The British College Kathmandu

Coursework Submission Coversheet

(Individual coursework only)

Faculty of Arts, Environment and Technology

77261181

LBU Student Id:

Award name: BSc (Hons) Computing

Module Code: COMP607

Module name: Production Project

Module run:

Coursework title: Secured Messaging Application using ASP.NET Core and SignalR.

Due Date:

Module Leader: (In LBU)

Module Supervisor: (In TBC) Mr. Resham Bahadur Pun

**TURNITIN** Checked: Yes No

Submission date & time:

Total Word Count:

Total Number of Pages:

**Academic Honesty Statement**: I confirm that this coursework is completely my own work, not submitted elsewhere at Leeds Beckett University or any other place. I've properly cited all information sources used in my assignment, and I understand that not doing so is considered academic dishonesty.

Signed: Date:

Teacher’s Signature: Date:

This grade is temporary and needs to be checked and confirmed by the appropriate exam organization.

Teacher’s Feedback



Secured Messaging Application

using ASP.NET Core and SignalR.

Module: Production Project

Date:

BSc. (Hons) in Computing

[Level 6: 2nd Semester]

Submitted To: Mr. Resham Bahadur Pun

Submitted By: Umang Shrestha

**Contents**

[**ABSTRACT** i](#_Toc135674700)

[**ACKNOWLEDGEMENT** ii](#_Toc135674701)

[**Introduction** 1](#_Toc135674702)

[**Project Brief** 1](#_Toc135674703)

[**Problem** 1](#_Toc135674704)

[**Solution** 1](#_Toc135674705)

[**Aims** 2](#_Toc135674706)

[**Objectives** 2](#_Toc135674707)

[**Explanation of the artefact** 2](#_Toc135674708)

[**Review of Literature** 3](#_Toc135674709)

[**Introduction to a Messaging Application** 3](#_Toc135674710)

[**Types of Messaging Applications** 4](#_Toc135674711)

[**Journal Article Review: The effect of privacy concerns, risk, control, and trust on individuals’ decisions to share personal information: A game theory-based approach** 5](#_Toc135674712)

[**Journal Article Review: Security and privacy of smartphone messaging applications** 6](#_Toc135674713)

[**Pre-Print Article Review: HOW DATA CAN BE USED AGAINST PEOPLE: A CLASSIFICATION OF PERSONAL DATA MISUSES** 6](#_Toc135674714)

[**Review of Technology** 8](#_Toc135674715)

[**C#** 8](#_Toc135674716)

[**Visual Studio** 8](#_Toc135674717)

[**SignalR** 8](#_Toc135674718)

[**ASP .NET Core MVC** 9](#_Toc135674719)

[**Entity Framework Core** 9](#_Toc135674720)

[**Tailwind CSS** 10](#_Toc135674721)

[**Microsoft SQL Server** 10](#_Toc135674722)

[**ML.NET** 10](#_Toc135674723)

[**GitHub** 11](#_Toc135674724)

[**Development Methodology** 11](#_Toc135674725)

[**Requirement Phase** 11](#_Toc135674726)

[**Design Phase** 12](#_Toc135674727)

[**Development Phase** 12](#_Toc135674728)

[**Testing Phase** 12](#_Toc135674729)

[**Review and Deployment Phase** 13](#_Toc135674730)

[**Gantt chart** 13](#_Toc135674731)

[**Product Design** 14](#_Toc135674732)

[**MoSCoW Prioritization** 14](#_Toc135674733)

[**Workflow Flowchart** 16](#_Toc135674734)

[**Entity Relationship Diagram** 19](#_Toc135674735)

[Data Dictionary 19](#_Toc135674736)

[**User Interface** 27](#_Toc135674737)

[**Implementation and Testing** 30](#_Toc135674738)

[**Implementation** 31](#_Toc135674739)

[**Testing** 31](#_Toc135674740)

[**Product Evaluation** 33](#_Toc135674741)

[**Scope** 33](#_Toc135674742)

[**Limitations** 33](#_Toc135674743)

[**Project Evaluation** 35](#_Toc135674744)

[**Personal Reflection** 35](#_Toc135674745)

[**Future Work** 35](#_Toc135674746)

[**Conclusion** 36](#_Toc135674747)

[**References** 37](#_Toc135674748)

**Table of Flowcharts**

[Flowchart- 1 User Registration and User Login 19](#_Toc135013033)

[Flowchart- 2 User Actions 20](#_Toc135013034)

[Flowchart- 3 Admin Actions 20](#_Toc135013035)

**Table of Tables**

[Table 1: MoSCoW Prioritization List 18](#_Toc135013046)

[Table 2: Data Dictionary 21](#_Toc135013047)

[Table 3: Testing 30](#_Toc135013048)

**Table of Gantt charts**

[Gantt Chart- 1 12](#_Toc135014279)

[Gantt Chart- 2 12](#_Toc135014280)

[Gantt Chart- 3 Project Timeline 12](#_Toc135014281)

# **ABSTRACT**

Data privacy has been a very controversial topic to discuss in recent years. This report introduces a solution to the problem in the form of a messaging application which is encrypted and secured. This report follows the details of understanding the problem and coming up with the solution. Along the way, the report discover various assets required for the success of the project and walks you through all the logic behind the application. The report also discusses all the security risks you are exposed to which can occur in real life and is not limited to your online activities and how severe of impact it has had on various victims throughout the years. You can also find a step-by-step explanation of the source code along with all the tools that were used for the development as well as how one can improve upon this application within the report.

# **ACKNOWLEDGEMENT**

I would like to take this moment to thank my supervisor Mr. Resham Bahadur Pun for all the constant guidance and support whenever I needed any assistance. I would also like to extend my gratitude towards all the tutors and friends at British College Kathmandu who helped me through various hurdles I had to face while completing this project. I am also very thankful to all the resources provided to me by British College Kathmandu and Leeds Beckett University throughout my academic journey through all these years.

# **Introduction**

The world has become much smaller than it has ever been, not figuratively but in an abstract way. Figuratively, the world is still the same size but the way us humans interact with each other have changed drastically. Before the rapid advent of technology, it would take us days if not months to send a message from a place to another. Even so, we could not ensure that the message reached the destination. Nowadays, all of that can be achieved within the palms of our hands. All of us familiar with are familiar with messaging applications. We use them in our day to day life to communicate with each other. Not only does messaging applications make communication via text possible, they even allow audio and video calls. All of us know what we expect from a messaging application but very few of us know what happens under the interface of those application. We shall be discussing those problem and address ways to solve them in the following section. (Quach, et al., 2022)

## **Project Brief**

### **Problem**

Although there are plenty of messaging application available for the consumers and a lot of them provide a very healthy set of utility, it is essential to be aware that not every developer have our best interests in their minds. There are bad actors amongst them like anything else in the world and a person who is not much tech-savvy can fall victim to their bad acting. Privacy is one of the major issues in online world today. With awareness among people about how their data is being jeopardized in bad faith by the companies who provide them services has made people look for alternatives. People want applications that do not share their personal information as well as sensitive content to a third-party. In order to address this problem and cater towards the current requirements of the users, this messaging application is being made and at the same time may also help raise awareness about data privacy among users. (Anonyome Labs, 2021)

### **Solution**

There are plenty of options that are already available in the market for people looking for this kind of application but the goal is not to create a brand-new solution but improve upon already existing technologies as well as make the technology accessible to more people. It is necessary to have alternatives to all the mainstream messaging applications just because it pushes the industry forward.

### **Aims**

* To create a secure messaging application.
* To protect privacy of the user.
* To create a platform where the user feels safe communicating.
* To create a healthy platform where the users feel safe.

### **Objectives**

* The user should be able to create accounts and add friends then communicate with them.
* The conversation should take place in real-time.
* The user should be able to report and flag people and the messages they sent if they deem it inappropriate.
* The user should have control over who they choose to communicate with.

### **Explanation of the artefact**

The messaging application shall be made using ASP .NET Core and SignalR. These tools shall help add the ability of real-time communication between parties. The idea behind the application is to make it secure for users to communicate and make sure that the information shared is not compromised. The application allows the user to share files and well as text-files. The users can also see the chat logs to see the past conversations they had with the people they connected with.

The application also provides the ability to connect with new people. It can be done only after creation of a user account. The application provides various security features such as lockout, two factor authentication, and passwords stored as encrypted hashes which makes it hard for any unauthorized person to get access to their accounts.

The application flaunts a modern user interface made with the help of Tailwind CSS Framework. It is a very powerful framework and help making user interfaces very convenient and easy. Microsoft SQL Server is used for database purposes in the application and it is mapped using the help of the Entity Framework Core. The application also has the ability of sentiment analysis which can be accessed through the admin dashboard by the admin to check for harmful behavior.

# **Review of Literature**

## **Introduction to a Messaging Application**

The applications which are used in order to send or receive messages in the form of text, images, audio, videos, etc. by connecting to another user over the same platform are messaging applications. It is a popular form of communication because they are free and instant.

The oldest form of any messaging application dates back to 1961 when MIT developed CTSS, short for Compatible Time-Sharing System. It was ground breaking and allowed up to 30 people to communicate in real-time. Then there were no new progressive events in the sectors for several years. CTSS was considered way ahead of its time and was 31 years before the first SMS was sent in the year 1992. (Rouse, 2016) Chat rooms became a thing for the first time in 1988 with Internet Relay Chat (IRC) which allowed users to join them over interconnected networks. In 1996, ICQ became of the first platforms that allowed connecting with new people and sharing file over the platform. (Knight, 2022)

After that, the ball started rolling with new and better messaging platforms coming one after another. We had AOL Instant Messenger (AIM) in 1997 which introduced friends list, Yahoo! Messenger in 1998, the infamous MSN Messenger in 1999 and QQ was also launched in China in the same year. In the 2000s, there were several messaging applications laying the groundwork for what we know of them today. We had Skype, BBM, Google Talk, WhatsApp, Viber, Line, and Facebook Messenger in this decade. In the 2010s, we have Telegram, Signal, Discord, and Snapchat. Direct Messages were also integrated into social media platforms such as Twitter and Instagram. (Oca, 2020)

Going in 2020s, the technology is expected to move towards augmented reality, blockchain technology, artificial intelligence, etc. but we are also faced with a plethora of new problems and concerns. Privacy is one of the biggest concerns of the users today and the way these messaging applications have dealt with our data has not been very reassuring and it is essential to create awareness and understand what is the natural progression of the technology and how do we prevent the violation of our privacy. (O'Reilly Media, Inc., 2009)

## **Types of Messaging Applications**

Messaging Applications can be broadly divided into the following four categories explained in brief:

1. **SMS/MMS**

These are the messaging applications which use standard messaging protocols to share messages over a cellular network. iMessage and Google Messages are the examples of these messaging applications. These come by default on all the smartphones and other cellular mobile devices.

1. **Internet-based Applications**

These are the messaging applications which use internet, usually Wi-Fi and data plan, to share messages with your connections. They are a glorified form of SMS since not only can you share text and images, you can also share videos, GIFs, voices, communicate through video calls, have secure encryption, and other similar features. Usually, they are more user interactive and intuitive. Snapchat, Signal, Viber, and Telegram are some of the examples of these messaging applications.

1. **Social Media Applications**

These are the messaging applications which are integrated into social media platforms. Previously, various social media platforms such as Twitter, TikTok, and Twitter did not have the functionality to message your connections and now these features co-exist. Messages are used in these social media platforms to share posts directly into messages, share stories and live streams, and exchange dialogue.

1. **Business Applications**

These are the messaging applications that exist within various collaboration style application mainly used for efficiency purposes in project-driven environment. Microsoft Teams, Basecamp, Skype, and Slack are examples of such applications. These applications are used for communication, project management, file sharing, and video meetings.

(Barry & Tom, 2010)

## **Journal Article Review: The effect of privacy concerns, risk, control, and trust on individuals’ decisions to share personal information: A game theory-based approach**

This study looks into the rising amount of privacy concerns in users with the rapid advent in technology as well as increasing ways of abuse of those technology and how it has affected the behavior of the users. The study was conducted on 418 participants from the US and based on correlation. It studies their willingness to share information for big data analysis and how it get affected by their privacy concerns.

The study was conducted as vehicle to understand the privacy concerns and assure the users that the data in a big data analysis is often not misused but is required for a greater understanding of the nature of the industry being discussed. The study discusses briefly about the pre-existing data security notions and policies which do not protect the user’s rights completely and how these can be changed. It argues that the organization liable for violation of such privacy are often failed to be held liable for those actions to an appropriate degree.

The study argues that big data analytics provides very valuable insights to organization, consumer, and the market as a whole. But due to organizations working in bad faith or simply showing disregard in maintaining strict standards are tainting the image of the whole scientific method. Finally, the study did conclude that there was a strong correlation between privacy concerns and lack of people’s willingness to take part in big data analytics. Privacy is every user’s fundamental right and it is necessary for all the actors in the industry to act in good faith and help protect them. It is necessary for us to give the alternatives who give the users autonomy a chance so that we can decrease the perceived privacy concerns and help shed good light towards big data analytics.

(Bernard, et al., 2021)

## **Journal Article Review: Security and privacy of smartphone messaging applications**

The study points out that there has been a massive surge in smartphone users in recent years. There has been various studies studying the applications and vulnerabilities caused by this advent in technology. After seeing the success of various messaging applications such as WhatsApp and WeChat, various imitators have tried to enter the market with similar applications.

Now, smartphone messaging applications have always tried to collect as little data as possible on sign up as possible. Often times, they only amount to the phone number. Then, the phone number receives an OTP to verify and then you successfully create an account. The study employs various exploits to see if they could take advantages of the vulnerabilities with this approach. The vulnerabilities they seem to have caught the most are authentication and account hijacking which means someone is able to take over your account by tricking you into giving them the OTP. Sender ID Spoofing or Message Manipulation is another common vulnerability. It means being able to spoof messages from your account without having the control over your account. These attacks were found in outdated applications.

In conclusion, the study was able to find and exploit various vulnerabilities which could lead to the bad actors having access of your account and your sensitive information could be compromised. Such exploits are only possible due to lack of oversight from the publisher’s or developer’s side and the oversight usually is a direct result of disregard of the privacy of their users. The study urges user to be very conscious of what kind of applications they are using and are willing to share their personal information with.

(Schrittwieser, et al., 2015)

## **Pre-Print Article Review: HOW DATA CAN BE USED AGAINST PEOPLE: A CLASSIFICATION OF PERSONAL DATA MISUSES**

The study claims that the topic of data privacy is shrouded with misinformation and confusion. Many users are still simply unaware of the importance of their data privacy and how their personal information could be or is being used against them. The study doesn’t simply look at the privacy concerns but also talks about how these data have been used in the past to spread paranoia and misinformation, targeted brainwashing and government surveillance. The study focuses on real-world events to demonstrate their points instead of simply saying ‘big data bad’.

The study points out that these data has been used in real world for various notorious reasons such as ridiculing a person for being themselves and putting more eyes on themselves. Voyeurism is a misconduct that has ruined life of various innocent people. It has been achieved by the means of spyware and hidden cameras in private places. Various users often receive threats that can cause them physical and mental harm and although it may not always materialize, it is still a major inconvenience while carrying out your daily activities. There will be attacks that will be targeted directly towards you based on your likes and dislikes offering you a reward for some unsuspecting thing you might be asked to do but it happens to be malicious.

People’s personal information getting compromised have led to various episodes of blackmail to pursue the blackmailer’s personal agendas. On top of that, various states are known to monitor all of your activities through cameras, transactions, and online presence in order to identify you political values and oppress your voice if it needs to be. There is a plethora of other ways you data can be used against you if compromised. To name a few, publication of nudes, discrimination based on your opinions in workplace, subject to bullying based on your insecurities and vulnerabilities, victim of a phishing attack, spam calls, real-life stalkers, sexual predatory behavior towards you, identity theft, etc. And these are just some examples and there are much more ways you could be harmed. Users tend to overlook and fixate on a few talked about issues in the mainstream and simply assume that they are not prone to such threats and they should not worry about it but they fail to realize that their personal information can be used against them in more ways than they could ever imagine and all of us can fall victim to such threats.

(Kröger, et al., 2021)

# **Review of Technology**

## **C#**

C# is a type of programming language which runs on the .NET Framework developed by Microsoft. It is an open-source and cross-platform programming language which is also object-oriented. This makes the language very versatile because of the range of application it allows us to create such as web applications, packaged applications, smartphone applications, etc. C# is popular because of its integration to other applications especially for game developers as well as the massive community it has been able to garner over the years. C# is the programming language of choice for this project. (Solis & Schrotenboer, 2018)

## **Visual Studio**

Every programming language goes hand in hand with an IDE. IDE stands for Integrated Development Environment which is just what the name says. It is an environment created for the sole purpose of writing development codes. Usually, they contain various quality of life features which make the job way easier for the developers. One can simply write a code in a notepad but the context and the additional help these IDE provide make for a seamless development process. Visual Studio is the IDE of choice for this project simply because of the comfort of having worked with this IDE in the past. (MacDonald, et al., 2010)

## **SignalR**

SignalR is an ASP.NET library. It is used to add real time functionality to an application based on web which is exactly what we need for this project. It helps the process of communication between the server and clients as it is prompted or required by either the server or the client. It is an open-source library which works very well across multiple connections which is the reason it if fit to become a chat box application. Microsoft has developed a tool known as SignalR that brings live interactive capabilities to software applications. SignalR's primary function is to allow servers to swiftly push updates to connected clients without waiting for a data request, promoting a real-time data exchange. This interaction model enables a two-way communication channel between client and server, leading to immediate data propagation. Incorporating any form of real-time web functionality in an app becomes more straightforward with SignalR. It eases the integration of quick content dispatch through technologies like WebSocket. SignalR's usage can be exemplified in real-time data reflecting dashboards or chat functionalities. It is used for APIs that execute functions on the client-side which means a very strong servers are not required for this. (Goswani, 2017)

## **ASP .NET Core MVC**

ASP. NET Core MVC is a type of web development framework. It employs the use of Model View Controllers i.e. MVC design pattern for the purpose of creation of web apps and APIs. The MVC helps distribute the application into three different components being the models, views and controllers. In this context, model refers to the overall logic and flow of the application. ASP.NET Core MVC leverages the Model-View-Controller (MVC) design pattern to structure web applications into three interconnected components, each with its distinct role. The Model component acts as the application's brain, handling data interaction and business logic. It interacts with the database, performs necessary calculations, and defines data structures without awareness of the View and Controller. This separation ensures that the manipulation of data is isolated from how it is displayed or interacted with, improving the maintainability and scalability of the application. The view refers to the UI of the application and the overall feel and look whereas finally the controllers refers to the user and client side interaction such as what action needs to be triggered when the users performs a certain action. This framework is being used in the project because it fits all the criteria necessary to build a chatbot. (Microsoft, 2023)

## **Entity Framework Core**

Entity Framework Core is an open-source version of the Entity Framework. It is a data access technology which helps us map the relationship between various objects in the database which allows for a seamless communication and reduces errors since any redundancy that could be caused by humans in completely removed. Entity Framework Core, often shortened to EF Core, is a tool built by Microsoft that simplifies database interaction for developers. It allows for communication with databases through .NET objects, removing the need for extensive manual coding. It's a resource available to everyone as it is open-source. EF Core simplifies various tasks like finding, adding, updating, or deleting data in a database. It also handles transactions and alterations in databases and supports stored procedures. It can operate with many different databases, such as SQL Server, SQLite, MySQL, and PostgreSQL. In summary, EF Core is a practical tool that eases the development of applications that require database interaction. It can help carry out simple CRUD operations and is a part of the ASP.NET framework. This is good for this project as we shall be employing the use of the Microsoft SQL Server. (DEVART, 2021)

## **Tailwind CSS**

Tailwind CSS is a framework which allows us to create responsive designs quickly without any rigorous knowledge of CSS. It is a very fast and reliable framework and is also open-source. It also consists of a range of pre-made components of UI which help make the front-end of a web application much easier. It employs the use of tools such as HTML, React, Angular, Vue, and other similar tools to create those components. This is perfect for this project because of the simplicity of creating those UI components with minimal effort and most effort could be employed on the logic and execution of the application. (Fitzgerald, 2022)

## **Microsoft SQL Server**

Microsoft SQL Server is a type of relational database management system i.e. RDBMS. It provides a bunch of tools which makes analytics and data handling easy when dealing with huge volumes of data. Microsoft SQL Server, often abbreviated as MS SQL Server, is a relational database management system (RDBMS) designed by Microsoft. Built on the SQL (Structured Query Language) framework, its primary function is to manage and store data in an organized manner. The system is highly scalable, catering to substantial data volumes while maintaining optimal performance. Noteworthy for its robust security features, MS SQL Server safeguards sensitive data through encryption methods and rigorous user access controls. Additionally, it includes automated backup and restore functions to ensure data preservation and recovery, fortifying its utility in data management. This is perfect for this project which is also attempting to achieve the same. It also works very fluently with all the programming languages and the tools used in this project. (PÉREZ, 2021)

## **ML.NET**

ML.NET is a machine learning framework for .NET. It has support for sentiment analysis, fraud detection, prediction, and other similar models. For this project, we are going to be using the sentiment analysis. Sentiment Analysis is going to be an integral part of the application where the admin can check a certain piece of text for harmful implications and this is a perfect framework for this application. (Shanu, 2020)

## **GitHub**

GitHub is a web-based service for project management and version tracking. It permits individuals and teams to collaboratively work on projects, primarily code-based ones. It offers a space, termed a repository, for storing, managing, and tracking alterations to your projects. Users can make duplicates of a project, known as forks, without impacting the original, and suggest modifications to others' projects through pull requests. Its integrated system for tracking versions enables users to review modifications, understand the progression, and if necessary, revert to a previous state. Through GitHub Actions, it also provides the capability to automate various processes in software development. These features make GitHub an essential resource for developers, both working individually or as part of a team.

# **Development Methodology**

This project is going to be based upon Agile Method of software development life cycle. For the development part, we shall be going to employ the use of iterative method.

Iterative Method of software development lifecycle means that the application shall be divided into multiple cycles of sub activities that needs to be done for the development rather than a linear progression. This helps the developer make incremental changes and understand how each component interact with each other. Iterative Method can be further divided into five phases which are briefly discussed below:

## **Requirement Phase**

This is the first phase of the iterative method of software development lifecycle. This is where we start planning about how you want to go about all the various parts of development process, how you want to divide them and how you want to start working on them, what sort of tools you are going to need and what do you expect out of your final product. You also want to think about the resources you are going to require and how accessible they are as well as the learning curve and your time management ability in terms of learning new technologies.

For the purpose of this project, all the subsystems and the features were thought through first, then the tools were decided based on familiarity and ease of use as well as utility to the end goals.

## **Design Phase**

Design is always going to be one of the major part of a system from the perspective of a user. Design refers to not only the user interface of a system but also how the user interacts, how it feels using the system, the accessibility features that are baked into the system, and how easily can a user navigate the system. Once you develop a design language for the system, you start building everything around it to ensure maximum efficiency.

For the purpose of this project, Tailwind CSS was decided to create the user interface. Once that was taken care of, it was decided that various quality of life additions needed to be made to the application such as reporting inappropriate behaviors, ability to check for harmful messages using sentimental analysis, and other similar features.

## **Development Phase**

Development is always going to be the most complex part of software development life cycle. You want to make sure have all your subsystems and components divided into different groups before you start working on development as it allows you to work on each individual chunk without hampering the other. You have already set all the necessary guidelines at this point and you just want to develop the application around those guidelines making sure that you tick all the boxes one at a time. Stop worrying about the big picture and make sure that each individual block of content are just like you envisioned and the complete picture shall come together at the end of it all.

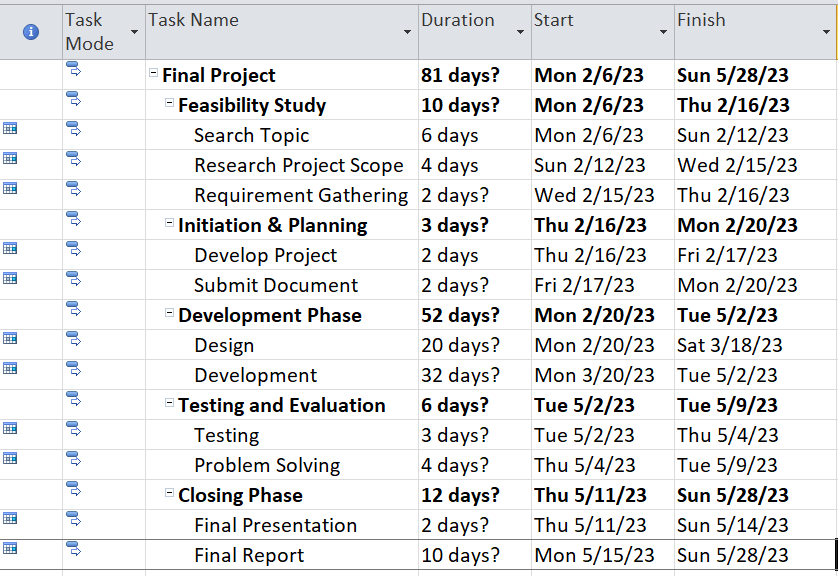
## **Testing Phase**

This is the phase of the development cycle when you are mostly done with the work. Now, you simply need to go through the application you have created and check if all the features are working as you intended them to. You might also want to check for any unforeseen circumstances that might have occurred after putting together the system even though the system might be working fine when it was in individual blocks. The first thing to test should always be the most important features of the application and then you should start look for any inconsistencies in the application as well as the user interface. You keep on repeating the process until the application becomes polished.

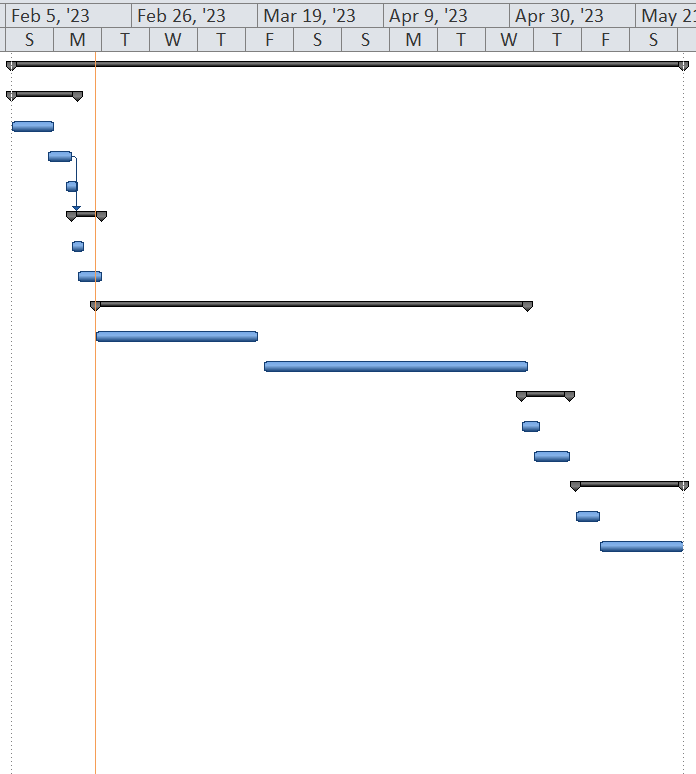
## **Review and Deployment Phase**

This is the final phase of the iterative method of the software development lifecycle. In this phase, you want to look into the final product and see if it fits your vision. After making necessary changes, you need to plan where you want to deploy the application and how one would be able to access it. This phase is mostly about cross-checking multiple times and making continuous revisions until you are satisfies. (Goodman, et al., 2012) (Whiteley, et al., 2021)

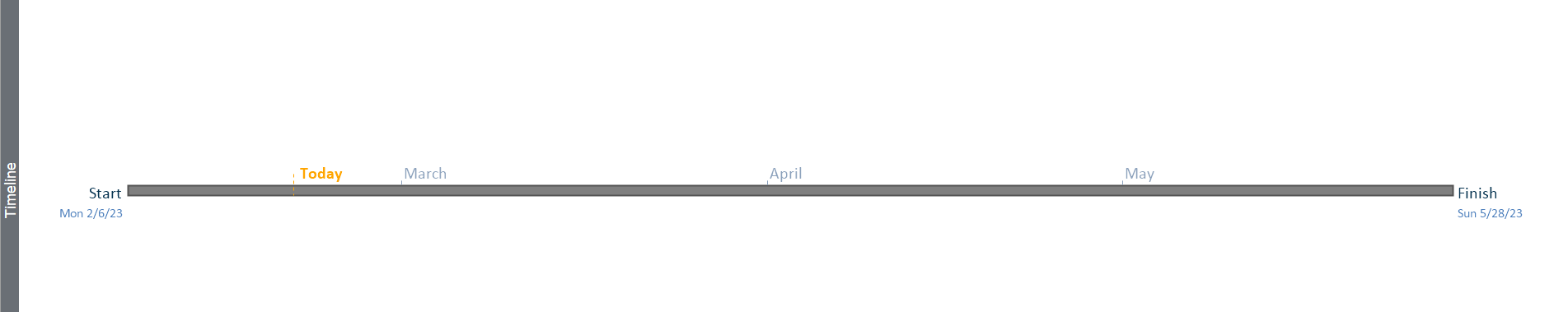
# **Gantt chart**



Gantt Chart-



Gantt Chart-



Gantt Chart- Project Timeline

# **Product Design**

There were various factors that were taken in consideration in the development lifecycle as we have already discussed above. There are various that are used for design of the application which are discussed one by one below starting with MoSCoW priority list.

## **MoSCoW Prioritization**

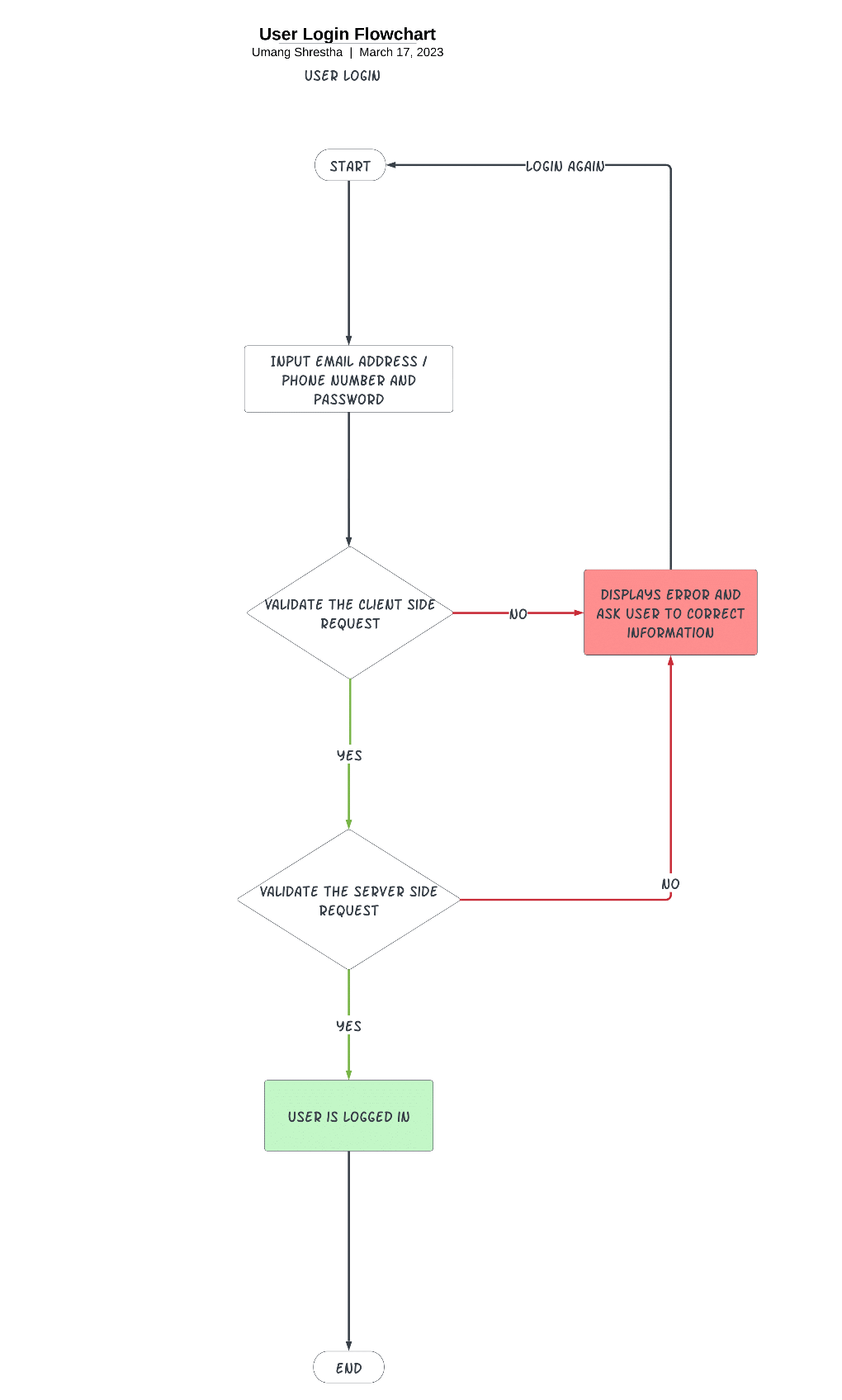
This tool helps us build a baseline of what we want from our application and where does the priority of the features stand. This helps us make educated decisions about what needs to be dealt with first and also helps create a flow in development process since it takes away the factor that is uncertainty.

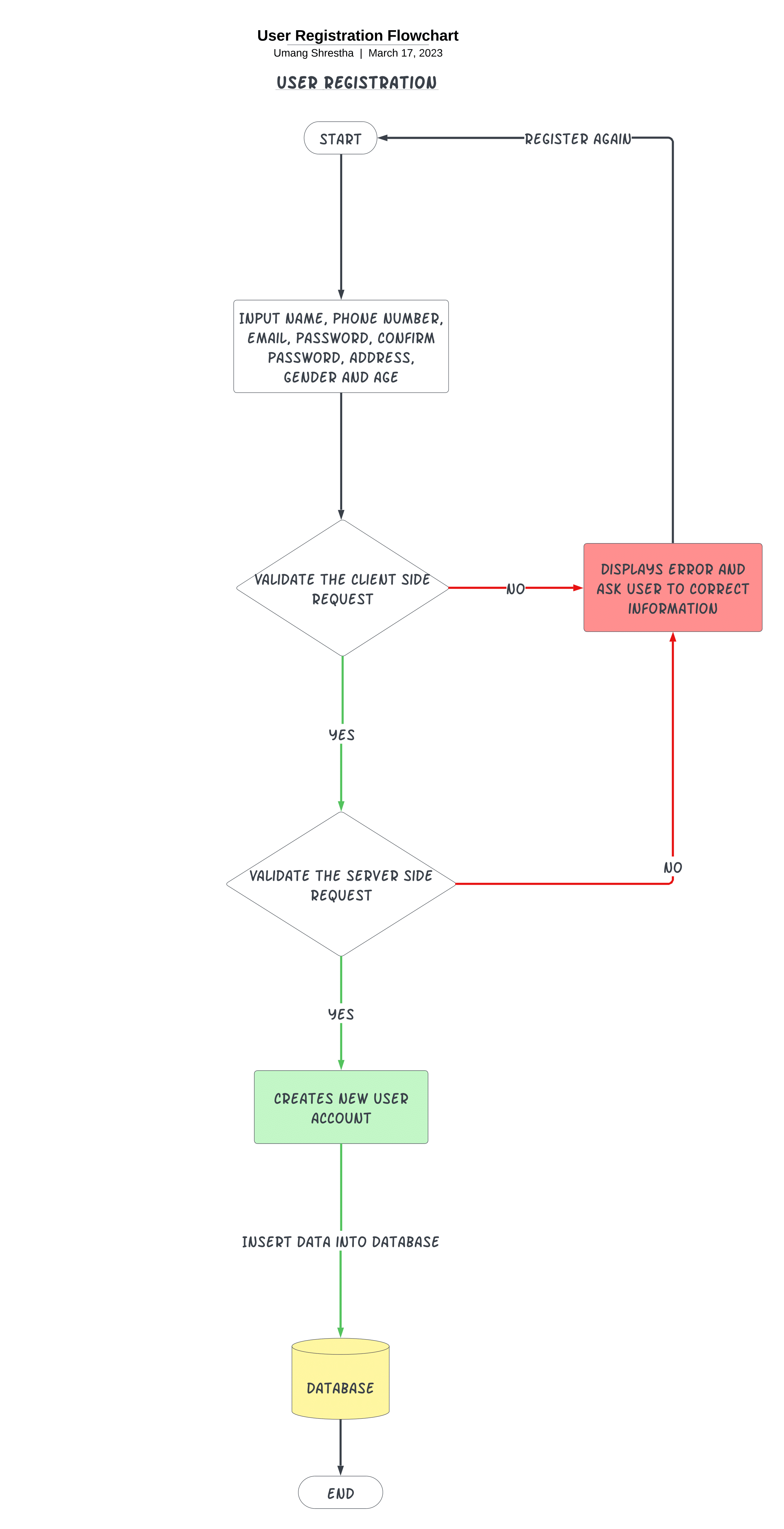
|  |  |  |
| --- | --- | --- |
| Test Code | Scenario | MoSCow |
| Functional Requirements | | |
| TC-000 | The users should have the ability to share files and text messages with each other in real-time. | Must Have |
| TC-001 | The users should be able to view the chat history. | Should Have |
| TC-002 | The users will have to login before exchanging conversations. | Must Have |
| TC-003 | The users will be able to create chat rooms. | Should Have |
| TC-004 | The users will have the ability to leave a chat room. | Should Have |
| TC-005 | The users will be able to update their password | Should Have |
| TC-006 | The users will have the ability to react to messages using an emoji. | Could Have |
| TC-007 | The users will be able to report inappropriate behavior. | Should Have |
| TC-008 | The users will have the ability to add other users as friends. | Could Have |
| TC-009 | The admin will have the ability to check for sentiment analysis. | Should Have |
| TC-010 | The admin should be able to update user information. | Could Have |
| TC-011 | The admin should have the ability to restrict an account from using the application temporarily or permanently. | Could Have |
| Non-Functional Requirements | | |
| TC-100 | The application will be able to handle huge amount of traffic. | Must Have |
| TC-101 | The application will have minimal data loss with almost no errors. | Should Have |
| TC-102 | The application will be intuitive and user-friendly. | Should Have |
| TC-103 | The source code for the application will be well-structured and easy to maintain. | Should Have |
| TC-104 | The application will provide encryption and authentication to protect from unauthorized access. | Should Have |

Table : MoSCoW Prioritization List

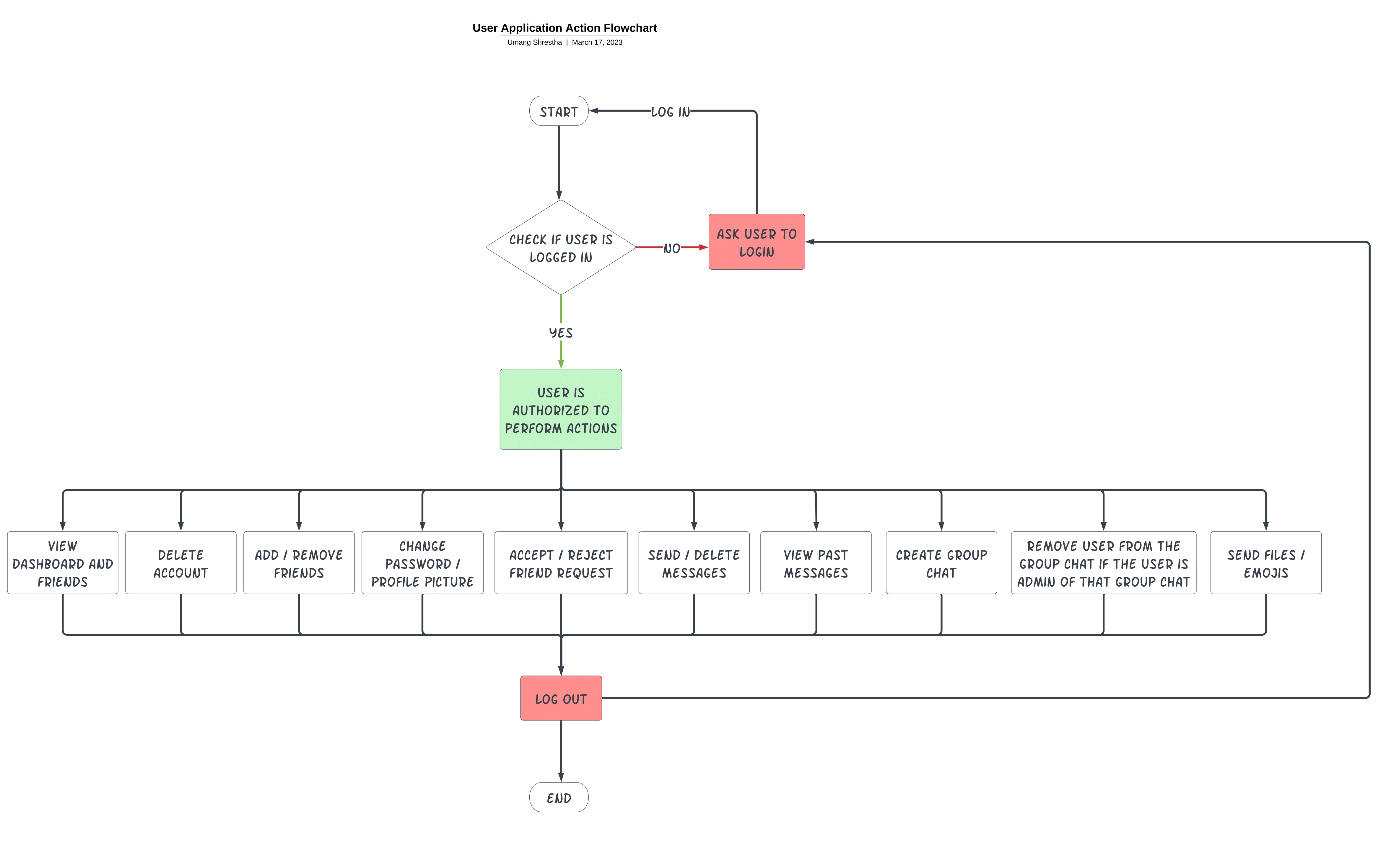
## **Workflow Flowchart**

The workflow flowchart is another tool which helps us understand how we shall program user interaction into the application. It helps us understand the user interface elements that are to come and help establish a picture of what the system would look like. Here, we have three different workflow flowchart. All three of them are meant to represent the interaction a user has with the application. In the Flowchat-1, we see the interaction while registering a new account, Flowchart-2 shows the interaction while logging in with a pre-existing account. Flowchart-3 shows all the user actions that are possible after logging in. Finally, Flowchart-4 shows the actions after logging in as an admin.

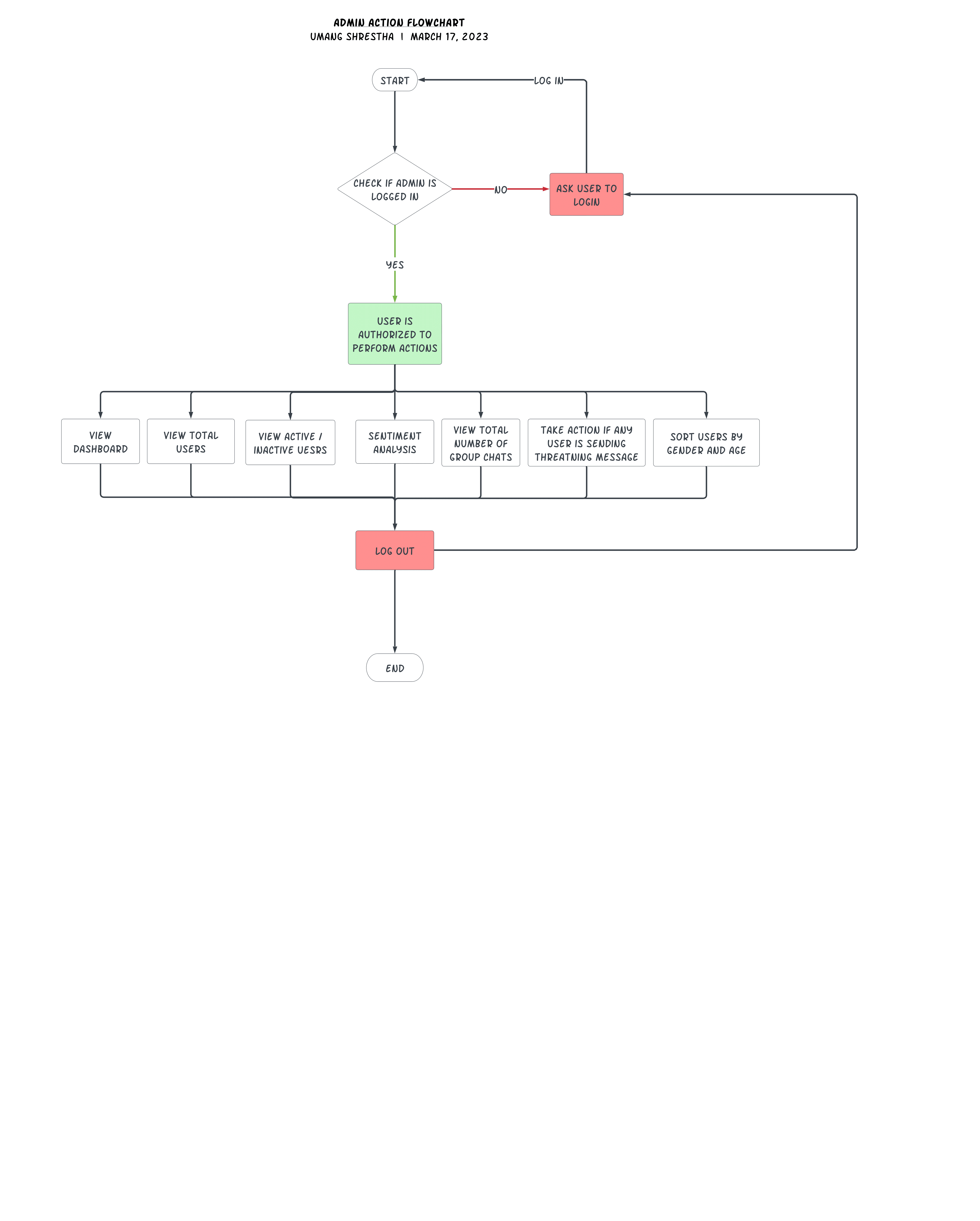




Flowchart- User Registration and User Login



Flowchart- User Actions



Flowchart- Admin Actions

## **Entity Relationship Diagram**

Based on the Entity Relationship Diagram and the Data Dictionary, we get a rough idea of what the application can and cannot do. Obviously, it is not going to show the complete features but it does show what actions and interactions can be taken which are recorded using different attributes and their description. We can also use the diagram to understand the interaction between these attributes.

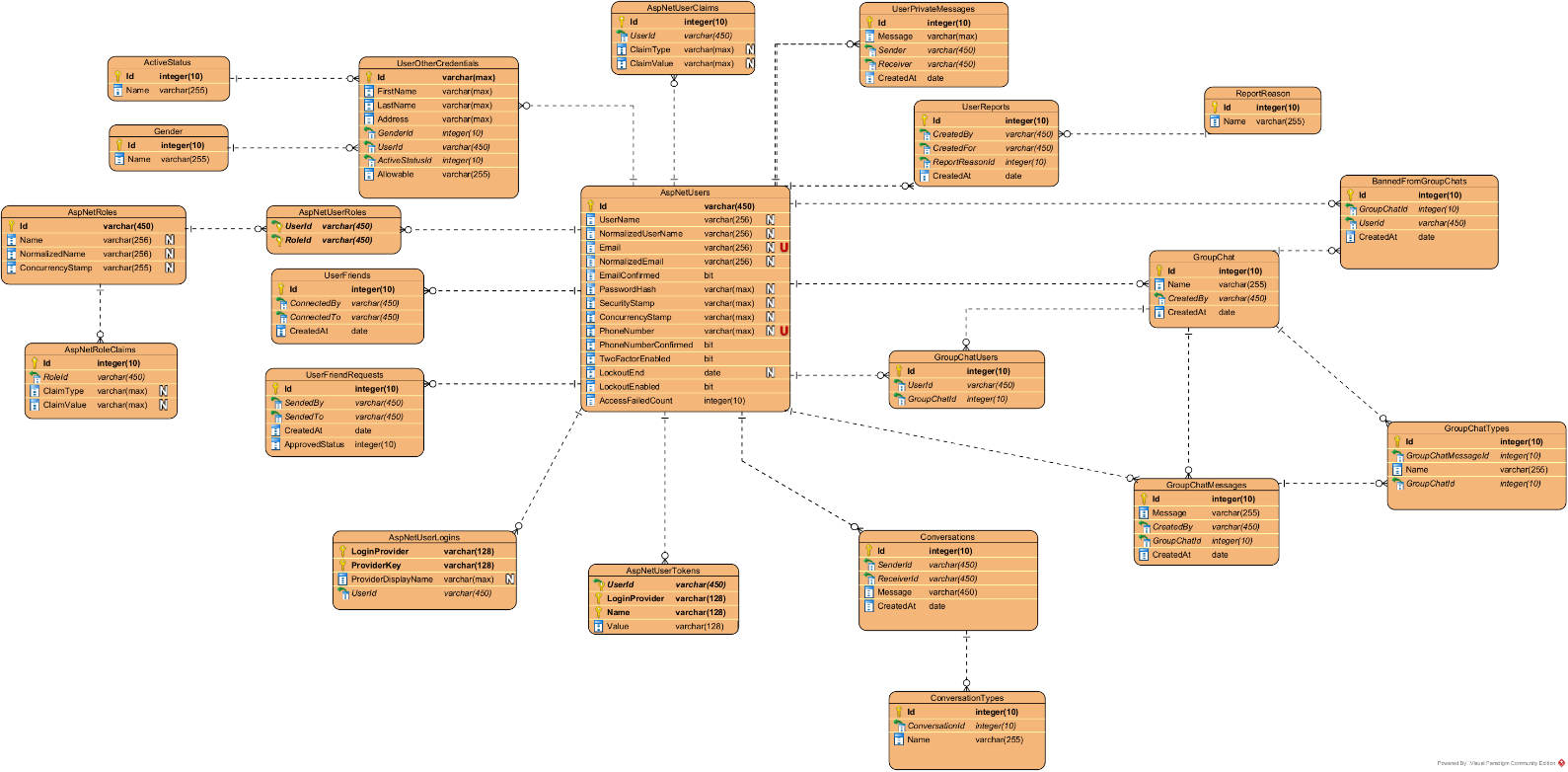


Figure : ERD

### Data Dictionary

Table : Data Dictionary

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | | Data type | | Field Length | | Constraint | | | Description | |
| AspNetUsers | | | | | | | | | | |
| Id | | varchar | | 450 | | Primary key | | | Unique identification string for each individual user. | |
| UserName | | varchar | | 256 | | Nullable | | | The name of the user. | |
| NormalizedUserName | | varchar | | 256 | | Nullable | | | The username in normalized form. | |
| Email | | varchar | | 256 | | Nullable | | | The email address of the user. | |
| NormalizedEmail | | varchar | | 256 | | Nullable | | | The email address of the user in normalized form. | |
| EmailConfirmed | | Bit | | default | | Not null | | | The confirmation status of the email address used by the user. | |
| PasswordHash | | varchar | | Max | | Nullable | | | The hash of the password created by the user. | |
| SecurityStamp | | varchar | | Max | | Nullable | | | The security stamp i.e. tracking of the most recent changes made to user profiles. | |
| ConcurrencyStamp | | varchar | | Max | | Nullable | | | The concurrency stamp i.e. the most up-to-date version of user information. | |
| PhoneNumber | | varchar | | Max | | Nullable | | | The phone number of the user. | |
| PhoneNumberConfirmed | | Bit | | default | | Not null | | | The confirmation status of the phone number used by the user. | |
| TwoFactorEnabled | | Bit | | default | | Not null | | | The status of whether the user has enabled two factor authentication or not. | |
| LockoutEnd | | Date | | default | | Nullable | | | The time when the user can try to login again after using wrong credentials multiple times. | |
| LockoutEnabled | | Bit | | default | | Not null | | | The status of whether the profile is locked out or not due to incorrect login attempts. | |
| AccessFailedCount | | int | | 10 | | Not null | | | The amount of times when there have been failed login attempt. | |
| AspNetUserLogins | | | | | | | | | | |
| LogInProvider | | varchar | | 128 | | Primary Key | | | The information of the authentication middleware provider for user authentication. | |
| ProviderKey | | varchar | | 128 | | Primary Key | | | A unique identification string for the authentication provider. | |
| ProviderDisplayName | | varchar | | max | | Nullable | | | The display name for the authentication provider. | |
| UserId | | Varchar | | 459 | | Not null | | | A unique user identification string. | |
| UserOtherCredentials | | | | | | | | | | |
| Id | | varchar | | Max | | Primary Key | | | A unique identification strings. | |
| FirstName | | Varchar | | max | | Not null | | | The first name of the user. | |
| LastName | | Varchar | | Max | | Not null | | | The last name of the user. | |
| Address | | Varchar | | Max | | Not null | | | The address of the user. | |
| GenderId | | Int | | 10 | | Not null | | | A unique index for the gender of the user. | |
| UserId | | Varchar | | 450 | | Not null | | | A user identification number for the user. | |
| ActiveStatusId | | Int | | 10 | | Not null | | | A check for whether the user is currently active or not. | |
| ActiveStatus | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Name | | Varchar | | 255 | | Not null | | | An array of various possible active status for a user. | |
| Gender | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Name | | Varchar | | 255 | | Not null | | | An array of various possible genders for a user. | |
| UserFriendRequests | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| SendedBy | | Varchar | | 450 | | Not null | | | The username of the person who created the friend request. | |
| SendedTo | | Varchar | | 450 | | Not null | | | The username of the person who is asked for the friend request. | |
| CreatedAt | | Date | | default | | Not null | | | The time when the friend request was made. | |
| ApprovedStatus | | Int | | 10 | | Not null | | | The status of the request by the person to whom the request was made. | |
| AspNetUserTokens | | | | | | | | | | |
| UserId | | varchar | | 450 | | Foreign Key | | | A unique identification string for the user. | |
| LoginProvider | | varchar | | 128 | | Primary Key | | | The authentication provider for the login. | |
| Name | | varchar | | 128 | | Primary Key | | | The name of the user. | |
| Value | | varchar | | 128 | | Not null | | | The value of tokens sent by the server for future sessions. | |
| UserFriends | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| ConnectedBy | | Varchar | | 450 | | Not null | | | The person who sent the friend request. | |
| ConnectedTo | | varchar | | 450 | | Not null | | | The person who accepted the friend request. | |
| CreatedAt | | Date | | default | | Not null | | | The time when the friend request was made. | |
| AspNetUserClaims | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| UserId | | Varchar | | 450 | | Not null | | | A unique identification string for the user. | |
| ClaimType | | Varchar | | Max | | Nullable | | | The type of the claim. | |
| ClaimValue | | Varchar | | Max | | Nullable | | | The value of the claim which is communicated between the user and the server. | |
| UserPrivateMessages | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Message | | Varchar | | Max | | Not null | | | The message exchanged between the parties. | |
| Sender | | Varchar | | 450 | | Not null | | | The sender of the said message. | |
| Reciever | | varchar | | 450 | | Not null | | | The receiver of the said message. | |
| CreatedAt | | Date | |  | | Not null | | | The time record of the moment the message was created or shared between the two parties. | |
| UserReports | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| CreatedBy | | Varchar | | 450 | | Not null | | | The name of the user who created the user report. | |
| CreatedFor | | Varchar | | 450 | | Not null | | | The name of the user who was reported. | |
| ReportReasonId | | Int | | 10 | | Not null | | | The index of the reason for the report to the user. | |
| CreatedAt | | Date | | default | | Not null | | | The exact time of when the report was created by the user. | |
| ReportReason | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Name | | Varchar | | 255 | | Not null | | | An array of possible reasons for a report. | |
| GroupChat | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Name | | Varchar | | 255 | | Not null | | | The name of the group chat. | |
| CreatedBy | | Varchar | | 450 | | Not null | | | The name of the user who created the group chat. | |
| CreatedAt | | Date | | default | | Not null | | | The time when the group chat was created at. | |
| GroupChatUsers | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| UserId | | Varchar | | 450 | | Not null | | | A unique identification string for the user. | |
| Group ChatId | | Int | | 10 | | Not null | | | A unique identification number for the group char. | |
| GroupChatMessages | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| Message | | Varchar | | 255 | | Not null | | | The message that was shared in the group chat. | |
| CreatedBy | | Varchar | | 450 | | Not null | | | The name of the user who created and sent the message. | |
| Group ChatId | | Int | | 10 | | Not null | | | A unique identification number to the group chat. | |
| CreatedAt | | Date | | default | | Not null | | | The moment in time when the message was shared in the group chat. | |
| Conversations | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| SenderId | | Varchar | | 450 | | Not null | | | A unique identification string for the person who sent a message. | |
| ReceiverId | | Varchar | | 450 | | Not null | | | A unique identification string for the person who received the message. | |
| Message | | Varchar | | 450 | | Not null | | | The message which was shared among the members in the group chat. | |
| CreatedAt | | Date | | default | | Not null | | | The moment in time when the messaged were shared in the group chat. | |
| AspNetUserRoles | | | | | | | | | | |
| UserId | | Varchar | | 450 | | Foreign Key | | | A unique identification string for the user. | |
| RoleId | | Varchar | | 450 | | Foreign Key | | | A unique identification string for the role of the user. | |
| AspNetRoles | | | | | | | | | | |
| Id | | Varchar | | 450 | | Primary Key | | | A unique identity string for roles of the user. | |
| Name | | Varchar | | 256 | | Not null | | | The name of the user. | |
| NormalizedName | | Varchar | | 256 | | Not null | | | The normalized name of the user. | |
| ConcurrencyStamp | | Varchar | | 255 | | Not null | | | A stamp which is used to check for concurrency in the dataset with certain changes made to it. | |
| AspNetRoleClaims | | | | | | | | | | |
| Id | | Int | | 10 | | Primary Key | | | A unique identification number. | |
| RoleId | | Varchar | | 450 | | Not null | | | A unique identification string for the roles of the user. | |
| ClaimType | | Varchar | | Max | | Not null | | | The type of claim. | |
| ClaimValue | | Varchar | | max | | Not null | | | The claim values. | |
| BannedFromGroupChats | | | | | | | | | | |
| Id | | Int | | 10 | | Not null | | | A unique identification string for the roles of the user. | |
| GroupChatId | | Int | | 10 | | Not null | | | A unique identification number of GroupChat. | |
| Name | | Varchar | | 255 | | Not null | | | Name of the Group Chat. | |
| UserId | | Int | | 10 | | Not null | | | A unique identification number of User. | |
| GroupChatTypes | | | | | | | | | | |
| Id | Int | | 10 | | | Not null | A unique identification number. | | | |
| GroupChatMessageId | Int | | 10 | | | Not null | A unique identification number of GroupChatMessages. | | | |
| Name | Varchar | | 255 | | | Not null | Name of Group chat types | | | |
| GroupChatId | Int | | 10 | | | Not null | A unique identification number of GroupChat. | | | |
| ConversationTypes | | | | | | | | | | |
| Id | Int | | | | 10 | | | Not null | | A unique identification number. |
| ConversationId | Int | | | | 10 | | | Not null | | A unique identification number. |
| Name | Varchar | | | | 255 | | | Not null | | Name of Conversation type. |

## **User Interface**

Before we start working on the backend intricacies, we are going to work on the frontend of the application. Following is how the user interface for the messaging application looks like:

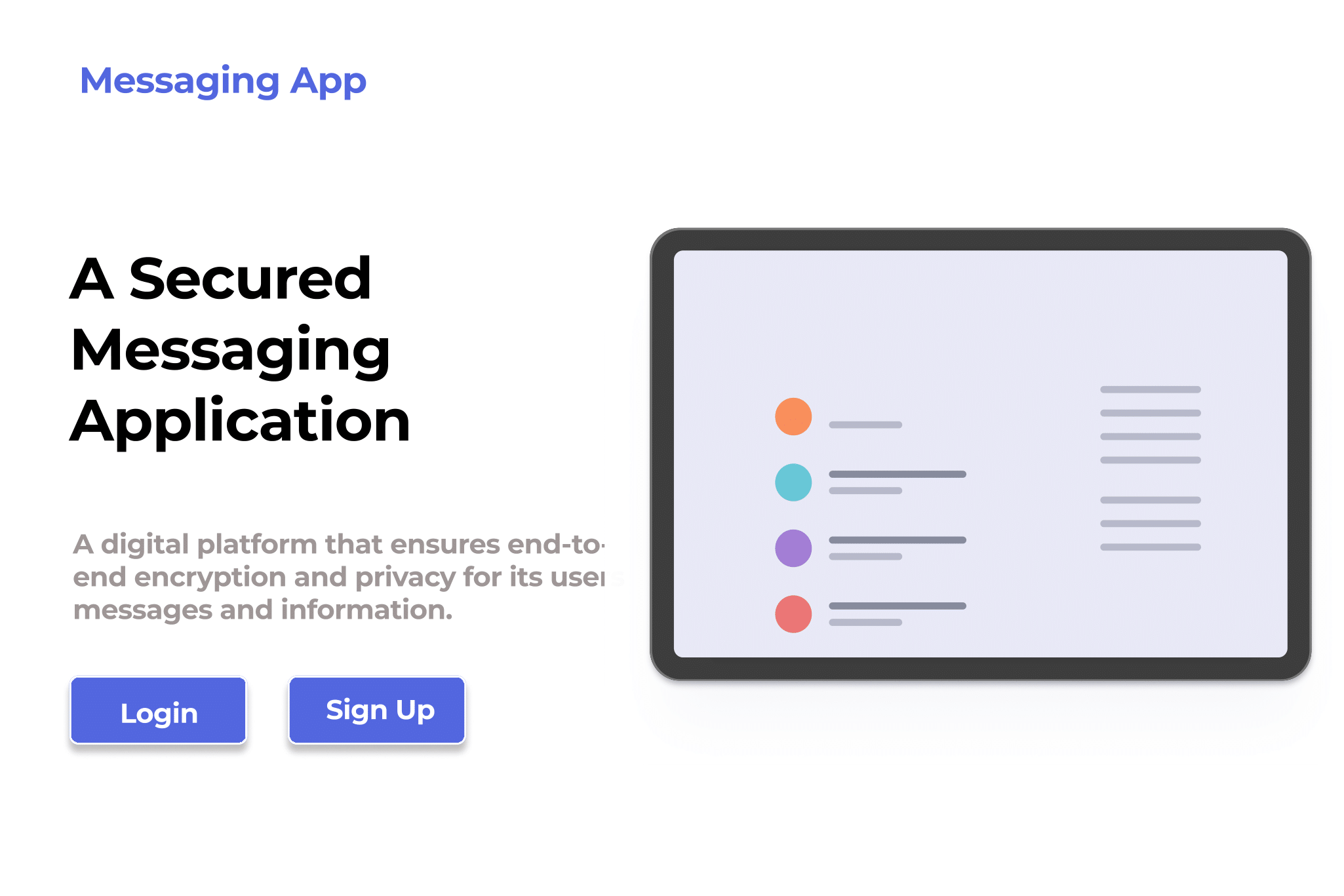


Figure : Homepage

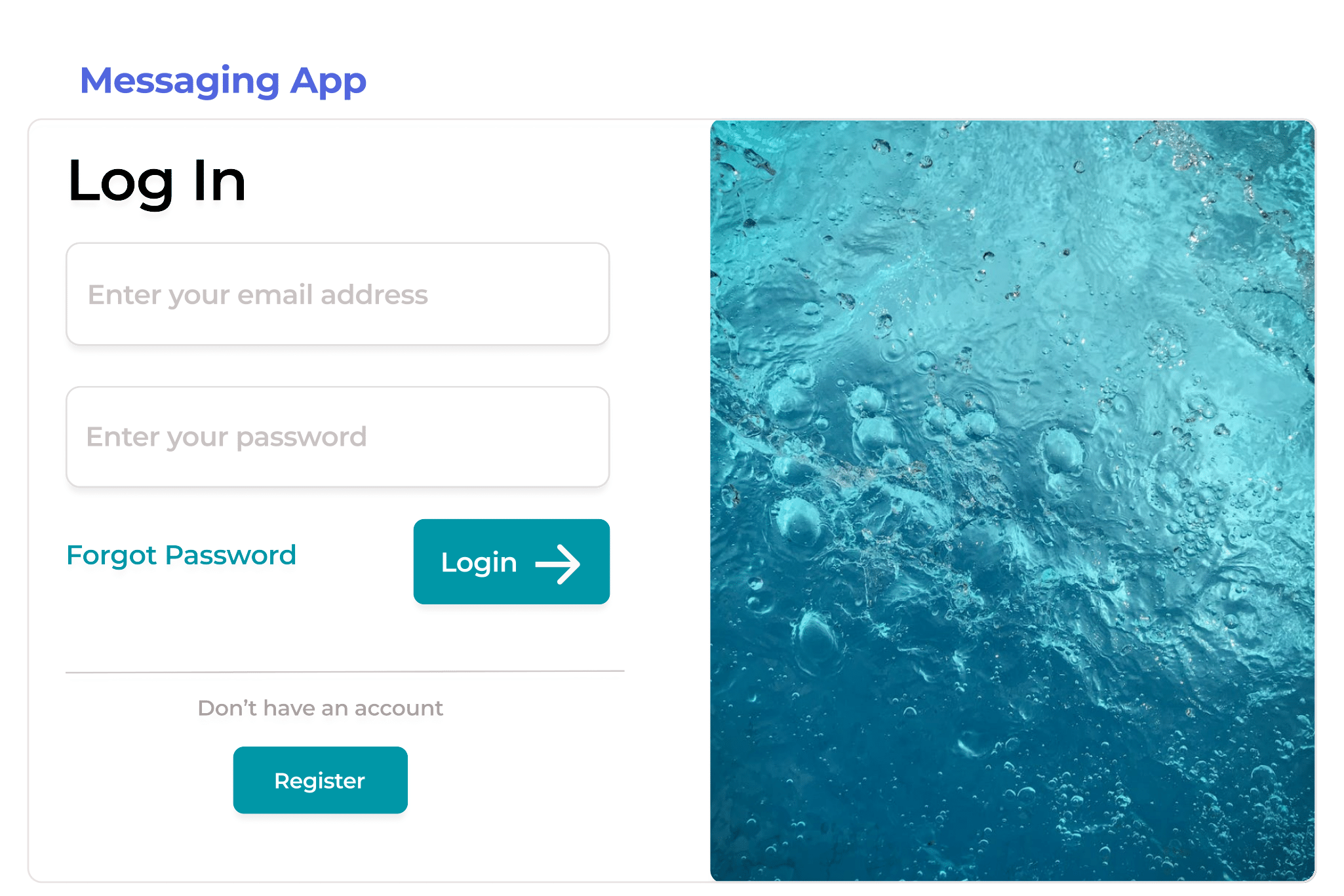
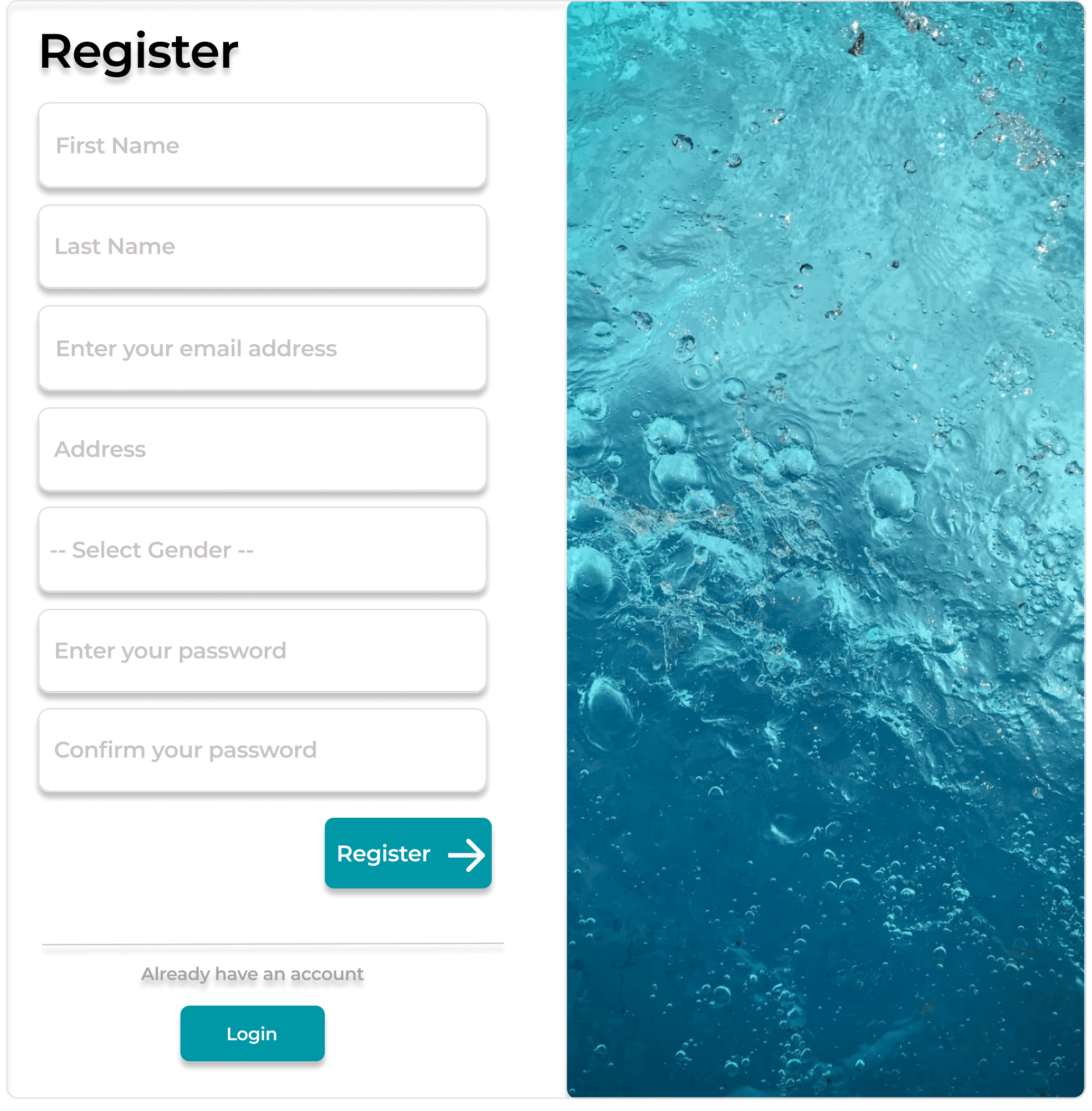


Figure : Register Page Figure : Login Page

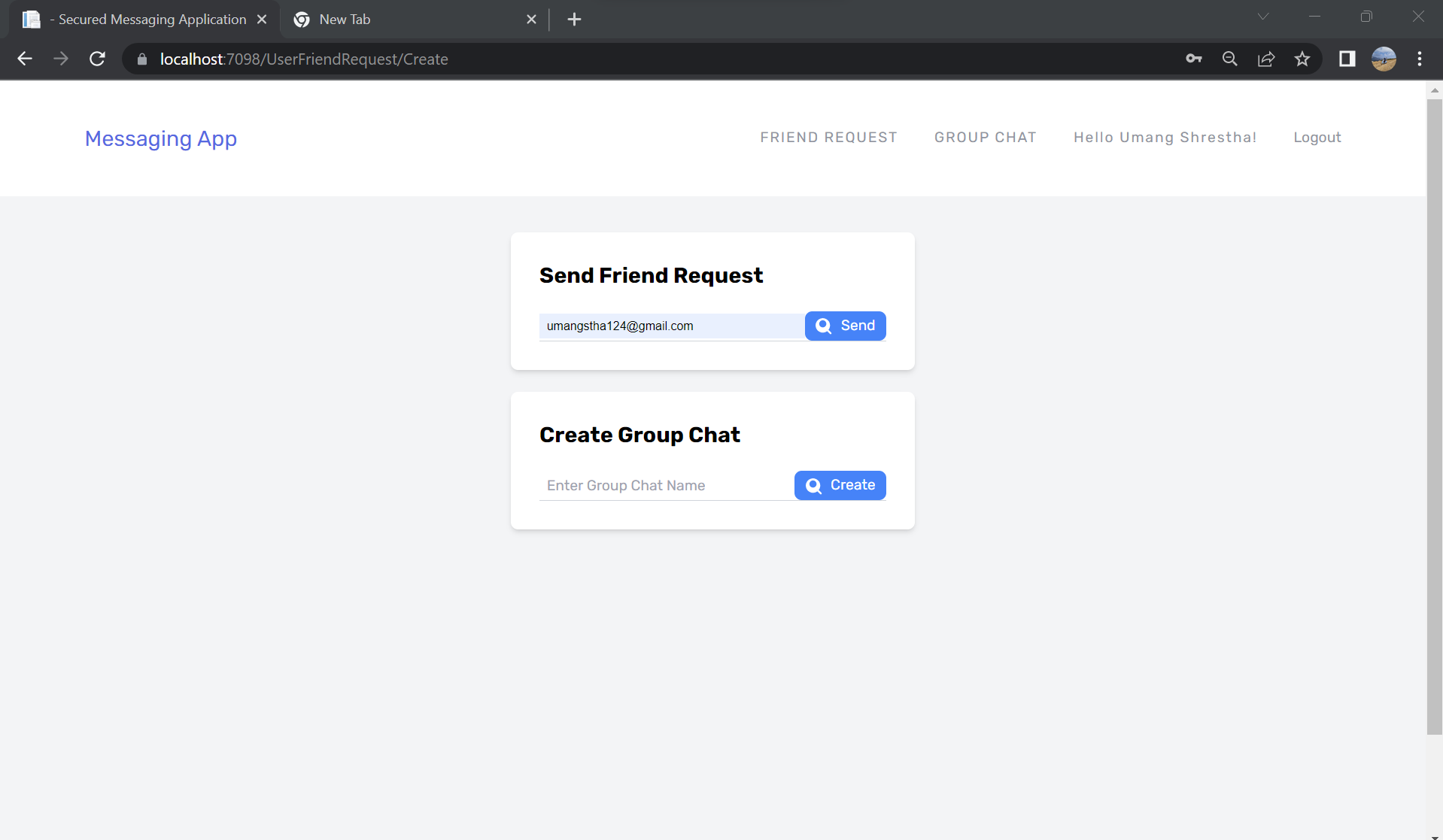


Figure : Send Friend Request

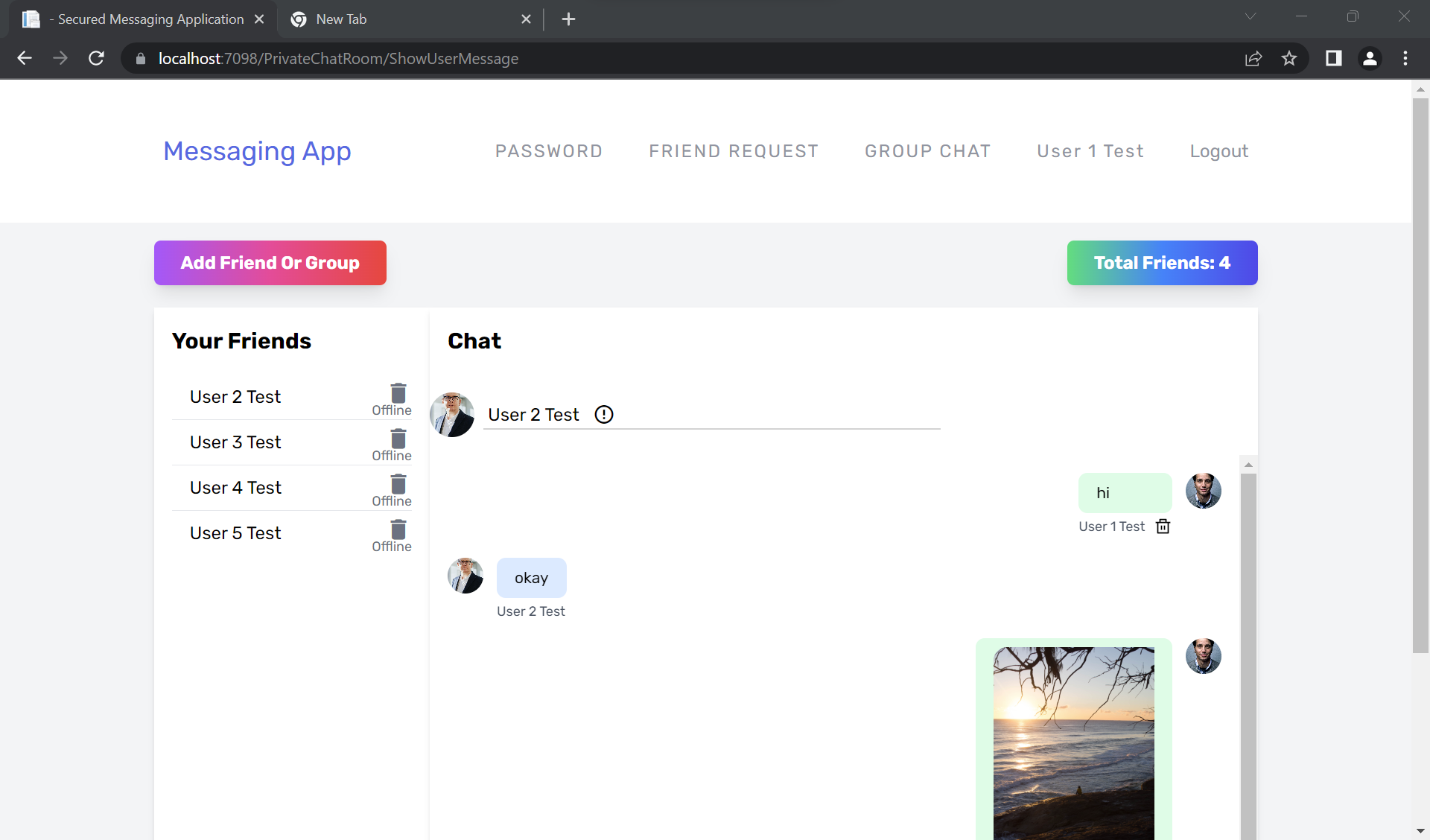


Figure : Chat Area

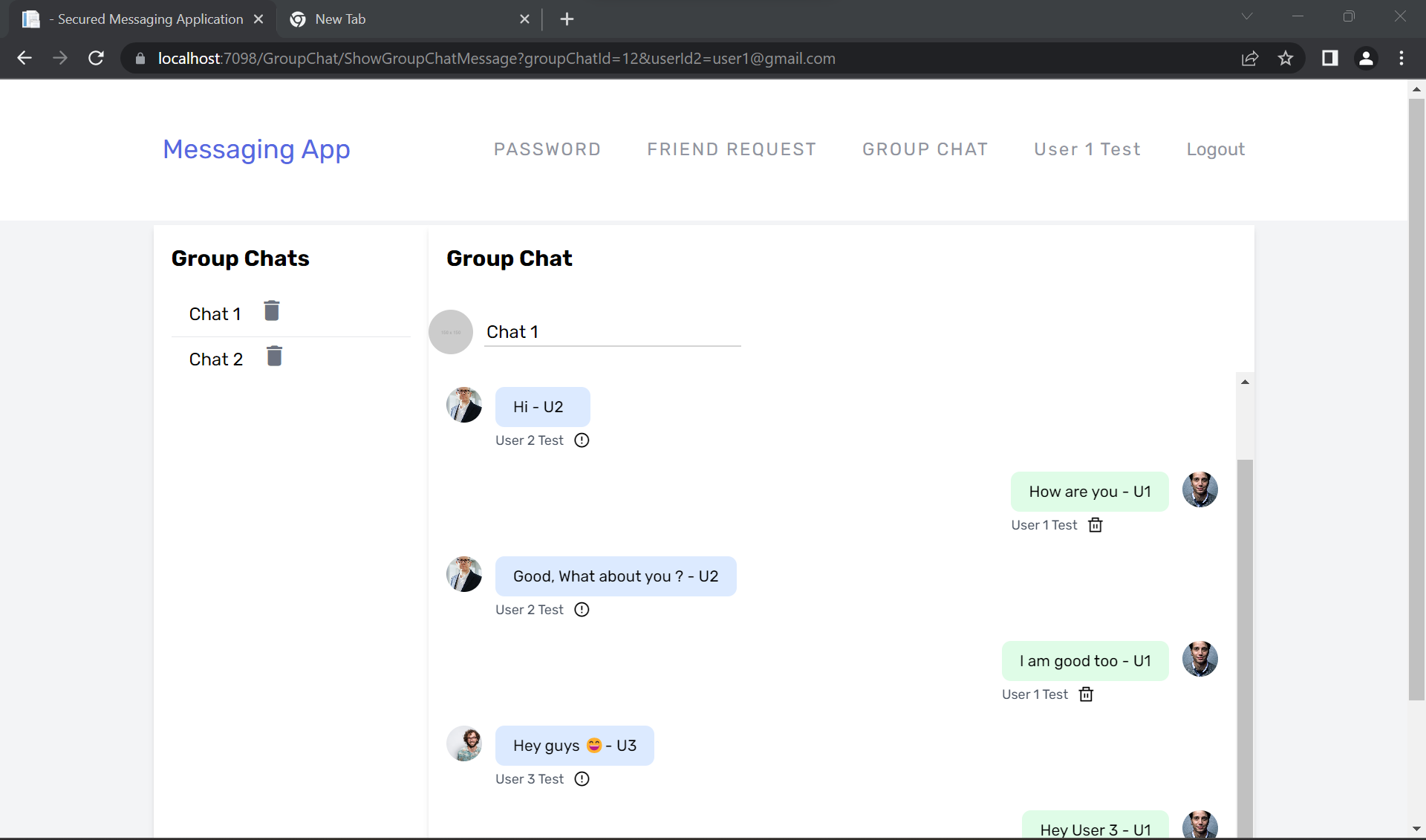


Figure : Group Chat Area

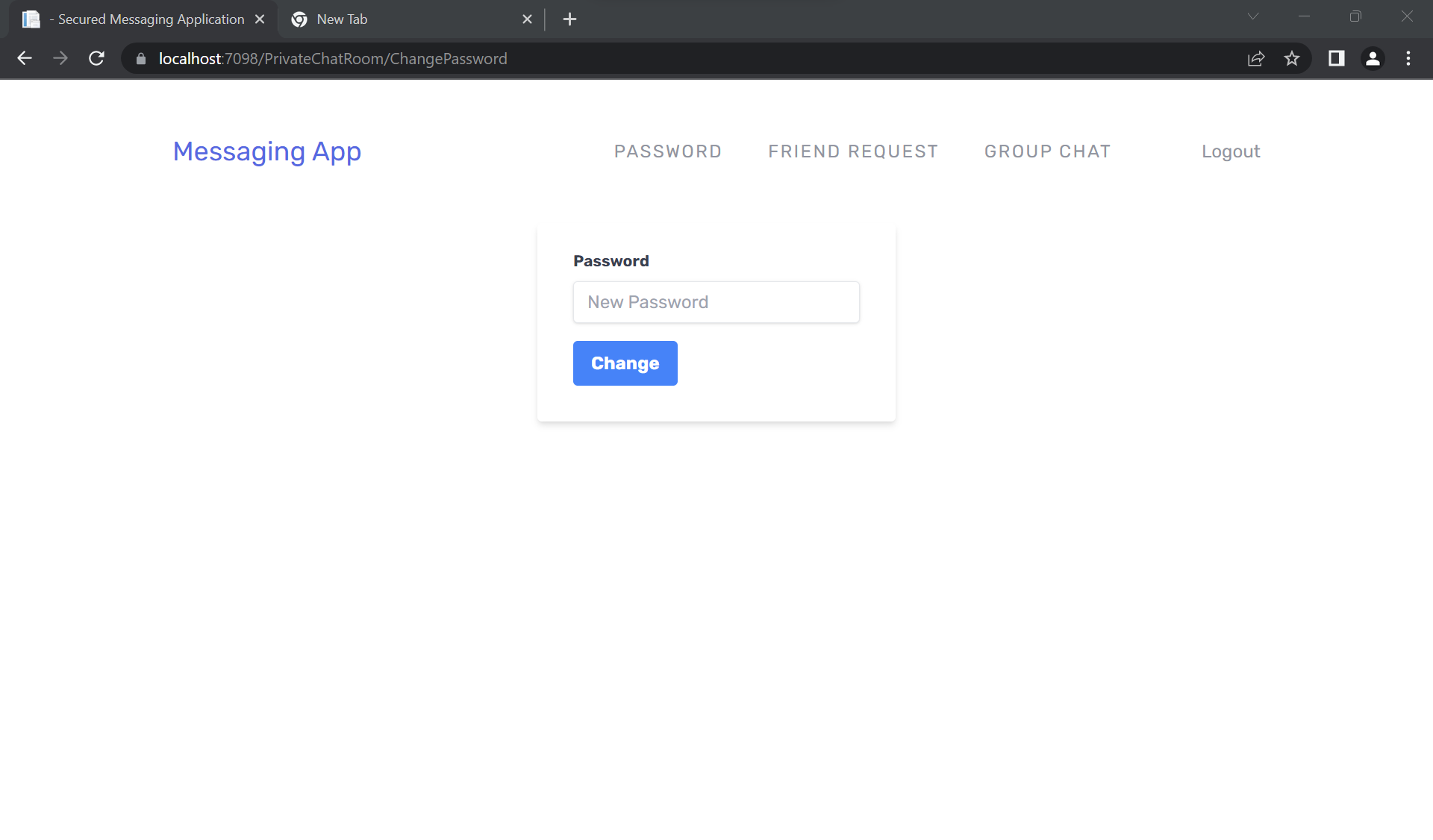
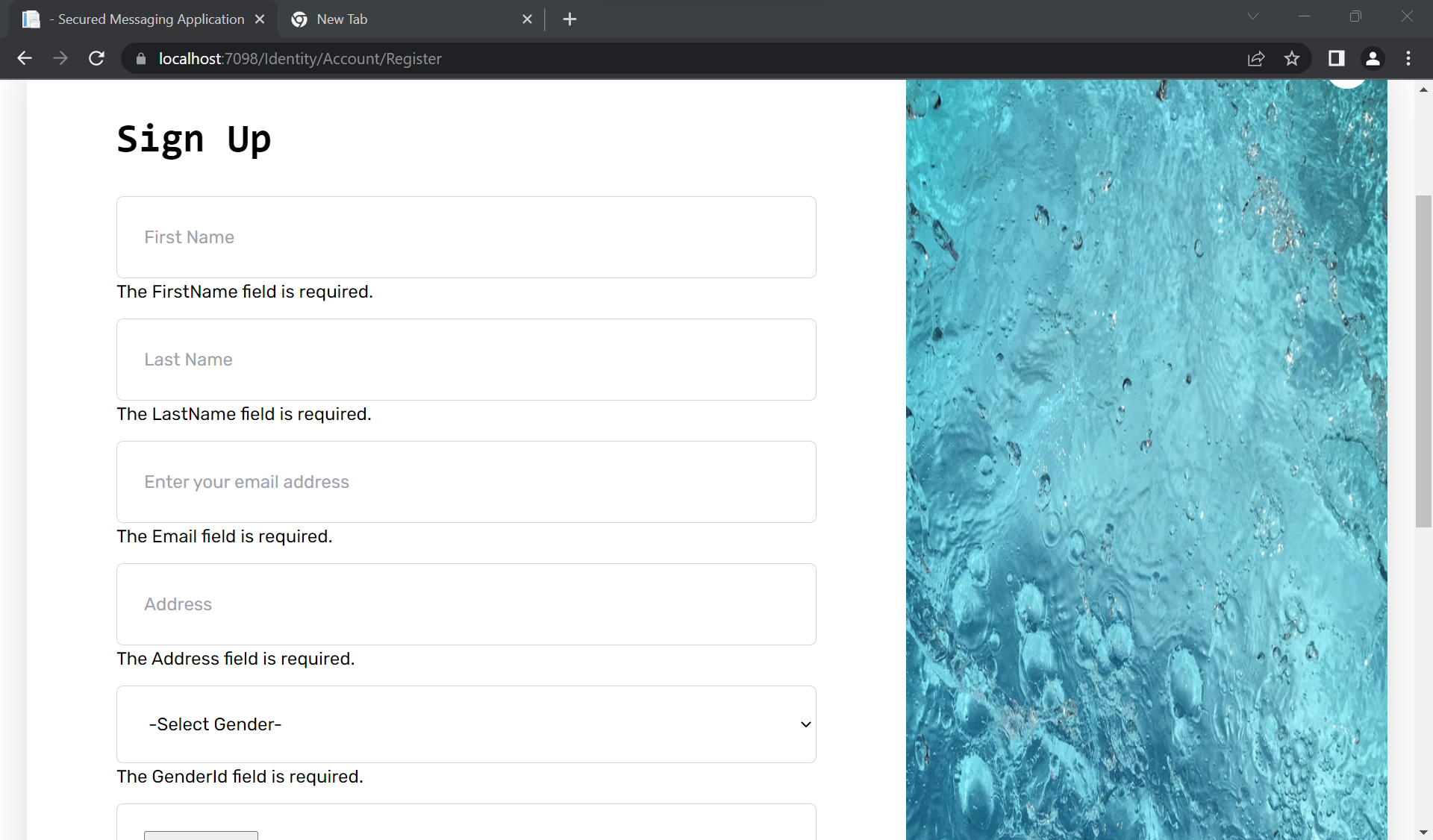
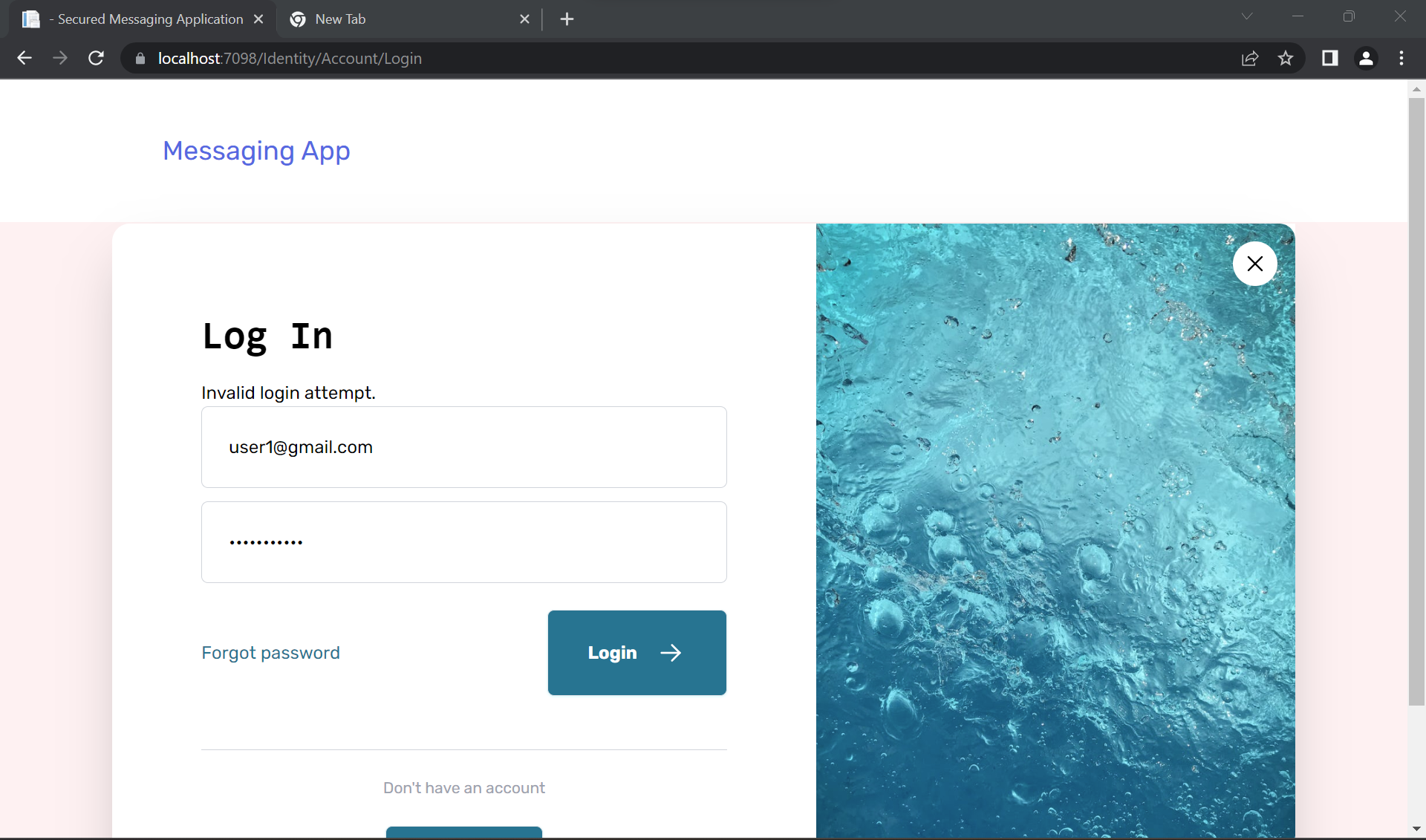


Figure : Password Reset Area

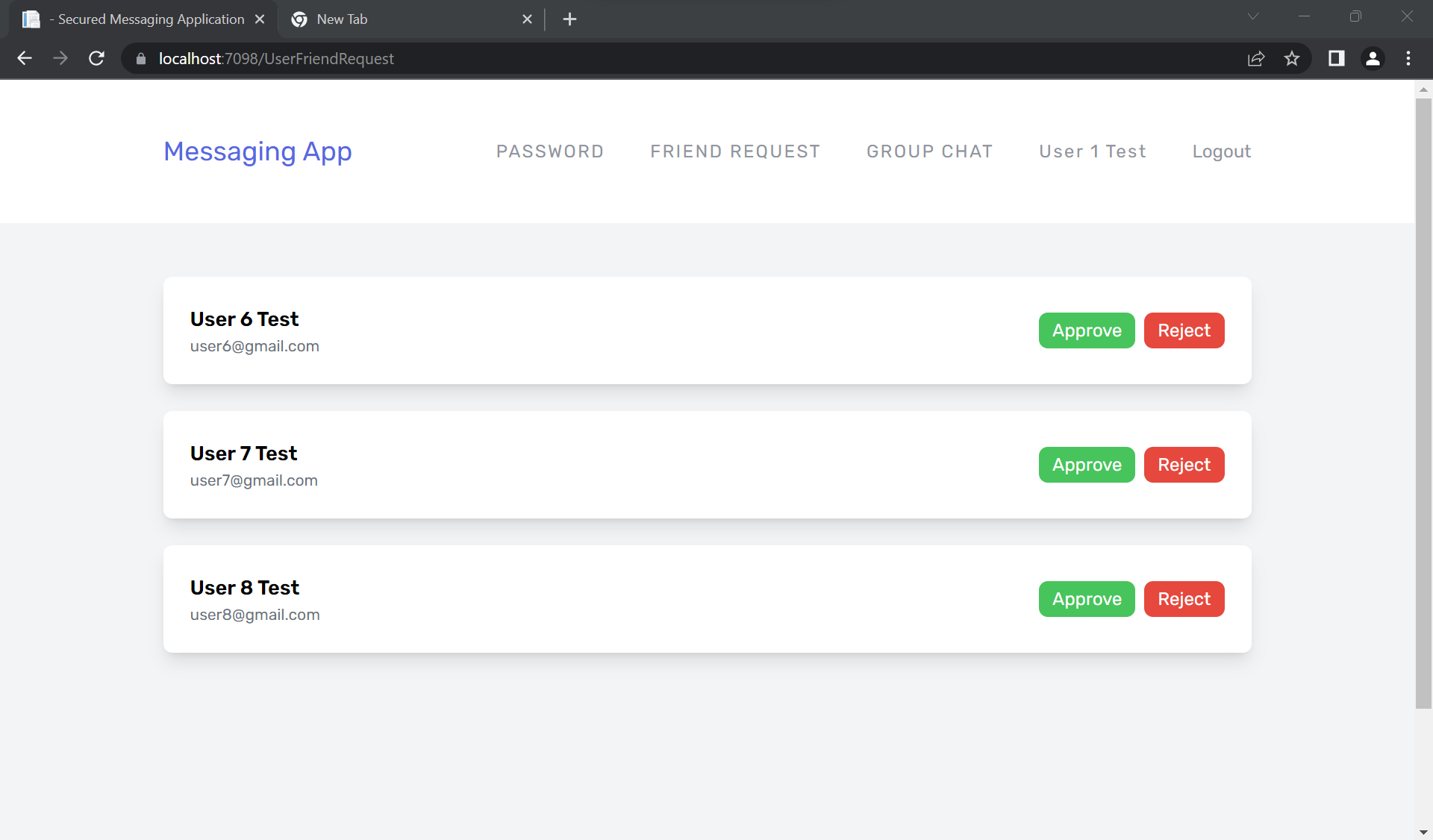
# **Implementation and Testing**



* If the user tries to Sign up without filling required field, then it will redirect back to sign up page and ask user to fill the required form.



* If user’s tries to login with invalid credentials, then it will return back to login page and display’s Invalid login attempt. It won’t let you know that if you have input wrong email or password because it may be insecure or unknown user can find that if your email or password is incorrect.



* User can View the friend request and Approve or Reject the friend request. If User approve the friend request, then they are connected each other and can chat/send private message. User Total number of friends count will also be updated. User can see if his/her friends are online or offline.

## **Implementation**

## **Testing**

Since, we are done with the development part, we shall work on checking if all the features are working as intended or not. For that purpose, we are testing all the test cases we had created earlier with the MoSCoW prioritization list.

Table : Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Code | Pre-Condition | Expected Result | Actual Result | Solution (if testing failed) | Status |
| Functional Requirement | | | | | |
| TC-000 |  |  |  |  |  |
| TC-001 |  |  |  |  |  |
| TC-002 |  |  |  |  |  |
| TC-003 |  |  |  |  |  |
| TC-004 |  |  |  |  |  |
| TC-005 |  |  |  |  |  |
| TC-006 |  |  |  |  |  |
| TC-007 |  |  |  |  |  |
| TC-008 |  |  |  |  |  |
| TC-009 |  |  |  |  |  |
| TC-010 |  |  |  |  |  |
| TC-011 |  |  |  |  |  |
| Non-Functional Requirements | | | | | |
| TC-105 |  | . |  |  |  |
| TC-106 |  |  |  |  |  |
|  |  |  |  |  |  |

# **Product Evaluation**

Now that we are done with the development and testing of the application, we must draw objective conclusions on it. Although the application delivers on everything that was promised, it is necessary to understand that not everything is perfect and there will always be flaws and there will always be a room for improvement. Having established that let us take a look into what the application has the potential to become and what are the things holding it back at its current state.

## **Scope**

* There is a market for it as established earlier since more people are caring about their privacy online.
* It doesn’t collect any personal data from the user which is a good feature that people care about.
* The ability to text, share files, and create group chat shows it has a potential to become a great collaboration tool.
* The ability to use sentiment analysis to filter harmful content has so much potential if made better.
* The freedom of being a stand-alone messaging application and not need a phone number or social media account unlike other options make it trustworthy since you are definitely not getting nagged by constant scam calls.

## **Limitations**

* There is no end-to-end encryption hence it is not completely secure since basic encryption might not be enough sometimes.
* It requires other users to use this platform in order to be secure and user base is going to be very limited at early stages and it will need to get a lot of traction before it becomes viable as a daily communication messaging application.
* There is no automated filtering system currently which can protect you from unwanted messages which can cause nuisances.
* It doesn’t have ability to make video and audio calls and it can be unappealing for people to switch over only for the sake of texting.
* The application only exists in the form of a web application which limits itself in terms of discoverability and ease of access. It also means that there is no cached data you can access when you happen to be offline. You need to be connected to the internet to do anything on this application.

# **Project Evaluation**

Now that we have made an objective evaluation, I am going to share my personal subjective evaluation and what I feel about the application that I have built and discuss what I could have done better.

## **Personal Reflection**

Working on this application has been both fun and stressful at the same time. Getting to employ everything I got to learn throughout my academic classes throughout the semesters and making something with real-life utility is so satisfying but at the same time countless hours spent into making a silly mistake, trying to debug it for hours and feeling like I am losing the last hair off my head is very stressful indeed. I would have liked to make something more out of this application but I was limited by the time and resources I had on top of my own personal limits. The guidance and help I have been getting from my supervisors, tutors, and friends have been very insightful in helping me make this messaging application. I got to learn so much about not only the technology behind making a secure messaging system but also the importance of data security, how harmful the impacts could be if my personal information were to be compromised.

If I had to be very honest, I am very much pleased, satisfied, humbled, and yet want to make something more out of this and I might as well just do that.

## **Future Work**

If I were to invest more hours into improving this application, my focus would be towards making the sentiment analysis automatic. The idea of automatic moderation is very cool and ideal for an application with an objective such as this. Another thing I would work towards if I were to work on this project is to package this into some form of smartphone application even if it is purely in the web application form and not a true smartphone app since it makes it more accessible to people. I would really like to work towards creating end-to-end encryption technology for this application further solidifying the secured status of the application. I would also like to make video calls and audio calls a thing for this app since it is a very integral part of a messaging application nowadays.

Overall, this is how I feel about working on this project and what improvement I would like to make to the application.

# **Conclusion**

We should now understand how important data privacy is for our safety. It is strongly advised that we do not share all of our personal information to any random place. It is essential to raise awareness to people who are unaware of such risks and make them understand how severe the consequences could be. It is also essential to adopt other safety measures than just a secure messaging application since your data could be compromised in many other daily applications that you use. Make sure to always practice safe internet browsing. As for choosing a secure messaging apps, there are plenty of them in the market. The most prominent one of them is Signal which is highly rated and widely used.

# **References**

Anonyome Labs, 2021. *How Bad is the Global Data Privacy Crisis?.* [Online]   
Available at: https://anonyome.com/2021/03/how-bad-is-the-global-data-privacy-crisis/#:~:text=Data%20privacy%20is%20an%20exploding,off%20being%20a%20comprehensive%20fix.  
[Accessed 15 April 2023].

Barry, B. & Tom, F. M., 2010. Instant Messaging: Standards, Protocols, Applications, and Research Directions. In: B. G. Kutais, ed. *Internet Policies and Issues.* s.l.:Nova Science Publisher.

Bernard, P., Plaisent, M., Hallman, S. & Dimodungo, M., 2021. The effect of privacy concerns, risk, control, and trust on individuals’ decisions to share personal information: A game theory-based approach. *Journal of Physics: Conference Series,* Issue 2090.

DEVART, 2021. *EF Core Introduction Tutorial: Definition, Architecture, Usage.* [Online]   
Available at: https://www.devart.com/what-is-entity-framework-core/  
[Accessed 13 March 2023].

Fitzgerald, A., 2022. *Tailwind CSS: What It Is, Why Use It & Examples.* [Online]   
Available at: https://blog.hubspot.com/website/what-is-tailwind-css  
[Accessed 13 March 2023].

Goodman, E., Kuniavsky, M. & Moed, A., 2012. Chapter 3 - Balancing Needs through Iterative Development. In: *Observing the User Experience (Second Edition).* s.l.:s.n., pp. 21-44.

Goswani, A., 2017. *What, Why and How About SignalR.* [Online]   
Available at: https://www.c-sharpcorner.com/UploadFile/abhijmk/what-why-and-how-about-signalr/  
[Accessed 13 March 2023].

Knight, S., 2022. *What Ever Happened to ICQ?.* [Online]   
Available at: https://www.techspot.com/article/1771-icq/  
[Accessed 12 March 2023].

Kröger, J. L., Miceli, M. & Müller, F., 2021. HOW DATA CAN BE USED AGAINST PEOPLE: A CLASSIFICATION OF PERSONAL DATA MISUSES. *Preprint.*

MacDonald, M., Freeman, A. & Szpuszta, M., 2010. Visual Studio. In: *Pro ASP.NET 4 in C#.* s.l.:s.n.

Microsoft, 2023. *Overview of ASP.NET Core MVC.* [Online]   
Available at: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-7.0  
[Accessed 13 March 2023].

Oca, B. M. d., 2020. *What happened to MSN Messenger?.* [Online]   
Available at: https://slidebean.com/story/what-happened-to-msn-messenger  
[Accessed 12 March 2023].

O'Reilly Media, Inc., 2009. *SMS Messaging Applications.* s.l.:O'Reilly Media, Inc..

PÉREZ, S. D., 2021. *WHAT IS MICROSOFT SQL SERVER AND WHAT IS IT FOR?.* [Online]   
Available at: https://intelequia.com/en/blog/post/what-is-microsoft-sql-server-and-what-is-it-for  
[Accessed 14 March 2023].

Quach, S. et al., 2022. Digital technologies: tensions in privacy and data. *Journal of the Academy of Marketing Science,* Volume 50, pp. 1299-1323.

Rouse, M., 2016. *Compatible Time Sharing System.* [Online]   
Available at: https://www.techopedia.com/definition/11282/compatible-time-sharing-system-ctss  
[Accessed 13 March 2023].

Schrittwieser, S., Kieseberd, P., Fruehwirt, P. & Weippl, E., 2015. Security and privacy of smartphone messaging applications. *International Journal of Pervasive Computing and Communications .*

Shanu, S., 2020. *Getting Started With Machine Learning .NET (ML.NET).* [Online]   
Available at: https://www.c-sharpcorner.com/article/getting-started-with-machine-learning-dotnet-ml-net/  
[Accessed 14 March 2023].

Solis, D. & Schrotenboer, C., 2018. Overview of C# Programming. In: *Illustrated C# 7.* s.l.:s.n., pp. 23-42.

Whiteley, A., Pollack, J. & Matous, P., 2021. The Origins of Agile and Iterative Methods. *Journal of Modern Project Management,* 8(3), pp. 220-29.