**Mini Project-1 report on**

**‘‘A-z Services’’**

*A mini project dissertation submitted in partial fulfilment of the requirement for the award of degree*

**MASTER OF COMPUTER APPLICATIONS**

by

**YASHWANTH KUMAR S**

**1BY23MC108**

**Under the Guidance of**

**Prof. SPURTHY S N**

Assistant Professor

Department of MCA

BMSIT&M

Bengaluru-560064



**Department of Master of Computer Applications**

**BMS Institute of Technology and Management**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru – 560064**

**OCTOBER - 2024**

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru – 560064**

**March-2024**

**Department of MCA**

(Accredited by NBA, New Delhi)



**CERTIFICATE**

This is to certify that **Mr. YASHWANTH KUMAR S** bearing USN **1BY23MC108** has successfully completed the VTU prescribed **Mini Project Work – 1 (22MCA208)** titled **A-z Services** at **Department of MCA, BMS Institute of Technology and Management, Bengaluru** under the guidance of **Ms.SPURTHY S N, ASSISTANT PROFFESSOR, Department of MCA** during the period from **July 2024 to October 2024.**

Signature of the Guide Signature of HoD

**Ms. SPURTHY S N Dr. M.Sridevi**

Assistant Professor Assistant Professor & Head

Department of MCA Department of MCA

BMSIT&M BMSIT&M

Bengaluru. Bengaluru

**DECLARATION**

I, YASHWANTH KUMAR S, a student of MCA at BMS Institute of Technology and Management, bearing USN 1BY23MC108, hereby declare that the mini project titled “A-z Services” has been carried out by me under the guidance of Ms. SPURTHY S N. This project is submitted in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications by Visvesvaraya Technological University during the academic year 2024. This report has not been submitted to any other organization or university for the award of any degree or certificate.

**ACKNOWLEDGEMENT**

I would like to take this opportunity to express my heartfelt gratitude to everyone who contributed to the successful completion of my project. I would especially like to acknowledge a few individuals to whom I owe sincere thanks and appreciation.

First and foremost, I extend my sincere gratitude to the BMSIT Management for providing excellent infrastructure and educational support, which have greatly contributed to shaping my career.

I would also like to express my deep appreciation to our Principal, Dr. Sanjay H.A., for his kind support and guidance throughout the course of this project.

My heartfelt thanks go to Dr. M. Sridevi, Head of the Department, for her invaluable insights and support in making this project possible.

I am profoundly grateful to my internal guide Assistant Professor Spurthy S N, and Project Coordinator Assistant Professor Venkatesh A for their unwavering support, encouragement, and valuable inputs throughout the entire process of completing this project.

I also extend my thanks to all the professors and non-teaching staff members who have contributed their help and support, directly or indirectly, in completing this project.

Finally, I would like to express my deepest gratitude to my parents and friends for their moral support and encouragement, which helped me in accomplishing this project.

Yashwanth Kumar S

1BY23MC108

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru – 560064**

**Department of MCA**



**VISION**

To develop quality professionals in Computer Applications who can provide sustainable solutions to the societal and industrial needs.

**MISSION**

Facilitate effective learning environment through quality education, state-of-the-art facilities, and orientation towards research and entrepreneurial skills.

**Programme Educational Objectives (PEOs)**

**PEO 1:** Develop innovative IT applications to meet industrial and societal needs.

**PEO 2:** Adapt themselves to changing IT requirements through life-long learning.

**PEO 3:** Exhibit leadership skills and advance in their chosen career.

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru – 560064**

**Department of MCA**

**Programme Outcomes (POs)**

**PO 1:** Apply knowledge of computing fundamentals, computing specialization, mathematics and domain knowledge to provide IT solutions.

**PO 2:** Identify, analyse and solve IT problems using fundamental principles of mathematics and computing sciences.

**PO 3:** Design, Develop and evaluate software solutions to meet societal and environmental concerns.

**PO 4:** Conduct investigations of complex problems using research-based knowledge and methods to provide valid conclusions.

**PO 5:** Select and apply appropriate techniques and modern tools for complex computing activities.

**PO 6:** Understand professional ethics, cyber regulations and responsibilities.

**PO 7:** Involve in life-long learning for continual development as an IT professional.

**PO 8:** Apply and demonstrate computing and management principles to manage projects in multidisciplinary environments by involving in different roles.

**PO 9:** Comprehend & write effective reports and make quality presentations.

**PO 10:** Understand the impact of IT solutions on socio-environmental issues.

**PO 11:** Work collaboratively as a member or leader in multidisciplinary teams.

**PO 12:** Identify potential business opportunities and innovate to create value for the society and seize that opportunity.

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru – 560064**

**Department of MCA**

**Course Outcomes (COs)**

**CO 1:** Analyse the given requirements.

**CO 2:** Design a suitable system model.

**CO 3:** Develop the solution using appropriate tools.

**CO 4**: Prepare effective documentation.

**CO 5:** Involve in team work.

# 

# ABSTRACT

The "**A-z Services**" mobile application is designed to simplify the process of hiring skilled professionals for various services such as plumbing, electrical work, and tutoring. The app provides a platform where users can easily browse through different service categories and connect with service providers in real-time. Built on the Android platform using Firebase for backend services, the application ensures smooth user authentication, data storage, and service management.

Users can create accounts by verifying their phone numbers through OTP (One-Time Password) authentication, while service providers and admins can manage services, receive notifications, and respond to customer requests. The app incorporates features such as real-time availability, profile management for service providers, and a rating and review system to maintain service quality.

This project aims to bridge the gap between users seeking reliable services and professionals looking to offer their skills, thus improving the overall experience of hiring skilled labor. The focus is on creating an intuitive, user-friendly interface and a robust backend to handle user data, services, and real-time communication effectively.

**Keywords**: Java, Firebase, Android Studio, User Authentication, Real-Time Database, Service Provider, Mobile Application, User Interface, Service Management.

**TABLE OF CONTENTS**

Page No.

1. **INTRODUCTION 01**
   1. Project description 01
   2. Technologies used 01
   3. Project Objectives 01
2. **LITERATURE SURVEY 02**
   1. Related Work
   2. Problem Definition 03
   3. Existing and Proposed System 03
   4. System study 03
      1. Feasibility 03
      2. Technical Feasibility 03
      3. Operational Feasibility 03
      4. Economic Feasibility 03
   5. System requirements 04
      1. Hardware Requirements 04
      2. Software Requirements 04
3. **SOFTWARE REQUIREMENTS SPECIFICATION 05**
   1. Functional requirements 05
   2. Non-Functional requirements 05
4. **SYSTEM DESIGN 06**
   1. System Architecture 06
   2. Context Diagram 07
5. **DETAILED DESIGN 08**
   1. Sequence diagram 08
   2. Data flow diagram 08
   3. Activity diagram 09
   4. Use Case diagram 10
6. **IMPLEMENTATION 11**
   1. Procedure
   2. Snippet code 12
   3. Screenshots (with Explanation) 41
7. **SOFTWARE TESTING 45**
   1. System Testing 45
   2. Test Cases 46
8. **CONCLUSION 48**
9. **FUTURE ENHANCEMENTS 49**
10. **REFERENCES 50**
11. **PLAGIARISM REPORT 51**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| SLNO | PARTICULARS | PAGE NO |
| 1. | System Architecture Diagram | 06 |
| 2. | Context Diagram | 07 |
| 3. | Sequence Diagram | 08 |
| 4. | Data Flow Diagram | 09 |
| 5. | Activity Diagram | 09 |
| 6. | Use Case Diagram | 11 |
| 7. | Firebase Console Screenshot | 41 |
| 8. | SDK setup configuration Screenshot | 42 |
| 9. | Home Screen of the Application | 43 |
| 10. | Registration Screen | 43 |
| 11. | Service Provider Dashboard | 44 |
| 12. | Admin Panel Screen | 44 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| SLNO | PARTICULARS | PAGE NO |
| 1 | Test Cases for Functionality Testing | 46 |
| 2 | Test Cases for Usability Testing | 47 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. **INTRODUCTION**

**1.1.Project Description**

The project,”**A-z Services”** is a mobile application designed to streamline the process of hiring professionals for various services, including plumbing, electrical work, and tutoring. The app serves as a platform connecting users with trained service providers, making it easier for users to find and hire the right professionals for their needs. With a user-friendly interface and efficient features, A-z Services aims to enhance user experience while ensuring quality service delivery.

**1.2. Technologies Used**

The A-z Services application is developed using the following technologies:

* + - **Android Studio**: The primary IDE used for developing the Android application.
    - **Firebase**: A cloud-based platform used for backend services, including user authentication, real-time database management, and analytics.
    - **Java**: The programming language employed for implementing application logic.
    - **XML**: Used for designing the user interface layouts.
    - **Gradle**: The build automation tool used to manage dependencies and build configurations.

**1.3. Project Objectives**

The objectives of the A-z Services project include:

* + - To provide a reliable platform for users to hire trained professionals for various services.
    - To implement user authentication and authorization features, ensuring secure access for both users and service providers.
    - To enable users to browse and select service providers based on their needs and preferences.
    - To facilitate easy communication and interaction between users and service providers through in-app messaging.
    - To incorporate a rating and review system, allowing users to share feedback on services received, thereby enhancing service quality.
    - To ensure a seamless user experience through an intuitive and responsive user interface.

**2.LITERATURE SURVEY**

**2,1,Related Work**

The increasing reliance on technology to facilitate everyday tasks has led to the development of various service-oriented applications. Research indicates that users prefer platforms that provide a seamless experience in hiring service professionals. Previous works have focused on specific service areas, such as home maintenance and tutoring, with applications that offer features like user reviews, service ratings, and real-time availability. These applications have proven effective in improving service accessibility and user satisfaction.

**2.2. Problem Definition**

Despite the growing number of service-oriented applications, challenges persist, including the lack of reliable service providers, difficulty in comparing different professionals, and insufficient communication channels between users and providers. Many existing platforms do not adequately verify the credentials of service providers, leading to a trust deficit among users. Additionally, users often face obstacles when trying to navigate multiple applications for different services, highlighting the need for a comprehensive solution.

**2.3. Existing and Proposed System**

**2.3.1 Existing Systems**

The existing systems primarily focus on specific service areas, such as plumbing, electrical work, and tutoring, without providing an integrated approach to service hiring. Users are often required to register on multiple platforms for different services, leading to fragmented user experiences. These systems typically offer limited functionalities and lack seamless integration between service providers and users. Common features in existing systems include:

* **Niche Focus**: Applications often cater to specific service sectors, making it challenging for users to find all the services they need in one place.
* **User Registration**: Multiple registrations across different platforms result in a cumbersome process for users, leading to potential confusion and frustration.
* **Limited Features**: Many existing systems provide basic functionalities such as service requests and user reviews but lack advanced features like real-time availability, integrated payment processing, and robust communication channels.

**2.3.2.Proposed System**

The proposed A-z Services application aims to consolidate various services into a single platform, enhancing the user experience by allowing access to multiple service categories in one place. Key features of the proposed system include:

* Integrated Platform: Users can access a wide range of services, from home maintenance to tutoring, all under one roof, reducing the need for multiple applications.
* User Authentication: The application will include a secure user authentication system, ensuring that user data is protected while allowing easy access to services.
* Service Categorization: Services will be categorized for easier navigation, enabling users to quickly find the service they need.
* Integrated Communication System: The application will feature an integrated messaging system for users and service providers to communicate efficiently, facilitating smoother service requests and updates.
* Real-Time Availability: Users can check the real-time availability of service providers, allowing for immediate scheduling of services.

**2.4. System Study**

**2.4.1. Feasibility**

A thorough feasibility study indicates that developing the A-z Services application is practical. The growing demand for service hiring platforms suggests a viable market, while advancements in technology facilitate the implementation of necessary features.

**2.4.2. Technical Feasibility**

The technical feasibility of the project is supported by the availability of robust development tools such as Android Studio and Firebase. These technologies enable efficient application development, user authentication, real-time database management, and analytics, ensuring a scalable and maintainable system.

**2.4.3. Operational Feasibility**

Operationally, the application is designed to enhance user experience by simplifying the hiring process. By providing an intuitive interface and integrating communication channels, the application ensures users can easily navigate and utilize its features, making it operationally feasible.

**2.4.4. Economic Feasibility**

From an economic perspective, the application can potentially generate revenue through service provider listings, premium features, and advertisements. The initial investment in development and marketing is expected to yield significant returns, considering the increasing demand for service-oriented applications.

**2.5. System Requirements**

**2.5.1. Hardware Requirement**

* **Processor:**
  + Minimum: Intel Core i5 (8th generation or later) or AMD Ryzen 5
  + Recommended: Intel Core i7 (10th generation or later) or AMD Ryzen 7
* **RAM:**
  + Minimum: 8 GB
  + Recommended: 16 GB or more
* **Storage:**
  + Minimum: 256 GB SSD
  + Recommended: 512 GB SSD or more
* **Graphics:** Integrated graphics are usually sufficient; dedicated GPU is optional.
* **Client Device:**
* Android smartphone with a minimum of 2 GB RAM and Android version 5.0 or above.
* **Server:** Firebase services for backend operations (no dedicated hardware needed).

**2.5.2. Software Requirements**

* **Operating System:**
* Windows 10 or macOS or Linux
* **Development Tools:**
* Android Studio (latest version)
* Java Development Kit (JDK) 8 or higher
* Firebase SDK
* Gradle (latest version)
* Android SDK Platform Tools

**3.SOFTWARE REQUIREMENTS SPECIFICATION**

**3.1. Functional Requirements**

The functional requirements outline the specific behaviors and functions of the A-z Services application. These include:

1. **User Authentication**
   * Users must be able to register and log in to the application using their phone number.
   * The system should send an OTP (One-Time Password) for verification during registration and login.
2. **Account Management**
   * Users should be able to create and manage their profiles, including personal details like name, location, and service preferences.
   * Admin users must have the ability to manage user accounts and service provider profiles.
3. **Service Browsing**
   * Users must be able to browse a list of available services categorized by type (e.g., plumbing, electrical, tutoring).
   * Each service should display relevant information, including service provider details and pricing.
4. **Booking Services**
   * Users must be able to select a service provider and book a service by providing necessary details such as service type, date, and time.
   * The application should allow users to cancel or reschedule their bookings.

**3.2. Non-Functional Requirements**

The non-functional requirements specify the quality attributes of the A-z Services application. These include:

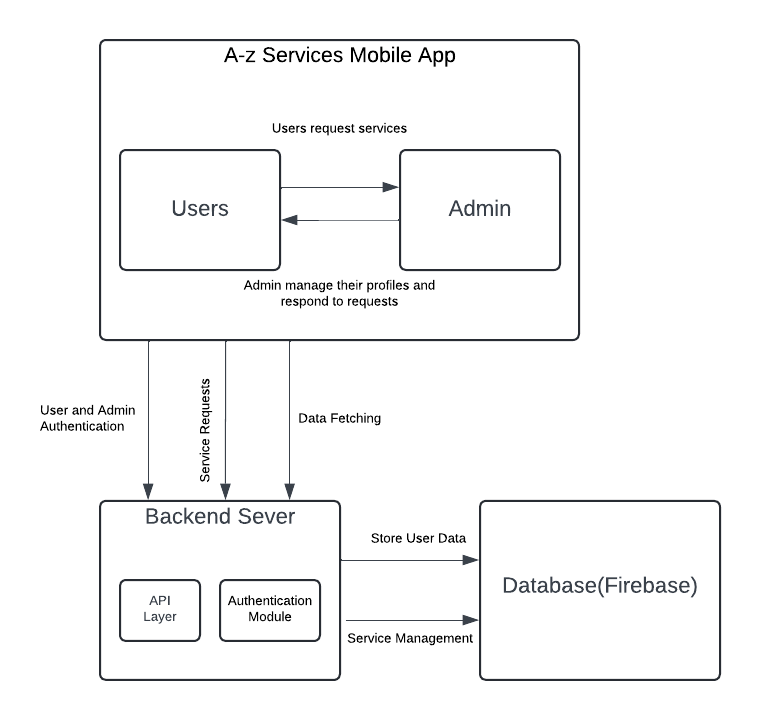
1. **Performance**
   * The application should be able to handle at least 1000 concurrent users without significant degradation in performance.
   * Response times for all actions (e.g., loading services, submitting bookings) should not exceed 2 seconds.
2. **Usability**
   * The application should have an intuitive user interface that allows users to navigate easily.
   * User documentation should be provided to assist users in understanding how to use the application.
3. **Security**
   * User data must be securely stored, with sensitive information (e.g., passwords, personal details) encrypted.
   * The application should implement measures to prevent unauthorized access and data breaches.
4. **Reliability**
   * The application should be available 99.9% of the time, with minimal downtime for maintenance.
   * Data consistency must be ensured, especially during transactions involving bookings and payments.
5. **Scalability**
   * The system should be designed to accommodate future growth, allowing for the addition of new services and users without significant rework.
   * The architecture should support easy integration of new features and enhancements.

**4. SYSTEM DESIGN**

**4.1. System Architecture**

The system architecture outlines the overall structure of the **A-z Services** mobile application, showcasing the interactions between various components such as the mobile app, backend server, and database. The architecture can be represented in a diagram that includes the following elements:

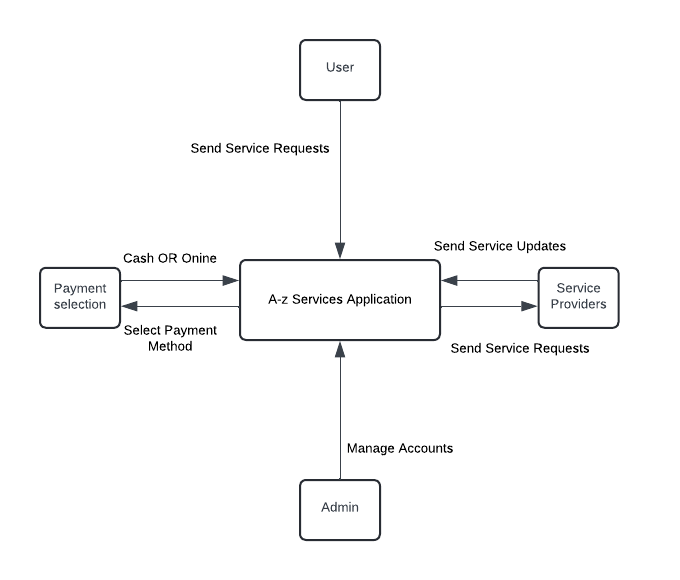
* **Mobile Application**: The user interface for both service users and service providers.
* **Backend Server**: Handles requests from the mobile app, processes data, and communicates with the database.
* **Database**: Stores user information, service listings and booking details.



**4.2. Context Diagram**

The context diagram provides a high-level view of the system and its interactions with external entities. It typically includes:

* **Actors**: External entities such as users (clients), service providers (admins), and external systems (like payment gateways).
* **System Boundary**: Defines what is included in the system (A-z Services app) and what is outside.
* **Interactions**: Arrows indicating how actors interact with the system, including the data flow.

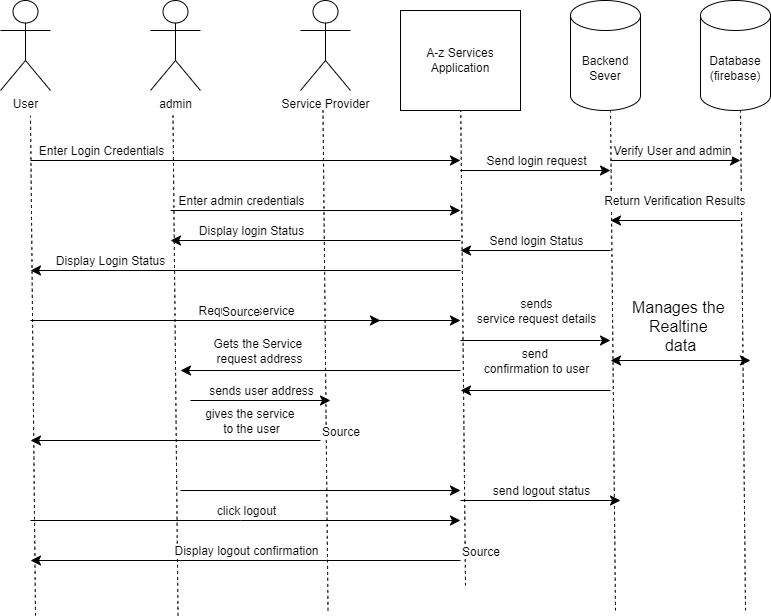


**5.DETAILED DESIGN**

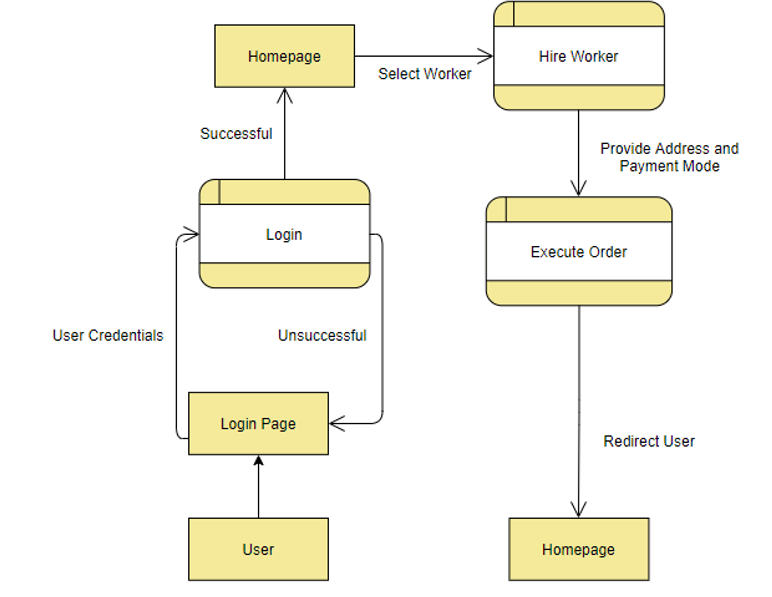
This section outlines the detailed design of the system, including various diagrams that represent the interactions and data flows within the application.

**5.1. Sequence Diagram**

The Sequence Diagram illustrates the interactions between users, the mobile application, and the database during the service request process. It shows how the system processes user requests and the flow of information.



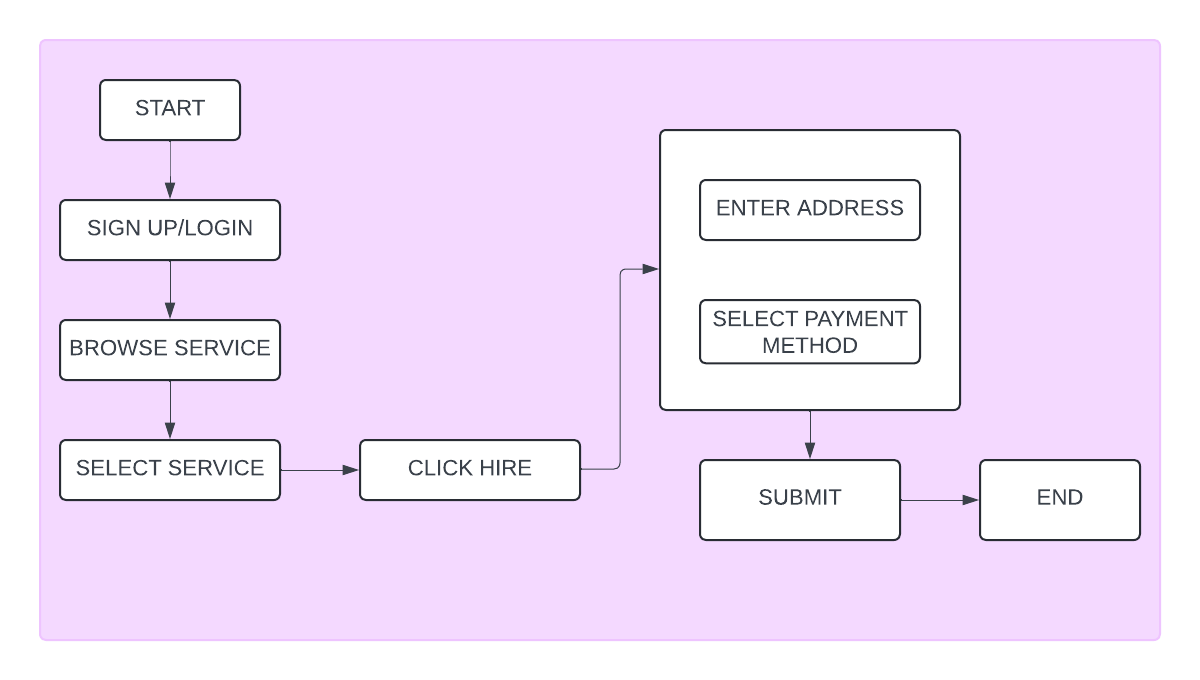
**5.2. Data Flow Diagram (DFD)**

* **Description**: The Data Flow Diagram provides a visual representation of how data moves within the system. It shows the relationships between different entities and the processes that handle data requests and responses.
* 

**5.3. Activity Diagram**

The Activity Diagram depicts the flow of activities in the application, focusing on user interactions and system responses. It highlights the various paths a user can take, including decision points and parallel processes.

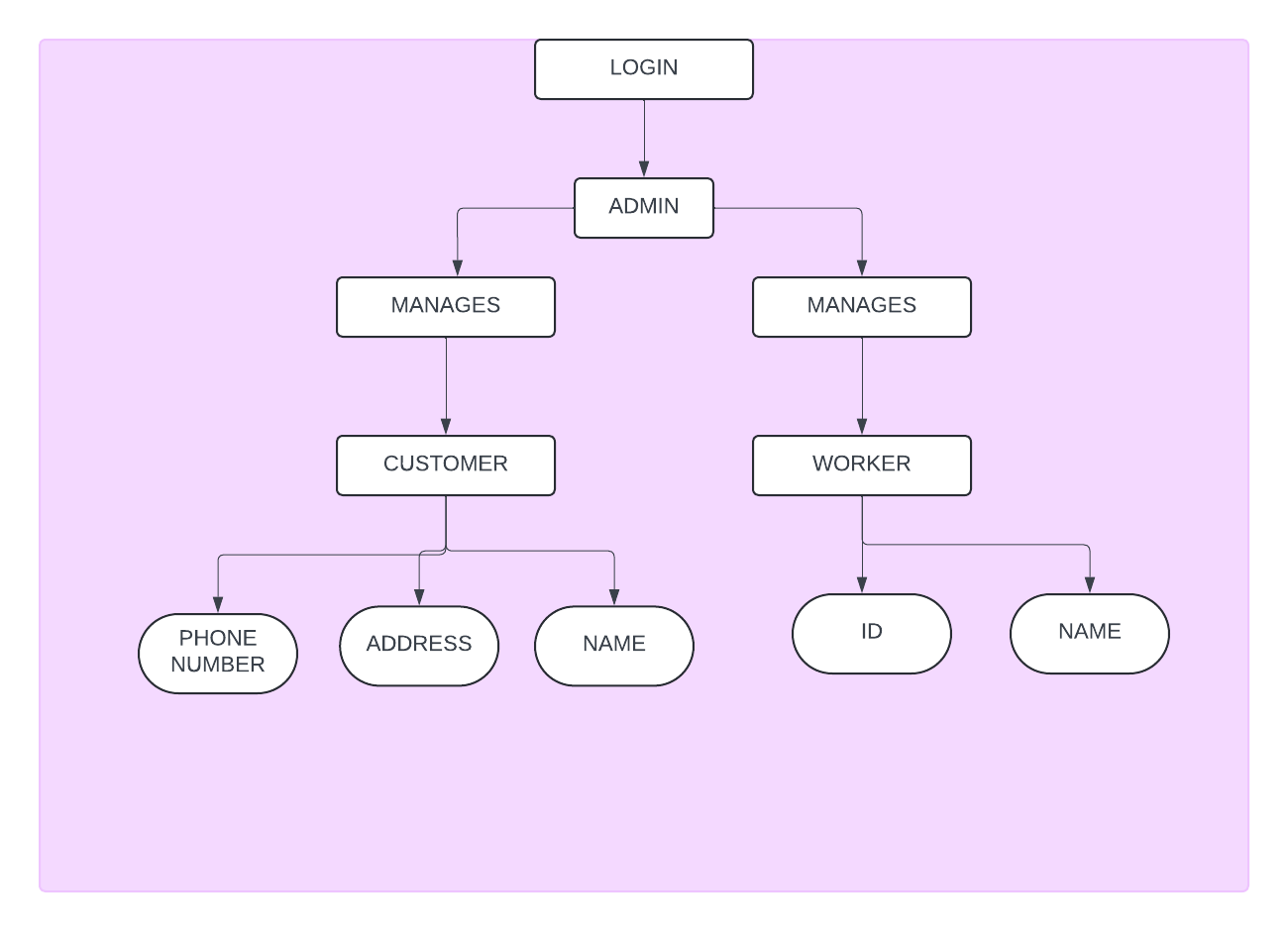
**5.3.1.USER ACTIVITY**



1. **Sign Up / Login**
   * User creates an account or logs into the system.
2. **Browse Services**
   * User navigates through different service categories (plumbing, electrical, etc.).
3. **Select Service**
   * User selects a specific service to request.
4. **Enter Details**
   * User inputs necessary details like address, service time, etc.
5. **Submit Service Request**
   * User submits the service request.
6. **End**:

* User logs out or returns to the main screen.

**5.3.2.** **ADMIN ACTIVITY**



1. **Sign Up / Login**

* Admin logs into the admin panel.

1. **View Dashboard**

* Admin views the main dashboard with options to manage users, service providers, and requests.

1. **Manage Users**

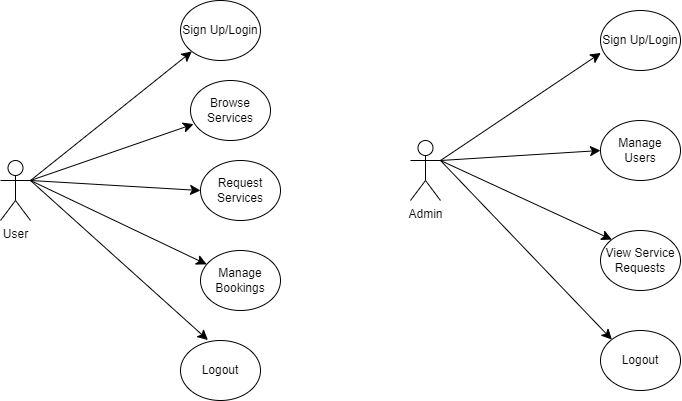
* Admin adds, removes, or updates user profiles.

1. **Manage Service Providers**

* Admin adds, removes, or updates service provider profiles.

**5.4. Use Case Diagram**

**5.4.1 USER and ADMIN USE CASE DIAGRAMs**



**6. IMPLEMENTATION**

**6.1. Procedure**

This section describes the step-by-step process involved in implementing the "A-z Services" application. It covers the initial setup, integration of features, and deployment procedures.

**1.Setup Environment**:

* + Install Android Studio and necessary SDKs.
  + Set up Firebase for authentication and database services.

**2.Create Project**:

* + Start a new Android project in Android Studio with the application name "A-z Services."

**3.Configure Dependencies**:

* + Update the build.gradle file to include necessary libraries (Firebase, Material Design, etc.).

**4.Design User Interface**:

* + Create XML layout files for all activities, including the splash screen, login, registration, and service selection screens.

**5.Implement Authentication**:

* + Use Firebase Authentication to handle user and service provider sign-ups and logins.
  + Implement OTP verification for mobile numbers.

**6.Database Integration**:

* + Use Firebase Realtime Database to store user profiles, service requests, and feedback.
  + Set up rules for data access and security.

**7.Service Management**:

* + Develop the functionality for service providers to add, manage, and respond to service requests.

**8.Testing**:

* + Conduct unit testing and user acceptance testing to ensure all features work as intended.
  + Fix any bugs identified during testing.

**9.Deployment**:

* + Prepare the app for deployment by generating a signed APK.
  + Publish the app on the Google Play Store following necessary guidelines.

**6.2. Snippet Code**

**Google-services.json (JavaScript Object Notation).**

{

"project\_info": {

"project\_number": "857621419367",

"firebase\_url": "https://a-z-services7-default-rtdb.firebaseio.com",

"project\_id": "a-z-services7",

"storage\_bucket": "a-z-services7.appspot.com"

},

"client": [

{

"client\_info": {

"mobilesdk\_app\_id": "1:857621419367:android:55fbd35b9c372d5a4e8bc2",

"android\_client\_info": {

"package\_name": "com.example.a\_zservices7"

}

},

"oauth\_client": [],

"api\_key": [

{

"current\_key": "AIzaSyB-CVvArem4-NXthqbfZuj1KGoLLJZjOTg"

}

],

"services": {

"appinvite\_service": {

"other\_platform\_oauth\_client": []

}

}

}

],

"configuration\_version": "1"

}

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.service.whatuwant">  
  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name="com.example.a\_zservices7.StandardCharges"></activity>  
 <activity android:name="com.example.a\_zservices7.AboutActivity" />  
 <activity android:name="com.example.a\_zservices7.SliderActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 <activity android:name="com.example.a\_zservices7.EmailActivity" />  
 <activity android:name="com.example.a\_zservices7.RequestedServiceActivity" />  
 <activity android:name="com.example.a\_zservices7.FinalActivity" />  
 <activity android:name="com.example.a\_zservices7.MessageActivity" />  
 <activity  
 android:name="com.example.a\_zservices7.HomeActivity"  
 android:label="@string/title\_activity\_home"  
 android:theme="@style/AppTheme.NoActionBar" />  
 <activity android:name="com.example.a\_zservices7.OtpActivity" />  
 <activity android:name="com.example.a\_zservices7.LoginActivity" />  
 <activity android:name="com.example.a\_zservices7.RegisterActivity" />  
 <activity android:name="com.example.a\_zservices7.welcomeActivity" />  
 <activity android:name="com.example.a\_zservices7.MainActivity" />  
  
 </application>  
  
</manifest>

**Java Code**

**MainActiviy**

package com.example.a\_zservices7;  
  
import android.content.Intent;  
import android.os.Handler;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.widget.TextView;  
  
import com.example.service.whatuwant.R;  
  
public class MainActivity extends AppCompatActivity {  
  
 private TextView welcome,welcome2,welcome3;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
   
 new Handler().postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Intent i=new Intent(MainActivity.this,welcomeActivity.class);  
 MainActivity.this.startActivity(i);  
 MainActivity.this.finish();  
 }  
 },3000);  
  
 }  
}

**WelcomeActivty**

package com.example.a\_zservices7;  
  
import android.content.Intent;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
  
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.example.a\_zservices7.Model.Users;  
import com.example.a\_zservices7.Prevalent.Prevalent;  
import com.example.service.whatuwant.R;  
  
public class welcomeActivity extends AppCompatActivity {  
  
 private Button Login, Register;  
 DatabaseReference mref;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_welcome*);  
  
 Login = (Button) findViewById(R.id.*login*);  
 Register = (Button) findViewById(R.id.*register*);  
  
 mref = FirebaseDatabase.*getInstance*().getReference();  
  
 Login.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i = new Intent(welcomeActivity.this, LoginActivity.class);  
 startActivity(i);  
 }  
 });  
  
 Register.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 startActivity(new Intent(welcomeActivity.this, RegisterActivity.class));  
 }  
 });  
 }  
}

**LoginActivty**

package com.example.a\_zservices7;  
  
import android.app.ProgressDialog;  
import android.content.Intent;  
import androidx.annotation.NonNull;  
import android.os.Bundle;  
import androidx.appcompat.app.AppCompatActivity;  
import android.text.TextUtils;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
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 com.example.a\_zservices7.Model.Users;  
import com.example.a\_zservices7.Prevalent.Prevalent;  
import com.example.service.whatuwant.R;  
  
public class LoginActivity extends AppCompatActivity {  
  
 private EditText Phone,Password;  
 private Button Login;  
 private DatabaseReference mref;  
 private ProgressDialog LoadingBar;  
 String myphone,mypassword;  
 TextView Newuser;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login*);  
  
  
 Phone=(EditText)findViewById(R.id.*loginphone*);  
 Password=(EditText)findViewById(R.id.*loginpassword*);  
 Login=(Button)findViewById(R.id.*loginbutton*);  
 Newuser=(TextView)findViewById(R.id.*Newuser*);  
  
 LoadingBar=new ProgressDialog(this);  
 mref= FirebaseDatabase.*getInstance*().getReference();  
 mypassword=Password.getText().toString();  
 myphone="+91"+Phone.getText().toString();  
  
  
 Newuser.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(LoginActivity.this,RegisterActivity.class);  
 startActivity(i);  
 }  
 });  
  
 Login.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if(TextUtils.*isEmpty*(Phone.getText().toString()))  
 {  
 Toast.*makeText*(LoginActivity.this, "Please enter your phone number..", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(TextUtils.*isEmpty*(Password.getText().toString()))  
 {  
 Toast.*makeText*(LoginActivity.this, "Please enter your password...", Toast.*LENGTH\_SHORT*).show();  
 }  
 else  
 {  
 LoginUser();  
 }  
 }  
 });  
 }  
  
 private void LoginUser() {  
  
  
  
 LoadingBar.setTitle("Login Account");  
 LoadingBar.setMessage("Please wait while we are checking our credentials..");  
 LoadingBar.setCanceledOnTouchOutside(false);  
 LoadingBar.show();  
  
   
  
 AllowAccessToUser(myphone,mypassword);  
 }  
  
 private void AllowAccessToUser(final String myphone, final String mypassword) {  
  
 if(Phone.getText().toString().equals("admin")&&Password.getText().toString().equals("admin"))  
 {  
 LoadingBar.dismiss();  
 Toast.*makeText*(LoginActivity.this, "Logged in Successfully..", Toast.*LENGTH\_SHORT*).show();  
 Intent i=new Intent(LoginActivity.this,RequestedServiceActivity.class);  
 // i.putExtra("Name",userdata.getName());  
 startActivity(i);  
 }  
 else  
 {  
 mref.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 if(dataSnapshot.child("Users").child("+91"+Phone.getText().toString()).exists())  
 {  
 final Users userdata=dataSnapshot.child("Users").child("+91"+Phone.getText().toString()).getValue(Users.class);  
 if(userdata.getPhone().equals("+91"+Phone.getText().toString()))  
 {  
  
 if(userdata.getPassword().equals(Password.getText().toString()))  
 {  
  
 LoadingBar.dismiss();  
 Toast.*makeText*(LoginActivity.this, "Logged in Successfully..", Toast.*LENGTH\_SHORT*).show();  
 Intent i=new Intent(LoginActivity.this,HomeActivity.class);  
 Prevalent.*currentOnlineUser*=userdata;  
 startActivity(i);  
  
  
 }  
 else  
 {  
 LoadingBar.dismiss();  
 Toast.*makeText*(LoginActivity.this, "please enter correct password..", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
 else  
 {  
 LoadingBar.dismiss();  
 Toast.*makeText*(LoginActivity.this, "please create your account first with this number ..", Toast.*LENGTH\_LONG*).show();  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 }  
}

**RegisterActivity**

package com.example.a\_zservices7;  
  
import android.content.Intent;  
import androidx.annotation.NonNull;  
import androidx.appcompat.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.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.example.service.whatuwant.R;  
  
public class RegisterActivity extends AppCompatActivity {  
  
 private EditText Name,Password,Phone;  
 private Button Register;  
 private DatabaseReference mref;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_register*);  
  
 Name=(EditText)findViewById(R.id.*nameentry*);  
 Password=(EditText)findViewById(R.id.*passwordentry*);  
 Phone=(EditText)findViewById(R.id.*phoneentry*);  
 Register=(Button)findViewById(R.id.*registerbutton*);  
 mref= FirebaseDatabase.*getInstance*().getReference();  
  
  
 Register.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if(TextUtils.*isEmpty*(Name.getText().toString()))  
 {  
 Toast.*makeText*(RegisterActivity.this, "Please enter your name first..", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(TextUtils.*isEmpty*(Phone.getText().toString()))  
 {  
 Toast.*makeText*(RegisterActivity.this, "Please enter your Phone Number first..", Toast.*LENGTH\_SHORT*).show();  
 }  
 else if(TextUtils.*isEmpty*(Password.getText().toString()))  
 {  
 Toast.*makeText*(RegisterActivity.this, "Please enter your password first..", Toast.*LENGTH\_SHORT*).show();  
 }  
  
 else  
 {  
 mref.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 if(!(dataSnapshot.child("Users").child(Phone.getText().toString()).exists()))  
 {  
 Intent i=new Intent(RegisterActivity.this,OtpActivity.class);  
 i.putExtra("Name",Name.getText().toString());  
 i.putExtra("Password",Password.getText().toString());  
 i.putExtra("Phone",Phone.getText().toString());  
 startActivity(i);  
 }  
 else  
 {  
 Toast.*makeText*(RegisterActivity.this, "Account with this number already exist..", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
  
 }  
 }  
 });  
 }  
}

**OtpActivity**

package com.example.a\_zservices7;  
  
import android.app.ProgressDialog;  
import android.content.Intent;  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
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.FirebaseException;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.PhoneAuthCredential;  
import com.google.firebase.auth.PhoneAuthProvider;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.example.service.whatuwant.R;  
  
import java.util.HashMap;  
import java.util.Map;  
import java.util.concurrent.TimeUnit;  
  
public class OtpActivity extends AppCompatActivity {  
  
 EditText OtpEntry;  
 Button Submit;  
 private ProgressDialog LoadingBar;  
 String Phone2,Phone,Name,Password,VerificationId;  
 private FirebaseAuth mAuth;  
 private DatabaseReference mref;  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_otp*);  
  
 Phone2=getIntent().getStringExtra("Phone");  
 Password=getIntent().getStringExtra("Password");  
 Name=getIntent().getStringExtra("Name");  
 Phone="+91"+Phone2;  
  
 mAuth=FirebaseAuth.*getInstance*();  
 mref= FirebaseDatabase.*getInstance*().getReference();  
  
 OtpEntry=(EditText)findViewById(R.id.*otpentry*);  
 Submit=(Button)findViewById(R.id.*otpbutton*);  
 LoadingBar=new ProgressDialog(this);  
// sendVerificationCode(Phone);  
  
 Submit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 String code=OtpEntry.getText().toString().trim();  
 if(code.isEmpty() || code.length() <6)  
 {  
 OtpEntry.setError("Enter Code..");  
 OtpEntry.requestFocus();  
 return;  
 }  
 LoadingBar.setTitle("Please Wait..");  
 LoadingBar.setMessage("Please Wait while we are checking our credentials...");  
 LoadingBar.show();  
// verifyCode(code);  
 CreateAccount(Phone,Name,Password);  
 }  
 });  
  
 }  
  
 private void verifyCode(String code)  
 {  
 PhoneAuthCredential credential=PhoneAuthProvider.*getCredential*(VerificationId,code);  
 //SignInWithCredential(credential);  
 CreateAccount(Phone,Name,Password);  
 }  
  
 private void CreateAccount(String phone, String name, String password)  
 {  
  
  
 Map<String,Object> details=new HashMap<String,Object>();  
 details.put("Phone",Phone);  
 details.put("Name",Name);  
 details.put("Password",Password);  
 details.put("Address","");  
 mref.child("Users").child(Phone).updateChildren(details)  
 .addOnCompleteListener(new OnCompleteListener<Void>() {  
 @Override  
 public void onComplete(@NonNull Task<Void> task) {  
  
 if(task.isSuccessful())  
 {  
 LoadingBar.dismiss();  
 Intent i=new Intent(OtpActivity.this,LoginActivity.class);  
 i.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 startActivity(i);  
 }  
 else  
 {  
 LoadingBar.dismiss();  
 Toast.*makeText*(OtpActivity.this, task.getException().getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
  
 /\*private void SignInWithCredential(final PhoneAuthCredential credential) {  
 mAuth.signInWithCredential(credential)  
 .addOnCompleteListener(new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
  
 if(task.isSuccessful())  
 {  
 Intent i=new Intent(OtpActivity.this,LoginActivity.class);  
 i.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 Toast.makeText(OtpActivity.this, credential.toString(), Toast.LENGTH\_SHORT).show();  
 startActivity(i);  
 }  
 else  
 {  
 Toast.makeText(OtpActivity.this, task.getException().getMessage(), Toast.LENGTH\_SHORT).show();  
 }  
 }  
 });  
 }\*/  
  
 private void sendVerificationCode(String phone) {  
  
 PhoneAuthProvider.*getInstance*().verifyPhoneNumber(  
 phone,  
 60,  
 TimeUnit.*SECONDS*,  
 this,  
 mCallbacks);  
 }  
 private PhoneAuthProvider.OnVerificationStateChangedCallbacks  
 mCallbacks=new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {  
  
 @Override  
 public void onCodeSent(String s, PhoneAuthProvider.ForceResendingToken forceResendingToken) {  
 super.onCodeSent(s, forceResendingToken);  
  
 VerificationId=s;  
 }  
  
 @Override  
 public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential) {  
  
 String code=phoneAuthCredential.getSmsCode();  
 if(code != null)  
 {  
 LoadingBar.setTitle("Please Wait..");  
 LoadingBar.setMessage("Please Wait while we are checking our credentials...");  
 LoadingBar.show();  
 verifyCode(code);  
 }  
 }  
  
 @Override  
 public void onVerificationFailed(FirebaseException e) {  
  
 Toast.*makeText*(OtpActivity.this, e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 };  
}

**HomeActivity**

package com.example.a\_zservices7;  
  
import android.content.Intent;  
import android.os.Bundle;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import android.view.View;  
import com.google.android.material.navigation.NavigationView;  
import androidx.core.view.GravityCompat;  
import androidx.drawerlayout.widget.DrawerLayout;  
import androidx.appcompat.app.ActionBarDrawerToggle;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.widget.Button;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import com.example.a\_zservices7.Prevalent.Prevalent;  
import com.example.service.whatuwant.R;  
  
public class HomeActivity extends AppCompatActivity  
 implements NavigationView.OnNavigationItemSelectedListener {  
  
 private Button Electrician,Plumber,Driver,Carpenter,BabySitter,Painter,Tutor,Guard,AcElectrician,ComputerOperator,Dumper,InteriorDesigner,Mason,SoftwareInstaller;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_home*);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.*toolbar*);  
 toolbar.setTitle("Home");  
 setSupportActionBar(toolbar);  
  
 FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.*fab*);  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if(Prevalent.*currentOnlineUser*.getPhone().equals("+911111111111"))  
 {  
 Intent i=new Intent(HomeActivity.this,LoginActivity.class);  
 Toast.*makeText*(HomeActivity.this, "Login First to Send Email..", Toast.*LENGTH\_LONG*).show();  
 }  
 else  
 {  
 Intent i=new Intent(HomeActivity.this,EmailActivity.class);  
 startActivity(i);  
 }  
  
 }  
 });  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  
 this, drawer, toolbar, R.string.*navigation\_drawer\_open*, R.string.*navigation\_drawer\_close*);  
 drawer.addDrawerListener(toggle);  
 toggle.syncState();  
  
 NavigationView navigationView = (NavigationView) findViewById(R.id.*nav\_view*);  
 navigationView.setNavigationItemSelectedListener(this);  
  
  
  
  
  
  
  
  
  
 View headerView=navigationView.getHeaderView(0);  
 TextView currentUser=headerView.findViewById(R.id.*currentuser*);  
  
 currentUser.setText(Prevalent.*currentOnlineUser*.getName());  
  
  
  
 Electrician=(Button)findViewById(R.id.*btnelectrician*);  
 Plumber=(Button)findViewById(R.id.*btnplumer*);  
 Driver=(Button)findViewById(R.id.*btndriver*);  
 Carpenter=(Button)findViewById(R.id.*btncarpenter*);  
 BabySitter=(Button)findViewById(R.id.*btnbabysitter*);  
 Painter=(Button)findViewById(R.id.*btnpainter*);  
 Tutor=(Button)findViewById(R.id.*btntutor*);  
 Guard=(Button)findViewById(R.id.*btnguard*);  
 AcElectrician=(Button)findViewById(R.id.*btnAcElectrician*);  
 ComputerOperator=(Button)findViewById(R.id.*btnComputeroperator*);  
 Dumper=(Button)findViewById(R.id.*btndumper*);  
 InteriorDesigner=(Button)findViewById(R.id.*btninteriordesigner*);  
 Mason=(Button)findViewById(R.id.*btnmason*);  
 SoftwareInstaller=(Button)findViewById(R.id.*btnsoftwareinstaller*);  
  
 Electrician.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Electrician");  
 startActivity(i);  
 }  
 });  
 Plumber.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Plumber");  
 startActivity(i);  
  
 }  
 });  
 Driver.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Driver");  
 startActivity(i);  
 }  
 });  
 Carpenter.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Carpenter");  
 startActivity(i);  
 }  
 });  
 BabySitter.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","BabySitter");  
 startActivity(i);  
 }  
 });  
 Painter.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Painter");  
 startActivity(i);  
 }  
 });  
  
 Tutor.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Tutor");  
 startActivity(i);  
 }  
 });  
 Guard.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Guard");  
 startActivity(i);  
 }  
 });  
 AcElectrician.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","AcElectrician");  
 startActivity(i);  
 }  
 });  
 ComputerOperator.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","ComputerOperator");  
 startActivity(i);  
 }  
 });  
 Dumper.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Dumper");  
 startActivity(i);  
 }  
 });  
 InteriorDesigner.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","InteriorDesigner");  
 startActivity(i);  
 }  
 });  
 Mason.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","Mason");  
 startActivity(i);  
 }  
 });  
 SoftwareInstaller.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent i=new Intent(HomeActivity.this,MessageActivity.class);  
 i.putExtra("RequestedWorker","SoftwareInstaller");  
 startActivity(i);  
 }  
 });  
 }  
  
 @Override  
 public void onBackPressed() {  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 if (drawer.isDrawerOpen(GravityCompat.*START*)) {  
 drawer.closeDrawer(GravityCompat.*START*);  
 } else {  
 if(Prevalent.*currentOnlineUser*.getPhone().equals("+911111111111"))  
 {  
 super.onBackPressed();  
  
 }  
 else  
 {  
  
 }  
 }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 getMenuInflater().inflate(R.menu.*home*, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
  
 /\* if (id == R.id.action\_settings) {  
 return true;  
 }  
 \*/  
  
 return super.onOptionsItemSelected(item);  
 }  
  
 @SuppressWarnings("StatementWithEmptyBody")  
 @Override  
 public boolean onNavigationItemSelected(MenuItem item) {  
 // Handle navigation view item clicks here.  
 int id = item.getItemId();  
  
 if (id == R.id.*nav\_about*)  
 {  
 Intent i=new Intent(HomeActivity.this,AboutActivity.class);  
 startActivity(i);  
 }  
 else if (id == R.id.*nav\_logout*) {  
  
 Intent i=new Intent(HomeActivity.this,welcomeActivity.class);  
 i.addFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK* | Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK*);  
 startActivity(i);  
  
 } else if (id == R.id.*nav\_standardcharge*) {  
  
 Intent i=new Intent(HomeActivity.this,StandardCharges.class);  
 startActivity(i);  
 }  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 drawer.closeDrawer(GravityCompat.*START*);  
 return true;  
 }  
}

**RequestedServiceActivity**

package com.example.a\_zservices7;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ArrayAdapter;  
import android.widget.ListView;  
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 com.example.service.whatuwant.R;  
  
import java.util.ArrayList;  
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.Set;  
  
public class RequestedServiceActivity extends AppCompatActivity {  
  
 ListView listView;  
 private ArrayAdapter<String> arrayAdapter;  
 private ArrayList<String> list=new ArrayList<>();  
  
 DatabaseReference mref;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_requested\_service*);  
  
 listView=(ListView)findViewById(R.id.*listview*);  
 mref= FirebaseDatabase.*getInstance*().getReference();  
  
 arrayAdapter=new ArrayAdapter<String>(this,android.R.layout.*simple\_list\_item\_1*,list);  
 listView.setAdapter(arrayAdapter);  
  
 mref.child("Services").addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 Set<String> set=new HashSet<String>();  
 Iterator i=dataSnapshot.getChildren().iterator();  
 while (i.hasNext())  
 {  
 set.add((String) ((DataSnapshot)i.next()).getValue());  
 }  
 list.clear();  
 list.addAll(set);  
 arrayAdapter.notifyDataSetChanged();  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 listView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {  
 @Override  
 public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {  
  
 String selected=listView.getItemAtPosition(position).toString();  
 Toast.*makeText*(RequestedServiceActivity.this, selected, Toast.*LENGTH\_SHORT*).show();  
 return false;  
 }  
 });  
  
 }  
}

**MessageActivity**

package com.example.a\_zservices7;  
  
import android.app.AlertDialog;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.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.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.Toast;  
  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.Task;  
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.example.a\_zservices7.Prevalent.Prevalent;  
import com.example.service.whatuwant.R;  
  
import java.text.DateFormat;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Map;  
  
public class MessageActivity extends AppCompatActivity {  
  
 String RequestedWorker;  
 EditText message;  
 Button Send;  
  
  
 RadioGroup radioGroup;  
 RadioButton radioButton;  
 DatabaseReference mref;  
 private ProgressDialog LoadingBar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_message*);  
  
 RequestedWorker=getIntent().getStringExtra("RequestedWorker");  
  
 Send=(Button)findViewById(R.id.*send*);  
 message=(EditText)findViewById(R.id.*sms*);  
  
 LoadingBar=new ProgressDialog(this);  
 mref= FirebaseDatabase.*getInstance*().getReference();  
  
 radioGroup=(RadioGroup)findViewById(R.id.*radioGroup*);  
  
  
  
  
 mref.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
  
 if ( dataSnapshot.child("Users").child(Prevalent.*currentOnlineUser*.getPhone()).child("Address").getValue().equals(""))  
 {  
  
 }  
 else  
 {  
 final AlertDialog.Builder builder=new AlertDialog.Builder(MessageActivity.this);  
 builder.setTitle("WUW Assistant!");  
 builder.setMessage("Do You Want To Use Address that you use Last Time");  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 mref.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 String myadd=dataSnapshot.child("Users").child(Prevalent.*currentOnlineUser*.getPhone()).child("Address").getValue().toString();  
 message.setText(myadd);  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
  
 }  
 });  
 builder.setNegativeButton("No", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 dialog.cancel();  
 }  
 });  
 builder.show();  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 Send.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
  
 if(TextUtils.*isEmpty*(message.getText().toString()))  
 {  
 Toast.*makeText*(MessageActivity.this, "Please provide your Address..", Toast.*LENGTH\_LONG*).show();  
 }  
 else  
 {  
 if(Prevalent.*currentOnlineUser*.getPhone().equals("+911111111111"))  
 {  
 Intent i=new Intent(MessageActivity.this,LoginActivity.class);  
 startActivity(i);  
 Toast.*makeText*(MessageActivity.this, "Login First..", Toast.*LENGTH\_SHORT*).show();  
 }  
 else  
 {  
 int radioId=radioGroup.getCheckedRadioButtonId();  
 radioButton=findViewById(radioId);  
  
  
 LoadingBar.setTitle("Please Wait..");  
 LoadingBar.setMessage("Sending message..");  
 LoadingBar.show();  
  
 String currentDateTimeString = DateFormat.*getDateTimeInstance*().format(new Date());  
 Date today = new Date();  
 SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss a");  
 String dateToStr = format.format(today);  
 //System.out.println(dateToStr);  
  
 String SMSmessage=message.getText().toString();  
 String send="My name is "+ Prevalent.*currentOnlineUser*.getName()+" i need a "+RequestedWorker  
 +" at address "+SMSmessage+" and i complete payment by "+radioButton.getText().toString()  
 +" mode and my phone number is "+Prevalent.*currentOnlineUser*.getPhone();  
  
 Map<String,Object> map=new HashMap<String,Object>();  
 map.put(Prevalent.*currentOnlineUser*.getPhone()+System.*currentTimeMillis*(),dateToStr+System.*currentTimeMillis*()+"\n"+send);  
  
 mref.addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
  
 dataSnapshot.getRef().child("Users").child(Prevalent.*currentOnlineUser*.getPhone()).child("Address").setValue(message.getText().toString().trim());  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 mref.child("Services").updateChildren(map)  
 .addOnCompleteListener(new OnCompleteListener<Void>() {  
 @Override  
 public void onComplete(@NonNull Task<Void> task) {  
  
 if(task.isSuccessful())  
 {  
 Toast.*makeText*(MessageActivity.this, "Message Send Successfully..", Toast.*LENGTH\_SHORT*).show();  
 LoadingBar.dismiss();  
 Intent i =new Intent(MessageActivity.this,HomeActivity.class);  
 startActivity(i);  
  
 }  
 else  
 {  
 LoadingBar.dismiss();  
 Toast.*makeText*(MessageActivity.this, task.getException().getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
  
 }  
  
 }  
  
 }  
  
 });  
 }  
}

**XML code**

**activity\_main.xml**

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

android:id="@+id/welcomeText"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome to A-z Services"

android:textSize="24sp" />

<Button

android:id="@+id/startButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Get Started" />

</LinearLayout>

**activity\_login.xml**

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

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

android:layout\_width="match\_parent"

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

<EditText

android:id="@+id/usernameInput"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Username"

android:layout\_marginTop="50dp"/>

<EditText

android:id="@+id/passwordInput"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:layout\_below="@id/usernameInput"

android:layout\_marginTop="20dp"/>

<Button

android:id="@+id/loginButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Login"

android:layout\_below="@id/passwordInput"

android:layout\_marginTop="20dp"/>

</RelativeLayout>

**activity\_register.xml**

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Register"

android:textSize="24sp"

android:layout\_gravity="center"/>

<EditText

android:id="@+id/registerUsername"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Username"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/registerEmail"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Email"

android:layout\_marginTop="20dp"/>

<EditText

android:id="@+id/registerPassword"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Password"

android:layout\_marginTop="20dp"/>

<Button

android:id="@+id/registerButton"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Register"

android:layout\_marginTop="20dp"/>

</LinearLayout>

**activity\_service\_list**

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Available Services"

android:textSize="24sp"

android:layout\_gravity="center"/>

<ListView

android:id="@+id/serviceListView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:layout\_marginTop="20dp"/>

</LinearLayout>

**6.3.Screenshots (with Explanation)**

Various key screenshots from the Firebase interface, including:

* Firebase Realtime Database: Displays the hierarchical data structure.
* Firebase Authentication: Shows the registered user list.
* Firebase Storage: Demonstrates how files are organized and managed.
* Firebase Analytics: Provides insights into user engagement and app activity.

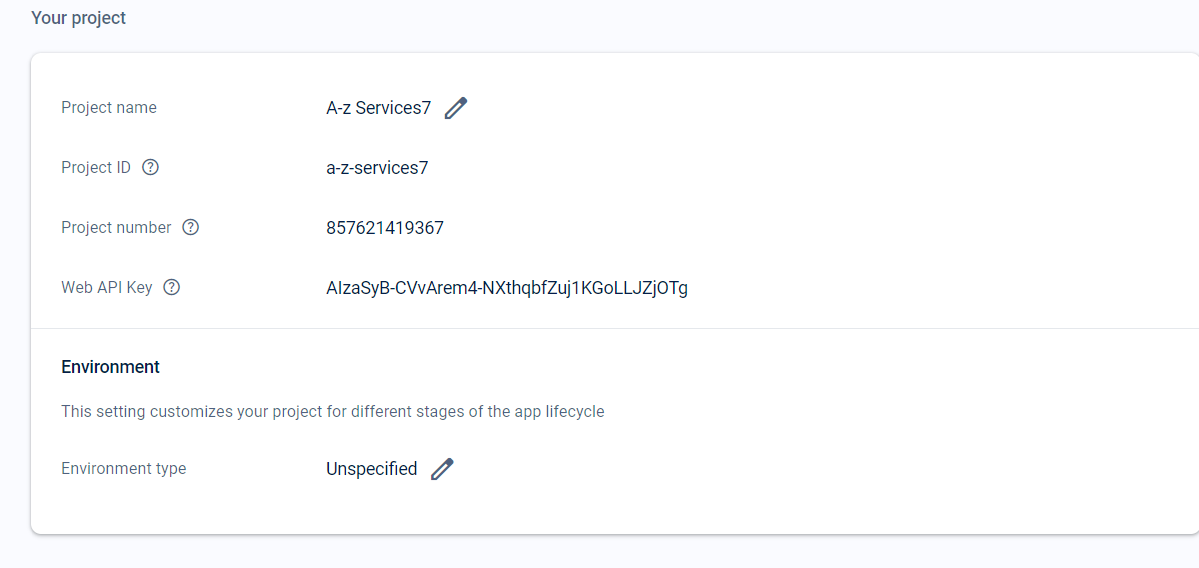
The use of Firebase in managing essential functionalities for real-time data, user authentication, and backend storage. This visual representation helps in understanding how Firebase integrates into the app's overall architecture.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated**

**Splash Screen**

* A screenshot of the initial splash screen displaying the welcome message "WUW."



**Selecting Screen**

* A Screenshot of the selection screen where users can create and login.

A screenshot of a phone

Description automatically generated

**Registration Screen**

* A screenshot of the registration screen where users input their details (name, phone number,) to create an account.

A screenshot of a phone

Description automatically generated

**Service List Screen and Hiring**

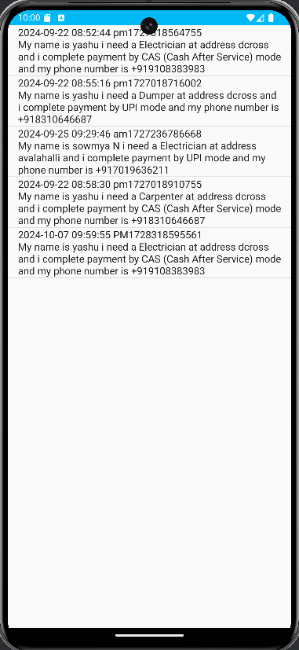
* A screenshot showing a list of available services (e.g., plumbing, electrical, tutoring) with options to select or request a service.
* You can click hire button and then next screen enter your address and select the payment method. And click hire that message will go to the admin. A screenshot of a phone

  Description automatically generated A screenshot of a phone

  Description automatically generated

**Admin Message Received**

* This screenshot shows the admin interface of the *A-z Services* app where the admin has received a notification about a new service request. The interface displays a message from a user, detailing the user's service request, including the type of service required (e.g., plumbing, electrical work), the user's location, and the requested date/time.



**7.SOFTWARE TESTING**

**7.1. System Testing**

System Testing is a critical phase in the software development lifecycle where the entire system is tested as a whole to ensure that all components and modules interact correctly. This testing evaluates the complete integrated system to verify that it meets the specified requirements.

In the context of **A-z Services**, system testing involves verifying that the app's features, such as user authentication, service categorization, real-time availability, and the interaction between the user interface and Firebase backend, function together seamlessly.

**Types of System Testing:**

**Functional Testing:**

* Ensures that the app's features work as expected. This includes creating accounts, logging in, verifying OTPs, managing services, and ensuring proper routing to user and admin interfaces.

**Usability Testing:**

* Ensures the user interface is intuitive and easy to use, with smooth navigation between screens and error handling for invalid inputs like incorrect phone numbers or passwords.

**Performance Testing:**

* Ensures that the app performs efficiently under various conditions, such as network issues, different device capabilities, and multiple simultaneous users.

**Security Testing:**

* Verifies the security measures in place, such as user authentication, password validation, and secure data transmission between the app and Firebase.

**Compatibility Testing:**

* Ensures that the app functions well across different Android devices with varying screen sizes and OS versions (Android 5.0 and above).

.

**7.2.** **Test Cases**

**7.2.1 Test Cases for Functional Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case IDA1:E11 | Input Values | Expected Result | Actual Result | Status |
| TC\_F01 | Name: John, Phone: +911234567890, Password: 123466 | Account created, OTP sent to phone | As Expected | Passed |
| TC\_F02 | Name: Admin, Phone: admin, Password: admin | Admin account created, OTP sent to phone | As Expected | Passed |
| TC\_F03 | OTP: 123456 (valid) | Successful login | As Expected | Passed |
| TC\_F04 | OTP: 6543212 (invalid) | Error: "Incorrect OTP" | As Expected | Passed |
| TC\_F05 | Phone: +911234567890, Password: wrongpass | Error: "Incorrect password" | As Expected | Passed |
| TC\_F06 | Select Service: Electrician | List of electricians displayed | As Expected | Passed |
| TC\_F07 | Click on Plumber category | List of plumbers displayed | As Expected | Passed |
| TC\_F08 | Name: Jane, Phone: 12345, Password: 1234 | Error: "Invalid phone number" | As Expected | Passed |
| TC\_F09 | Click "Forgot Password" | Password reset OTP sent to registered phone | As Expected | Passed |
| TC\_F10 | Admin login: Phone: +919876543210, Password: 223445 | Admin dashboard displayed | As Expected | Passed |

**7.2.2 Test Cases for Usability Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Input Values** | **Expected Result** | **Actual Result** | **Status** |
| TC\_U01 | Launch app | Splash screen visible for 3 seconds, then login/signup screen appears | As Expected | Passed |
| TC\_U02 | Click back from "Create Account" screen | Returns to login/signup screen | As Expected | Passed |
| TC\_U03 | Enter incorrect phone number format | Error message: "Invalid phone number format" | As Expected | Passed |
| TC\_U04 | Enter valid phone number, leave other fields blank | Error: "Please fill all fields" | As Expected | Passed |
| TC\_U05 | Select service from category list | Smooth transition to service provider list | As Expected | Passed |
| TC\_U06 | Click logout | Confirmation dialog displayed before logging out | As Expected | Passed |
| TC\_U17 | Try to navigate without filling password | Error message: "Password is required" | As Expected | Passed |

### 8.CONCLUSION

The **A-z Services** app successfully streamlines the process of hiring professionals for various services by integrating multiple service categories into a single platform. The app offers real-time service management and OTP-based user authentication, ensuring secure and efficient operations for both users and service providers. Admins can easily manage service requests, monitor provider activities, and respond promptly to client needs.

By implementing Firebase for backend services, the app ensures reliable user authentication, data storage, and real-time communication. Key features such as user and service provider registration, real-time service availability, and an intuitive UI have been developed to enhance user experience. The app minimizes the complexities of managing service requests, offering a simple, convenient, and secure platform.

This project has demonstrated the successful integration of mobile app development with modern cloud technologies. In conclusion, the A-z Services app will significantly simplify the hiring process and improve service accessibility for users, ultimately enhancing customer satisfaction and service efficiency.

**9. FUTUREENHANCEMENTS**

**1.Payment Integration:** Implement secure payment gateways (e.g., PayPal, Stripe) to allow users to pay for services directly through the app.

**2.Rating and Review System:** Enable users to rate and review service providers after the completion of services, helping improve transparency and quality.

**3.Real-time Chat**: Introduce a messaging system for users and service providers to communicate directly, enabling seamless coordination and faster responses.

**4.Geolocation Tracking:** Add real-time tracking of service providers, so users can see the provider’s estimated time of arrival.

**5.Service History:** Allow users and service providers to view past services and transaction history, making it easier to manage and reference previous jobs.

**6.Push Notifications:** Notify users about upcoming appointments, service status updates, and promotional offers to keep them engaged and informed.

**7.Multilingual Support:** Expand the app’s accessibility by incorporating multiple languages to cater to diverse user bases.

**8.Referral Program:** Introduce a referral system that rewards users for recommending the app to others, increasing user acquisition.

**9.Admin Dashboard Enhancements:** Provide additional reporting tools and analytics for admins to monitor service provider performance and user feedback.

**10.Discount and Coupon Integration:** Allow users to apply discount codes or coupons for services, incentivizing more usage of the app.

### .REFERENCES

### ****1.Android Developers Documentation****: "Guide to App Architecture." Official Android Documentation, Google. Available at: <https://developer.android.com/>

**2.Firebase Documentation**: "Firebase Authentication for Android." Firebase Official Documentation, Google. Available at: https://firebase.google.com/docs/android/setup

**3.Android Studio User Guide**: "Using Android Studio." JetBrains. Available at: <https://developer.android.com/studio/intro>

**4.Java Platform, Standard Edition**: "Java SE Documentation." Oracle. Available at: <https://docs.oracle.com/javase/8/docs/>

**5.XML Programming for Android**: "Professional Android 4 Application Development" by Reto Meier. ISBN: 978-1118102275.

**6.Firebase Cloud Messaging (FCM)**: "Setting Up Firebase Cloud Messaging." Firebase Documentation. Available at: <https://firebase.google.com/docs/cloud-messaging>.

**Online Tools :**

1. Chat GPT Open AI.
2. Youtube.
3. Lucidchart.

**11.PLAGIARISM REPORT**