Chat Hub: A Scalable Real-Time Communication Platform

Problem Statement:

Popular real-time chat applications like WhatsApp, Telegram, and Slack provide basic messaging and file sharing but lack features that enhance productivity and control. Users face difficulties with multitasking since only one chat can be active at a time, limited options for organizing or pinning important messages, no restoration of deleted messages, and cluttered interfaces without dedicated spaces for files or pinned content. Moreover, most platforms do not integrate intelligent support such as AI-based replies or content verification. These gaps highlight the need for a more organized, flexible, and intelligent communication platform.

Abstract:

This project presents a next-generation real-time messaging application designed to overcome the limitations of existing chat platforms. Built with a scalable architecture using FastAPI, React, PostgreSQL, and Redis, the system enables seamless one-to-one and group communication with advanced features such as multi-window chat management, unlimited message pinning, message restoration through a trash system, and dedicated side panels for organized access to files and pinned content. Additionally, the platform integrates AI-driven support for auto-replies, summarization, and information verification, enhancing both usability and trust. By combining reliability, productivity-focused tools, and intelligent assistance, the application provides a comprehensive communication solution for students, professionals, and organizations.

Objectives:

- To design and implement a real-time messaging system with one-to-one and group chat support.
- To provide multi-window chat management for efficient multitasking.
- To allow unlimited message pinning and organized side panels for quick access to files and pinned content.
- To implement a message restoration mechanism for deleted messages via a trash system.
- To build a scalable and secure backend using FastAPI, PostgreSQL, and Redis with WebSocket

Proposed System:

The system architecture uses FastAPI for the backend with PostgreSQL for persistent storage and Redis for real-time communication handling. React with Tailwind CSS powers the frontend to deliver a responsive and user-friendly interface. WebSocket technology enables live messaging, typing indicators, and read receipts.

Team:

Chiranth Gowda K C (Captain)
Pradeep pawar
Dinesh
Charan T M