**“AURAL”**

A project report submitted in partial fulfillment

Of requirement for the award of the degree of

**BACHELOR OF COMPUTER APPLICATIONS**



**RANI CHANNAMMA UNIVERSITY, BELAGAVI.**

**Submitted by:**

**Mr. Mallikarjun Rayar**

**Under the guidance of**

**Prof. Aparna Mutnalkar**



**Karnatak Law Society’s**

**GOGTE COLLEGE OF COMMERCE**

**Tilakwadi, Belgaum-590006**

**2010 - 2021**

**Karnatak Law Society’s**

**GOGTE COLLEGE OF COMMERCE**

**Tilakwadi, Belgaum.**

**Bachelor of Computer Applications**

****

**CERTIFICATE**

**This is to certify that**

**Mr. Mallikarjun Rayar Reg No. M1810447**

**Has satisfactorily completed the project work entitled**

**“AURAL”**

**FOR THE FULFILLMENT OF BACHELOR OF COMPUTER APPLICATIONS OF RANI CHANNAMMA UNIVERSITY, BELAGAVI, FOR THE YEAR 2019-2020**

**Guide Coordinator Director, BCA section**

**Prof. Vaishali Kale Prof. V. S. Jalihal Dr. S. G.Sugur**

**EXAMINERS**

**1. 2.**

***DEDICATED TO***

***TEACHERS AND***

***FRIENDS***

**ACKNOWLEDGEMENT**

***“A helping hand, kind soul these are the ingredients that help to make a success out of any effort.”***

We take this opportunity to acknowledge the contribution of each individual who has in some way or the other helped me in completing this project successfully. We express our gratitude to our institute, **Karnatak Law Society’s Gogte College of Commerce (BCA Department), Belagavi** and our Director **Dr.S.G.Sugur, BCA section** for being the source of encouragement.

We also enhance our gratitude to our beloved **Prof.Venugopal Jalihal**, Co-Ordinator BCA Dept for his constant inspiration and necessary resources and working environment in the college.

With pleasure, we use this occasion to concede our heartfelt thanks to **Prof. Vaishali Kale** our internal guide who have guided us with their valuable suggestions & guidance.

Our expressions extend unbounded to thank our most beloved parents & family members who have always been a moral support & strong pillars at every stage of our life with cheer enthusiasm. We dedicate our work to them. Last but not least, we are thankful to the Almighty for giving us moral support, which helped us during the successful completion of the project.

**With heartfelt thanks to One & All,**

***Mallikarjun Rayar***

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **TOPICS** | **PAGE NO.** |
| 1. | Introduction   1. Introduction 2. Objectives of Project 3. Scopes of Project | 1 |
| 2. | System Analysis   1. Problem Statement 2. System Requirements 3. Functional Requirements 4. Non-Functional Requirements 5. Methadology 6. Pern Chart 7. Gantt Chart | 3 |
| 3 | System Design   1. System Specification 2. Data Flow Diagram 3. Use Case Diagram 4. Entity-Relationship Diagram | 12 |
| 4. | System Implementation   1. System Specification 2. Technologies used 3. Coding | 19 |
| 5. | System Evaluation   1. Software Testing 2. Test Cases 3. Input/Output Scenes | 82 |
| 6. | Conclusion   * 1. Conclusion   6.2 System Requirements | 95 |
| 7. | References  7.1 Sites  7.2 Books | 97 |

**1.INTRODUCTION**

**1.1 Introduction**.

* The technology has been available for years but the acceptance it was quit recent.
* Our project is converting the PDF file to listenable voice. It is made up of two application, which runs on the user’s Pc and server application, which runs on any Pc on the network.
* To start listening to the PDF, user should get connected to server with security measures.

**1.2 Objectives**

* The technology has been available for years but the acceptance it was quit recent.
* The main objective of Aural application is to convert the PDF files to listenable format.
* It speak out all the information about the PDF file.

**1.3 Scope of the Project**

* Instant Conversion of PDF to audio.
* Authentication of user.
* Real Time feel to the users.
* Small in size.
* Easy to use.

**2.SYSTEM ANALYSIS**

**2.1 ProblemStatement**

This project is to create a application with a server and users to enable the user to convert the PDF file and listen on the go.To develop an instant converting solution to enable users to seamlessly listen and understand the PDF file information.The project should very easy to use enabling even a novice person to use it.

**2.2 SystemRequirements**

**HardwareRequirements**

* System**:** i5 Processor and above.
* Hard Disk**:** 500 GB and above.
* RAM: 8 GB and above.

**SoftwareRequirements**

* Operating system: Windows
* Browser: Chrome
* Coding language: Android SDK, JavaScript
* IDE**:** Android Studio.
* Code-Editor: Visual Studio Code
* Backend Service**:** Firebase.

**Runtime Environment**

* Device : Smartphone / Tablet.
* OS: Android.
* Minimum Version : 4.4 Android Kitkat.

**2.3 FunctionalRequirements**

* **User Sign-up/Sign-in:**

User must be able to Signup and Signin for the application through a email id.At initial phase the user need to Signup,if the user already have an account then the user can proced with the Signin

* **Uploading PDF file:**

The application should require PDF file to convert that file to audio format.

* **Selecting Page:**

User should mention page number of the PDF file which he would like to listen to.So that it is convenient to the user to listen particular information of the file.

* **Set Speed Rate:**

User should be able to set the audio speed rate at which user want to listen.

* **Listen / Start:**

Now user can be able to start listening to the PDF file, by clicking on the Listen/Start button, and application start’s reading the the information inside the PDF file.

* **Stop :**

User must be able to stop listening to the PDF file information by clicking on the Stop button.

* **Sign-out:**

The user can Signout from his account.

**2.4 Non-functional Requirements**

* **Scalability**

App should be able to provide instant conversion services.

* **Privacy**

Users account should be encrypted to maintain privacy.

* **Robustness**

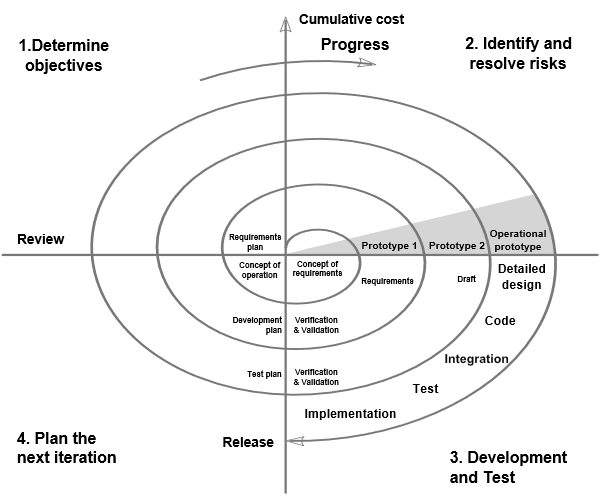
In case user’s device crashes, a backup of their account details must be stored on remote database servers to enable recoverability .

* **Performance**

Application must be lightweight and must convert PDF files to audio instantly.

**2.5 Methodology Adopted**

**Spiral model:**

****

**Figure 2.1**

**Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. **Each loop of the spiral is called a Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks.

As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model. The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions**

Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

1. **Identify and resolve Risks**

During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.

1. **Development and test**

During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

1. **Review and plan for the next Phase**

In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

**Risk Handling in Spiral Model**

A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports coping up with risks by providing the scope to build a prototype at every phase of the software development.

**Advantages of Spiral Model**

* **Risk Handling**

The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.

* **Good for large projects**

It is recommended to use the Spiral Model in large and complex projects.

* **Flexibility in Requirements**

Change requests in the Requirements at later phase can be incorporated accurately by using this model.

* **Customer Satisfaction**

Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

**Disdvantages of Spiral Model**

* **Complex**

The Spiral Model is much more complex than other SDLC models.

* **Expensive**

Spiral Model is not suitable for small projects as it is expensive.

* **Too much dependable on Risk Analysis**

The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.

* **Difficulty in time management**

As the number of phases is unknown at the start of the project, so time estimation is very difficult.

**2.6 PERT CHART**

A PERT chart is a graphic representation of a project’s schedule, showing the sequence of tasks, which tasks can be performed simultaneously, and the critical path of tasks that must be completed on time in order for the project to meet its completion deadline.A PERT chart can document an entire project or a key phase of a project.

The chart allows a team to avoid unrealistic timetables and schedule expectations, to help identify and shorten tasks that are bottlenecks, and to focus attention on most critical tasks. Because it is primarily a project-management tools, a PERT chart is most useful for planning and tracking entire projects or for scheduling and tracking the implementation phase of a planning or improvement effort.

**PERT CHART**

|  |  |
| --- | --- |
| Initial Study | |
| 20-12-2019 | 30-12-2019 |

|  |  |
| --- | --- |
| Requirement Analysis | |
| 31-12-2019 | 17-01-2020 |

|  |  |
| --- | --- |
| Design | |
| 18-01-2020 | 31-01-2020 |

|  |  |
| --- | --- |
| Coding | |
| 01-02-2020 | 29-02-2020 |

|  |  |
| --- | --- |
| Testing | |
| 01-03-2020 | 15-03-2020 |

|  |  |
| --- | --- |
| Final Document | |
| 15-03-2020 | 31-03-2020 |

**Figure 2.2**

**2.7 GANTT CHART**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Weeks | | | | | | | | | | | | | | |
|  | Tasks | 1 | 2 | | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 1 | Initial Study |  | |  | | | | | | | | | | | | |
| 2 | Requirement Analysis |  | |  | | |  | | | | | | | | | |
| 3 | Design |  | | | | |  | |  | | | | | | | |
| 4 | Coding |  | | | | | | |  | | | |  | | | |
| 5 | Testing and Implementation |  | | | | | | | | | | |  | |  | |
| 6 | Final Document |  | | | | | | | | | | | | |  | |

**Figure 2.3**

**3.SYSTEM DESIGN**

**3.1 System Specification**

**Hardware requirements:**

* Hard Disk: 500 GB and above.
* RAM: 8 GB and above.
* System: i5 Processor and above.

**3.2 Data Flow Diagrams**

**Data Flow Diagrams (DFD)**

A Data Flow Diagram is graphical aid for defining systems input, process and outputs. It represents increasing information flow and function details.

A DFD shows the flow of data through a system. A system may be an organization, a manual procedure, software system, mechanical system, a hardware system or any combinations of these. A DFD shows the movement of data through different processes in the system DFD’s are made of a number of symbols, which represents system components like process, data flow and external entities.

1. **Process**

Process show that systems do each process can have one or more inputs or outputs. A process in DFD is represented as hollow cylinder. And each process has unique name and number.

**b)Data Store**

A file or data store is repository of data. Each data store is represented by an open sided rectangle and each store and has unique name. The symbol of data store is.

1. **External Entities**

External entities are outside the system but they either supply input into system or use the system output. They may be an organization’s customer or others which system interacts. External entities, which supply data to the system, are sometimes called as ‘sinks’. These are represented by a rectangle in the DFD

1. **Data Flow**

Data flow indicates the passage of data in the system, from where the data flows. It is indicated by an arrow which indicates the direction of flow. The arrow is labeled by the name of the data flow.

Flow of data in the system could be any one of the following:

* From a data store to a process.
* From source to process.
* From a process to a sink.

**Context Level DFD of the proposed system:**

Signup View

Admin

Users

Update

Signin

**Figure 3.1**

**Top Level DFD:**

Authentication DB

REDIRECT TO LOGIN PAGE

STORES

STORES

MESSAGES,PICS

USER INFO

MAIL ID,PASSWORD

CONTROLS

ADMIN INFO

STATUS DB

MESSAGE DB

USERS

ADMIN

**Figure 3.2**

**3.3Use Case Diagram**

Users

**Figure 3.3**

**3.4 Entity Relationship**

**Diagram:**

Manages

Signin

Signup

Home

**Figure 3.4**

**4.SYSTEM IMPLEMENTATION**

**4.1 System Specification**

**Hardware Requirements**

* System: i5 Processor and above.
* Hard Disk: 500 GB and above.
* RAM: 8 GB and above.

**Software Requirements:**

* Operating system**:** Windows
* Coding language**:** Android SDK, JavaScript
* IDE:Android Studio.
* Code-Editor: Visual Studio Code
* Backend Service: Firebase.

**Runtime Environment:**

* Device : Smartphone / Tablet.
* OS: Android.
* Minimum Version : 4.4 Android Kitkat.

**Tools and Languages:**.

* I**DE**

1..Android Studio

* **Front End**

1.React Native

2.HTML, CSS, JavaScript

* **Middleware**

1.Java

2.Android SDK

3.Node js

* **Backend Services**

1.Firebase

2.GitHub Pages

**4.2 TECHNOLOGY USED**

1. **React Native:-**

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It’s based on React, Facebook’s JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly “native,” all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS −

* **React has wider audience** – You don’t have to decide which audience to target, i.e. iOS or Android users, as cross platform software runs on both, which gives you access to wider user base.
* **Platform consistency** – There are some navigation and design differences between iOS and Android, which – in cross-platform development – are dealt with by default, thanks to the shared codebase, This helps with creating a consistent app brand identity on both platform with less effort than if build on native.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML-esque markup, known as JSX. Then, under the hood, the React Native “bridge” invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user’s location.

1. **Java:-**

Java is a general-purpose computer **programming language** that is [concurrent](https://howtodoinjava.com/java-concurrency-tutorial/), class-based, [object-oriented](https://howtodoinjava.com/oops/object-oriented-principles/), and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers **“write once, run anywhere” (WORA)**, meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

For example, you can write and compile a Java program on UNIX and run it on Microsoft Windows, Macintosh, or UNIX machine without any modifications to the source code. WORA is achieved by compiling a Java program into an intermediate language called **byte code**. The format of byte code is platform-independent. A virtual machine, called the [Java Virtual Machine (JVM)](https://howtodoinjava.com/java/basics/jdk-jre-jvm/), is used to run the byte code on each platform.

1. **Node js:-**

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), [JavaScript](https://en.wikipedia.org/wiki/JavaScript) runtime environment that executes JavaScript code outside of a [web browser](https://en.wikipedia.org/wiki/Web_browser). Node.js lets developers use JavaScript to write command line tools and for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting)—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page) content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying [web-application](https://en.wikipedia.org/wiki/Web_application) development around a single programming language, rather than different languages for server- and client-side scripts.

Though .js is the standard [filename extension](https://en.wikipedia.org/wiki/Filename_extension) for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an [event-driven architecture](https://en.wikipedia.org/wiki/Event-driven_architecture) capable of [asynchronous I/O](https://en.wikipedia.org/wiki/Asynchronous_I/O). These design choices aim to optimize [throughput](https://en.wikipedia.org/wiki/Throughput) and [scalability](https://en.wikipedia.org/wiki/Scalability) in web applications with many input/output operations, as well as for [real-time Web](https://en.wikipedia.org/wiki/Real-time_Web) applications (e.g., [real-time communication](https://en.wikipedia.org/wiki/Real-time_communication) programs and [browser games](https://en.wikipedia.org/wiki/Browser_game)). The Node.js [distributed development](https://en.wikipedia.org/wiki/Distributed_development) project was previously governed by the Node.js Foundation, and has now merged with the [JS Foundation](https://en.wikipedia.org/wiki/JS_Foundation) to form the [OpenJS Foundation](https://en.wikipedia.org/wiki/OpenJS_Foundation), which is facilitated by the [Linux Foundation](https://en.wikipedia.org/wiki/Linux_Foundation)'s Collaborative Projects program.

1. **Firebase:**

**Firebase** is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google’s infrastructure. Firebase is categorized as a [NoSQL](https://www.educative.io/edpresso/whats-the-difference-betweensql-and-nosql) database program, which stores data in JSON-like documents.

In Firebase, a document is a set of key-value pairs defined by a schema. A group of documents makes up a collection.

**Key Features**

1. **Authentication:**

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

### Real-time database:

Data is synced across all clients in real-time and remains available even when an app goes offline. It uses JSON tree to save the data.

1. **Cloud Firestore:**

Cloud Firestore is a cloud-hosted, NoSQL database that your ios, android, and web app can access directly via native SDKs. It uses document/collection model to save the data

### Hosting:

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

### Test lab:

The application is tested on virtual and physical devices located in Google’s data centers.

### Cloud Messaging:

Firebase Cloud Messaging, formerly known as Google Cloud Messaging, is a cross-platform cloud solution for messages and notification for Android, iOS, and Web application, which currently can be used at no cost.

1. **Android SDK:**

Every time Google releases a new version of Android, a corresponding SDK is also released. To be able to write programs with the latest features, developers must download and install each version’s SDK for the particular phone.

The development platforms that are compatible with SDK include operating systems like Windows (XP or later), Linux (any recent Linux distribution) and Mac OS X (10.4.9 or later). The components of Android SDK can be downloaded separately. Third party add-ons are also available for download.

Although the SDK can be used to write Android programs in the command prompt, the most common method is by using an integrated development environment (IDE). The recommended IDE is Eclipse with the Android Development Tools (ADT) plug-in. However, other IDEs, such as NetBeans or IntelliJ, will also work. Most of these IDEs provide a graphical interface enabling developers to perform development tasks faster. Since Android applications are written in Java code, a user should have the Java Development Kit (JDK) installed.

**4.3 Code**

Android Manifest:

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.myapp2">  
  
<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
<uses-permission android:name="android.permission.CAMERA" />  
<uses-permission android:name="android.permission.RECORD\_AUDIO" />  
  
<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=".Pdf\_View\_Activity"></activity>  
<activity android:name=".TakePhotoActivity" />  
<activity android:name=".ForegetPassActivity" />  
<activity android:name=".QRCodeActivity" />  
<activity android:name=".ImageViewerActivity" />  
<activity  
 android:name=".ChatActivity"  
 android:parentActivityName=".MainActivity" />  
<activity android:name=".ProfileActivity" />  
<activity  
 android:name=".FindFriendsActivity"  
 android:parentActivityName=".MainActivity" />  
<activity android:name=".PhoneLoginActivity" />  
<activity android:name=".GroupChatActivity" />  
<activity android:name=".SettingsActivity" /><activity android:name=".RegisterActivity" />  
<activity android:name=".LoginActivity" />  
<activity android:name=".MainActivity">  
<intent-filter>  
<action android:name="android.intent.action.MAIN" />  
  
<category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>  
</activity>  
<activity  
 android:name="com.theartofdev.edmodo.cropper.CropImageActivity"  
 android:theme="@style/Base.Theme.AppCompat" />  
</application>  
  
</manifest>

**JAVA CODE:**

**REGISTER.JAVA**

package com.example.codingcafe.whatsapp;

import android.app.ProgressDialog;

import android.content.Intent;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

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

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

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.iid.FirebaseInstanceId;

public class RegisterActivity extends AppCompatActivity

{

private Button CreateAccountButton;

private EditText UserEmail, UserPassword;

private TextView AlreadyHaveAccountLink;

private FirebaseAth mAuth;

privateDatabaseReference RootRef;

private ProgressDialog loadingBar;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

mAuth = FirebaseAuth.getInstance();

RootRef = FirebaseDatabase.getInstance().getReference();

InitializeFields();

AlreadyHaveAccountLink.setOnClickListener(new View.OnClickListener()

{

@Override

public void onClick(View view)

{

SendUserToLoginActivity();

}

});

CreateAccountButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

CreateNewAccount();

}

});

}

private void CreateNewAccount()

{

String email = UserEmail.getText().toString();

String password = UserPassword.getText().toString();

if (TextUtils.isEmpty(email))

{

Toast.makeText(this, "Please enter email...", Toast.LENGTH\_SHORT).show();

}

if (TextUtils.isEmpty(password))

{

Toast.makeText(this, "Please enter password...", Toast.LENGTH\_SHORT).show();

}

else

{

loadingBar.setTitle("Creating New Account");

loadingBar.setMessage("Please wait, while we wre creating new account for you...");

loadingBar.setCanceledOnTouchOutside(true);

loadingBar.show();

mAuth.createUserWithEmailAndPassword(email, password)

.addOnCompleteListener(new OnCompleteListener<AuthResult>() {

@Override

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

{

if (task.isSuccessful())

{

String deviceToken = FirebaseInstanceId.getInstance().getToken();

String currentUserID = mAuth.getCurrentUser().getUid();

RootRef.child("Users").child(currentUserID).setValue("");

RootRef.child("Users").child(currentUserID).child("device\_token")

.setValue(deviceToken);

SendUserToMainActivity();

Toast.makeText(RegisterActivity.this, "Account Created Successfully...", Toast.LENGTH\_SHORT).show();

loadingBar.dismiss();

}

else

{

String message = task.getException().toString();

Toast.makeText(RegisterActivity.this, "Error : " + message, Toast.LENGTH\_SHORT).show();

loadingBar.dismiss();

}

}

});

}

}

private void InitializeFields()

{

CreateAccountButton = (Button) findViewById(R.id.register\_button);

UserEmail = (EditText) findViewById(R.id.register\_email);

UserPassword = (EditText) findViewById(R.id.register\_password);

AlreadyHaveAccountLink = (TextView) findViewById(R.id.already\_have\_account\_link);

loadingBar = new ProgressDialog(this);

}

private void SendUserToLoginActivity()

{

Intent loginIntent = new Intent(RegisterActivity.this, LoginActivity.class);

startActivity(loginIntent);

}

private void SendUserToMainActivity()

{

Intent mainIntent = new Intent(RegisterActivity.this, MainActivity.class);

mainIntent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(mainIntent);

finish();

}

}

**Login.java:**

**package** com.example.myapp2;  
  
**import** androidx.annotation.NonNull;  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.app.ProgressDialog;  
**import** android.content.Intent;  
**import** android.icu.text.Collator;  
**import** android.os.Bundle;  
**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.android.gms.tasks.OnCompleteListener;  
**import** com.google.android.gms.tasks.Task;  
**import** com.google.firebase.auth.AuthResult;  
**import** com.google.firebase.auth.FirebaseAuth;  
**import** com.google.firebase.auth.FirebaseUser;  
**import** com.google.firebase.database.DatabaseReference;  
**import** com.google.firebase.database.FirebaseDatabase;  
**import** com.google.firebase.iid.FirebaseInstanceId;  
  
**public class** LoginActivity **extends** AppCompatActivity {  
  
**private** Button LoginButton, PhoneLoginButton;  
**private** EditText UserEmail, UserPassword;  
**private** TextView NeedNewAccountLink,ForgetPass;  
**private** TextView ForgetPasswordLink;  
**private** FirebaseUser currentUser;  
**private** ProgressDialog loadingBar;  
**private** FirebaseAuth mAuth;  
**private** DatabaseReference UsersRef;  
 @Override  
**protected void** onCreate(Bundle savedInstanceState) {  
**super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_login);  
  
 mAuth = FirebaseAuth.getInstance();  
 UsersRef = FirebaseDatabase.getInstance().getReference().child(**"Users"**);InitializeFields();  
 NeedNewAccountLink.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
**public void** onClick(View view) {  
 SendUserToregActivity();  
 }  
 });  
  
 ForgetPass.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
**public void** onClick(View view) {  
 SendUserToForgotPasswordActivity();  
 }  
 });  
  
 LoginButton.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
**public void** onClick(View view)  
 {  
 AllowUserToLogin();  
 }  
 });  
  
 PhoneLoginButton.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
**public void** onClick(View view)  
 {  
 Intent phoneLoginIntent = **new** Intent(LoginActivity.**this**, PhoneLoginActivity.**class**);  
 startActivity(phoneLoginIntent);  
 }  
 });  
 }  
  
**private void** AllowUserToLogin() {  
 String email = UserEmail.getText().toString();  
 String password = UserPassword.getText().toString();  
**if** (TextUtils.isEmpty(email))  
 {  
 Toast.makeText(**this**, **"Please enter email..."**, Toast.LENGTH\_SHORT).show();  
 }  
**else if** (TextUtils.isEmpty(password))  
 {  
 Toast.makeText(**this**, **"Please enter password..."**, Toast.LENGTH\_SHORT).show();  
 }  
**else**{  
 loadingBar.setTitle(**"Sign In"**);  
 loadingBar.setMessage(**"Please wait...."**);  
 loadingBar.setCanceledOnTouchOutside(**true**);  
 loadingBar.show();  
  
 mAuth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener(**new** OnCompleteListener<AuthResult>() {  
 @Override  
**public void** onComplete(@NonNull Task<AuthResult> task)  
 {  
**if** (task.isSuccessful())  
 {  
  
 String currentUserId = mAuth.getCurrentUser().getUid();  
**if** (task.isSuccessful())  
 {  
 SendUserToMainActivity();  
 Toast.makeText(LoginActivity.**this**, **"Logged in Successful..."**, Toast.LENGTH\_SHORT).show();  
 loadingBar.dismiss(); }}  
**else**{  
 String message = task.getException().toString();  
 Toast.makeText(LoginActivity.**this**, **"Error : "** + message, Toast.LENGTH\_SHORT).show();  
 loadingBar.dismiss();  
 }  
 }  
 });  
 }  
 }  
**private void** InitializeFields() {  
 LoginButton = (Button) findViewById(R.id.login\_button);  
 PhoneLoginButton = (Button) findViewById(R.id.phone\_login\_button);  
 UserEmail = (EditText) findViewById(R.id.login\_email);  
 UserPassword = (EditText) findViewById(R.id.login\_password);  
**NeedNewAccountLink** = (TextView) findViewById(R.id.***need\_new\_account\_link***);  
**ForgetPasswordLink** = (TextView) findViewById(R.id.***forget\_password\_link***);  
**loadingBar** = **new** ProgressDialog(**this**);  
**ForgetPass**=(TextView)findViewById(R.id.***forget\_password\_link***);  
 }  
  
**private void** SendUserToMainActivity()  
 {  
 Intent mainIntent = **new** Intent(LoginActivity.**this**, MainActivity.**class**);  
 mainIntent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
 startActivity(mainIntent);  
  
 finish();  
 }  
  
**private void** SendUserToregActivity() {  
 Intent regintent = **new** Intent(LoginActivity.**this**, RegisterActivity.**class**);  
 startActivity(regintent);  
 }  
  
**private void** SendUserToForgotPasswordActivity() {  
  
 Intent regintent = **new** Intent(LoginActivity.**this**, ForegetPassActivity.**class**);  
 startActivity(regintent);  
 }  
}

**CHAT\_ACTIVITY.JAVA**

package com.example.codingcafe.whatsapp;

import android.content.Context;

import android.support.annotation.NonNull;

import android.support.v7.app.ActionBar;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.support.v7.widget.Toolbar;

import android.text.TextUtils;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.TextView;

import android.widget.Toast;

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

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

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.ChildEventListener;

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.squareup.picasso.Picasso;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import de.hdodenhof.circleimageview.CircleImageView;

public class ChatActivity extends AppCompatActivity

{

private String messageReceiverID, messageReceiverName, messageReceiverImage, messageSenderID;

private TextView userName, userLastSeen;

private CircleImageView userImage;

private Toolbar ChatToolBar;

private FirebaseAuth mAuth;

private DatabaseReference RootRef;

private ImageButton SendMessageButton, SendFilesButton;

private EditText MessageInputText;

private final List<Messages> messagesList = new ArrayList<>();

private LinearLayoutManager linearLayoutManager;

private MessageAdapter messageAdapter;

private RecyclerView userMessagesList;

private String saveCurrentTime, saveCurrentDate;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_chat);

mAuth = FirebaseAuth.getInstance();

messageSenderID = mAuth.getCurrentUser().getUid();

RootRef = FirebaseDatabase.getInstance().getReference();

messageReceiverID = getIntent().getExtras().get("visit\_user\_id").toString();

messageReceiverName = getIntent().getExtras().get("visit\_user\_name").toString();

messageReceiverImage = getIntent().getExtras().get("visit\_image").toString();

IntializeControllers();

userName.setText(messageReceiverName);

Picasso.get().load(messageReceiverImage).placeholder(R.drawable.profile\_image).into(userImage);

SendMessageButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

SendMessage();

}

});

DisplayLastSeen();

}

private void IntializeControllers()

{

ChatToolBar = (Toolbar) findViewById(R.id.chat\_toolbar);

setSupportActionBar(ChatToolBar);

ActionBar actionBar = getSupportActionBar();

actionBar.setDisplayHomeAsUpEnabled(true);

actionBar.setDisplayShowCustomEnabled(true);

LayoutInflater layoutInflater = (LayoutInflater) this.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

View actionBarView = layoutInflater.inflate(R.layout.custom\_chat\_bar, null);

actionBar.setCustomView(actionBarView);

userName = (TextView) findViewById(R.id.custom\_profile\_name);

userLastSeen = (TextView) findViewById(R.id.custom\_user\_last\_seen);

userImage = (CircleImageView) findViewById(R.id.custom\_profile\_image);

SendMessageButton = (ImageButton) findViewById(R.id.send\_message\_btn);

SendFilesButton = (ImageButton) findViewById(R.id.send\_files\_btn);

MessageInputText = (EditText) findViewById(R.id.input\_message);

messageAdapter = new MessageAdapter(messagesList);

userMessagesList = (RecyclerView) findViewById(R.id.private\_messages\_list\_of\_users);

linearLayoutManager = new LinearLayoutManager(this);

userMessagesList.setLayoutManager(linearLayoutManager);

userMessagesList.setAdapter(messageAdapter);

Calendar calendar = Calendar.getInstance();

SimpleDateFormat currentDate = new SimpleDateFormat("MMM dd, yyyy");

saveCurrentDate = currentDate.format(calendar.getTime());

SimpleDateFormat currentTime = new SimpleDateFormat("hh:mm a");

saveCurrentTime = currentTime.format(calendar.getTime());

}

private void DisplayLastSeen()

{

RootRef.child("Users").child(messageReceiverID)

.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.child("userState").hasChild("state"))

{

String state = dataSnapshot.child("userState").child("state").getValue().toString();

String date = dataSnapshot.child("userState").child("date").getValue().toString();

String time = dataSnapshot.child("userState").child("time").getValue().toString();

if (state.equals("online"))

{

userLastSeen.setText("online");

}

else if (state.equals("offline"))

{

userLastSeen.setText("Last Seen: " + date + " " + time);

}

}

else

{

userLastSeen.setText("offline");

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

@Override

protected void onStart()

{

super.onStart();

RootRef.child("Messages").child(messageSenderID).child(messageReceiverID)

.addChildEventListener(new ChildEventListener() {

@Override

public void onChildAdded(DataSnapshot dataSnapshot, String s)

{

Messages messages = dataSnapshot.getValue(Messages.class);

messagesList.add(messages);

messageAdapter.notifyDataSetChanged();

userMessagesList.smoothScrollToPosition(userMessagesList.getAdapter().getItemCount());

}

@Override

public void onChildChanged(DataSnapshot dataSnapshot, String s) {

}

@Override

public void onChildRemoved(DataSnapshot dataSnapshot) {

}

@Override

public void onChildMoved(DataSnapshot dataSnapshot, String s) {

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

private void SendMessage()

{

String messageText = MessageInputText.getText().toString();

if (TextUtils.isEmpty(messageText))

{

Toast.makeText(this, "first write your message...", Toast.LENGTH\_SHORT).show();

}

else

{

String messageSenderRef = "Messages/" + messageSenderID + "/" + messageReceiverID;

String messageReceiverRef = "Messages/" + messageReceiverID + "/" + messageSenderID;

DatabaseReference userMessageKeyRef = RootRef.child("Messages")

.child(messageSenderID).child(messageReceiverID).push();

String messagePushID = userMessageKeyRef.getKey();

Map messageTextBody = new HashMap();

messageTextBody.put("message", messageText);

messageTextBody.put("type", "text");

messageTextBody.put("from", messageSenderID);

messageTextBody.put("to", messageReceiverID);

messageTextBody.put("messageID", messagePushID);

messageTextBody.put("time", saveCurrentTime);

messageTextBody.put("date", saveCurrentDate);

Map messageBodyDetails = new HashMap();

messageBodyDetails.put(messageSenderRef + "/" + messagePushID, messageTextBody);

messageBodyDetails.put( messageReceiverRef + "/" + messagePushID, messageTextBody);

RootRef.updateChildren(messageBodyDetails).addOnCompleteListener(new OnCompleteListener() {

@Override

public void onComplete(@NonNull Task task)

{

if (task.isSuccessful())

{

Toast.makeText(ChatActivity.this, "Message Sent Successfully...", Toast.LENGTH\_SHORT).show();

}

else

{

Toast.makeText(ChatActivity.this, "Error", Toast.LENGTH\_SHORT).show();

}

MessageInputText.setText("");

}

});

}

}

}

**CHATS\_FRAGMENT.JAVA**

package com.example.codingcafe.whatsapp;

import android.content.Intent;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v4.app.Fragment;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

import com.firebase.ui.database.FirebaseRecyclerAdapter;

import com.firebase.ui.database.FirebaseRecyclerOptions;

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.squareup.picasso.Picasso;

import de.hdodenhof.circleimageview.CircleImageView;

public class ChatsFragment extends Fragment

{

private View PrivateChatsView;

private RecyclerView chatsList;

private DatabaseReference ChatsRef, UsersRef;

private FirebaseAuth mAuth;

private String currentUserID="";

public ChatsFragment() {

// Required empty public constructor

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

PrivateChatsView = inflater.inflate(R.layout.fragment\_chats, container, false);

mAuth = FirebaseAuth.getInstance();

currentUserID = mAuth.getCurrentUser().getUid();

ChatsRef = FirebaseDatabase.getInstance().getReference().child("Contacts").child(currentUserID);

UsersRef = FirebaseDatabase.getInstance().getReference().child("Users");

chatsList = (RecyclerView) PrivateChatsView.findViewById(R.id.chats\_list);

chatsList.setLayoutManager(new LinearLayoutManager(getContext()));

return PrivateChatsView;

}

@Override

public void onStart()

{

super.onStart();

FirebaseRecyclerOptions<Contacts> options =

new FirebaseRecyclerOptions.Builder<Contacts>()

.setQuery(ChatsRef, Contacts.class)

.build();

FirebaseRecyclerAdapter<Contacts, ChatsViewHolder> adapter =

new FirebaseRecyclerAdapter<Contacts, ChatsViewHolder>(options) {

@Override

protected void onBindViewHolder(@NonNull final ChatsViewHolder holder, int position, @NonNull Contacts model)

{

final String usersIDs = getRef(position).getKey();

final String[] retImage = {"default\_image"};

UsersRef.child(usersIDs).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.exists())

{

if (dataSnapshot.hasChild("image"))

{

retImage[0] = dataSnapshot.child("image").getValue().toString();

Picasso.get().load(retImage[0]).into(holder.profileImage);

}

final String retName = dataSnapshot.child("name").getValue().toString();

final String retStatus = dataSnapshot.child("status").getValue().toString();

holder.userName.setText(retName)

if (dataSnapshot.child("userState").hasChild("state"))

{

String state = dataSnapshot.child("userState").child("state").getValue().toString();

String date = dataSnapshot.child("userState").child("date").getValue().toString();

String time = dataSnapshot.child("userState").child("time").getValue().toString();

if (state.equals("online"))

{

holder.userStatus.setText("online");

}

else if (state.equals("offline"))

{

holder.userStatus.setText("Last Seen: " + date + " " + time);

}

}

else

{

holder.userStatus.setText("offline");

}

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

Intent chatIntent = new Intent(getContext(), ChatActivity.class);

chatIntent.putExtra("visit\_user\_id", usersIDs);

chatIntent.putExtra("visit\_user\_name", retName);

chatIntent.putExtra("visit\_image", retImage[0]);

startActivity(chatIntent);

}

});

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

@NonNull

@Override

public ChatsViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i)

{

View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.users\_display\_layout, viewGroup, false);

return new ChatsViewHolder(view);

}

};

chatsList.setAdapter(adapter);

adapter.startListening();

}

public static class ChatsViewHolder extends RecyclerView.ViewHolder

{

CircleImageView profileImage;

TextView userStatus, userName;

public ChatsViewHolder(@NonNull View itemView)

{

super(itemView);

profileImage = itemView.findViewById(R.id.users\_profile\_image);

userStatus = itemView.findViewById(R.id.user\_status);

userName = itemView.findViewById(R.id.user\_profile\_name);

}

}

}

**CONTACTS.JAVA**

package com.example.codingcafe.whatsapp;

public class Contacts {

public String name, status, image;

public Contacts()

{

}

public Contacts(String name, String status, String image) {

this.name = name;

this.status = status;

this.image = image;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public String getImage() {

return image;

}

public void setImage(String image) {

this.image = image;

}

}

CONTACTS\_FRAGMENT.JAVA

package com.example.codingcafe.whatsapp;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v4.app.Fragment;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import com.firebase.ui.database.FirebaseRecyclerAdapter;

import com.firebase.ui.database.FirebaseRecyclerOptions;

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.squareup.picasso.Picasso;

import de.hdodenhof.circleimageview.CircleImageView;

public class ContactsFragment extends Fragment

{

private View ContactsView;

private RecyclerView myContactsList;

private DatabaseReference ContacsRef, UsersRef;

private FirebaseAuth mAuth;

private String currentUserID;

public ContactsFragment() {

// Required empty public constructor

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

ContactsView = inflater.inflate(R.layout.fragment\_contacts, container, false);

myContactsList = (RecyclerView) ContactsView.findViewById(R.id.contacts\_list);

myContactsList.setLayoutManager(new LinearLayoutManager(getContext()));

mAuth = FirebaseAuth.getInstance();

currentUserID = mAuth.getCurrentUser().getUid();

ContacsRef = FirebaseDatabase.getInstance().getReference().child("Contacts").child(currentUserID);

UsersRef = FirebaseDatabase.getInstance().getReference().child("Users");

return ContactsView;

}

@Override

public void onStart()

{

super.onStart();

FirebaseRecyclerOptions options =

new FirebaseRecyclerOptions.Builder<Contacts>()

.setQuery(ContacsRef, Contacts.class)

.build();

final FirebaseRecyclerAdapter<Contacts, ContactsViewHolder> adapter

= new FirebaseRecyclerAdapter<Contacts, ContactsViewHolder>(options) {

@Override

protected void onBindViewHolder(@NonNull final ContactsViewHolder holder, int position, @NonNull Contacts model)

{

final String userIDs = getRef(position).getKey();

UsersRef.child(userIDs).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.exists())

{

if (dataSnapshot.child("userState").hasChild("state"))

{

String state = dataSnapshot.child("userState").child("state").getValue().toString();

String date = dataSnapshot.child("userState").child("date").getValue().toString();

String time = dataSnapshot.child("userState").child("time").getValue().toString();

if (state.equals("online"))

{

holder.onlineIcon.setVisibility(View.VISIBLE);

}

else if (state.equals("offline"))

{

holder.onlineIcon.setVisibility(View.INVISIBLE);

}

}

else

{

holder.onlineIcon.setVisibility(View.INVISIBLE);

}

if (dataSnapshot.hasChild("image"))

{

String userImage = dataSnapshot.child("image").getValue().toString();

String profileName = dataSnapshot.child("name").getValue().toString();

String profileStatus = dataSnapshot.child("status").getValue().toString();

holder.userName.setText(profileName);

holder.userStatus.setText(profileStatus);

Picasso.get().load(userImage).placeholder(R.drawable.profile\_image).into(holder.profileImage);

}

else

{

String profileName = dataSnapshot.child("name").getValue().toString();

String profileStatus = dataSnapshot.child("status").getValue().toString();

holder.userName.setText(profileName);

holder.userStatus.setText(profileStatus);

}

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

@NonNull

@Override

public ContactsViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i)

{

View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.users\_display\_layout, viewGroup, false);

ContactsViewHolder viewHolder = new ContactsViewHolder(view);

return viewHolder;

}

};

myContactsList.setAdapter(adapter);

adapter.startListening();

}

public static class ContactsViewHolder extends RecyclerView.ViewHolder

{

TextView userName, userStatus;

CircleImageView profileImage;

ImageView onlineIcon;

public ContactsViewHolder(@NonNull View itemView)

{

super(itemView);

userName = itemView.findViewById(R.id.user\_profile\_name);

userStatus = itemView.findViewById(R.id.user\_status);

profileImage = itemView.findViewById(R.id.users\_profile\_image);

onlineIcon = (ImageView) itemView.findViewById(R.id.user\_online\_status);

}

}

}

**DEPENDENCIES:**

implementation 'com.android.support:appcompat-v7:28.0.0-rc01'

implementation 'com.android.support:design:28.0.0-rc01'

implementation 'com.android.support:support-v4:28.0.0-rc01'

implementation 'com.android.support.constraint:constraint-layout:1.1.2'

implementation 'com.google.firebase:firebase-auth:11.8.0'

implementation 'com.google.firebase:firebase-database:11.8.0'

implementation 'com.google.firebase:firebase-messaging:11.8.0'

implementation 'com.google.firebase:firebase-storage:11.8.0'

implementation 'com.firebaseui:firebase-ui-database:3.2.2'

implementation 'com.android.support:support-v4:28.0.0-rc01'

implementation 'de.hdodenhof:circleimageview:2.2.0'

implementation 'com.theartofdev.edmodo:android-image-cropper:2.7.+'

implementation 'com.squareup.picasso:picasso:2.71828'

**MAIN\_ACTIVITY.JAVA**

package com.example.codingcafe.whatsapp;

import android.content.DialogInterface;

import android.content.Intent;

import android.support.annotation.NonNull;

import android.support.design.widget.TabLayout;

import android.support.v4.view.ViewPager;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.Toolbar;

import android.text.TextUtils;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.EditText;

import android.widget.Toast;

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

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

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.HashMap;

public class MainActivity extends AppCompatActivity

{

private Toolbar mToolbar;

private ViewPager myViewPager;

private TabLayout myTabLayout;

private TabsAccessorAdapter myTabsAccessorAdapter;

private FirebaseUser currentUser;

private FirebaseAuth mAuth;

private DatabaseReference RootRef;

private String currentUserID;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mAuth = FirebaseAuth.getInstance();

currentUser = mAuth.getCurrentUser();

currentUserID = mAuth.getCurrentUser().getUid();

RootRef = FirebaseDatabase.getInstance().getReference();

mToolbar = (Toolbar) findViewById(R.id.main\_page\_toolbar);

setSupportActionBar(mToolbar);

getSupportActionBar().setTitle("WhatsApp")

myViewPager = (ViewPager) findViewById(R.id.main\_tabs\_pager);

myTabsAccessorAdapter = new TabsAccessorAdapter(getSupportFragmentManager());

myViewPager.setAdapter(myTabsAccessorAdapter);

myTabLayout = (TabLayout) findViewById(R.id.main\_tabs);

myTabLayout.setupWithViewPager(myViewPager);

}

@Override

protected void onStart()

{

super.onStart();

if (currentUser == null)

{

SendUserToLoginActivity();

}

else

{

updateUserStatus("online");

VerifyUserExistance();

}

}

@Override

protected void onStop()

{

super.onStop();

if (currentUser != null)

{

updateUserStatus("offline");

}

}

@Override

protected void onDestroy()

{

super.onDestroy();

if (currentUser != null)

{

updateUserStatus("offline");

}

}

private void VerifyUserExistance()

{

String currentUserID = mAuth.getCurrentUser().getUid();

RootRef.child("Users").child(currentUserID).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if ((dataSnapshot.child("name").exists()))

{

Toast.makeText(MainActivity.this, "Welcome", Toast.LENGTH\_SHORT).show();

}

else

{

SendUserToSettingsActivity();

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu)

{

super.onCreateOptionsMenu(menu);

getMenuInflater().inflate(R.menu.options\_menu, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item)

{

super.onOptionsItemSelected(item);

if (item.getItemId() == R.id.main\_logout\_option)

{

updateUserStatus("offline");

mAuth.signOut();

SendUserToLoginActivity();

}

if (item.getItemId() == R.id.main\_settings\_option)

{

SendUserToSettingsActivity();

}

if (item.getItemId() == R.id.main\_create\_group\_option)

{

RequestNewGroup();

}

if (item.getItemId() == R.id.main\_find\_friends\_option)

{

SendUserToFindFriendsActivity();

}

return true;

}

private void RequestNewGroup()

{

AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this, R.style.AlertDialog);

builder.setTitle("Enter Group Name :");

final EditText groupNameField = new EditText(MainActivity.this);

groupNameField.setHint("e.g Coding Cafe");

builder.setView(groupNameField);

builder.setPositiveButton("Create", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i)

{

String groupName = groupNameField.getText().toString();

if (TextUtils.isEmpty(groupName))

{

Toast.makeText(MainActivity.this, "Please write Group Name...", Toast.LENGTH\_SHORT).show();

}

else

{

CreateNewGroup(groupName);

}

}

});

builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i)

{

dialogInterface.cancel();

}

});

builder.show();

}

private void CreateNewGroup(final String groupName)

{

RootRef.child("Groups").child(groupName).setValue("")

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

Toast.makeText(MainActivity.this, groupName + " group is Created Successfully...", Toast.LENGTH\_SHORT).show();

}

}

});

}

private void SendUserToLoginActivity()

{

Intent loginIntent = new Intent(MainActivity.this, LoginActivity.class);

loginIntent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(loginIntent);

}

private void SendUserToSettingsActivity()

{

Intent settingsIntent = new Intent(MainActivity.this, SettingsActivity.class);

startActivity(settingsIntent);

}

private void SendUserToFindFriendsActivity()

{

Intent findFriendsIntent = new Intent(MainActivity.this, FindFriendsActivity.class);

startActivity(findFriendsIntent);

}

private void updateUserStatus(String state)

{

String saveCurrentTime, saveCurrentDate;

Calendar calendar = Calendar.getInstance();

SimpleDateFormat currentDate = new SimpleDateFormat("MMM dd, yyyy");

saveCurrentDate = currentDate.format(calendar.getTime());

SimpleDateFormat currentTime = new SimpleDateFormat("hh:mm a");

saveCurrentTime = currentTime.format(calendar.getTime());

HashMap<String, Object> onlineStateMap = new HashMap<>();

onlineStateMap.put("time", saveCurrentTime);

onlineStateMap.put("date", saveCurrentDate);

onlineStateMap.put("state", state);

RootRef.child("Users").child(currentUserID).child("userState")

.updateChildren(onlineStateMap);

}

}

**MESSAGE\_ADAPTER.JAVA**

package com.example.codingcafe.whatsapp;

import android.graphics.Color;

import android.support.annotation.NonNull;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

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.squareup.picasso.Picasso;

import java.util.List;

import de.hdodenhof.circleimageview.CircleImageView;

public class MessageAdapter extends RecyclerView.Adapter<MessageAdapter.MessageViewHolder>

{

private List<Messages> userMessagesList;

private FirebaseAuth mAuth;

private DatabaseReference usersRef;

public MessageAdapter (List<Messages> userMessagesList)

{

this.userMessagesList = userMessagesList;

}

public class MessageViewHolder extends RecyclerView.ViewHolder

{

public TextView senderMessageText, receiverMessageText;

public CircleImageView receiverProfileImage;

public ImageView messageSenderPicture, messageReceiverPicture;

public MessageViewHolder(@NonNull View itemView)

{

super(itemView);

senderMessageText = (TextView) itemView.findViewById(R.id.sender\_messsage\_text);

receiverMessageText = (TextView) itemView.findViewById(R.id.receiver\_message\_text);

receiverProfileImage = (CircleImageView) itemView.findViewById(R.id.message\_profile\_image);

messageReceiverPicture = itemView.findViewById(R.id.message\_receiver\_image\_view);

messageSenderPicture = itemView.findViewById(R.id.message\_sender\_image\_view);

}

}

@NonNull

@Override

public MessageViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i)

{

View view = LayoutInflater.from(viewGroup.getContext())

.inflate(R.layout.custom\_messages\_layout, viewGroup, false);

mAuth = FirebaseAuth.getInstance();

return new MessageViewHolder(view);

}

@Override

public void onBindViewHolder(@NonNull final MessageViewHolder messageViewHolder, int i)

{

String messageSenderId = mAuth.getCurrentUser().getUid();

Messages messages = userMessagesList.get(i);

String fromUserID = messages.getFrom();

String fromMessageType = messages.getType();

usersRef = FirebaseDatabase.getInstance().getReference().child("Users").child(fromUserID);

usersRef.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.hasChild("image"))

{

String receiverImage = dataSnapshot.child("image").getValue().toString();

Picasso.get().load(receiverImage).placeholder(R.drawable.profile\_image).into(messageViewHolder.receiverProfileImage);

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

messageViewHolder.receiverMessageText.setVisibility(View.GONE);

messageViewHolder.receiverProfileImage.setVisibility(View.GONE);

messageViewHolder.senderMessageText.setVisibility(View.GONE);

messageViewHolder.messageSenderPicture.setVisibility(View.GONE);

messageViewHolder.messageReceiverPicture.setVisibility(View.GONE);

if (fromMessageType.equals("text"))

{

if (fromUserID.equals(messageSenderId))

{

messageViewHolder.senderMessageText.setVisibility(View.VISIBLE);

messageViewHolder.senderMessageText.setBackgroundResource(R.drawable.sender\_messages\_layout);

messageViewHolder.senderMessageText.setTextColor(Color.BLACK);

messageViewHolder.senderMessageText.setText(messages.getMessage() + "\n \n" + messages.getTime() + " - " + messages.getDate());

}

else

{

messageViewHolder.receiverProfileImage.setVisibility(View.VISIBLE);

messageViewHolder.receiverMessageText.setVisibility(View.VISIBLE);

messageViewHolder.receiverMessageText.setBackgroundResource(R.drawable.receiver\_messages\_layout);

messageViewHolder.receiverMessageText.setTextColor(Color.BLACK);

messageViewHolder.receiverMessageText.setText(messages.getMessage() + "\n \n" + messages.getTime() + " - " + messages.getDate());

}

}

}

@Override

public int getItemCount()

{

return userMessagesList.size();

}

}

MESSAGES.JAVA

package com.example.codingcafe.whatsapp;

public class Messages

{

private String from, message, type, to, messageID, time, date, name;

public Messages()

{

}

public Messages(String from, String message, String type, String to, String messageID, String time, String date, String name) {

this.from = from;

this.message = message;

this.type = type;

this.to = to;

this.messageID = messageID;

this.time = time;

this.date = date;

this.name = name;

}

public String getFrom() {

return from;

}

public void setFrom(String from) {

this.from = from;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getTo() {

return to;

}

public void setTo(String to) {

this.to = to;

}

public String getMessageID() {

return messageID;

}

public void setMessageID(String messageID) {

this.messageID = messageID;

}

public String getTime() {

return time;

}

public void setTime(String time) {

this.time = time;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**PHONE\_LOG.JAVA**

package com.example.codingcafe.whatsapp;

import android.app.ProgressDialog;

import android.content.Intent;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

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

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

import com.google.firebase.FirebaseException;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;

import com.google.firebase.auth.PhoneAuthCredential;

import com.google.firebase.auth.PhoneAuthProvider;

import java.util.concurrent.TimeUnit;

public class PhoneLoginActivity extends AppCompatActivity

{

private Button SendVerificationCodeButton, VerifyButton;

private EditText InputPhoneNumber, InputVerificationCode;

private PhoneAuthProvider.OnVerificationStateChangedCallbacks callbacks;

private FirebaseAuth mAuth;

private ProgressDialog loadingBar;

private String mVerificationId;

private PhoneAuthProvider.ForceResendingToken mResendToken;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_phone\_login);

mAuth = FirebaseAuth.getInstance();

SendVerificationCodeButton = (Button) findViewById(R.id.send\_ver\_code\_button);

VerifyButton = (Button) findViewById(R.id.verify\_button);

InputPhoneNumber = (EditText) findViewById(R.id.phone\_nnumber\_input);

InputVerificationCode = (EditText) findViewById(R.id.verification\_code\_input);

loadingBar = new ProgressDialog(this);

SendVerificationCodeButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

String phoneNumber = InputPhoneNumber.getText().toString();

if (TextUtils.isEmpty(phoneNumber))

{

Toast.makeText(PhoneLoginActivity.this, "Please enter your phone number first...", Toast.LENGTH\_SHORT).show();

}

else

{

loadingBar.setTitle("Phone Verification");

loadingBar.setMessage("please wait, while we are authenticating your phone...");

loadingBar.setCanceledOnTouchOutside(false);

loadingBar.show();

PhoneAuthProvider.getInstance().verifyPhoneNumber(

phoneNumber, // Phone number to verify

60, // Timeout duration

TimeUnit.SECONDS, // Unit of timeout

PhoneLoginActivity.this, // Activity (for callback binding)

callbacks); // OnVerificationStateChangedCallbacks

}

}

});

VerifyButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

SendVerificationCodeButton.setVisibility(View.INVISIBLE);

InputPhoneNumber.setVisibility(View.INVISIBLE);

String verificationCode = InputVerificationCode.getText().toString();

if (TextUtils.isEmpty(verificationCode))

{

Toast.makeText(PhoneLoginActivity.this, "Please write verification code first...", Toast.LENGTH\_SHORT).show();

}

else

{

loadingBar.setTitle("Verification Code");

loadingBar.setMessage("please wait, while we are verifying verification code...");

loadingBar.setCanceledOnTouchOutside(false);

loadingBar.show();

PhoneAuthCredential credential = PhoneAuthProvider.getCredential(mVerificationId, verificationCode);

signInWithPhoneAuthCredential(credential);

}

}

});

callbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

@Override

public void onVerificationCompleted(PhoneAuthCredential phoneAuthCredential)

{

signInWithPhoneAuthCredential(phoneAuthCredential);

}

@Override

public void onVerificationFailed(FirebaseException e)

{

loadingBar.dismiss();

Toast.makeText(PhoneLoginActivity.this, "Invalid Phone Number, Please enter correct phone number with your country code...", Toast.LENGTH\_SHORT).show();

SendVerificationCodeButton.setVisibility(View.VISIBLE);

InputPhoneNumber.setVisibility(View.VISIBLE);

VerifyButton.setVisibility(View.INVISIBLE);

InputVerificationCode.setVisibility(View.INVISIBLE);

}

public void onCodeSent(String verificationId,

PhoneAuthProvider.ForceResendingToken token)

{

// Save verification ID and resending token so we can use them later

mVerificationId = verificationId;

mResendToken = token;

loadingBar.dismiss();

Toast.makeText(PhoneLoginActivity.this, "Code has been sent, please check and verify...", Toast.LENGTH\_SHORT).show();

SendVerificationCodeButton.setVisibility(View.INVISIBLE);

InputPhoneNumber.setVisibility(View.INVISIBLE);

VerifyButton.setVisibility(View.VISIBLE);

InputVerificationCode.setVisibility(View.VISIBLE);

}

};

}

private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {

mAuth.signInWithCredential(credential)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

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

if (task.isSuccessful())

{

loadingBar.dismiss();

Toast.makeText(PhoneLoginActivity.this, "Congratulations, you're logged in successfully...", Toast.LENGTH\_SHORT).show();

SendUserToMainActivity();

}

else

{

String message = task.getException().toString();

Toast.makeText(PhoneLoginActivity.this, "Error : " + message, Toast.LENGTH\_SHORT).show();

}

}

});

}

private void SendUserToMainActivity()

{

Intent mainIntent = new Intent(PhoneLoginActivity.this, MainActivity.class);

startActivity(mainIntent);

finish();

}

}

**PROFILE.JAVA**

package com.example.codingcafe.whatsapp;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

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

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

import com.google.firebase.auth.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.squareup.picasso.Picasso;

import java.util.HashMap;

import de.hdodenhof.circleimageview.CircleImageView;

public class ProfileActivity extends AppCompatActivity

{

private String receiverUserID, senderUserID, Current\_State;

private CircleImageView userProfileImage;

private TextView userProfileName, userProfileStatus;

private Button SendMessageRequestButton, DeclineMessageRequestButton;

private DatabaseReference UserRef, ChatRequestRef, ContactsRef, NotificationRef;

private FirebaseAuth mAuth;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_profile);

mAuth = FirebaseAuth.getInstance();

UserRef = FirebaseDatabase.getInstance().getReference().child("Users");

ChatRequestRef = FirebaseDatabase.getInstance().getReference().child("Chat Requests");

ContactsRef = FirebaseDatabase.getInstance().getReference().child("Contacts");

NotificationRef = FirebaseDatabase.getInstance().getReference().child("Notifications");

receiverUserID = getIntent().getExtras().get("visit\_user\_id").toString();

senderUserID = mAuth.getCurrentUser().getUid();

userProfileImage = (CircleImageView) findViewById(R.id.visit\_profile\_image);

userProfileName = (TextView) findViewById(R.id.visit\_user\_name);

userProfileStatus = (TextView) findViewById(R.id.visit\_profile\_status);

SendMessageRequestButton = (Button) findViewById(R.id.send\_message\_request\_button);

DeclineMessageRequestButton = (Button) findViewById(R.id.decline\_message\_request\_button);

Current\_State = "new";

RetrieveUserInfo();

}

private void RetrieveUserInfo()

{

UserRef.child(receiverUserID).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if ((dataSnapshot.exists()) && (dataSnapshot.hasChild("image")))

{

String userImage = dataSnapshot.child("image").getValue().toString();

String userName = dataSnapshot.child("name").getValue().toString();

String userstatus = dataSnapshot.child("status").getValue().toString();

Picasso.get().load(userImage).placeholder(R.drawable.profile\_image).into(userProfileImage);

userProfileName.setText(userName);

userProfileStatus.setText(userstatus);

ManageChatRequests();

}

else

{

String userName = dataSnapshot.child("name").getValue().toString();

String userstatus = dataSnapshot.child("status").getValue().toString();

userProfileName.setText(userName);

userProfileStatus.setText(userstatus);

ManageChatRequests();

}

}

@Override

public void onCancelled(DatabaseError databaseError) ;

}

});

}

private void ManageChatRequests()

{

ChatRequestRef.child(senderUserID)

.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.hasChild(receiverUserID))

{

String request\_type = dataSnapshot.child(receiverUserID).child("request\_type").getValue().toString();

if (request\_type.equals("sent"))

{

Current\_State = "request\_sent";

SendMessageRequestButton.setText("Cancel Chat Request");

}

else if (request\_type.equals("received"))

{

Current\_State = "request\_received";

SendMessageRequestButton.setText("Accept Chat Request");

DeclineMessageRequestButton.setVisibility(View.VISIBLE);

DeclineMessageRequestButton.setEnabled(true);

DeclineMessageRequestButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

CancelChatRequest();

}

});

}

}

else

{

ContactsRef.child(senderUserID)

.addListenerForSingleValueEvent(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.hasChild(receiverUserID))

{

Current\_State = "friends";

SendMessageRequestButton.setText("Remove this Contact");

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

if (!senderUserID.equals(receiverUserID))

{

SendMessageRequestButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

SendMessageRequestButton.setEnabled(false);

if (Current\_State.equals("new"))

{

SendChatRequest();

}

if (Current\_State.equals("request\_sent"))

{

CancelChatRequest();

}

if (Current\_State.equals("request\_received"))

{

AcceptChatRequest();

}

if (Current\_State.equals("friends"))

{

RemoveSpecificContact();

}

}

});

}

else

{

SendMessageRequestButton.setVisibility(View.INVISIBLE);

}

}

private void RemoveSpecificContact()

{

ContactsRef.child(senderUserID).child(receiverUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ContactsRef.child(receiverUserID).child(senderUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

SendMessageRequestButton.setEnabled(true);

Current\_State = "new";

SendMessageRequestButton.setText("Send Message");

DeclineMessageRequestButton.setVisibility(View.INVISIBLE);

DeclineMessageRequestButton.setEnabled(false);

}

}

});

}

}

});

}

private void AcceptChatRequest()

{

ContactsRef.child(senderUserID).child(receiverUserID)

.child("Contacts").setValue("Saved")

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ContactsRef.child(receiverUserID).child(senderUserID)

.child("Contacts").setValue("Saved")

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestRef.child(senderUserID).child(receiverUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestRef.child(receiverUserID).child(senderUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

SendMessageRequestButton.setEnabled(true);

Current\_State = "friends";

SendMessageRequestButton.setText("Remove this Contact");

DeclineMessageRequestButton.setVisibility(View.INVISIBLE);

DeclineMessageRequestButton.setEnabled(false);

}

});

}

}

});

}

}

});

}

}

});

}

private void CancelChatRequest()

{

ChatRequestRef.child(senderUserID).child(receiverUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestRef.child(receiverUserID).child(senderUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

SendMessageRequestButton.setEnabled(true);

Current\_State = "new";

SendMessageRequestButton.setText("Send Message");

DeclineMessageRequestButton.setVisibility(View.INVISIBLE);

DeclineMessageRequestButton.setEnabled(false);

}

}

});

}

}

});

}

private void SendChatRequest()

{

ChatRequestRef.child(senderUserID).child(receiverUserID)

.child("request\_type").setValue("sent")

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestRef.child(receiverUserID).child(senderUserID)

.child("request\_type").setValue("received")

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

HashMap<String, String> chatNotificationMap = new HashMap<>();

chatNotificationMap.put("from", senderUserID);

chatNotificationMap.put("type", "request");

NotificationRef.child(receiverUserID).push()

.setValue(chatNotificationMap)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

SendMessageRequestButton.setEnabled(true);

Current\_State = "request\_sent";

SendMessageRequestButton.setText("Cancel Chat Request");

}

}

});

}

}

});

}

}

});

}

}

**REQUEST\_FRAGMENT.JAVA**

package com.example.codingcafe.whatsapp;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.support.annotation.NonNull;

import android.support.v4.app.Fragment;

import android.support.v7.widget.LinearLayoutManager;

import android.support.v7.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import com.firebase.ui.database.FirebaseRecyclerAdapter;

import com.firebase.ui.database.FirebaseRecyclerOptions;

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

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

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.squareup.picasso.Picasso;

import de.hdodenhof.circleimageview.CircleImageView;

public class RequestsFragment extends Fragment

{

private View RequestsFragmentView;

private RecyclerView myRequestsList;

private DatabaseReference ChatRequestsRef, UsersRef, ContactsRef;

private FirebaseAuth mAuth;

private String currentUserID;

public RequestsFragment() {

// Required empty public constructor

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

RequestsFragmentView = inflater.inflate(R.layout.fragment\_requests, container, false);

mAuth = FirebaseAuth.getInstance();

currentUserID = mAuth.getCurrentUser().getUid();

UsersRef = FirebaseDatabase.getInstance().getReference().child("Users");

ChatRequestsRef = FirebaseDatabase.getInstance().getReference().child("Chat Requests");

ContactsRef = FirebaseDatabase.getInstance().getReference().child("Contacts");

myRequestsList = (RecyclerView) RequestsFragmentView.findViewById(R.id.chat\_requests\_list);

myRequestsList.setLayoutManager(new LinearLayoutManager(getContext()));

return RequestsFragmentView;

}

@Override

public void onStart()

{

super.onStart();

FirebaseRecyclerOptions<Contacts> options =

new FirebaseRecyclerOptions.Builder<Contacts>()

.setQuery(ChatRequestsRef.child(currentUserID), Contacts.class)

.build();

FirebaseRecyclerAdapter<Contacts, RequestsViewHolder> adapter =

new FirebaseRecyclerAdapter<Contacts, RequestsViewHolder>(options) {

@Override

protected void onBindViewHolder(@NonNull final RequestsViewHolder holder, int position, @NonNull Contacts model)

{

holder.itemView.findViewById(R.id.request\_accept\_btn).setVisibility(View.VISIBLE);

holder.itemView.findViewById(R.id.request\_cancel\_btn).setVisibility(View.VISIBLE);

final String list\_user\_id = getRef(position).getKey();

DatabaseReference getTypeRef = getRef(position).child("request\_type").getRef();

getTypeRef.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.exists())

{

String type = dataSnapshot.getValue().toString();

if (type.equals("received"))

{

UsersRef.child(list\_user\_id).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.hasChild("image"))

{

final String requestProfileImage = dataSnapshot.child("image").getValue().toString();

Picasso.get().load(requestProfileImage).into(holder.profileImage);

}

final String requestUserName = dataSnapshot.child("name").getValue().toString();

final String requestUserStatus = dataSnapshot.child("status").getValue().toString();

holder.userName.setText(requestUserName);

holder.userStatus.setText("wants to connect with you.");

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

CharSequence options[] = new CharSequence[]

{

"Accept",

"Cancel"

};

AlertDialog.Builder builder = new AlertDialog.Builder(getContext());

builder.setTitle(requestUserName + " Chat Request");

builder.setItems(options, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i)

{

if (i == 0)

{

ContactsRef.child(currentUserID).child(list\_user\_id).child("Contact")

.setValue("Saved").addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ContactsRef.child(list\_user\_id).child(currentUserID).child("Contact")

.setValue("Saved").addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestsRef.child(currentUserID).child(list\_user\_id)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestsRef.child(list\_user\_id).child(currentUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

Toast.makeText(getContext(), "New Contact Saved", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

});

}

}

});

}

}

});

}

if (i == 1)

{

ChatRequestsRef.child(currentUserID).child(list\_user\_id)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestsRef.child(list\_user\_id).child(currentUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

Toast.makeText(getContext(), "Contact Deleted", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

});

}

}

});

builder.show();

}

});

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

else if (type.equals("sent"))

{

Button request\_sent\_btn = holder.itemView.findViewById(R.id.request\_accept\_btn);

request\_sent\_btn.setText("Req Sent");

holder.itemView.findViewById(R.id.request\_cancel\_btn).setVisibility(View.INVISIBLE);

UsersRef.child(list\_user\_id).addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if (dataSnapshot.hasChild("image"))

{

final String requestProfileImage = dataSnapshot.child("image").getValue().toString();

Picasso.get().load(requestProfileImage).into(holder.profileImage);

}

final String requestUserName = dataSnapshot.child("name").getValue().toString();

final String requestUserStatus = dataSnapshot.child("status").getValue().toString();

holder.userName.setText(requestUserName);

holder.userStatus.setText("you have sent a request to " + requestUserName);

holder.itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

CharSequence options[] = new CharSequence[]

{

"Cancel Chat Request"

};

AlertDialog.Builder builder = new AlertDialog.Builder(getContext());

builder.setTitle("Already Sent Request");

builder.setItems(options, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialogInterface, int i)

{

if (i == 0)

{

ChatRequestsRef.child(currentUserID).child(list\_user\_id)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

ChatRequestsRef.child(list\_user\_id).child(currentUserID)

.removeValue()

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

Toast.makeText(getContext(), "you have cancelled the chat request.", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

});

}

}

});

builder.show();

}

});

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

@NonNull

@Override

public RequestsViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i)

{

View view = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.users\_display\_layout, viewGroup, false);

RequestsViewHolder holder = new RequestsViewHolder(view);

return holder;

}

};

myRequestsList.setAdapter(adapter);

adapter.startListening();

}

public static class RequestsViewHolder extends RecyclerView.ViewHolder

{

TextView userName, userStatus;

CircleImageView profileImage;

Button AcceptButton, CancelButton;

public RequestsViewHolder(@NonNull View itemView)

{

super(itemView);

userName = itemView.findViewById(R.id.user\_profile\_name);

userStatus = itemView.findViewById(R.id.user\_status);

profileImage = itemView.findViewById(R.id.users\_profile\_image);

AcceptButton = itemView.findViewById(R.id.request\_accept\_btn);

CancelButton = itemView.findViewById(R.id.request\_cancel\_btn);

}

}

}

**SETTINGS.JAVA**

package com.example.codingcafe.whatsapp;

import android.app.ProgressDialog;

import android.content.Intent;

import android.net.Uri;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.support.v7.widget.Toolbar;

import android.text.TextUtils;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

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

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

import com.google.firebase.auth.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.storage.FirebaseStorage;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.UploadTask;

import com.squareup.picasso.Picasso;

import com.theartofdev.edmodo.cropper.CropImage;

import com.theartofdev.edmodo.cropper.CropImageView;

import java.util.HashMap;

import de.hdodenhof.circleimageview.CircleImageView;

public class SettingsActivity extends AppCompatActivity

{

private Button UpdateAccountSettings;

private EditText userName, userStatus;

private CircleImageView userProfileImage;

private String currentUserID;

private FirebaseAuth mAuth;

private DatabaseReference RootRef;

private static final int GalleryPick = 1;

private StorageReference UserProfileImagesRef;

private ProgressDialog loadingBar;

private Toolbar SettingsToolBar;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_settings);

mAuth = FirebaseAuth.getInstance();

currentUserID = mAuth.getCurrentUser().getUid();

RootRef = FirebaseDatabase.getInstance().getReference();

UserProfileImagesRef = FirebaseStorage.getInstance().getReference().child("Profile Images");

InitializeFields();

userName.setVisibility(View.INVISIBLE);

UpdateAccountSettings.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

UpdateSettings();

}

});

RetrieveUserInfo();

userProfileImage.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view)

{

Intent galleryIntent = new Intent();

galleryIntent.setAction(Intent.ACTION\_GET\_CONTENT);

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

startActivityForResult(galleryIntent, GalleryPick);

}

});

}

private void InitializeFields()

{

UpdateAccountSettings = (Button) findViewById(R.id.update\_settings\_button);

userName = (EditText) findViewById(R.id.set\_user\_name);

userStatus = (EditText) findViewById(R.id.set\_profile\_status);

userProfileImage = (CircleImageView) findViewById(R.id.set\_profile\_image);

loadingBar = new ProgressDialog(this);

SettingsToolBar = (Toolbar) findViewById(R.id.settings\_toolbar);

setSupportActionBar(SettingsToolBar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setDisplayShowCustomEnabled(true);

getSupportActionBar().setTitle("Account Settings");

}

@Override

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)

{

super.onActivityResult(requestCode, resultCode, data);

if (requestCode==GalleryPick && resultCode==RESULT\_OK && data!=null)

{

Uri ImageUri = data.getData();

CropImage.activity()

.setGuidelines(CropImageView.Guidelines.ON)

.setAspectRatio(1, 1)

.start(this);

}

if (requestCode == CropImage.CROP\_IMAGE\_ACTIVITY\_REQUEST\_CODE)

{

CropImage.ActivityResult result = CropImage.getActivityResult(data);

if (resultCode == RESULT\_OK)

{

loadingBar.setTitle("Set Profile Image");

loadingBar.setMessage("Please wait, your profile image is updating...");

loadingBar.setCanceledOnTouchOutside(false);

loadingBar.show();

Uri resultUri = result.getUri();

StorageReference filePath = UserProfileImagesRef.child(currentUserID + ".jpg");

filePath.putFile(resultUri).addOnCompleteListener(new OnCompleteListener<UploadTask.TaskSnapshot>() {

@Override

public void onComplete(@NonNull Task<UploadTask.TaskSnapshot> task)

{

if (task.isSuccessful())

{

Toast.makeText(SettingsActivity.this, "Profile Image uploaded Successfully...", Toast.LENGTH\_SHORT).show();

final String downloaedUrl = task.getResult().getDownloadUrl().toString();

RootRef.child("Users").child(currentUserID).child("image")

.setValue(downloaedUrl)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

Toast.makeText(SettingsActivity.this, "Image save in Database, Successfully...", Toast.LENGTH\_SHORT).show();

loadingBar.dismiss();

}

else

{

String message = task.getException().toString();

Toast.makeText(SettingsActivity.this, "Error: " + message, Toast.LENGTH\_SHORT).show();

loadingBar.dismiss();

}

}

});

}

else

{

String message = task.getException().toString();

Toast.makeText(SettingsActivity.this, "Error: " + message, Toast.LENGTH\_SHORT).show();

loadingBar.dismiss();

}

}

});

}

}

}

private void UpdateSettings()

{

String setUserName = userName.getText().toString();

String setStatus = userStatus.getText().toString();

if (TextUtils.isEmpty(setUserName))

{

Toast.makeText(this, "Please write your user name first....", Toast.LENGTH\_SHORT).show();

}

if (TextUtils.isEmpty(setStatus))

{

Toast.makeText(this, "Please write your status....", Toast.LENGTH\_SHORT).show();

}

else

{

HashMap<String, Object> profileMap = new HashMap<>();

profileMap.put("uid", currentUserID);

profileMap.put("name", setUserName);

profileMap.put("status", setStatus);

RootRef.child("Users").child(currentUserID).updateChildren(profileMap)

.addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task)

{

if (task.isSuccessful())

{

SendUserToMainActivity();

Toast.makeText(SettingsActivity.this, "Profile Updated Successfully...", Toast.LENGTH\_SHORT).show();

}

else

{

String message = task.getException().toString();

Toast.makeText(SettingsActivity.this, "Error: " + message, Toast.LENGTH\_SHORT).show();

}

}

});

}

}

private void RetrieveUserInfo()

{

RootRef.child("Users").child(currentUserID)

.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot)

{

if ((dataSnapshot.exists()) && (dataSnapshot.hasChild("name") && (dataSnapshot.hasChild("image"))))

{

String retrieveUserName = dataSnapshot.child("name").getValue().toString();

String retrievesStatus = dataSnapshot.child("status").getValue().toString();

String retrieveProfileImage = dataSnapshot.child("image").getValue().toString();

userName.setText(retrieveUserName);

userStatus.setText(retrievesStatus);

Picasso.get().load(retrieveProfileImage).into(userProfileImage);

}

else if ((dataSnapshot.exists()) && (dataSnapshot.hasChild("name")))

{

String retrieveUserName = dataSnapshot.child("name").getValue().toString();

String retrievesStatus = dataSnapshot.child("status").getValue().toString();

userName.setText(retrieveUserName);

userStatus.setText(retrievesStatus);

}

else

{

userName.setVisibility(View.VISIBLE);

Toast.makeText(SettingsActivity.this, "Please set & update your profile information...", Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

}

private void SendUserToMainActivity()

{

Intent mainIntent = new Intent(SettingsActivity.this, MainActivity.class);

mainIntent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

startActivity(mainIntent);

finish();

}

}

**TAB\_ACCESSOR.JAVA**

package com.example.codingcafe.whatsapp;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.support.v4.app.FragmentManager;

import android.support.v4.app.FragmentPagerAdapter;

public class TabsAccessorAdapter extends FragmentPagerAdapter

{

public TabsAccessorAdapter(FragmentManager fm)

{

super(fm);

}

@Override

public Fragment getItem(int i)

{

switch (i)

{

case 0:

ChatsFragment chatsFragment = new ChatsFragment();

return chatsFragment;

case 1:

GroupsFragment groupsFragment = new GroupsFragment();

return groupsFragment;

case 2:

ContactsFragment contactsFragment = new ContactsFragment();

return contactsFragment;

case 3:

RequestsFragment requestsFragment = new RequestsFragment();

return requestsFragment;

default:

return null;

}

}

@Override

public int getCount()

{

return 4;

}

@Nullable

@Override

public CharSequence getPageTitle(int position)

{

switch (position)

{

case 0:

return "Chats";

case 1:

return "Groups";

case 2:

return "Contacts";

case 3:

return "Requests";

default:

return null;

}

}

}

**5.SYSTEM EVALUATION**

**5.1 Software Testing:**

The primary and larger objective of testing is to deliver quality software. Quality software is one that is devoid of error and meets with customer’s stated requirements.

If errors are found, then the software must be debugged to locate these errors in the various parts of the program. Corrections are then made. The program/system must be tested once again after corrections have been implemented – this time with additional objective of finding out whether or not corrections in one part of the system have introduced any new errors elsewhere in the system.

Once all errors are found, then another objective must be accomplished that is to check whether or not the system is doing what it is supposed to do. So another aspect of testing is that it must also ensure that the system meets with user requirements.

Here are some of the testing techniques.

**1.** Unit Testing

**2.** Integration Testing

**3.** Module Testing

**4.** Subsystem Testing

**5.** Black Box Testing

**6.** White Box Testing

**7.** Equivalence Partitioning

**8.** Ad-hoc Testing

**9.** Boundary Value Analysis

The Testing Technique used in this project is as follows.

**1. Unit Testing**

Unit testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of sources code assigned areas. The developers use test data of the quality assurance team. Unit testing is performed on each of the modules like top trending tweets,retweets,user information etc.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

**2. Integration Testing**

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing bottom-up Integration testing and Top down Integration testing.

**Bottom-up**

1. This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.

**Top-Down**

1. This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests in the complete application, preferably in scenarios designed to mimic those it will encounter in customer’s computers, systems and network. Integration testing is performed on user information module to check whether each of the components work in proper manner when they are combined.

**3. Module Testing**

Module testing is the testing of complete code objects as produced by the compiler when built from source. It is a collection of independent components such as an object class, an abstract data-type or some loser collection of procedures and functions. A module encapsulates related components so can be tested without other modules. Module testing is done on pie chart, histogram etc.

**4. Sub System Testing**

This phase of testing involves collecting of modules, which have been integrated into subsystems. Subsystems may be independently designed and implemented. The most common problems, which arise in the large software systems, are sub system interface mismatches. The subsystem test process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

**5. System testing**

System testing is the testing of behaviour of a complete and fully integrated software product based on the software requirements specification (SRS) document. In main focus of this testing is to evaluate functional end user requirements.

**5.2 Test Cases**

**POSITIVE TESTING OF LOGIN FUNCTIONALITY:**

Testing for **Login** Functionality with valid email and password.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Title** | **Description** | **Input Data** | **TestCase Steps** | **Expected Result** | **Actual Result** | **Status** |
| 1 | User  Signin | Testing Signin functionality of User successful Sigin | Valid Email  Valid Password | 1.Open the app and enter  2.Type valid email id in email section  3.Enter Valid Password  4.Click on Signin | Home screen | Siggned in to app, Home Screen visible | Test Case Passed |

Testing under **Signin** functionality with Valid Email and Incorrect Password.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Title** | **Description** | **Input Data** | **TestCase Steps** | **Expected Result** | **Actual Result** | **Status** |
| 2 | User | Testing Signin functionality of User for unsuccessful Sigin | Valid Email  Invalid Password | 1.Open the app  2.Type valid email id in email section  3.Enter Invalid Password  4.Click Signin | Display Error Invalid Password | Display Error | Test Case Passed |

**NEGATIVE TESTING OF LOGIN FUNCTIONALITY**

Testing under **Signin** functionality with Invalid Email and Password.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Title** | **Description** | **Input Data** | **Test Case Steps** | **Expected Result** | **Actual Result** | **Status** |
| 3 | User Signin | Testing Signin functionality of User for unsuccessful signin with blank Email and blank Password | Blank  Email  Blank Password | 1.Open the app  2.Type blank email id in email section | Display Error  Enter Valid Id | Display Error | Test Case Passed |

**POSITIVE TESTING OF REGISTRATION FUNCTIONALITY:**

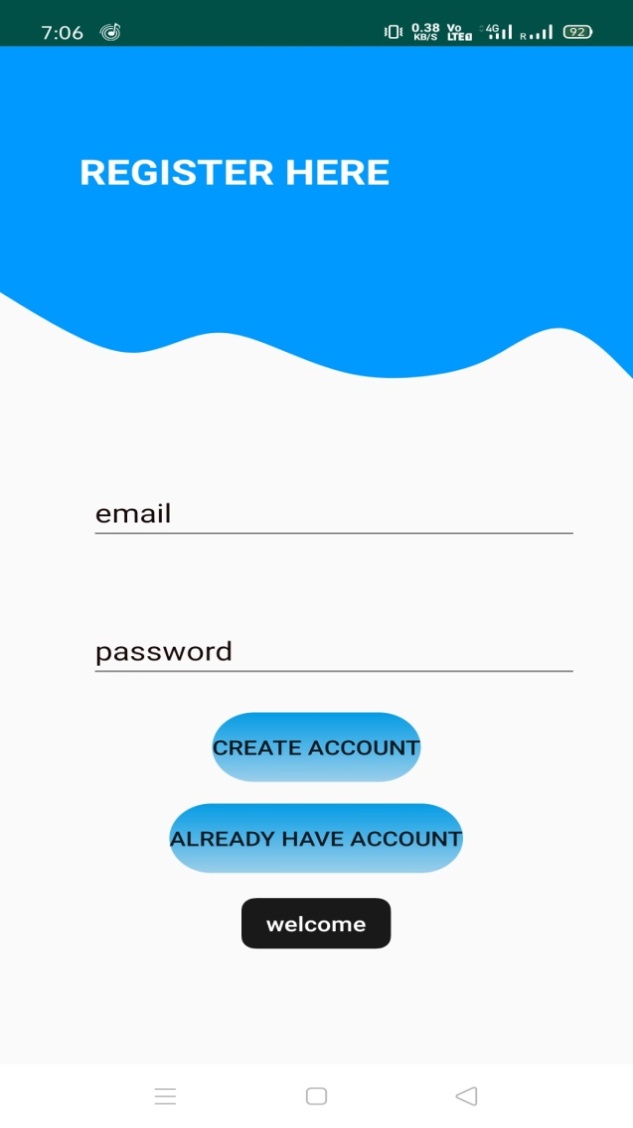
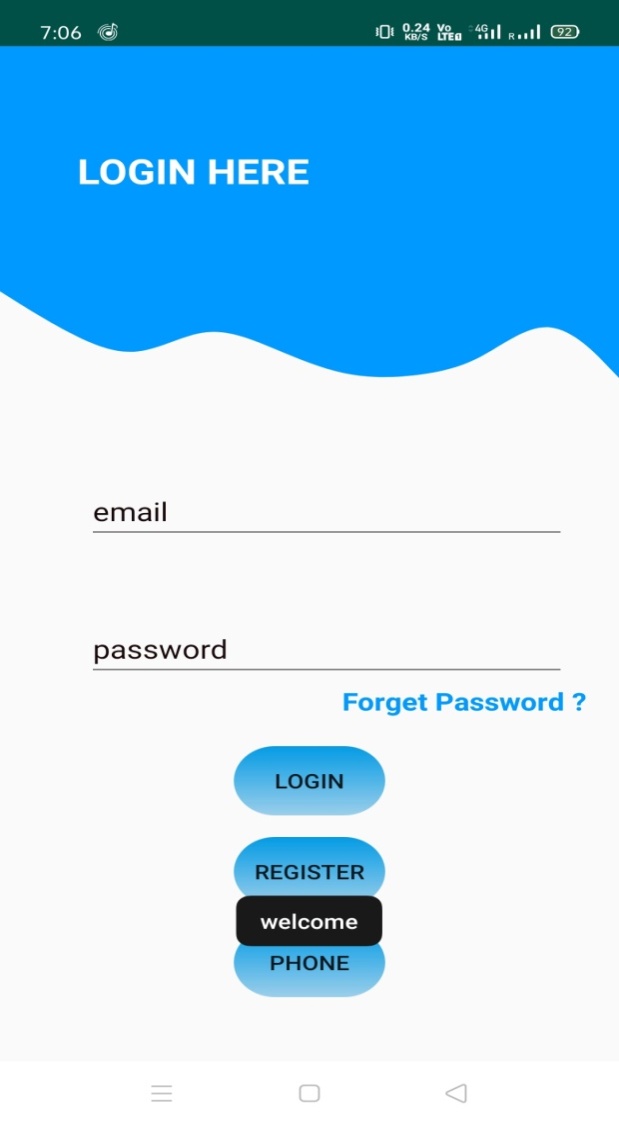
Testing under **Registration/Signup**  functionality with following details in **Aural app**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test**  **Case ID** | **Test**  **Title** | **Description** | **Input Data** | **TestCase Steps** | **Expected Result** | **Actual Result** | **Status** |
| 4 | User registration /Signup | Testing Register  functionality of User  for successful register | Valid Email, Usernam,Password | 1.Open the app .  2.On the Signin Page click on the “don’t have an account create here”  3.Type valid Email,  Password. | Users email Id and verified,  Redirect to Home screen | User registered/Signup inside the app | Test Case Passed |

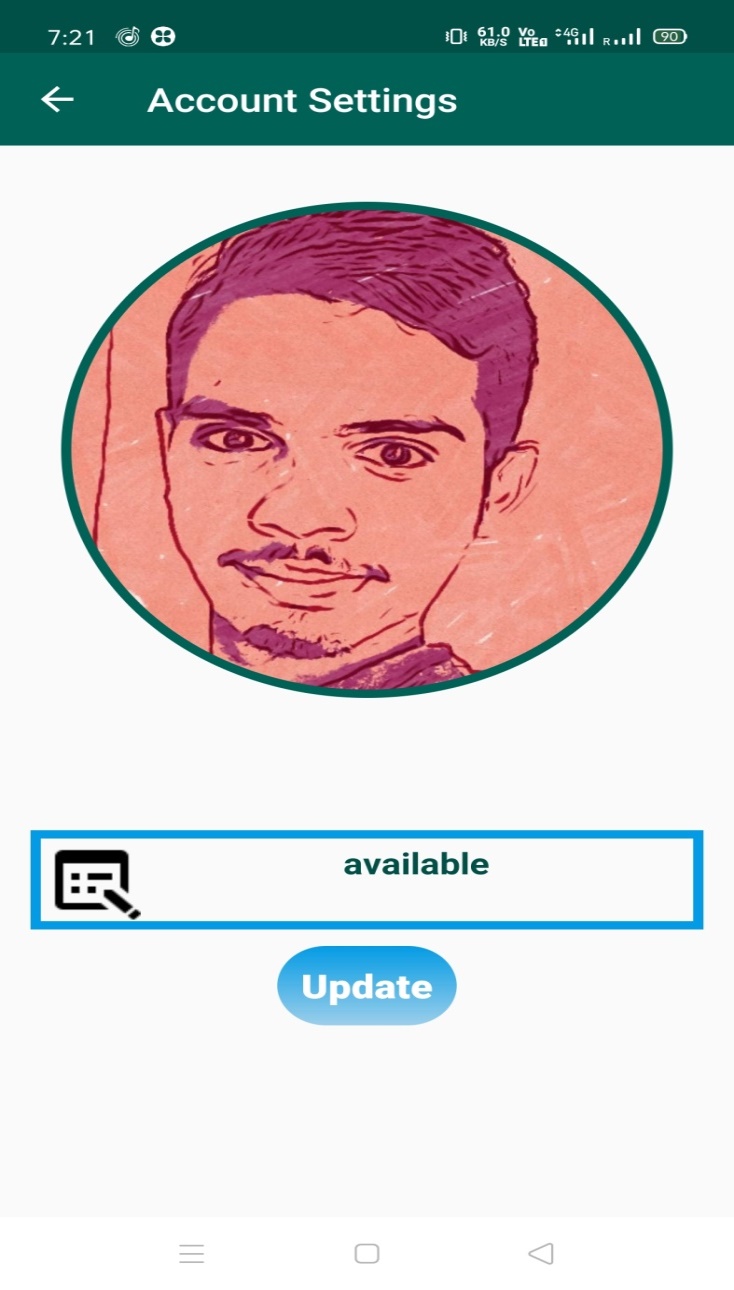
Positive Testing for Aural App

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Module** | **Sub Module** | **Input Data** | **Expected Result** | **Actual Result** | **Status** |
| 1. | PDF Module | Selecting File | Valid PDF file | Display selected file name | PDF file name displayed | Pass |
| 2. | Register/Signup Module | User register/  Signup | Email, Password | After click on “Signup”  Redirect to Home | User is registered | Pass |
|  |  | Sign in | Click on “Already have an account” | Directed to Signin page | Signin page | Pass |
| 3. | Speed Rate | Audio speed | Select the value in range | Audio speed increased. | Speed increased | Pass |
| 4. | Page Module | Page Number | Enter the page number | Application speak out the selected page. | Speaks the selected page | Pass |
| 5 | Listen Module | Start speaking | Select the PDF, set the rate, enter the page no. | Application reads the PDF file | Able to listen the audio | Pass |
| 6 | Stop Module | Stop speaking | Click on the Stop button | Audio stops | Application stopped reading | Pass |
| 7 | Sign-out Module | Sign-out from App | Click on the Signout button | Signout from the current account and directed at login page | Signout is Successful | Pass |

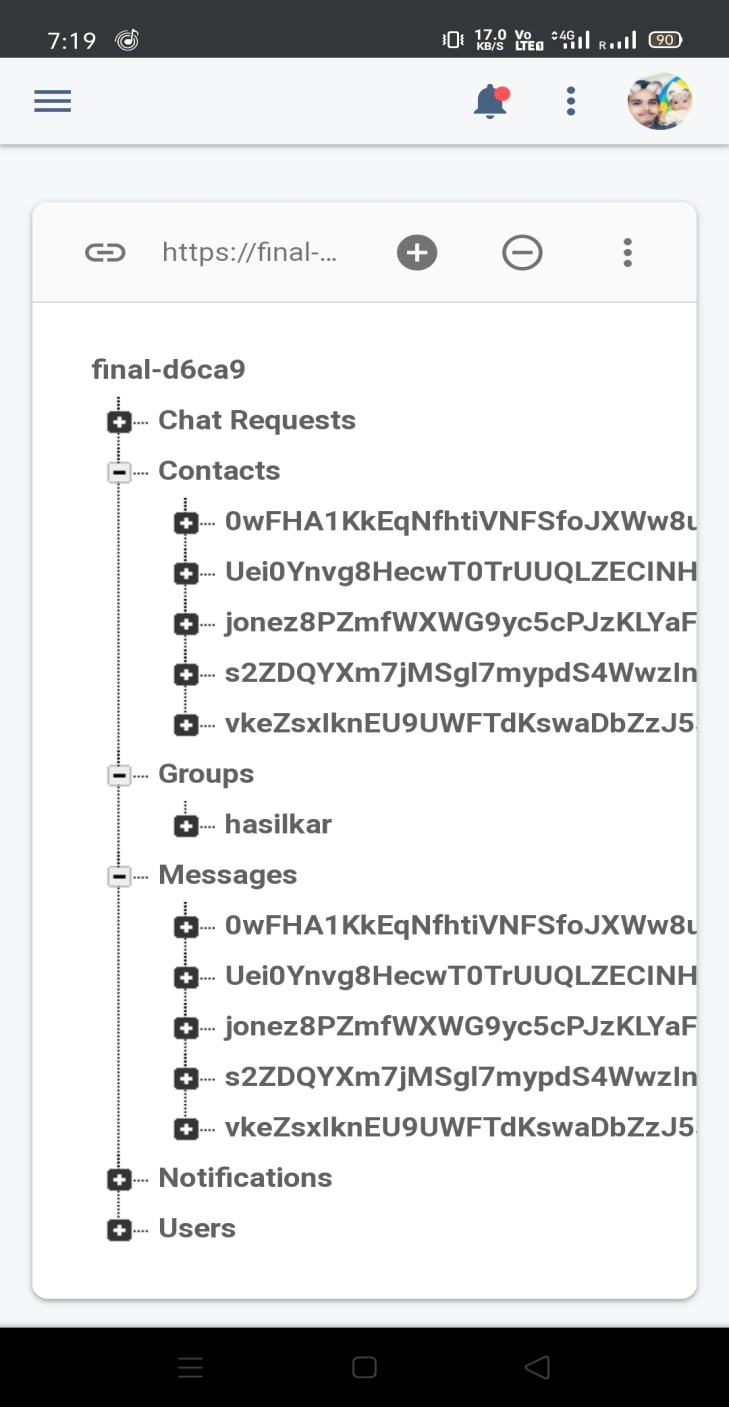
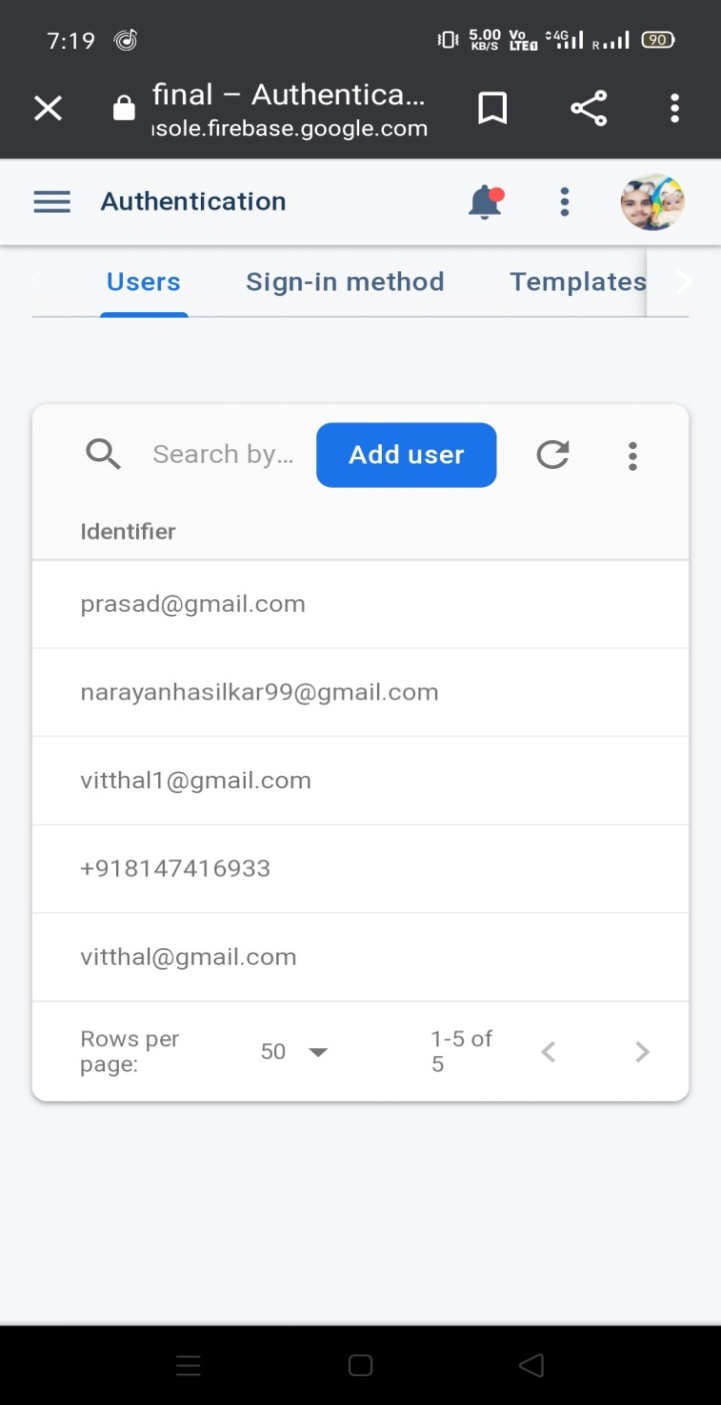
**5.3 Input And Output Screens**











**6.CONCLUSION**

**6.1 Conclusion**

* We studied the efficient use of developing chat app an android application and making the great use of it.
* We focused on the authentication processs and privacy of user and the feasible way of tranfering messages.
* We use these results o determine the usage of this application is very easy and everyone can make use of it by easy and feasible ways of communication.
* Main objective to develop the app was to provide communication of messages around various geo-graphical region peoples.

**6.2Future Enhancements**

* In our project we can develop Web Application for our app where user access through browser as well.
* We can implement pattern or biometric authentication for our application.
* In our project the video/audio calling module can be added to users.
* Real time online transfer of any data can be implemented.
* We can even develop our applications for IOS and Windows Operating System or Devices.

**7. REFERENCES**

**7.1 SITES:**

* Firebase reference:-[https://firebase.google.com/docs/android](https://firebase.google.com/docs/android/setup?gclid=CjwKCAjw7e_0BRB7EiwAlH-goG8jaksLh93eLkYf1-8OBbik5gciX87k-XHfsYSOnPGAU55s0Z6k2xoCAwkQAvD_BwE)
* Android Studio:-<https://codinginflow.com/>
* Android SDK and JAVA:-[https://www.tutorialspoint.com/android/](https://www.tutorialspoint.com/android/android_fragments.htm)
* Node js reference : <https://nodejs.org/en/docs/>
* Android with Firebase tutorials :<https://www.javatpoint.com/firebase>

**7.2 BOOKS:**

* Android Studio Development Essential By Neil Smith
* Software engineering seventh edition by Ian Sommerville.