SIMPLE CHAT APPLICATION USING PYTHON

Prof.Shinde S.P[1,] Shrutika Kale[2], Shrutika Bansode[3], Vaishnavi Jadhav[4], Siddhi Kanhe[5]
Professor .Sau. Sundarbai. Manik. Adsul. Polytechnic, Chas , Ahilyanagar, India[1]
Students .Sau. Sundarbai. Manik. Adsul. Polytechni , Chas, Ahilyanagar, India[2,3,4,5]

Abstract—Real-time communication systems have become an essential part of modern web applications. This paper describes the development of a Simple Chat Application using Python. The application follows a client–server architecture and enables instant message exchange through Flask and Socket.IO. User authentication and chat history storage are handled using SQLite. The system is lightweight, responsive, and suitable for academic and small-scale communication environments.

Keywords—Chat Application, Python, Flask, Socket.IO, SQLite, Client–Server Architecture

I. INTRODUCTION

Instant messaging applications enable fast and reliable communication over the internet. Understanding how such systems work is important for computer engineering students. This project focuses on the design and implementation of a simple chat application that supports real-time communication using web technologies.

II. OBJECTIVES

The objectives of the proposed system are:
• To understand client–server communication
• To implement real-time messaging
• To design a secure login system
• To store chat messages in a database

III. SYSTEM REQUIREMENTS

Hardware Requirements:
• Computer or Laptop
• Internet Connection

Software Requirements:
• Python Programming Language
• Flask Framework
• Flask-SocketIO
• SQLite Database
• PyCharm IDE

IV. PROPOSED SYSTEM

The proposed chat application is based on a client–server model. Users interact with the system through a web interface. The server manages user authentication and message routing, while the database stores user information and chat history.

V. SYSTEM ARCHITECTURE

The architecture of the system consists of three major components:

1. Client Layer: Developed using HTML, CSS, and JavaScript. It provides the user interface for login and chat.

2. Application Server: Implemented using Flask. It handles user authentication, session management, and communication events.

3. Database Layer: SQLite database stores user credentials and message history. This architecture ensures secure data handling and efficient communication.

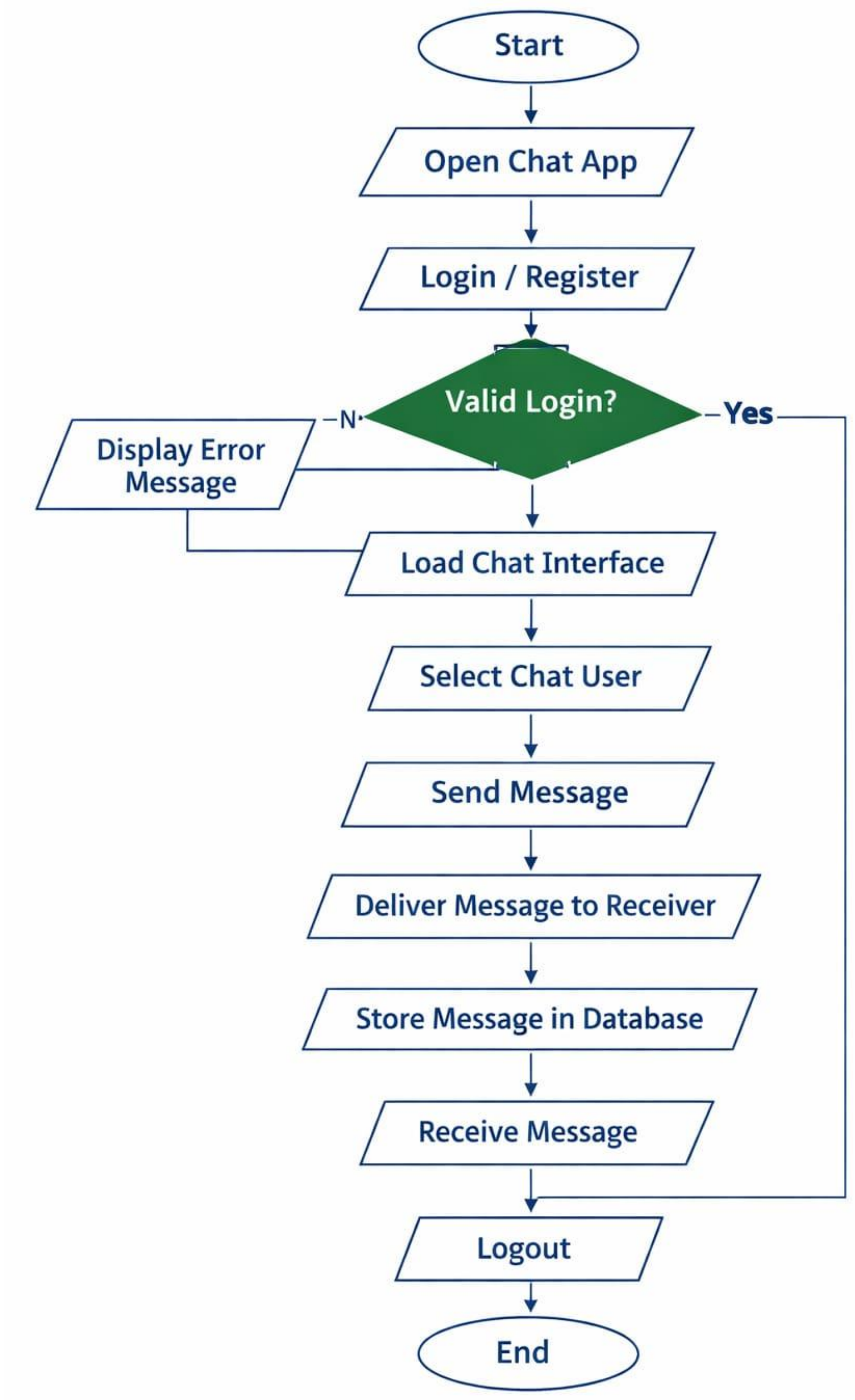
VI. IMPLEMENTATION

The chat application is implemented using Python. Flask handles routing and user sessions, while Socket.IO enables real-time bidirectional communication. SQLite is used as a lightweight database to store messages and user details.

VII. FLOWCHART DESCRIPTION

The flowchart explains the working of the Simple Chat Application step by step:

1. Start the application
2. User registers or logs in
3. Server validates user credentials
4. If valid, chat interface is loaded
5. User sends and receives messages
6. Messages are stored in the database
7. User logs out
8. End



VIII. RESULTS AND DISCUSSION

The application was tested with multiple users. Messages were delivered instantly without page refresh. Chat history was successfully stored and retrieved after user login.

IX. FUTURE SCOPE

The system can be enhanced by adding group chat, file sharing, message read receipts, and end-to-end encryption. A mobile application version can also be developed.

X. CONCLUSION

This paper presented a simple and efficient chat application developed using Python. The system demonstrates real-time communication, database integration, and client–server architecture. It is suitable for academic learning and can be extended for advanced use cases.

XI. REFERENCES

[1] Flask Documentation, https://flask.palletsprojects.com
[2] Flask-SocketIO Documentation
[3] SQLite Documentation
[4] Python Official Documentation