A

Report

On



**[Vaarta: Multi-Lingual Messaging & media Sharing Application]**

Minor Project

For

Partial fulfillment of

Computer Science & Engineering

Session (2023-24)

Submitted By:

**[Aman Raj Singh: 2215990009]**

**[Prashant Kumar: 2215990045]**

**[Varsha Shrotriya:2115001096]**

Submitted To:

**Miss. Robin Khurana**

**Project Mentor**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**G.L.A UNIVERSITY, MATHURA (U.P.)**

ACKNOWLEDGEMENT

It plunges me in exhilaration in taking privilege in expressing our heartfelt gratitude to **Miss Robin Khurana,** Lecturer of CSE department, GLA University, for providing every facility, for their constant encouragement, suggestions, constant supervision, and abundant support throughout our project.

Thanks to all the teaching and non-teaching staff of CSE department for their support and also to our Team-mates for their valuable Co-operation.

**Project Guide**

**CONTENTS**

**S.NO. Page No.**

1. Project Abstract 1-2
2. Cost Estimation 6-8
3. SRS 9-19
   1. Introduction
      1. Purpose
      2. Scope
      3. References
      4. Software Development Methodology
   2. System Analysis
      1. Study of the Existing System
      2. Problems in Existing System
      3. Proposed System
      4. Advantages of proposed system
      5. System Feature
      6. Requirement Specification
         1. Functional Requirements
   3. System Requirements Study
      1. Software Requirements
      2. Hardware Requirements
   4. User Requirement Document(URD) 21-26
      1. Use-Case Diagram
      2. Activity Diagram
   5. System Design 27-29
      1. System Feature
      2. DFD
      3. Sequence Diagram 30-34
      4. Data Dictionary

4.6.4.1 Table Structure 35-37

* + 1. Class Diagram 41

1. Screen Shots 43-58 59-61
2. Coding 60-63
   1. Coding Standards & Guidelines 63-70
   2. Technology Trends
      1. Front End
      2. Back End
      3. Other Tools

6.Testing & Implementation 70-72

6.1 Test Case Description 72-73

* 1. Result Table 73-74

7. Conclusion 74-75

**PROJECT ABSTRACT**

**Project Title: Vaarta: Multi-Lingual Messaging and Media Sharing Application**

**Project Description:** Messaging applications are very popular among Internet users and Smartphone’s owners. A Hundred million smartphone owners use chat applications on a monthly basis. These chat applications offer the communication free of charge and the majority of them are free to install which makes it very appealing for the potential customers. We can use it every day like business, traveling, and education Sector. Android Studio is used for development of this Android Application and All Data will we stored In Google Firebase Fire store Database.

This application provides many features like real-time messaging, media sharing, video/ voice conferencing, multilingual message translation etc. Anyone could use this application to chat with foreigners without any language barrier. The UI design of this application is greatly influenced by the already existing messaging application to give the user a fresh but familiar UX experience. The main feature of this application is to provide multilingual translation of text, voice, images, and Documents will be translated automatically to user's selected native language.

In this span of Time people have developed a lot of ways to communicate to each other, but there is one way of communication that hasn't changed for years, that is our languages. But in the diversity of the world people have developed many languages to communicate to each other for example American use English and Indian use Hindi as we see in a broader perspective it doesn't look like a problem but let's say an Indian wanted to talk to an American without leaving his/her native language they would only have two options

1. Learn English

2. Auto Translate

It from Google but what if we say that they have another option called “VAARTA”. Now there may be a question in your head "what is “Vaarta?". So “Vaarta” is a messaging application that helps users get real-time message translations and Real Time Voice Translations while they are communicating with the other person with the help of latest Microsoft Azure Translator API and it happens in an instance with the help of the Firebase Fire-store real-time database. Our Application “Vaarta” will truly change how the people communicate to each other by completely removing the language barrier and helping people in various field of work let it be in tourism, Multinational Corporation or even in your average day-to-day life of a normal human being. Everyone could be free to express themselves without the hassle of worrying if the other person understands them.

**COST ESTIMATION**

**EFFORT AND COST ESTIMATION**

**FUNCTION POINT MODEL**

It is based on the visible features of the system that are weighed accordingly to produce an overall score. The intent is to construct a measure of product size that can be available easily in the development process. It is based on the notion of function points regarding as a measure of functionality of the system. The starting point of the construction of the model is to determine the number of items occurring in the system.

The items are as follows:

1. **UI design** - more beautiful and more complex UI design requires more budget. It takes a lot of time to design a great UI. Designers make the layouts in sketch/Photoshop/etc. and then write XML code for that. So this is a major point. It required a budget to create a professional UI Screens. It required a lot of time of developer.
2. **Features** - This is also a major point. What all features required in the app matters? For example to integrate chat feature the developer will need to implement a lot of code. And also create an Image / Audio / video Sending Feature. So as complexity of features increases, budget increases.
3. **Server side** - if some app requires server side scripting or computing, the developer will charge more. Because it is an extra part of work. Like the landing page of Vaarta Have Separated website to reach the target users. It required to develop a website on webserver so it want cost to manage that all things.

1. **Database** - management and creation of database in Firebase is initial stage is free but if you want to commercialize that application so that you have to pay on Google Firebase to get the space on database that will also increase some budget of developer.
2. **Number of screens** - generally it doesn't matters much but some complex apps requires lot of screens (activates and fragments). So this can also be considered as a factor.
3. **Bug fixes** - An application will have bugs. Some of them will be discovered by users when they attempt to do something that was not anticipated by the designer or developer. Why: you want to take care of your users. It’s expected that apps post regular update patches.

Frequency: Depends. If a critical issue is discovered (i.e. security breach, crash that prevents someone from using the app, etc.) - a patch is expected sooner. Lower priority bugs can be released on a longer schedule.

Cost: Depends. Some bugs are easy to find and fix others not so much. Double the estimate and add test time.

1. **Minor updates**

Even applications in maintenance mode can be expected to be refreshed periodically.

Why: Do you need to update a copyright year? Are there changes to content? Have you rebranded and color scheme no longer match? Found a better way to do something? Frequency and cost: Depends

1. **Infrastructure**

Another flexible category that includes the cost of equipment and licenses to maintain a development environment and any needed server support (for example, for website, web service, databases, etc.)

Where fi specifies the detailed factors contributing to the overall notion of complexity.

The various factors are as follows-

* Reliable Backup and Recovery
* Distributed Functions
* Database cost
* Play Product Cost
* API Cost
* Maintenance Cost

**SYSTEM**

**REQUIREMENT**

**SPECIFICATION**

**(SRS)**

# 

# 4.1 Introduction

## 4.1.1 Purpose

To ‘**Vaarta**’ is a mobile chatting application with ability to translate messages end to end thus allowing the user to communicate in their native language without any delay .Traditional live chat systems are unable to perform multi-lingual, two-way smart chats. **Vaarta** have pushed the boundaries to assist companies by understanding their difficulties while communicating with customers who speak another language.

## Intended Audience and Reading Suggestions

Messaging intended for Project Managers, Developers, End users and Quality Assurance engineers [Testing]. Translated Voice and Text Bases system followed by Infrastructure requirement**.**

**4.1.2 Scope**

## Language Scope

**Language:**

* **JAVA** – Programming Language.
* **XML** – Scripting/Structure Language.

## Project Scope

The scope of this project is to provide best user experience smooth and real time chat it may help collecting perfect management of user details It provides a general architecture for chat applications, and anyone or organization can use it as the basis for providing instant messaging capabilities. The application is written in an object-oriented language called Java. Clients need to install the app through play store. Only network access is needed for communicating with each other

1. It will satisfy the user requirement.
2. It will be easy to understand by the user and operator.
3. It will be easy to operate.
4. It will be have a good user interface.
5. Will be expandable.

## 4.1.3 References:

1) Java 1. Complete reference of Java - By HerbertSchildt

2. [www.w3schools.com](http://www.w3schools.com)

3. [www.java.sun.com](http://www.java.sun.com)

2) Database a) Firebase Fire store And Storage Database

3) API’s a) Jit-si-meet API for Video/voice calling

4) UML the Unified Modeling Language User Guide

## Design and Implementation Constraints

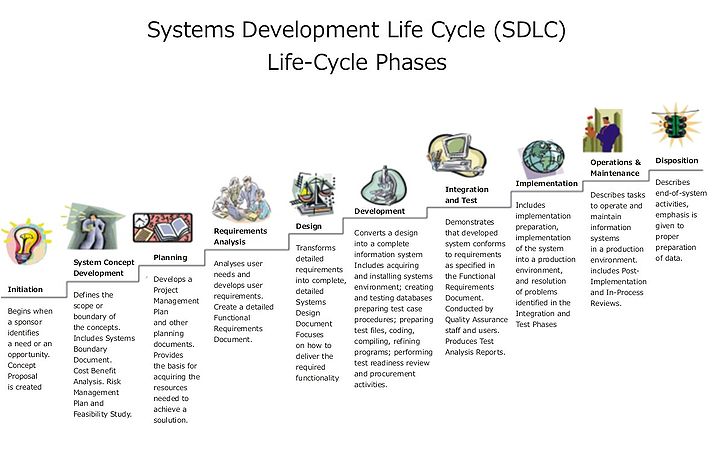
The Vaarta Chatting Application Users will register through Email And password and they will be communicate with real time chatting video/voice calling. Application gives feature to chat with own mother language its gives seamless user experience.

The main objective of the project on chatting application is to manage the details of online chat, real time message translates, chat history, chat profile, users. It manages all the information about online chat, emojis chat, and users online. The project is totally built at administrative end and thus only the administrative is guaranteed the access. The purpose of the project is to build an application to reduce the manual work for managing the online chat, translating messages in many languages it tracks all the details about User’s interaction.

* Text Messaging For End Users
* Video And Voice Calling
* Text And Audio Translation For End Users
* Image Audio Video Documents Sharing

**Software Development Methodology:**

* **Project planning, feasibility study**: Establishes a high-level view of the intended project and determines its goals.
* **Systems analysis, requirements definition**: Refines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.
* **Systems design**: Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.
* **Implementation**: The real code is written here.
* **Integration and testing**: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.
* **Acceptance, installation, deployment**: The final stage of initial development, where the software is put into production and runs actual business.
* **Maintenance**: What happens during the rest of the software's life: changes, correction, additions, and moves to a different computing platform and more.



**SYSTEM ANALYSIS**

**Existing System:**

In Existing Messaging System There will we lot Of Application Like what’s App Instagram telegram Will provide text Messaging Audio And Video Calling and file and document sharing also but these have language barrier for overseas users so we have proposed the messaging system that can overcome these language barriers.

* Text Messaging
* Audio And Video calling
* Document And files sharing
* Users Current Location
* AR camera with Different Filters

**Disadvantage of Existing System:**

* Some of Application Will not is provided Users Data Security.
* These All Have Language Barrier in Messaging System.
* Document language Barriers Overseas Users Can’t understands.
* Users will be not able to Communicate with their Native Language.

**Proposed System:**

* Traditional live chat systems are unable to perform multi-language two-way smart chats. **Vaarta** have pushed the boundaries to assist companies by understanding their pain points when communicating with customers who speak another language.
* Chatting Translation allows Users to interact with forager’s Hassle free without worrying about the language barrier.
* Increases convenience, diversity and enrolments from overseas
* Provide text Message Translation on Users native Language
* Provide Audio Calling Translation On Users Native Language
* Provide Document Image PDF Files Text Translation On Users Native language
* Vaarta Ensure The users Data Security and Privacy
* Vaarta is developed during the Smart India Initiative.

**Advantage of Proposed System**

* Translate Chat is a prototype of a hybrid mobile messaging app whose goal is help users overcome language barriers by translating texts as they chat in real time .
* Users Can Communicate With Their Native Language.
* The users will not Face Any Problem While communicating With Overseas Users
* Users Will not Face Any Problem While Reading Document Images and Other File In Different Language

# System Features

1. Simple UI

Now days, there are huge amount of users and huge amount of application so UI must be simple so user do not face a problem when he/she is using this application. a simple UI will allow user to use the application in a familiar way.

1. Real-time Language Translation

In this, language is automatically translated to desired language using the ml kit provided by Google Firebase. This feature provides great opportunity to those who are capable to do freelancing so they can talk to foreigner client. also it is useful in every sector like traveling, business, etc..

1. Dark and Light mode

The now a days, user use their devices day and night so their eyes getting harm so, this application automatically change their theme according to device theme setting..

1. Cloud storage and backup

Synchronization with a cloud service will allow the app to save and store message history, images, audios, and other types of files, and keep them up-to-date without clogging the device’s memory. It’s possible to access all the files at any time from that storage method if the Internet connection is active. In addition, if the user makes changes to files and messages from their devices, it will apply to the existing copies, all thanks to the cloud.

1. Video/Audio call

Audio and video call is most used communication way because we can see the face expression in video and audio with its tune. Then text that just share word which does not show any emotion or any thing.so, we introduce video and audio call in this application.

1. authentication

During registration in this application user has to provide a number so he/she get an OTP (one time password) to register itself. With this number of fake account will reduce because they can't make an account without any their true identity. And only one number can register one account.

1. Users Profile Update

This feature allow user to update their profile as their need. Like their name, address, Gmail, phone number and profile picture.

1. Document Sharing

Now days, user use many application for different purposes but they can't get single application with multiple features. So, overcome this we add new feature in this application "Document sharing". User can share their document using this application like famous application what app, telegram do

# FUNCTIONAL REQUIREMENTS

## User Interfaces

The Application Provide Good User Interface End Users Will easily Interact With UI components easily The Application Vaarta UI inspired by other Messaging applications That are Already Available Available In Market Vaarta Provide Seamless User Experience To the End Users.

## Communication Interfaces

The application concentrates on the online and communicates over the internet/intranet. A well connected internet connection either using a modem or cable or Wi-Fi or any other form should exist. TCP/IP configured, http supported protocol configuration should exist. The client only requires Internet Connectivity for communication. **Software Interfaces**

The incoming data to the product would be raw text data and images. The outgoing data would be the text and images. A database is maintained to store the text and URL information about the images. Google Firebase is the database with a version of minimum 2003 as requirement.

# NON FUNCTIONAL REQUIREMENTS

## Performance Requirements

Good band width, less congestion on the network. Identifying the shortest route to reach the destination would enhance performance.

## Safety Requirements

No harm is expected from the use of the product either to the OS or any data that resides on the client system.

## Product Security Requirements

The product is protected from un-authorized users from using it. The system allows only authenticated users to work on the application. The users of the system are Admin End Users All over the World.

SYSTEM REQUIREMENT STUDY

**Software Requirement:**

* Android Studio
* Android Operating System
* Firebase Database

**Hardware Requirement (Minimum)**

* Microsoft Windows 7/8/10 (32-bit or 64-bit)
* 3 GB RAM minimum, 8 GB RAM recommended (plus 1 GB for the Android Emulator)
* 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolutions.

**USER REQUIREMENT**

Use-case Diagram:

The **“user model view”** encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of “use case diagrams”. These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

## Purpose of Use Case Diagrams:

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

**Activity Diagram**

**“Activity diagrams”** render the activities of a class participating in behavior. These diagrams describe the behavior of a class in response to internal processing rather than external events. Activity diagrams describe the processing activities within a class.

**Basic notation in Activity Diagram:**

**Initial node:** The filled in circle is the starting point of the diagram.

**Activity Final node:** The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes.

**Activity:** The rounded rectangle represents activities that occur.

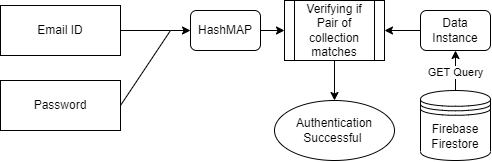
**Flow/Edge:** The arrow on the diagram.

**Fork:** A black bar with one flow going into it and several leaving it. This denotes the beginning of parallel activity.

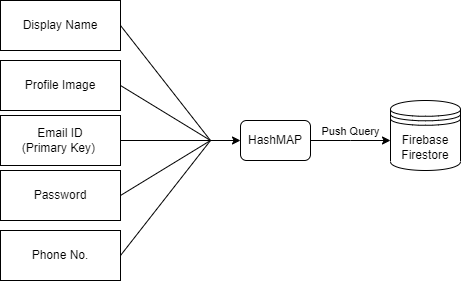
**Join:** A black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue. This denotes the end of parallel processing.

**Decision:** A diamond with one flow entering and several leaving.

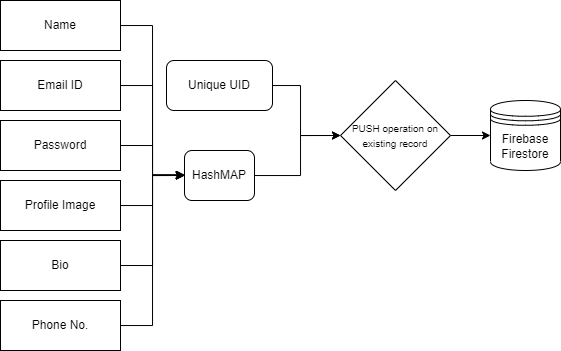
**Login Module**



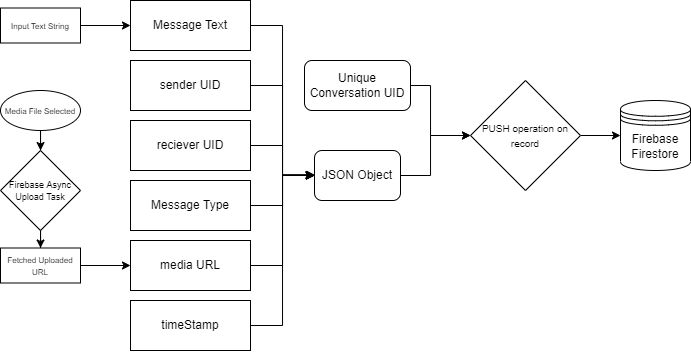
**Registration Module**



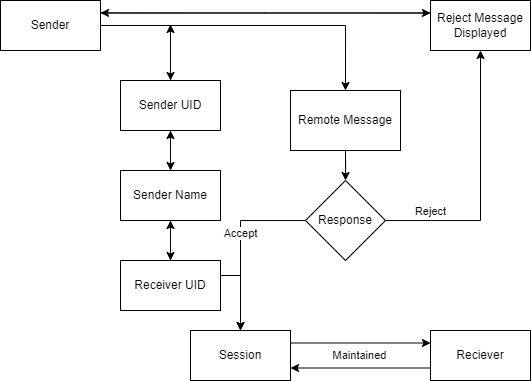
**Profile Update Activity**



**Messaging Activity**



**Audio/Video Calling Activity**

****

SYSTEM DESIGN

**INTRODUCTION**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

**DATA FLOW DIAGRAMS**

. The development of DFD’S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purposesingle process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

**DFD SYMBOLS:**

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

Process that transforms data flow.

Source or Destination of data

Data flow

Data Store

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out. Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD

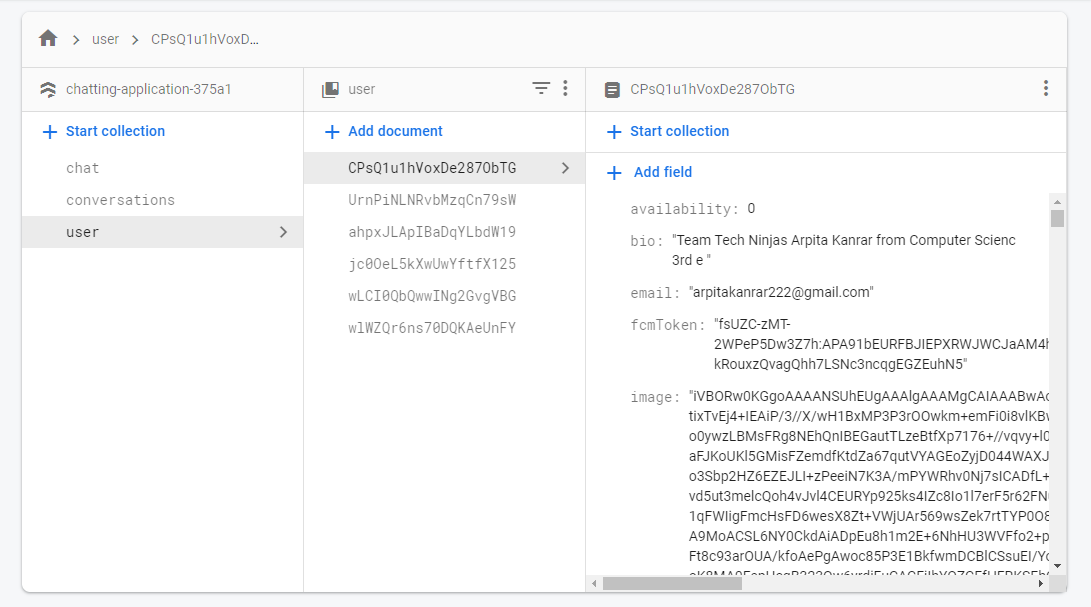
SEQUENCE DIAGRAM:

UML sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system

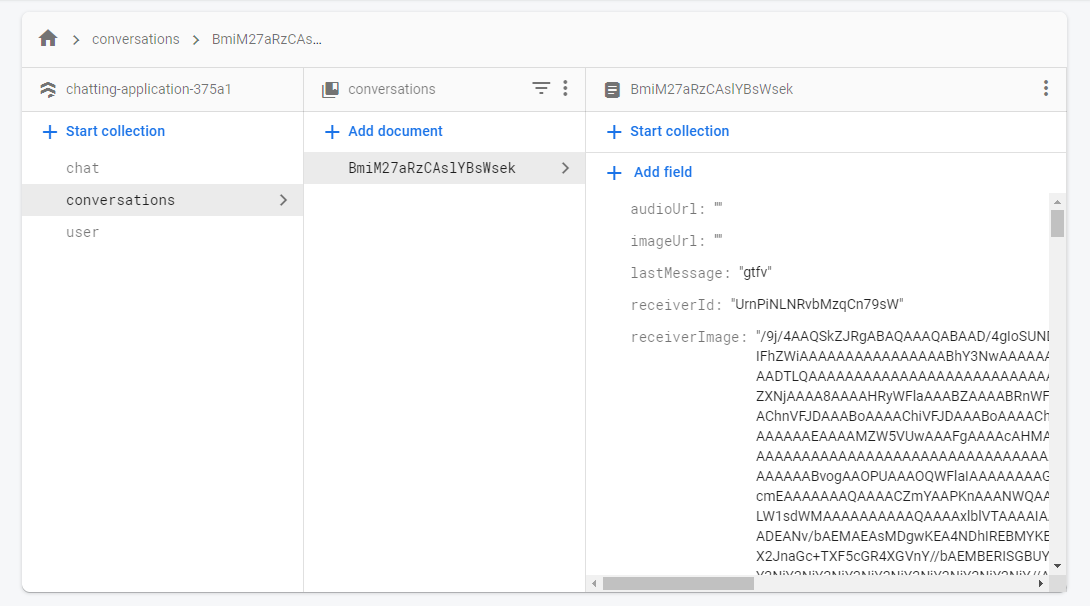
A Sequence diagram is a two dimensional in nature. On the horizontal access, it shows the life of the object that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects.

**FIREBASE DATABASE TABLE:**

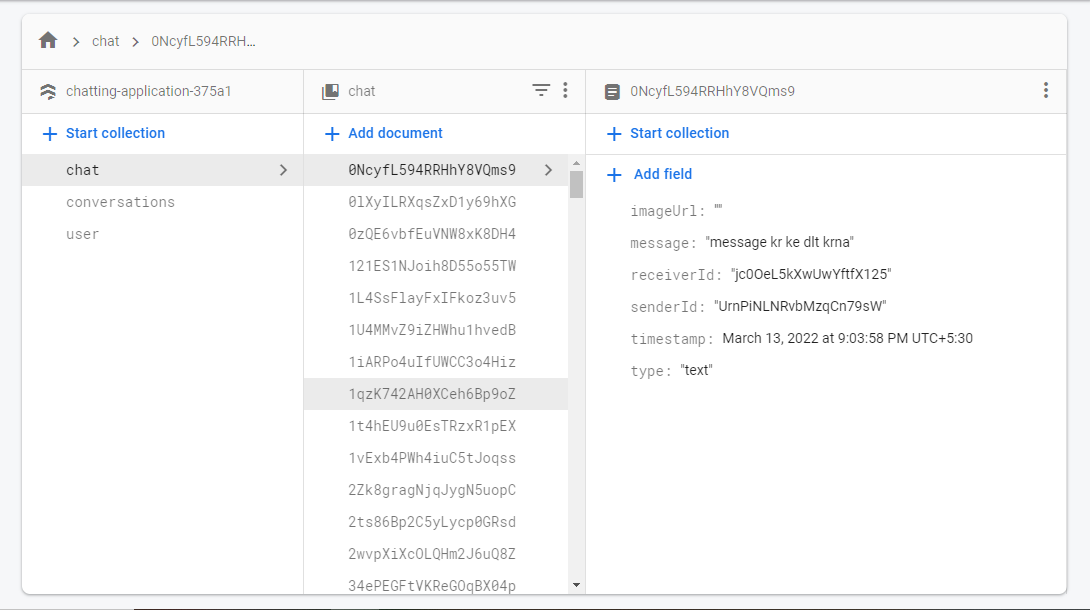
**Users Table**

****

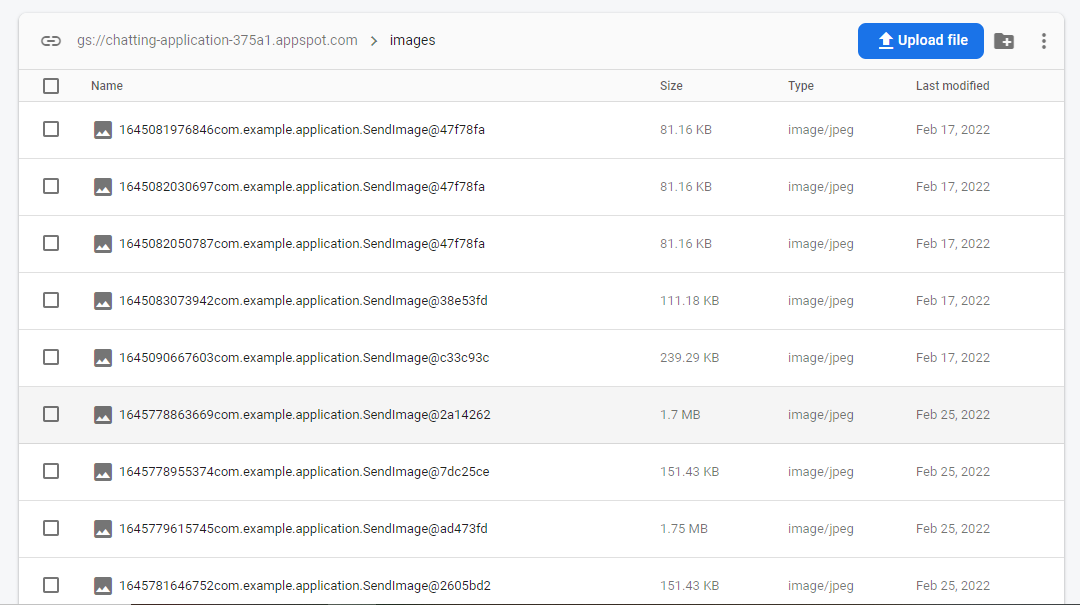
**Conversations Table**

****

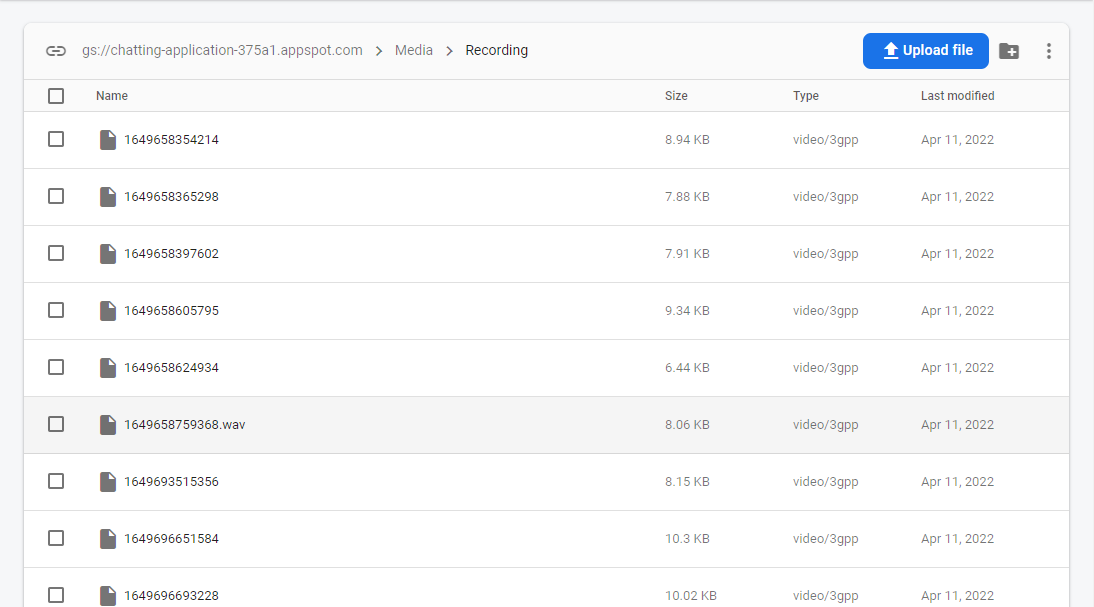
**Chat Table**

****

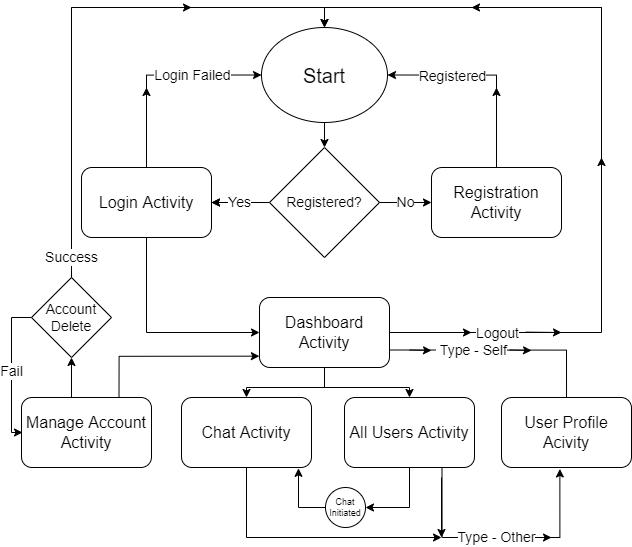
**Image Storage Data**

****

**Recording**

****

**Data Flow Diagram (DFD):**



**Figure 1**

**Proposed System Architecture:**

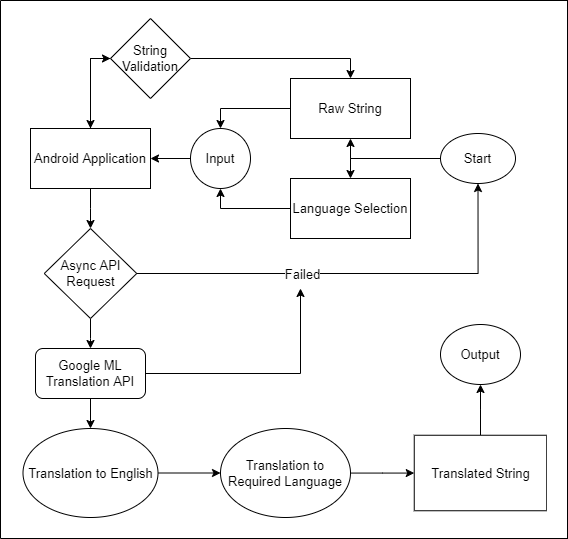
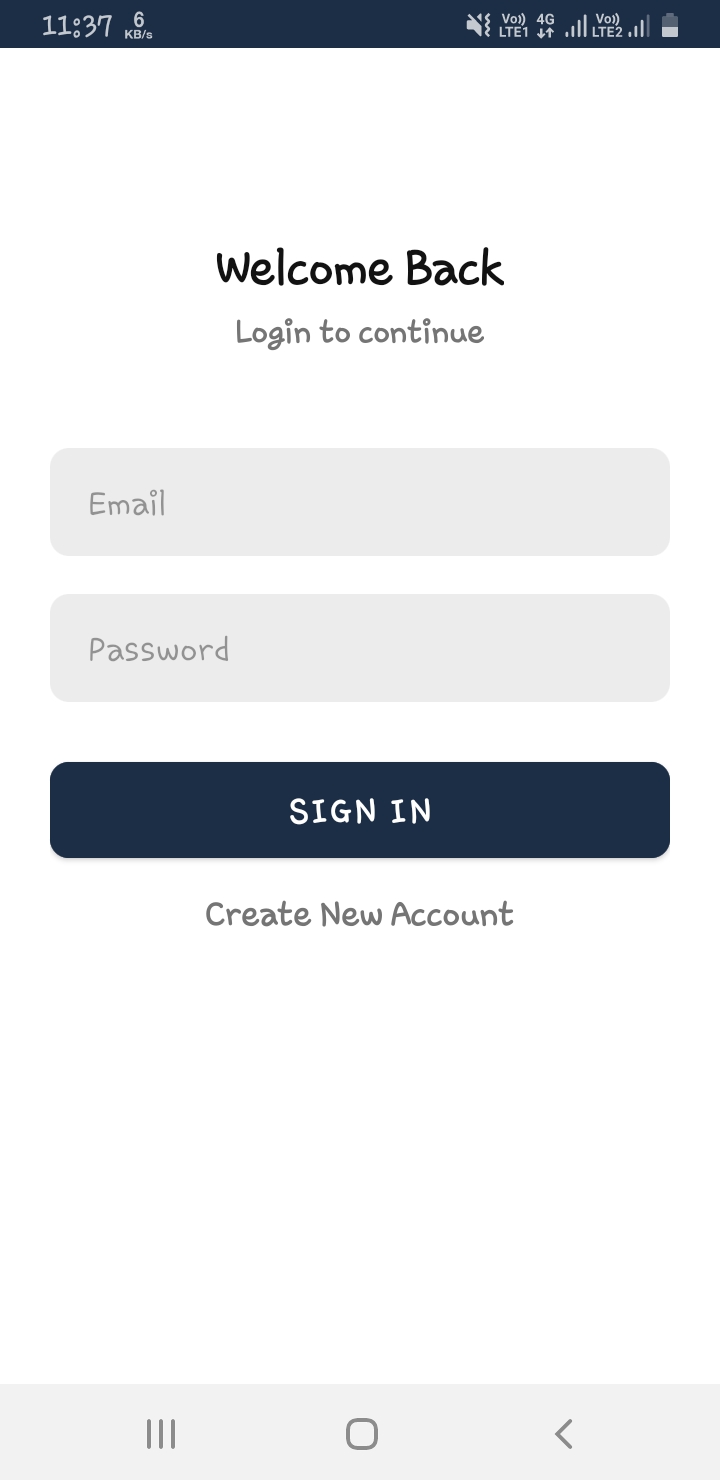


Figure 2

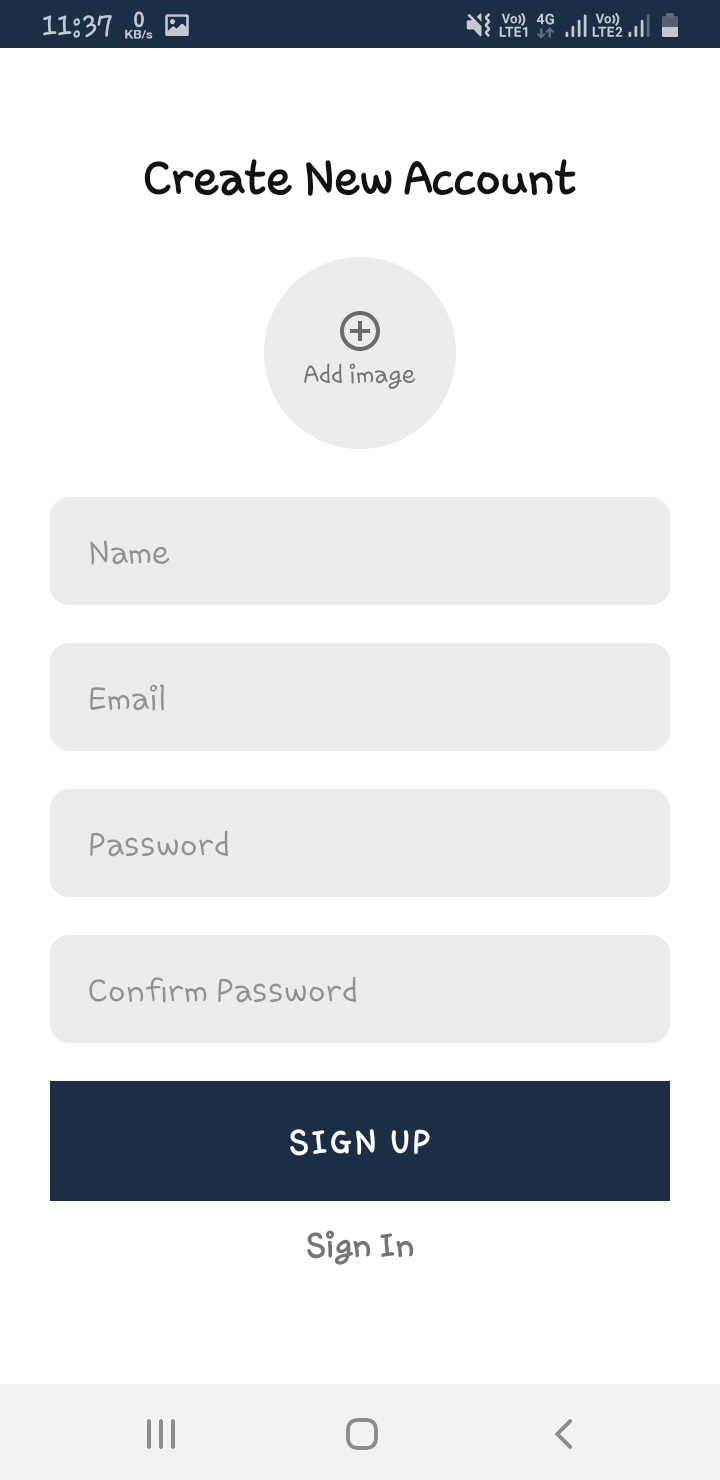
**VAARTA SNAPSHOTS:**



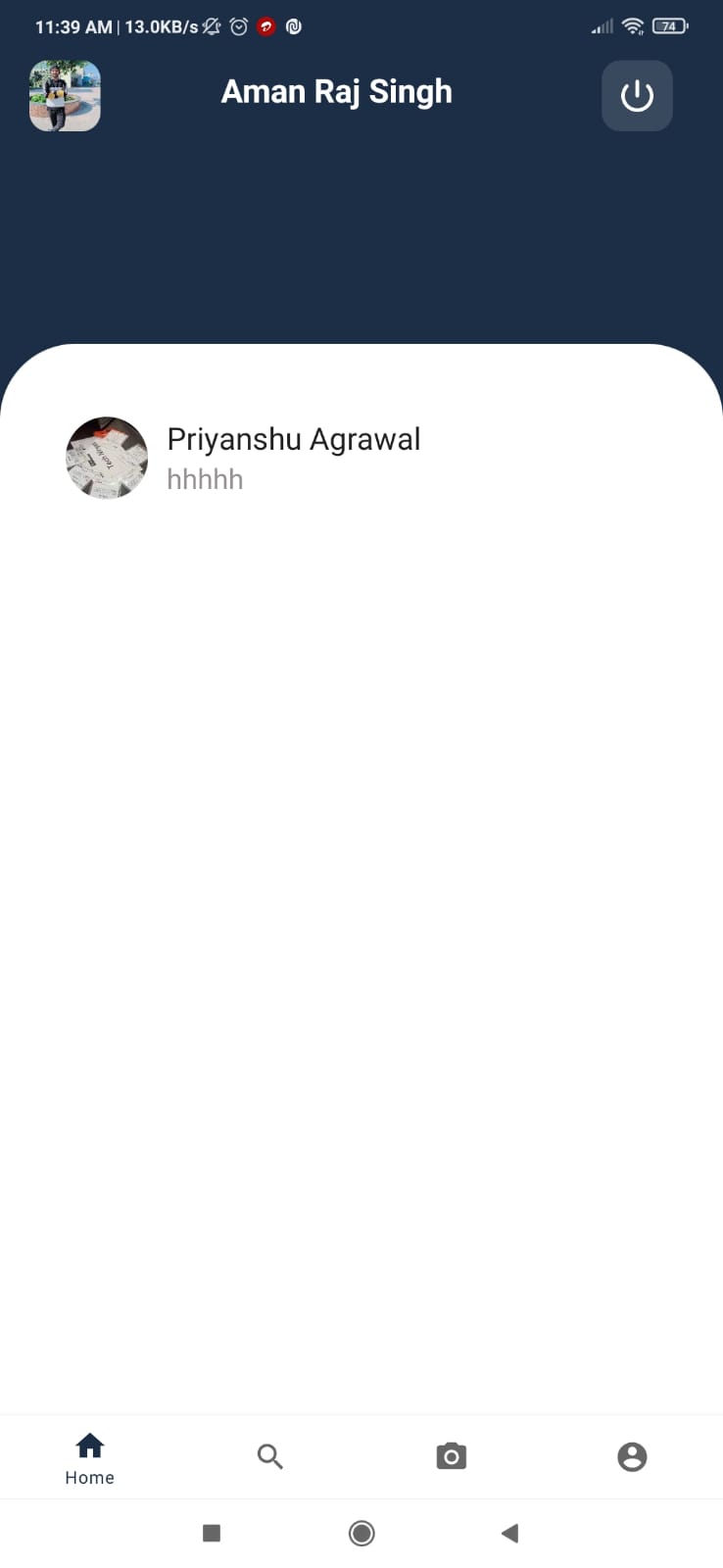
Splash Screen



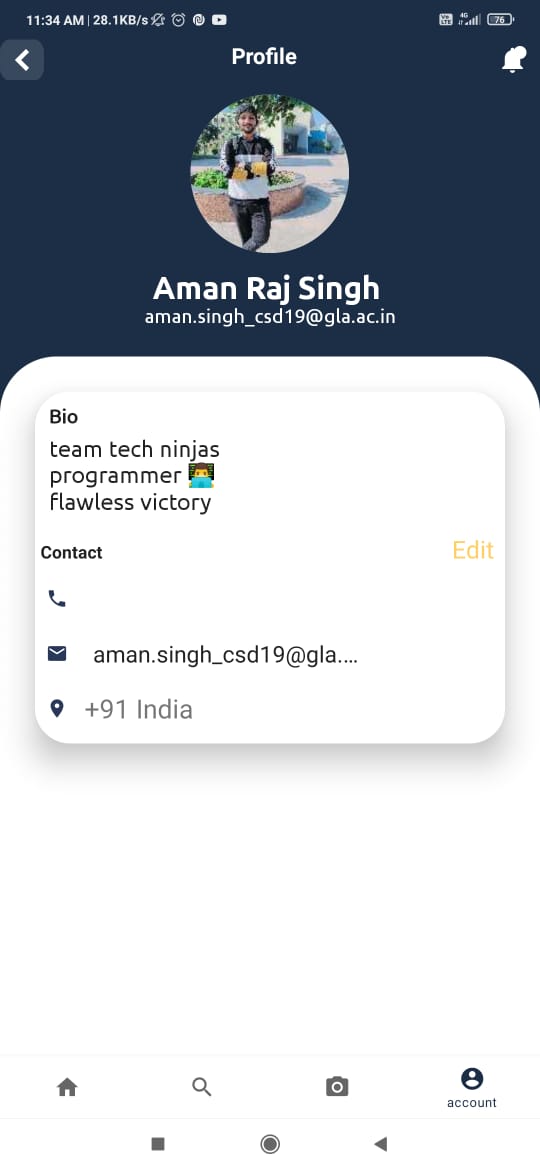
**Login Activity**



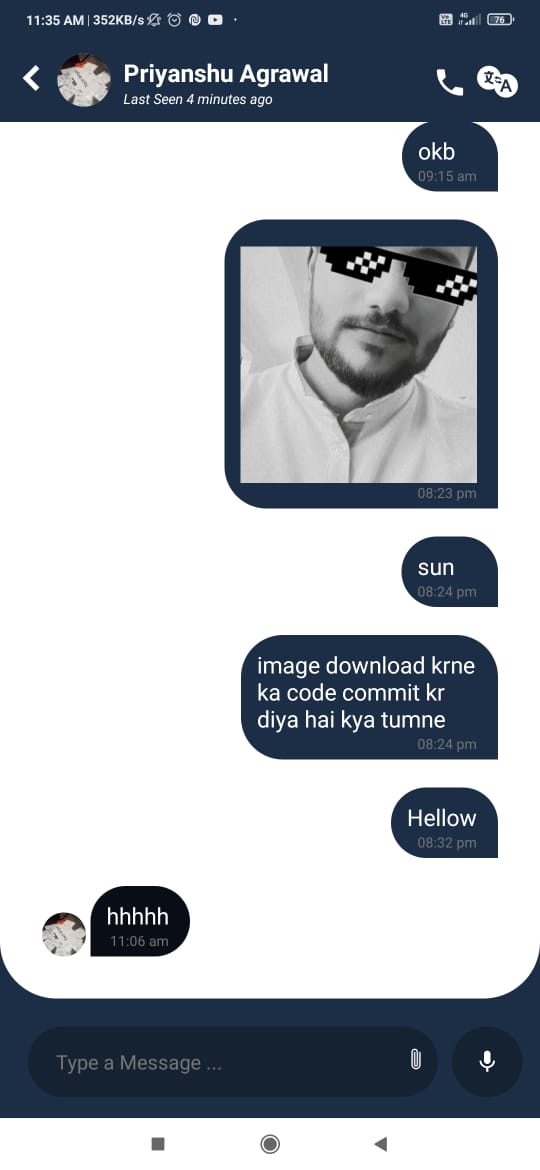
**Sign-up Activity**



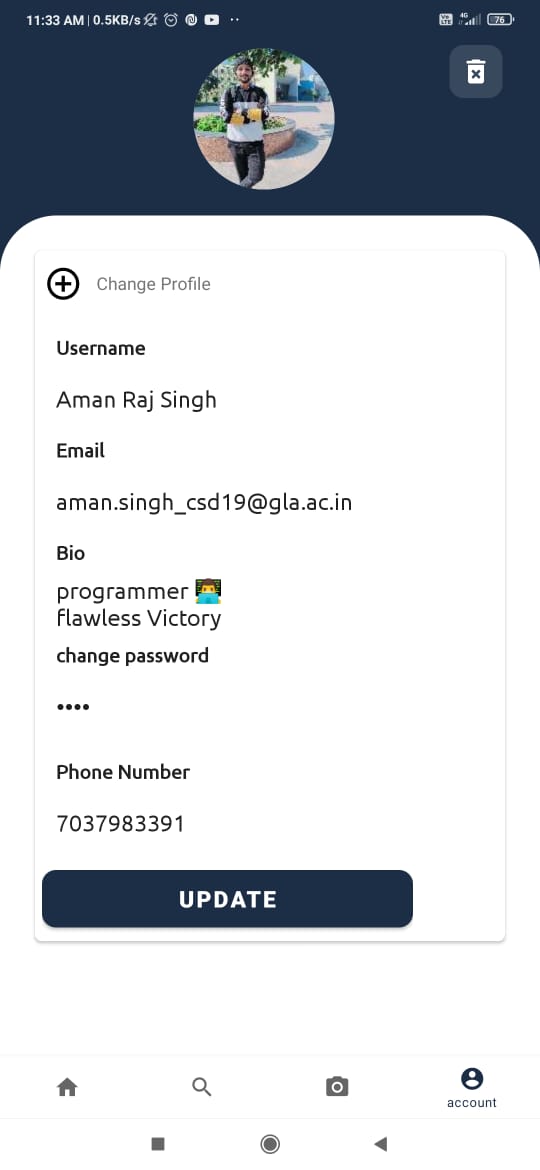
**Dashboard Activity**



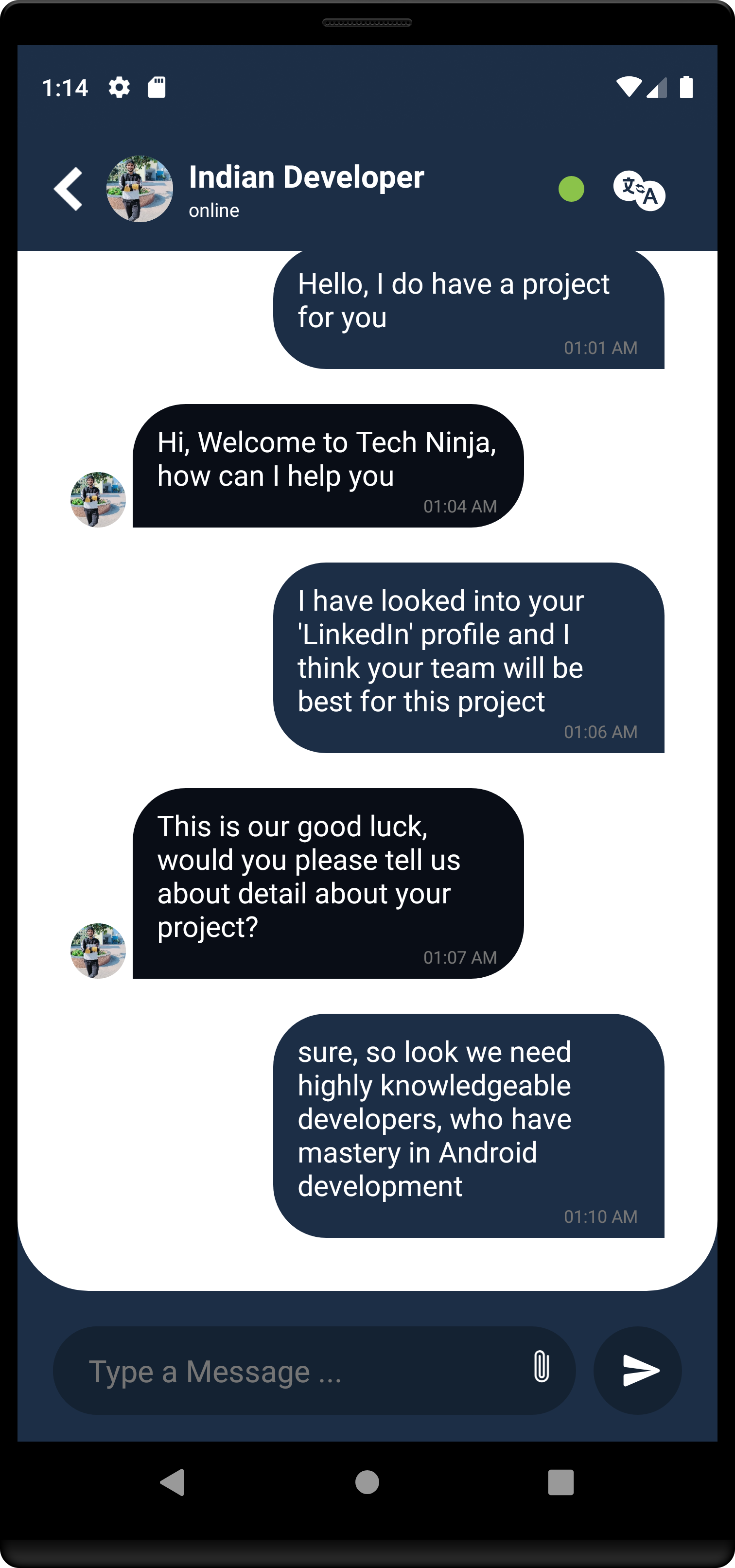
**Profile Activity**

****

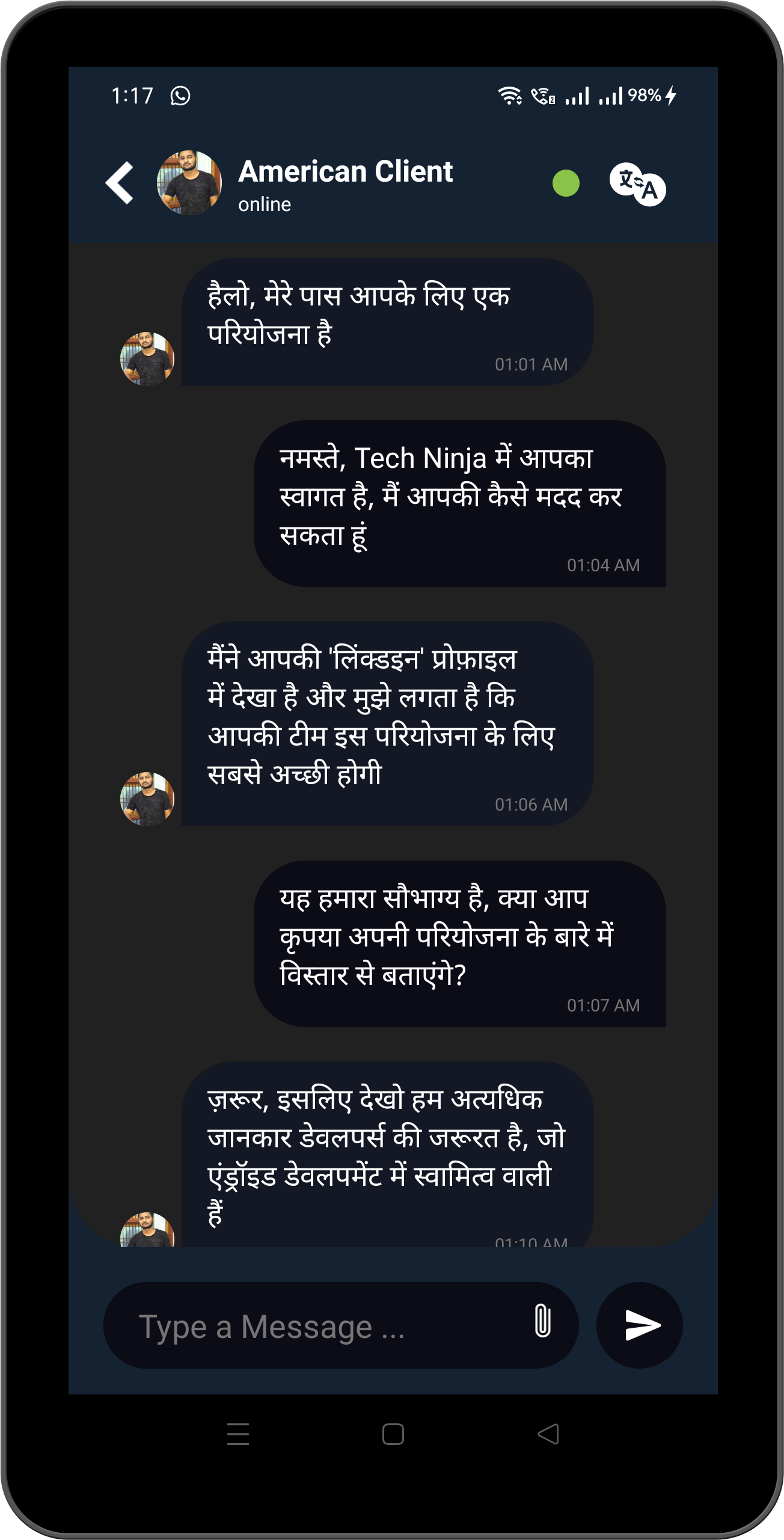
**Chat Activity**



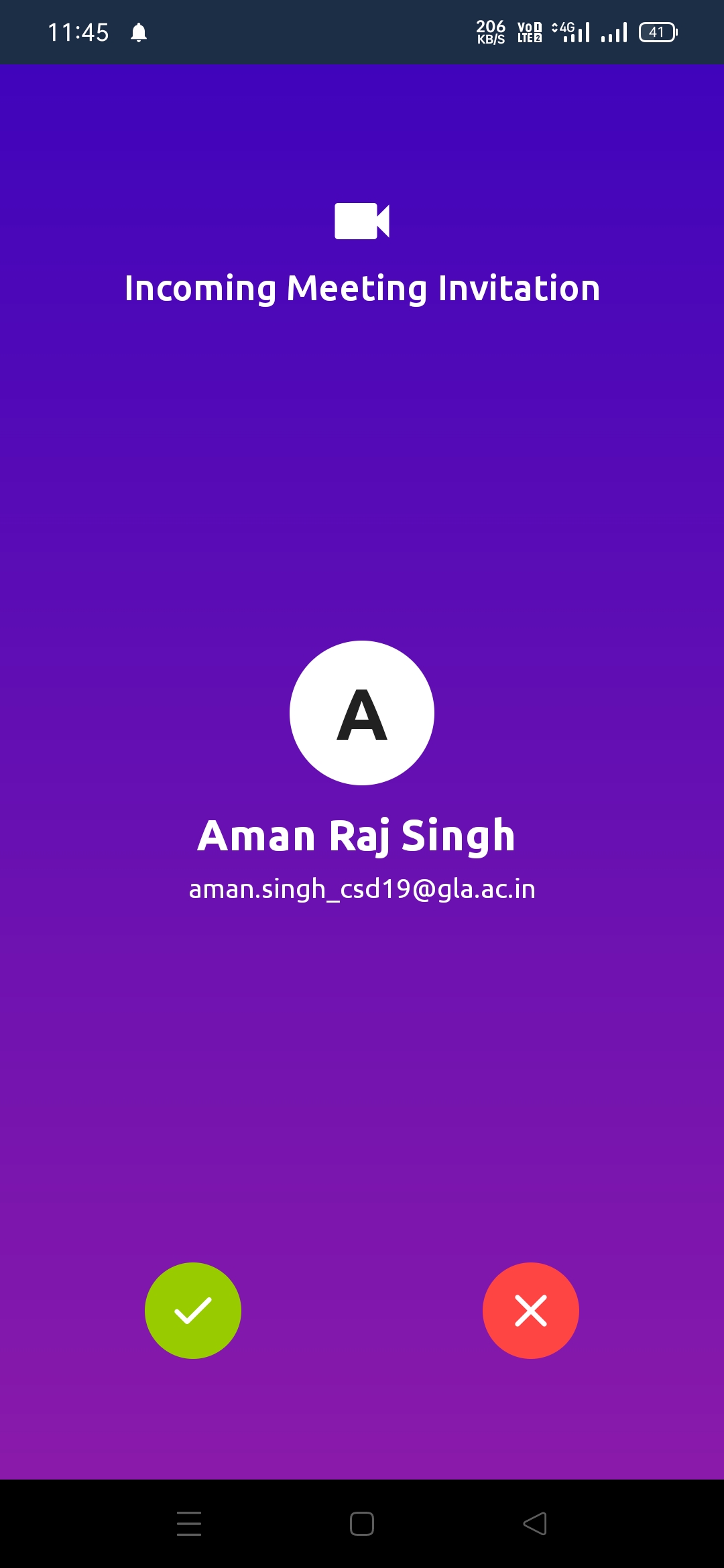
**Profile Update Activity**



Text Translate Example



Text Translate Example



Audio/Video Calling Activity

Video Calling Example

**CODING**

**Login Activity:**

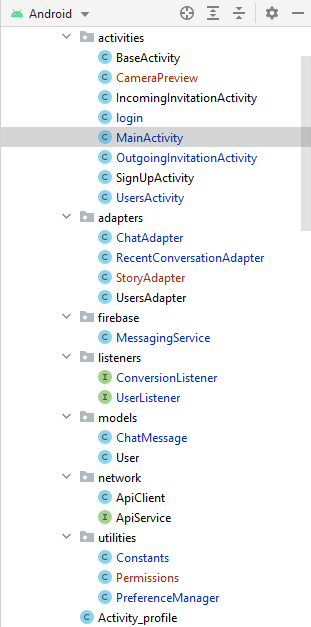
**public class** login **extends** AppCompatActivity {  
  
 **BottomNavigationView BottomNavigationView**;  
 **private** LoginBinding **binding**;  
 **private** PreferenceManager **preferenceManager**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 **preferenceManager** = **new** PreferenceManager(getApplicationContext());  
 **binding** = LoginBinding.*inflate*(getLayoutInflater());  
 setContentView(**binding**.getRoot());  
 setListerners();  
 }  
  
  
  
 **private void** setListerners() {  
 **binding**.**textCreateNewAccount**.setOnClickListener(v ->  
 startActivity(**new** Intent(getApplicationContext(),SignUpActivity.**class**)));  
  
 **binding**.**buttonsignIn**.setOnClickListener(v ->{  
 **if** (isValidSignInDetails()) {  
 signIn();  
 }  
 });  
 }  
  
 **private void** signIn() {  
 loading(**true**);  
 FirebaseFirestore database = FirebaseFirestore.*getInstance*();  
 database.collection(Constants.***KEY\_COLLECTION\_USERS***)  
 .whereEqualTo(Constants.***KEY\_EMAIL***, **binding**.**inputEmail**.getText().toString().trim())  
 .whereEqualTo(Constants.***KEY\_PASSWORD***, **binding**.**inputpassword**.getText().toString())  
 .get()  
 .addOnCompleteListener(task -> {  
 **if** (task.isSuccessful() && task.getResult() != **null** && task.getResult().getDocuments().size() > 0) {  
 DocumentSnapshot documentSnapshot = task.getResult().getDocuments().get(0);  
 **preferenceManager**.putBoolean(Constants.***KEY\_IS\_SIGNED\_IN***, **true**);  
 **preferenceManager**.putString(Constants.***KEY\_USER\_ID***, documentSnapshot.getId());  
 **preferenceManager**.putString(Constants.***KEY\_EMAIL***, documentSnapshot.getString(Constants.***KEY\_EMAIL***));  
 **preferenceManager**.putString(Constants.***KEY\_PASSWORD***, documentSnapshot.getString(Constants.***KEY\_PASSWORD***));  
 **preferenceManager**.putString(Constants.***KEY\_NAME***, documentSnapshot.getString(Constants.***KEY\_NAME***));  
 **preferenceManager**.putString(Constants.***KEY\_IMAGE***, documentSnapshot.getString(Constants.***KEY\_IMAGE***));  
 Intent intent = **new** Intent(getApplicationContext(), MainActivity.**class**);  
 intent.addFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK*** | Intent.***FLAG\_ACTIVITY\_CLEAR\_TASK***);  
 startActivity(intent);  
 } **else** {  
 loading(**false**);  
 showToast(**"Unable to sign in"**);  
 }  
 });  
 }  
  
 **private void** loading (Boolean isLoading){  
 **if**(isLoading) {  
 **binding**.**buttonsignIn**.setVisibility(View.***INVISIBLE***);  
 **binding**.**progressBar**.setVisibility(View.***VISIBLE***);  
 }**else** {  
 **binding**.**progressBar**.setVisibility(View.***INVISIBLE***);  
 **binding**.**buttonsignIn**.setVisibility(View.***VISIBLE***);  
 }  
 }  
  
 **private void** showToast(String message) {  
 Toast.makeText(getApplicationContext(), message, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 **private** Boolean isValidSignInDetails(){  
 **if**(**binding**.**inputEmail**.getText().toString().trim().isEmpty()) {  
 showToast(**"Enter email"**);  
 **return false**;  
 }**else if**(!Patterns.***EMAIL\_ADDRESS***.matcher(**binding**.**inputEmail**.getText().toString().trim()).matches()){  
 showToast(**"Enter valid email"**);  
 **return false**;  
 }**else if**(**binding**.**inputpassword**.getText().toString().trim().isEmpty()){  
 showToast(**"Enter Password"**);  
 **return false**;  
 }**else**{  
 **return true**;  
 }  
 }

**Sign Up Activity:**

**public class** SignUpActivity **extends** AppCompatActivity {  
  
 **private** String **encodedImage**;  
 **private** SignupBinding **binding** ;  
 **private** PreferenceManager **preferenceManager**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 **binding** = SignupBinding.*inflate*(getLayoutInflater());  
 setContentView(**binding**.getRoot());  
 **preferenceManager** = **new** PreferenceManager(getApplicationContext());  
 setListerners();  
 }  
*// this is for Signup Button If details wass correct then Signup Button Works othervide Not* **private void** setListerners() {  
 **binding**.**textSignIn**.setOnClickListener(v -> onBackPressed());  
 **binding**.**buttonsignUp**.setOnClickListener(v -> {  
 **if** (isValidSignUpDetails()){  
 SignUp();  
 }  
 });  
  
*// this is for add image click and redirect to the image folder* **binding**.**layoutImage**.setOnClickListener(v -> {  
 Intent intent = **new** Intent(Intent.***ACTION\_PICK***, MediaStore.Images.Media.***EXTERNAL\_CONTENT\_URI***);  
 intent.addFlags(Intent.***FLAG\_GRANT\_READ\_URI\_PERMISSION***);  
 **pickImage**.launch(intent);  
 });  
 }  
  
 **public void** gologin(View view){  
 Intent i = **new** Intent(**this**, login.**class**);  
 startActivity(i);  
 }  
  
 *//this used for print toast in a string* **private void** showToast(String message){  
 Toast.*makeText*(getApplicationContext(),message,Toast.***LENGTH\_SHORT***).show();  
 }  
  
 *//This is for the signup details send to firestore databse .* **private void** SignUp(){  
 loading(**true**);  
 FirebaseFirestore database = FirebaseFirestore.*getInstance*();  
 HashMap<String, Object> user = **new** HashMap<>();  
 user.put(Constants.***KEY\_NAME***,**binding**.**inputName**.getText().toString());  
 user.put(Constants.***KEY\_EMAIL***,**binding**.**inputEmail**.getText().toString());  
 user.put(Constants.***KEY\_PASSWORD***,**binding**.**inputpassword**.getText().toString();  
 user.put(Constants.***KEY\_IMAGE***, **encodedImage**);  
 database.collection(Constants.***KEY\_COLLECTION\_USERS***)  
 .add(user).addOnSuccessListener(documentReference -> {  
 loading(**false**);  
 **preferenceManager**.putBoolean(Constants.***KEY\_IS\_SIGNED\_IN***, **true**);  
 **preferenceManager**.putString(Constants.***KEY\_USER\_ID***, documentReference.getId());  
 **preferenceManager**.putString(Constants.***KEY\_NAME***, **binding**.**inputName**.getText().toString());  
 **preferenceManager**.putString(Constants.***KEY\_IMAGE***, **encodedImage**);  
 Intent intent = **new** Intent(getApplicationContext(), OtpSendActivity.**class**);  
 intent.addFlags(Intent.***FLAG\_ACTIVITY\_NEW\_TASK*** | Intent.***FLAG\_ACTIVITY\_CLEAR\_TASK***);  
 startActivity(intent);  
 })  
 .addOnFailureListener(exception ->{  
 loading(**false**);  
 showToast(exception.getMessage());  
 });  
 }  
  
 *// this is for image frame is equal to the given frame in signup page .* **private** String encodeImage (Bitmap bitmap){  
 **int** previewWidth = 150;  
 **int** previewHeight= bitmap.getHeight() \* previewWidth / bitmap.getWidth();  
 Bitmap previewBitmap = Bitmap.*createScaledBitmap*(bitmap, previewWidth, previewHeight, **false**);  
 ByteArrayOutputStream byteArrayOutputStream = **new** ByteArrayOutputStream();  
 previewBitmap.compress(Bitmap.CompressFormat.***JPEG***, 50, byteArrayOutputStream);  
 **byte**[] bytes = byteArrayOutputStream.toByteArray();  
 **return** Base64.*encodeToString*(bytes, Base64.***DEFAULT***);  
 }  
  
 *// this is for add image in a sign up page ...?* **private final** ActivityResultLauncher<Intent> **pickImage** = registerForActivityResult(  
 **new** ActivityResultContracts.StartActivityForResult(),  
 result ->{  
 **if** (result.getResultCode() == ***RESULT\_OK***) {  
 **if** (result.getData() != **null**) {  
 Uri imageUri = result.getData().getData();  
 **try** {  
 InputStream inputStream = getContentResolver().openInputStream(imageUri);  
 Bitmap bitmap = BitmapFactory.*decodeStream*(inputStream);  
 **binding**.**imageProfile**.setImageBitmap(bitmap);  
 **binding**.**textAddImage**.setVisibility(View.***GONE***);  
 **encodedImage** = encodeImage(bitmap);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 );  
  
 *//here is end  
  
 // This is all About Enter details if details are correct order than signup successfull othervide show toast not succesfull  
 // if else condition in every methord* **private** Boolean isValidSignUpDetails(){  
 **if** (**encodedImage** == **null**){  
 showToast(**"Select Profile Picture"**);  
 **return false**;  
 }**else if** (**binding**.**inputName**.getText().toString().trim().isEmpty()){  
 showToast(**"Enter Name "**);  
 **return false**;  
 }**else if** (**binding**.**inputEmail**.getText().toString().trim().isEmpty()){  
 showToast(**"Enter Email"**);  
 **return false**;  
 }**else if** (!Patterns.***EMAIL\_ADDRESS***.matcher(**binding**.**inputEmail**.getText().toString()).matches()) {  
 showToast(**"Enter Valid Image"**);  
 **return false**;  
 }**else if** (**binding**.**inputpassword**.getText().toString().trim().isEmpty()) {  
 showToast(**"Enter Password "**);  
 **return false**;  
 }**else if** (**binding**.**inputConfirmpassword**.getText().toString().trim().isEmpty()){  
 showToast(**"Confirm Your Password"**);  
 **return false**;  
 }**else if** (!**binding**.**inputpassword**.getText().toString().equals(**binding**.**inputConfirmpassword**.getText().toString())){  
 showToast(**"Password & Confirm Password Must Be Same "**);  
 **return false**;  
 }**else** {  
 **return true**;  
 }  
 }  
  
 *// This is for Loading Button if we click on sigh up Loading will start* **private void** loading (**boolean** isLoading){  
 **if** (isLoading){  
 **binding**.**buttonsignUp**.setVisibility(View.***INVISIBLE***);  
 **binding**.**progressBar**.setVisibility(View.***VISIBLE***);  
 }**else** {  
 **binding**.**progressBar**.setVisibility(View.***INVISIBLE***);  
 **binding**.**buttonsignUp**.setVisibility(View.***VISIBLE***);  
 }  
 }

**Main Activity Java:**

**public class** MainActivity **extends** BaseActivity **implements** ConversionListener {  
 **private** ActivityMainBinding **binding**;  
 **private** PreferenceManager **preferenceManager**;  
 **private int REQUEST\_CODE\_BATTERY\_OPTIMIZATIONS** = 1;  
 **private** StoryAdapter **StoryAdapter**;  
 **private** String **conversionId** = **null**;  
 **private** List<ChatMessage> **conversations**;  
 **private** RecentConversationAdapter **conversationsAdapter**;  
 **private** FirebaseFirestore **database**;  
 BottomNavigationView **bottomNavigationView**;  
  
 *//hear is the bottom navigation bar of this activity* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 **binding** = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(**binding**.getRoot());  
 **preferenceManager** = **new** PreferenceManager(getApplicationContext());  
 loadUserDetails();  
 getToken();  
 init();  
 setListerners();  
 listenConversation();  
 checkForBatteryOptimization();  
  
  
  
 **bottomNavigationView** = findViewById(R.id.***bottomNavigation***);  
  
 **bottomNavigationView**.setOnNavigationItemSelectedListener(**new** BottomNavigationView.OnNavigationItemSelectedListener() {  
 @Override  
 **public boolean** onNavigationItemSelected(@NonNull MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** R.id.***menu\_search***:  
 startActivity(**new** Intent(getApplicationContext()  
 , UsersActivity.**class**));  
 overridePendingTransition(0, 0);  
 **return true**;  
 **case** R.id.***menu\_account***:  
 startActivity(**new** Intent(getApplicationContext()  
 , Activity\_profile.**class**));  
 overridePendingTransition(0, 0);  
 **return true**;  
 **case** R.id.***menu\_music***:  
 startActivity(**new** Intent(getApplicationContext()  
 , camera.**class**));  
 overridePendingTransition(0, 0);  
 **return true**;  
 **case** R.id.***menu\_home***:  
 **return true**;  
 }  
 **return false**;  
 }  
 });  
 }  
  
 **private void** init() {  
 **conversations** = **new** ArrayList<>();  
 **conversationsAdapter** = **new** RecentConversationAdapter(  
 **conversations**, **this**);  
 **binding**.**conversationRecycleView**.setAdapter(**conversationsAdapter**);  
 **database** = FirebaseFirestore.*getInstance*();  
 **conversationsAdapter**.notifyDataSetChanged();  
 }  
  
  
 **public void** goProfile() {  
 Intent intent = **new** Intent(MainActivity.**this**, Activity\_user\_profile.**class**);  
 Bundle extras = **new** Bundle();  
 extras.putString(**"token"**, **preferenceManager**.getString(Constants.***KEY\_PCM\_TOKEN***));  
 extras.putString(**"userType"**, **"self"**);  
 intent.putExtras(extras);  
 startActivity(intent);  
 }  
  
 **private void** setListerners() {  
 **binding**.**imageSignOut**.setOnClickListener(v -> signout());  
 **binding**.**imageProfile**.setOnClickListener(v -> goProfile());  
  
 }  
  
 **boolean doubleBackToExitPressedOnce** = **false**;  
  
 @Override  
 **public void** onBackPressed() {  
 **if** (**doubleBackToExitPressedOnce**) {  
 **super**.onBackPressed();  
 **return**;  
 }  
  
 **this**.**doubleBackToExitPressedOnce** = **true**;  
 Toast.*makeText*(**this**, **"Please click BACK again to exit"**, Toast.***LENGTH\_SHORT***).show();  
  
 **new** Handler(Looper.*getMainLooper*()).postDelayed(**new** Runnable() {  
  
 @Override  
 **public void** run() {  
 **doubleBackToExitPressedOnce**=**false**;  
 }  
 }, 2000);  
 }  
  
  
 **private void** loadUserDetails() {  
 **binding**.**textName**.setText(**preferenceManager**.getString(Constants.***KEY\_NAME***));  
 **byte**[] bytes = Base64.*decode*(**preferenceManager**.getString(Constants.***KEY\_IMAGE***), Base64.***DEFAULT***);  
 Bitmap bitmap = BitmapFactory.*decodeByteArray*(bytes, 0, bytes.**length**);  
 **binding**.**imageProfile**.setImageBitmap(bitmap);  
 }  
  
 **private void** showToast(String meassage) {  
  
 Toast.*makeText*(getApplicationContext(), meassage, Toast.***LENGTH\_SHORT***).show();  
 }  
  
 **private void** getToken() {  
 FirebaseMessaging.*getInstance*().getToken().addOnSuccessListener(**this**::updateToken);  
 }  
  
 **private void** updateToken(String token) {  
 **preferenceManager**.putString(Constants.***KEY\_PCM\_TOKEN***, token);  
 FirebaseFirestore database = FirebaseFirestore.*getInstance*();  
 DocumentReference documentReference =  
 database.collection(Constants.***KEY\_COLLECTION\_USERS***).document(  
 **preferenceManager**.getString(Constants.***KEY\_USER\_ID***)  
 );  
 documentReference.update(Constants.***KEY\_PCM\_TOKEN***, token)  
 .addOnFailureListener(e -> showToast(**"unable to update token"**));  
 }  
  
 **private void** signout() {  
 showToast(**"Signing out..."**);  
 FirebaseFirestore database = FirebaseFirestore.*getInstance*();  
 DocumentReference documentReference =  
 database.collection(Constants.***KEY\_COLLECTION\_USERS***).document(  
 **preferenceManager**.getString(Constants.***KEY\_USER\_ID***));  
 HashMap<String, Object> updates = **new** HashMap<>();  
 updates.put(Constants.***KEY\_PCM\_TOKEN***, FieldValue.*delete*());  
 documentReference.update(updates)  
 .addOnSuccessListener(unused -> {  
 **preferenceManager**.clear();  
 startActivity(**new** Intent(getApplicationContext(), login.**class**));  
 finish();  
 })  
 .addOnFailureListener(e -> showToast(**"Unable to sign out"**));  
 }  
  
 @Override  
 **public void** onConversionClicked(User user) {  
 Intent intent = **new** Intent(getApplicationContext(), ChatActivity.**class**);  
 intent.putExtra(Constants.***KEY\_USER***, user);  
 intent.putExtra(Constants.***KEY\_CONVERSATION\_ID***,user);  
 startActivity(intent);  
 }  
  
  
  
 **private void** listenConversation(){  
 **database**.collection(Constants.***KEY\_COLLECTION\_CONVERSATIONS***)  
 .whereEqualTo(Constants.***KEY\_SENDER\_ID***,**preferenceManager**.getString(Constants.***KEY\_USER\_ID***))  
 .addSnapshotListener(**eventListener**);  
 **database**.collection(Constants.***KEY\_COLLECTION\_CONVERSATIONS***)  
 .whereEqualTo(Constants.***KEY\_RECEIVER\_ID***,**preferenceManager**.getString(Constants.***KEY\_USER\_ID***))  
 .addSnapshotListener(**eventListener**);  
 }  
  
  
 **private final** EventListener<QuerySnapshot> **eventListener** = (value, error) -> {  
 **if**(error != **null**) {  
 **return**;  
 }  
 **if**(value != **null**) {  
 **for** (DocumentChange documentChange : value.getDocumentChanges()) {  
 **if**(documentChange.getType() == DocumentChange.Type.***ADDED***) {  
 String senderId = documentChange.getDocument().getString(Constants.***KEY\_SENDER\_ID***);  
 String receiverId = documentChange.getDocument().getString(Constants.***KEY\_RECEIVER\_ID***);  
 ChatMessage chatMessage = **new** ChatMessage();  
 chatMessage.**senderId** = senderId;  
 chatMessage.**receiverId** = receiverId;  
 **if** (**preferenceManager**.getString(Constants.***KEY\_USER\_ID***).equals(senderId)) {  
 chatMessage.**conversionImage** = documentChange.getDocument().getString(Constants.***KEY\_RECEIVER\_IMAGE***);  
 chatMessage.**conversionName** = documentChange.getDocument().getString(Constants.***KEY\_RECEIVER\_NAME***);  
 chatMessage.**conversionId**=documentChange.getDocument().getString(Constants.***KEY\_RECEIVER\_ID***);  
 } **else** {  
 chatMessage.**conversionImage** = documentChange.getDocument().getString(Constants.***KEY\_SENDER\_IMAGE***);  
 chatMessage.**conversionName** = documentChange.getDocument().getString(Constants.***KEY\_SENDER\_NAME***);  
 chatMessage.**conversionId** = documentChange.getDocument().getString(Constants.***KEY\_SENDER\_ID***);  
 }  
 chatMessage.**message** = documentChange.getDocument().getString(Constants.***KEY\_LAST\_MESSAGE***);  
 chatMessage.**dateObject** = documentChange.getDocument().getDate(Constants.***KEY\_TIMESTAMP***);  
 **conversations**.add(chatMessage);  
 **conversationsAdapter**.notifyDataSetChanged();  
 }**else if** (documentChange.getType() == DocumentChange.Type.***MODIFIED***) {  
 **for** (**int** i = 0; i < **conversations**.size(); i++) {  
 String senderId = documentChange.getDocument().getString(Constants.***KEY\_SENDER\_ID***);  
 String receiverId = documentChange.getDocument().getString(Constants.***KEY\_RECEIVER\_ID***);  
 **if** (**conversations**.get(i).**senderId**.equals(senderId) && **conversations**.get(i).**receiverId**.equals(receiverId)) {  
 **conversations**.get(i).**message** = documentChange.getDocument().getString(Constants.***KEY\_LAST\_MESSAGE***);  
 **conversations**.get(i).**dateObject** = documentChange.getDocument().getDate(Constants.***KEY\_TIMESTAMP***);  
 **break**;  
 }  
 }  
 }  
 }  
 Collections.*sort*(**conversations**,(obj1, obj2) -> obj2.**dateObject**.compareTo(obj1.**dateObject**));  
 **conversationsAdapter**.notifyDataSetChanged();  
 **binding**.**conversationRecycleView**.smoothScrollToPosition(0);  
 **binding**.**conversationRecycleView**.setVisibility(View.***VISIBLE***);  
 **binding**.**progressBar**.setVisibility(View.***GONE***);  
 }  
 };  
  
 **public void** onConversionLongClicked(**int** position ,User user) {  
 Toast.*makeText*(MainActivity.**this**, **conversionId**, Toast.***LENGTH\_SHORT***).show();  
 AlertDialog dialog = **new** AlertDialog.Builder(**this**)  
 .setTitle(**"Waring"**)  
 .setMessage(**"Are you Sure You want to Delete This Chat"**)  
 .setPositiveButton(**"ok"**, **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
*// Toast.makeText(MainActivity.this, conversionId, Toast.LENGTH\_SHORT).show();* FirebaseFirestore database = FirebaseFirestore.*getInstance*();  
 DocumentReference documentReference =  
 database.collection(Constants.***KEY\_COLLECTION\_CONVERSATIONS***).document(**conversionId**);  
 documentReference.delete().addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if** (task.isSuccessful()) {  
 **conversations**.remove(position);  
 **conversationsAdapter**.notifyItemRemoved(position);  
  
 dialog.dismiss();  
 } **else** Toast.*makeText*(MainActivity.**this**, **"Unable To Delete"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
 }).setNegativeButton(**"Cancel"**, **null**)  
 .show();  
 }  
  
  
 **private void** checkForBatteryOptimization(){  
 **if** (Build.VERSION.***SDK\_INT***>=Build.VERSION\_CODES.***M***){  
 PowerManager powerManager = (PowerManager) getSystemService(***POWER\_SERVICE***);  
 **if** (!powerManager.isIgnoringBatteryOptimizations(getPackageName())){  
 AlertDialog.Builder builder = **new** AlertDialog.Builder(MainActivity.**this**);  
 builder.setTitle(**"warning"**);  
 builder.setMessage(**"Battery optimization is enable It can interrupt running Background service "**);  
 builder.setPositiveButton(**"disable"**, (dialog, which) -> {  
 Intent intent = **new** Intent(Settings.***ACTION\_IGNORE\_BATTERY\_OPTIMIZATION\_SETTINGS***);  
 startActivityForResult(intent , **REQUEST\_CODE\_BATTERY\_OPTIMIZATIONS**);  
 });  
 builder.setNegativeButton(**"Cancel"**, (dialog, which) -> dialog.dismiss());  
 builder.create().show();  
 }  
 }  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, @Nullable Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
 **if**(requestCode==**REQUEST\_CODE\_BATTERY\_OPTIMIZATIONS**);  
 checkForBatteryOptimization();  
 }  
 **protected void** onResume() {  
 **super**.onResume();  
 **bottomNavigationView**.setSelectedItemId(R.id.***menu\_home***);  
 }  
  
 **private void** Loading(Boolean isloading) {  
 **if** (isloading) {  
 **binding**.**progressBar**.setVisibility(View.***VISIBLE***);  
 } **else** {  
 **binding**.**progressBar**.setVisibility(View.***INVISIBLE***);  
 }  
 }



**Java Activity’s**

There are all java activity’s which has been used in Vaarta Android application these have all Database Connectivity and database Adapter and all users models.

**Coding Standards:**

1. Introduction.. 2

2. Synchronization.. 2

3. Comments. 2

4. Transactions. 2

5. Servlets coding.. 3

6. DB Connection Pooling.. 4

**1.**    **Introduction**

This document describes a set of standards for developing programs based on J2EE architecture It is meant to be used not only by programmers specifically writing Java code, but also by programmers creating programs which automatically *generate* Java code.

**2.**    **Synchronization**

* Use the code or method synchronized whenever sees the method/code cannot be used asynchronously, when a particular value computed in a method/codes input to the next object call
* Usage of synchronous will affect the performance at the same time we must use in certain cases otherwise system will malfunction. Choose carefully and do not miss to use the synchronous key word wherever required.
* You can find the need (If you missed in design) if application is giving wrong output under load
* This happens because unsynchronized changes to shared data
* Use synchronize access to shared data
* But first try to avoid use of access to shared data if not possible then use synchronous keyword for synchronization.

**3.**    **Comments**

* Must use comments in the class, make sure you have used right comments, comments should not be confusing, must explain the underneath code implementation; please refer to i2labs Coding standards.

**Technology Trends**

**Back End (JAVA)**

**INTRODUCTION:**

Java was conceived by James Gosling, Patrick Naughton, ChrisWarth, Ed Frank and Mike Sheridan at SUN Micro Systems Incorporation in 1991. It took 18 months to develop the first working version. This language was initially called “OAK”, but was renamed “JAVA” in 1995. Before the initial implementation of OAK in 1992 and the public announcement of Java in 1995, many more contributed to the design and evolution of the language.

**OVERVIEW OF JAVA**:

An Object Oriented Programming Language(OOPL) developed at Sun Microsystems. A Virtual Machine Run Time Environment that can be embedded in web browser (IE, NN). Java is a powerful but lean object oriented programming language.It has generated a lot of excitement because it makes it possible to program for Internet by creating applets, programs that can be embedded in web page.

The context of an applet is limited only by one’s imagination. For example, an applet can be an animation with sound, an interactive game or a ticker tape with constantly updated stock prices. Applets can be serious application like word processor or spreadsheet.

But Java is more than a programming language for writing applets. It is being used more and more for writing standalone applications as well. It is becoming so popular that many people believe it will become standard language for both general purpose and Internet programming. There are many buzzwords associated with Java, but because of its spectacular growth in popularity, a new buzzword has appeared ubiquitous. Indeed, all indications are that it will soon be everywhere.

***Java builds on the strength of C++. It has taken the best features of C++ and discarded the more problematic and error prone parts. To this lean core, it has added garbage collection (automatic memory management), multithreading (the capacity for one program to do more than one thing at a time), security capabilities. The result is simple, elegant, powerful and easy to use.***

Java is actually a platform consisting of three components:

* Java Programming Language.
* Java Library of Classes and Interfaces.
* Java Virtual Machine.

It also has a Standardized set of Packages (Class, Interfaces):

* Creating Graphical User Interfaces
* Controlling Multimedia Data
* Communicating over Network

The following sections will say more about these components:

#### FEATURES OF JAVA:

#### PORTABILITY:

One of the biggest advantages Java offers is that it is portable. An application written in Java will run on all the major platforms. Any computer with a Java based browser can run the applications or applets written in the Java Programming Language. A programmer no longer has to write one program to run on a Macintosh, another program to run on a windows machine, still another to run on a UNIX machine, and so on. In other words, with Java, developers write their programs only once. The virtual machine is what gives Java is cross platform capabilities.

Rather than being compiled into machine language, which is different for each operating systems and computer architecture, Java code is compiled into byte codes. With other languages, the program can understand. The problem is that other computers with different machine instruction set cannot understand that language. Java code, on the other hand is compiled into byte codes rather than a machine language. These byte codes go to the Java virtual machine, which executes them directly or translate them into the language that is understood by the machine running it. In Summary, these means that with the JDBC API extending Java, A programmer writing Java code can access all the major relational databases on any platform that supports the Java virtual machine.

**OBJECT\_ORIENTED**:

The Java programming language is object oriented, which makes program design focus on what you are dealing with rather than on how you are going to do something. This makes it more useful for programming in sophisticated projects because one can break the things down into understandable components. A big benefit is that these components can then be reused.

Object oriented languages use the paradigm of classes. In simplest term, a class includes both the data and the functions to operate on the data. You can create an instance of a class, also called an object, which will have all the data members and functionality of its class. Because of this, you can think of a class as being like template, with each object being a specific instance of a particular type of class.

The class paradigm allows one to encapsulate data so that specific data values are those using the data cannot see function implementation. Encapsulation makes it possible to make the changes in code without breaking other programs that use that code. If for example the implementation of a function is changed, the change is invisible to the programmer who invokes that function, and it does not affect his/her program, except hopefully to improve it. Java includes inheritance, or the ability to derive new classes from existing classes. The derived class, also called a subclass, inherits all the data and the functions of the existing class, referred to as the parent class. A subclass can add new data members to those inherited from the parent class. As far as methods are concerned, the subclass can reuse the inherited methods as it is, changed them, and/or add its own new methods.

**EASY**

In addition to being portable and object oriented, Java facilitates writing correct code. Programmers spend less time writing Java code and a lot less time debugging it. In fact, developers have reported slashing development time by as much as two thirds.

The following is a list of some of Java’s features that make it easier to write correct code:

**GARBAGE COLLECTION**:

Automatically takes care of allocating and reallocating memory, a huge potential source of errors. If an object is no longer being used (has no references to it), then it is automatically removed from memory. Dynamic binding ispossible and often very useful, but static binding with strict type checking is used when possible.

#### SIMPLICITY:

Makes Java easier to learn and use correctly. Java keep it simple by having just one way to do something instead of having several alternatives, as in some languages. Java also stays lean by not including multiple inheritance, which eliminates the errors and ambiguity that arise when you create a subclass that inherits from two or more classes. Java lets you add functionality to a class throws by the use of interfaces.

**JAVA INCLUDES A LIBRARY OF CLASSES AND INTERFACES:**

The Java platform includes an extensive class library so that programmers can use already existing classes, as it is, create subclasses to modify existing classes, or implement interfaces to augment the capabilities of classes.

Both classes and interfaces contain data members (fields) and functions (methods), but there are major differences. In a class, fields may be either variable or constant, and methods are fully implemented. In an interface, fields must be constants, and methods are just prototypes with no implementations. The prototypes give the method signature (the return type, the function name, and the number of parameters with the type for each parameter), but the programmer must supply implementations.

###### **To use an interface, a programmer defines a class, declares that it implements the Interface, and then implements all the methods in that interface as part of the class. These methods are implemented in a way that is appropriate for the class in which the methods are being used. Interfaces let one add functionality to a class and give a great deal of flexibility in doing it. In other words interfaces provide most of theadvantages of multiple inheritances without its disadvantages.**

***A package is a collection of related Java classes and interfaces. The following list, though not complete, gives example of some Java packages and what they cover.***

* Java.lang: The basic classes. This package is so basic that it automatically is included in any Java program. It includes classes dealing with numeric, strings, objects, runtime, security, and threads.
* Java.io: Classes that manages reading data from input streams and writing data to the output streams.
* Java.util: Miscellaneous utility classes, including generic data structures, bit sets, time, date, the string manipulation, random number generation, system properties, notification and enumeration of data structures.
* Java.net: Classes for network support.
* Java.awt: Classes that manage user interface components such as windows, dialog boxes, buttons, and checkboxes, lists, menus, scrollbars, and text fields, the “AWT” stands for Abstract Window Toolkit.
* Java.awt.image: Classes for managing image data, including color models, dropping color flittering, setting pixel values, and grabbing snapshots.
* Java.applet: The Applet class, which provides the ability to write applets, this package also includes several interfaces that, connect an applet to its documents and to its document and to its document and to recourses for playing audio.
* Java.sql: The JDBC API, classes and interfaces that access databases and send SQL Statements.

The first three packages listed, java.lang, java.io and java.util form the foundation, they are basic classes and interfaces for general-purpose programming.

Java development kit version1.1 added some new packages, with JDBC being one of them. Other new packages include such thing as Remote Method Invocation, Security and Java Beans, the new API for creating reusable components.

In Java, packages serve as the foundation for building other packages, as discussed in the following section.

#### EXTENSIBILITY:

A big plus for Java is the fact it can be extended. It was purposely written to be lean with the emphasis on doing what it does very well, instead of trying to do everything from the beginning; it was return so that extending it is very easy. Programmers can modify existing classes or write their own new classes or they can write a whole new package. The JDBC API, the java.sql package, is one example of a foundation upon which extensions are being built. Other extensions are being added or worked on in area such as multimedia, Internet Commerce, conferencing, and telephony.

In addition to extensions there are also main tools being developed to make existing capabilities easier to use. For example, there is already a tool that greatly Simplifies creating and laying out Graphical User Interfaces such as menus, Dialog boxes and buttons.

#### SECURITY:

It is important that a programmer not be able to write subversive code for Applications or applets. This is especially true with the Internet being used more and more extensively for services such as electronic commerce and electronic distribution of software and multimedia content.

The Java platform builds in security in four ways.

* The way memory is Allocated and laid out: In Java an object’s location in memory is not determined until The runtime, as opposed to C and C++, where the compiler makes memory layout Decisions. As the result, a programmer cannot look at a class definition and figure out how it might be laid out in memory. Also since, Java has no pointers, a programmer cannot forge pointers to memory.
* The way incoming code is checked: The Java virtual machine doesn’t trust any incoming code and subjects it to what is called byte code verification. The byte code Verifier, part of the virtual machine, checks that the format of incoming code is correct

Incoming code doesn’t forge pointers, it doesn’t violate access restrictions, it accesses objects what they are.

* The way classes are loaded: The Java byte code loader, another part of the virtual machine, whether classes loaded during program execution are local or from across a network. Imported classes cannot be substituted for built in classes, and built in classes cannot accidentally reference classes brought in over a network.
* The way access is restricted for untested code: The Java security manager allows user to restrict untested Java applets so that they cannot access the local network, files and other resources.

#### JAVA PERFORMS WELL:

Java performance is better than one might expect. Java has many advantages, such as having built in security and being interpreted as well as compiled, do have a cost attached to them. However, various optimizations have been built in, and the byte code Interpreter can run very fast the cost it doesn’t have to do any checking. As a result, Java has done quite respectably in performance tests. Its performance numbers for interpreting byte codes are usually more than adequate to run interactive graphical end user applications.

For situations that require unusually high performance, byte codes can be translated on the fly, generating the final machine code for the particular CPU on which the application is running at run time. High level interpreted scripting language generally offer great portability and fast prototyping but poor performance. Low level compiled language like C and C++ offer great performance but require large amounts of time for writing and debugging code because of problems with areas such as memory management, pointers and multiple inheritances. Java offers good performance with the advantages of high level languages but without the disadvantages of C and C++.

**ROBUST:**

The multi plat formed environment of the WEB places extraordinary demands on a program, because it must execute reliably in a variety of systems. Thus the ability to create robust programs was given a high priority in the design of Java. To gain reliability, Java restricts you in a few key areas to force you to find your mistakes early in program developments. At the same time, Java frees you from having to worry about many of the most common cause of programming errors. Because Java is strictly typed language, it checks your code at compile time. However, it also checks your code at run time. In fact, many hard to track down bugs that often turn up in hard to reproduce runtime situations are simply impossible to create in Java. Knowing that what you have written will behave in a predictable way under diverse conditions is a key feature of Java to understand how Java robust.

Consider two main reasons for program failure:

* Memory management mistakes and mishandled exceptional conditions (run time errors).
* Memory management can be difficult, tedious task in traditional programming environments.

For example in C/C++ the programmer must manually allocate and free all dynamic memory. This sometimes leads to problems. For example some programmers some times forget the free memory that has been previously allocated. Or worse, they may free some memory that another part of their code is still using. Java virtually eliminates these problems by managing memory allocations and reallocations. Java helps in this area by providing object oriented exception handling. In a well-written Java a program should manage program all run time errors.

### SCALABILITY

Java platform is designed to scale well, from portable consumer electronic devices to powerful desktop and server machines. The virtual machine takes a small foot print and Java byte code is optimized to be small and compact. As a result, Java accommodates the need for low storage and for low bandwidth transmission over the Internet. In addition the Java operating system offers a standalone Java platform that eliminates host operating system overhead while still supporting the full Java platform. API makes Java ideal for low cost network computers whose sole purpose is to access the Internet.

#### MULTITHREADED:

Multithreading is simply the ability of a program to do more than one thing at a time. For example an application could be faxing a document at the same time it is printing another document. Or a program could process new inventory figures while it maintains a feed for current prices. Multithreading is particularly important in multimedia. A multimedia program might often be running a movie, running a audio track and displaying text all at the same time.

#### IMPORTANT TO THE INTERNET:

The Internet helped catapult Java to the forefront of programming and Java in turn has a profound effect on the Internet. The reason is simple. Java expands the universe of objects that can move about freely in cyberspace. In a network, there are two broad categories of objects transmitted between the server, your personal computer, passive information and dynamic, active programs. For example, when you read your e-mail, you are viewing passive data. Even when you download a program, the program’s code is still only passive data until you execute it. However, there is a second type of object that can be transmitted to your computer, a dynamic, self executing program. Such a program would be an active agent on the client computer, yet it would be initiated by the server. As desirable as dynamic, networked programs are, they also present serious problems in the areas of security and portability. Prior to Java cyberspace was effectively closed to half the entities that now live there. Java addresses these concerns and doing so, has opened the door to an exiting a new form of program.

**JAVA DATA BASE CONNECTIVITY:**

**INTRODUCTION:**

Java Database Connectivity (JDBC) is a front-end tool for connecting to a server to ODBC in that respect, However JDBC can connect only Java clients and it uses ODBC for the connectivity. JDBC is essentially a low-level application programming interface. It is called a low-level API since any data manipulation, storage and retrieval has to be done by the program itself. Some tools which provide a higher-level abstraction or expected shortly.

The next question that needs to be answered is why we need JDBC, once we have ODBC on hand. We can use the same ODBC to connect the entire database and ODBC is a proven technology. Problem for doing this is ODBC gives a ‘C’ language API, which uses pointers extensively. Since Java does not have any pointers and is object-oriented sun Microsystems, inventor of Java developed to suit its needs.

**REQUIREMENTS TO USE JDBC:**

To use JDBC you need a basic knowledge of database and SQL. Apart from this you need the jdk1.1 (Java Development Kit 1.1) or a version of Java since jdk1.1 and above come bundled with JDBC software.

After that you need to have a back-end database engine for which a JDBC driver is available. When JDBC drivers are not available JDBC-ODBC bridge drivers are used to access the database through ODBC. Back-end is not need when JDBC driver is capable of storing and retrieving the data itself, or if JDBC-ODBC bridge and the ODBC driver can be store and retrieve the information.

###### **DATABASE MODELS:**

JDBC and accessing the database through applets, and JDBC API via an intermediate server resulted in a new type of database model which is different from the client-servers through which the request should go it is named as single tier, two tier and multi tier architecture.

##### **JDBC DRIVER TYPES:**

***The JDBC drivers that we are aware of at this time fit into one of four categories:***

* JDBC-ODBC Bridge plus ODBC driver: The Java Soft bridge product provides JDBC access via ODBC drivers. Note that ODBC binary code and in many cases database client code must be loaded on each client machine that uses this driver. As a result, this kind of driver is most appropriate on a corporate network where client installations are not a major problem, or for application server code written in Java in three-tier architecture.
* Native-API partly-Java driver: This kind of driver converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS. Note that, like the bridge driver, this style of driver requires that some binary code be loaded on each client machine.
* JDBC-Net all-Java driver: This driver translates JDBC calls into a DBMS-independent net protocol that is then translated to a DBMS protocol by server. This net server middle ware is able to connect its all-Java clients to many different databases. The specific protocol used depends on the vendor. In general, this is the most flexible JDBC alternative. It is likely that all vendors of this solution will provide products suitable for Internet use. In order for these products to also support Internet access, they must handle the additional requirements for security, access through firewalls, etc., that the Web imposes. Several vendors are adding JDBC drivers to their existing database middle ware products.
* Native-protocol all-Java driver: This kind of driver converts JDBC calls into the network protocol used by DBMS directly. This allows a direct call from the client machine to the DBMS server and is a practical solution for Internet access. Since many of these protocols are proprietary, the database vendors themselves will be the primary source. Several database vendors have these in progress.

Eventually, we expect the last two drivers will be preferred way to access database from JDBC. And the first two driver categories are interim solutions where direct all-Java drivers are not yet available. The last driver is in some sense the ideal one. However, there are many cases where JDBC-Net all-Java driver may be preferable. For example, where a thin DBMS- independent client is desired, or if a DBMS-independent protocol is standardized and implemented directly by many DBMS vendors.

**TESTING AND IMPLEMENTATION**

**6.1 Test Case Description**

**6.1.1 Definition**

Testingistheprocessofdetectingerrors.Testingperformsavery critical role for quality assurance and for ensuring thereliability of software .Theresults of testingareusedlateronduringmaintenancealso.

6.1.2 Psychology of Testing

Theaimoftestingisoftentodemonstrate thatprogramworksbyshowing thatithas noerrors.The basicpurposeoftestingphaseistodetecttheerrorsthatmaybepresentin theprogram.Henceone should not start testingwiththeintentofshowingthata programworks,buttheintentshouldbetoshowthataprogramdoesn’t work.Testing is theprocessof executingaprogramwiththeintentoffindingerrors.

6.1.3 Testing Objectives

The main objectiveoftesting istouncoverahostoferrors,systematicallyandwith minimumeffortandtime.Statingformally,we cansay,

• Testingisaprocessof executingaprogramwiththeintentoffindingan error.

• Asuccessfultest isonethat uncoversan asyet undiscoverederror.

• A goodtestcaseisonethat hasahighprobabilityoffindingerror,ifitexists.

• Thetestsareinadequatetodetect possiblypresent errors.

• Thesoftwaremoreorlessconfirmstothequalityandreliablestandards.

**6.2Types of Testing**

**6.2.1 Unit Testing**

Unittestingfocusesverificationeffortonthesmallestunitofsoftwarei.e.themodule.Usingthe detailed design and the processspecificationstestingis doneto uncovererrorswithintheboundaryofthemodule.Allmodulesmustbesuccessfulinthe unittest beforethestartoftheintegrationtestingbegins.

Inthisprojecteachservicecanbethoughtofamodule.Therearethreebasic modules.Givingdifferentsetsofinputshastestedeachmodule.Whendeveloping a moduleaswellasfinishingthedevelopmentsothateachmoduleworkswithoutany error.Theinputsarevalidatedwhen acceptingfromtheuser.

Inthisapplicationdeveloperteststheprogramsupassystem.Softwareunitsin asystemarethemodulesandroutinesthatareassembledandintegratedtoforma specificfunction.Unittesting isfirstdoneonmodules, independent ofoneanotherto locateerrors.Thisenablestodetecterrors.Through this, errorsresultingfrominteraction betweenmodulesinitiallyavoided.

**6.2.2Link Testing**

Link testingdoesnottestsoftware butrathertheintegrationofeachmodulein system.Theprimaryconcernisthecompatibilityofeachmodule.TheProgrammer tests wheremodulesaredesignedwithdifferentparameters,length,typeetc.

**6.2.3 Integration Testing**

After theunittesting wehavetoperformintegration testing.Thegoalhereisto seeifmodulescanbe integratedproperly,the emphasisbeingon testinginterfaces betweenmodules. This testingactivitycanbeconsideredastestingthedesignandhence theemphasisontestingmoduleinteractions.Inthisprojectintegrating allthemodulesformsthemainsystem. When integrating allthemodulesIhavecheckedwhethertheintegration effectsworkingofany oftheservicesbygivingdifferentcombinationsof inputswithwhichthetwoservicesrun perfectlybeforeintegration.

**6.2.4 Code Testing**

Thisstrategy examinesthelogicoftheprogram.Tofollowthismethodwedeveloped sometestdatathatresultedinexecutingevery instruction intheprogramandmodule i.e. every pathistested.Systemsarenotdesignedasentirenoraretheytestedassingle systems.Toensurethatthecoding isperfecttwotypesoftestingisperformedorforthat matterisperformed orthatmatterisperformedorforthatmatterisperformed onall systems.

**6.2.5 Validation Testing**

Validation can be defined in many ways, but a simple definitions is that validation succeeds when software functions in a manner that can be reasonable expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirements.

**6.2.6 Black Box Testing**

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

**6.2.7 White Box Testing**

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current me

**6.6 System Documentation:**

All design and test documentation should be finalized and entered in the library for future reference. The library is the central location for maintenance of the new system. The format, organization, and language of each documentation should be in line with system standards.

**TEST CASES**

**Module: Login**

**Filename: login.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Input** | **Actual Output** | **Obtained Output** | **Desc** |
| **Valid Login** | **User Id, Password** | **Login Success** | **Login Success** | **Test Passed!**  **Control Transferred to Dashboard** |
| **Invalid Login** | **User Id, Password** | **Login Failed** | **Login Failed** | **Test Passed!**  **Try Again** |
| **Invalid Login** | **Null,Null** | **Login Failed** | **Login Failed** | **Test Passed!**  **Try Again** |

**Module: Sign up**

**Filename: signup.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Actual Output | Obtained Output | Desc |
| Register Users | Users Info and Password | Success | Success | Test Passed.  User register and associated with Password |
| Register Users | Users Info and Password | Failed | Failed | Test Passed.  Invalid or incomplete data. Try again. |

**Module : Chatting and Translation**

**Filename : ChatActivity.java**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Case | Input | Actual Output | Obtained Output | Desc |
| Message Send | Text Message  Text Translation | Success | Success | Test Passed.  Message And Notification send to user. |
| Message received | Text Message  Text Translation | Failed | Failed | Test Passed.  Internet Problem try to connect Internet |

**Future Scope:-**

i- It will satisfies the user requirements to initiate chats internationally.

ii- It will be easy to understand and use.

iii- It will have Material UI polished Interface.

iv- It will be maintained and optimized regularly.

v- User’s privacy will be kept safe.

**Limitations**

1. Dependency on the Software kit to maintain an alternative at all times.
2. When Internet Is slow API translate request will be Slow.
3. Send Document and audio file will have some latency.

**CONCLUSION**

**CONCLUSION**

The integration of chatting system for users operations will have a great potential to reducing operational errors, poor accessibility to record information, and poor security of data entry. Almost 90% published studies and reviews according to findings did not meet the rigorous quality standard thereby resulting in poorly generalized standards across all chatting system. The author did literature review based on articles, journals and the internet to gather facts about the current chatting system. Questionnaires and interviews were also used as important resources to obtain accurate information about the prevailing situation in the problem domain. This study is proven that the integration of the chatting system will help in reducing the insecurity in chatting system. The major problems with the current system has been analyzed and examined and a solution proposed to address the issues through proper identification and evaluation of methods, tools, and techniques used to develop solution.