Design and Implementation of a Real Time Chat Application.

Thesis Submitted in Partial Fulfilment of the Requirement
for the Degree of

B.Sc.
In
Computer Science [Software Engineering]
By

OBADJERE, Eric Nyerhovwo

To
The Department of Computer Science
Baze University, Abuja

DECEMBER, 2020

DECLARATION

This is to certify that this Thesis entitled Design and Implementation of a Real Time Chat Application, which is submitted by Eric Obadjere Nyerhovwo in partial fulfilment of the requirement for the award of degree for B.Sc. in Information Technology to the Department of Computer Science, Baze University Abuja, Nigeria, comprises of only my original work and due acknowledgment has been made in the text to all other materials used.

Date: 11 January 2021 Name of Student: Eric Obadjere Nyerhovwo

APPROVED BY …..…………………

Head

Dept. of Computer Science

CERTIFICATION

This is to certify that this Thesis entitled Design and Implementation of a Real Time Chat Application, which is submitted by Eric Obadjere Nyerhovwo in partial fulfilment of the requirement for the award of degree for B.Sc. in Information Technology to the Department of Computer Science, Baze University Abuja, Nigeria is a record of the candidate’s own work carried out by the candidate under my/our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Date: Supervisor: Dr Amit Mishra

Date: Supervisor: Mr Charles Isah Saidu

APPROVAL

This is to certify that the research work, Dental Management System and the subsequent preparation by Eric Obadjere Nyerhovwo with BU/18A/IT/3034 has been approved by the Department of Computer Science, Faculty of Computing and Applied Science, Baze University, Abuja, Nigeria.

By

Dr Amit Mishra Date
1st Supervisor

Mr Charles Isah Saidu Date
2nd Supervisor

Dr. Chandrashekhar Uppin Date
Head of Department

Prof M. B. Hammawa Date
Dean, Faculty of Computing and Applied Science

Prof. A. B. Garko Date
External Examiner

DEDICATION

I hereby dedicate this thesis (Design and Implementation of a chat application) to my mum "**Mrs Esther Obadjere**" who keep on pushing me through to always be the best, my classmates who made the years of study a favourable one, my friends who keep on supporting me and my lecturers who guided me to this point.

I also want to appreciate my Departmental HOD "**Dr. C. V Uppin**" for the assistance through the project and my project superiors "**Dr Amit Mishra**" and "**Mr Charles Isah Saidu**" for their guidance during the project design and Implementation stages.

ABSTRACT

CHATTY was created with the mind set to not just be another chat application but to add a level of clean UI (user interface) or a solid app structure function over a secure broadcast network. CHATTY is an effort for a more modern approach to internet security on a communication medium. The design of the UI (user interface) was greatly influenced by the Already exiting Chat Applications so as to give it users a fresh but familiar UI (user interface) design. The Implementation of CHATTY was based on a new but effective set up of flutter, node, hive, and MongoDB where flutter was for the app structure and UI (user interface) tool, node for the server setup, hive for flutter local database and MongoDB for the central network database. An end to end connection stream was used for data transfer from client to server and back to client. In conclusion the application worked well without a lag and having a 95% acceptance when tested with a potential user.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CPU	Central Processing Unit
ERD	Entity Relationship Diagram
IT	Information Technology

CHAPTER 1: INTRODUCTION

CHATY is a social-networking tool that leverages on technology advancement thereby allowing its users communicate and share media. It offers a wonderful one stop shop experience for keeping in touch with people you know. It can be used for messaging, placing voice messages, making voice and video calls, share updates and photos, share locations, enhance local socializing in pidgin English, play games and make monetary financial transactions.

BACKGROUND AND MOTIVATION

CHATY has the potential to become a widely used socializing app in Africa with specific focus on the West African sub region with a grass root pilot scheme take off in Nigeria due to the continent rich multi ethnic culture and a wide range of her special way of communication called pidgin English in Nigeria. The numerical strength of the Nigerian population which is estimated to be above 200million people presently affords a viable economic space as a rallying point for this application.

With the gradual acceptance of adult literacy gaining roots in Africa and the various government policies to drive education to the grass roots, CHATY will be a welcome development and very user friendly to the average Nigerian citizen among which includes the local peasant farmers, market women, artisans, the low-income earners, less privilege and the lower social stratification individuals who are numerous in population. The introduction of the Pidgin English feature is a primary drive and force of attraction as it helps break the rank of the academic division between those who claim the Queens English is the best, thereby creating fun and increased cross fertilization of ideas, with a resultant economic boost.

While emphasis must be made to fore runners of similar inventions, which cuts across technology divides, it must be well understood that the basic fundamental of the CHATY which is embedded in messaging service, evolve as an app where you can comfortably make financial transactions as well.

STATEMENT OF PROBLEMS

Starting any application or service has many problems but one of the main problems is which tool, language, stack or framework to build one's service or application on. As building a real time application has to do with slow latency message delivery which in turn means latency, data transfer size over the network must be as low as possible. Other problems include sms messaging, cross platform permissions for android and IOS.

AIMS AND OBJECTIVE.

MESSAGING

One of the primary use of CHATTY is messaging. Just like other social apps, you have a list of conversations that you're engaged in. This feature is pivotal as you can add people in a variety of ways aside the conventional way of details collection. When fully operational, you will be amazed how individuals will have to scan their phones during details collection.

This is made possible as each CHATTY user will have a unique barcode known as a QR code. One person can scan the other user's QR code to add them to CHATTY. Users of CHATTY can also use a phone number to add a person to their contact list and even search for people nearby.

CHATTY enormous social power will be let loosed as it will translate to become one of the main ways people will have to communicate in Nigeria and sub-Sahara Africa. Even when doing business, people will prefer CHATTY to email. With inbuilt plug ins that allows individuals to load pictures and videos and their contacts can make comments about them, then the barrier of environmental distance will be broken and knowledge will increase as people will be able to explore the beautiful landscape of Africa without really travelling to those regions physically.

FILE TRANSFER

With the sophistication design of the CHATTY app, individuals will be able to share files without size constrains ranging from images, videos, to large documents files like zip, dmg, and so on. As long as you have a smart phone with android or iOS running on it, it can be linked to your CHATTY app to achieve this.

The uniqueness of this app is that is works seamlessly with your use case for it and saves you the stress of uploading and downloading the file as the mobile device become a host you can download files from with the simple aid of your smart phone at download cost, thereby helping the world become a closer with ease.

SOCIAL BARRIER BREAKER

With the introduction of the Pidgin English feature, CHATTY is intended to ensure we all socialize within our limits and create more friends irrespective of our ranks and file. Just like the BBC pidgin that is bringing the news room closer to the lower strata, CHATTY hopes to rekindle the spirit of humanity in her natural state devoid of privileges some individuals have over others. In conceptualizing this app, careful thought and study has been given to a methodology of really defining the peaceful co-existence of human and, CHATTY will really serve as a means of promoting and forestalling unity as it helps in defining the place of equality of the human race devoid of sentiment, religion and ethnicity.

DELIVERABLES

There are 5 major things I hope to achieve with this application, which include.

I. Speed in usage

II. Easy and friendly UI

III. Privacy Protection

IV. Promoting Unity

V. Economic boost

PRE-DEVELOPMENT ANALYSIS

Kindly introduce the table from the SOURCE.

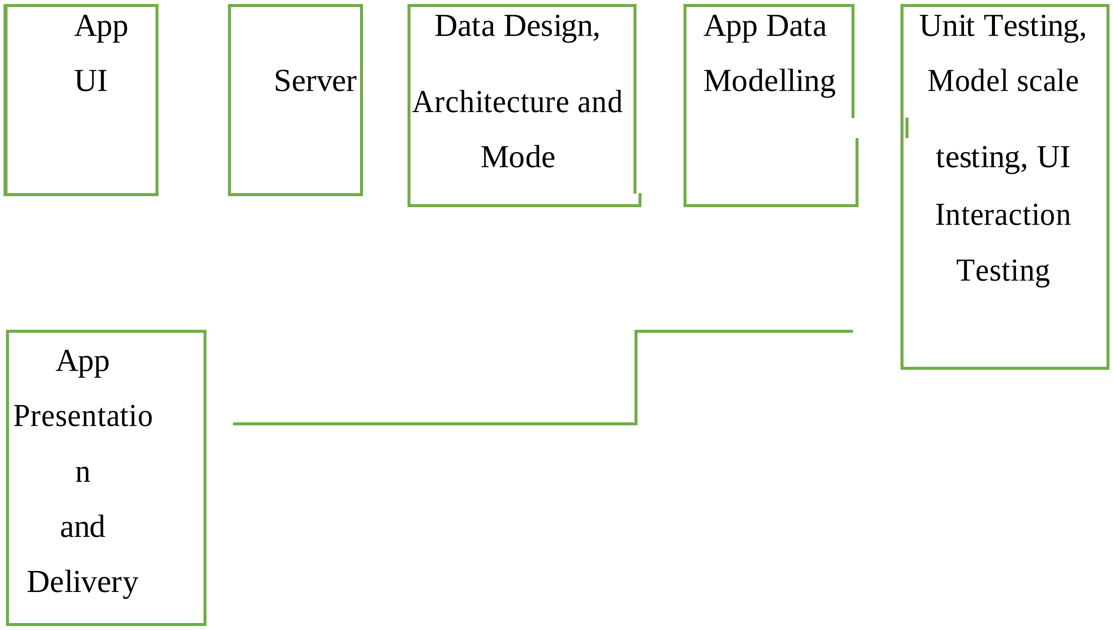

App UI
Server
Data Design, Architecture and Mode
App Data Modelling
Unit Testing, Model scale testing, UI Interaction Testing
App Presentation and Delivery

Figure 1.1: The source table

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Messaging apps now have more global users than traditional social networks—which mean they will play an increasingly important role in the distribution of digital information in the future. In 2016, over 2.5 billion people used at least one messaging app. That's one-third of the world's entire population, with users ranging from various age grades. Today, it's common place for offices to use a messaging app for internal communication in order to coordinate meetings, share pitch decks, and plan happy hours. And with the latest bot technology, chat apps are becoming a hub for employees to do work in their apps without leaving the chat console. For many people, chat apps are a given part of their workday. But how did these chat apps become so popular?

2.2 Historical Overview

2.2.1 INSTANT MESSAGING: CHILD OF THE 90'S

Chat apps (and their subsets, chat rooms) bring to remembrance images of the 1990s, with its dial-up internet and classic sitcoms, however, commercial chat apps date back to the 1980s. CompuServe released CB Simulator in 1980, and 1985 brought the launch of Commodore's Quantum Link (also known as Q-Link). An online service, it allowed multi-user chat, email, file sharing, and games.

If Q-Link sounds familiar, that's because it is: in 1991, the company changed its name to America Online (AOL). But AOL wouldn't launch its signature product, AOL Instant Messenger (AIM), until 1997. In the meantime, the Vodafone GSM network enabled the first SMS in 1992. And in 1996, ICQ launched as the first widely-adopted instant messaging platform.

The late 1990s brought dramatic changes in the chat app market. Both Yahoo! And MSN launched their own instant messengers (in 1998 and 1999, respectively), and AOL bought out competitor ICQ's parent company, Mirabilis, for a fee around $287 million upfront, with an additional $120 million paid out later. That's about $612.7 million today!

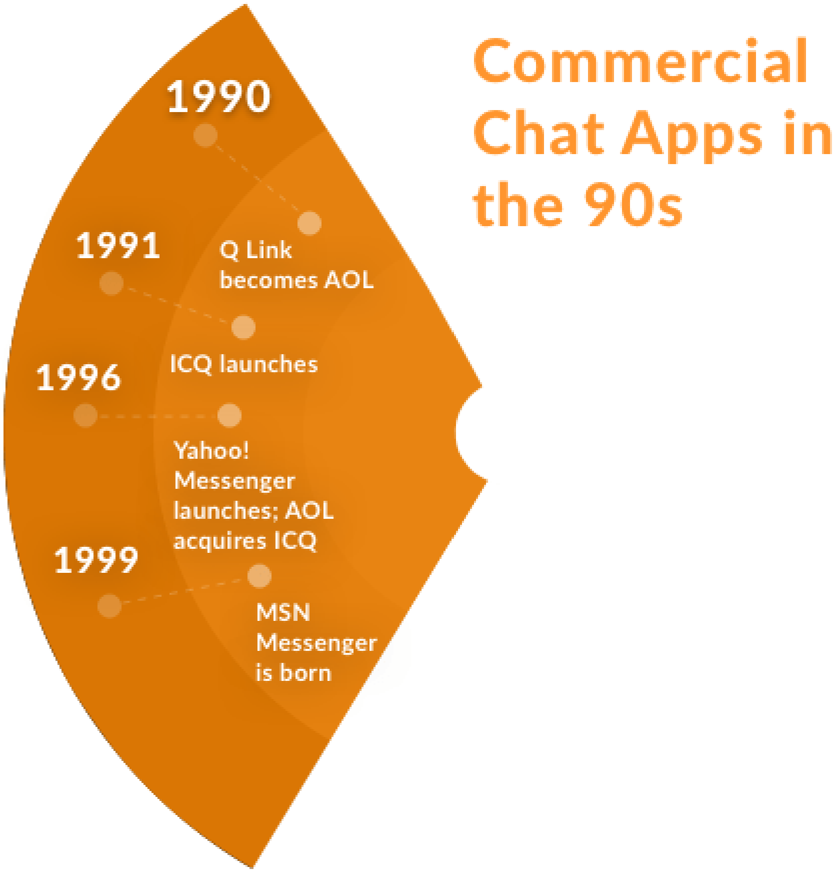

Commercial Chat Apps in the 90s
1990
Q Link becomes AOL
1991
ICQ launches
1996
Yahoo! Messenger launches; AOL acquires ICQ
1999
MSN Messenger is born

Figure 2.1: Commercial chat apps in 90's

As AIM pioneered Chabot, like SmarterChild, it also increased its market share. Its rivalry with MSN Messenger began almost as soon as the competitor launched in 1999. By 2006, AIM controlled 52% of the IM market. Its dominance, however, was short lived. Struggling to monetize and facing increasing competition from new apps like Skype and Google Talk, AIM fell out of favour, eventually eliminating its entire development team in 2012.

2.2.2. COMMERCIAL CHAT APPS IN THE 10s

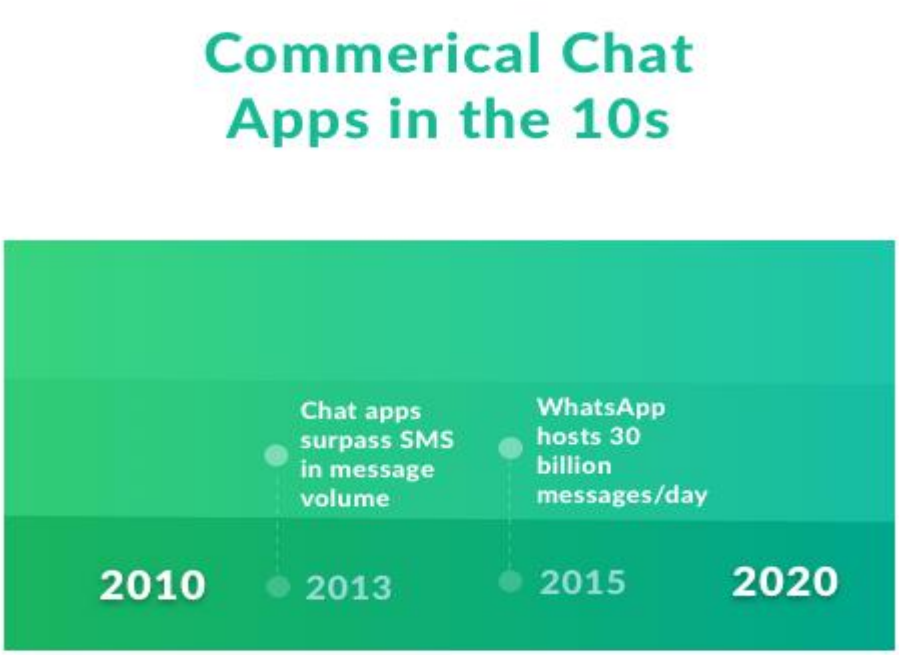


Figure 2.2: Commercial chat apps in 10’s

With the inception of smart phones, chat apps continued to thrive; in 2013, chat apps finally surpassed SMS in message volume. By 2015, WhatsApp alone hosted 30 billion messages per day; SMS logged only 20 billion. And in the summer of 2016, Facebook Messenger hit one billion users.

2.2.3. THE SLOW GROWTH OF ENTERPRISE CHAT APPS

Despite developing around the same time, the history of enterprise chat apps is markedly different than the story of their consumer-facing counterparts. The very first enterprise chat apps do not enjoy the same place in our collective memory as AIM and ICQ.

Early contender Yammer launched in 2008; Microsoft acquired the platform in 2012. Clearspace began in 2006, rebranding several times until its rebirth as Jive six years later. Other enterprise chat programs, usually integrated with other social features like blogs and wikis, blipped in and out of existence.

None of this initial crop of enterprise apps proved a runaway success, and many theories exist as to why later programs have overshadowed them. One thing is certain: these programs predated the rise of smart phones, and mobility certainly fomented the creation of second-generation commercial apps like WhatsApp and Snapchat.

2.2.3.1 ENTERPRISE CHAT: THE NEXT GENERATION

As the first decade of the new millennium closed, an enterprise chat app renaissance slowly began. Though email had been widely used for the previous twenty years, companies soon began looking for a better way to communicate quickly; email-based workflows are slower and do not allow for many business functions that are now critical to work, like screen-sharing or video calls. Some one-to-one chat applications existed, like GChat and Outlook Messenger, but group messaging applications had yet to take off.

In January 2010, three graduates of Rensselaer Polytechnic Institute—Chris Rivers, Garret Heaton, and Pete Curley—launched Hipchat, a web-based chat and instant messaging service. Shortly thereafter, Atlassian acquired it in March 2012. Its premium version addressed several enterprise concerns by adding screens haring, history retention controls, and the ability to run within corporate firewalls.

In August 2013, the enterprise chat space exploded. Slack, which stands for "Searchable Log of All Conversation and Knowledge," grew out of an internal tool used during the development of *Glitch*, a defunct online game. Though not the first (or even the most revolutionary) office chat app, Slack's growth—it now has four million users—has dwarfed most other programs. A mere 1.25 years after launching, it reached a valuation of $1 billion. By April 2015, Slack was worth almost three times that.

Slack–with its channels, DMs, and private groups–is reminiscent of IRC, which might help explain its popularity. A host of similar apps have since incorporated these features. 2015 saw the launch of Cisco Spark, which has since evolved into an entire ecosystem of SDKs, APIs, and a developer site.

Microsoft designed MS Teams to compete directly with Slack; it launched in early 2017 as an integrated component of Office 365.

As businesses prioritize digital transformation, enterprise chat apps continue to enjoy overwhelming popularity. What's to come? Chatbots, machine learning, and increased integration are all popular with both enterprise and commercial users, who discover new ways to chat every day.

2.2.4 REGIONAL AND DEMOGRAPHIC STRONGHOLDS

When devising one's strategy for messaging apps, it's vital to select the right platform mix for the targeted populace, based on TWO core criteria:

1. **Regional Strongholds:** Only a small group of apps like WhatsApp, Facebook Messenger, and Viber can be said to be truly global—and even those platforms struggle in certain countries. Meanwhile, messengers like WeChat, LINE, and KakaoTalk completely dominate specific markets but have negligible traction in others.

2. **Demographics:** It's a common misconception that messaging apps are a uniformly millennial phenomenon. Some apps like Snapchat and LINE skew both young and female, but others like Tango (which boasts 100 million monthly, active users, by estimate) predominantly appeal to those aged 25–54 and strongly over-index with Hispanic and African-American users.

2.2.5 INDUSTRY CHALLENGES

1. **FRAGMENTATION:** The social media landscape is entering a period of hyper-fragmentation that may be a challenge to publishers: Facebook, Twitter, and Instagram continue to loom large, but social media managers can now launch official channels on roughly 10 chat apps with over 50 million monthly, active users each.
2. **ANALYTICS:** For organizations accustomed to robust, real-time data, the lack of good analytics tools for messaging apps remains a major deterrent to adoption. The challenge is twofold: Strong analytics dashboards take time to build, and many messengers are privacy-centric by nature.

2.2.6 INDUSTRY OPPORTUNITIES

1. **HIGHER ENGAGEMENT:** Since many chat apps provide publishers with push notifications or chatbot experiences (programmable robots that converse with users), they can deliver significantly higher engagement rates.
2. **AUDIENCE DEVELOPMENT:** With billions of active users across multiple major chat apps, there is the opportunity in building large audiences fairly quickly on several platforms.
3. **A CHANCE TO CONNECT WITH USERS IN A NEW WAY:** Messaging apps offer a host of features not unavailable on social networks or other platforms. Programmers can creatively leverage these tools to socialize in new ways.

2.3 Related Work

2.3.1 WHAT THE FUTURE HOLDS

Predictions are always prone to inevitable ridicule and failure, but there are some that are worth making.

2.3.1.1 THE GROWING IMPORTANCE OF SECURITY, CIRCUMVENTION, AND DATA RESTRICTIONS

As government snooping, personal privacy, and security become issues for many people globally, those living in countries where these are particular concerns will increasingly look for platforms that enable them to both communicate securely and receive accurate information, unfiltered by government censors.

2.3.1.2 THE EMERGENCE OF REGIONALIZED AND LOCAL MESSAGING-APP ECOSYSTEMS

This is the era to launch CHATTY as it focuses on specific target audience in Africa.

2.3.1.3 MESSAGING WILL BECOME LIKE ELECTRICITY

While messaging is currently a clearly defined function of specific apps, the future is likely to be one wherein the capability is baked-in to nearly all digital technologies and services. The point where a messaging app begins and ends will begin to blur. Already, app classification is getting trickier, especially as social media platforms update their in-app messaging capabilities, moving them closer to chat app experiences.

2.4 Summary

CHATTY is a social-networking tool that leverages on technology advancement thereby allowing its users communicate and share media. It offers a wonderful one stop shop experience for keeping in touch with people you know. It can be use for messaging, placing voice messages, making voice and video calls, share updates and photos, share locations, enhance local socializing in pidgin English, play games and make monetary financial transactions.

CHATTY has the potential to become a widely used socializing app in Africa with specific focus on the West African sub region with a grass root pilot scheme take off in Nigeria due to the continent rich multi ethnic culture and a wide range of her special way of communication called pidgin English in Nigeria. The numerical strength of the Nigerian population which is estimated to be above 200million people presently affords a viable economic space as a rallying point for this application.

With the gradual acceptance of adult literacy gaining roots in Africa and the various government policies to drive education to the grass roots, CHATTY will be a welcome development and very user friendly to the average Nigerian citizen among which includes the local peasant farmers, market women, artisans, the low-income earners, less privilege and the lower social stratification individuals who are numerous in population. The introduction of the Pidgin English feature is a primary drive and force of attraction as it helps break the rank of the academic division between those who claim the Queens English is the best, thereby creating fun and increased cross fertilization of ideas, with a resultant economic boost.

While emphasis must be made to fore runners of similar inventions, which cuts across technology divides, it must be well understood that the basic fundamental of the CHATTY which is embedded in messaging service, evolve as an app where you can comfortably make financial transactions as well.

CHAPTER 3: REQUIREMENTS, ANALYSIS, AND DESIGN

3.1 Overview

Communication is a means for people to exchange messages. It has started since the beginning of human creation. Distant communication began as early as 1800 century with the introduction of televisions, telegraphs and then telephony. Interestingly enough, telephone communication stands out as the fastest growing technology, from fixed lines to mobile wireless, from voice call to data transfer. The emergence of computer network and telecommunication technologies bears the same objective that is to allow people to communicate. All this while, much efforts has been drawn towards consolidating the device into one and therefore indiscriminate the service. Chatting is a method of using technology to bring people and ideas together despite of the geographical barriers. The technology has been available for years, but the acceptance was quite recent. This project is an example of a real time chat app, it is made up of the user application which runs on the user mobile and the server application, which runs on any PC on the network. To start chatting the user should get connected to a server where he can do group and private chatting.

3.2 Requirements Specifications

Table 3.1: Requirement Specification

RQID	NAME	FUNCTIONAL	NON FUNCTIONAL	DESCRIPTION	PRIORITY	ACTOR
0.01	User registration	Yes		Functionality for user to create account	High	User
0.02	Login	Yes		Functionality for user to get access	High	User

0.03	Logout	Yes		Functionality for user to delete session	High	User
0.04	Add friend	Yes		Functionality to be friend with each other	Medium	User
0.05	Friend list	Yes		Functionality to see list of friends	High	User
0.06	Remove friend	Yes		Functionality to remove friend	Medium	User
0.07	Find friend	Yes		To search/find	High	User
0.08	Block friend	Yes		Functions to block friend	Medium	User
0.09	Best friend	Yes		To make close friend	Low	User
0.10	Profile	Yes		User profile	Medium	User
0.11	Send message	Yes		To send message	High	User
0.12	Message status	Yes		To show message time	Medium	User
0.13	Delete message	Yes		To delete message	Medium	User
0.14	History	Yes		See past message	High	User
0.15	Feedback	Yes		User can send feedback	Low	User & Admin
0.16	Privacy		Yes	User privacy	High	Admin
0.17	Robustness		Yes	Dealing with errors	High	Admin
0.18	Performance		Yes	Application performance must be better	High	Admin
0.19	Usability		Yes	Easy for newbies	High	Admin
0.20	Reliability		Yes	Trusted by users	High	Admin
0.21	Supportability		Yes	Being supportive	High	Admin
0.22	portability		Yes	Application runs in different systems	High	admin

3.3 Use case Diagram

3.3.1 Use case Table

Table 3.2: Use Case Table of Chat Application

Level 0	Level 1	Level 2	actor
Chat application	Authentication system	Register Login logout	user
Chat application	Contact form	Friend list Find friend Add friend Remove friend	user
Chat application	Chat form	Send message	user
Chat application	maintenance	User profile Data base	admin

3.3.2 Authentication Service

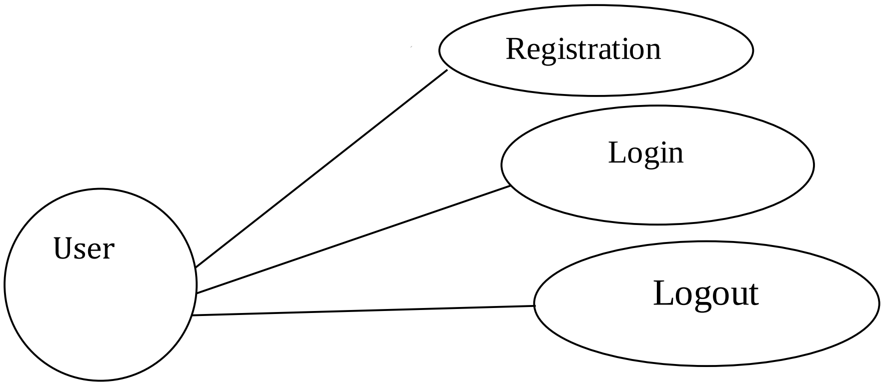


Figure 3.1: Use Case Diagram of Authentication Service

3.3.3 Contact Form

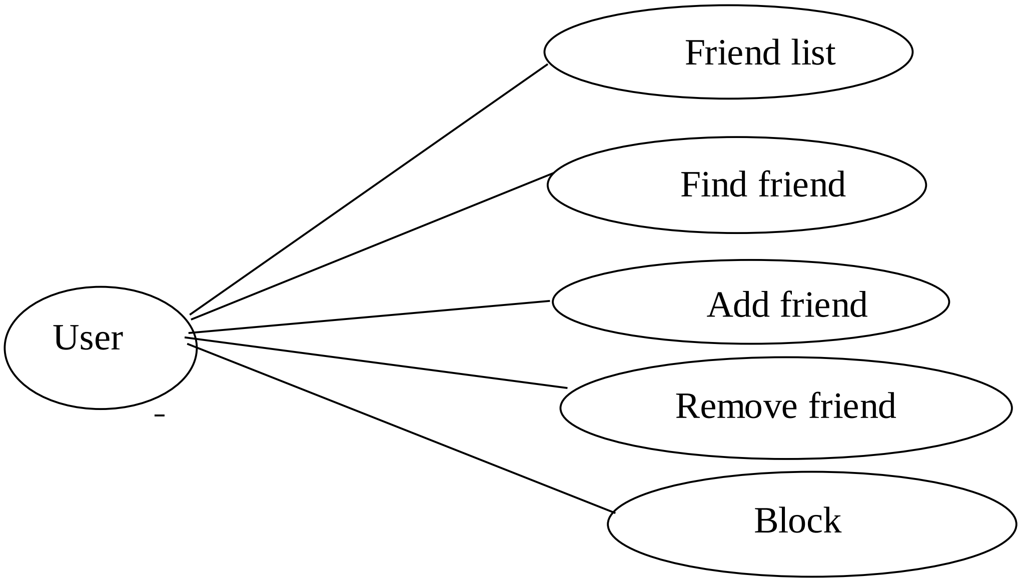


Figure 3.2: Use Case Diagrams of Contact Forms

3.3.4 Chat Form

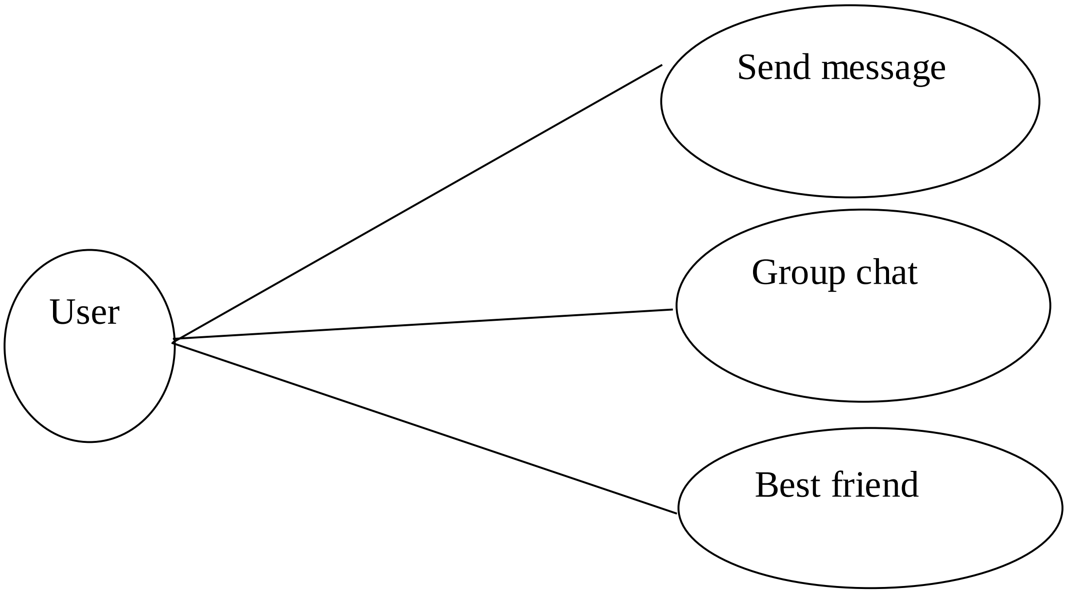


Figure 3.3: Use Case Diagram of Chat Form

3.8 System Design

3.8.1 Application Architecture

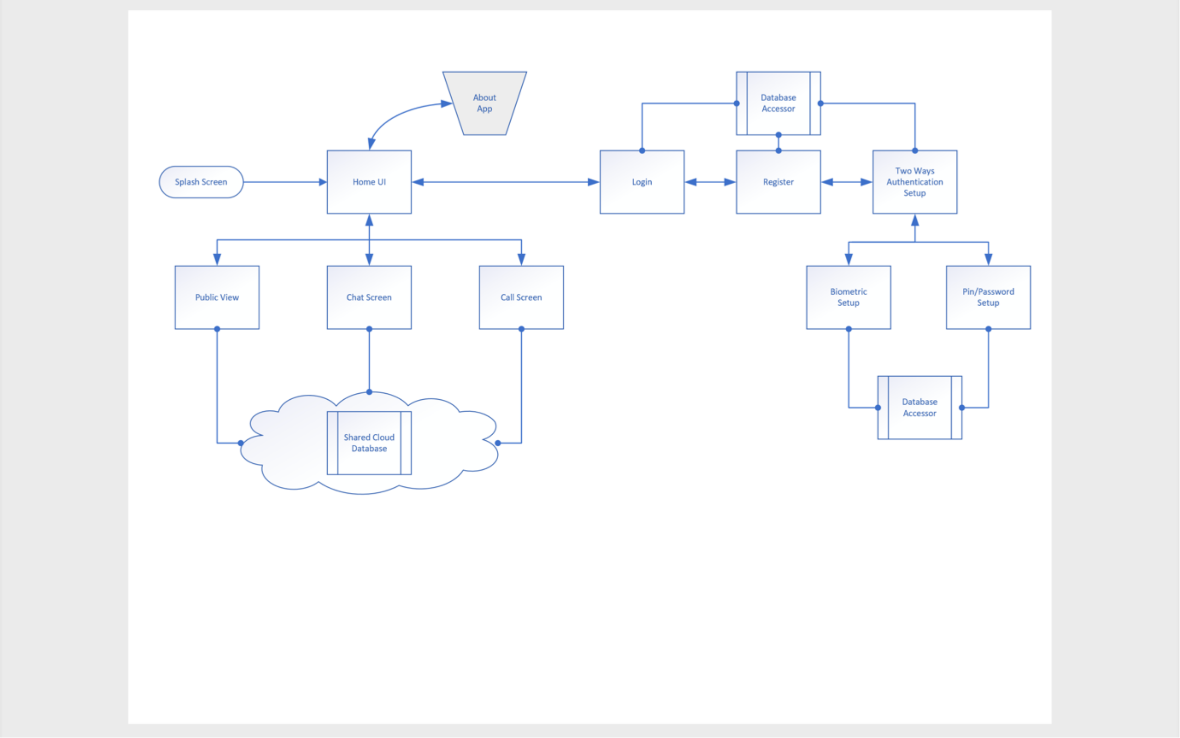


Figure 3.4: A sample of the Project Application Architecture

3.8.2 Project Entity Relationship Diagram (ERD)

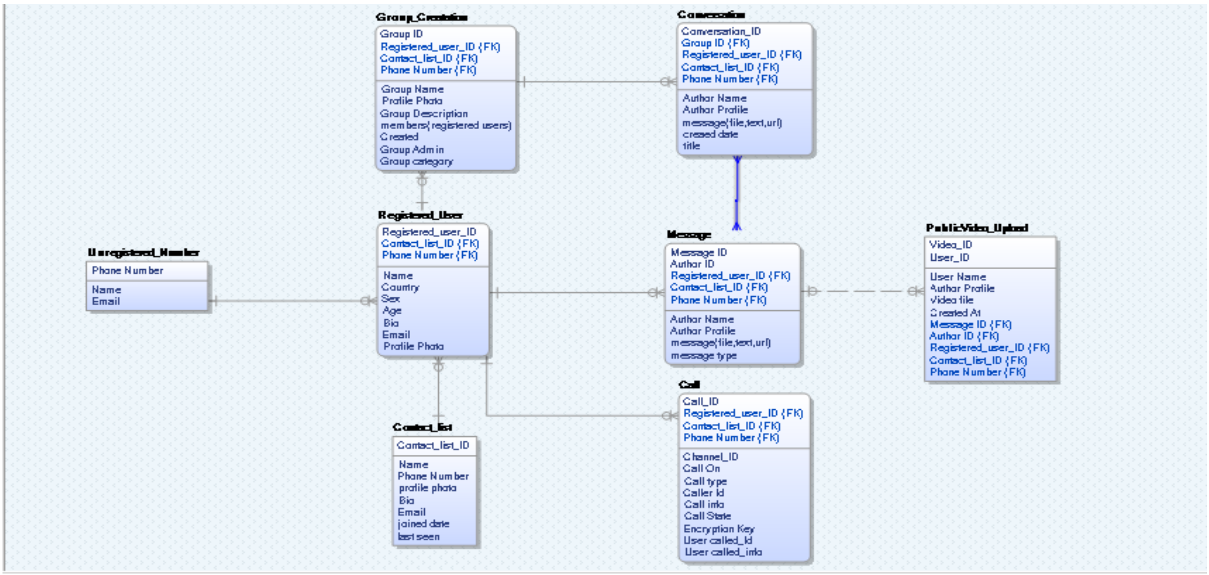


Figure 3.5: A sample of the Project Entity Relationship Diagram (ERD)

3.8.3 Monitor

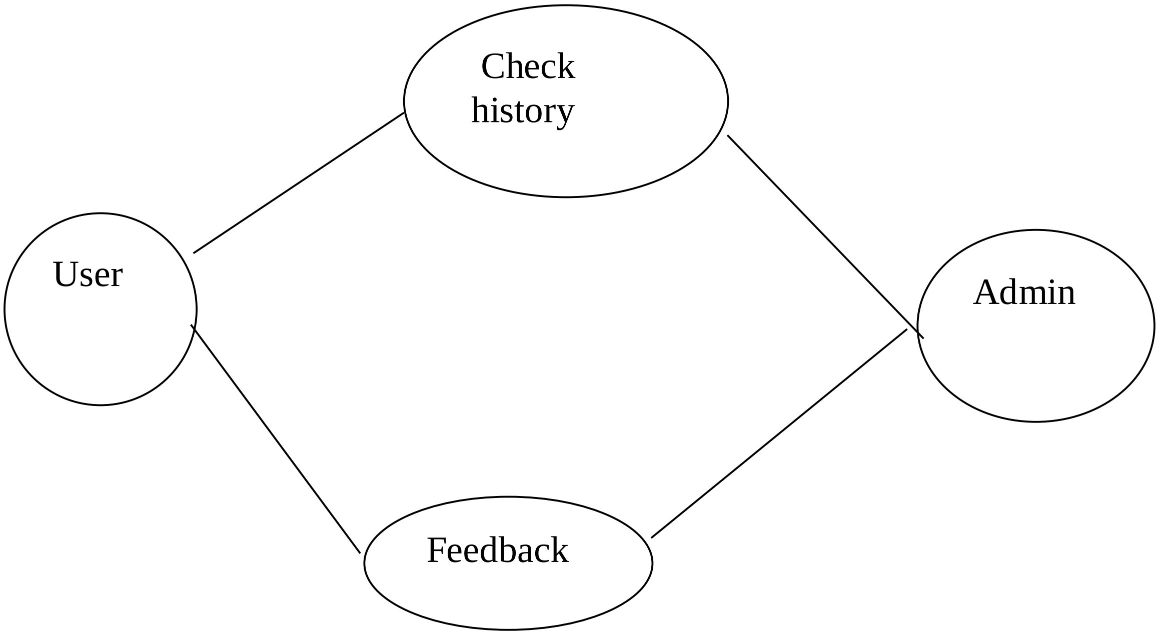


Figure 3.6: Use Case Diagram of Monitor

3.8.4 User Interface Design

Figure 3.7: splash screen

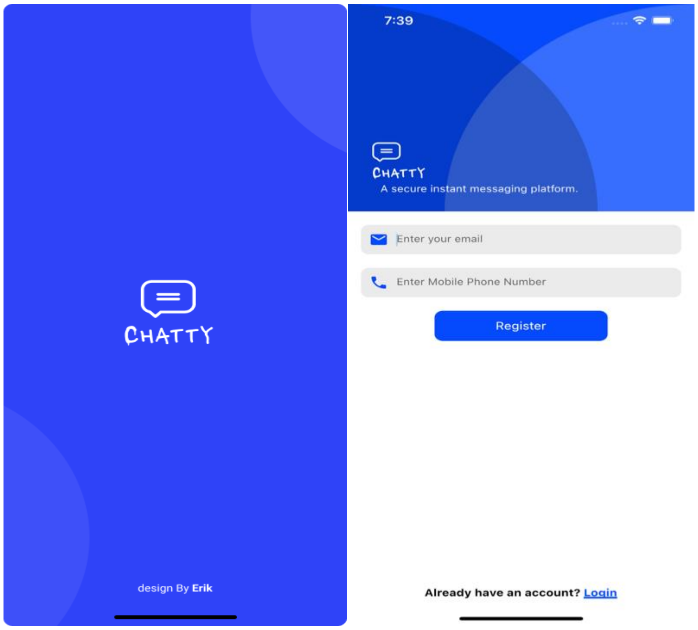
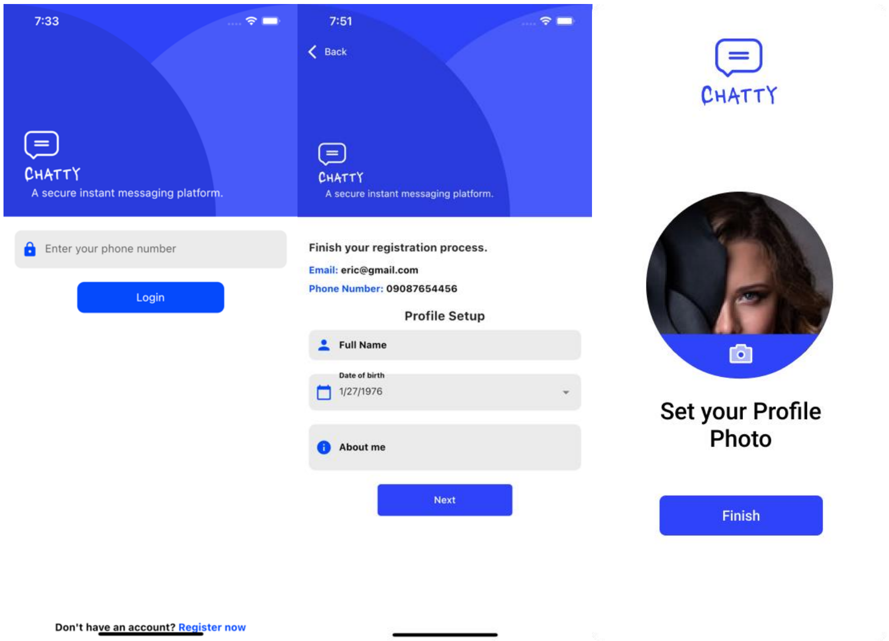

CHATTY
design By Erik
7:39
CHATTY
A secure instant messaging platform.
Enter your email
Enter Mobile Phone Number
Register
Already have an account? Login

Figure 3.8: Register Screen

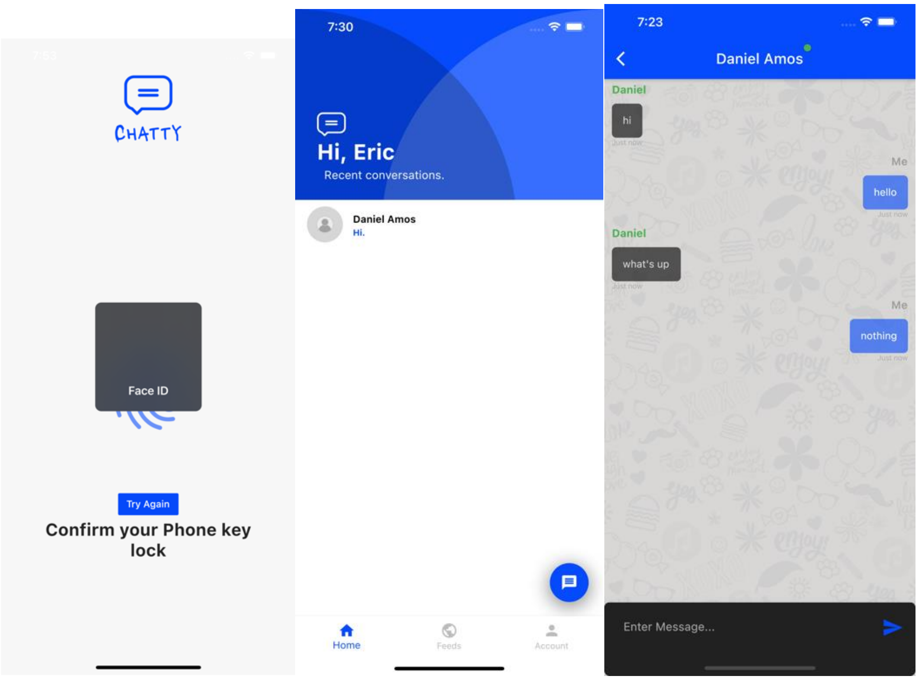


Finish your registration process.

Email: eric@gmail.com

Phone Number: 09087654456

Date of birth 1/27/1976



Daniel Amos
Hi.

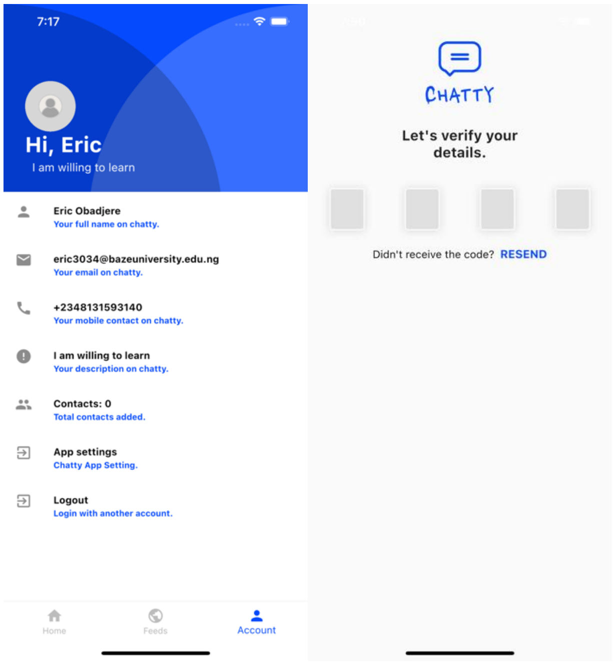
Daniel
hi

Me
hello

Daniel
what's up

Me
nothing

Figure 3.9: Steps to take to reach the chat screen



Eric Obadjere
Your full name on chatty.

eric3034@bazeuniversity.edu.ng
Your email on chatty.

+2348131593140
Your mobile contact on chatty.

I am willing to learn
Your description on chatty.

Contacts: 0
Total contacts added.

App settings
Chatty App Setting.

Logout
Login with another account.

Figure 3.9.1: client profile

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1 Overview

This document provides the requirement for the design and implementation of a chat application. Both functional and non-functional requirements are being documented. This project will create a chat application with a server and allow users to be able to chat with themselves. Instant messaging solution will be proffered so that users will be able to communicate seamlessly and ensuring that even a novice can use this chatting application, thereby ensuring it is not too complex, so that it can cut across a wide range of audience while specifically considering African population with the interaction using Pidgin English.

4.2 Project Scope and features

1. Broadcasting chat server application is going to be a text communication software, it will be able to communicate between two computers using point to point communication
2. There is limitation on live chat as it does not support audio communication.
3. The easy usability breaks the complexity syndrome.

4.3 Methodology

4.3.1 Project Workability

The user interacts with the tool using GUI. The GUI operates in two forms, which are contacts forms and chat forms. The contacts forms contain the list of all friends and the chat form will be used to chat with friends.

4.3.2 Project Deliverables

1. An Android app
2. Documentation
3. Readme file

4.3.3 Hardware Interface

Android phone

256 MB minimum RAM required

Internet or LAN connections

Processor with speed of 500MHz

4.3.4 Constraints & Limitations

The system must be connected to the internet. User can use or install this app on android devices. This app does not have audio and video calling system.

4.3.5 Project risk management

Identifying the necessary threat of the project either internal or external will help a great deal in ensuring the success of the project. To guide again risk significantly much brainstorming was carried out to factor in all facets of the project details.

4.4 Tools and Technology

Quality Planning

Software QFD

Product Innovation

Brainstorming, Mind-Map, TRIZ/ARIZ, Innovative algorithms

Software Analysis

Brainstorming

Mind-Map

Design patterns

UML tools and technique

Database modelling tools

Mongodb Compass, Mongoose, Mongodb Driver

Software Development Methodology

Agile, scrum

Programming Language

Typescript, JavaScript, Dart, Json, yaml, swift, java

4.5 Testing

Installation test, Functional test, Load test, Performance profiling, Data integrity test, & automated test.

HIGHER LEVEL ITEMS TO BE TESTED

1. Chat application and supporting infrastructure
2. Application running on different client devices

HIGHER LEVEL ITEMS NOT TO BE TESTED

1 SRS of chat application

2 User Manual of chat application

3 Already existing chat application

4 Manual processes related to the application

5 Any legacy system

LOWER LEVEL ITEMS TO BE TESTED

1 User Profile

2 Chatting

3 Add Friend

4 Remove Friend

5 Find Friend

6 Registrar

7 Login

8 verification

9 biometric Login

10 Logout

LOWER LEVEL ITEMS NOT TO BE TESTED
1 User

4.6 Use Guide

1 Processes

2 Registrations for new Members

3 Login Features

4 Adding Friend

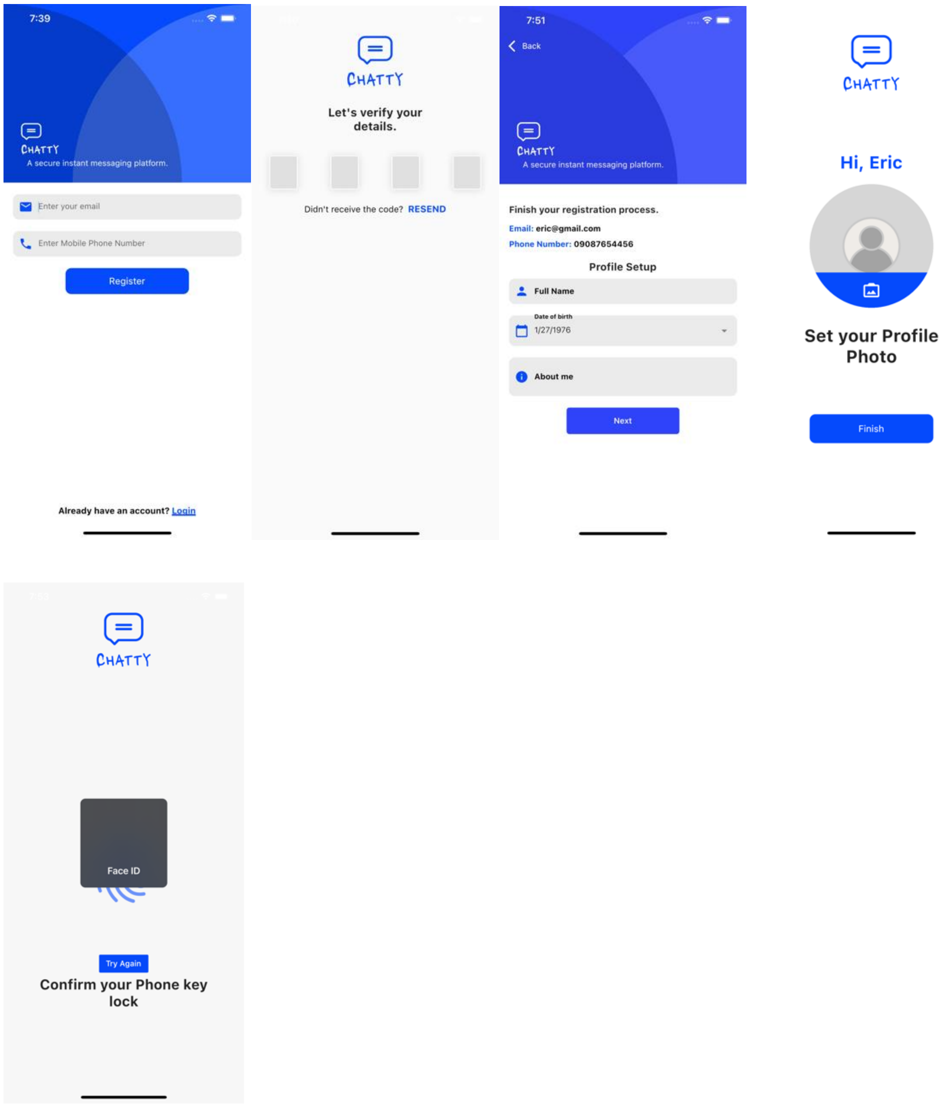
5 Chat Forms

6 Settings

4.6.1 Processes

Installing the chatty on an android mobile device is as simple as installing the apk file.

4.6.2 Registration for new Members


7:39
CHATTY
A secure instant messaging platform.
Enter your email
Enter Mobile Phone Number
Register
Already have an account? Login
CHATTY
Let's verify your details.
Didn't receive the code? RESEND
7:51
Back
CHATTY
A secure instant messaging platform.

Finish your registration process.

Email: eric@gmail.com

Phone Number: 09087654456

Profile Setup

Full Name

Date of birth 1/27/1976

About me

Next
CHATTY
Hi, Eric
Set your Profile Photo
Finish
CHATTY
Face ID
Try Again
Confirm your Phone key lock

Figure 4.1: Registration for new member

To register a new user, Chatty app require the using to provide their email and phone number. It checks if the phone number or email already has a user in the database, if it doesn't the user is then taken to a screen where they fill in their details which include full name, date of birth and a short description of themselves. The user is then asked to confirm their email and phone number through the email message or SMS sent to them which contains a verification code link to their account. Finally, the user goes through it local biometric lock and adds their profile picture if need be, then done. The user will now have a full active account.

4.6.1 Login Features

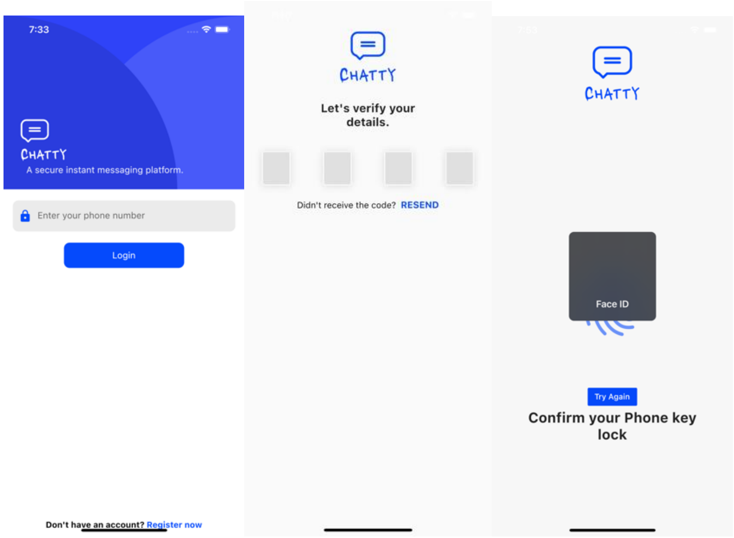
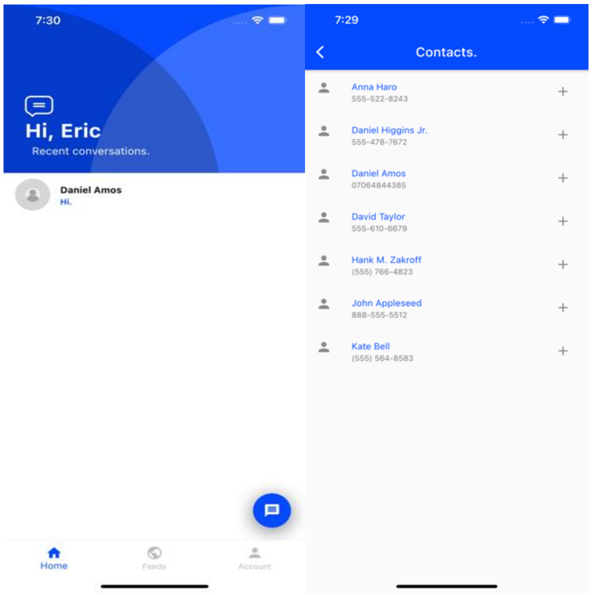

7:33
CHATTY
A secure instant messaging platform.
Enter your phone number
Login
Don't have an account? Register now
CHATTY
Let's verify your details.
Didn't receive the code? RESEND
CHATTY
Face ID
Try Again
Confirm your Phone key lock

Figure 4.2: Login features

Chatty uses a two-factor login system which includes biometric authentication and email or phone number verification. First the phone number is verified to belong to a user and then a verification code is sent to both the users' email and phone number if the phone number was found to belong to a user. Next the user is asking to continue with the login process with the device local biometric authentication, which could be either face id or fingerprint authentication.

4.6.4 Adding Friends



7:30

Hi, Eric

Recent conversations.

Daniel Amos

Hi.

7:29

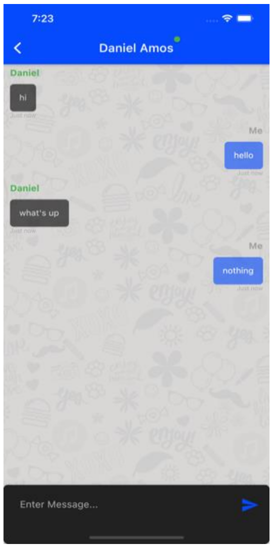
Contacts.

Anna Haro	555-522-8243
Daniel Higgins Jr.	555-478-7672
Daniel Amos	07064844385
David Taylor	555-610-6679
Hank M. Zakroff	(555) 766-4823
John Appleseed	888-555-5512
Kate Bell	(555) 564-8583

Figure 4.3: Adding Friends

Adding friends through the chatty app can be done by clicking floating button at the bottom right corner of the app which after giving the app permission brings a list of all your contact list saved on your mobile phone.

4.6.5 Chat Form



Daniel
hi
Just now

Me
hello
Just now

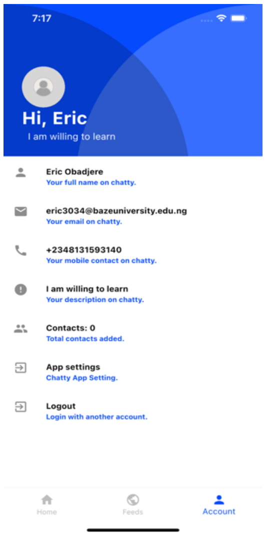
Daniel
what's up
Just now

Me
nothing
Just now

Figure 4.4: Chat Form

The Chat Form hold and displays the current conversation going on between two users of the app, this conversation is not saved locally or on a live server, so a new session is created for every new instance in time. The chat form has two main items which are the message areas which on-going conversations can be seen and the chat form which carry an input field to send new messages area.

4.6.6 Account Settings



Hi, Eric
I am willing to learn

Eric Obadjere
Your full name on chatty.

eric3034@bazeuniversity.edu.ng
Your email on chatty.

+2348131593140
Your mobile contact on chatty.

I am willing to learn
Your description on chatty.

Contacts: 0
Total contacts added.

App settings
Chatty App Setting.

Logout
Login with another account.

Figure 4.5: Account Setting

Using the Account settings is very simple and clear, as there are not too many functions on it. The Account settings allows you to logout, see your contact list count, view your details and open the app settings and app info.

4.7 Summary

There is always room for improvement in any application. This project deals extensively with text communication with special feature addition of Pidgin English which makes it user friendly to those at the bottom of the societal pyramid as well, thereby capturing a sizeable numerical group within the West Africa region. With the app's friendly user interface, it brings technology closer to the general populace and the grass-rooters.

CHAPTER 5: DISCUSSION, CONCLUSION, AND RECOMMENDATIONS

5.1 Overview

The project successfully delivered on all requirement specification specified by the user. Care was ensured during the design to make sure data integrity is maintained and to avoid all forms of redundancies associated with data.

The user is assured a very friendly interface, behind which there are wide ranging technical details that went in. The user guide is a mere formality because, the project was specially created bearing in mind interaction and designs that would make users feel as though they have used a system such as this.

This project has also been built in such a manner that future changes or modifications that are required can easily be implemented without affecting the functionality of the system. This project is used on android environment, and can be used on any version so it can be used by individuals with different levels of android devices. The technical document that is provided in the report of this project will help developers understand the internal workings of the system.

5.2 Objective Assessment

After building the system, I achieved these objectives:

- I was able to develop a multi lingual chat application
- I was able to develop an interactive system that incorporated Pidgin English into chatting

I was able to evaluate the existing literature of instant messaging

.

5.3 Limitations and Challenges

The following challenges were observed after development of this system

- Only registered users can use the system
- Internet must be available to use the application
- There must be minimum of two users per time for interactive chatting

5.4 Future Enhancements

- Video calls will be added
- Voice recording can be added
- Enhancing different text style and font size
- Introduction of animations
- Instant document attachment

5.5 Recommendations

Research done in regard to this project, coupled with the fact that we now live in a world that everything is now becoming digitalized, the following recommendations were derived:

- The development of chat application shouldn’t and wouldn’t stop here, constant improvement and research needs to be conducted to make sure it is in line with best technology practices.
- The development of local content chatting application should be encouraged as this will help improve the economy bas of Nigeria.
- Every class of the societal stratification should be encouraging to use this localized chatting application as it improves our education standard and exposure to the outside world.

5.6 Summary

Messaging apps now have more global users than traditional social networks—which mean they will play an increasingly important role in the distribution of information in the future. While chat platforms initially rose to prominence by offering a low-cost, web-based alternative to SMS, over time they evolved into multimedia hubs that support photos, videos, games, payments, and more. To harness the growing population of Africa in general and Nigeria in particular was the introduction of the Pidgen English chatting leverage, which is unique in its own right.

REFERENCES

1. Aaron, P., (2020) "Webrtc vs websockets." http://stackoverflow.com/a/18825175, sep 2013. Accessed: 2017-04-19.

2. Ackley, B., (2020) "Webrtc samples." https://webrtc.github.io/samples/, sep 2013. Accessed: 2017-04-19.

3. Abby, T., (2020) "User stories: An agile introduction." http://www.agilemodeling.com/ artifacts/userStory.htm. Accessed: 2016-10-20.

4. Bairam, W., (2020) "Making the switch from making the switch from node.js to golang." http://blog.digg.com/post/141552444676/ making-the-switch-from-nodejs-to-golang, mar 2016. Accessed: 2016-10- 19.

5. Callum, S., (2020) "Sinon - best practices for spies, stubs and mocks." https://semaphoreci.com/community/tutorials/best-practices-for-spies-stubsand-mocks-in-sinon-js, 2016. Accessed: 2016-10-03.

6. Caleb, P., (2020) "Why is nosql faster than sql?." http://softwareengineering. stackexchange.com/questions/175542/why-is-nosql-faster-than-sql, nov 2012. Accessed: 2016-10-21.

7. Dakari, U., (2020) "Why nosql." http://softwareengineering.stackexchange.com/ questions/175542/why-is-nosql-faster-than-sql. Accessed: 2016- 10-21.

8. Dashin, O., & Shanda, R., (2020) "Exploring the different types of nosql databases." https://www. 3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases, may 2015. Accessed: 2017-04-29.

9. Daen, I., (2020) "Composition vs inheritance - react." https://facebook.github.io/react/ docs/composition-vs-inheritance.html. Accessed: 2016-12-18.

10. D. T. Andrew Hunt, The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Professional, first ed., october 1999.

11. Faryn, F., (2020) "Mongodb manual 3.4." https://docs.mongodb.com/manual/reference/ limits/. Accessed: 2017-08-30.

12. Gandale, A., (2020) “6 rules of thumb for mongodb schema design.” http://blog.mongodb.org/post/87200945828/ 6-rules-of-thumb-for-mongodb-schema-design-part-1, may 2014. Accessed: 2016-10-18.

13. Hakam, T., (2020) “Scaling secret: Real-time chat.” https://medium.com/always-be-coding/ scaling-secret-real-time-chat-d8589f8f0c9b#.m5jigxq6x, may 2015. Accessed: 2016-10-18.

14. Hale, B., (2020) “Analysis of json use cases.” https://blogs.oracle.com/xmlorb/entry/ analysis_of_json_use_cases, apr 2013. Accessed: 2016-11-08.

15. Icharus, H.,“Crud cycle (create, read, update and delete cycle).” http:// searchdatamanagement.techtarget.com/definition/CRUD-cycle. Accessed: 2016-11-08.

16. Idys, E., & Sam, D.,(2020) “About native xmlhttp.” https://msdn.microsoft.com/en-us/library/ ms537505(v=vs.85).aspx. Accessed: 2016-12-18.

17. Kaija, S.,(2020) “Please. don’t patch like an idiot..” http://williamdurand.fr/2014/02/14/ please-do-not-patch-like-an-idiot/, aug 2016. Accessed: 2016-12-18.

18. K.D, & Ernest, A.,(2020)“Rfc 5789 - patch method for http.” https://tools.ietf.org/html/rfc5789, mar 2010. Accessed: 2016-12-18.

19. Lang, Y.,(2020)“Perfomance tips | google cloud platform.” https://cloud.google.com/ storage/docs/json_api/v1/how-tos/performance#patch. Accessed: 2016- 12-18.

20. Najib, M.,(2020)“container vs component?.” https://github.com/reactjs/redux/issues/ 756#issuecomment-141683834, sep 2015. Accessed: 2016-11-13.

21. Oisin, H., (2020) “Html5 websocket: A quantum leap in scalability for the web.” http://www. websocket.org/quantum.html. Accessed: 2016-11-13.

22. Sam, T., (2020) “Building a scalable node.js express app.” https://medium.com/@zurfyx/ building-a-scalable-node-js-express-app-1be1a7134cfd, feb 2017. Accessed: 2017-04-13.

23. Sadhbh, I., “Strategy design pattern.” https://sourcemaking.com/design_patterns/ strategy. Accessed: 2016-11-14.

24. T’Marius, A., (2020)“Dynamic testing.” https://www.tutorialspoint.com/software_testing_ dictionary/dynamic_testing.htm, apr 2017. Accessed: 2017-04-03.

APPENDICES

Appendix A - Project Document

The project documentation for an application for online commodity and delivery system

DETAILED PROJECT DOCUMENTATION

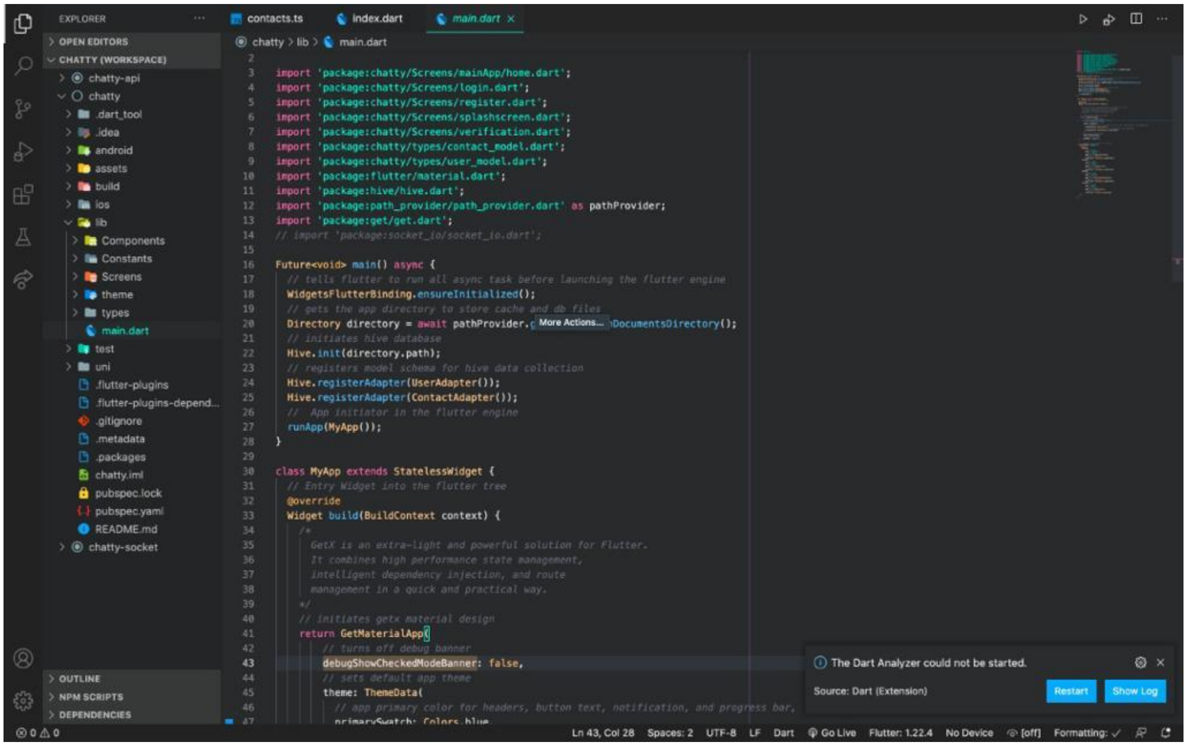
Candidate Name: Eric Obadjere Nyerhovwo

Student Id No: BU/18A/IT/3034

Design and implementation of a Real Time Chat

Course of Study: B.Sc. Software Engineering

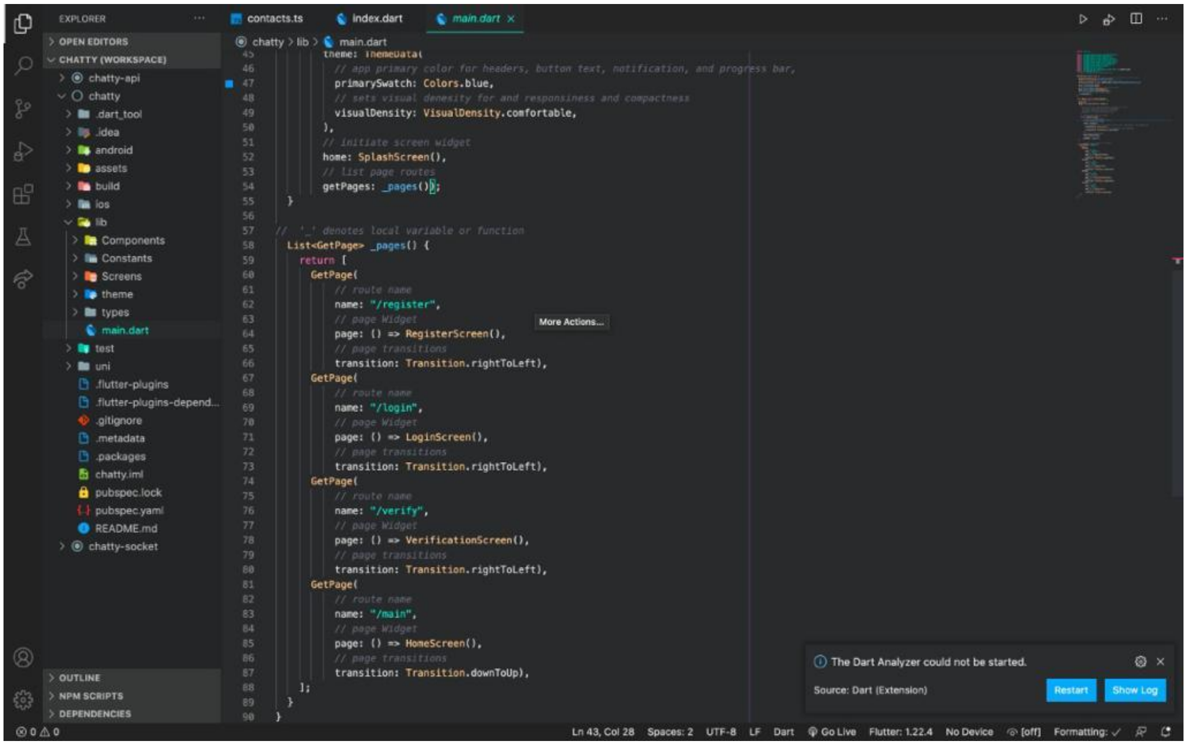
Appendix B – Source Code



```dart

import 'package:chatty/Screens/mainApp/home.dart';
import 'package:chatty/Screens/login.dart';
import 'package:chatty/Screens/register.dart';
import 'package:chatty/Screens/splashscreen.dart';
import 'package:chatty/Screens/verification.dart';
import 'package:chatty/types/contact_model.dart';
import 'package:chatty/types/user_model.dart';
import 'package:flutter/material.dart';
import 'package:hive/hive.dart';
import 'package:path_provider/path_provider.dart' as pathProvider;
import 'package:get/get.dart';
// import 'package:socket_io/socket_io.dart';

Future<void> main() async {
  // tells flutter to run all async task before launching the flutter engine
  WidgetsFlutterBinding.ensureInitialized();
  // gets the app directory to store cache and db files
  Directory directory = await pathProvider.g[illegible]DocumentsDirectory();
  // initiates hive database
  Hive.init(directory.path);
  // registers model schema for hive data collection
  Hive.registerAdapter(UserAdapter());
  Hive.registerAdapter(ContactAdapter());
  //  App initiator in the flutter engine
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // Entry Widget into the flutter tree
  @override
  Widget build(BuildContext context) {
    /*
      GetX is an extra-light and powerful solution for Flutter.
      It combines high performance state management,
      intelligent dependency injection, and route
      management in a quick and practical way.
    */
    // initiates getx material design
    return GetMaterialApp(
      // turns off debug banner
      debugShowCheckedModeBanner: false,
      // sets default app theme
      theme: ThemeData(
        // app primary color for headers, button text, notification, and progress bar,
        primarySwatch: Colors.blue
```

The Dart Analyzer could not be started.

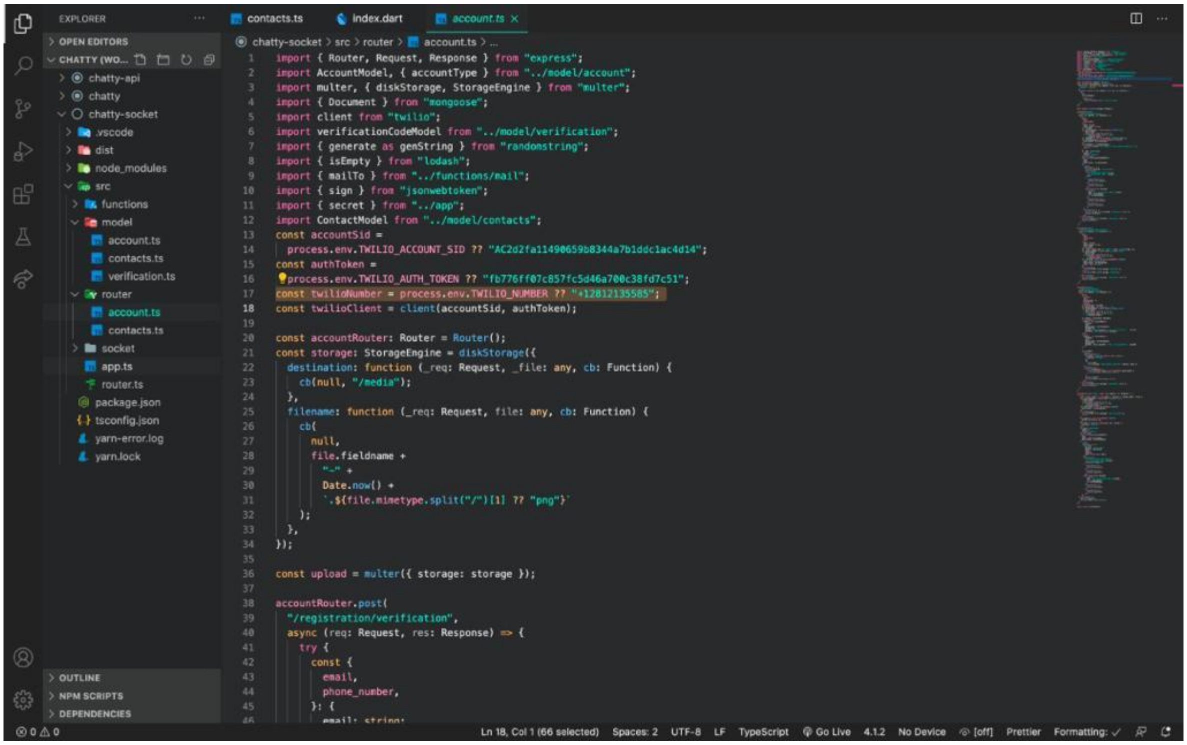
Source: Dart (Extension)



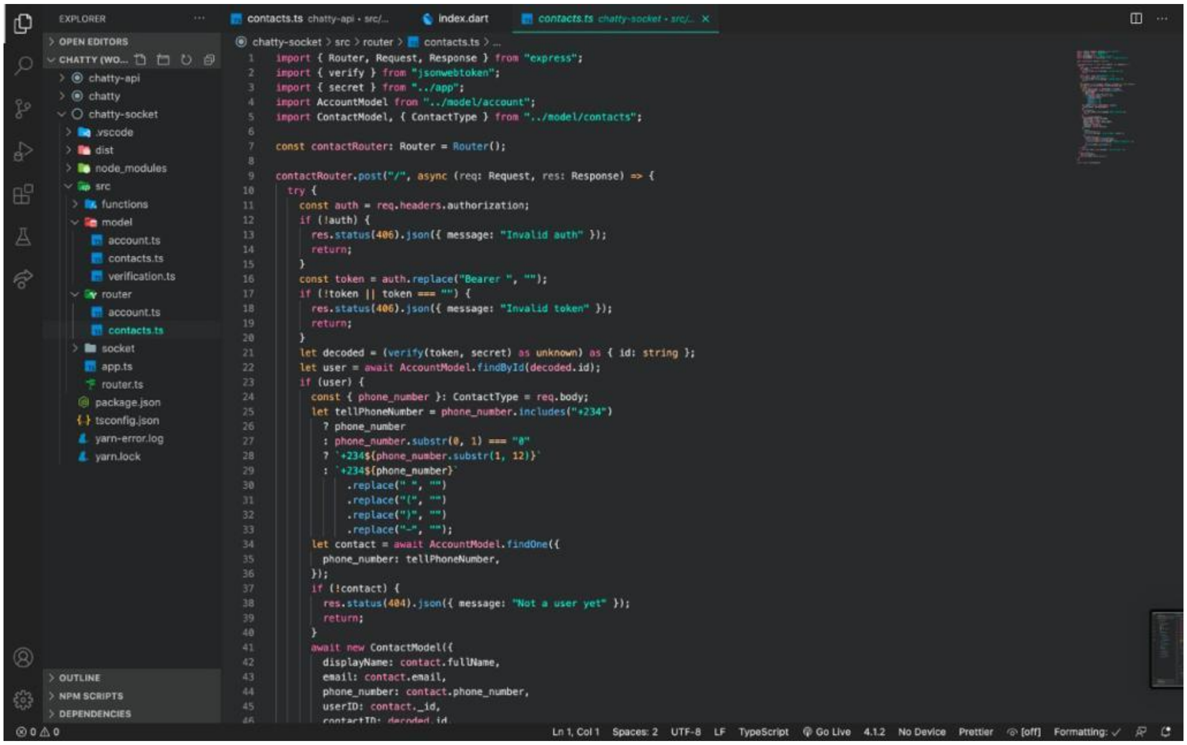
```dart
      theme: ThemeData(
        // app primary color for headers, button text, notification, and progress bar,
        primarySwatch: Colors.blue,
        // sets visual denesity for and responsiness and compactness
        visualDensity: VisualDensity.comfortable,
      ),
      // initiate screen widget
      home: SplashScreen(),
      // list page routes
      getPages: _pages());
  }

//  '_' denotes local variable or function
  List<GetPage> _pages() {
    return [
      GetPage(
        // route name
        name: "/register",
        // page Widget
        page: () => RegisterScreen(),
        // page transitions
        transition: Transition.rightToLeft),
      GetPage(
        // route name
        name: "/login",
        // page Widget
        page: () => LoginScreen(),
        // page transitions
        transition: Transition.rightToLeft),
      GetPage(
        // route name
        name: "/verify",
        // page Widget
        page: () => VerificationScreen(),
        // page transitions
        transition: Transition.rightToLeft),
      GetPage(
        // route name
        name: "/main",
        // page Widget
        page: () => HomeScreen(),
        // page transitions
        transition: Transition.downToUp),
    ];
  }
}
```

The Dart Analyzer could not be started.

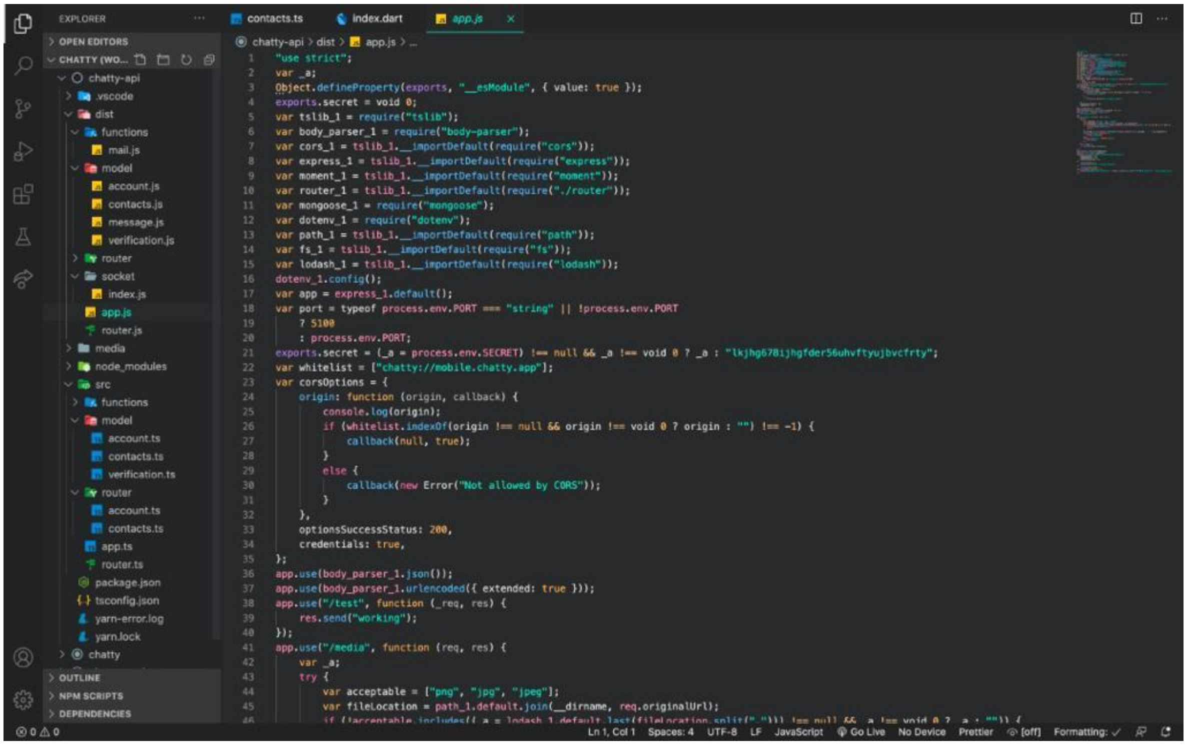
Source: Dart (Extension)



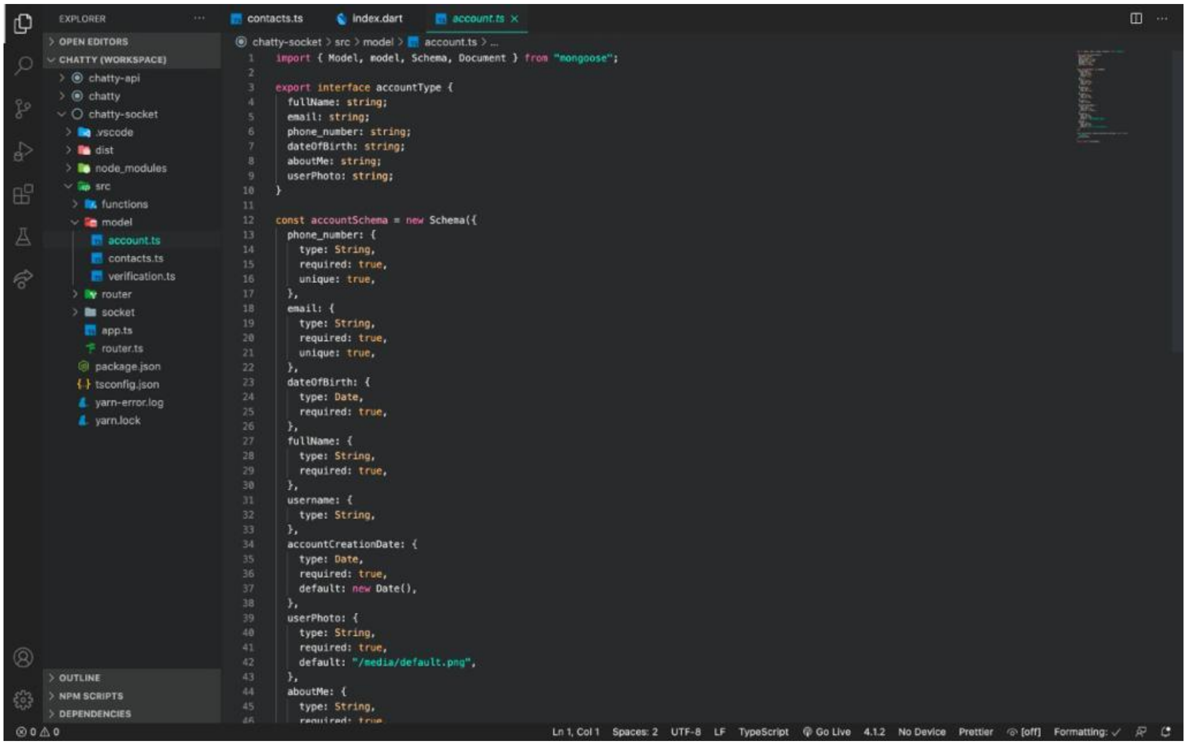
```typescript
import { Router, Request, Response } from "express";
import AccountModel, { accountType } from "../model/account";
import multer, { diskStorage, StorageEngine } from "multer";
import { Document } from "mongoose";
import client from "twilio";
import verificationCodeModel from "../model/verification";
import { generate as genString } from "randomstring";
import { isEmpty } from "lodash";
import { mailTo } from "../functions/mail";
import { sign } from "jsonwebtoken";
import { secret } from "../app";
import ContactModel from "../model/contacts";
const accountSid =
  process.env.TWILIO_ACCOUNT_SID ?? "AC2d2fa11490659b8344a7b1ddc1ac4d14";
const authToken =
  process.env.TWILIO_AUTH_TOKEN ?? "fb776ff07c857fc5d46a700c38fd7c51";
const twilioNumber = process.env.TWILIO_NUMBER ?? "+12812135585";
const twilioClient = client(accountSid, authToken);

const accountRouter: Router = Router();
const storage: StorageEngine = diskStorage({
  destination: function (_req: Request, _file: any, cb: Function) {
    cb(null, "/media");
  },
  filename: function (_req: Request, file: any, cb: Function) {
    cb(
      null,
      file.fieldname +
        "-" +
        Date.now() +
        `.${file.mimetype.split("/")[1] ?? "png"}`
    );
  },
});

const upload = multer({ storage: storage });

accountRouter.post(
  "/registration/verification",
  async (req: Request, res: Response) => {
    try {
      const {
        email,
        phone_number,
      }: {
        email: string;
```



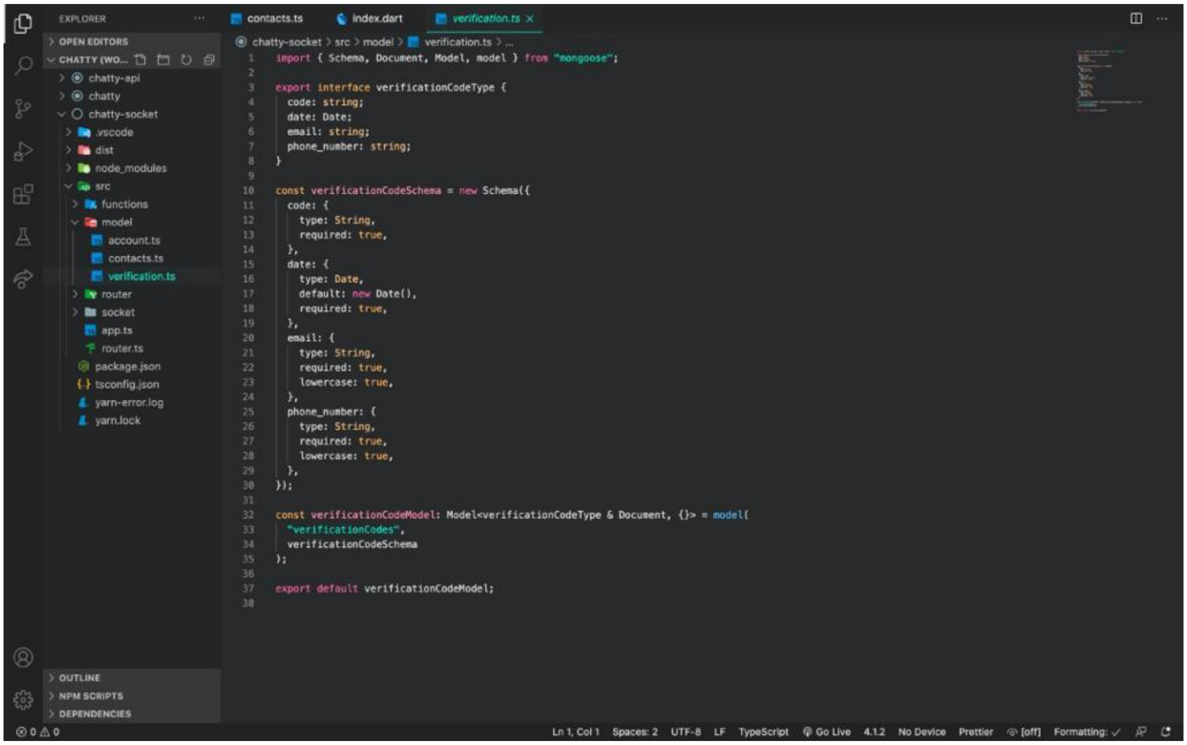
```typescript
import { Router, Request, Response } from "express";
import { verify } from "jsonwebtoken";
import { secret } from "../app";
import AccountModel from "../model/account";
import ContactModel, { ContactType } from "../model/contacts";

const contactRouter: Router = Router();

contactRouter.post("/", async (req: Request, res: Response) => {
  try {
    const auth = req.headers.authorization;
    if (!auth) {
      res.status(406).json({ message: "Invalid auth" });
      return;
    }
    const token = auth.replace("Bearer ", "");
    if (!token || token === "") {
      res.status(406).json({ message: "Invalid token" });
      return;
    }
    let decoded = (verify(token, secret) as unknown) as { id: string };
    let user = await AccountModel.findById(decoded.id);
    if (user) {
      const { phone_number }: ContactType = req.body;
      let tellPhoneNumber = phone_number.includes("+234")
        ? phone_number
        : phone_number.substr(0, 1) === "0"
        ? `+234${phone_number.substr(1, 12)}`
        : `+234${phone_number}`
            .replace(" ", "")
            .replace("(", "")
            .replace(")", "")
            .replace("-", "");
      let contact = await AccountModel.findOne({
        phone_number: tellPhoneNumber,
      });
      if (!contact) {
        res.status(404).json({ message: "Not a user yet" });
        return;
      }
      await new ContactModel({
        displayName: contact.fullName,
        email: contact.email,
        phone_number: contact.phone_number,
        userID: contact._id,
        contactID: decoded.id
```



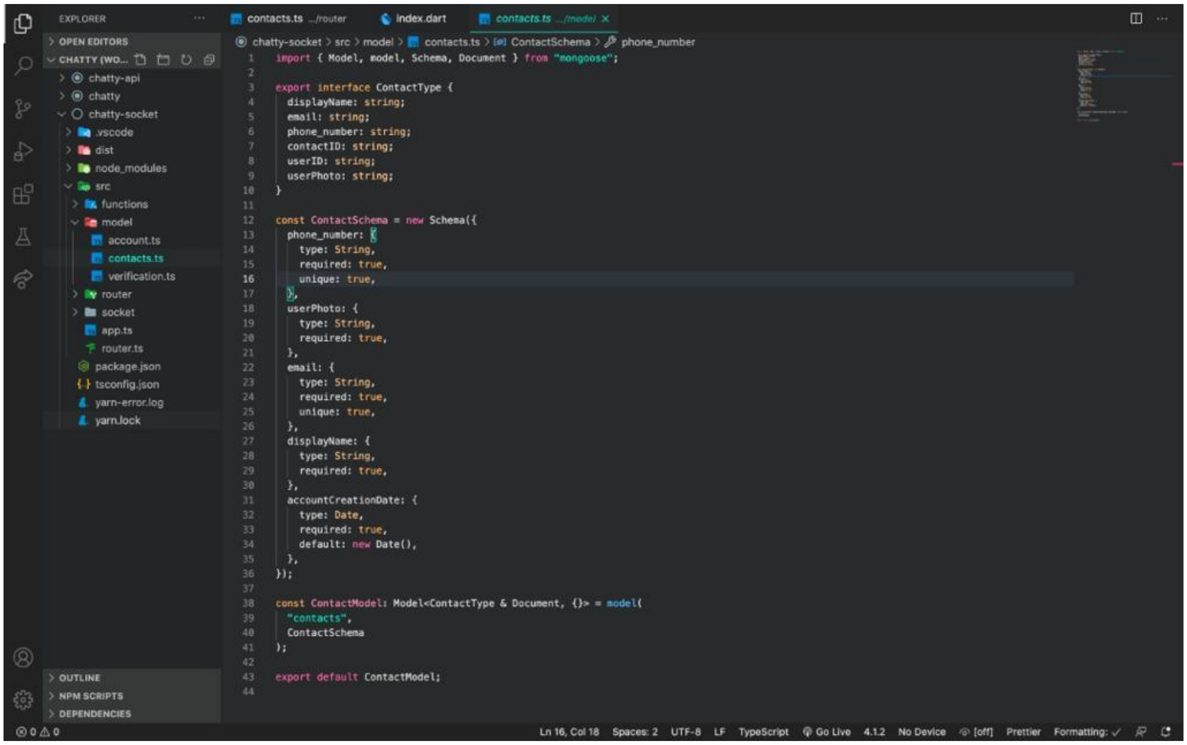
```
"use strict";
var _a;
Object.defineProperty(exports, "__esModule", { value: true });
exports.secret = void 0;
var tslib_1 = require("tslib");
var body_parser_1 = require("body-parser");
var cors_1 = tslib_1.__importDefault(require("cors"));
var express_1 = tslib_1.__importDefault(require("express"));
var moment_1 = tslib_1.__importDefault(require("moment"));
var router_1 = tslib_1.__importDefault(require("./router"));
var mongoose_1 = require("mongoose");
var dotenv_1 = require("dotenv");
var path_1 = tslib_1.__importDefault(require("path"));
var fs_1 = tslib_1.__importDefault(require("fs"));
var lodash_1 = tslib_1.__importDefault(require("lodash"));
dotenv_1.config();
var app = express_1.default();
var port = typeof process.env.PORT === "string" || !process.env.PORT
    ? 5100
    : process.env.PORT;
exports.secret = (_a = process.env.SECRET) !== null && _a !== void 0 ? _a : "lkjhg678ijhgfder56uhvftyujbvcfrty";
var whitelist = ["chatty://mobile.chatty.app"];
var corsOptions = {
    origin: function (origin, callback) {
        console.log(origin);
        if (whitelist.indexOf(origin !== null && origin !== void 0 ? origin : "") !== -1) {
            callback(null, true);
        }
        else {
            callback(new Error("Not allowed by CORS"));
        }
    },
    optionsSuccessStatus: 200,
    credentials: true,
};
app.use(body_parser_1.json());
app.use(body_parser_1.urlencoded({ extended: true }));
app.use("/test", function (_req, res) {
    res.send("working");
});
app.use("/media", function (req, res) {
    var _a;
    try {
        var acceptable = ["png", "jpg", "jpeg"];
        var fileLocation = path_1.default.join(__dirname, req.originalUrl);
        if (!acceptable.includes((_a = lodash_1.default.last(fileLocation.split(".")))) !== null && _a !== void 0 ? _a : "")) {
```



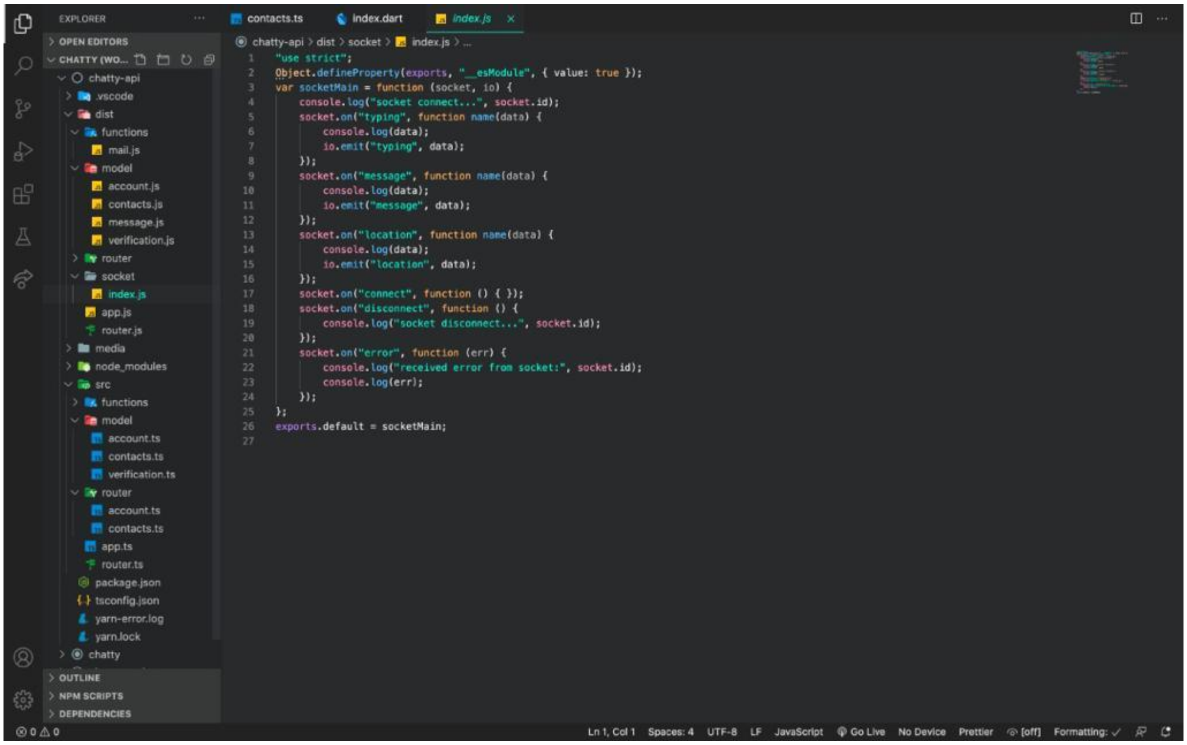
```
import { Model, model, Schema, Document } from "mongoose";

export interface accountType {
  fullName: string;
  email: string;
  phone_number: string;
  dateOfBirth: string;
  aboutMe: string;
  userPhoto: string;
}

const accountSchema = new Schema({
  phone_number: {
    type: String,
    required: true,
    unique: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  dateOfBirth: {
    type: Date,
    required: true,
  },
  fullName: {
    type: String,
    required: true,
  },
  username: {
    type: String,
  },
  accountCreationDate: {
    type: Date,
    required: true,
    default: new Date(),
  },
  userPhoto: {
    type: String,
    required: true,
    default: "/media/default.png",
  },
  aboutMe: {
    type: String,
    required: true
```



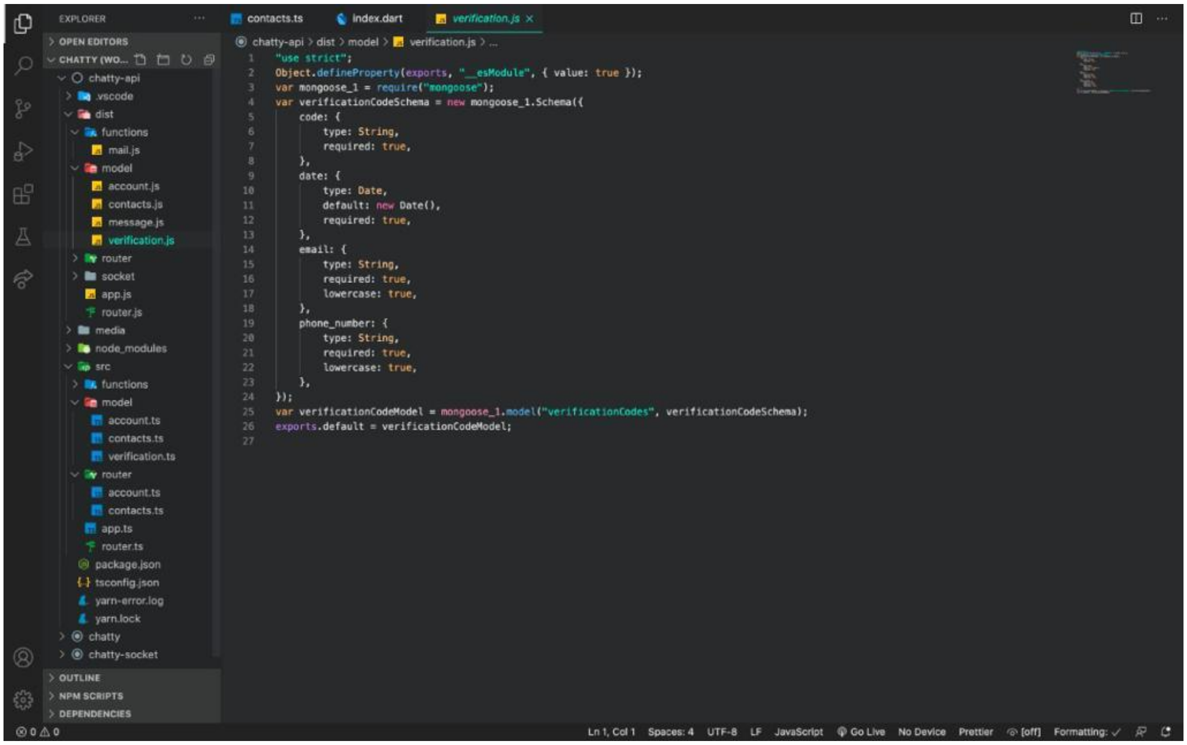
```typescript
import { Schema, Document, Model, model } from "mongoose";

export interface verificationCodeType {
  code: string;
  date: Date;
  email: string;
  phone_number: string;
}

const verificationCodeSchema = new Schema({
  code: {
    type: String,
    required: true,
  },
  date: {
    type: Date,
    default: new Date(),
    required: true,
  },
  email: {
    type: String,
    required: true,
    lowercase: true,
  },
  phone_number: {
    type: String,
    required: true,
    lowercase: true,
  },
});

const verificationCodeModel: Model<verificationCodeType & Document, {}> = model(
  "verificationCodes",
  verificationCodeSchema
);

export default verificationCodeModel;
```



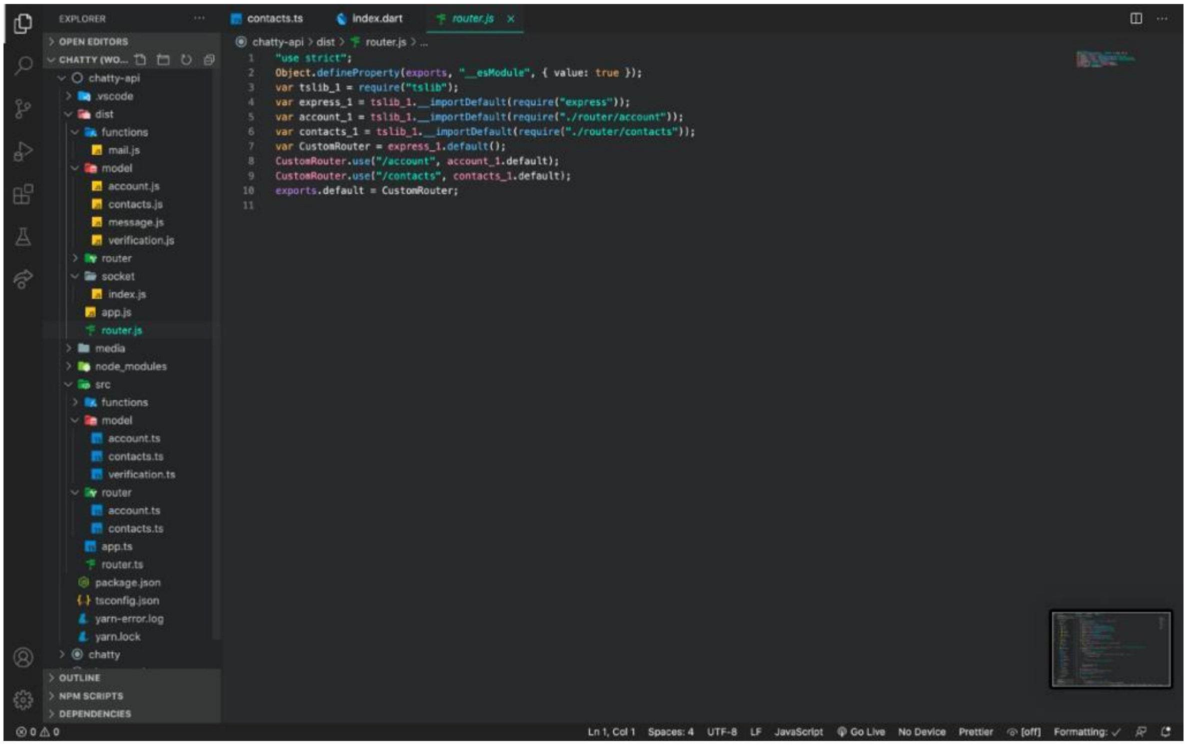
```typescript
import { Model, model, Schema, Document } from "mongoose";

export interface ContactType {
  displayName: string;
  email: string;
  phone_number: string;
  contactID: string;
  userID: string;
  userPhoto: string;
}

const ContactSchema = new Schema({
  phone_number: {
    type: String,
    required: true,
    unique: true,
  },
  userPhoto: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  displayName: {
    type: String,
    required: true,
  },
  accountCreationDate: {
    type: Date,
    required: true,
    default: new Date(),
  },
});

const ContactModel: Model<ContactType & Document, {}> = model(
  "contacts",
  ContactSchema
);

export default ContactModel;
```



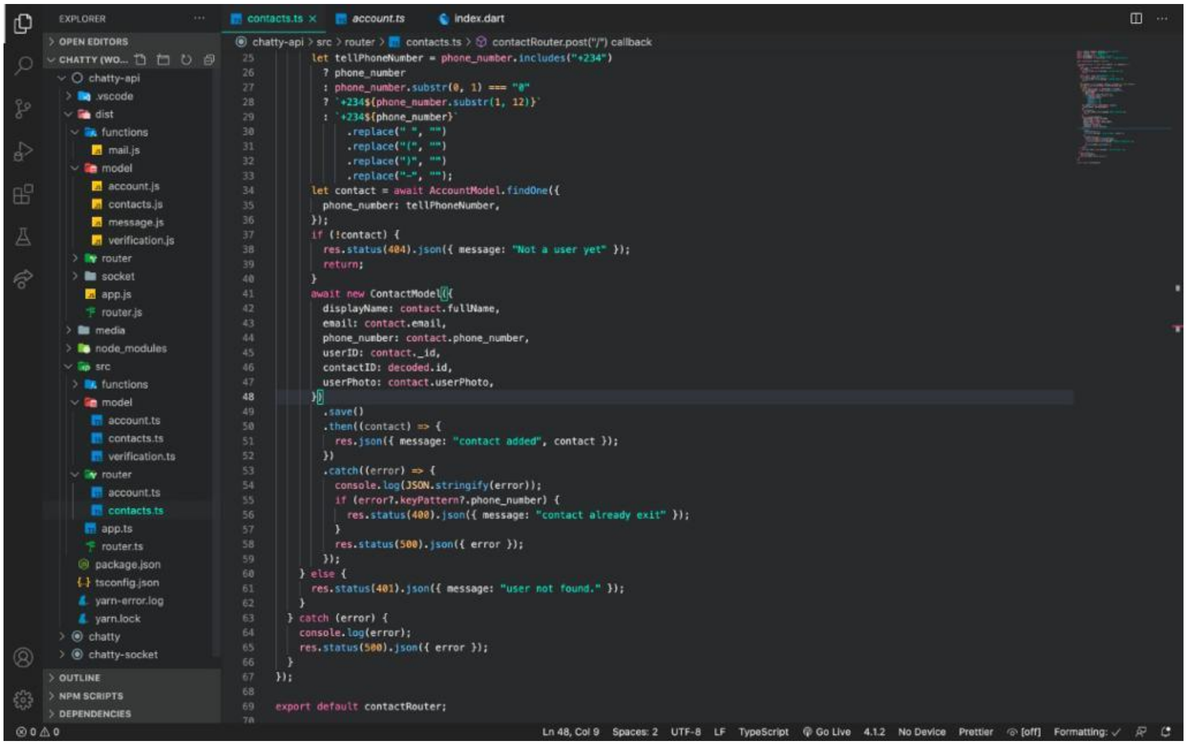
```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
var socketMain = function (socket, io) {
    console.log("socket connect...", socket.id);
    socket.on("typing", function name(data) {
        console.log(data);
        io.emit("typing", data);
    });
    socket.on("message", function name(data) {
        console.log(data);
        io.emit("message", data);
    });
    socket.on("location", function name(data) {
        console.log(data);
        io.emit("location", data);
    });
    socket.on("connect", function () { });
    socket.on("disconnect", function () {
        console.log("socket disconnect...", socket.id);
    });
    socket.on("error", function (err) {
        console.log("received error from socket:", socket.id);
        console.log(err);
    });
};
exports.default = socketMain;
```



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
var mongoose_1 = require("mongoose");
var verificationCodeSchema = new mongoose_1.Schema({
    code: {
        type: String,
        required: true,
    },
    date: {
        type: Date,
        default: new Date(),
        required: true,
    },
    email: {
        type: String,
        required: true,
        lowercase: true,
    },
    phone_number: {
        type: String,
        required: true,
        lowercase: true,
    },
});
var verificationCodeModel = mongoose_1.model("verificationCodes", verificationCodeSchema);
exports.default = verificationCodeModel;
```



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
var tslib_1 = require("tslib");
var express_1 = tslib_1.__importDefault(require("express"));
var account_1 = tslib_1.__importDefault(require("./router/account"));
var contacts_1 = tslib_1.__importDefault(require("./router/contacts"));
var CustomRouter = express_1.default();
CustomRouter.use("/account", account_1.default);
CustomRouter.use("/contacts", contacts_1.default);
exports.default = CustomRouter;
```



```
      let tellPhoneNumber = phone_number.includes("+234")
        ? phone_number
        : phone_number.substr(0, 1) === "0"
        ? `+234${phone_number.substr(1, 12)}`
        : `+234${phone_number}`
          .replace(" ", "")
          .replace("(", "")
          .replace(")", "")
          .replace("-", "");
      let contact = await AccountModel.findOne({
        phone_number: tellPhoneNumber,
      });
      if (!contact) {
        res.status(404).json({ message: "Not a user yet" });
        return;
      }
      await new ContactModel({
        displayName: contact.fullName,
        email: contact.email,
        phone_number: contact.phone_number,
        userID: contact._id,
        contactID: decoded.id,
        userPhoto: contact.userPhoto,
      })
        .save()
        .then((contact) => {
          res.json({ message: "contact added", contact });
        })
        .catch((error) => {
          console.log(JSON.stringify(error));
          if (error?.keyPattern?.phone_number) {
            res.status(400).json({ message: "contact already exit" });
          }
          res.status(500).json({ error });
        });
    } else {
      res.status(401).json({ message: "user not found." });
    }
  } catch (error) {
    console.log(error);
    res.status(500).json({ error });
  }
});

export default contactRouter;
```

Appendix C – Test Cases

HIGHER LEVEL ITEMS TO BE TESTED

1. Chat application and supporting infrastructure
2. Application running on different client devices

HIGHER LEVEL ITEMS NOT TO BE TESTED

1 SRS of chat application

2 User Manual of chat application

3 Already existing chat application

4 Manual processes related to the application

5 Any legacy system

LOWER LEVEL ITEMS TO BE TESTED

1 User Profile

2 Chatting

3 Add Friend

4 Remove Friend

5 Find Friend

6 Registrar

7 Login

8 verification

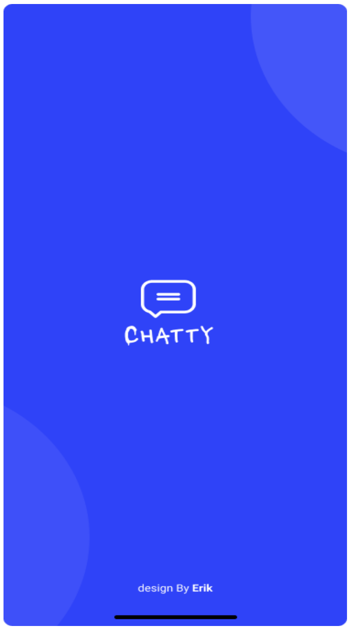
9 biometric Login

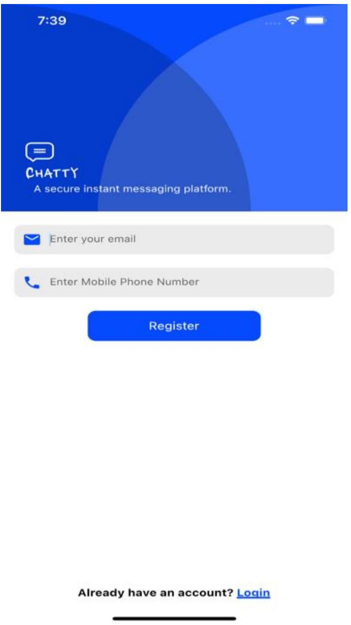
10 Logout

LOWER LEVEL ITEMS NOT TO BE TESTED

1 User

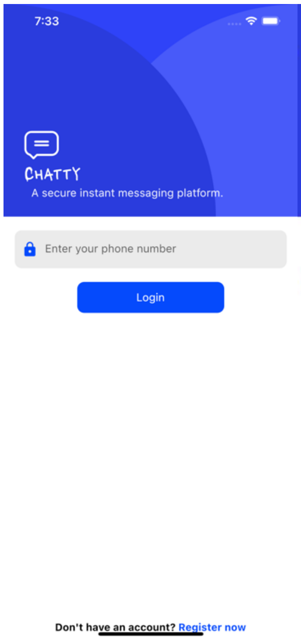
Appendix D – Pictorial User Guide/Manual





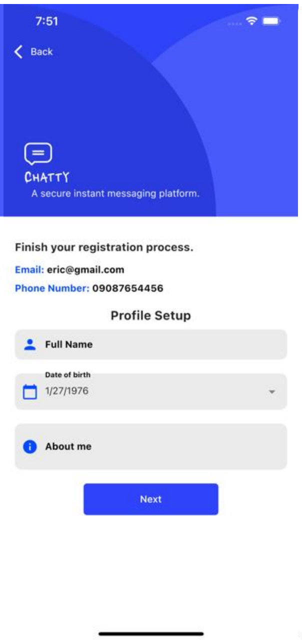
A secure instant messaging platform.

Already have an account? Login



A secure instant messaging platform.

Don't have an account? Register now



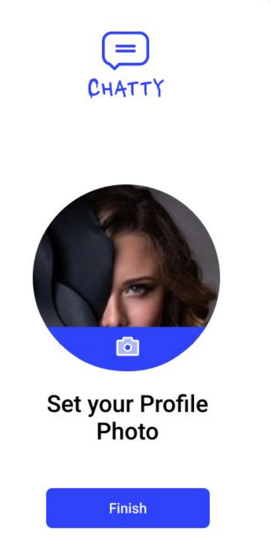
A secure instant messaging platform.

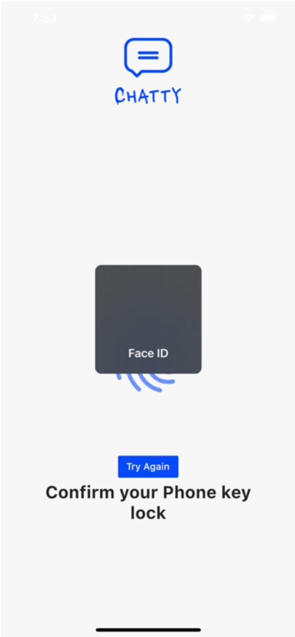
Finish your registration process.

Email: eric@gmail.com

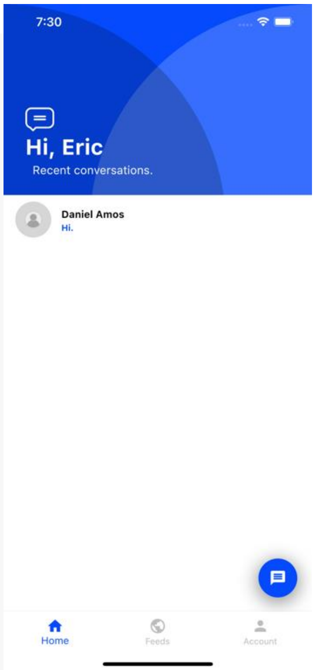
Phone Number: 09087654456

Date of birth 1/27/1976





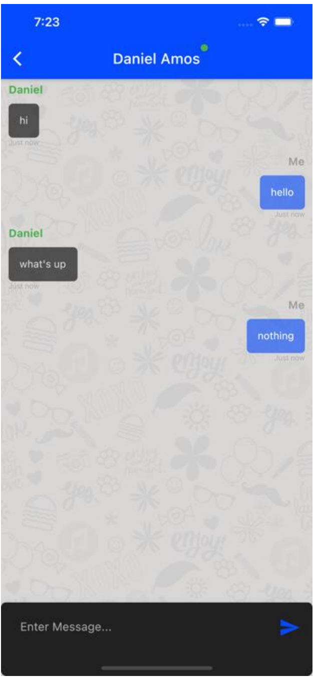
Confirm your Phone key lock



Hi, Eric

Recent conversations.

Daniel Amos
Hi.



Daniel
hi
Just now

Me
hello
Just now

Daniel
what's up
Just now

Me
nothing
Just now