A SYNOPSIS ON

CHAT APPLICATION PROJECT

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Utkarsh Joshi	**2261582**
Harshita Joshi	**2261258**
Ravi Mahar	**2261464**
Vikash Bhaisora	**2261605**

Under the Guidance of

Prince Kumar

Assistant Professor

Project Team ID: 95

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **"Chat Application Project"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Prince Kumar**, **Assistant Professor**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Utkarsh Joshi	**2261582**
Harshita Joshi	**2261258**
Ravi Mahar	**2261464**
Vikash Bhaisora	**2261605**

The above-mentioned students shall be working under the supervision of the undersigned on the **"Chat Application Project".**

Signature

Supervisor

Signature

Head of the Department

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members: with Date **Signature**

1.

2.

Table of Contents

Chapter 1
Introduction and Problem Statement

1.1 Introduction

In the era of digital transformation, real-time communication has become an integral part of both personal and professional life. The ability to send messages instantly, share files, and collaborate efficiently has led to the widespread adoption of messaging applications. Traditional communication methods, such as emails or SMS, lack the immediacy and convenience required in today's fast-paced environment. Popular applications like WhatsApp, Slack, and Microsoft Teams have set high standards for real-time messaging. However, most existing solutions operate as closed-source platforms, limiting customization and data privacy control. Organizations and communities looking for a scalable, secure, and customizable messaging system often face challenges in adapting these proprietary solutions to their needs.

A modern chat application should fulfil the following essential criteria:

- Real-time messaging
- Group communication
- Secure communication
- Cross-platform compatibility

With these requirements in mind, this project aims to develop a secure, scalable, and real-time chat application that can be used for personal and professional communication.

1.2 Problem Statement

The current landscape of instant messaging is dominated by proprietary applications, which often impose limitations in terms of data control, customization, and scalability. There is a growing need for an open-source or custom-built chat application that offers flexibility while ensuring security and real-time performance.

Challenges:

- Security Issues
- Scalability Concerns
- Limited Customization
- Integration Barriers
- High Cost
-

Goal:

To develop a real-time chat application with end-to-end encryption, scalability, intuitive UI, multimedia sharing, and cross-platform support.

Chapter 2
Background/ Literature Survey

2.1 Background

The evolution of messaging systems has been marked by significant technological milestones, starting from SMS and email to more advanced instant messaging services such as Yahoo Messenger, Google Talk, and more recently, Slack, Telegram, and Signal. The focus has gradually shifted from simple message delivery to real-time, secure, and integrated communication platforms.

As messaging applications became more central to communication workflows, researchers and developers have explored a variety of approaches to enhance their efficiency, scalability, and security. Key technologies like Web Sockets and MQTT have revolutionized real-time communication, while secure protocols like OAuth2, JWT, and AES/RSA encryption have strengthened user authentication and data protection.

2.2 Core Technologies and Tools

- Web Sockets: Enables full-duplex communication channels over a single TCP connection for real-time updates.
- Socket.IO: A JavaScript library built on Web Sockets, simplifying implementation for real-time apps.
- MQTT: Lightweight messaging protocol used for small devices and high-latency networks.
- Django + PostgreSQL + Redis: Backend stack commonly used for rapid development and performance optimization.
- Security Technologies: End-to-End Encryption (E2EE), JSON Web Tokens (JWT) for user authentication, OAuth 2.0 for secure access delegation.

2.3 Research

- Wang et al. (2020): Used message queues for concurrency
- Gupta et al. (2019): Cloud scaling strategies
- Miller et al. (2021): AES-256/RSA encryption
- Kumar et al. (2020): Multi-layer authentication

2.4 Comparison

Feature	WhatsApp	Slack	Teams	Proposed
Real-Time	☑	☑	☑	☑
Encryption	☑	✗	☑	☑
Group Chat	☑	☑	☑	☑
Open Source	✗	✗	✗	☑
Customizable	✗	✗	✗	☑

2.5 Conclusion

While existing tools provide a robust set of features, their closed-source nature, limited customizability, and privacy trade-offs make them unsuitable for many niche applications and privacy-conscious environments. The proposed solution aims to bridge this gap by offering a fully customizable, open-source, and secure real-time chat application, suitable for deployment in a variety of industries and communities.

Chapter 3
Objectives

The primary objective of this project is to develop a secure, scalable, and user-friendly Chat Application Tool that allows users to communicate in real time. In today's digital age, communication through chat applications has become a necessity. Applications like WhatsApp, Telegram, Slack, and Microsoft Teams have revolutionized the way we interact. However, the challenge lies in ensuring realtime performance, privacy, and a smooth user experience. With increasing concerns around security and data privacy, the objective is to design an application that not only meets these communication needs but also provides robust encryption mechanisms and modern deployment strategies.

The objectives of the proposed work are as follows:

1. To develop a secure real-time chat application using modern web technologies The aim is to leverage modern frameworks and protocols to support real-time messaging, while maintaining an efficient and responsive interface.

2. To implement end-to-end encryption for ensuring data privacy Communication between users will be encrypted using secure algorithms such as AES for content encryption and RSA for secure key exchange, ensuring only the intended recipient can read the message.

3. To provide functionalities for group messaging and media sharing Users should be able to create groups, add members, and exchange text and multimedia content such as images and files in real time.

4. To ensure the application is scalable and deployable on cloud platforms The application architecture will support horizontal scaling and can be easily containerized and deployed on cloud platforms such as Heroku, AWS, or Digital Ocean.

5. To create a user-friendly interface for seamless communication experience The front-end should be intuitive, responsive, and accessible, ensuring a smooth user experience across desktop and mobile browsers.

Chapter 4
Hardware and Software Requirements

This section outlines the hardware and software requirements necessary for the development, testing, and deployment of the chat application.

4.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	Processor	Intel i5 or higher
2	RAM	8 GB minimum
3	Storage	256 GB SSD or higher
4	Network	Stable Internet Connection

A mid-range development machine or virtual machine setup will be sufficient for running and testing the server and client-side application.Deployment would require cloud infrastructure supporting Docker and web services.

4.2 Software Requirements

Sl. No	Name of the Software & Specification
1	Operating System: Windows 10 / Ubuntu Linux / macOS
2	Programming Language: Python 3.8+
3	Web Framework: Django 4.x, Django Channels
4	Real-time: Library Socket.IO
5	Frontend Stack: HTML5, CSS3, JavaScript, React (optional)
6	Database: PostgreSQL
7	Deployment Tools: Docker, Gunicorn, Nginx, Heroku
8	Version Control: Git / GitHub
9	Package Manager: pip, npm

The choice of Django as a backend framework is based on its scalability, security features, and extensive community support.

Socket.IO facilitates real-time bidirectional communication between the client and server using Web Sockets.

PostgreSQL ensures robust and reliable data storage. The application is packaged and containerized using Docker for cross-platform deployment.

Chapter 5

Possible Approach/ Algorithms

The development of a real-time chat application involves multiple subsystems, each responsible for specific tasks such as message transmission, encryption, user authentication, file handling, and UI rendering. The approach adopted here ensures modularity, scalability, and security.

5.1 Frontend Interface

The frontend is developed using HTML, CSS, and JavaScript to ensure responsiveness and cross-browser compatibility. React or Vue.js may optionally be integrated for dynamic component rendering. The client connects to the backend through WebSocket protocol using the Socket.IO client library.

- User interface includes:
- Registration/Login Forms
- Chat Room Interface
- Media Upload Option
- Message Threads with Timestamps

5.2 Backend Server

The backend is implemented using Django REST Framework for core logic and Django Channels to handle asynchronous WebSocket communication. The backend is responsible for:

- Authenticating users using JWT tokens or session-based auth.
- Managing chat room creation and user participation.
- Handling real-time messages and multimedia.
- Managing database queries efficiently.
- Logging and error tracking.

5.3 Real-Time Communication Protocol

Communication is enabled through WebSockets, which allow full-duplex communication. Django Channels integrates WebSocket functionality into Django.

- Rooms/Namespaces: Used to manage group chats.
- Events: Defined to handle different types of messages.
- Socket.IO manages event-driven message flow.

5.4 End-to-End Encryption

To ensure secure message transmission:

- AES (Advanced Encryption Standard) is used for symmetric encryption of message content.
- RSA is used for exchanging AES keys securely between users.
- Django middleware is extended to integrate cryptographic libraries like PyCryptodome or Fernet.

Sample Encryption Workflow:

1. Client encrypts message with AES.
2. AES key is encrypted using recipient's RSA public key.
3. Both encrypted message and key are sent to server.
4. Recipient decrypts AES key using private RSA key and then decrypts message.

5.5 Pseudo Code for Message Broadcast

```
Function sendMessage(user, message):
  encryptedMessage = encrypt(message, user.publicKey)
  for recipient in chatGroup:
    deliverToSocket(recipient.socketID, encryptedMessage)
```

References

[1] Wang et al., “Scalable chat systems using microservices,” *Journal of Web Applications*, 2020.

[2] Gupta et al., “Cloud-native communication platforms,” *IEEE Conf. on Cloud Computing*, 2019.

[3] Miller et al., “Securing chat apps using AES and RSA,” *International Journal of Cybersecurity*, 2021.

[4] Kumar et al., “Authentication in messaging apps,” *ACM Symposium on Applied Computing*, 2020.

[5] Django Channels Documentation. [Online]. Available: https://channels.readthedocs.io

[6] Socket.IO Documentation. [Online]. Available: https://socket.io/docs