Tribhuvan University

Faculty of Humanities and Social Science

ChatMate – Chat Application

A Project Report on

"Chat Application"

Submitted to

Department of Computer Application

Academia International College

In Partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted By

Prabhat Gurung

(6-2-346-25-2020)

Pawon Shrestha

(2-346-24-2020)

2081/03/11

Under the Supervision of

Bishwas Ram Mathema

Tribhuvan University

Faculty of Humanities and Social Science

Academia International College

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by "**Prabhat Gurung**" and "**Pawon Shrestha**" entitled "**ChatMate**" in partial fulfillment of the requirements of the degree of Bachelor of Computer Application is recommended for the final evaluation.

…….

SIGNATURE

Bishwas Ram Mathema

SUPERVISOR

Department of Computer Application

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Faculty of Humanities and Social Science

Academia International College

LETTER OF APPROVAL

This is to certify that this project is prepared by "**Prabhat Gurung**" and "**Pawon Shrestha**" entitled "**ChatMate**" in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. **Mr. Bishwas Ram Mathema** **Supervisor** **Academia International College** **Gwarko, Lalitpur**	**……..** **Mr. Santosh Rijal** **Coordinator** **Academia International College** **Gwarko, Lalitpur**
…….. **Internal Examiner**	**……..** **External Examiner**

STUDENT'S DECLARATION

We hereby declare that we are the only author of this work and that no sources other than those listed have been used in this work.

…….	…….
Prabhat Gurung	Pawon Shrestha
Date:	Date:

ABSTRACT

ChatMate is a dynamic platform built for seamless real-time communication, allowing users to engage in both individual and group chats, addressing the demand for efficient messaging solutions. With a scalable infrastructure powered by Web Socket technology, it ensures fast message delivery. MongoDB is used to manage user data and messages, structured through Prisma ORM to optimize data retrieval and relationships. Prioritizing security, the app employs strong authentication methods like OAuth and JWT, alongside AES encryption to protect message content. Users can easily share multimedia, such as images, for a richer experience. The intuitive interface is designed with accessibility and responsiveness in mind, making it suitable for diverse audiences. Looking ahead, ChatMate aims to introduce video sharing, video calls, chatbots, and user analytics to enhance its functionality. Its goal is to improve user communication by combining instant messaging, strong security, and multimedia sharing in a user-centric environment.

Keywords: Chat Application, Real-Time Messaging, Encryption, ORM, Web Socket, MongoDB Database, Intuitive User Interface

ACKNOWLEDGEMENT

We are grateful to the department of computer application, Academia International College for providing us an opportunity to work on a major project as a part of our Sixth Semester project. We are delighted to express our deep sense of gratitude and indebtedness to our supervisor Mr. Bishwas Ram Mathema, for his invaluable guidance, encouragement and even monitoring to spare time despite his busy schedule for project's progress reviews.

Our special thanks goes to our colleagues and everyone who directly and indirectly extended their hands in making this project a success.

Prabhat Gurung

TU Exam Roll No: 6-2-346-25-2020

Pawon Shrestha

TU Exam Roll No: 2-346-24-2020

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheet
DFD	Data Flow Diagram
ERD	Entity-Relationship Diagram
HTML	Hyper Text Markup Language
ORM	Object Relational Mapper
TS	TypeScript

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Introduction

ChatMate is a communication tool designed to enhance and simplify user connections. It effectively addresses a range of communication needs, serving as an ideal platform for both personal one-on-one interactions and lively group discussions. With a focus on providing a seamless messaging experience, ChatMate allows users to send and receive messages instantly, ensuring conversations remain engaging and dynamic.

Central to ChatMate's functionality is its real-time messaging feature. This capability allows users to engage in smooth conversations, making it easy to stay in touch with friends, family, or colleagues. Immediate notifications for new messages help users maintain active dialogues without delays. The user experience is designed to feel intuitive, fostering spontaneous interactions that reflect the immediacy of modern communication.

ChatMate extends beyond simple text messaging by allowing users to share multimedia content with ease. The app supports various media formats, including photos, videos, and audio clips, enriching conversations with vivid content. Sharing moments as they occur is straightforward; users can quickly capture images or videos and send them with just a few taps. This integration of multimedia enhances interactions, enabling users to express themselves more creatively.

ChatMate extends beyond simple text messaging by allowing users to share multimedia content with ease. The app supports various media formats, including photos, videos, and audio clips, enriching conversations with vivid content. Sharing moments as they occur is straightforward; users can quickly capture images or videos and send them with just a few taps. This integration of multimedia enhances interactions, enabling users to express themselves more creatively.

ChatMate features an intuitive and user-friendly interface designed for easy navigation. Its clean and straightforward design ensures that users of all ages and technical abilities can engage with the app without difficulty. Features are logically organized, allowing users to quickly access the tools they need. This focus on accessibility creates a welcoming environment for all users, encouraging broader adoption of the app.

Understanding the importance of privacy in communication, ChatMate employs advanced security measures, including end-to-end encryption. This technology ensures that messages remain secure and can only be accessed by the intended recipients, protecting user's conversations from potential threats. By prioritizing user privacy, ChatMate fosters trust and confidence among its users, making them feel secure while engaging in personal and professional communications.

1.2 Problem Statement

In today's fast-paced digital world, effective communication is vital for maintaining personal and professional relationships. However, many current chat apps complicate this process with their complex interfaces and numerous features, making simple interactions challenging.

Issues with Current Chat Apps:

- Overwhelming menus and interfaces.
- Slow performance due to excessive features.
- Ineffective tools for group chat management.
- Difficult and delayed image sharing with potential quality issues.

ChatMate aims to transform the user experience by emphasizing simplicity and efficiency.

1.3 Objectives

a) To provide a chat application that is easy to use and efficient, focusing on essential features that enhance communication without unnecessary complexity.

b) To create a user interface that is intuitive and straightforward, allowing users to navigate effortlessly and perform tasks such as messaging, image sharing, and group management with ease.

c) To simplify communication processes, enhance user satisfaction, and facilitate smooth interaction among users.

d) To implement robust security measures to protect user data and ensure privacy during all interaction within the application.

e) To ensure compatibility across various device screen enabling smoother user experience due to responsive screen.

1.4 Scope and Limitation

1.4.1 Scopes:

ChatMate is tailored to meet a wide range of communication needs, whether users are engaged in private one-on-one chats or group discussions. Its real-time messaging feature ensures seamless conversation flow, with instant message delivery. This allows users to stay actively engaged in conversations without interruption, enhancing communication.

The major scope of **ChatMate** can be listed as follows:

- It provides instant, seamless communication for personal, and group chats in real time.
- It easily shares images in various formats to enrich conversations.
- It efficiently create and manage group chats for both small and large groups.
- The intuitive, clean design ensures easy navigation for users of all ages.
- AES encryption guarantees secure and private conversations for all users.

1.4.2 Limitations:

The limitations are listed below:

- Real-time communication is disrupted without a reliable internet connection, reducing the app's effectiveness in areas with poor or no coverage.
- It does not allow users to send or receive messages while offline, making it unusable during internet outages or when there's no connection.
- Group management is limited to creation of group, adding members, and deletion.
- Large files like high-resolution images may exceed the app's limits, causing inconvenience for users need to share sizeable image.
- Users seeking more personalization may find the app lacking, restricting their ability to tailor the app to their preference.

1.5 Development Methodology

1.5.1 Agile Methodology:

Adopting Agile methodology for ChatMate's development brings significant benefits that align with the project's goals, enhancing flexibility and adaptability to quickly adjust the project scope based on evolving user needs and market dynamics. Its iterative framework allows for continuous improvement, facilitating regular updates to the application with each sprint, followed by reviews that assess achievements and areas for growth. This

approach emphasizes frequent value delivery, enabling users to access new features and enhancements regularly, which greatly improves their overall experience.

Agile also encourages stakeholder engagement through consistent feedback, ensuring the development team remains aligned with user expectations and can prioritize key features. Its collaborative nature enhances communication among cross-functional teams, resulting in better problem-solving and a more cohesive product. By segmenting the project into manageable parts, Agile reduces the risk of failure and allows for early detection of issues through ongoing testing and feedback.

Furthermore, Agile's commitment to transparency provides all team members with a clear view of the project's progress, fostering accountability and trust within the group. Daily stand-up meetings and regular sprint retrospectives promote open dialogue and continuous learning, empowering team members to share ideas and take ownership of their responsibilities, which increases engagement and motivation. The adaptive planning process enables the team to incorporate new insights and emerging technologies without being limited by rigid initial plans, ensuring ChatMate remains relevant and innovative in response to user feedback and market competition.

Ultimately, Agile’s focus on user satisfaction ensures that ChatMate effectively meets user needs, leading to higher adoption rates and increased loyalty, while fostering a more user-centered, collaborative, and sustainable development process.

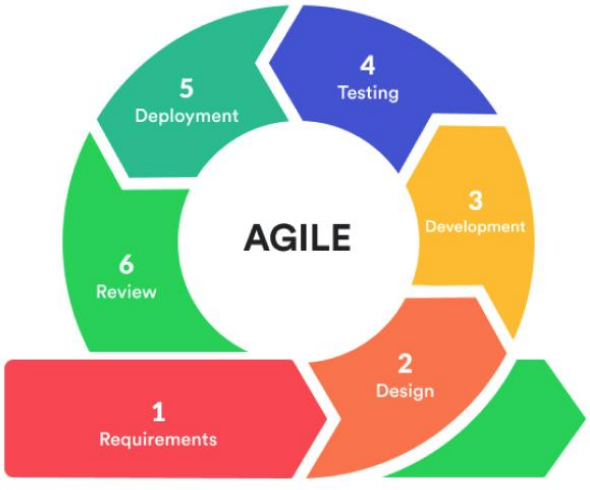

Figure 1.1: Agile Methodology

i. Concept and Initiation:

The Agile Methodology starts with the concept and initiation phase, where the project team identifies user needs and defines the project scope. This foundational stage is critical as it ensures that all team members understand the project's goals and objectives. By gathering insights from stakeholders, the team can clearly specify the essential features and functionalities required, aligning their work with user expectations.

ii. Planning

After the initiation phase, the planning stage focuses on creating a product backlog, which is a prioritized list of features, improvements, and fixes, commonly referred to as user stories. The team engages in sprint planning to establish the duration of the sprint-typically lasting one to four weeks and selects specific user stories to address during this time. This structured approach ensures that the team has clear objectives and direction, enabling effective and focused development efforts.

iii. Execution

During the execution phase, the actual development work occurs. The team collaborates on the chosen user stories while conducting daily stand-up meetings to share updates on progress, discuss any challenges, and outline plans for the day ahead. This consistent communication promotes alignment and accountability within the team, with the ultimate goal of delivering a potentially shippable product increment by the end of the sprint, ensuring that the product continues to evolve based on user needs.

iv. Review

At the conclusion of each sprint, the team holds a sprint review to present the completed work to stakeholders and gather feedback. This step is vital for validating the product and confirming that it meets user expectations. The feedback collected during this review provides the team with valuable insights into their effectiveness in addressing user needs, guiding future iterations and enhancements.

v. Retrospective

Following the sprint review, the team conducts a sprint retrospective. This meeting provides an opportunity for team members to reflect on the sprint, discussing successes, areas for improvement, and potential process optimizations. The retrospective is a crucial part of the Agile approach, fostering a culture of continuous improvement and

ensuring that the lessons learned are applied in upcoming sprints to enhance overall team performance.

vi. Iteration

The iteration phase involves refining the product backlog based on the feedback received during the review and retrospective. New user stories can be introduced, existing ones may be reprioritized, and adjustments can be made as needed. The team then begins a new sprint, repeating the cycle of planning, execution, review, and retrospective, allowing the product to develop in response to user needs and market trends.

vii. Release

Once the product achieves a satisfactory level of quality and completeness, the team prepares for the final release. This phase involves conducting final testing, creating necessary documentation, and deploying the product for users. A successful release ensures that the product is ready to provide value to users and meets their expectations effectively.

viii. Feedback and Maintenance

After the release, the team continues to collect user feedback to guide ongoing development efforts. This feedback loop is essential for adapting the product to evolving user requirements and market conditions. Ongoing maintenance includes addressing any bugs, implementing new features, and ensuring that the product remains relevant and effective over time to maximize user satisfaction.

1.6 Report Organization

a) Chapter 1:

This chapter puts a brief emphasis on background, overview, problem statement, objectives, scope and limitations of the project.

b) Chapter 2:

This chapter defines and describes background study and literature review done on existing system along with their merits and demerits.

c) Chapter 3:

This chapter focuses on the different requirements of the system. It presents the system analysis and design including functional, non-functional, feasibility analysis, entity relational diagram, data flow diagram, design of the system architecture, database schema, interface design and algorithm details.

d) Chapter 4:

This chapter focuses on the tools used in the system development, implementing details and result of test performed.

e) Chapter 5:

This chapter marks the end of the document highlighting brief summary of lesson learnt, outcomes, and conclusion of the project explaining what have been done and what further improvements could be done.

f) Chapter 6:

This chapter contains references taken from various sources.

g) Chapter 7:

This chapter contains Appendix which includes screenshots.

Chapter 2: Background Study and Literature Review

2.1 Background Study

A comprehensive study of existing chat applications needed to be done in order to ensure the successful development of our chat application (ChatMate). This study aims to offer valuable insights into the advantages, disadvantages, and distinctive aspects of current solution, allowing us to determine the key requirements and potential areas for improvement.

Conduct a thorough analysis of widely-used chat applications such as Telegram, WhatsApp, Viber, Facebook Messenger, to identify their strengths and weaknesses. This comprehensive examination will guide the design and development of our chat application, enabling us to create a platform that will excel in user experience (UX), security, customization, and scalability.

Furthermore, this will enable us to address current market deficiencies and integrate innovative features. By analyzing the success and shortcomings of existing applications, we can develop a more robust and competitive product that aligns with the evolving needs of users. This is essential for ensuring that ChatMate not only meets but exceeds industry standards and user expectations

2.2 Literature Review

The rapid evolution of technology has significantly impacted the development and security of web based live chat application. Various studies have explored different dimensions of these applications, from advanced encryption techniques to end-to-end security, simplified chat room systems, and real-time communication tools with enhanced features.

One study focuses on enhancing the security of live chat applications through the integration of Advanced Encryption Standard (AES) and Rivest Shamir Adleman (RSA) algorithms. The AES algorithm, a symmetric block cipher, efficiently encrypts and decrypts messages, while the RSA algorithm, an asymmetric block cipher, secures data by encrypting it in blocks before conversion to cipher text. Implementing these cryptographic methods ensures that messages remain secure during transmission and storage. The application, developed using PHP, CodeIgniter, and MySQL, encrypts the messages and PNG files automatically, providing robust end-to-end security for user communications [1].

Another significant contribution to the field is the development of chat applications that ensure end-to-end security. These applications enable users to exchange text messages, images, and files securely. A key focus of the study is to identify and implement essential security requirements, such as secure session management and secure storage to protect user data comprehensively. The proposed solution is compared with other popular chat applications, highlighting its effectiveness in providing secure communication on Android platforms [2].

The evolution of chat room systems addresses the need for reliable and recorded online communication within organizations. A study on a simple chat room system implemented using HTML, PHP, CSS, JavaScript, and AJAX, demonstrates advancements in bidirectional communication between clients and servers. This system facilitates seamless user interaction and records chat sessions for future references. While it does not incorporate advanced encryption methods, it provides a foundation to further enhancements in secure communication [3].

The ChatterBox project represents a significant advancement in real-time chat applications. Developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js), ChatterBox offers a multi-platform, real-time communication tool that includes features like audio and video calls, screen-sharing and resource exchange. The use of modern technologies ensures that the application is secure, extensible, and easily manageable. Compared to traditional systems, ChatterBox provides a higher level of security and functionality, making it a reliable choice for modern communication needs [4].

A study focusing on Web Real-Time Communication (WebRTC) technology highlights, the implementation of video and chat applications using a mesh network topology. WebRTC enables real-time media and data exchange between browsers, facilitated by a signaling process. The study demonstrates the use of Socket.io, Node.js, and Express.js, to create a signaling channel for browser-to-browser communication. The implementation using Chrome and Firefox ensures efficient peer-to-peer communication, allowing for chat, video conferencing, and recording. This approach leverages modern web technologies to establish a robust communication channel in real-time applications [5].

An enhanced chat application introduces innovation features to address specific communication needs, particularly in professional and educational settings. This application integrates a paint-like editor within the chat window, allowing users to draw

and send diagrams. Such functionality is valuable in scenarios where visual aids are essential for effective communication. Additional features such as predictive texting, customizable themes, and voice-to-smiley conversion further enhance the user experience, making the application versatile and user-friendly [6].

Chapter 3: System Analysis and Design

3.1 System Analysis

The development of ChatMate encompasses the design and implementation of a software platform aimed at enabling project management, collaboration, and communication among users. To ensure the effective delivery of such a complex system, it is vital to conduct a comprehensive analysis of both the functional and non-functional requirements, user needs, and technical architecture. This system analysis presents the fundamental components that underpin ChatMate, including the development approach, core system functionalities, user interactions, and considerations for future scalability.

3.1.1 Requirement Analysis

Requirement analysis is to be done as it is a critical part for the project development life cycle. Requirement analysis are of two types:

- Functional Requirements
- Non-Functional Requirements.

i. Functional Requirements

This section provides the requirement overview of the system. Different functional requirement of the system have been listed below:

a) For User Module:

- User should be able to create accounts securely and login using credentials or alternative methods like social media accounts.
- User profile be able to view profile information (optional).
- User should be able to change name and profile picture.
- User should be able to logout from the system.

b) For Chat Module:

- Provide users the ability to send and receive messages in real-time.
- Image sharing capabilities
- Allow users to create, add, and delete group chats with multiple participants.
- Ensure data encryption for messages, secure storage of the encrypted data.

Use-Case Diagram

A use-case diagram is a graphical representation used in software engineering and system analysis to depict the functional requirements of a system from the perspective of its users or external actors. It provides a high-level overview of how users interact with a system and the different ways the system responds to those interactions.

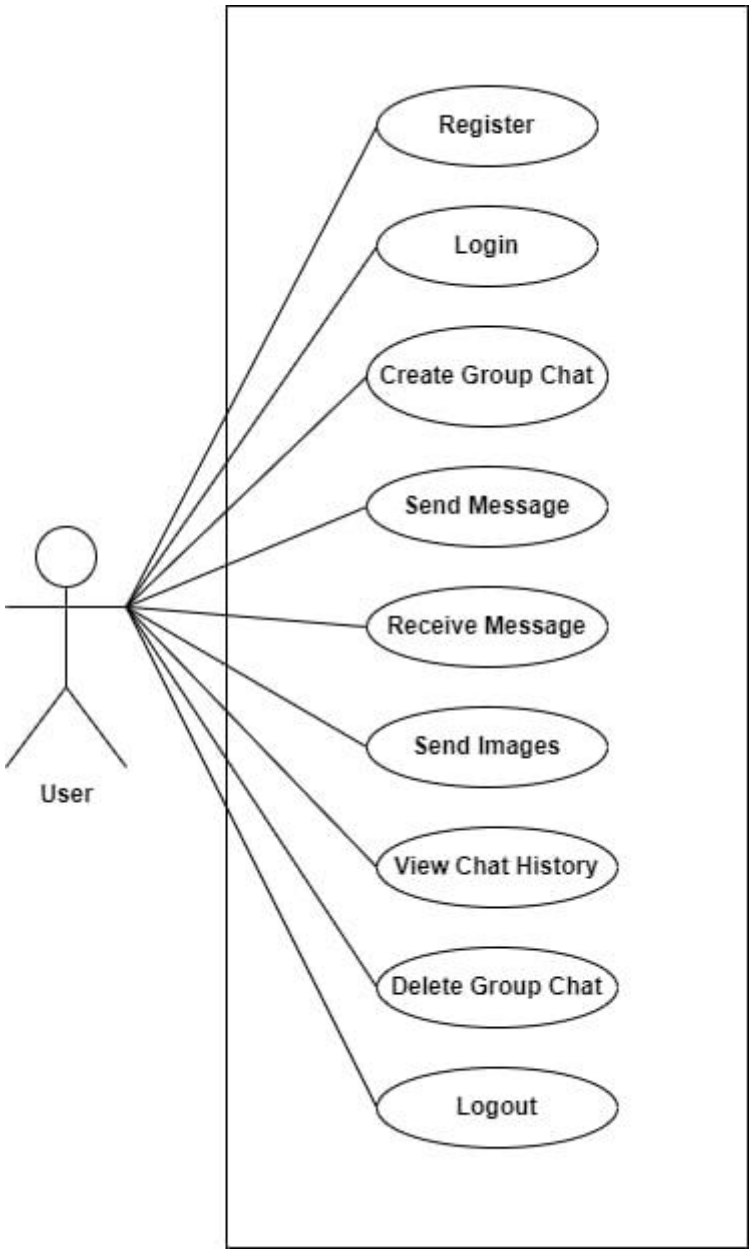

Figure 3.1: Use Case Diagram

ii. Non-Functional Requirements:

Different non-functional requirement of the system have been listed below:

- **Security:** Implement robust security measure to protect user data and communications, including encryption of message in transit and at storage.
- **Performance:** Ensure that the application responds quickly to user interactions, with minimal latency for sending messages, loading conversations, and sharing media.
- **Reliability:** Maintain high availability so that users can access the application reliably with minimal downtime or service disruptions.
- **Scalability:** Application should be able to handle increasing numbers of users and messages without compromising performance.
- **Usability:** The system ensure intuitive user-interface, making it easy for users to navigate. It also provides responsive design for seamless experience across all device screens (Desktop, Mobile, Tablet etc.).

3.1.2 Feasibility Analysis

i. Technical Feasibility

The system is technically feasible as the requirement for the development of the system is available. The necessary hardware and software required for the development and implementation of the system is also available. The programming language and technologies for the project is also available for free.

ii. Operational Feasibility

The system is operationally practical with a fundamental understanding of computers and internet access. User are able to easily access the system as it is user-friendly with good and intuitive User-Interface (UI). This simplicity and ease of use facilitate successful implementation.

iii. Economic Feasibility

The system was developed to be economically feasible and cost effective. All of the tools and resources used to develop the system are open source and free. The system does not necessitate the use of expensive hardware or software. All necessary

project resources are freely available, eliminating any additional financial obligations.

iv. Schedule Feasibility

The system is completed within the scheduled time and does not exceeds the scheduled time. The backend section took most of the time compared to the frontend section in the project development.

- **Gantt Chart**

Gantt chart is a graphical depiction of a project schedule. It is widely used in project management as it helps to display activities at different phases of time. The Gantt chart in our project was created using teamgantt as shown in the figure below.

Figure 3.2: Gantt chart

3.1.3 Object Modelling: Object & Class Diagram

A class diagram is a visual representation of the relationships and dependencies between classes in the Unified Modeling Language (UML). In this setting, a class outlines the methods and variables of an object, which is a distinct entity in a program or the code unit representing it. Class diagrams are valuable tools in all types of object-oriented programming (OOP).

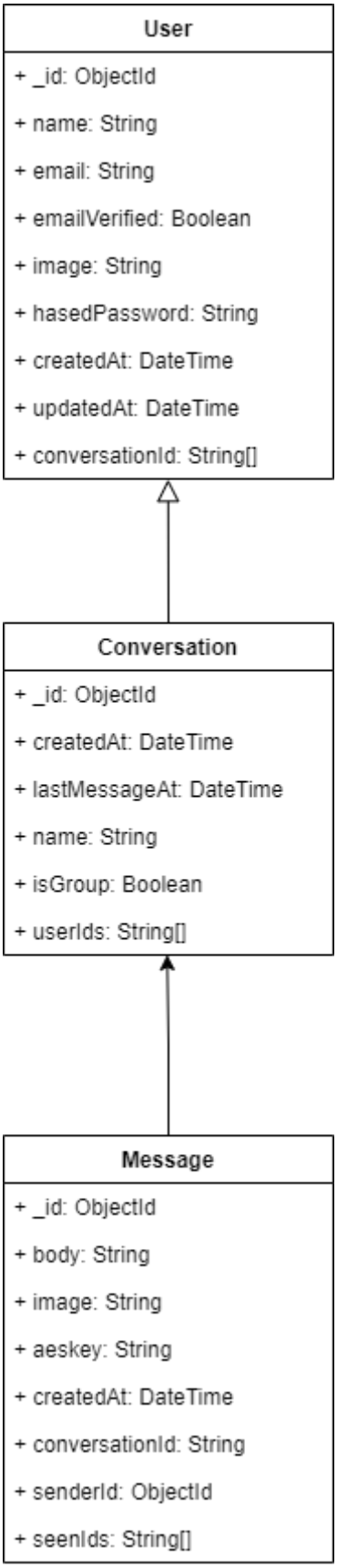

Figure 3.3: Object & Class Diagram

3.1.4 Dynamic Modelling: State & Sequence Diagram

Sequence diagrams for ChatMate are a dynamic modeling tool that visually depict the flow of messages between users and system components during communication. They illustrate the interactions between various modules, such as messaging, file sharing, and encryption, over time. By providing a visual representation of the system's behavior during specific user actions, these diagrams demonstrate how messages are encrypted, transmitted, and received in real-time, helping to clarify the interactions between components within the chat application.

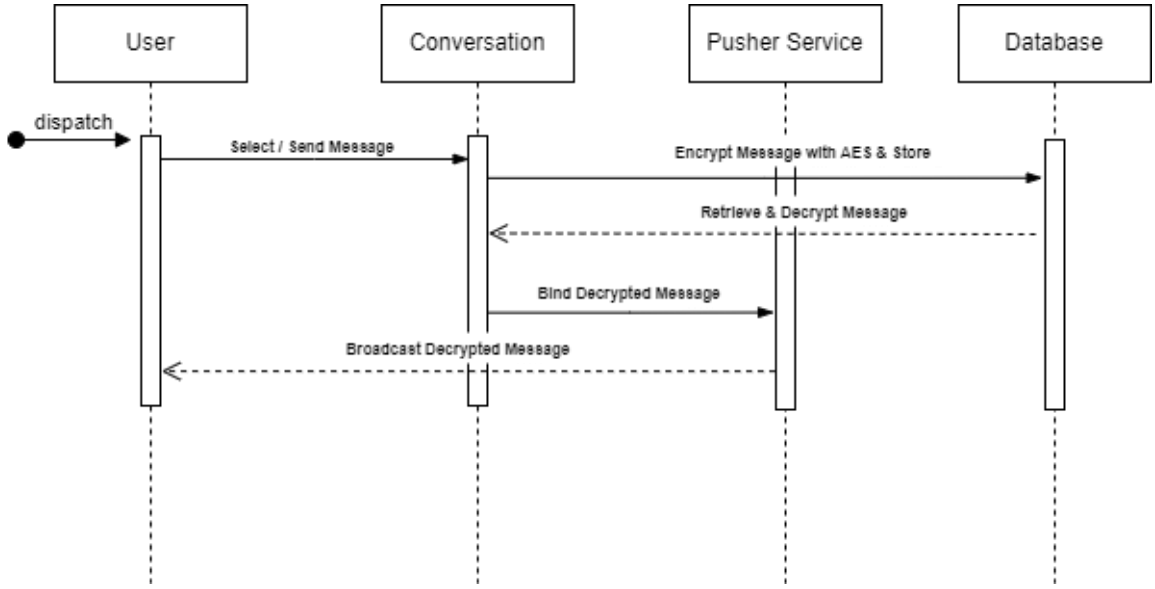

Figure 3.4: State & Sequence Diagram

3.1.5 Process Modelling: Activity Diagram

Activity diagrams serve as blueprints for processes, illustrating the flow of actions and decisions within a system from start to finish. They outline each step, including decision points and alternative paths, providing a clear understanding of how a process unfolds.

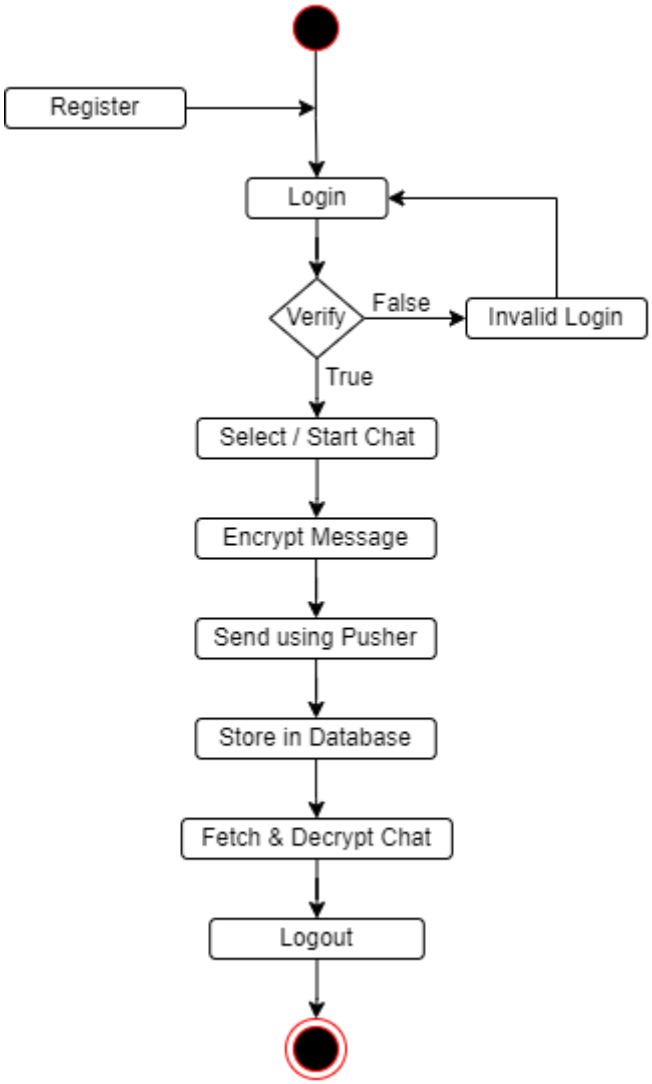

Figure 3.5: Activity Diagram

The Activity Diagram represents the flow of actions in ChatMate application:

- **Register:** A new user starts by registering an account in the system.
- **User Login:** The existing user logs in using their credentials or google provider.
- **Verify:** The system checks if the login credentials are valid.
    - If valid, then the process continues
    - If invalid, the user is shown an error toast message "Invalid Credential" message.
- **Select / Start Chat:** Once logged in, users can select an existing chat or start a new one.
- **Encrypt Message:** The message composed by the user is encrypted using AES encryption algorithm for secure transmission.

- **Send using Pusher:** The encrypted message is send over the network through pusher.
- **Store in Database:** The encrypted message is then saved in the database.
- **Fetch & Decrypt Chat:** The message is then fetched from the database and decrypted for the user.
- **Logout:** The user can then logout after they have finished sending messages.

3.2 System Design

System Design is essential in developing a chat application website to ensure it meets both functional and performance requirements while delivering a smooth user experience. It involves defining the system's elements, including its architecture, modules, components, the interfaces between those components, and the flow of data through the system. A key aspect of software design is the analysis of software requirements.

3.2.1 Refinement of Classes and Object

A key element of software design is the software requirement analysis. In this project, a single database is used to store the information of users who have registered their accounts within the system. The diagram below represents the Object Model with three key classes: User, Conversation, and Message. Each class include attributes that defines the structure in the application, which corresponds to how entities are represented in the database.

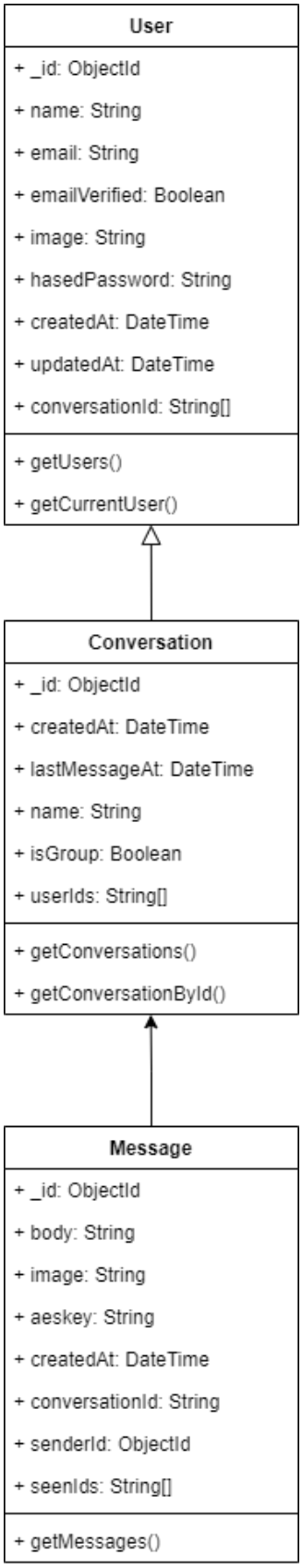

Figure 3.6: Refinement of Classes and Object

3.2.2 Component Diagram

A component diagram divides the system under development into distinct high-level functions. Each component is dedicated to a specific purpose within the overall system and interacts with other components only when necessary.

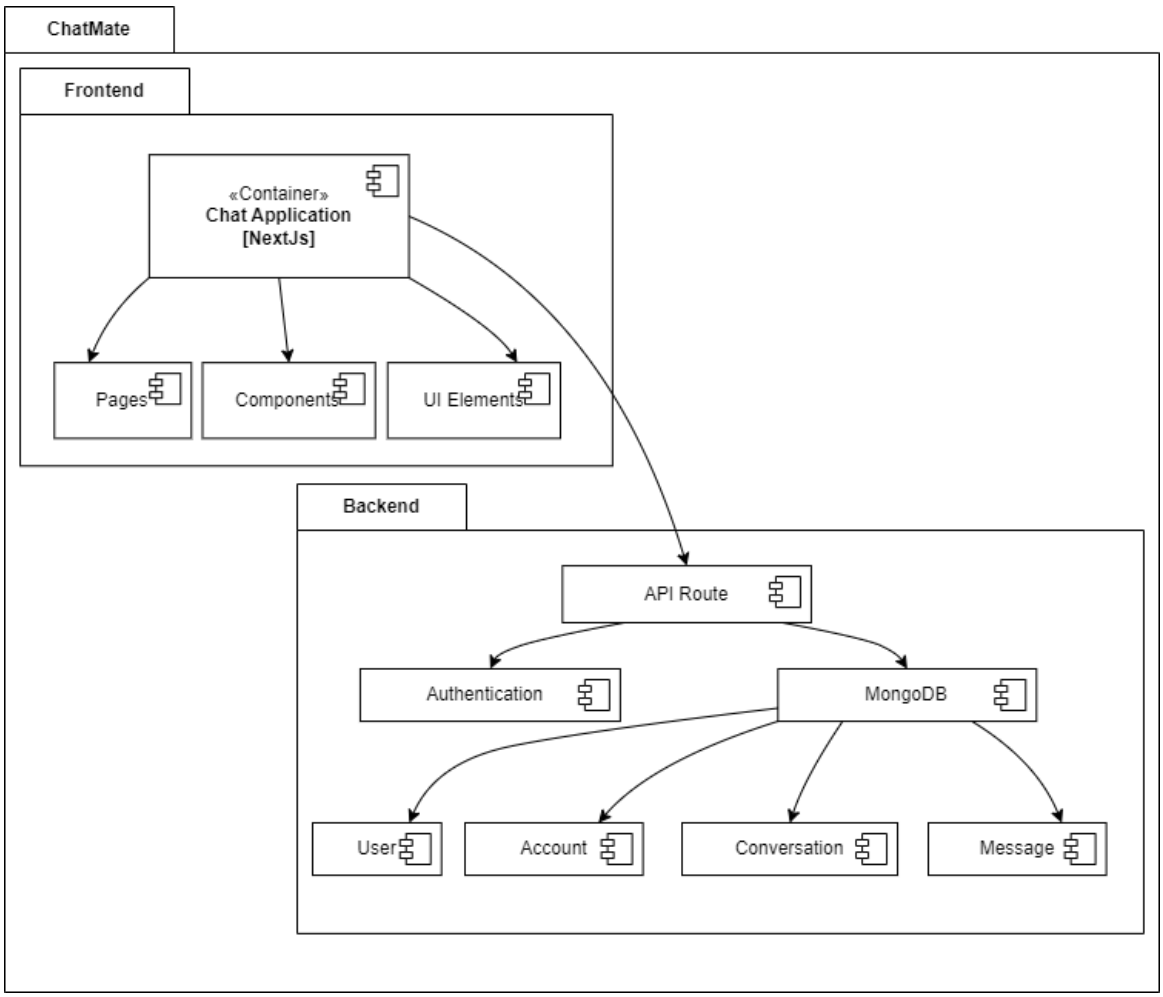

Figure 3.7: Component Diagram

The Frontend contains different elements:

- **Pages:** These refer to the individual pages in the Next.js application. Each page can have its own layout and functionality, such as login, chat room, user profile, etc.
- **Components:** These are reusable React components used across different pages to provide functionality. Examples include chat windows, input fields, and buttons.
- **UI Elements:** These are likely the basic building blocks of the user interface such as typography, icons, and other design components that make up the UI.

The Backend is responsible for managing server-side operations and database interactions.

- **API Routes:** These are the backend endpoints in Next.js that handle server-side logic. They process requests from the frontend (such as sending messages or authenticating users) and communicate with the database.
- **Authentication:** Handles user login, signup, and validation processes. It connects with both the User and Account models to verify credentials and manage user sessions.
- **MongoDB:** This is the database used to store all application data.
    - **User:** Stores user-related information such as profile details, email, and authentication tokens.
    - **Account:** Handles information related to user accounts including linked OAuth providers (Google).
    - **Conversation:** Stores the data regarding individual or group conversations, such as participants and timestamps.
    - **Message:** Contains actual encrypted chat messages sent by users along with metadata such as sender information and timestamps.

The API route interacts with Authentication and MongoDB to handle all server-side requests. MongoDB stores and retrieves data for users, accounts, conversations, and messages, ensuring persistent storage.

3.2.3 Deployment Diagram

Deployment diagrams are a type of structural diagram used to model the physical aspects of an object-oriented system. They are commonly used to represent the static deployment view of a system, showing the topology of the hardware.

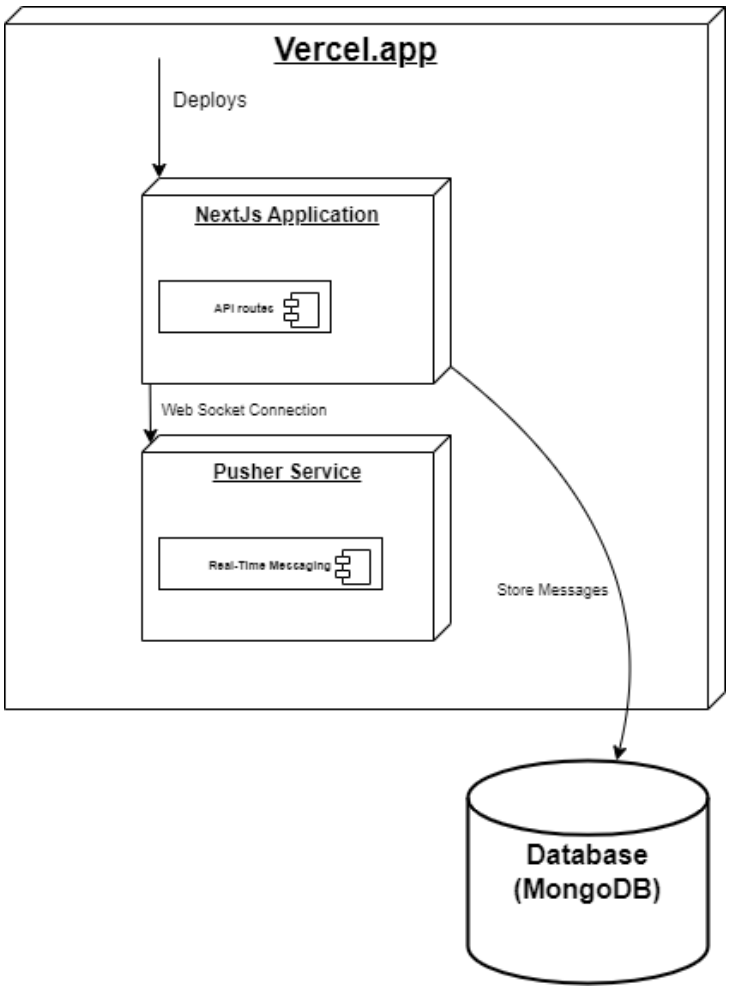

Figure 3.8: Deployment Diagram

The figure above shows the deployment diagram for ChatMate. In this setup, ChatMate is hosted on vercel platform, a cloud platform optimized for frontend framework like NextJs. The API routes are responsible for handling user requests, sending/receiving messages, and interacting with MongoDB database. The pusher service manages Real-Time Messaging between users, ensuring messages are sent and received in real-time via a WebSocket connection. MongoDB is used as the database to store messages and any other relevant data, such as user information and chat history.

3.3 Algorithm Details

i. Description of Algorithm:

AES (Advanced Encryption Standard)

AES (Advanced Encryption Standard) is a widely used symmetric encryption algorithm known for its efficiency and security and is commonly used for encrypting data.

Step 1. Generate AES Key:

- Choose key size 128 bits, 192 bits, or 256 bits and generate the key

Step 2. Encrypt Message with AES.

- Encrypt the message with AES using the AES Key generated
- Transmit the AES key

Step 3. Decrypt Message with AES.

- Decrypt the message with the transmitted AES Key.

Comparison of AES with other Counterparts:

There are other encryption algorithm which are commonly used in cryptography such as RC4, DES (Data Encryption Standard) and RSA (Rivest-Shamir-Adleman) etc. The comparison of AES with these algorithms are presented in the table below:

	AES	**DES**	**RSA**	**RC4**
Type	Block Cipher	Block Cipher	Public-Key Cipher	Stream Cipher
Key-Length	128, 192, 256 bits	56 bits	1024, 2048, 4096 bits	40-2048 bits
Speed	Fast	Moderate	Slow	Very Fast
Security	High (widely considered secure)	Moderate (considered insecure due to small key size)	High (but computationally expensive)	Vulnerable to known attacks (WEP crack)
Encryption Mode	Symmetric	Symmetric	Asymmetric	Symmetric
Best Use Case	Highly secure applications	Legacy systems	Secure communication, digital signatures	Real-time applications with limited resources

Advantages of AES Algorithm for Chat Application:

- **Strong Security:** AES is considered highly secure and has been extensively tested by the cryptographic community. It is resistant to all known practical cryptographic attacks. It offers encryption strengths of 128, 192, or 256 bits, providing a robust level of protection for sensitive messages and data in a chat application.
- **Efficiency and Performance:** AES is known for its speed and efficiency in both hardware and software implementations. It encrypts and decrypts data quickly, making it ideal for real-time messaging in chat applications where performance is crucial.
- **Low Resource Consumption:** AES is designed to be lightweight, making it suitable for devices with limited processing power, such as mobile phones. This is important for chat applications that run on diverse platforms, from powerful servers to smartphones. It consumes minimal memory and CPU resources, ensuring efficient operation on both client and server sides of the chat application.

Drawbacks of AES Algorithm:

- **Symmetric Key Management:** AES is a symmetric encryption algorithm, meaning the same key is used for both encryption and decryption. Without a secure key exchange mechanism (like Diffie-Hellman or RSA), the system is vulnerable during the key distribution phase. If the encryption key is intercepted or improperly managed, the security of the chat application is compromised.
- **Vulnerability to Key Guessing Attacks:** If a weak key is chosen, AES can be vulnerable to brute-force attacks. AES itself is resistant to cryptanalysis, but weak key management or insufficient randomness can undermine its security.

Justification for using AES Algorithm:

Using the AES (Advanced Encryption Standard) algorithm in a chat application like ChatMate is a strong choice due to several factors that align well with both performance and security requirements.

- **Strong, Well-Tested Security:** AES is widely regarded as one of the most secure encryption algorithms available, having been thoroughly tested and adopted globally by governments, financial institutions, and tech companies.
- **Efficiency and Performance:** Chat applications require real-time message encryption and decryption without noticeable delays to maintain a smooth user experience. AES is known for its efficiency in both hardware and software, allowing it to process data rapidly with minimal impact on performance.

Chapter 4: Implementation and Testing

4.1 Implementation

4.1.1 Tools Used (CASE tools, Programming languages, Database platforms)

Tools used for the development of this project are as follows:

Framework

- **NextJs**

  Next.js is a popular React framework used for building server-rendered applications with enhanced performance and flexibility. Next.js simplifies the development process by offering built-in routing, optimized image handling, and API routes, making it a powerful choice for developing full-stack React applications. Its hybrid nature allows developers to choose between static or dynamic rendering, depending on the project's needs.

- **Tailwind CSS**

  Tailwind CSS is a utility-first CSS framework that allows developers to create custom designs quickly and responsively by composing classes directly in their markup.

Libraries

- **Pusher** is a service that enables real-time communication between clients and servers, making it easy to build interactive features like chat applications and notifications.
- **Bcrypt** is a library used for hashing passwords securely, ensuring that user credentials are protected against attacks.
- **Crypto-js** provides a collection of cryptographic algorithms for JavaScript, allowing developers to encrypt and decrypt data easily.
- **Next-Auth** is an authentication library designed for Next.js applications, offering a simple way to implement various authentication methods, including social logins.

Database

- **MongoDB**

  MongoDB is a NoSQL database that uses a flexible, document-oriented data model to store information in JSON-like format (BSON). This structure allows for easy

scalability and the ability to handle unstructured data, making it ideal for modern applications that require rapid development and agile iteration.

Documentation

- **Draw.io**
  Draw.io was used to create diagrams like ER and DFD for the system analysis and design of ChatMate.
- **Drawsql.app**
  It was used to display graphical representation of all the tables used in the database
- **Excel**
  Gantt chart was created on Excel for project planning.

4.1.2 Implementation Details of Modules

User Module:

In the user module, an account is created by filling the register form details which includes, full name, phone number, email, username and password or by using google provider using next-auth. User can also set their profile image and name later.

Chat Module:

Users are able to initiate chat with other users, create group chats with multiple participants or delete the group chat.

4.2 Testing

In unit testing, the system was evaluated to verify its behavior. To ensure the system is thoroughly tested prior to deployment, these test cases were created. The tests done are outlined below:

4.2.1 Test Cases for Unit Testing

Table 4.1: User Login using Valid Data

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check User login using valid data	Email: prabhat@mail.com Password: 123	Logged into the Chat page	Redirect to Chat	Pass

Table 4.2: User Login using Invalid Data

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check User login using invalid data	Email: 123@mail.com Password: hello	Display an Error Message	Display Error Message	Fail

Table 4.3: User Register with full details

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	User enter full details	Name: Prabhat Gurung Email: prabhatgurung34@gmail.com Password: 123	Logged into the chat page	Redirect to Chat	Pass

Table 4.4: User Register with incomplete details

ID	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	User enter incomplete details	Name: Prabhat Gurung Email: Password: 123	Display an Error Message	Display Error Message	Fail

4.2.2 Test Cases for System Testing

The following aspects have been verified through system testing:

- The app starts without any error on localhost: 3000.
- The components for UI are loaded successfully.
- Chat Message are encrypted on send and on database and decrypted on receive.
- Images links are stored in the database as expected.

Chapter 5: Conclusion and Future Recommendation

5.1 Conclusion

In summary, ChatMate is entering an exciting new phase with its beta version now demonstrating essential features and successfully integrated modules. While the current functionalities such as the authentication system, group chat creation, and visibility for users and non-users provide a solid foundation, there is still significant work ahead. The increasing complexity of the application highlights the need for ongoing improvements and refinements to fully realize the vision for ChatMate.

Looking ahead, numerous opportunities for further development will enhance user experience and engagement. Enhancements to the user interface, additional project information, and updated user profiles will create a more intuitive platform. Implementing profile customization while adding a robust search engine will facilitate smoother interactions. At the end of the project, it was determined that the application could benefit from several enhancements. Tester feedback provided some of these suggestions, while others emerged from collaboration among all parties involved. Further improvements can be pursued in future development phases.

5.2 Lesson Learnt / Outcome

In the following project, we have gained valuable problem-solving skills, teamwork, work independently to find solutions, communication skills, writing skills as well as management skills

- **Problem-Solving Skills:**
  We learned various problem-solving techniques and skills. We were also able to learn to recognize different errors and how to tackle them.
- **Teamwork:**
  As this was a team project, it taught us how to work and collaborate with our peers together. We learned to divide difficult task amongst each other and solve it.
- **Communication Skills:**
  As a team, we had to communicate our ideas, problem and solution to our partners. This project helped develop our communication skills as we had to describe out the problems in an understandable manner.

- **Writing Skills:**
  We have learned to create document like proposal, and documentation related to the project.
- **Management Skills:**
  We learned which component to give more priority to in order to manage our time efficiently. We also learned to divide project based on the difficulty to make the development task less cumbersome.

The final outcome of this project is that users are able to register and login using google provider or manually through providing the credentials. They are able to initiate conversation with other users, create group chats, add members to the group chat, or delete the group chat. Users can also share images using Next Cloudinary.

5.3 Future Recommendation

It would have been advantageous to prioritize programming the project over focusing on documentation during its development. The system is designed to be flexible and can be updated based on user needs. This project can be seen as a foundation for something more substantial, suggesting that further work is still needed.

- Search feature to find specific user can be added
- Functionality for file uploads to accommodate various multimedia formats could be implemented.
- Better encryption for AES Key can be implemented.
- Functionality for group admins and members can be incorporated.
- Greater freedom for profile customization can be implemented.
- Additional sign-in options for social media can be incorporated.

References

[1] E. Kho, V. C. Mawardi, and N. J. Perdana, "Web-based live chat application uses advanced encryption standard methods and rivest shamir adleman," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1007, no. 1, 2020, doi: 10.1088/1757-899X/1007/1/012153.

[2] V. Vaibhav, "Developing an End-to-End Secure Chat Application," *Int. J. Sci. Eng. Technol.*, vol. 12, no. 3, pp. 1–7, 2024, doi: 10.61463/ijset.vol.12.issue3.176.

[3] A. Thakur and K. Dhiman, "Chat Room Using HTML, PHP, CSS, JS, AJAX," pp. 2–5, 2021, doi: 10.6084/M9.FIGSHARE.14869167.

[4] N. Zala, J. Fiaidhi, and V. Agrawal, "ChatterBox- A Real Time Chat Application," *Researcchgate*, 2022, [Online]. Available: https://www.researchgate.net/publication/366228723_ChatterBox-_A_Real_Time_Chat_Application

[5] M. Abdullah SAEED and N. Moaid EDAN, "Design and Implement Video and Chat Application Using Mesh Network," *MINAR Int. J. Appl. Sci. Technol.*, vol. 4, no. 4, pp. 71–84, 2022, doi: 10.47832/2717-8234.13.7.

[6] B. Avinash Bamane *et al.*, "'Enhanced Chat Application,'" *Glob. J. Comput. Sci. Technol. Netw.*, vol. 12, no. 11, 2012, [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d4ba31265f5fe8f08b8e92da679b8d8e8a32539f

Appendix

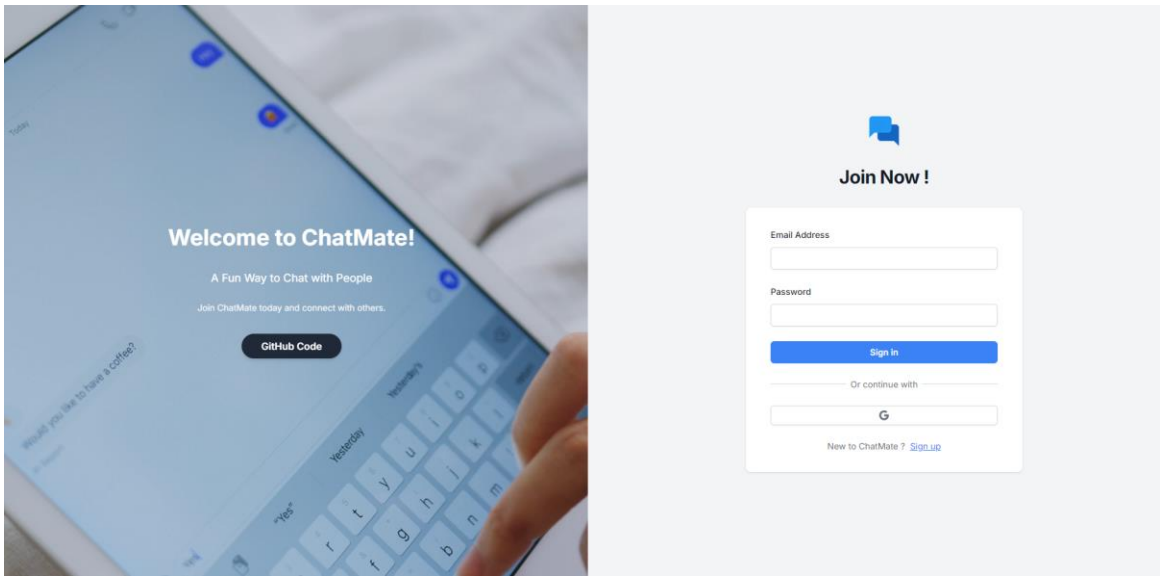

Figure a: Login Page

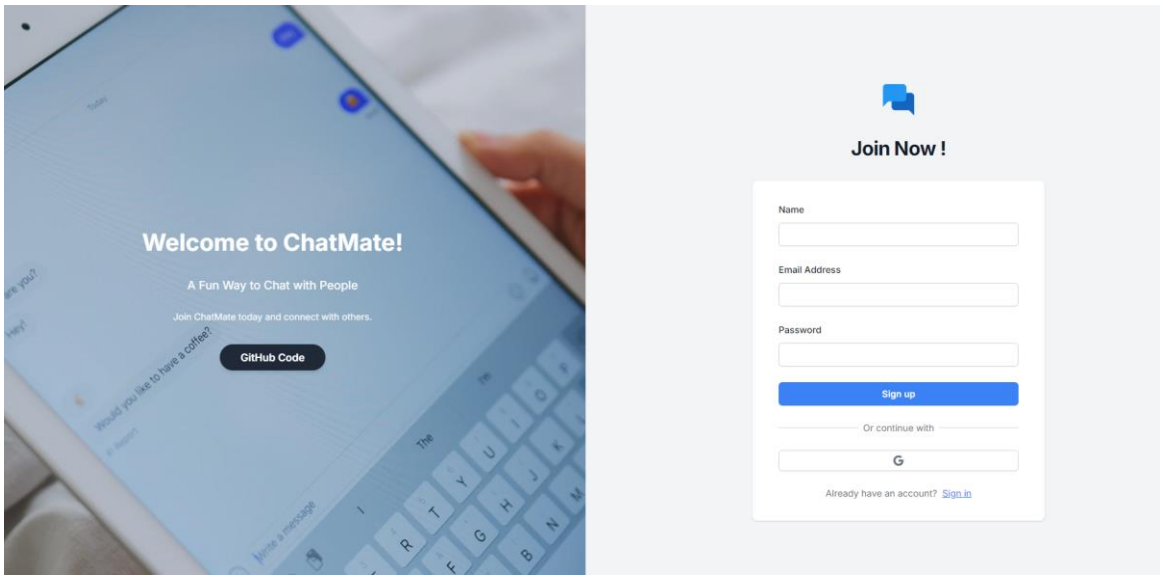

Figure b: Registration Page

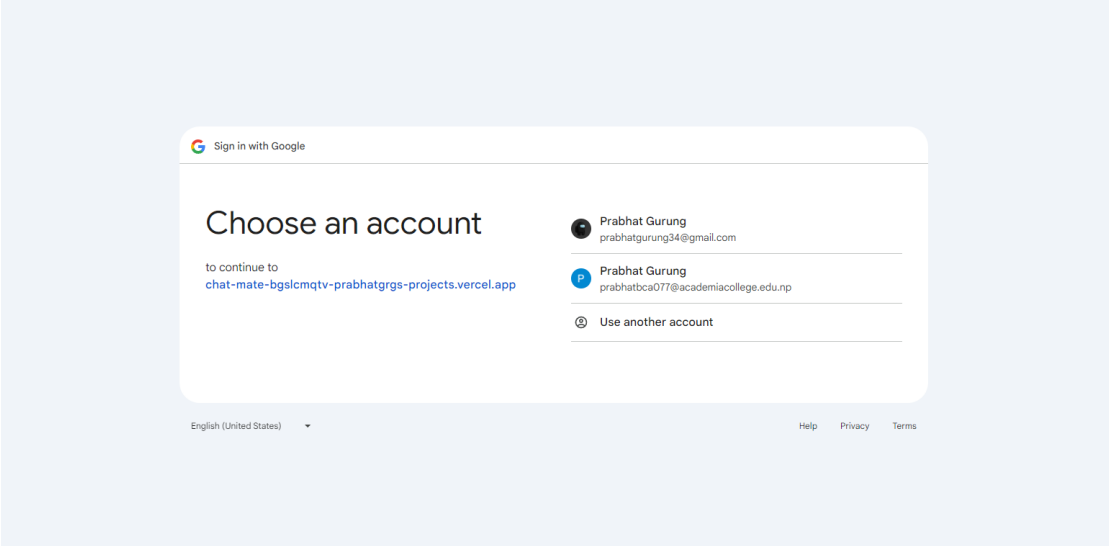

Figure c: Login & Register using Google Provider

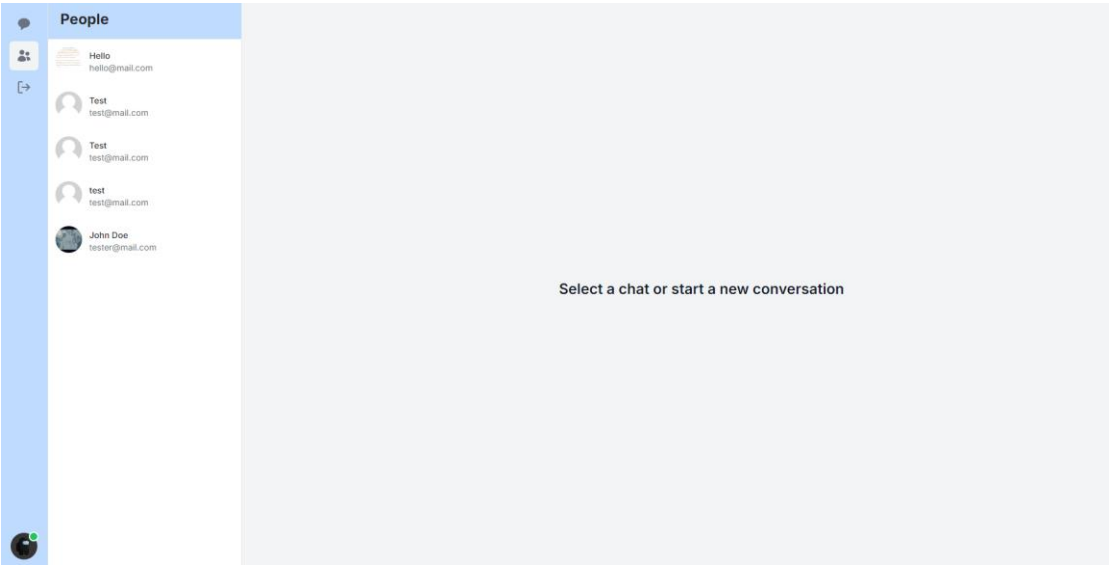

Figure d: Chat Interface

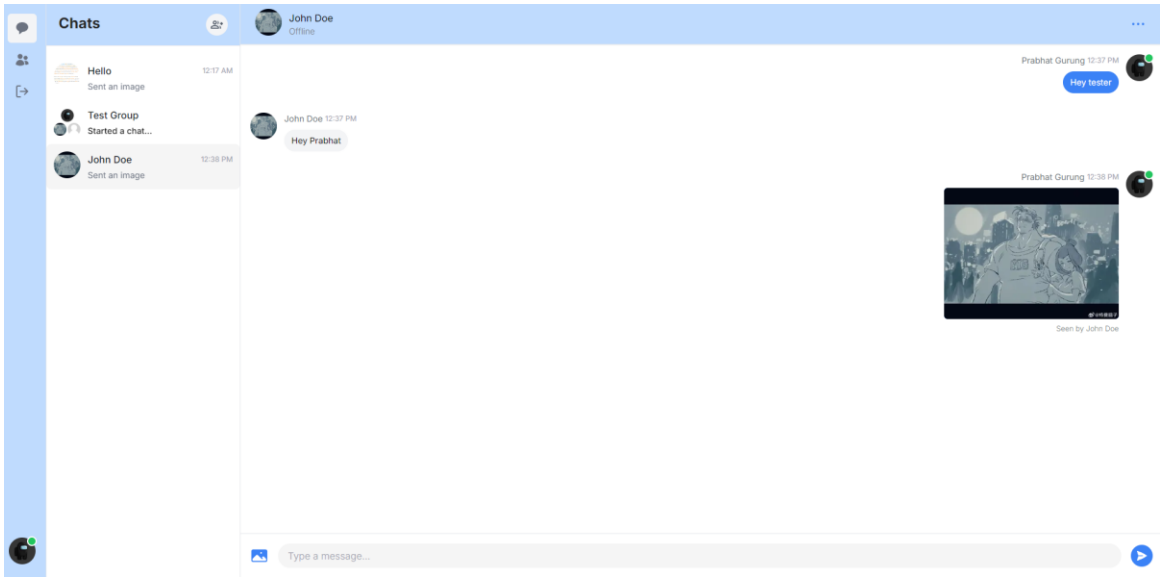

Figure e: Message Interface

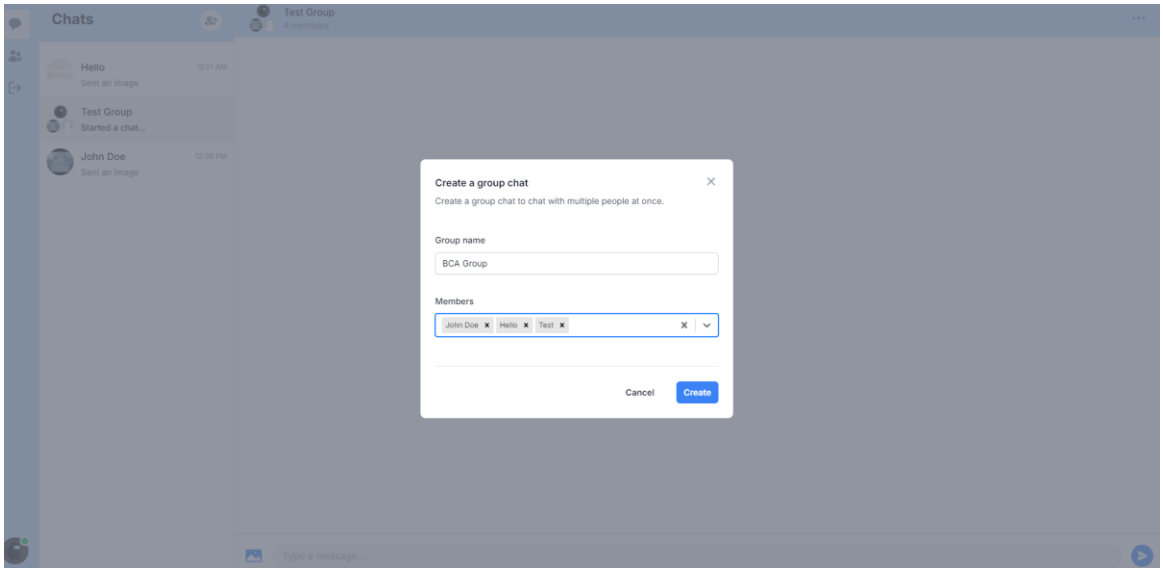

Figure f: Group Creation