Chat Application using FTP-enabled system

Submitted By
Muhammad Hasnain
22K-5099

Ali Masood
22K-5127
.
.
Syed Shees Ali
22K-5047

Department of Computer Science
National University of Computer & Emerging Sciences

1. Motivation

The motivation behind this project stems from the need for a lightweight, secure, and efficient communication tool that leverages the File Transfer Protocol (FTP) for messaging. Traditional chat applications rely on centralized servers, which can be prone to downtime, latency, and security vulnerabilities. By using FTP, this project aims to create a decentralized chat app that ensures data privacy, reduces dependency on central servers, and demonstrates the versatility of FTP beyond file transfers. This project also serves as an academic exploration of network protocols and their unconventional applications.

2. Overview

2.1 Significance of the Project

- **Importance:** The project highlights the adaptability of FTP for real-time communication, offering an alternative to conventional chat systems.
- **Practicality:** It provides a proof-of-concept for decentralized messaging, which can be useful in environments with restricted internet access or where server infrastructure is limited.
- **Academic Value:** The project deepens understanding of network protocols, socket programming, and client-server architectures.

2.2 Description of the Project

The chat app enables users to send and receive messages by leveraging FTP for data exchange. Users connect to an FTP server where messages are stored as files, which are then retrieved by recipients. The system includes:

- **User Authentication:** Secure login using FTP credentials.
- **Message Handling:** Messages are written to and read from designated directories on the FTP server.
- **Real-Time Updates:** Polling or notification mechanisms to check for new messages.

2.3 Background of the Project

Research included studying FTP protocols (RFC 959), existing chat applications, and decentralized systems. Tools like Python's **ftplib** and **socket** library were explored for implementation.

2.4 Project Category

This is a **Product-based** project focusing on software development and protocol innovation.

3. Features / Scope / Modules

1. **FTP-Based Messaging**
    - Messages are transferred as files via FTP, ensuring compatibility with any FTP server.
    - Supports text-based communication with minimal latency.
2. **User Authentication**
    - Secure login using FTP server credentials.
    - Isolated directories for each user to maintain privacy.
3. **Decentralized Architecture**
    - No central chat server; relies on existing FTP infrastructure.
    - Reduces single points of failure.
4. **Cross-Platform Compatibility**
    - Works on any OS with FTP support (Windows, Linux, macOS).
5. **Extensible Design**
    - Can be expanded to support file sharing, group chats, or encryption.

4. Project Planning

A Gantt chart outlines the following phases:

- **Phase 1 (2 weeks): Setup & Auth**

  - Set up FTP server and MySQL database.
  - Implement user authentication (signup/login) with password hashing.
  - Design basic UI templates.
- **Phase 2 (3 weeks): Core Features**

  - Implement WebSockets for real-time messaging.
  - Enable file upload/download via FTP.
- **Phase 3 (2 weeks): Advanced Features**

  - Implement multimedia sharing (images/docs).
  - Add message seen status via WebSockets.
- **Phase 4 (1 week): Testing and documentation**

  - Debug and optimize performance.
  - Finalize documentation.

Responsibilities:

Muhammad Hasnain: Functional login system, FTP server config, UI prototypes.

Syed Shees Ali: Real-time chat, WebSocket integration.

Ali Masood: File-sharing module, message seen info, test reports.

5. Project Feasibility

1. **Technical Feasibility:** Achievable with Python and FTP libraries. Risks include latency due to file-based messaging.
2. **Economic Feasibility:** Low cost; uses open-source tools and existing FTP servers.
3. **Schedule Feasibility:** 8-week timeline is realistic for the defined scope.

6. Hardware and Software Requirements

- **Hardware:** Any computer with internet access.
- **Software:** Python 3.x, ftplib, FTP server (e.g., FileZilla Server), IDE (PyCharm/VSCode).

7. Diagrammatic Representation of the Overall System

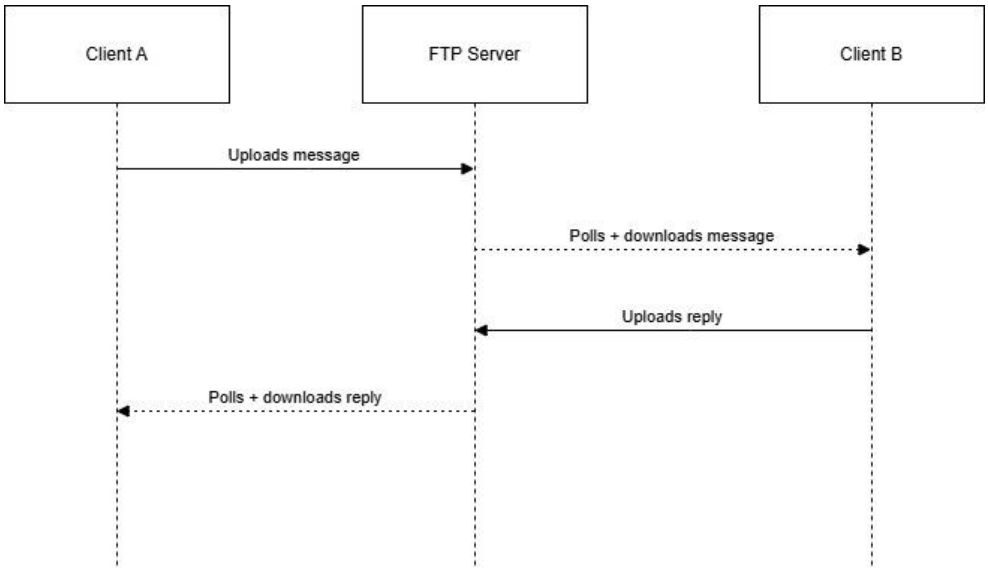


Figure 1. Architecture of the FTP-based chat app.

8. References

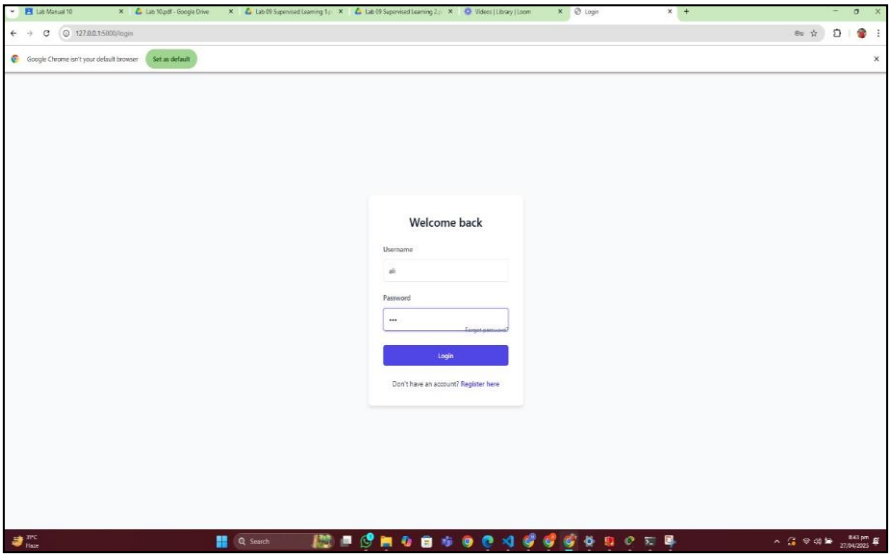
[1] J. Postel, J. Reynolds, "File Transfer Protocol (FTP)", RFC 959, 1985.

[2] Python Software Foundation, "ftplib Documentation", https://docs.python.org/3/library/ftplib.html.

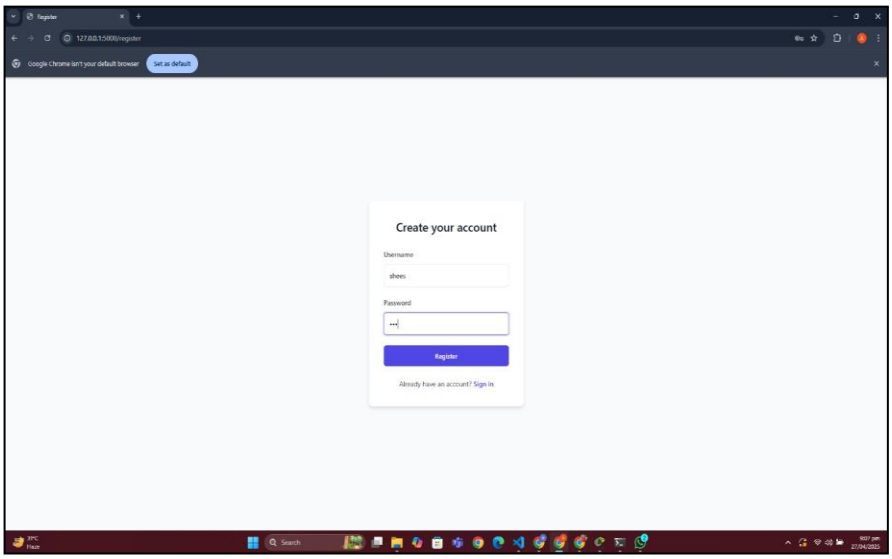
GitHub Repository: https://github.com/muhammadhasnain115/CN-Project-chat-app

9. Project Screen Shots:

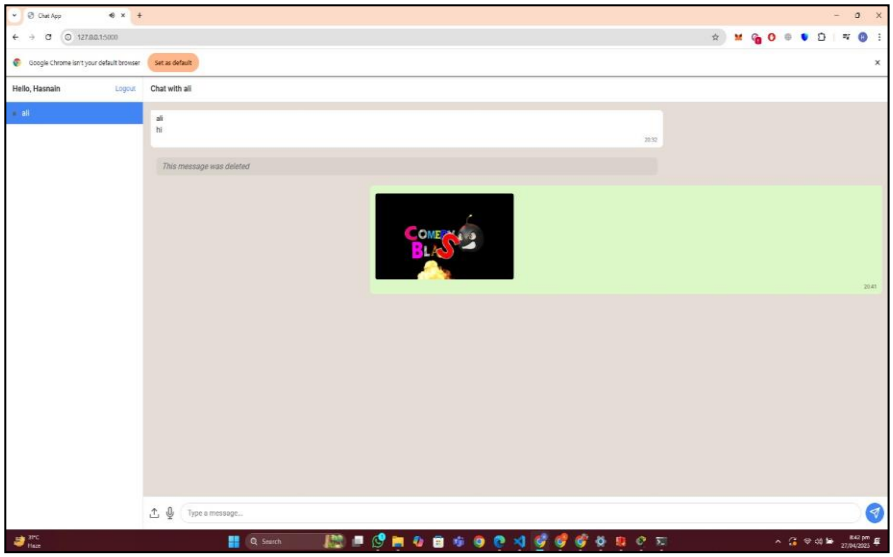
Log-in page



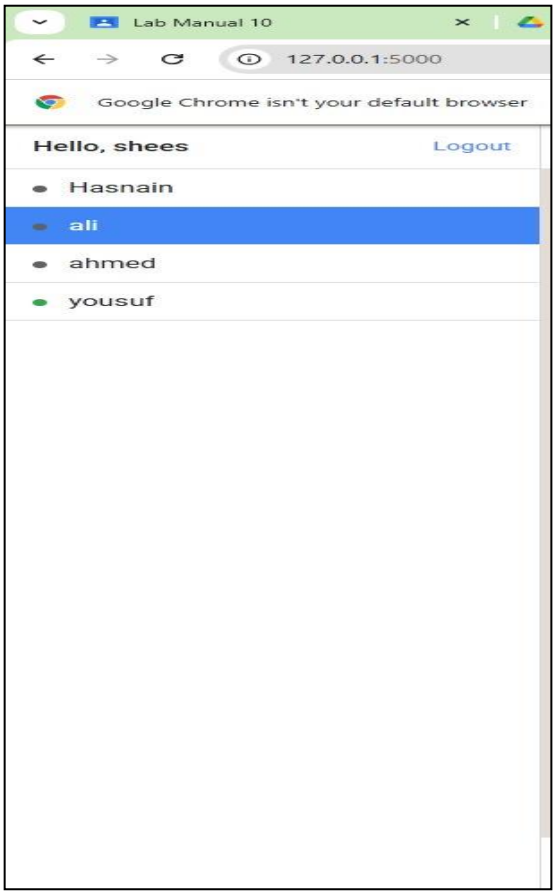
Register page



Chat page



Multiple Users



Project Video Demonstration link :

https://www.loom.com/share/c00c231d67434cd5932e8ff6128cd2ae?sid=a50417a3-b0ec-4e60-b0c3-4594b5498fab