

TRIBHUVAN UNIVERSITY



College of Applied Business and Technology

Chabahil, Kathmandu, Nepal

Project Report

On

“Guff Gaff Hub: Web Chat Application”

A Project Report Submitted in Partial Fulfilment of the Requirement of
Bachelor of Science in Computer Science & Information Technology

Submitted by:

Anil Raj Rimal (23554/076)

Himani Bhattarai (23561/076)

Sandesh Shrestha (23580/076)

Under the Supervision of:

Mr. Santosh Sharma

April, 2024

College of Applied Business and Technology

Chabahil, Kathmandu, Nepal



Project Report

On

“Guff Gaff Hub: Web Chat Application”

A Project Report Submitted in Partial Fulfilment of the Requirement of
Bachelor of Science in Computer Science & Information Technology

Submitted by:

Anil Raj Rimal (23554/076)

Himani Bhattarai (23561/076)

Sandesh Shrestha (23580/076)

Under the Supervision of:

Mr. Santosh Sharma

April, 2024

ACKNOWLEDGEMENT

To everybody who contributed to the success of this endeavour, we are eternally grateful. We would like to begin by expressing our deep appreciation to our project supervisor Mr. Santosh Sharma for his important advice, ongoing support, and helpful criticism.

We also want to express our profound gratitude to our instructors Mr. Indra Chaudhary and Mr. Tekendra Nath Yogi whose guidance and instruction have made a significant contribution to our development as a researcher and as a student.

We also want to express our gratitude to everyone who took part in the research and contributed to the information we needed. Finally, we want to thank our loved ones and close friends for their continuous support and inspiration. Throughout our academic career, their support and devotion have been now continual source of inspiration. Once more, we want to express our gratitude to everyone who helped make this initiative a success. We are grateful.

ABSTRACT

This project report, titled "Guff Gaff Hub: Web Chat Application" represents our collaborative effort in fulfilling the Bachelor of Science in Computer Science and Information Technology requirements. Designed to provide a secure platform, the web-based application facilitates seamless online conversations, ensuring the privacy of exchanged text messages through the implementation of the Advanced Encryption Standard (AES) algorithm. Our key objectives include creating a user-friendly web chat environment, prioritizing communication security, and fostering an online community.

Keywords: Web Chat App, Online Communication, AES Encryption, Privacy, User-Friendly Interface.

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	i
ABSTRACT.....	ii
LIST OF ABBREVIATIONS	v
LIST OF FIGURES	vi
LIST OF TABLES	viii
CHAPTER 1 : INTRODUCTION.....	1
1.1 Introduction	1
1.2 Problem Statement	1
1.3 Objectives.....	1
1.4 Scope and Limitation	2
1.5 Development Methodology.....	2
1.6 Report Organization	3
CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW	5
2.1 Background Study	5
2.2 Literature Review	5
CHAPTER 3 : SYSTEM ANALYSIS	8
3.1 System Analysis	8
3.1.1 Requirement Analysis	8
3.1.2 Feasibility Study	10
3.1.3 Analysis.....	12
CHAPTER 4 : SYSTEM DESIGN	15
4.1 Design.....	15
4.2 Algorithm Details	20
CHAPTER 5 : IMPLEMENTATION AND TESTING	27
5.1 Implementation.....	27
5.1.1 Tools Used	27
5.1.2 Implementation Details of Modules.....	29
5.2 Testing.....	30
5.2.1 Test Cases for Unit Testing.....	30
5.2.2 Test Cases for System Testing	32

5.3	Result Analysis.....	33
CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATIONS		34
6.1	Conclusion.....	34
6.2	Future Recommendations.....	34
REFERENCES.....		35
APPENDICES		36

LIST OF ABBREVIATIONS

AES	Advanced Encryption Algorithm
API	Application Programming Interface
CSS	Cascading Style Sheet
DB	Database
DFD	Data Flow Diagram
DH	Diffie Hellman
ER	Entity Relationship
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
ICSET	International Conference on Engineering and Technology
JWT	JSON Web Token
NPM	Node Package Manager
RSA	Rivest–Shamir–Adleman
RDBMS	Relational Database Management System
SDLC	Software Development Life Cycle
SQL	Structured Query Language
TC	Test Case

LIST OF FIGURES

Figure: 3.1 Use Case Diagram	9
Figure: 3.2 Gantt Chart	11
Figure: 3.3 ER Diagram	12
Figure:3.4 Level 0 DFD for Guff Gaff Hub	13
Figure:3.5 Level 1 DFD for User Management Process.....	13
Figure:3.6 Level 1 DFD for Chat Management Process.....	14
Figure:4.1 Database Design	15
Figure:4.2 Login Form.....	16
Figure:4.3 Registration From.....	16
Figure:4.4 Homepage Interface	17
Figure:4.5 Chat Interface	18
Figure:4.6 Profile Interface	18
Figure:4.7 Group Create Interface	19
Figure:4.8 AES State Array	21
Figure:4.9 AES Algorithm Process.....	21
Figure:4.10 Add Round Key Operation.....	22
Figure:4.11 Sub-Bytes Operation	22
Figure:4.12 Shift Rows Operation	22
Figure:4.13 Mix Columns Operations	23
Figure:4.14 Add Round Key Last Step.....	23
Figure:4.15 AES Example	23
Figure:4.16 Key Generation for AES Example	24
Figure:4.17 Add Round Key for AES Example	24
Figure:4.18 Sub-Bytes for AES Example	25
Figure:4.19 Shift Rows for AES Example.....	25
Figure:4.20 Mix Columns for AES Example	25
Figure:4.21 Add Round Key for AES Example	26
Figure:4.22 Ciphertext Conversion for AES Example	26
Figure A.1 Login Page	36
Figure A.2 Registration Page	36
Figure A.3 Chat Homepage	37
Figure A.4 Group Chat	37

Figure A.5 One-to-one chat	38
Figure A.6 Profile Page.....	38
Figure A.7 Cloudinary Image Dashboard.....	38
Figure A.8 Algorithm Config code	39
Figure A.9 Application.properties	40
Figure A.10 Cloudinary Setup	40
Figure A.11 React WebSocket Configuration	41
Figure A.12 Realtime Chat Controller	42

LIST OF TABLES

Table 5.1: Test Case for User Registration	30
Table 5.2: Test Case for User Login	30
Table 5.3: Test Case for Group Creation	31
Table 5.4: Test Case for User Profile Customization	31
Table 5.5: Test Case for Sending Messages	32
Table 5.6: Test Case for Saving Picture in Cloudinary	32
Table 5.7: Test Case for System Usability Testing.....	33

CHAPTER 1 : INTRODUCTION

1.1 Introduction

In the journey of communication technology, chat apps have come a long way. From basic text platforms to today's multimedia-rich environments, these apps have adapted to our changing needs. As digital conversations became a daily norm, there arose a demand for secure, feature-rich, and good-looking messaging platforms.

Guff Gaff Hub steps into this landscape as a fresh take on messaging apps. It's not just keeping up with the evolution; it's setting a new standard. Using technologies like React, Tailwind CSS, and Spring Boot (JAVA), Guff Gaff Hub focuses on user privacy. It uses advanced encryption, like JWT for secure logins and AES for confidential messages. With its easy-to-use interface and real-time messaging, Guff Gaff Hub stands out as a secure and enjoyable space for digital conversations.

More than just features, Guff Gaff Hub is about creating a secure and user-friendly place for people to connect. The frontend, designed with React and Tailwind CSS, ensures a visually appealing and easy-to-navigate interface. The backend, powered by Java Spring Boot, ensures the necessary strength and security. In a changing digital world, Guff Gaff Hub emerges as a symbol of innovation, combining a good design, real-time communication, and advanced security features—an example of the next generation of messaging apps.

1.2 Problem Statement

In the ever-evolving world of digital communication, privacy and security in messaging applications remain pressing concerns. Many existing platforms compromise user data, lacking robust encryption and secure authentication measures. The need for a messaging app that seamlessly combines aesthetic appeal with top-notch security features is evident.

Guff Gaff Hub addresses this gap by offering a user-friendly interface while prioritizing advanced encryption techniques like JWT and AES. The challenge is to create a messaging platform that not only meets but exceeds user expectations for both design and security, ensuring a safe and enjoyable communication experience.

1.3 Objectives

The objective of the project can be stated as:

- To design a simple and friendly interface for smooth and efficient communication.
- To use strong security methods to protect logins and keep messages confidential.

1.4 Scope and Limitation

Guff Gaff-Hub focuses on providing a user-friendly web messaging platform with a core emphasis on secure communication. The scope encompasses real-time messaging, both in individual and group settings. The app prioritizes simplicity and security, ensuring a straightforward and private environment for users engaged in digital conversations.

The different limitations of this project are:

- Web-based system, limiting access for users without a consistent internet connection.
- Missing multimedia file-sharing features found in comparable chat applications.
- Message content is secured with AES but unable to implement any key exchange algorithm for key security like DH or RSA.

1.5 Development Methodology

Waterfall Model

The Waterfall model, known for its sequential and linear approach, is suitable for projects with well-defined and stable requirements. This model ensures a systematic progression through distinct phases, enabling thorough documentation and clarity at each step. Its simplicity makes it ideal for projects where changes are expected to be minimal, providing a structured framework for efficient development. [1]

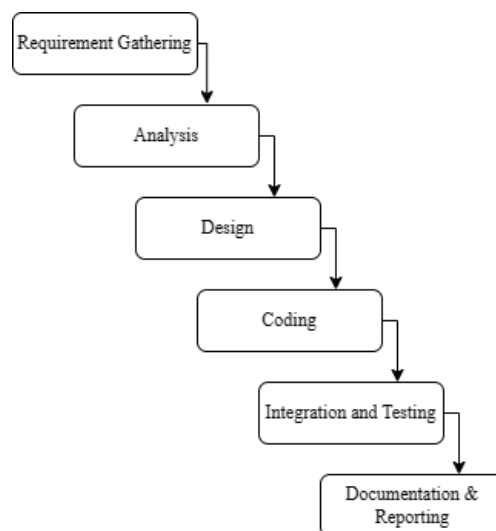


Figure: 1.1 Waterfall Model

Stages of Waterfall Model

The different stages that are involved in Waterfall Model include:

- **Requirement Gathering:** This is the initial phase; project requirements are thoroughly gathered. This is the most important phase for overall project.
- **Analysis:** In this phase, the requirements gathered from previous stage is analysed and documented to create a comprehensive specification that serves as a foundation for other stages.
- **Design:** The design phase translates gathered requirements into a detailed blueprint, outlining system architecture, components, and their interactions.
- **Coding:** Actual system development takes place in this phase, where the design is translated into code.
- **Integration and Testing:** Components are integrated into a complete system during this phase, followed by thorough testing to ensure the entire system functions cohesively according to specified requirements.
- **Documentation & Report:** The final phase involves comprehensive documentation of the entire development process, including user manuals and reports. This documentation serves as a valuable resource for future maintenance and enhancements.

1.6 Report Organization

The report has been prepared following the guidelines provided by Tribhuvan University. The report is separated into different chapters. Each chapter consists of various sub chapters with its contents. The preliminary section of the report consists of Title Page, Acknowledgement, Abstract, Table of Contents, List of Abbreviations, List of Figures, and List of Tables.

The main report is divided into six chapters, which includes:

Chapter 1: Introduction

It includes the general overview of the system and the project as a whole. It includes the Problem Statement, Objectives, Scope/Limitations and the Development Methodology for the project and the system being developed.

Chapter 2: Background Study and Literature Review

It includes the study of the current scenario/environment the system will be deployed into. It includes study of the current trends, preferences of people, the existing system, areas of improvement among others.

Chapter 3: System Analysis

It includes the requirements and feasibility analysis of the system that can be generated through the studies presented in the previous two chapters. It will also include the Flowchart, ER and DFD for the system which specifies the workflow, entities, attributes and their relationships.

Chapter 4: System Design

It includes the design of the database, forms and interface of the system. It also includes the implementation details of the selected methodology and details of the algorithm used.

Chapter 5: Implementation and Testing

It includes the details of different design and development tools used and the implementation and details of the modules presented in the form of code snippets of functions, classes. It also includes the testing of the system with different test cases as per the requirement.

Chapter 6: Conclusion and Future Recommendations

It includes the summary of the system and the project as a whole. It also includes the possibilities/aspects which the system can implement in the future.

The final part of the report consists of References and Appendices. The references are listed in accordance to the IEEE referencing standards and the Appendices include the screenshots of the system and the major source code snippets.

CHAPTER 2 : BACKGROUND STUDY AND LITERATURE REVIEW

2.1 Background Study

The surge in digital communication has been accompanied by a proliferation of web-based chat applications, catering to diverse user needs. These platforms offer real-time interaction, enabling users to connect seamlessly across distances. However, the evolving landscape highlights persistent concerns, particularly in terms of user privacy, security, and the overall user experience. The demand for a web chat app that addresses these challenges while maintaining simplicity and accessibility has become increasingly apparent. [2]

In response to the identified gaps in existing web chat apps, Guff Gaff Hub emerges as a solution that combines aesthetic appeal, user-friendliness, and advanced security features. Recognizing the significance of a messaging platform that prioritizes both form and function, Guff Gaff Hub is designed to provide a secure, enjoyable, and straightforward space for digital conversations. By integrating technologies like React, Tailwind CSS, Spring Boot, and advanced encryption techniques, Guff Gaff Hub aims to redefine the web chat app landscape, offering users a modern and trustworthy platform for their communication needs.

2.2 Literature Review

In 2016, Diotra Henriyan, Devie Pratama Subiyanti, and Rizki Fauzian presented "Designing and deploying a real-time web-based chat server" at the International Conference on Engineering and Technology (ICSET). The research emphasized the need for a real-time and multi-site chat application accessible to many users. The study utilized Node.js as the programming language and MongoDB for website development, providing a clear framework for building an efficient chat server [3].

In 2020, Jhalak Mittal, Arushi Garg, and Shivani Sharma contributed to the literature with "Online Chat Request," published in the International Journal of Research in Engineering, IT, and Social Sciences. Focusing on securing speech requests in online chat, the researchers implemented the XSalsa20 algorithm for its high security and minimal impact on mobile battery life. Their work highlighted XSalsa20's role in enhancing user trust and ensuring confidentiality in online speech transactions [4].

Shifting focus to the current scenario, as of March 6th, 2024, the popularity of chat applications in Nepal is evident. WhatsApp Messenger maintains its top position in the Communication category on the Google Play Store, reflecting consistent user preference. Snapchat secures the 2nd position, maintaining global ranking stability, while Messenger, holding its ground, remains among the top 3 Communication apps. These trends underscore the enduring presence and significance of these communication platforms in the local app landscape [4].

Our goal is to develop Guff Gaff Hub, a web chat app prioritizing real-time messaging, an intuitive interface, and robust security with the use of the Advanced Encryption Standard (AES). Drawing insights from previous studies, we leverage React, Tailwind CSS, and Java Spring Boot for efficiency. Emphasizing user privacy, we implement AES encryption for message confidentiality.

Study of Existing Systems

1. WhatsApp:

WhatsApp is a widely-used messaging application that allows users to send text messages, make voice and video calls, and share multimedia content. Acquired by Facebook, it has become a staple in global communication, offering a user-friendly interface for both personal and business use. [5]

Good Features:

- Ensures secure and private conversations.
- High-quality audio and video communication.

Lacking Features:

- Constraints on file size for sharing.
- Restrictions on integrating with third-party applications.

2. Chat Sansar

Chat Sansar offers online chat rooms where users can communicate in real-time through text, audio, and video, even interacting with 3D characters. Providing a versatile platform, users engage in one-on-one or group conversations, with rooms organized around countries, topics, or themes. The chat service stands out by not requiring registration, sign-up, or account creation, allowing users to join anonymously. Moderated by chat operators and

staff, it emphasizes user safety while fostering new connections and casual conversations.[5]

Good Features:

- Rooms organized around countries, topics, or themes.
- Users can join without registration, ensuring privacy.

Lacking Features:

- While the platform supports 3D characters, the extent of interaction may be limited.
- While promoting safety, user awareness of online safety practices is crucial.

CHAPTER 3 : SYSTEM ANALYSIS

3.1 System Analysis

3.1.1 Requirement Analysis

i. Functional Requirements

Functional requirements are the compass that guides the development of software, spelling out the specific functionalities and features crucial for a system's success. They offer a detailed roadmap, answering the question of "what" the software should do, and act as a framework for developers and stakeholders to ensure the end product aligns with user needs. In essence, functional requirements form the foundation upon which a robust and user-friendly system is built.

Some of the functional requirements for this web app are listed below:

- User Registration and Authentication
- Real-Time Messaging
- Group Creation and Management
- Security and Privacy Measures

Use Case Descriptions

1. User:
 - Registration and Login: The user initiates the registration process, providing necessary details. Upon successful registration, the user can log in securely.
 - Search other registered users: Users can search for and discover other registered users within the application.
 - Send and Receive One-to-One Chat: Users initiate and engage in one-to-one chat conversations, sending and receiving text messages in real-time.
 - Customize Profile: Users have the ability to personalize their profiles, including updating profile pictures and adding relevant information.
 - Create Group by Selecting Registered Users: The user can create a group by selecting and adding registered users to facilitate group conversations.
 - Send and Receive Group Chats: Users interact within created groups, sending and receiving group chat messages.
 - Logout: The user can log out of the application, ensuring account security.

2. System:

- **Encrypt/Decrypt Chat Message:** The system ensures the security and privacy of chat messages by encrypting and decrypting them using advanced encryption standards.
- **Connects Sender/Receiver:** Facilitates the connection between chat message senders and receivers, ensuring smooth and real-time communication.

3. Cloudinary:

- **Saves Images on Cloud:** Cloudinary is responsible for securely saving and managing images uploaded by users, providing a reliable cloud storage solution for multimedia content.

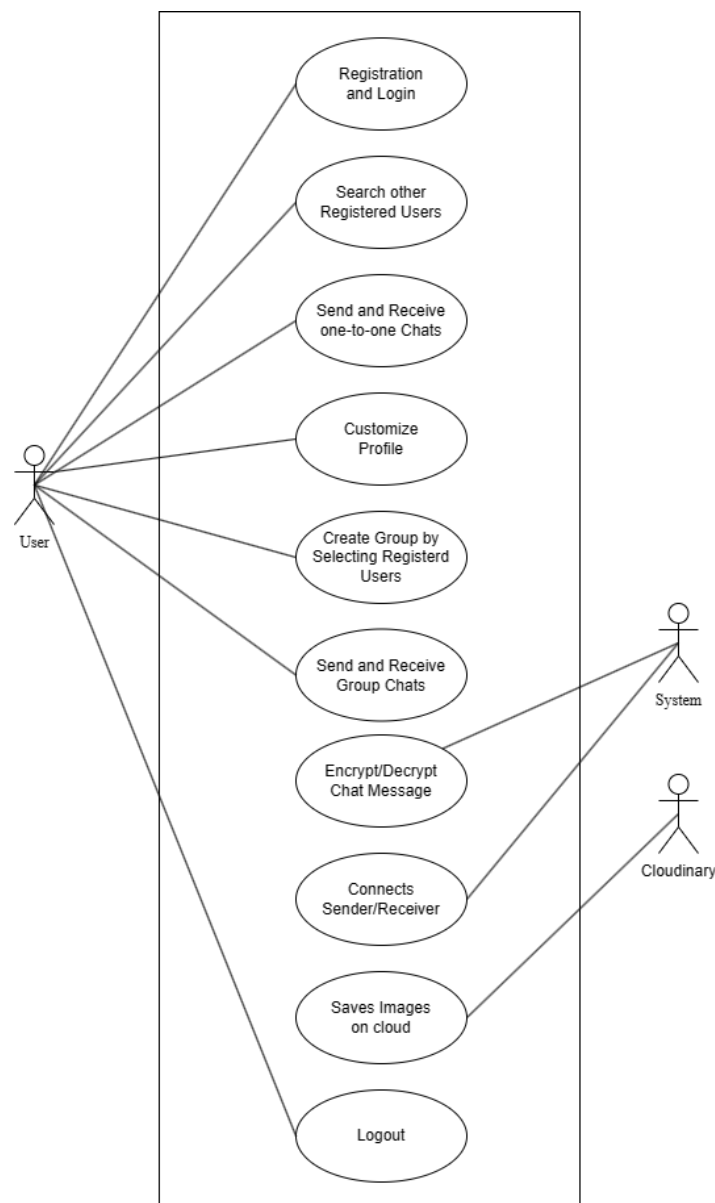


Figure: 3.1 Use Case Diagram

ii. Non-Functional Requirements

Performance: The system must exhibit low latency, ensuring that messages are sent and received with minimal delay, providing an efficient real-time communication experience.

Security: All user data, including messages and profile information, must be encrypted using industry-standard protocols, ensuring the confidentiality and integrity of user data.

Scalability: The system should be scalable to accommodate a growing user base, ensuring consistent performance even as the number of users and messages increases

Reliability: Guff Gaff Hub should aim for high system availability, minimizing downtime and ensuring users can access the application reliably.

Usability: The user interface must be intuitive, providing a user-friendly experience for easy navigation and accessibility across various devices.

3.1.2 Feasibility Study

A feasibility study, also known as feasibility analysis, is an analysis of the viability of an idea. It describes a preliminary study undertaken to determine and document a project's viability. The result of this analysis is used in making the decision whether to proceed with the project or not. In short, a feasibility analysis evaluates the project's potential for success, following feasibility analysis was performed prior to working on the project:

1) Technical Feasibility

All the tools and software products required to construct this web app is easily available in the web. It does not require any special environment to execute. It needs a web server and a DBMS to operate. The operation makes use of Internet and all these aspects are affordable. The application requires simple user interface but understanding logics are complex. It can be done with some assistance from our supervisor.

System Configuration:

- **Hardware Configuration:** Devices like Laptops, Desktop Computers etc. that supports browser and can run spring boot Java, note.js and MySQL Database.
- **Software Configuration:** Operating system with support environment for spring boot Java, note.js and MySQL Database.

Hence, the proposed web chat application is Technically feasible.

2) Operational Feasibility

All the function of this web chat app are possible to create. The database stores the login credential, encrypted message content and the Cloudinary stores the images. The configuration used by system are easier to establish. The web app will operate over the system thus making the user available for messaging.

Hence, the proposed web chat application is operationally feasible.

3) Economic Feasibility

The economic feasibility is clear. After checking the costs and benefits, it appears to be a good investment. With a big market for messaging apps and potential revenue from features or ads, it seems promising. The setup is designed to keep ongoing expenses in check. Overall, it looks like a wise financial move that should bring in more money than it requires.

4) Schedule Feasibility

The Schedule Feasibility is based on a well-structured plan, encompassing key development stages. From feasibility study to testing and documentation, each phase has a designated timeframe.

Activity	1st Week	2nd Week	3rd Week	4th Week	5th Week	6th Week	7th Week	8th Week	9th Week	10th Week	11th Week	12th Week
Requirements Gathering												
Analysis												
Design												
Coding												
Integration and Testing												
Documentation & Report												

Figure: 3.2 Gantt Chart

3.1.3 Analysis

- Data Modelling using ER Diagram

ER Diagrams visually map how data entities connect, using symbols for entities, relationships, and attributes. This visual guide streamlines the database design process, ensuring a clear and efficient representation of data organization.

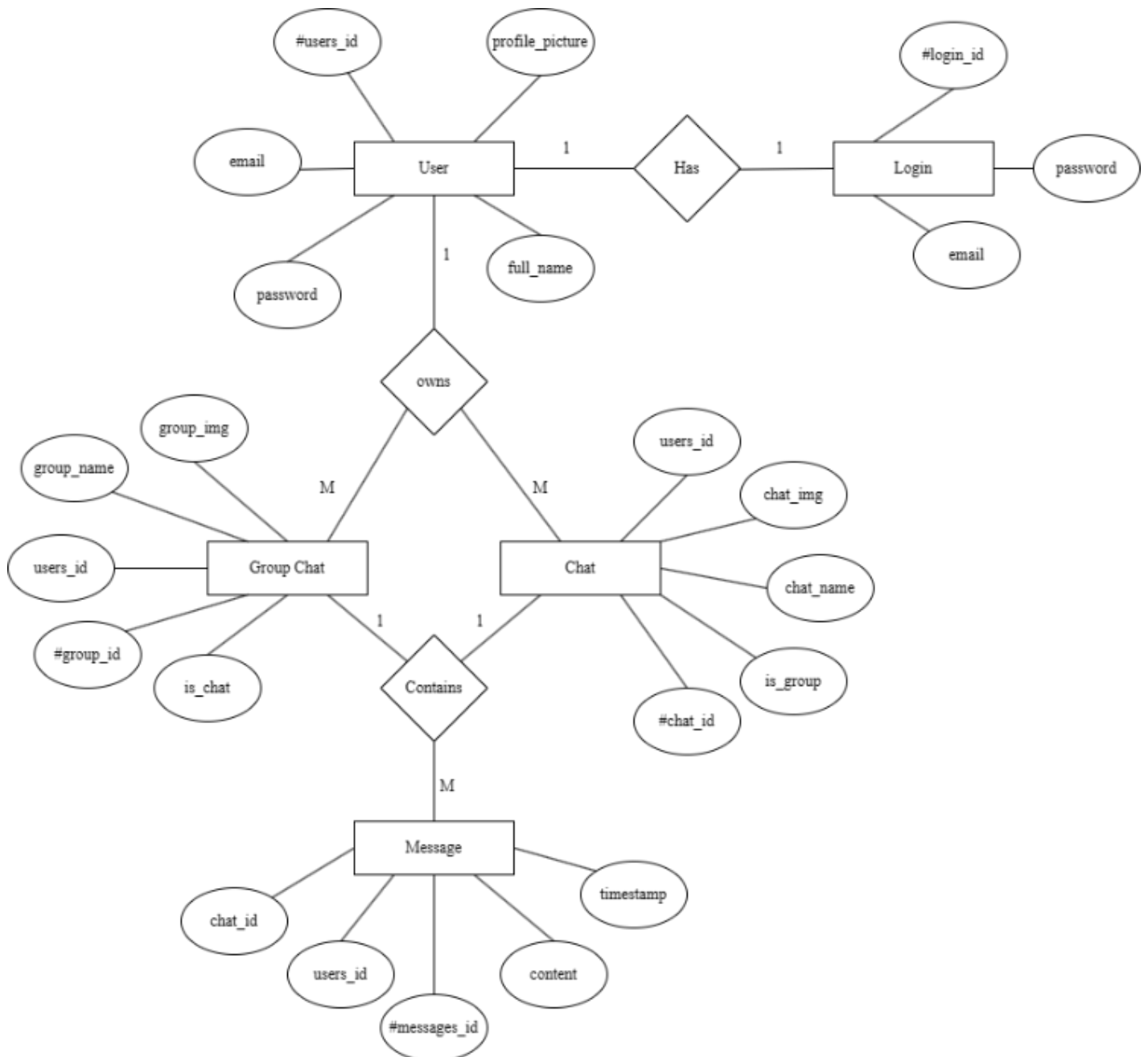


Figure: 3.3 ER Diagram

- Process Modelling using DFD

The context level data flow diagram describes the whole system. The 0 level DFD describes all user modules who operate the system. The Below DFD of the Guff Gaff Hub shows the two entities can operate the overall system, User and Chat Room.

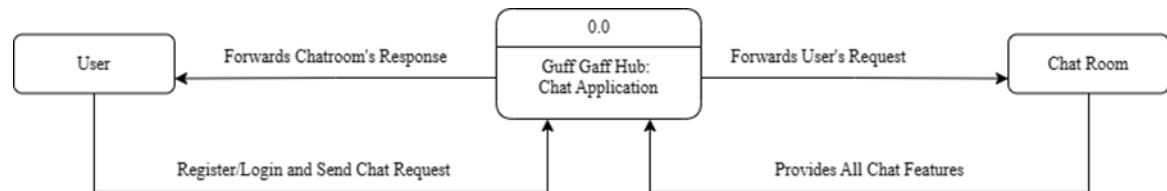


Figure:3.4 Level 0 DFD for Guff Gaff Hub

- Level 1 DFD – User Management Process

User include all the peoples who sends or receive message using our web app. The Users can register to our application and then login as well. After that they can update their profile information; like they can change their name, they can also change the profile picture. They are allowed to send text-based messages to single user or a group as well.

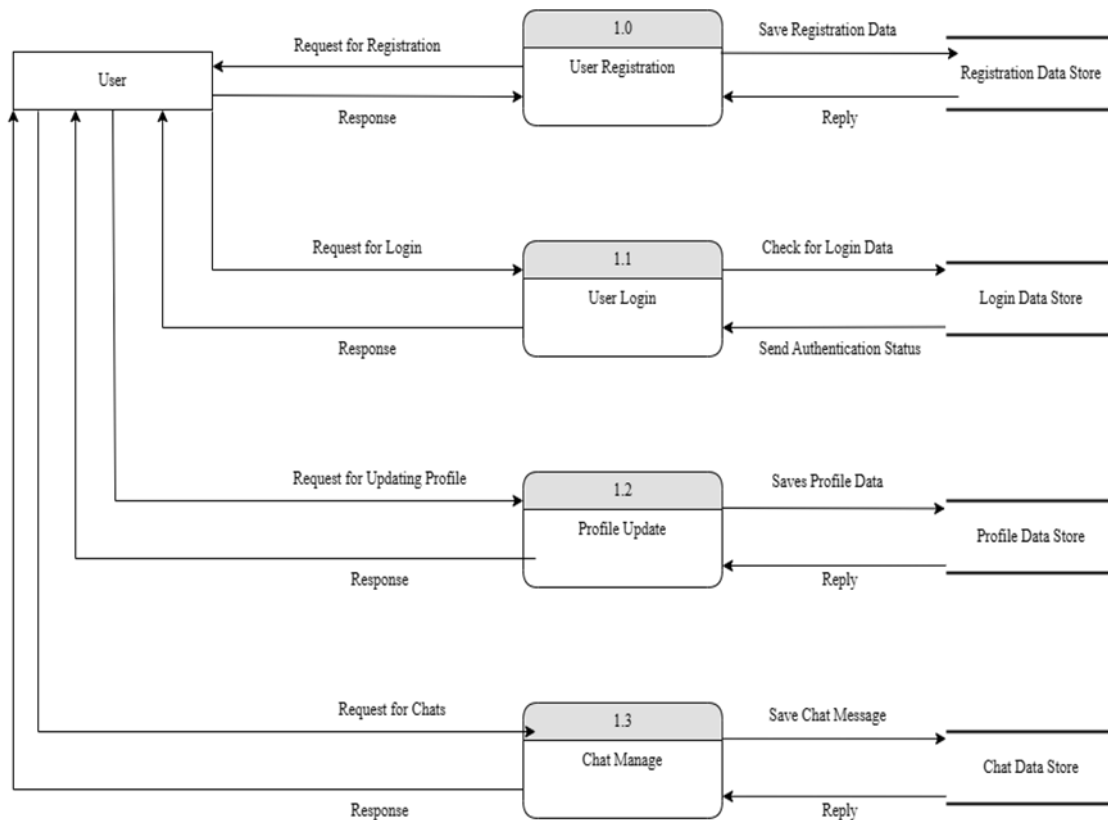


Figure:3.5 Level 1 DFD for User Management Process

- Level 1 DFD – Chat Management Process

The Chat Management Process is integral to the seamless functioning of our chat application. It orchestrates key functionalities, including the creation and updating of chat rooms, message handling, participant management, and the closure of chat rooms. Understanding this process is crucial for gaining insights into the intricate dynamics of our chat platform.

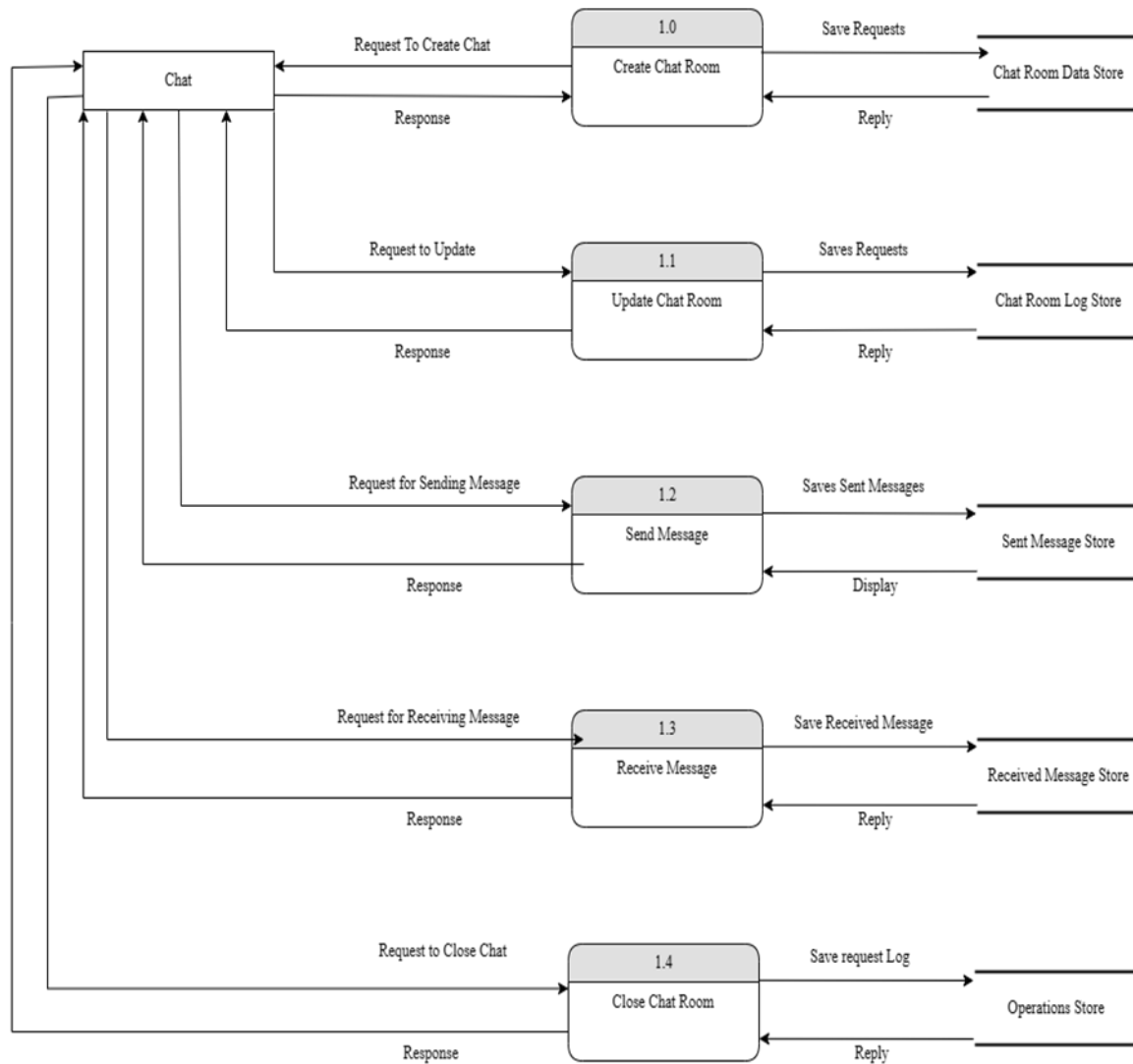


Figure:3.6 Level 1 DFD for Chat Management Process

CHAPTER 4 : SYSTEM DESIGN

4.1 Design

- Database Design

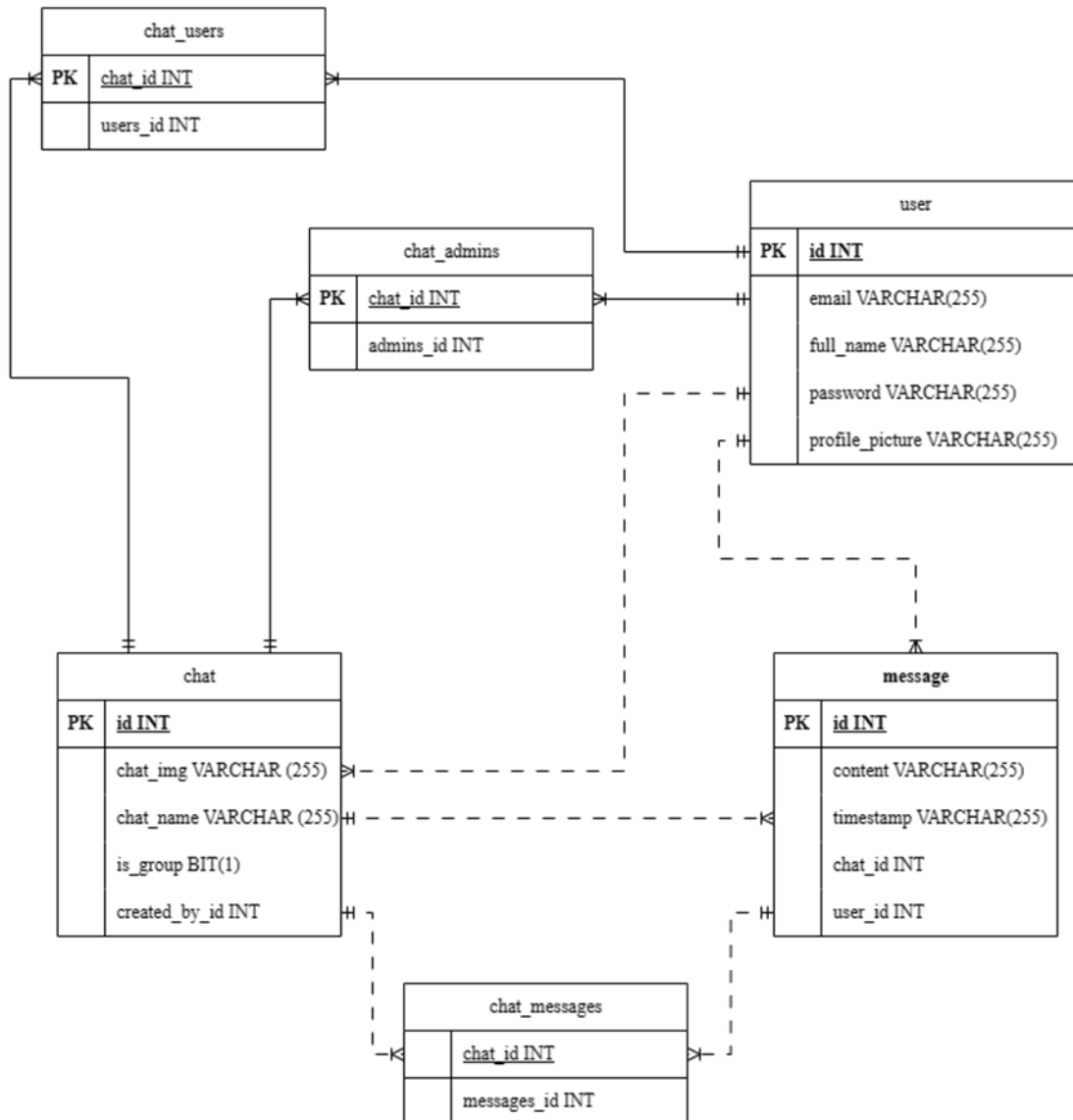
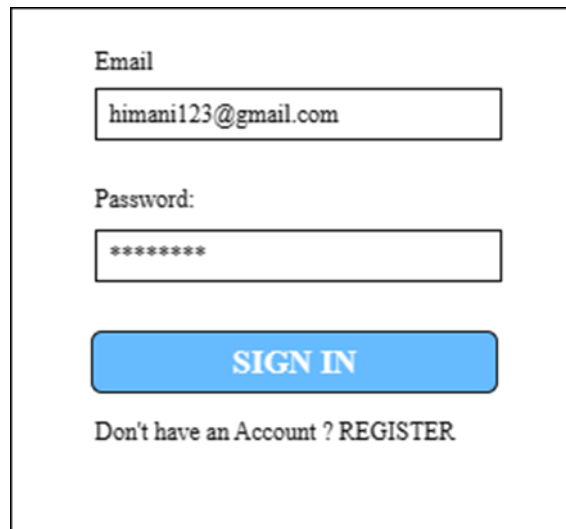


Figure:4.1 Database Design

- Form Design

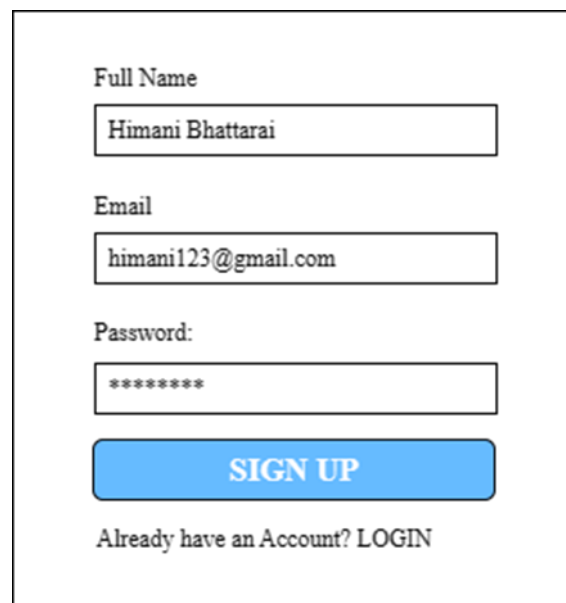
Form design is a critical aspect of user interface development, focusing on creating efficient and user-friendly input interfaces. These forms serve as gateways for users to interact with the system, providing a structured and intuitive way to input data. Design considerations include user experience, data accuracy, and an interface that aligns seamlessly with user needs. The goal is to optimize the data entry process, ensuring a smooth and effective user interaction with the system. [6]

This web chat application consists a Login and a User Registration form:



The login form is a rectangular box with a light gray background. It contains the following elements from top to bottom: a label 'Email' above a text input field containing 'himani123@gmail.com'; a label 'Password:' above a text input field containing seven asterisks '*****'; a blue button with rounded corners and the text 'SIGN IN' in white; and a link 'Don't have an Account ? REGISTER' at the bottom.

Figure:4.2 Login Form



The registration form is a rectangular box with a light gray background. It contains the following elements from top to bottom: a label 'Full Name' above a text input field containing 'Himani Bhattarai'; a label 'Email' above a text input field containing 'himani123@gmail.com'; a label 'Password:' above a text input field containing seven asterisks '*****'; a blue button with rounded corners and the text 'SIGN UP' in white; and a link 'Already have an Account? LOGIN' at the bottom.

Figure:4.3 Registration From

- Interface Design

Interface design encompasses the creation of user-friendly and visually appealing interactions between users and a system. It involves crafting interfaces, be it graphical or command-line, that facilitate efficient and intuitive user navigation. This design phase prioritizes user experience, emphasizing clarity, ease of use, and accessibility. The goal is to ensure that the interface not only meets functional requirements but also enhances user satisfaction by providing a seamless and engaging interaction with the system. [7]

The interfaces on the web chat application:



Figure:4.4 Homepage Interface



Figure:4.5 Chat Interface

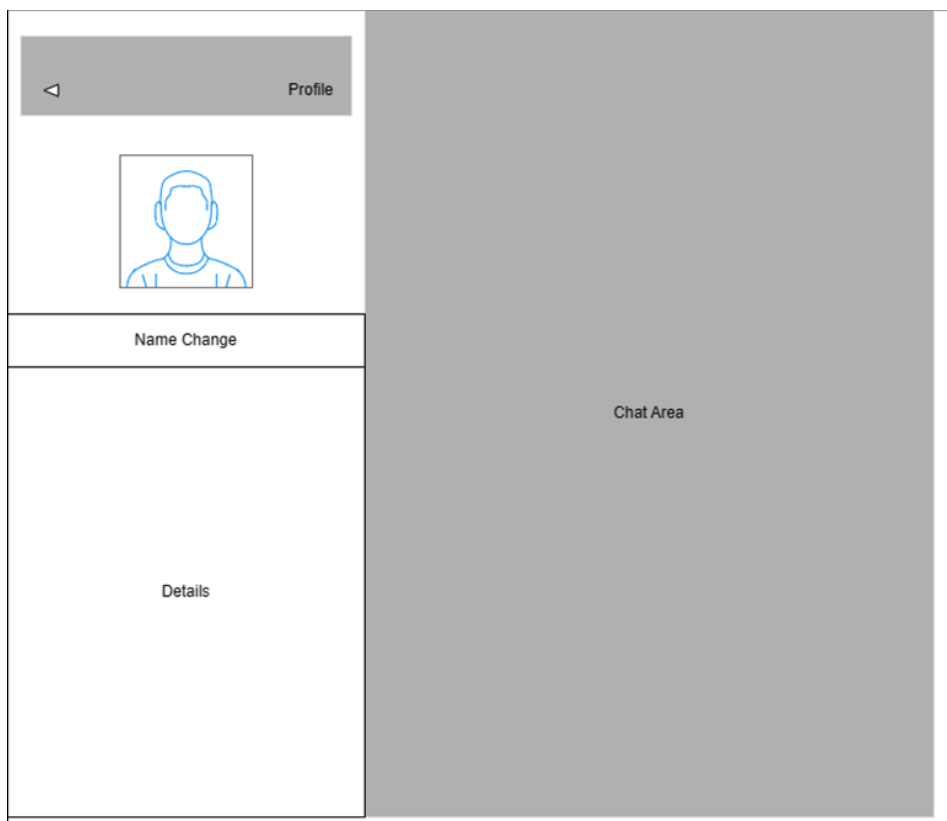


Figure:4.6 Profile Interface

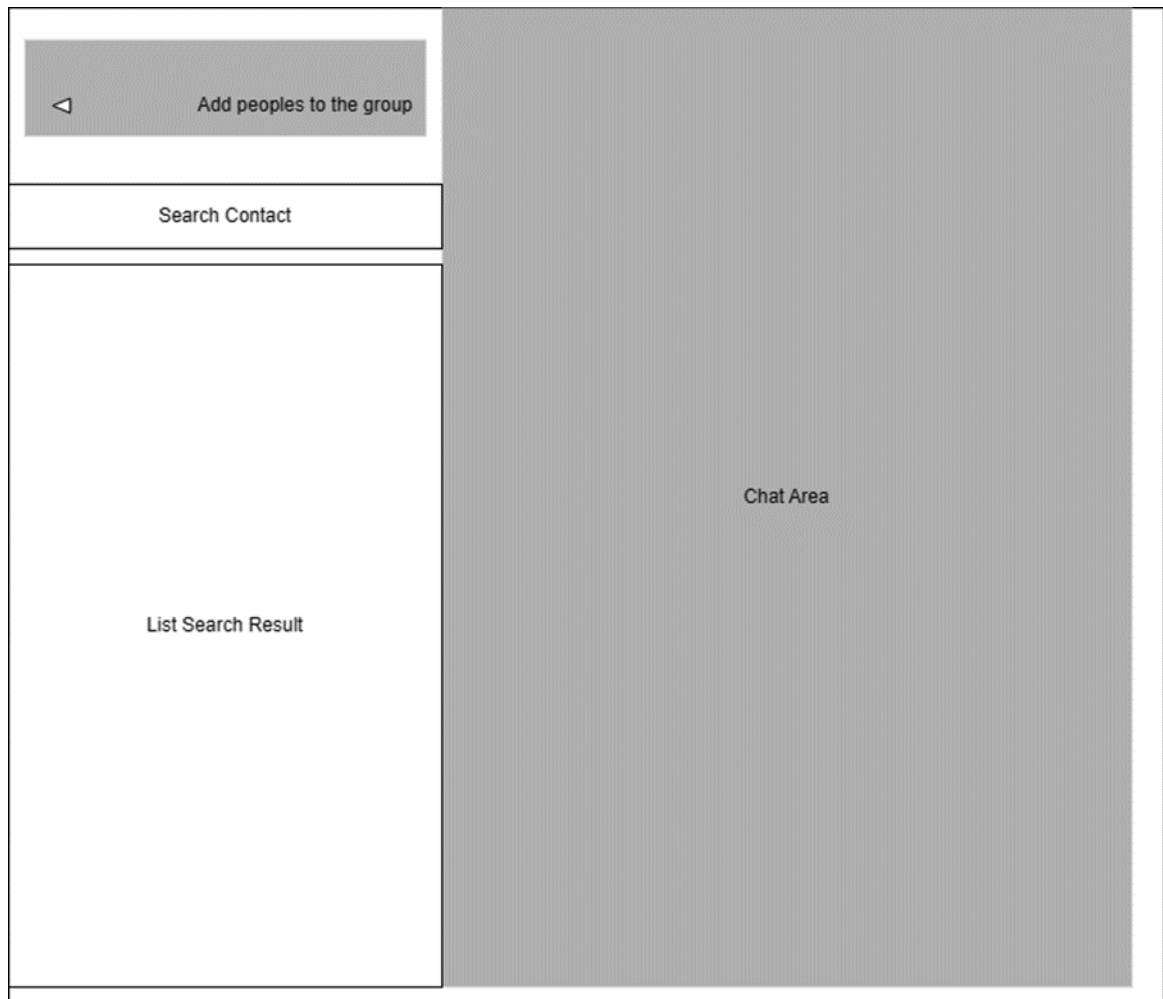


Figure:4.7 Group Create Interface

4.2 Algorithm Details

Advance Encryption Standard (AES)

Overview

The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the ciphertext.

It is based on a substitution-permutation network, also known as an SP network. It consists of a series of linked operations, including replacing inputs with specific outputs (substitutions) and others involving bit shuffling (permutations). [8]

Features

- **SP Network:** It works on an SP network structure rather than a Feistel cipher structure, as seen in the case of the DES algorithm.
- **Key Expansion:** It takes a single key up during the first stage, which is later expanded to multiple keys used in individual rounds.
- **Byte Data:** The AES encryption algorithm does operations on byte data instead of bit data. So, it treats the 128-bit block size as 16 bytes during the encryption procedure.
- **Key Length:** The number of rounds to be carried out depends on the length of the key being used to encrypt data. The 128-bit key size has ten rounds, the 192-bit key size has 12 rounds, and the 256-bit key size has 14 rounds.

Working of AES

To understand the way AES works, we first need to learn how it transmits information between multiple steps. Since a single block is 16 bytes, a 4x4 matrix holds the data in a single block, with each cell holding a single byte of information.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Figure:4.8 AES State Array

The matrix shown in the image above is known as a state array. Similarly, the key being used initially is expanded into $(n+1)$ keys, with n being the number of rounds to be followed in the encryption process. So, for a 128-bit key, the number of rounds is 16, with no. of keys to be generated being $10+1$, which is a total of 11 keys. [8]

Steps to be followed in AES

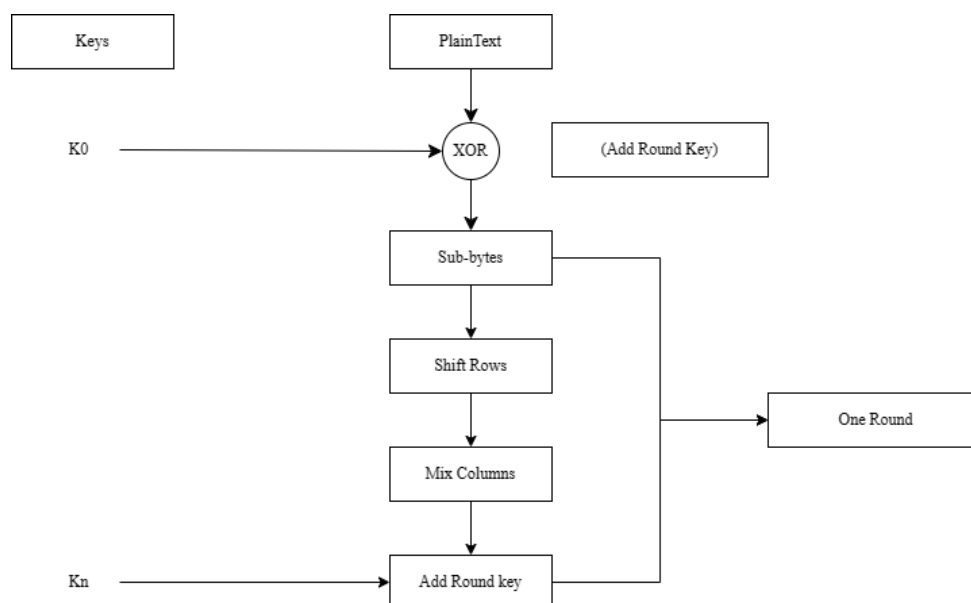


Figure:4.9 AES Algorithm Process

The mentioned steps are to be followed for every block sequentially. Upon successfully encrypting the individual blocks, it joins them together to form the final ciphertext. The steps are as follows:

Add Round Key: You pass the block data stored in the state array through an XOR function with the first key generated (K_0). It passes the resultant state array on as input to the next step.

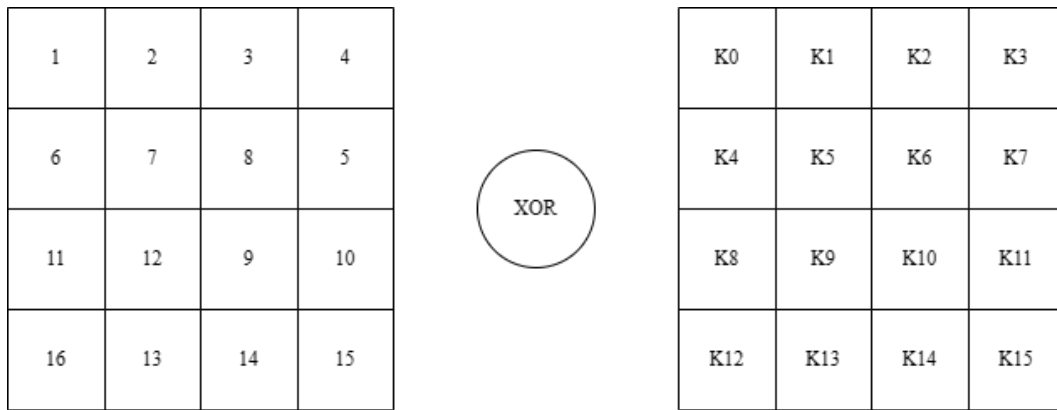


Figure:4.10 Add Round Key Operation

Sub-Bytes: In this step, it converts each byte of the state array into hexadecimal, divided into two equal parts. These parts are the rows and columns, mapped with a substitution box (S-Box) to generate new values for the final state array.

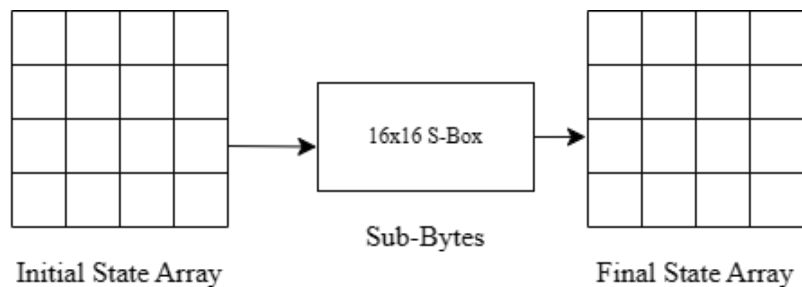


Figure:4.11 Sub-Bytes Operation

Shift Rows: It swaps the row elements among each other. It skips the first row. It shifts the elements in the second row, one position to the left. It also shifts the elements from the third row two consecutive positions to the left, and it shifts the last row three positions to the left.

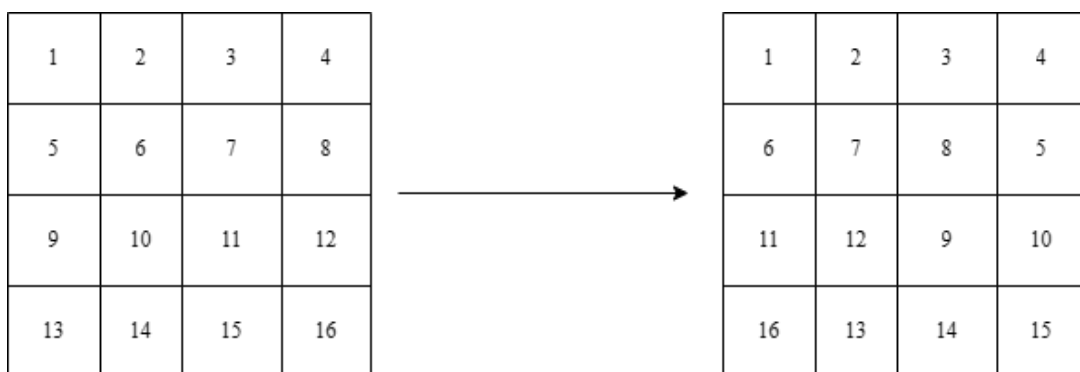


Figure:4.12 Shift Rows Operation

Mix Columns: It multiplies a constant matrix with each column in the state array to get a new column for the subsequent state array. Once all the columns are multiplied with the same constant matrix, you get your state array for the next step. This particular step is not to be done in the last round.

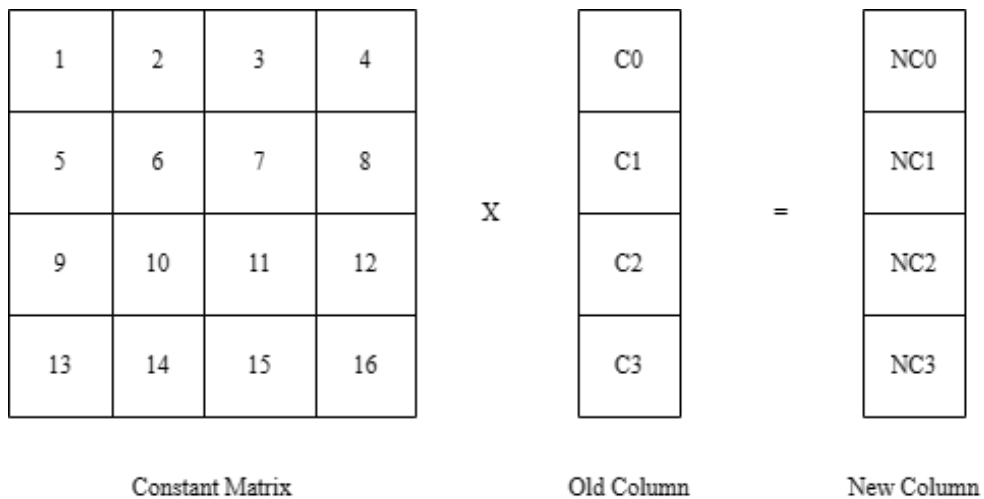


Figure:4.13 Mix Columns Operations

Add Round Key: The respective key for the round is XOR'd with the state array is obtained in the previous step. If this is the last round, the resultant state array becomes the ciphertext for the specific block; else, it passes as the new state array input for the next round.

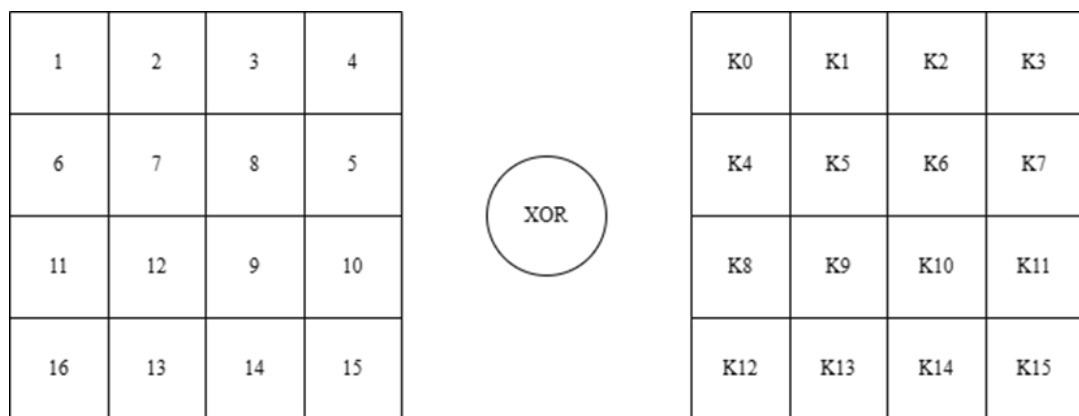


Figure:4.14 Add Round Key Last Step

After understanding the basic step now needed to go through the encryption procedure, understand this example to follow along.

Plaintext - Two One Nine Two															
T	w	o		O	n	e		N	i	n	e		T	w	o
54	77	6F	20	4F	6E	65	20	43	69	6E	25	20	54	77	6F

Plaintext in Hex Format

54 77 6F 20 4F 6E 65 20 43 69 6E 25 20 54 77 6F

Encryption key - Thats my Kung Fu															
T	h	o	t	s		m	y		K	u	n	g		F	u
54	68	61	74	73	20	6D	79	20	4B	75	6E	67	20	46	75

Encryption Key in Hex Format

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Figure:4.15 AES Example

As seen in the image above, the plaintext and encryption convert keys to hex format before the operations begin. Accordingly, we can also generate the keys for the next ten rounds, as:

Keys generated for every round

- Round 0: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- Round 1: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
- Round 2: 56 08 20 07 C7 1A B1 8F 76 43 55 69 A0 3A F7 FA
- Round 3: D2 60 0D E7 15 7A BC 68 63 39 E9 01 C3 03 1E FB
- Round 4: A1 12 02 C9 B4 68 BE A1 D7 51 57 A0 14 52 49 5B
- Round 5: B1 29 3B 33 05 41 85 92 D2 10 D2 32 C6 42 9B 69
- Round 6: BD 3D C2 B7 B8 7C 47 15 6A 6C 95 27 AC 2E 0E 4E
- Round 7: CC 96 ED 16 74 EA AA 03 1E 86 3F 24 B2 A8 31 6A
- Round 8: 8E 51 EF 21 FA BB 45 22 E4 3D 7A 06 56 95 4B 6C
- Round 9: BF E2 BF 90 45 59 FA B2 A1 64 80 B4 F7 F1 CB D8
- Round 10: 28 FD DE F8 6D A4 24 4A CC C0 A4 FE 3B 31 6F 26

Figure:4.16 Key Generation for AES Example

We need to follow the same steps explained above, sequentially extracting the state array and passing it off as input to the next round. The steps are as follows:

Add Round Key

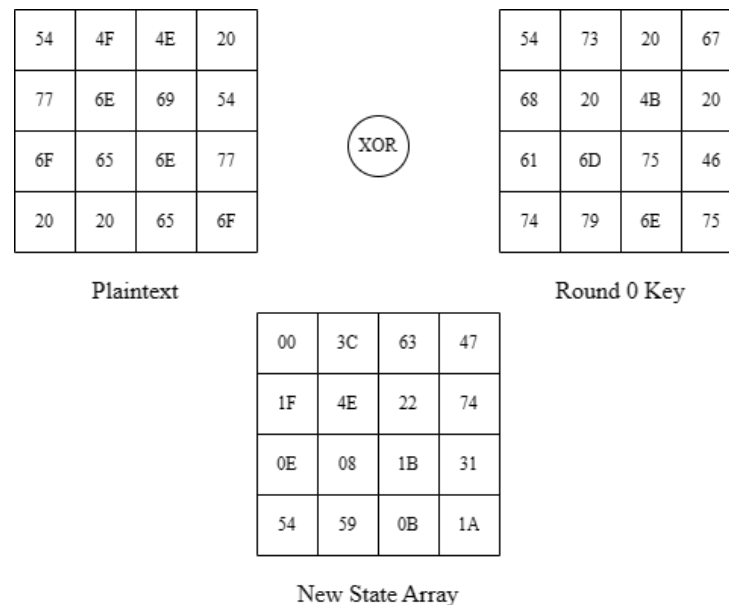


Figure:4.17 Add Round Key for AES Example

Sub-Bytes: It passes the elements through a 16x16 S-Box to get a completely new state array.

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

New State Array

Figure:4.18 Sub-Bytes for AES Example

Shift Rows:

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

→

63	EB	9F	A0
2F	93	92	C0
AF	C7	AB	30
A2	20	CB	2B

Old State Array

New State Array

Figure:4.19 Shift Rows for AES Example

Mix Columns:

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

X

63	EB	9F	A0
2F	93	92	C0
AF	C7	AB	30
A2	20	CB	2B

→

BA	84	E8	1B
75	A4	8D	40
F4	8B	06	7D
7A	32	0E	5D

Constant Matrix

Old State Array

New State Array

Figure:4.20 Mix Columns for AES Example

Add Round Key:

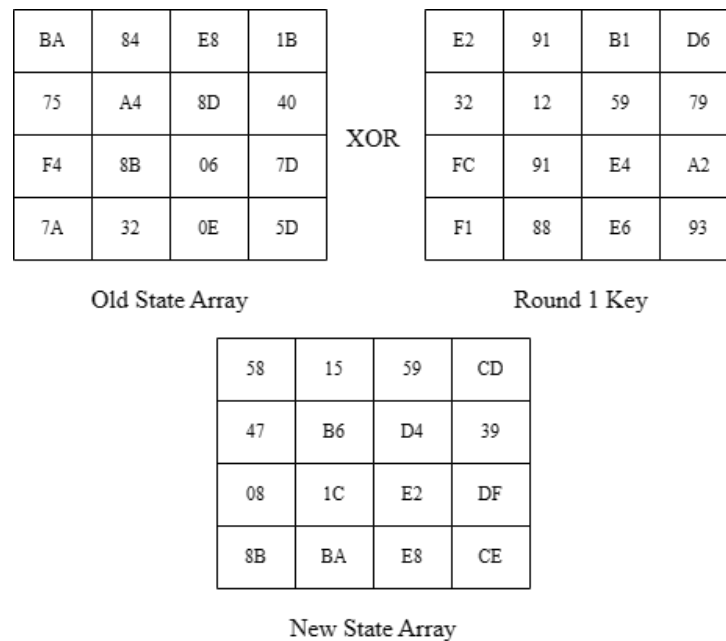


Figure:4.21 Add Round Key for AES Example

This state array is now the final ciphertext for this particular round. This becomes the input for the next round. Depending on the key length, you repeat the above steps until you complete round 10, after which you receive the final ciphertext. [8]

Final State Array after Round 10

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

AES Final Output

29 C3 50 5F 57 14 20 F6 40 22 99 B3 1A 02 D7 3A



Ciphertext

Figure:4.22 Ciphertext Conversion for AES Example

CHAPTER 5 : IMPLEMENTATION AND TESTING

5.1 Implementation

5.1.1 Tools Used

Design and Development Tools

- **IntelliJ IDEA:** IntelliJ IDEA stands out as a robust integrated development environment (IDE) tailored for Java development. Packed with smart coding assistance, advanced refactoring, and seamless integration with build tools, it fosters a highly productive coding experience. Its extensive set of features, including version control and testing support, makes it a go-to choice for Java developers seeking an efficient and feature-rich development environment.

In the project, IntelliJ IDEA was used as the main code editor to write the codes for the platform. We choose IntelliJ Idea as the primary code editor for the project as it is a freemium software with a user-friendly interface and support for multiple programming languages and added functionality like debugging, syntax highlighting, intelligent code compilation, code refactoring, embedded Git, etc.

- **Draw.io:** Draw.io is a versatile online diagramming tool that simplifies the creation of visual representations. With its drag-and-drop interface, it allows for effortless design of flowcharts, diagrams, and various visual aids. Particularly useful for system architects and project planners, Draw.io facilitates collaborative diagramming, enabling teams to work together seamlessly. Its flexibility and ease of use make it an invaluable tool for visualizing project components and system structures.

In the project, Draw.io was used as a tool to develop diagrams and prototypes. We choose Draw.io as the tool for diagramming and prototyping as it is open source, easy to use, both desktop and web based with proper syncing, easy sharing options, wireframing feature and easy modification.

Front End Tools

- **React:** React is a JavaScript library for building user interfaces. It allows you to create reusable UI components, manage state efficiently, and build dynamic and interactive web applications.
- **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework that provides a set of low-level utility classes to build designs directly in your markup. It promotes a rapid and highly customizable styling approach.
- **Redux:** Redux is a state management library for JavaScript applications, particularly those built with React. It helps manage the application's state in a predictable way, making it easier to understand, test, and maintain.
- **npm (Node Package Manager):** npm is the default package manager for Node.js and is widely used for managing and installing JavaScript packages. It plays a crucial role in handling dependencies, scripts, and project configurations.

Back End Tools

- **Java with Spring Boot:** Java is a versatile, object-oriented programming language widely used for back-end development. Spring Boot is a framework that simplifies the development of Java applications, providing conventions and defaults for common tasks. It's particularly known for its ease of use, quick setup, and robust features for building RESTful APIs.
- **MySQL:** MySQL is a popular open-source relational database management system (RDBMS). It is widely used for its reliability, scalability, and support for SQL queries. MySQL is an excellent choice for applications requiring structured data storage and relational database capabilities.
- **Postman:** Postman is a powerful tool for API development and testing. It allows you to create, test, and document APIs by sending HTTP requests and analysing responses. Postman simplifies the process of API testing, making it easier to ensure the correctness and reliability of your back-end services.

5.1.2 Implementation Details of Modules

Modules are the partitions of any project done to ease the task of development. Different modules are designed so that debugging and other development phases get the easiest implementation.

The different modules of the chat app are:

i. Home Page

The home page for the chat app is the Login Page itself. First of all, a user needs to login in order to get into the chat app and use the chat feature.

ii. User Landing Page

The user after login, or sign up are redirected to the landing page. This page displays the chat cards that any users was involved into. In order to chat, users need to click and open that chat card in order to see old message or send a new one.

iii. Profile Page

The Profile Page allows users to change their name. This page also lets users to upload or change the profile picture.

iv. Create Group Page

This Page allows the user to create a group. User can search registered players and add/remove them while creating a group. This also allows the user to add a group name and also to add a group profile picture as well.

v. Chat Page

The Chat Page allows users to communicate with each other. The communication can be one to one or a group. They can simply send/Receive text-based message on this page.

5.2 Testing

5.2.1 Test Cases for Unit Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. The following test scenarios were used to test the system after the build is completed.

Table 5.1: Test Case for User Registration

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Register with valid details	User registration successful	User registered successfully	Pass
TC-02	Attempt registration without required details	Fail to register	Registration Failed	Pass
TC-03	Attempt registration with existing email	Fail to register	Registration Failed	Pass

Table 5.2: Test Case for User Login

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Login with correct credentials	User login successful	User registered successfully	Pass
TC-02	Attempt login without entering credentials	Fail to login	Login Failed	Pass
TC-03	Attempt login with incorrect email	Fail to login	Login Failed	Pass
TC-04	Attempt login with incorrect password	Fail to login	Login Failed	Pass

Table 5.3: Test Case for Group Creation

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Create group with unique name and valid members	Group creation successful	Successfully created a group	Pass
TC-02	Attempt to create group with existing name	Group creation fail	Failed to create a group	Pass
TC-03	Create group without adding members	Group creation fail	Failed to create a group	Pass
TC-04	Add multiple users to create a group	User addition successful	Successfully added group users	Pass

Table 5.4: Test Case for User Profile Customization

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Update display name and profile picture	Name and Profile update successfully	Name and Profile updated successfully	Pass
TC-02	Attempt to update profile without entering details	Fail to update profile	Failed to update profile	Pass

Table 5.5: Test Case for Sending Messages

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Send a text message to an individual user	Message sent successfully	Message received by the intended user	Pass
TC-02	Send a text message to a group	Message sent successfully to the group	Message received by all group members	Pass
TC-04	Attempt to send message with emoji	Fail to send the emoji message	Failed to send an emoji message	Pass

Table 5.6: Test Case for Saving Picture in Cloudinary

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Upload and save a picture to Cloudinary	Picture uploaded successfully	Picture saved on Cloudinary matches expected content	Pass
TC-02	Attempt to upload an invalid picture format	Fail to upload the picture	Proper error message displayed; upload rejected	Pass

5.2.2 Test Cases for System Testing

System testing is defined as testing of a complete and fully integrated software product. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such should require no knowledge of

inner design of code or logic. One of the types of system testing is the usability testing which is performed in the system.

Table 5.7: Test Case for System Usability Testing

Test Id	Test Case Description	Expected Result	Observed Result	Remarks
TC-01	Test every functionality of a chat app	Every functionality should be working	Everything is working	Pass
TC-02	Chat message content should be saved in encrypted format in the Database	Message content should save in encrypted format	Successfully saved message content in encrypted format	Pass

5.3 Result Analysis

All the modules of the system are working as expected and the output of the system is up to expectations. The developed web application provides us with the feature of registering a new user, login, sending single chat message, sending group chat message, creating group, customizing profile, search registered users and most importantly saving message content in encrypted form in the database.

This application uses an efficient algorithm to encrypt and decrypt the message content and also uses a socket programming to establishes the connection between the sender and the receiver.

Main focus is given on creating the user-friendly interface so that user can navigate throughout the system very easily. User can view their old messages, send new one and edit profile info through the profile system very easily. From the user landing page, they can navigate to any of the chat and carry on their conversation.

Overall, the web chat application is working according to expectations and without any problem.

CHAPTER 6 : CONCLUSION AND FUTURE RECOMMENDATIONS

6.1 Conclusion

The development of Guff Gaff Hub, a secure and feature-rich web-based chat application, has been a rewarding endeavour. The integration of technologies such as React, Tailwind CSS, and Spring Boot has resulted in a visually appealing and robust frontend, backed by a secure and efficient backend. The implementation of real-time messaging, group creation, and individual messaging features enhances the user experience, promoting seamless communication.

The incorporation of Advanced Encryption Standard (AES) and Cloudinary for image uploading ensures the privacy and smooth sharing of messages within the application. The utilization of Spring Security and JSON Web Tokens (JWT) contributes to a secure login system, assuring users of the confidentiality of their interactions.

6.2 Future Recommendations

Guff Gaff Hub, a web chat application designed for universal connectivity and communication, exhibits considerable potential in today's digital landscape with the rising number of users. However, to unlock further capabilities and enhance user experience, the integration of multimedia messaging support and the introduction of calling facilities are suggested. These additions would contribute to a more dynamic and versatile platform, catering to diverse communication preferences.

Moving forward, the application's potential can be maximized by considering the following recommendations.

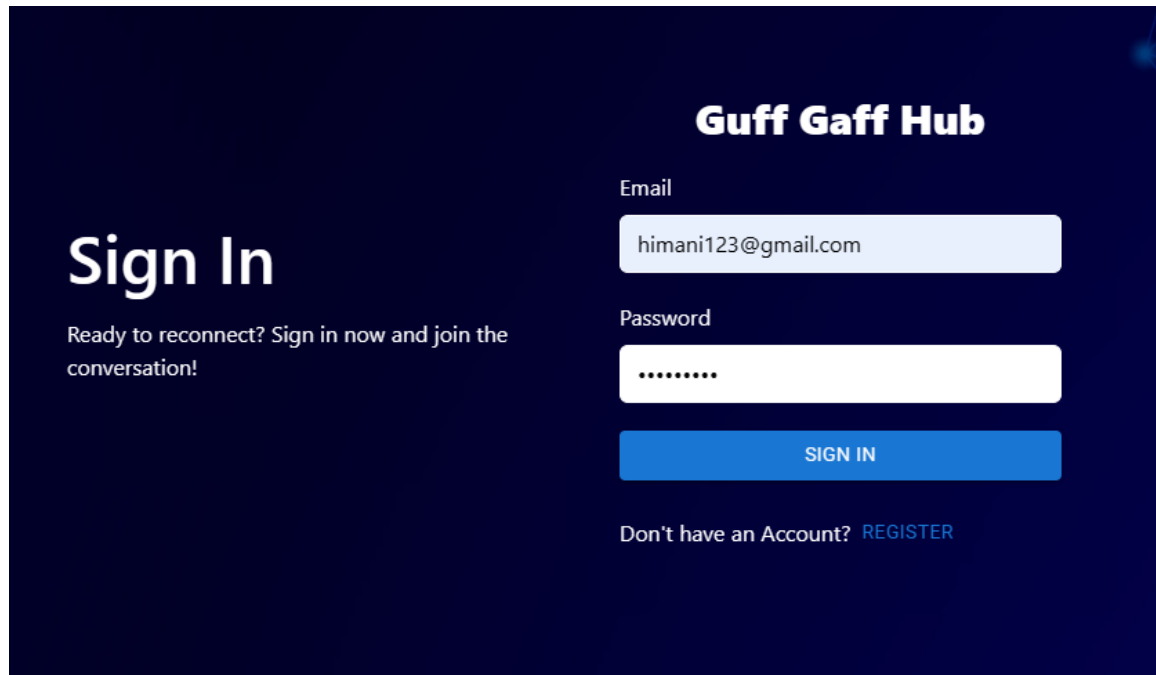
- Enable users to share images and videos for a more expressive communication experience.
- Introduce calling features to enhance real-time communication within the application.
- Prioritize a simple and intuitive design for an effortless user experience.
- Integrate any one of the key exchange algorithms to ensure proper key security in a chat app.

REFERENCES

- [1]GfG, “Waterfall Model - Software Engineering,” GeeksforGeeks, <https://www.geeksforgeeks.org/waterfall-model/> (accessed Nov. 18, 2023).
- [2]S. Reporter, “Chat app development: Revolutionizing Communication in the Digital age,” The Edinburgh Reporter, <https://theedinburghreporter.co.uk/2023/10/chat-app-development-revolutionizing-communication-in-the-digital-age/> (accessed Nov. 21, 2023).
- [3]Design and implementation of web based real time chat interfacing ..., <https://ieeexplore.ieee.org/document/7849628/> (accessed Dec. 6, 2023).
- [4]I. Publication, “Development of Chat Application,” International Journal for Research in Applied Science & Engineering Technology (IJRASET), https://www.academia.edu/81380770/Development_of_Chat_Application (accessed Dec. 7, 2023).
- [5]Top communication apps ranking - most popular communication apps ..., <https://www.similarweb.com/top-apps/google/nepal/communication/> (accessed Dec. 21, 2023).
- [6]Design thought of registration and login. order to save user’s..., https://www.researchgate.net/figure/Design-thought-of-registration-and-login-order-to-save-users-registration-time-it-can_fig1_325340109 (accessed Jan. 13, 2024).
- [7]“Importance of user interface design in Mobile applications,” InvoZone, <https://invozone.com/blog/the-importance-of-user-interface-design-in-mobile-applications/> (accessed Mar. Jan 18, 2024).
- [8]B. K. Jena, “What is AES encryption and how does it work?” Simplilearn.com, Feb. 09, 2023. <https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption> (accessed Mar. 10, 2024)

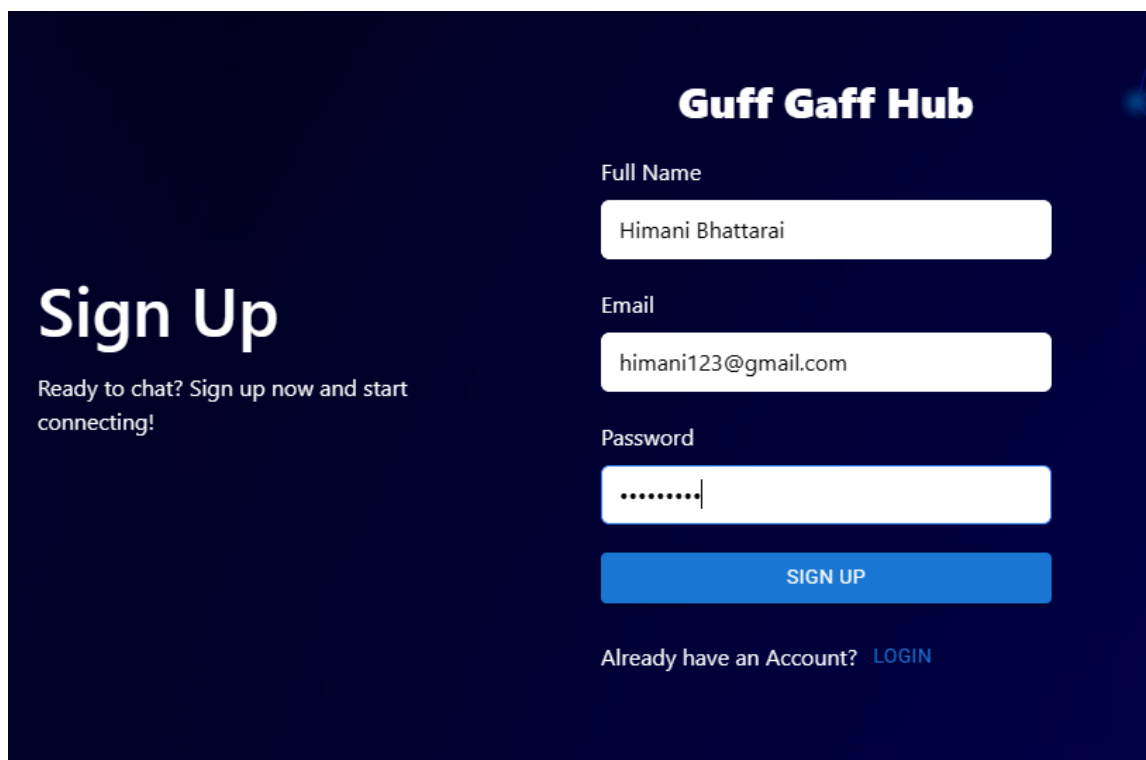
APPENDICES

SNAPSHOTS



The image shows the 'Sign In' page for 'Guff Gaff Hub'. The page has a dark blue background. On the left, the text 'Sign In' is in large white font, followed by 'Ready to reconnect? Sign in now and join the conversation!'. On the right, there is a form with two input fields: 'Email' (containing 'himani123@gmail.com') and 'Password' (containing seven dots). Below the password field is a blue 'SIGN IN' button. At the bottom right, there is a link: 'Don't have an Account? [REGISTER](#)'.

Figure A.1 Login Page



The image shows the 'Sign Up' page for 'Guff Gaff Hub'. The page has a dark blue background. On the left, the text 'Sign Up' is in large white font, followed by 'Ready to chat? Sign up now and start connecting!'. On the right, there is a form with three input fields: 'Full Name' (containing 'Himani Bhattarai'), 'Email' (containing 'himani123@gmail.com'), and 'Password' (containing seven dots). Below the password field is a blue 'SIGN UP' button. At the bottom right, there is a link: 'Already have an Account? [LOGIN](#)'.

Figure A.2 Registration Page

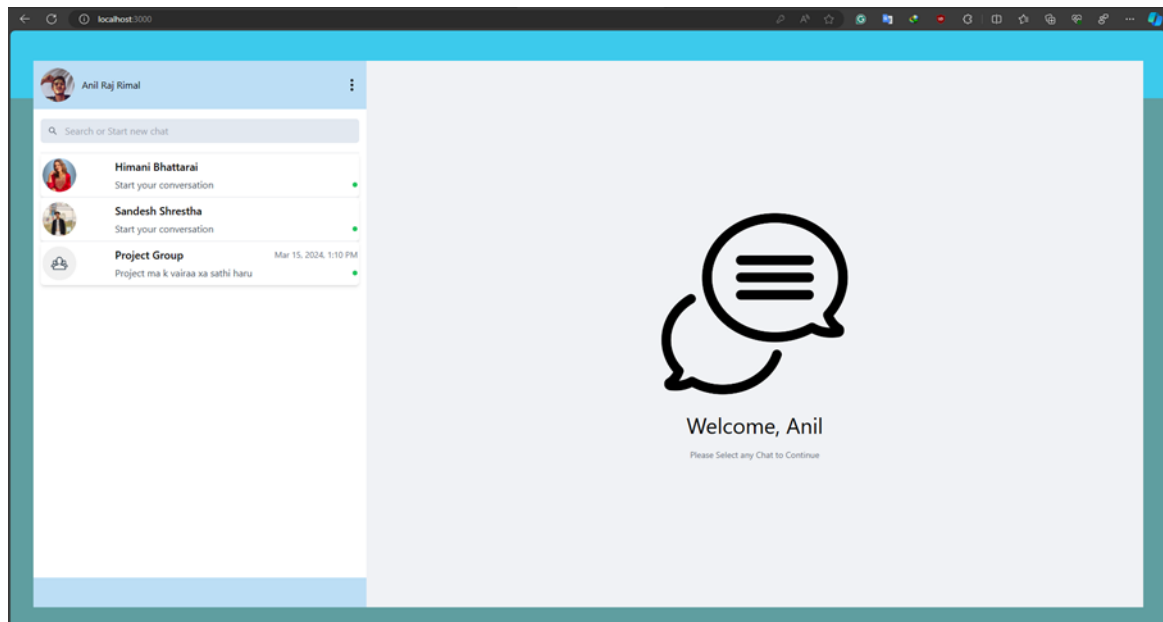


Figure A.3 Chat Homepage

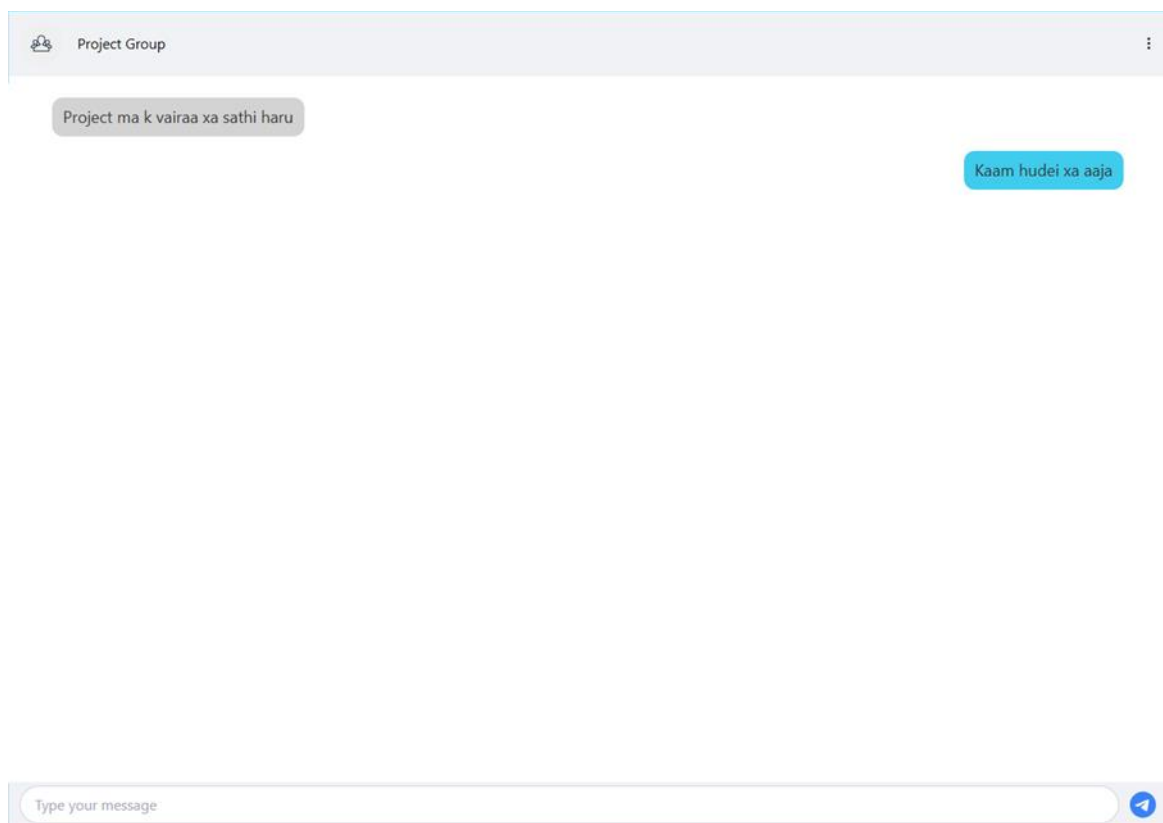


Figure A.4 Group Chat

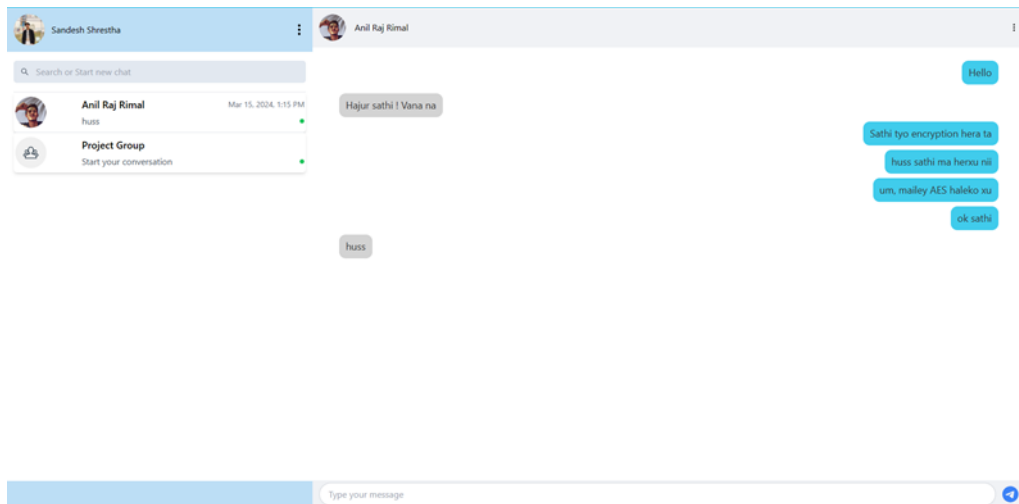


Figure A.5 One-to-one chat

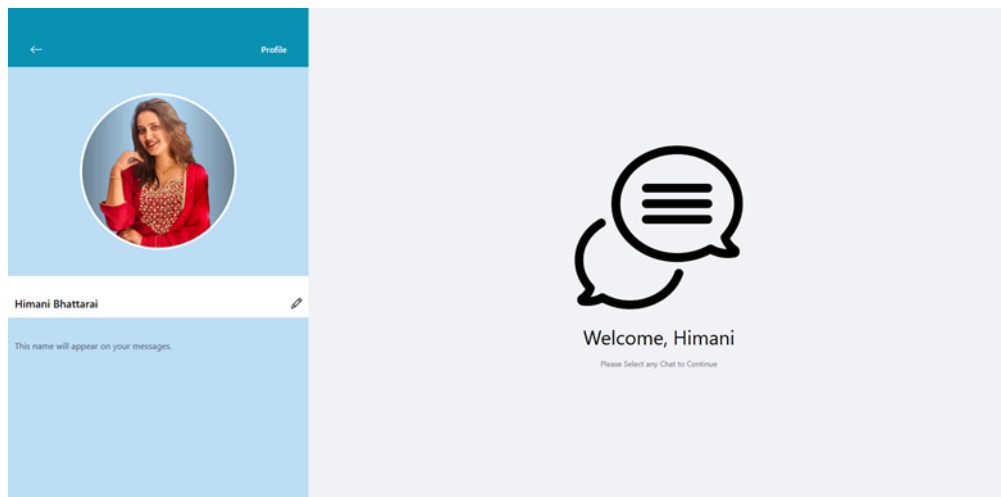


Figure A.6 Profile Page

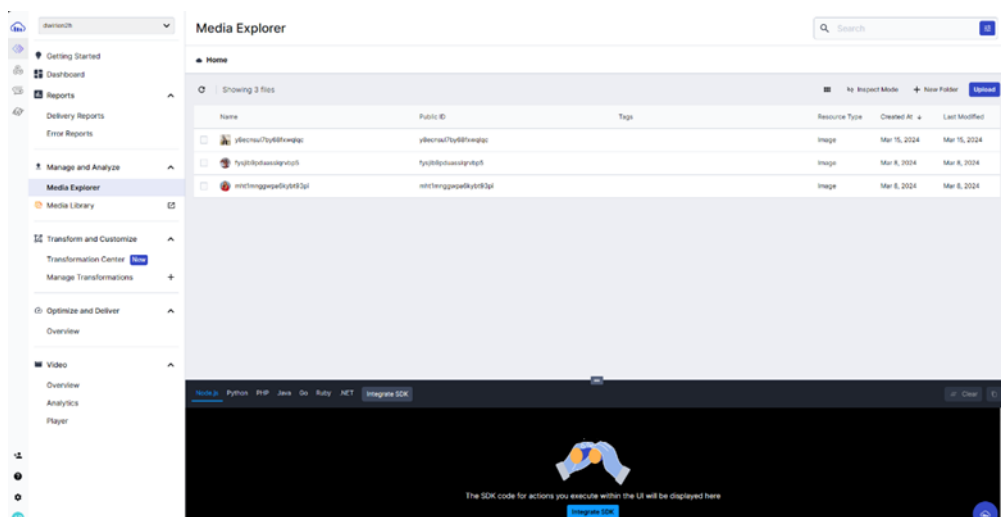


Figure A.7 Cloudinary Image Dashboard

SOURCE CODE

```
package com.chatapp.config;

import jakarta.persistence.AttributeConverter;
import lombok.SneakyThrows;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.util.SerializationUtils;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;
import java.security.GeneralSecurityException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;

2 usages
@Configuration
public class AesEncryptor implements AttributeConverter<Object,String> {
    1 usage
    @Value("${aes.encryption.key}")
    private String encryptionKey;
    2 usages
    private final String encryptionCipher="AES";
    3 usages
    private Key key;
    3 usages
    private Cipher cipher;

    1 usage
    private Key getKey() {
        if (key==null)
            key=new SecretKeySpec(encryptionKey.getBytes(),encryptionCipher);

        return key;
    }

    3 usages
    private Cipher getCipher() throws GeneralSecurityException {
        if (cipher==null)
            cipher=Cipher.getInstance(encryptionCipher);
        return cipher;
    }

    2 usages
    private void initCipher(int encryptMode) throws GeneralSecurityException {
        getCipher().init(encryptMode,getKey());
    }

    2 usages
    @SneakyThrows
    @Override
    public String convertToDatabaseColumn(Object attribute) {
        if (attribute==null)
            return null;
        initCipher(Cipher.ENCRYPT_MODE);
        byte[] bytes= SerializationUtils.serialize(attribute);
        return Base64.getEncoder().encodeToString(getCipher().doFinal(bytes));
    }

    2 usages
    @SneakyThrows
    @Override
    public Object convertToEntityAttribute(String dbData) {
        if (dbData==null)
            return null;
        initCipher(Cipher.DECRYPT_MODE);
        byte[] bytes=getCipher().doFinal(Base64.getDecoder().decode(dbData));
        return SerializationUtils.deserialize(bytes);
    }
}
```

Figure A.8 Algorithm Config code

```

server.port = 8080
# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/chatapp_db
spring.datasource.username=root
spring.datasource.password=admin123

# Hibernate Properties
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update

# Logging SQL Statements
#spring.jpa.show-sql=true

#Encryption configuration key
aes.encryption.key=this-is-test-key

```

Figure A.9 Application.properties

```

const uploadToCloudinary = (pics) => {
  const data = new FormData();
  data.append("file", pics);
  data.append("upload_preset", "vz082orr");
  data.append("cloud_name", "dwirien2h");

  fetch("https://api.cloudinary.com/v1_1/dwirien2h/image/upload", {
    method: "POST",
    body: data,
  })
  .then((res) => res.json())
  .then((data) => {
    console.log(data);
    if (data) {
      setTempPicture(data.url.toString());

      const data2 = {
        id: auth.reqUser.id,
        token: localStorage.getItem("token"),
        data: { profilePicture: data.url.toString() },
      };
      dispatch(updateUser(data2));
    } else {
      console.error("Error: URL not found in response data");
    }
  })
  .catch((error) => {
    console.error("Error uploading to Cloudinary:", error);
  });
};

```

Figure A.10 Cloudinary Setup

```

// Function to establish a WebSocket connection
const connect = () => {
  const socket = new SockJS("http://localhost:8080/ws");
  const temp = over(socket);
  setStompClient(temp);

  const headers = {
    Authorization: `Bearer ${token}`,
    "X-XSRF-TOKEN": getCookie("XSRF-TOKEN")
  };
  // Connect to WebSocket server
  temp.connect(headers, onConnect, onError);
};

// Function to get a specific cookie by name
function getCookie(name) {
  const value = `; ${document.cookie}`;
  const parts = value.split(`; ${name}=`);
  if (parts.length === 2) {
    return parts.pop().split(";").shift();
  }
}

// Callback for WebSocket connection error
const onError = (error) => {
  console.log("WebSocket Connection Error: ", error);
};

// Callback for successful WebSocket connection
const onConnect = () => {
  console.log("WebSocket Connected ");
  setIsConnected(true);
};

// Effect to handle incoming new messages from WebSocket
useEffect(() => {
  if (message.newMessage && stompClient) {
    console.log("Sending message:", message.newMessage);
    stompClient.send("/app/message", {}, JSON.stringify(message.newMessage));
  }
}, [message.newMessage]);

// Effect to set the messages state from the store
useEffect(() => {
  if (message.messages) {
    setMessages(message.messages);
  }
}, [message.messages]);

// Callback to handle received messages from WebSocket
const onMessageReceiver = (payload) => {
  console.log("Received Message:", JSON.parse(payload.body));

  const receivedMessage = JSON.parse(payload.body);
  setMessages((messages) => [...messages, receivedMessage]);
};

// Effect to subscribe to a group chat when connected to WebSocket
useEffect(() => {
  if (isConnected && stompClient && auth.reqUser && currentChat) {
    const subscription = stompClient.subscribe(
      "/group" + currentChat.id.toString(), onMessageReceiver
    );

    return () => {
      subscription.unsubscribe();
    };
  }
}, [isConnected, stompClient, auth.reqUser, currentChat, onMessageReceiver]);

// Effect to establish a WebSocket connection
useEffect(() => {
  connect();
}, []);

useEffect(() => {
  setMessages(message.messages)
}, [message.messages]);

```

Figure A.11 React WebSocket Configuration

```

package com.chatapp.controllers;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.handler.annotation.Payload;
import org.springframework.messaging.handler.annotation.SendTo;
import org.springframework.messaging.simp.SimpMessagingTemplate;

import com.chatapp.model.Message;

no usages
public class RealtimeChatController {

    1 usage
    @Autowired
    private SimpMessagingTemplate simpMessagingTemplate;

    no usages
    @MessageMapping("/message")
    @SendTo("/group/public")
    public Message receiveMessage(@Payload Message message) {
        simpMessagingTemplate.convertAndSend( destination: "/group/" + message.getChat().getId().toString(), message);
        return message;
    }
}

```

Figure A.12 Realtime Chat Controller