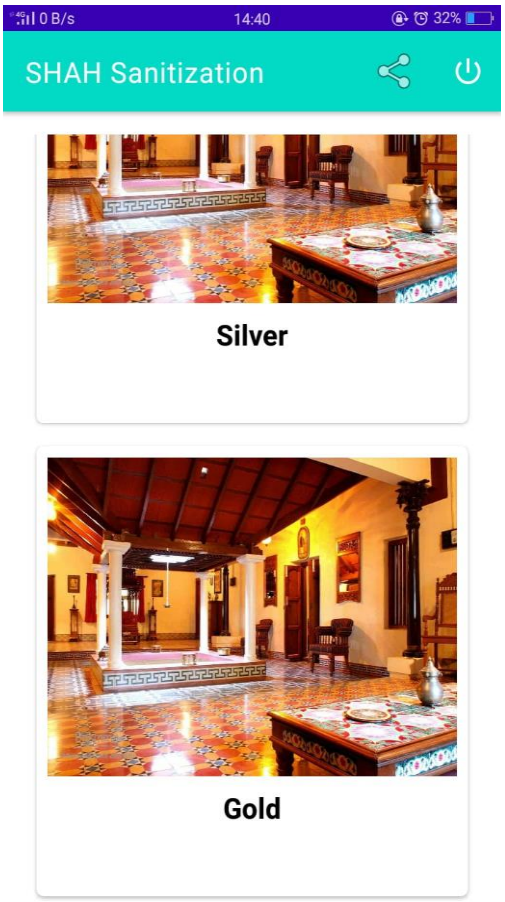
SIGN UP

Already have an account? LogIn


3 KB/s 14:39 32%

Home

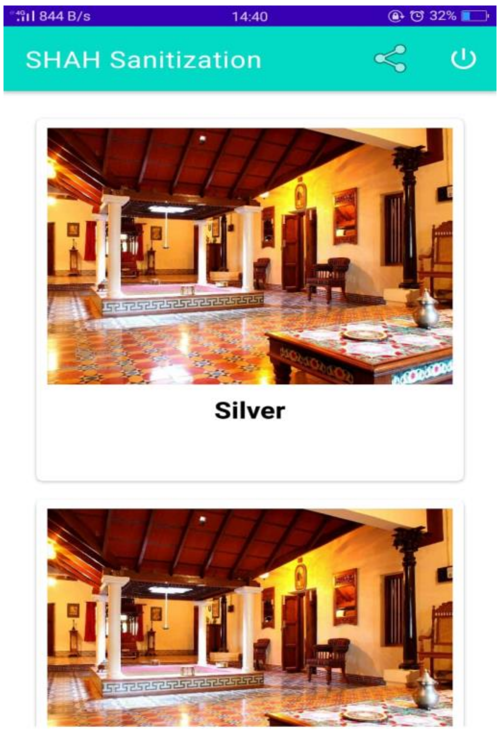
Hospitals & Clinics


0 B/s 14:40 32%

SHAH Sanitization

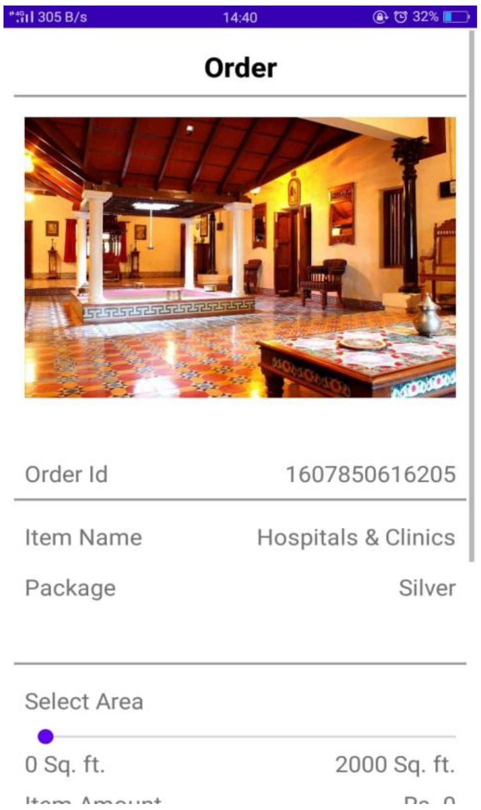
Silver

Gold


844 B/s 14:40 32%

SHAH Sanitization

Silver


305 B/s 14:40 32%

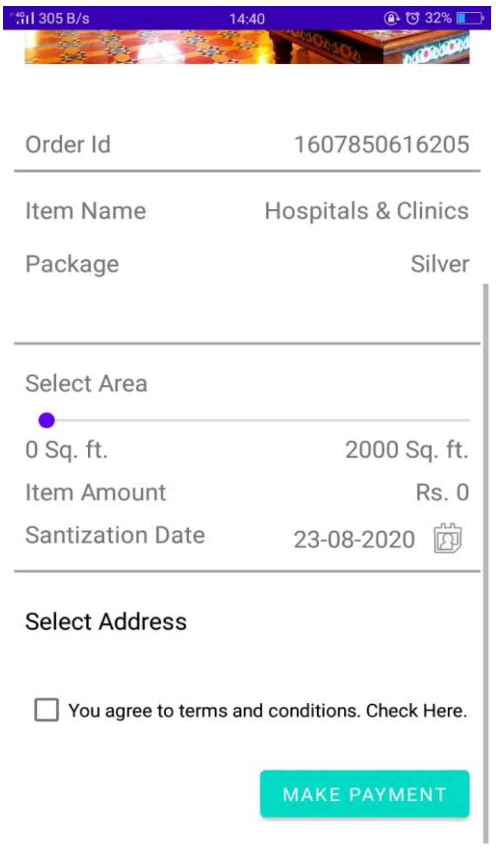
Order

Order Id	1607850616205
Item Name	Hospitals & Clinics
Package	Silver

Select Area

0 Sq. ft. 2000 Sq. ft.

Item Amount Rs. 0



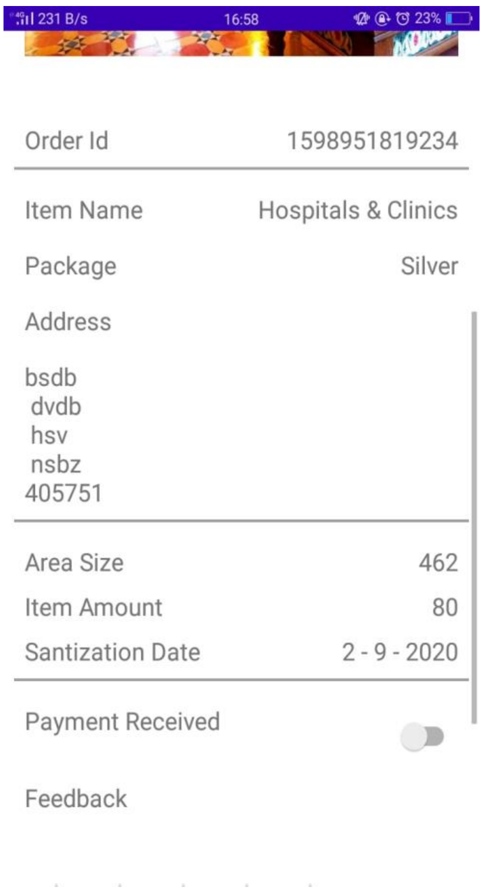
Order Id	1607850616205
Item Name	Hospitals & Clinics
Package	Silver

Select Area

0 Sq. ft.	2000 Sq. ft.
Item Amount	Rs. 0
Santization Date	23-08-2020

Select Address

☐ You agree to terms and conditions. Check Here.



Order Id	1598951819234
Item Name	Hospitals & Clinics
Package	Silver

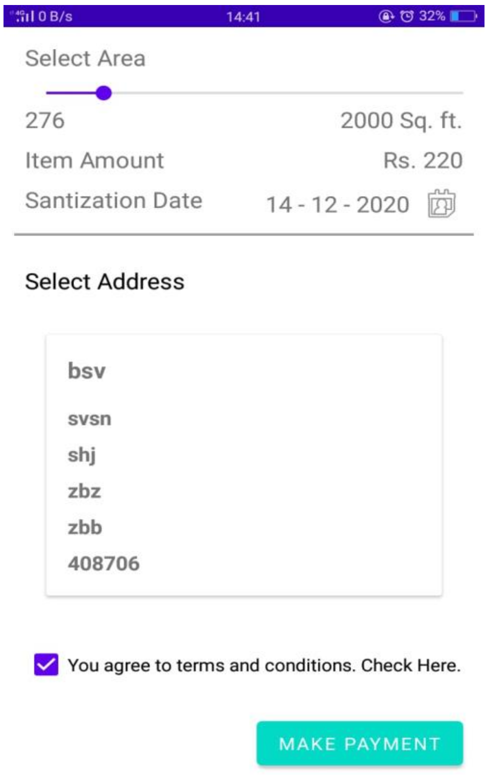
Address

bsdb
dvdb
hsv
nsbz
405751

Area Size	462
Item Amount	80
Santization Date	2 - 9 - 2020

Payment Received

Feedback



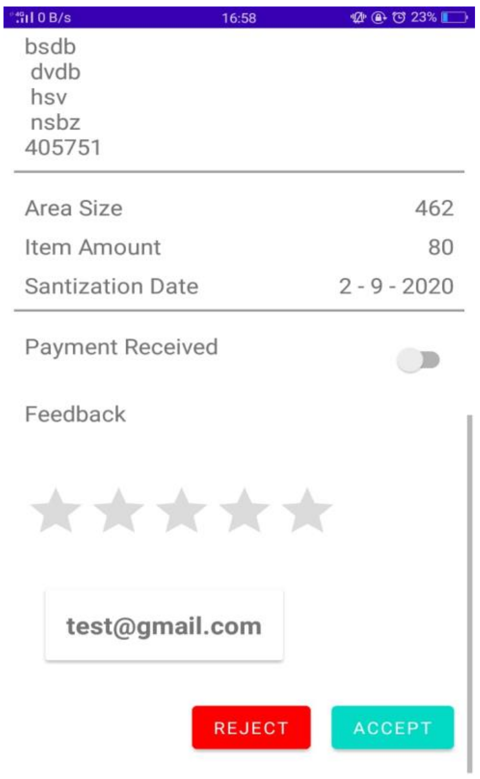
Select Area

276	2000 Sq. ft.
Item Amount	Rs. 220
Santization Date	14 - 12 - 2020

Select Address

bsv
svsn
shj
zbz
zbb
408706

☑ You agree to terms and conditions. Check Here.



bsdb
dvdb
hsv
nsbz
405751

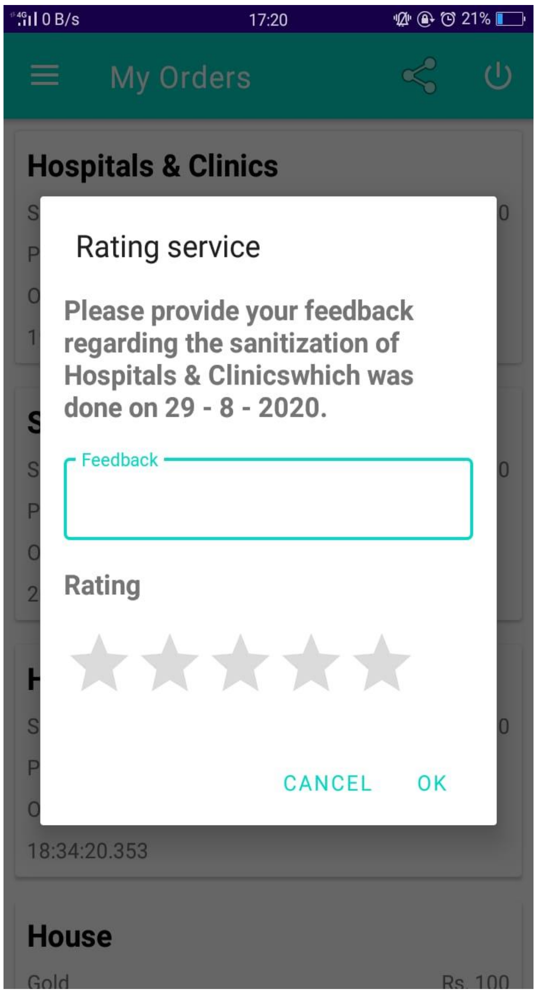
Area Size	462
Item Amount	80
Santization Date	2 - 9 - 2020

Payment Received

Feedback

☆☆☆☆☆

test@gmail.com



My Orders

Hospitals & Clinics

Rating service

Please provide your feedback regarding the sanitization of Hospitals & Clinicswhich was done on 29 - 8 - 2020.

Feedback:

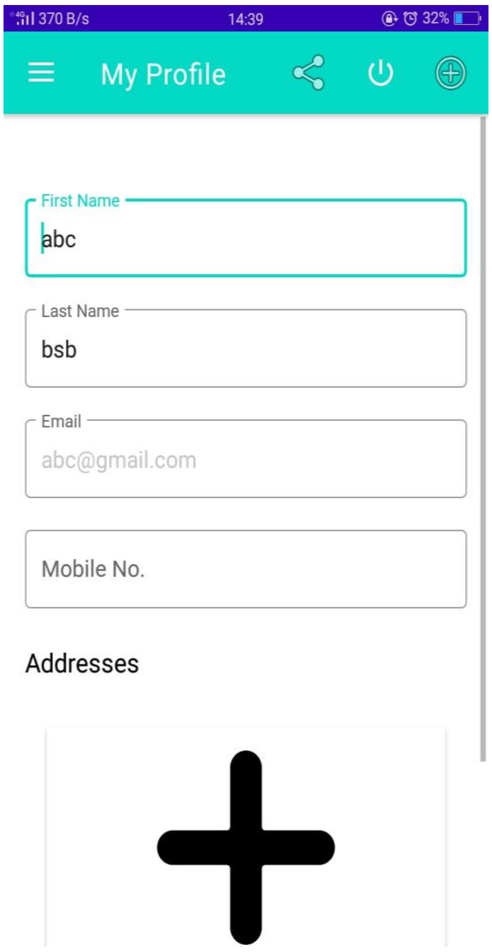
Rating ☆☆☆☆☆

CANCEL OK

18:34:20.353

House

Gold Rs. 100



My Profile

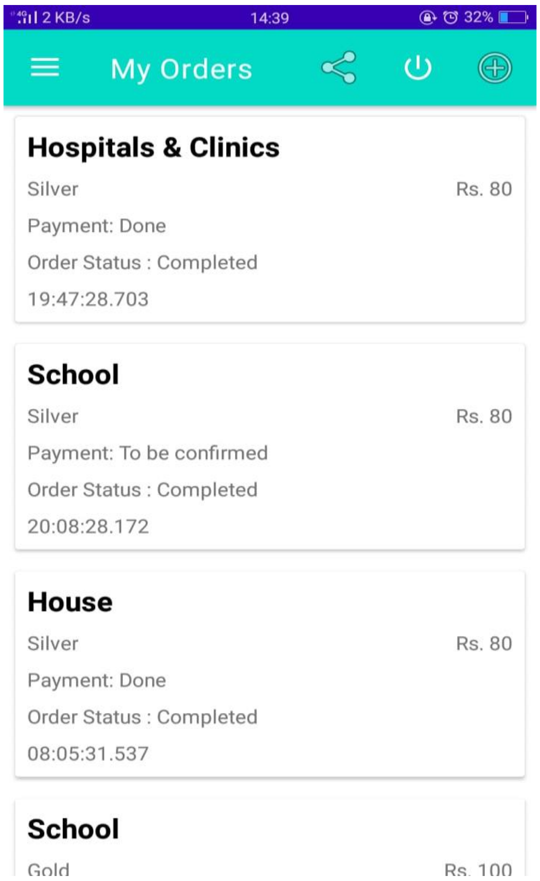
First Name: abc

Last Name: bsb

Email: abc@gmail.com

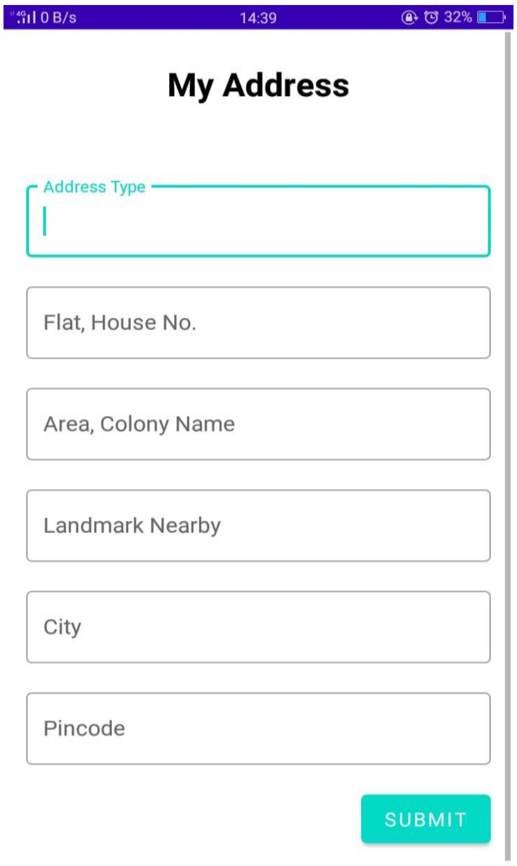
Mobile No.

Addresses



My Orders

Hospitals & Clinics	
Silver	Rs. 80
Payment: Done	
Order Status : Completed	
19:47:28.703	
School	
Silver	Rs. 80
Payment: To be confirmed	
Order Status : Completed	
20:08:28.172	
House	
Silver	Rs. 80
Payment: Done	
Order Status : Completed	
08:05:31.537	
School	
Gold	Rs. 100



My Address

Address Type:

Flat, House No.

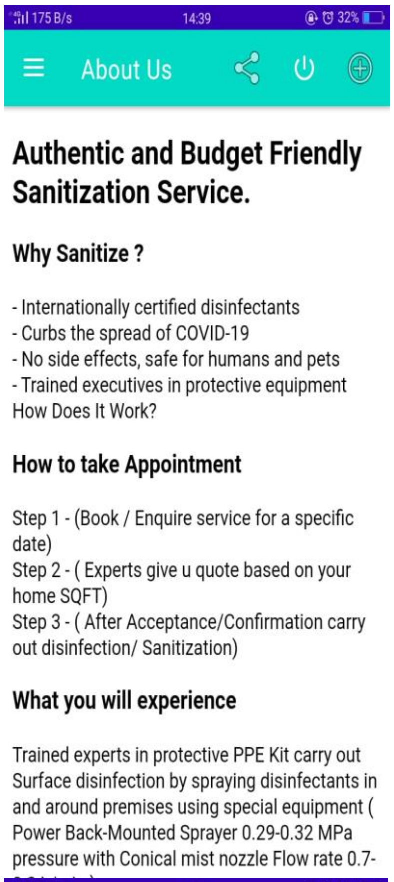
Area, Colony Name

Landmark Nearby

City

Pincode

SUBMIT



Authentic and Budget Friendly Sanitization Service.

Why Sanitize ?

- Internationally certified disinfectants
- Curbs the spread of COVID-19
- No side effects, safe for humans and pets
- Trained executives in protective equipment
How Does It Work?

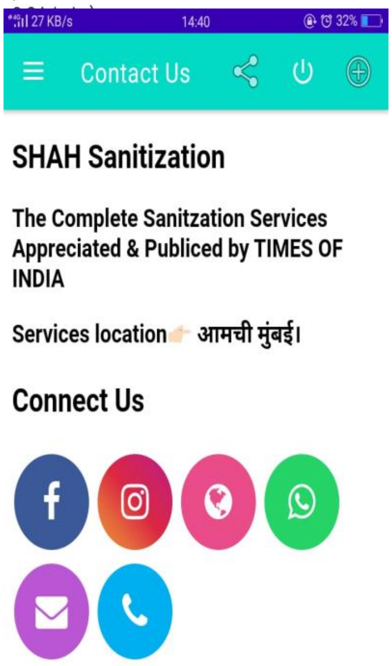
How to take Appointment

Step 1 - (Book / Enquire service for a specific date)
Step 2 - (Experts give u quote based on your home SQFT)
Step 3 - (After Acceptance/Confirmation carry out disinfection/ Sanitization)

What you will experience

Trained experts in protective PPE Kit carry out Surface disinfection by spraying disinfectants in and around premises using special equipment (Power Back-Mounted Sprayer 0.29-0.32 MPa pressure with Conical mist nozzle Flow rate 0.7-

-



SHAH Sanitization

The Complete Sanitzation Services Appreciated & Publiced by TIMES OF INDIA

Services location👉 आमची मुंबई।

Connect Us

CHAPTER 05
PROGRAM CODE

Program Code

```
package in.application.shahsanitization.utils;

import android.content.Context;
import android.content.SharedPreferences;
import android.net.Uri;
import android.util.Log;

import androidx.annotation.NonNull;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
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 com.google.firebase.messaging.FirebaseMessaging;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

import in.application.shahsanitization.R;
import in.application.shahsanitization.model.AddressModel;
import in.application.shahsanitization.model.ItemCategories;
import in.application.shahsanitization.model.ItemCategoriesFb;
import in.application.shahsanitization.model.ItemSubCategories;
import in.application.shahsanitization.model.OrderDataModel;
import in.application.shahsanitization.model.UserAccountDataModel;
import in.application.shahsanitization.model.UserDataModel;

public class HelperMethods {
   private static String PREF_NAME = "shahsanitization";
   private static int PREF_MODE = 0;
   private Context context;
   private FirebaseDatabase mDatabase;
   private FirebaseAuth mAuth;
   private DatabaseReference mUserRef;
   private DatabaseReference mItemRef;
   private DatabaseReference mOrderRef;
   private SharedPreferences mPref;

   private DatabaseReference mAccountCheckRef;

   public HelperMethods(Context mCnx) {
      context = mCnx;
      mAuth = FirebaseAuth.getInstance();
      mDatabase = FirebaseDatabase.getInstance();
      mUserRef = mDatabase.getReference().child("user_data");
      mItemRef = mDatabase.getReference().child("item_list");
      mOrderRef = mDatabase.getReference().child("item_orders");
      mAccountCheckRef =
mDatabase.getReference().child("user_type_data");
      mPref = context.getSharedPreferences(PREF_NAME,
PREF_MODE);
   }

   private void updateUserData(final UserDataModel dataModel, final
FirebaseResponseHandler mResponse, final UserAccountDataModel
accountDataModel) {
//       dataModel.setFbKey(mUserRef.push().getKey());

mUserRef.child(dataModel.getFbKey()).setValue(dataModel).addOn
SuccessListener(new OnSuccessListener<Void>() {
         @Override
         public void onSuccess(Void aVoid) {
                updateSharedPrefString("user_key",
dataModel.getFbKey());
                updateSharedPrefLong("user_id", dataModel.getId());
                updateSharedPrefLong("user_created_on",
dataModel.getCreationTime());
                updateSharedPrefString("user_first_name",
dataModel.getFirstName());
                updateSharedPrefString("user_phone_number",
dataModel.getNumber());
                updateSharedPrefString("user_last_name",
dataModel.getLastName());
                updateSharedPrefString("user_email",
dataModel.getEmail());
                updateSharedPrefString("user_type",
dataModel.getUserType());
                if (dataModel.getUserType().equals("customer")) {
                    String key = mAccountCheckRef.push().getKey();
                    UserAccountDataModel model = new
UserAccountDataModel(dataModel.getEmail(),
dataModel.getFbKey(), "customer", true, key);
                    updateAccountType(model, mResponse);
                } else {
                    accountDataModel.setUserKey(dataModel.getFbKey());
                    updateAccountType(accountDataModel, mResponse);
                }
//                checkUserType(dataModel, mResponse);
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                mResponse.onFailure(e.getLocalizedMessage());
            }
        });
    }

    public void checkUserTypeNew(final UserDataModel
userDataModel, final FirebaseResponseHandler mResponse) {
        userDataModel.setFbKey(mUserRef.push().getKey());
        mAccountCheckRef.addListenerForSingleValueEvent(new
ValueEventListener() {
```

```
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                int status = -1;
                String type = "";
                UserAccountDataModel dataModel = null;
                for (DataSnapshot data : snapshot.getChildren()) {
                    dataModel =
data.getValue(UserAccountDataModel.class);
                    if
(dataModel.getEmail().equals(userDataModel.getEmail())) {
                        if (dataModel.getActivated()) {
                            status = 1;
                            type = dataModel.getType();
                        } else {
                            status = 0;
                        }
                    }
                }
                if (status == -1) {

FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_key")).addOnSuccessListener(new
OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.e("TAG", "Subscribed");
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {

                            Log.e("TAG", e.getLocalizedMessage());
                        }
                    });
                    userDataModel.setUserType("customer");
                    updateUserData(userDataModel, mResponse,
dataModel);
//                    updateAccountType(userDataModel.getEmail(),
userDataModel.getFbKey(), "customer", mResponse);
```

```
            } else if (status == 1) {
                if (type.equals("admin")) {

FirebaseMessaging.getInstance().subscribeToTopic(type).addOnSucc
essListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.e("TAG", "Subscribed");
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {

                            Log.e("TAG", e.getLocalizedMessage());
                        }
                    });
                    userDataModel.setUserType("admin");
                } else {

FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_key")).addOnSuccessListener(new
OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Log.e("TAG", "Subscribed");
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {

                            Log.e("TAG", e.getLocalizedMessage());
                        }
                    });
                    userDataModel.setUserType("service");
                }
                updateUserData(userDataModel, mResponse,
dataModel);
            } else {
                mResponse.onFailure("clean_logout");
            }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }


//    public void checkUserType(final UserDataModel userDataModel,
final FirebaseResponseHandler mResponse) {
//        mAccountCheckRef.addListenerForSingleValueEvent(new
ValueEventListener() {
//            @Override
//            public void onDataChange(@NonNull DataSnapshot
snapshot) {
//                int status = -1;
//                String type = "";
//                for (DataSnapshot data : snapshot.getChildren()) {
//                    UserAccountDataModel dataModel =
data.getValue(UserAccountDataModel.class);
//                    if
(dataModel.getEmail().equals(userDataModel.getEmail())) {
//                        if (dataModel.getActivated()) {
//                            status = 1;
//                            type = dataModel.getType();
//                        } else {
//                            status = 0;
//                        }
//                    }
//                }
//                if (status == -1) {
//
FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_key")).addOnSuccessListener(new
OnSuccessListener<Void>() {
//                        @Override
//                        public void onSuccess(Void aVoid) {
//                            Log.e("TAG", "Subscribed");
//                    }
//                }).addOnFailureListener(new OnFailureListener() {
//                    @Override
//                    public void onFailure(@NonNull Exception e) {
//
//                        Log.e("TAG", e.getLocalizedMessage());
//                    }
//                });
//                updateAccountType(userDataModel.getEmail(),
userDataModel.getFbKey(), "customer", mResponse);
//            } else if (status == 1) {
//                if (type.equals("admin")) {
//
FirebaseMessaging.getInstance().subscribeToTopic(type).addOnSucc
essListener(new OnSuccessListener<Void>() {
//                        @Override
//                        public void onSuccess(Void aVoid) {
//                            Log.e("TAG", "Subscribed");
//                        }
//                    }).addOnFailureListener(new OnFailureListener() {
//                        @Override
//                        public void onFailure(@NonNull Exception e) {
//
//                            Log.e("TAG", e.getLocalizedMessage());
//                        }
//                    });
//                } else {
//
FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_key")).addOnSuccessListener(new
OnSuccessListener<Void>() {
//                        @Override
//                        public void onSuccess(Void aVoid) {
//                            Log.e("TAG", "Subscribed");
//                        }
//                    }).addOnFailureListener(new OnFailureListener() {
//                        @Override
//                        public void onFailure(@NonNull Exception e) {
//
//                            Log.e("TAG", e.getLocalizedMessage());
//                          }
//                      });
//                  }
//                  updateSharedPrefString("user_type", type);
//                  mResponse.onSuccess(true, "User Data Updated");
//              } else {
//                  mResponse.onFailure("clean_logout");
//              }
//          }
//
//          @Override
//          public void onCancelled(@NonNull DatabaseError error) {
//
//          }
//      });
//  }


    public void getUserData(final String email, final
FirebaseResponseHandler mResponse) {

        mUserRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                UserDataModel dataModel = null;
                for (DataSnapshot userData : snapshot.getChildren()) {
                    if (userData.getValue(UserDataModel.class) != null) {
                        if
(userData.getValue(UserDataModel.class).getEmail().equals(email))
{
                            Log.e("TAG", email);

                            dataModel =
userData.getValue(UserDataModel.class);
                            Log.e("TAG 1", dataModel.getEmail());
                            updateSharedPrefString("user_type",
dataModel.getUserType());
```

```
                        updateSharedPrefString("user_key",
dataModel.getFbKey());
                        updateSharedPrefString("user_first_name",
dataModel.getFirstName());
                        updateSharedPrefString("user_last_name",
dataModel.getLastName());
                        updateSharedPrefString("user_email",
dataModel.getEmail());
                        updateSharedPrefLong("user_id",
dataModel.getId());
                        updateSharedPrefLong("user_created_on",
dataModel.getCreationTime());

                    }
                }
            }
            if (dataModel.getUserType().equals("customer") ||
dataModel.getUserType().equals("service")) {

FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_key")).addOnSuccessListener(new
OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Log.e("TAG", "Subscribed");
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {

                        Log.e("TAG", e.getLocalizedMessage());
                    }
                });
            } else {

FirebaseMessaging.getInstance().subscribeToTopic(getSharedPrefStri
ng("user_type")).addOnSuccessListener(new
OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Log.e("TAG", "Subscribed");
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {

                        Log.e("TAG", e.getLocalizedMessage());
                    }
                });
            }
            mResponse.onSuccess(true, "Logged In Successfully");
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            mResponse.onFailure(error.getMessage());
        }
    });
}

   private void updateAccountType(UserAccountDataModel model,
final FirebaseResponseHandler mResponse) {
//       String key = mAccountCheckRef.push().getKey();
//       UserAccountDataModel model = new
UserAccountDataModel(email, userKey, type, true, key);

mAccountCheckRef.child(model.getKey()).setValue(model).addOnS
uccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
//             updateSharedPrefString("user_type", type);

            mResponse.onSuccess(true, "User Data Updated");
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            mResponse.onFailure(e.getLocalizedMessage());
        }
    });
    }

    public void uploadToDb(FirebaseResponseHandler mHandler) {
        final Boolean isThereAnyIssue;
        final String msg = "";
        ArrayList<ItemCategoriesFb> mFirebaseData = new
ArrayList<>();
        ArrayList<ItemSubCategories> mSubItems = new
ArrayList<>();
        String key = mItemRef.push().getKey();
        mSubItems.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Silver", "", 80, key));
        mSubItems.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "Gold",
"", 100, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(),
"Hospitals & Clinics", "", Calendar.getInstance().getTimeInMillis(),
mSubItems, 0, key));

        ArrayList<ItemSubCategories> mSubItems1 = new
ArrayList<>();
        key = mItemRef.push().getKey();
        mSubItems1.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Silver", "", 80, key));
        mSubItems1.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "Gold",
"", 100, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(), "House",
"", Calendar.getInstance().getTimeInMillis(), mSubItems1, 0, key));

        ArrayList<ItemSubCategories> mSubItems2 = new
ArrayList<>();
        key = mItemRef.push().getKey();
        mSubItems2.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Silver", "", 80, key));
```

```
        mSubItems2.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "Gold",
"", 100, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(),
"School", "", Calendar.getInstance().getTimeInMillis(), mSubItems2,
0, key));

        ArrayList<ItemSubCategories> mSubItems3 = new
ArrayList<>();

        key = mItemRef.push().getKey();
        mSubItems3.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Silver", "", 80, key));
        mSubItems3.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "Gold",
"", 100, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(), "Shops",
"", Calendar.getInstance().getTimeInMillis(), mSubItems3, 0, key));

        ArrayList<ItemSubCategories> mSubItems4 = new
ArrayList<>();

        key = mItemRef.push().getKey();
        mSubItems4.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Silver", "", 80, key));
        mSubItems4.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "Gold",
"", 100, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(), "Office",
"", Calendar.getInstance().getTimeInMillis(), mSubItems4, 0, key));

        ArrayList<ItemSubCategories> mSubItems5 = new
ArrayList<>();
        key = mItemRef.push().getKey();
        mSubItems5.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "2/3
Wheeler Vehicles", "", 40, key));
        mSubItems5.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(), "4
Wheeler Vehicles", "", 70, key));
        mSubItems5.add(new
ItemSubCategories(Calendar.getInstance().getTimeInMillis(),
"Truck/Bus", "", 80, key));
        mFirebaseData.add(new
ItemCategoriesFb(Calendar.getInstance().getTimeInMillis(),
"Vehicles", "", Calendar.getInstance().getTimeInMillis(),
mSubItems5, 0, key));

        for (ItemCategoriesFb mFbData : mFirebaseData) {

mItemRef.child(mFbData.getFbKey()).setValue(mFbData).addOnSuc
cessListener(new OnSuccessListener<Void>() {

                @Override
                public void onSuccess(Void aVoid) {
//                    msg = "Data Added Successfully";
//                    isThereAnyIssue = false;
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
//                    isThereAnyIssue = true;
//                    msg = e.getLocalizedMessage();
                }
            });
        }
//        if (isThereAnyIssue) {
//            mHandler.onFailure(msg);
//        } else {
        mHandler.onSuccess(true, msg);
//        }
    }
    public void getItemsFromDb(final FirebaseDataFetchResponse
mHandler) {
        final ArrayList<ItemCategories> mItems = new ArrayList<>();
        mItemRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                for (DataSnapshot itemData : snapshot.getChildren()) {
                    ItemCategoriesFb data =
itemData.getValue(ItemCategoriesFb.class);
                    if (data.getName().equals("Hospitals & Clinics")) {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.hospital, data.getDescription(),
data.getFbKey()));
                    }
                    else if (data.getName().equals("House")) {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.house2, data.getDescription(),
data.getFbKey()));
                    } else if (data.getName().equals("School")) {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.shops, data.getDescription(),
data.getFbKey()));
                    } else if (data.getName().equals("Shops")) {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.shops, data.getDescription(),
data.getFbKey()));
                    } else if (data.getName().equals("Office")) {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.office, data.getDescription(),
data.getFbKey()));
                    } else {
                        mItems.add(new ItemCategories(data.getId(),
data.getName(), R.mipmap.vehicle2, data.getDescription(),
data.getFbKey()));
                    }
                }
                mHandler.onSuccess(true, mItems);
            }
            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                mHandler.onFailure(error.getMessage());
            }
        });

    }


    public void getSubItemsFromDb(final FirebaseDataFetchResponse
mHandler, String key) {
        final ArrayList<ItemCategories> mItems = new ArrayList<>();
        mItemRef.child(key).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                ItemCategoriesFb data =
snapshot.getValue(ItemCategoriesFb.class);
                mHandler.onSuccessItems(data);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                mHandler.onFailure(error.getMessage());
            }
        });

    }

    public void updateSharedPrefString(String key, String value) {
        mPref.edit().putString(key, value).apply();
    }

    public void updateSharedPrefLong(String key, Long value) {
        mPref.edit().putLong(key, value).apply();
    }

    public void clearSharedPref() {
        mPref.edit().clear().apply();
    }

    public String getSharedPrefString(String key) {
        return mPref.getString(key, "");
    }

    public Long getSharedPrefLong(String key) {
        return mPref.getLong(key, 0L);
    }

    public void placeOrder(Long id, String name, String packageName,
Integer price, String serviceDate, int area, String address, int status,
final FirebaseResponseHandler mHandler) {
        String key = mOrderRef.push().getKey();
        boolean paymentDone = false;
        boolean isCorona = false;
        if (status == -1) {
            isCorona = true;
        }
        OrderDataModel dataModel = new OrderDataModel(id, name,
packageName, getSharedPrefString("user_first_name") + " " +
getSharedPrefString("user_last_name"),
getSharedPrefString("user_key"), address, paymentDone, 0, "", "", "",
"", id, serviceDate, price, area, key, isCorona, 0.0f);

mOrderRef.child(key).setValue(dataModel).addOnSuccessListener(n
ew OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                try {
                    sendOrderPlacedNotification("admin");
                } catch (Exception e) {
                    Log.e("TAG", e.getLocalizedMessage());
                }
                mHandler.onSuccess(true, "Order Placed");
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                mHandler.onFailure(e.getLocalizedMessage());
            }
        });
    }

    public void getAllOrders(final FirebaseDataFetchResponse
mResponse) {
        mOrderRef.addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {
                ArrayList<OrderDataModel> orders = new ArrayList<>();
                for (DataSnapshot itemData : snapshot.getChildren()) {
                    if (getSharedPrefString("user_type").equals("admin")) {

orders.add(itemData.getValue(OrderDataModel.class));
                    } else {
                        Log.e("Orders",
itemData.getValue(OrderDataModel.class).getServiceBoyKey());
                        Log.e("Orders",
itemData.getValue(OrderDataModel.class).getStatus() + "");
                        Log.e("Keys", getSharedPrefString("user_key"));
                        if
(itemData.getValue(OrderDataModel.class).getServiceBoyKey().equa
ls(getSharedPrefString("user_key"))) {
                            if
(itemData.getValue(OrderDataModel.class).getStatus() == 1) {

orders.add(itemData.getValue(OrderDataModel.class));
                            }
                        }
                    }
                }
                Log.e("TAG", orders.size() + "");
                mResponse.onSuccessOrders(orders);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
```

```
            mResponse.onFailure(error.getMessage());
        }
    });
}

public void getUserOrder(final FirebaseDataFetchResponse
mResponse) {
    final String userKey = getSharedPrefString("user_key");
    mOrderRef.addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            ArrayList<OrderDataModel> orders = new ArrayList<>();
            for (DataSnapshot itemData : snapshot.getChildren()) {
                OrderDataModel dataModel =
itemData.getValue(OrderDataModel.class);
                if (dataModel.getUserKey().equals(userKey)) {
                    orders.add(dataModel);
                }
            }
            mResponse.onSuccessOrders(orders);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            mResponse.onFailure(error.getMessage());
        }
    });
}

public void getSpecificOrder(final FirebaseDataFetchResponse
mResponse, String key) {

    mOrderRef.child(key).addListenerForSingleValueEvent(new
ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot
snapshot) {
            ArrayList<OrderDataModel> orders = new ArrayList<>();
```

```
                OrderDataModel dataModel =
snapshot.getValue(OrderDataModel.class);
                orders.add(dataModel);
                mResponse.onSuccessOrders(orders);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                mResponse.onFailure(error.getMessage());
            }
        });
    }

    public void getSpecificUserDetails(final
FirebaseDataFetchResponse mHandler, String key) {
        mUserRef.child(key).addListenerForSingleValueEvent(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
snapshot) {

mHandler.onSuccessUserData(snapshot.getValue(UserDataModel.cla
ss));
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                mHandler.onFailure(error.getMessage());
            }
        });
    }

    public void updateOrder(final OrderDataModel model, String key,
final FirebaseResponseHandler mResponse, final boolean
sentNotification) {


mOrderRef.child(key).setValue(model).addOnSuccessListener(new
OnSuccessListener<Void>() {
            @Override
```

```
          public void onSuccess(Void aVoid) {
             if (sentNotification) {
                try {
                   if (model.getStatus() == 1) {
                      sendOrderAcceptNotification(model.getUserKey());

sendServiceNotification(model.getServiceBoyKey());
                   } else if (model.getStatus() == 2) {

sendOrderCompletedNotification(model.getUserKey());
                   } else if (model.getStatus() == -1) {
                      sendRejectedNotification(model.getUserKey());
                   }
                } catch (JSONException e) {
                   e.printStackTrace();
                }
             }
             mResponse.onSuccess(true, "Order Updated");
          }
       }).addOnFailureListener(new OnFailureListener() {
          @Override
          public void onFailure(@NonNull Exception e) {
             mResponse.onFailure(e.getLocalizedMessage());
          }
       });
    }

    public void saveAddress(final FirebaseResponseHandler
mResponse, final AddressModel mAddress) {
       mAddress.setFbKey(mUserRef.push().getKey());

mUserRef.child(getSharedPrefString("user_key")).addListenerForSin
gleValueEvent(new ValueEventListener() {
          @Override
          public void onDataChange(@NonNull DataSnapshot
snapshot) {
             UserDataModel userDataModel =
snapshot.getValue(UserDataModel.class);
             userDataModel.getAddress().add(mAddress);
```

```
            updateUserData(mResponse, userDataModel, "Address
Added");
         }

         @Override
         public void onCancelled(@NonNull DatabaseError error) {
            mResponse.onFailure(error.getMessage());
         }
      });
   }


   public void getAddress(final FirebaseDataFetchResponse
mResponse) {

mUserRef.child(getSharedPrefString("user_key")).addListenerForSin
gleValueEvent(new ValueEventListener() {
         @Override
         public void onDataChange(@NonNull DataSnapshot
snapshot) {
            UserDataModel userDataModel =
snapshot.getValue(UserDataModel.class);
            mResponse.onSuccessUserData(userDataModel);
         }

         @Override
         public void onCancelled(@NonNull DatabaseError error) {
            mResponse.onFailure(error.getMessage());
         }
      });
   }

   public void getServiceData(final FirebaseDataFetchResponse
mResponse) {
      mAccountCheckRef.addListenerForSingleValueEvent(new
ValueEventListener() {
         @Override
         public void onDataChange(@NonNull DataSnapshot
snapshot) {
```

```
                ArrayList<UserAccountDataModel> mModels = new
ArrayList<>();
                for (DataSnapshot data : snapshot.getChildren()) {
                    UserAccountDataModel dataModel =
data.getValue(UserAccountDataModel.class);
                    if (dataModel.getType().equals("service")) {
                        mModels.add(dataModel);
                    }
                }
                mResponse.onSuccessUserAccountData(mModels);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
                mResponse.onFailure(error.getMessage());
            }
        });
    }

    public void addServiceData(final FirebaseResponseHandler
mResponse, UserAccountDataModel model) {
        String key = mAccountCheckRef.push().getKey();
        model.setKey(key);

mAccountCheckRef.child(key).setValue(model).addOnSuccessListen
er(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                mResponse.onSuccess(true, "Service Added");
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                mResponse.onFailure(e.getLocalizedMessage());
            }
        });
    }

    private void updateUserData(final FirebaseResponseHandler
mResponse, final UserDataModel userDataModel, final String msg) {
```

```
mUserRef.child(getSharedPrefString("user_key")).setValue(userData
Model).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                mResponse.onFailure(e.getLocalizedMessage());
            }
        }).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                mResponse.onSuccess(true, msg);
            }
        });
    }

    public void sendOrderAcceptNotification(String key) throws
JSONException {
        JSONObject notificationObj = new JSONObject();
        notificationObj.put("title",
context.getString(R.string.app_name));
        notificationObj.put("body", "Your order has been accepted.
Service person assigned.");
        notificationObj.put("receiver", "customer");
        notificationObj.put("sender", "admin");
        notificationObj.put("key", key);
        notificationObj.put("id",
Calendar.getInstance().getTimeInMillis());
        sendNotification(notificationObj, key);
    }

    public void sendOrderCompletedNotification(String key) throws
JSONException {
        JSONObject notificationObj = new JSONObject();
        notificationObj.put("title",
context.getString(R.string.app_name));
        notificationObj.put("body", "Your Santizaton has done.");
        notificationObj.put("receiver", "customer");
        notificationObj.put("sender", "service");
        notificationObj.put("key", key);
        notificationObj.put("id",
Calendar.getInstance().getTimeInMillis());
        sendNotification(notificationObj, key);
    }

    public void sendServiceNotification(String key) throws
JSONException {
        JSONObject notificationObj = new JSONObject();
        notificationObj.put("title",
context.getString(R.string.app_name));
        notificationObj.put("body", "Admin has allocated you new
order.");
        notificationObj.put("receiver", "service");
        notificationObj.put("sender", "admin");
        notificationObj.put("key", key);
        notificationObj.put("id",
Calendar.getInstance().getTimeInMillis());
        sendNotification(notificationObj, key);
    }

    public void sendRejectedNotification(String key) throws
JSONException {
        JSONObject notificationObj = new JSONObject();
        notificationObj.put("title",
context.getString(R.string.app_name));
        notificationObj.put("body", "Admin has rejected your order.");
        notificationObj.put("receiver", "customer");
        notificationObj.put("sender", "admin");
        notificationObj.put("key", key);
        notificationObj.put("id",
Calendar.getInstance().getTimeInMillis());
        sendNotification(notificationObj, key);
    }



    public void sendOrderPlacedNotification(String key) throws
JSONException {
        JSONObject notificationObj = new JSONObject();
        notificationObj.put("title",
context.getString(R.string.app_name));
```

```
        notificationObj.put("body", "New order has been placed.");
        notificationObj.put("receiver", "admin");
        notificationObj.put("sender", "customer");
        notificationObj.put("key", key);
        notificationObj.put("id",
Calendar.getInstance().getTimeInMillis());
        sendNotification(notificationObj, key);
    }

    private void sendNotification(JSONObject notificationObj, String
topic) {
        RequestQueue mRequestQue =
Volley.newRequestQueue(context);

        JSONObject json = new JSONObject();
        try {
            json.put("to", "/topics/" + topic);
            json.put("data", notificationObj);
            String url = "https://fcm.googleapis.com/fcm/send";
            JsonObjectRequest request = new
JsonObjectRequest(Request.Method.POST, url, json, new
Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    Log.d("MUR",
                            "onResponse:" + response);
                }
            }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.d("MUR",
                            "onError:" + error);
                }
            }) {
                @Override
                public Map<String, String> getHeaders() throws
AuthFailureError {
                    Map<String, String> header = new HashMap<String,
String>();
                    header.put("content-type", "application/json");
                    header.put("authorization",
"key=AAAAh8xnBC4:APA91bHQb054yI7YqWdYKgQ3PGFieRxl0
uE1XFDoM5J1abUxN7vSO6Z9WKqvarKetKXl02Ad6JPcYXbaL2U
2VoQMsMwlHYbtetSr-
adCd_a1_5gtGR9AkfavjvJbZqcVMjrhSJvgi2Q2");
                    return header;
                }
            };
            mRequestQue.add(request);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    public Uri generatePaymentLink(String orderDetails, String price)
{

        return Uri.parse("upi://pay").buildUpon()
                .appendQueryParameter("pa", "7506812373@ybl")
                .appendQueryParameter("pn",
getSharedPrefString("user_key"))
                .appendQueryParameter("tn", "orderDetails")
                .appendQueryParameter("am", price)
                .appendQueryParameter("cu", "INR")
                .build();
    }

    public void updateUserData(UserDataModel data, final
FirebaseResponseHandler mResponse) {

mUserRef.child(getSharedPrefString("user_key")).setValue(data).add
OnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                mResponse.onSuccess(true, "Profile Udpated");
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                mResponse.onFailure(e.getLocalizedMessage());
            }
        });
    }


}
```

CHAPTER 06

FUTURE WORK

FUTURE WORK

FUTURE SCOPE OF PROJECT

In future we can easily add new services if vendor wants too. We can apply artificial intelligence for giving recommendations and for measuring the size of the place through image and video processing. If the vendors agrees to we can also add multiple vendors to make it more of a market place experience than a single store

CHAPTER 07

CONCLUSION

CONCLUSION

The Sanitization service booking app will help the user by making the whole process of find the correct personnel for the job to booking a appointment for sanitization and disinfectant of a particular place, more convenient and easy. This will also help the vendor to focus more on the service than on the process of getting a order.

CHAPTER 08

BIBLIOGRAPHY & REFERANCE WEB SITE

BIBLIOGRAPHY & REFERANCE WEB SITE

Bibliography:

1. Murach's HTML5 and CSS3

2. Software Engineering – Roger Pressman

3. System Analysis and Design – Alias M. Awad

Websites:

1. www.microsoft.com

2. www.w3schools.com

3. www.dotnetspider.com