SIMATS ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES,
CHENNAI– 602105

TITLE

# Distributed OS Chat: Developing a Distributed Multi-User Chat System for Operating Systems with Python

CSA0491 - Operating Systems for Process Scheduling

A CAPSTONE PROJECT REPORT

Submitted to

SAVEETHA SCHOOL OF ENGINEERING

By
V. Madhan Reddy
192211759
G. Karthik Reddy
192211751
Y. Abhinay Reddy
192211751
Guided by
DR .G. Mary Valentina

## Abstract:

The importance of sophisticated chat systems has been highlighted by the spread of distributed computing and the requirement for effective, real-time communication. This project presents a Python-based Distributed Multi-User Chat System designed specifically for Operating Systems. This system offers a reliable, scalable, and fault-tolerant platform for real-time communications by utilizing the inherent characteristics of distributed operating systems. By incorporating important distributed systems concepts like concurrency, synchronization, and decentralization, the design effectively handles the challenges presented by multi-user situations.

The solution ensures compatibility and interoperability by facilitating smooth communication across various operating systems through the use of networking libraries and the asynchronous programming characteristics of Python. The architecture optimizes resource utilization among nodes and supports both client-server and peer-to-peer topologies, enabling flexible deployment options.

## Introduction

In an age defined by the interconnectedness of global networks and the pervasive integration of technology into everyday life, the need for efficient and seamless communication platforms has never been more pressing. From remote collaboration to real-time information sharing, the demand for robust, scalable, and user-friendly chat systems spans industries and communities worldwide. However, traditional centralized architectures often struggle to meet the demands of modern distributed environments, where users are dispersed across diverse operating systems and geographic locations. Recognizing these challenges, we introduce Distributed OS Chat-a pioneering solution designed to transcend the limitations of existing chat systems by harnessing the power of distributed computing and the versatility of the Python programming language. Distributed OS Chat is not merely another chat application; it represents a paradigm shift in how we conceive and implement real-time communication in distributed operating environments. By combining cutting-edge distributed computing principles with the simplicity and accessibility of Python, Distributed OS Chat aims to redefine the landscape of multi-user chat systems, offering a scalable, reliable and platform-agnostic solution for the interconnected world of tomorrow.

## Process:

1. Analysis of Requirements
   Determine Your Goals: Specify the chat system's primary features, performance objectives, and user needs.
   Establish Constraints: Recognize your time, money, and technical constraints.

   Examine User Needs: Conduct user research to customize features such as platform compatibility, encryption, and UI/UX.

   System Design Architecture:

2. Select a system architecture based on a client-server, hybrid, or peer-to-peer (P2P) model.
   Protocol Definition: Create error-handling, user authentication, and message-passing communication protocols.
   Describe the data structures that are used to store system logs, chat histories, and user information.
3. Choosing Technology
   Language Used for Programming: Choose Python because of its many network connection libraries (such as socket and Asuncion) and user-friendliness.
   Frameworks & Libraries: Choose the right Python modules for asynchronous

## Objective:

Using Python, a distributed multi-user chat system for operating systems is being developed with the goal of building a scalable, effective, and safe real-time communication platform across distributed contexts. The goal of this system is to satisfy the complex needs of multi-user chat functionality while showcasing the real-world implementation of distributed computing ideas. The project's particular objectives are as follows:

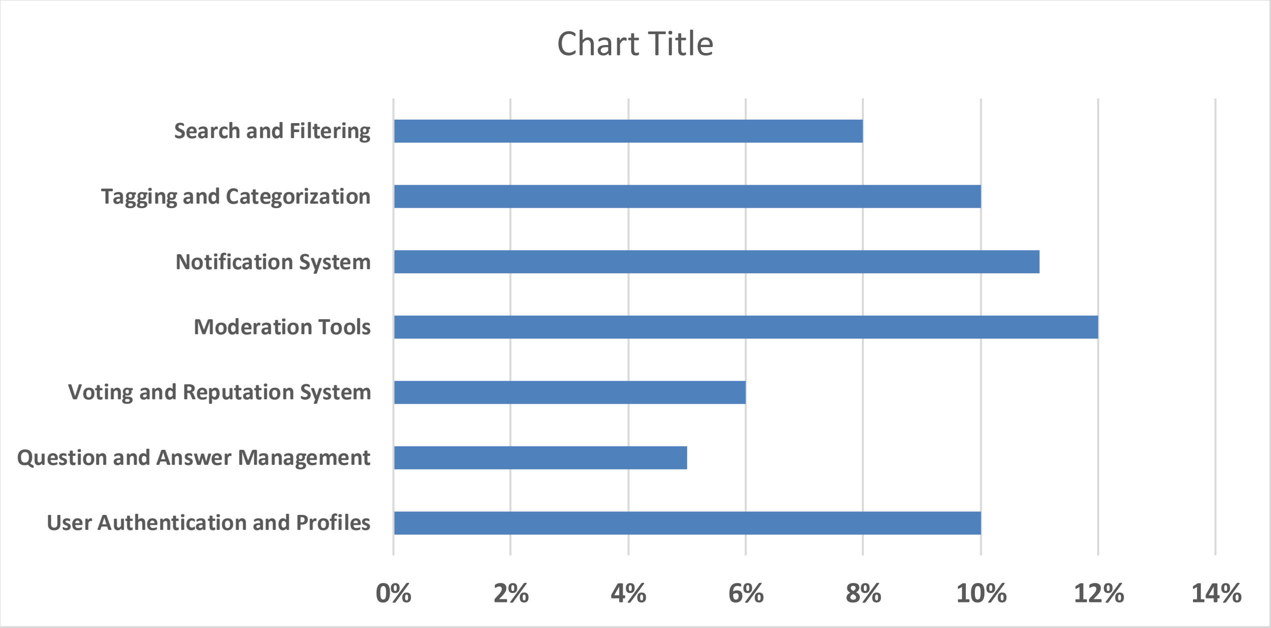
Scalability: Use distributed system architectures to divide load and resources wisely in order to build a chat system that can effectively handle an increasing number of users and concurrent chats without seeing a drop in performance.

Real-Time Communication: Make sure the system enables prompt message delivery so users may have smooth real-time interactions. This is essential to preserving the context and flow of talks.

## Literature Review:

Distributed Systems and networking: Start by exploring foundational literature on distributed systems, including principles of distributed computing, network protocols, and architectures. This might include seminal texts like "Distributed Systems: Principles and Paradigms" by Andrew S. Tanenbaum and Maarten Van Steen. Developing and Implementation of Distributed Chat Applications using WPF and WCF - The development of distributed chat applications involves several tools and technologies. In particular, the researcher describes an application that provides multi chatting between computers on the Internet by Daadoo, Motaz.

## Gantt Chart



## Conclusion:

In summary, utilizing Python to create a Distributed Multi-User Chat System is a big step toward maximizing the potential of distributed computing to improve real-time communication. This project illustrates the practical application of theoretical concepts in distributed systems by addressing important goals including scalability, real-time performance, cross-platform interoperability, security, fault tolerance, and user-friendly design. Python is a great choice for our project because of its adaptability and abundance of libraries, which have made it possible to implement sophisticated features like encryption and asynchronous communication with ease.

The architecture of the system, built for resilience and adaptability, not only satisfies the needs of digital communication today but also establishes the framework for upcoming improvements. By protecting users' information, the emphasis on security and privacy promotes trust in the digital.