**Messenger**

Internship Report Submitted to

**AYYA NADAR JANAKI AMMAL COLLEGE, SIVAKASI**

(Autonomous, Re-accredited (4th cycle) with ‘A+’ Grade (CGPA of 3.48 out of 4) by NAAC, Recognized as College of Excellence & Mentor Institution by UGC and STAR College by DBT and Ranked 58th at National Level in NIRF, 2020)

Affiliated to the **Madurai Kamaraj University,** in partial

fulfilment of the requirements for the award of the

Degree of

**BACHELOR OF SCIENCE**

**In**

**COMPUTER SCIENCE**

**Under the Guidance of**

**Mrs. Meena M.C.A., M.Phil.,**

**By**

**J.R.Amresh Chandh Reg. No: 18UN12**

****

**UNDER GRADUATE DEPARTMENT OF COMPUTER SCIENCE (SF)**

**AYYA NADAR JANAKI AMMAL COLLEGE (AUTONOMOUS),**

**SIVAKASI – 626 124**

**July - 2020**

**Dr. J. JEBAKUMARI BEULAH VASANTHI, M.C.A., M.Phil., Ph.D.,**

Head of the Department,

Department of Computer Science (SF),

Ayya Nadar Janaki Ammal College (Autonomous),

Sivakasi – 626124.

**CERTIFICATE**

This is to certify that this Internship report entitled as **“Messenger”** being submitted by **J. R. Amresh Chandh (Reg. No: 18UN12),** final year student of Under Graduate Department of Computer Science (SF), Ayya Nadar Janaki Ammal College (Autonomous), Sivakasi is a bonafide record of work carried out by her under the guidance of **Mrs.K. Meena M.C.A., M.Phil.,** Assistant Professor, Department of Computer Science (SF), Ayya Nadar Janaki Ammal College (Autonomous), Sivakasi.

Place : Sivakasi **Head of the Department**

Date :

**Mrs.K.Meena M.C.A.,M.Phil.,**

Assistant Professor,

Department of Computer Science (SF),

Ayya Nadar Janaki Ammal College (Autonomous),

Sivakasi– 626124.

**CERTIFICATE**

This is to certify that this Internship report entitled as **“Messenger”** being submitted by **J.R. Amresh Chandh (Reg. No: 18UN12 ),** final year student of Under Graduate Department of Computer Science (SF), Ayya Nadar Janaki Ammal College (Autonomous), Sivakasi is a bonafide record of work carried out by her under my guidance and supervision.

It is further certified that to the best of my knowledge, this report or any part thereof has not been submitted in this college or elsewhere for the award of any other degree or diploma.

Place : Sivakasi **Signature of the Guide**

Date :

**J.R.Amresh Chandh Reg. No: 18UN12**

II B.Sc. (CS) –B,

Department Computer Science (SF),

Ayya Nadar Janaki Ammal College (Autonomous),

Sivakasi.

**DECLARATION**

This Internship report entitled as “**Messenger**” has been carried out by me at the Under Graduate Department of Computer Science (SF), Ayya Nadar Janaki Ammal College (Autonomous), Sivakasi in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Science.

I further declare that this report or any part thereof has not been submitted in this College or elsewhere for the award of any other degree or diploma.

Place: Sivakasi, **Signature of the Candidate**

Date:

**ACKNOWLEDGEMENT**

First of all, I thank the “Lord Almighty” for the traces of his blessing showered upon us which is the vital factor for the successful completion of this Internship work.

I express my sincere thanks to our College Management and our beloved principal

**Dr. C. ASHOK, B.Sc., M.P.Ed, M.Phil., D.Y.Ed., Ph.D.** for providing us an opportunity to undergo this Internship report.

I express my sincere thanks to the respected Head of the Department  
 **Dr. J. JEBAKUMARI BEULAH VASANTHI, M.C.A., M. Phil., Ph.D.** for her valuable suggestions and encouragement in completing this Internship report.

I express my profound gratitude to our guide, Guide name Assistant Professor, Department of Computer Science (SF) for his/her great support and encouragement to complete this report successfully.

Finally, we extend our heartfelt thanks to our parents and friends

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **CONTENTS** | **PAGE NO.** |
| **1.** | **ABSTRACT** |  |
| **2.** | **SYSTEM ANALYSIS**  2.1 Existing System  2.2 Proposed System |  |
| **3.** | **SYSTEM REQUIREMENTS**  3.1 Hardware Requirements  3.2 Software Requirements |  |
| **4.** | **ABOUT THE SOFTWARE**  4.1 Front-End  4.2 Back-End |  |
| **5.** | **SYSTEM DESIGN**    5.1 Project Description |  |
| **6.** | **SYSTEM IMPLEMENTATION**  6.1 Source Code  6.2 Screen Shots |  |
| **7.** | **CONCLUSION** |  |
| **APPENDIX** | **BIBLIOGRAPHY** | **A1** |

**1.ABSTRACT**

This project entitled “**Messenger**” is an Android chat application that allows users to chat with each other. The main feature of this application is the user can send image file instantly through this application.

This application is exclusively for our society. This application provides security for the users. Users can create individual chatrooms and share messages within them.

**2. SYSTEM ANALYSIS**

**2.1 Existing System**

Now days people send messages through SMS. There is no acknowledgement

about the message that the user received it or read it. There is no security while sending the

message. They can’t send images through messages. We didn’t know that the user is

currently available or not to send the message.

**2.2 Proposed System**

This application is completely user friendly that the user can send or receive

message instantly. There is acknowledgement for the user that the message is sent / seen.

This application provides complete data security for the user. The user can send images.

Through this application the User can set profile pictures. The user can know that the other

user is online or offline.

**3. SYSTEM SPECIFICATIONS**

**3.1 Hardware Requirement**

**Processor:** Intel Core I / AMD Ryzen 3

**Ram:** 8GB

**Hard Disk / SSD:** 2 to 5 GB

**3.2 Software Requirement**

* Windows 10 64bit
* Android Studio
* jdk-8-windows-i586
* FireBase

**4. SOFTWARE SPECIFICATION**

**4.1 Android**

Android is a Linux-based operating system designed primarily for touch screen mobile devices such as Smartphone’s and tablet computers. Android is open source and Google releases the code under the Apache License. Android is based on Linux and offers us a great deal of customization in widgets and over millions of apps. Most of them are free of cost and can be installed on phone just by clicking on install tab of the respective app in the Google Play Store app which comes along with the android Phone.

Android is one the hottest mobile operating systems available today. Samsung is the Largest Manufacturer of android phones and tablets. LG, HTC, Sony, Moto, are other top manufacturers of android phones and tablets. Some local manufacturers like Micromax, Karbon, Hawai, also use android Phones on their portable devices.

**Logo**



**Android Logo**

**Android Version**

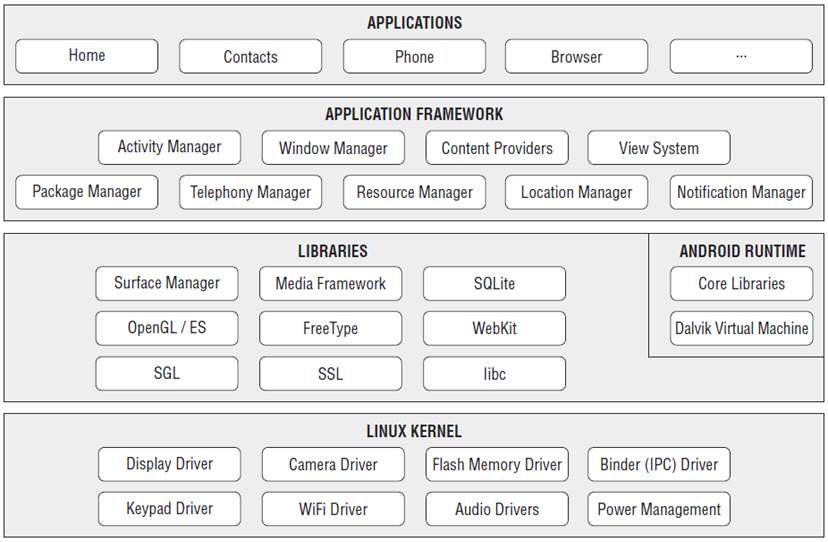
Android is released in series of Versions. Starting from 1.0 version (where 2.0, 3.0 … are latest releases). Google name these versions with some food items like ice cream, jelly bean, sandwich etc. which is one of the specialty of android versions.



**Android Version**

Here are some of the Versions released by android.

* 1.0 – Android beta.
* 1.5 – Android Cupcake.
* 1.6 – Android Donut.
* 2.0/2.1 – Eclair.
* 2.2.x – Froyo.
* 2.3.x – Gingerbread.
* 3.x – Honeycomb (used mainly for tablets.)
* 4.0.x – Ice Cream Sandwich (both for phones and tablets.)
* 4.1 – 4.3 – Jelly Bean
* 4.4 – KitKat
* 5.0 – Lollipop
* 6.0 - Marsh mellow
* 7.0 - Nougat
* 8.0 - Oreo
* 9.0 - Pie
* 10.0 – Q



**Android Libraries**

This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows

1. **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
2. **android.content** − Facilitates content access, publishing and messaging between applications and application components.
3. **android.database** − Used to access data published by content providers and includes SQLite database management classes.
4. **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
5. **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
6. **android.text** − Used to render and manipulate text on a device display.
7. **android.view** −The fundamental building blocks of application user interfaces.
8. **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
9. **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

**Android Runtime**

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**ANDROID STUDIO**

**Instant Run**

When you click Run or Debug, Android Studio's Instant Run feature pushes code and resource changes to your running app. It intelligently understands the changes and often delivers them without restarting your app or rebuilding your APK, so you can see the effects immediately.

**Intelligent code editor**

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

**Fast and feature-rich emulator**

The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

**Configure Builds Without Limits**

Android Studio's project structure and Gradle-based builds provide the flexibility you need to generate APKs for all device types.

**Optimized for all Android devices**

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug.

***4.2*****Firebase**

  Firebase is Google's cloud computing and development platform.It provides authentication, Realtime DataBase with NOSQL, Storage and Hosting service.

**SYSTEM DESIGN**

***5.1*****Project Description**

  A software system is always divided into several sub systems that makes it easier for the development. A software system that is structured into several subsystems makes it easy for the development and testing. The different subsystems are known as the modules and the process of dividing an entire system into subsystems is known as modularization or decomposition. The system under consideration has been divided into several modules taking in consideration the mentioned criteria.

**User Login:**

This is the opening page for the user, user can newly register to this application. If the user is already registered, the user can just login.

**User Details:**

This Fragment consist basic details like username, Gmail Id, and Address of the user. The user can also update, name and address.

**Search Block:**

In search block we find other user who has register in this application. By clicking on their name we can choose whether to send message or visit their profile.

**Chat Block:**

In chat block the user can send message to other user. Here we can the the latest messages and we can know which users are online right now.

**Profile Block:**

Here the user can set profile and cover pictures and link their social media account.

**User Logout:**

Clicking on the three dot menu on the top right corner to Logout the current user from the database.

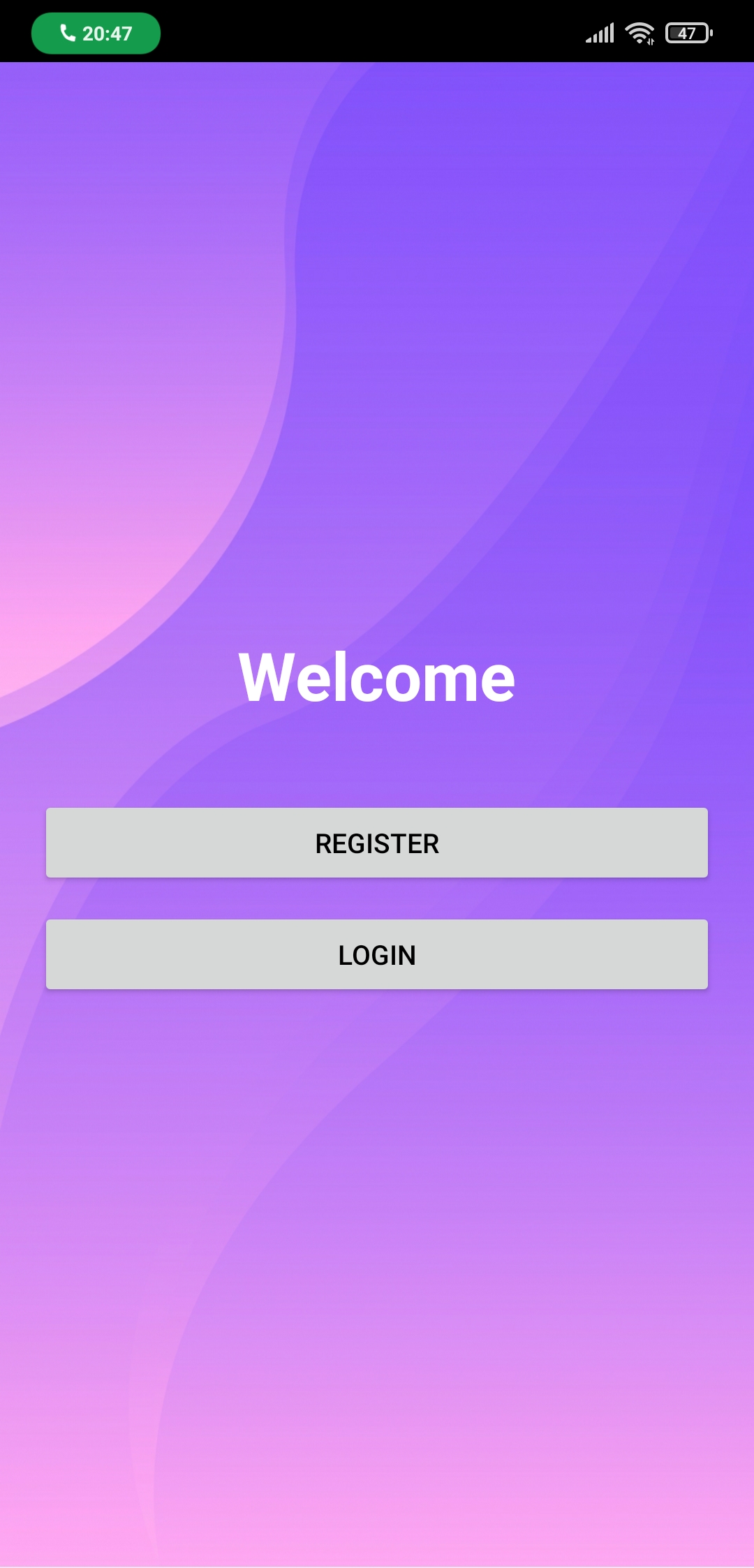
**6. SYSTEM IMPLEMENTATION**

**6.1 SOURCE CODE:**

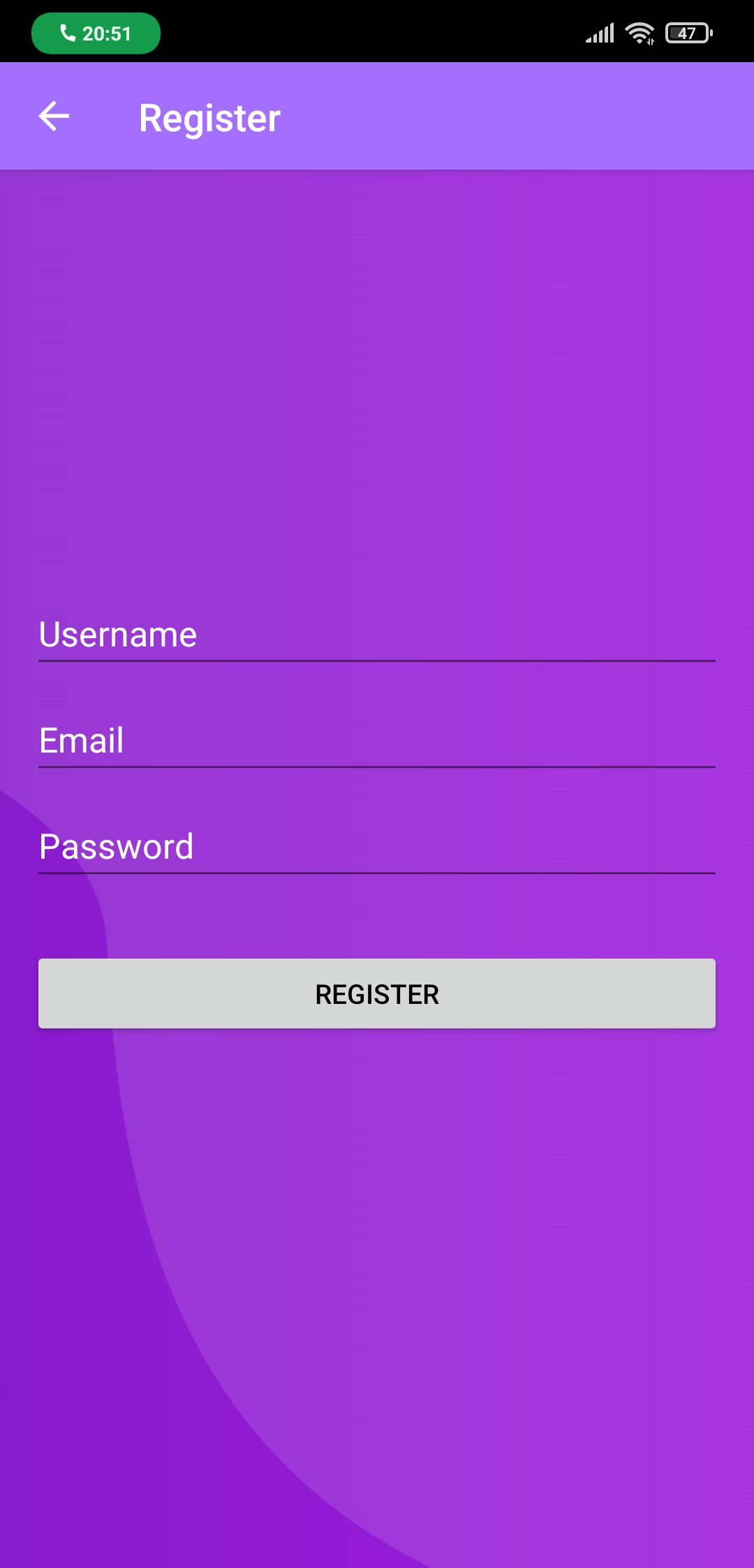
package com.amresh.messenger  
  
import android.content.Intent  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import android.view.Menu  
import android.view.MenuItem  
import androidx.appcompat.widget.Toolbar  
import androidx.fragment.app.Fragment  
import androidx.fragment.app.FragmentManager  
import androidx.fragment.app.FragmentPagerAdapter  
import androidx.viewpager.widget.ViewPager  
import com.amresh.messenger.Fragments.ChatFragment  
import com.amresh.messenger.Fragments.SearchFragment  
import com.amresh.messenger.Fragments.SettingsFragment  
import com.amresh.messenger.ModelClasses.Chat  
import com.amresh.messenger.ModelClasses.ChatList  
import com.amresh.messenger.ModelClasses.Users  
import com.google.android.material.tabs.TabLayout  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.auth.FirebaseUser  
import com.google.firebase.database.\*  
import com.squareup.picasso.Picasso  
  
import kotlinx.android.synthetic.main.activity\_main.\*  
  
class MainActivity : AppCompatActivity() {  
  
 var refUsers: DatabaseReference? = null  
 var firebaseUser: FirebaseUser? = null  
  
  
 override fun onCreate(savedInstanceState: Bundle?)  
 {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 setSupportActionBar(toolbar\_main)  
  
 firebaseUser = FirebaseAuth.getInstance().*currentUser* refUsers = FirebaseDatabase.getInstance().*reference*.child("Users").child(firebaseUser!!.*uid*)  
  
  
 val toolbar: Toolbar = findViewById(R.id.*toolbar\_main*)  
 setSupportActionBar(toolbar)  
 *supportActionBar*!!.*title* = ""  
  
 val tabLayout: TabLayout = findViewById(R.id.*tab\_layout*)  
 val viewPager: ViewPager = findViewById(R.id.*view\_pager*)  
  
 val ref = FirebaseDatabase.getInstance().*reference*.child("chats")  
 ref.addValueEventListener(object : ValueEventListener{  
 override fun onDataChange(p0: DataSnapshot)  
 {  
 val viewPagerAdapter = ViewPagerAdapter(*supportFragmentManager*)  
 var countUnreadMessages = 0  
 for(dataSnapshot in p0.*children*)  
 {  
 val chat = dataSnapshot.getValue(Chat::class.*java*)  
 if(chat!!.getReceiver().*equals*(firebaseUser!!.*uid*) && !chat.isIsSeen())  
 {  
 countUnreadMessages += 1  
 }  
  
 }  
 if (countUnreadMessages == 0)  
 {  
 viewPagerAdapter.addFragment(ChatFragment(), "chats")  
  
 }  
 else  
 {  
 viewPagerAdapter.addFragment(ChatFragment(), "($countUnreadMessages) chats")  
 }  
 viewPagerAdapter.addFragment(SearchFragment(), "search")  
 viewPagerAdapter.addFragment(SettingsFragment(), "settings")  
 viewPager.*adapter* = viewPagerAdapter  
 tabLayout.setupWithViewPager(viewPager)  
 }  
  
 override fun onCancelled(p0: DatabaseError)  
 {  
  
 }  
 })  
  
  
 //display the username and profile picture  
 refUsers!!.addValueEventListener(object : ValueEventListener{  
 override fun onDataChange(p0: DataSnapshot) {  
 if (p0.exists())  
 {  
 val user: Users? = p0.getValue(Users::class.*java*)  
  
 user\_name.*text* = user!!.getUsername()  
 Picasso.get().load(user.getProfile()).placeholder(R.drawable.*profile*).into(profile\_image)  
 }  
 }  
  
 override fun onCancelled(p0: DatabaseError) {  
  
 }  
 })  
 }  
  
 override fun onCreateOptionsMenu(menu: Menu): Boolean {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 *menuInflater*.inflate(R.menu.*menu\_main*, menu)  
 return true  
 }  
  
 override fun onOptionsItemSelected(item: MenuItem): Boolean {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 when (item.*itemId*) {  
 R.id.*action\_logout* -> {  
  
 FirebaseAuth.getInstance().signOut()  
 val intent = Intent(this@MainActivity, WelcomeActivity::class.*java*)  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TASK* or Intent. *FLAG\_ACTIVITY\_NEW\_TASK*)  
 startActivity(intent)  
 finish()  
  
 return true  
 }  
 }  
 return false  
 }  
internal class ViewPagerAdapter(fragmentManager: FragmentManager) :  
 FragmentPagerAdapter(fragmentManager)  
{  
 private val fragments: ArrayList<Fragment>  
 private val titles: ArrayList<String>  
  
 init {  
 fragments = ArrayList<Fragment>()  
 titles = ArrayList<String>()  
 }  
 override fun getItem(position: Int): Fragment {  
 return fragments[position]  
  
 }  
  
 override fun getCount(): Int {  
  
 return fragments.size  
 }  
  
 fun addFragment(fragment: Fragment, title: String )  
 {  
 fragments.add(fragment)  
 titles.add(title)  
 }  
  
 override fun getPageTitle(i: Int): CharSequence? {  
  
 return titles[i]  
 }  
}  
  
  
 private fun updateStatus(status: String)  
 {  
 val ref = FirebaseDatabase.getInstance().*reference*.child("Users").child(firebaseUser!!.*uid*)  
  
 val hashMap = HashMap<String, Any>()  
  
 hashMap["status"] = status  
 ref!!.updateChildren(hashMap)  
  
  
 }  
  
 override fun onResume() {  
 super.onResume()  
  
 updateStatus("Online")  
 }  
  
 override fun onPause() {  
 super.onPause()  
  
 updateStatus("Offline")  
 }  
  
}

package com.amresh.messenger  
  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Toast  
import androidx.appcompat.widget.Toolbar  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.database.DatabaseReference  
import com.google.firebase.database.FirebaseDatabase  
import kotlinx.android.synthetic.main.activity\_register.\*  
  
class RegisterActivity : AppCompatActivity() {  
  
 private lateinit var mAuth: FirebaseAuth  
 private lateinit var refUsers: DatabaseReference  
 private var firebaseUserID: String = ""  
  
  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_register*)  
  
  
 val toolbar: Toolbar = findViewById(R.id.*toolbar\_register*)  
 setSupportActionBar(toolbar)  
 *supportActionBar*!!.*title* = "Register"  
 *supportActionBar*!!.setDisplayHomeAsUpEnabled(true)  
 toolbar.setNavigationOnClickListener **{** val intent = Intent(this@RegisterActivity, WelcomeActivity::class.*java*)  
 startActivity(intent)  
 finish()  
 **}** mAuth = FirebaseAuth.getInstance()  
  
 register\_btn.setOnClickListener **{** registerUser()  
 **}** }  
  
 private fun registerUser()  
 {  
 val username: String = username\_register.*text*.toString()  
 val email: String = email\_register.*text*.toString()  
 val password: String = password\_register.*text*.toString()  
  
 if(username == "")  
 {  
 Toast.makeText(this@RegisterActivity, "Please Enter the username", Toast.*LENGTH\_LONG*).show()  
 }  
 else if(email == "")  
 {  
 Toast.makeText(this@RegisterActivity, "Please Enter the Email", Toast.*LENGTH\_LONG*).show()  
 }  
 else if(password == "")  
 {  
 Toast.makeText(this@RegisterActivity, "Please Enter the Password", Toast.*LENGTH\_LONG*).show()  
 }  
 else  
 {  
 mAuth.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener**{**task **->** if(task.*isSuccessful*)  
 {  
 firebaseUserID = mAuth.*currentUser*!!.*uid* refUsers = FirebaseDatabase.getInstance().*reference*.child("Users").child(firebaseUserID)  
  
 val userHashMap = HashMap<String, Any>()  
 userHashMap["uid"] = firebaseUserID  
 userHashMap["username"] = username  
 userHashMap["profile"] = "https://firebasestorage.googleapis.com/v0/b/messenger-ad6da.appspot.com/o/profile.jpg?alt=media&token=a549b6cf-eba8-4619-940b-fca77e3ce3cf"  
 userHashMap["cover"] = "https://firebasestorage.googleapis.com/v0/b/messenger-ad6da.appspot.com/o/cover.jpg?alt=media&token=d0318e97-6e7d-46e4-972b-05b68f5f2010"  
 userHashMap["status"] = "offline"  
 userHashMap["search"] = username.*toLowerCase*()  
 userHashMap["facebook"] = "https://m.facebook.com"  
 userHashMap["instagram"] = "https://m.instagram.com"  
 userHashMap["website"] = "https://www.google.com"

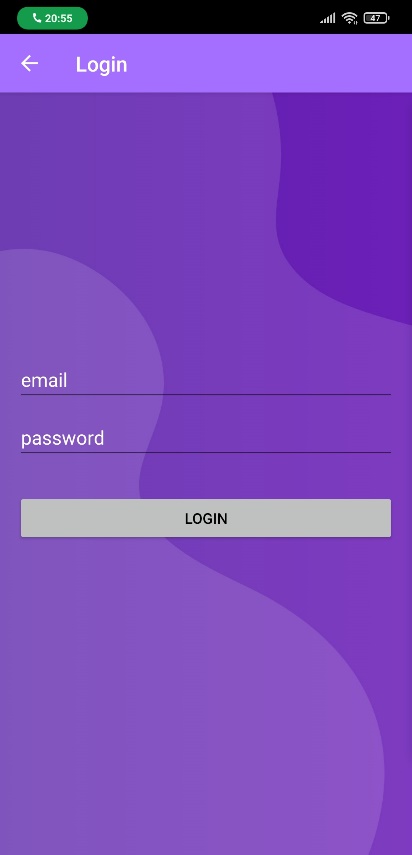
**6.2 SCREEN SHOTS**



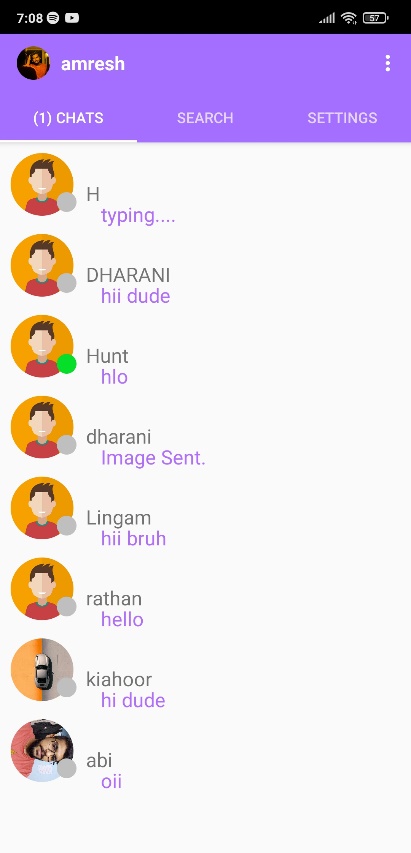
**Fig 6.2.1 Home**



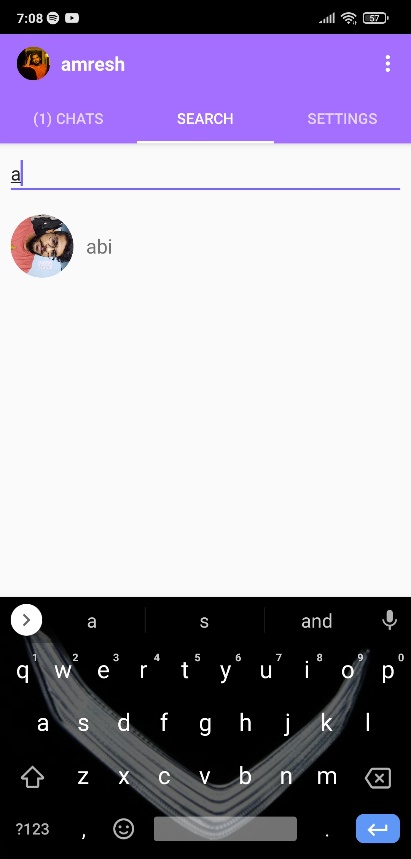
**Fig 6.2.2 Register**



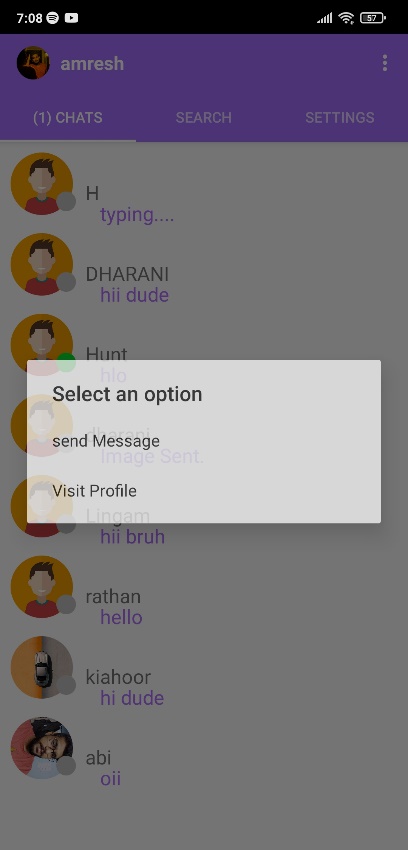
**Fig 6.2.3 Login**



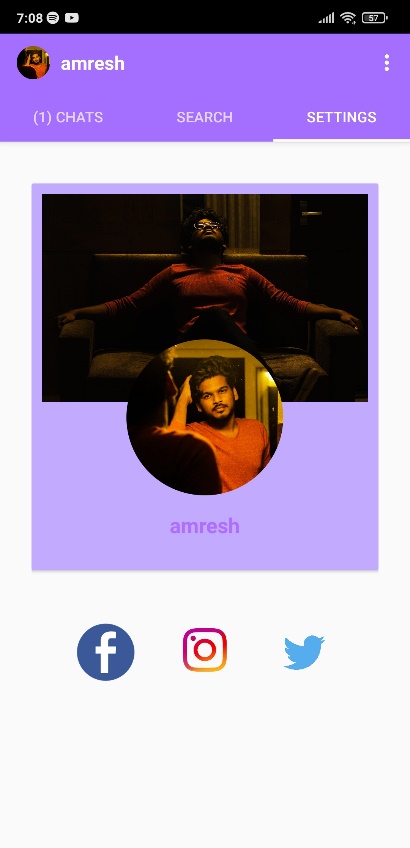
**Fig 6.2.4 Chat Room**

****

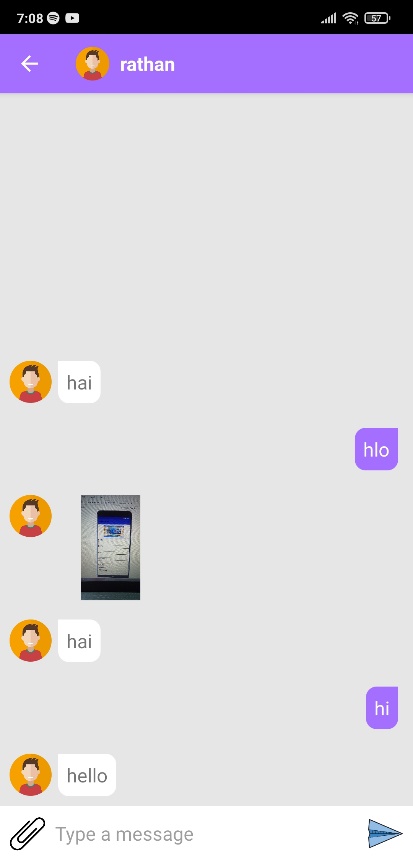
**Fig 6.2.5 Search**

****

**Fig 6.2.6 Action**

****

**Fig 6.2.7 Profile**

****

` **Fig 6.2.8 Chat**

**7. CONCLUSION**

This application will be more user friendly. The user can send messages or images instantly. This is a basic messaging application with minimal features.