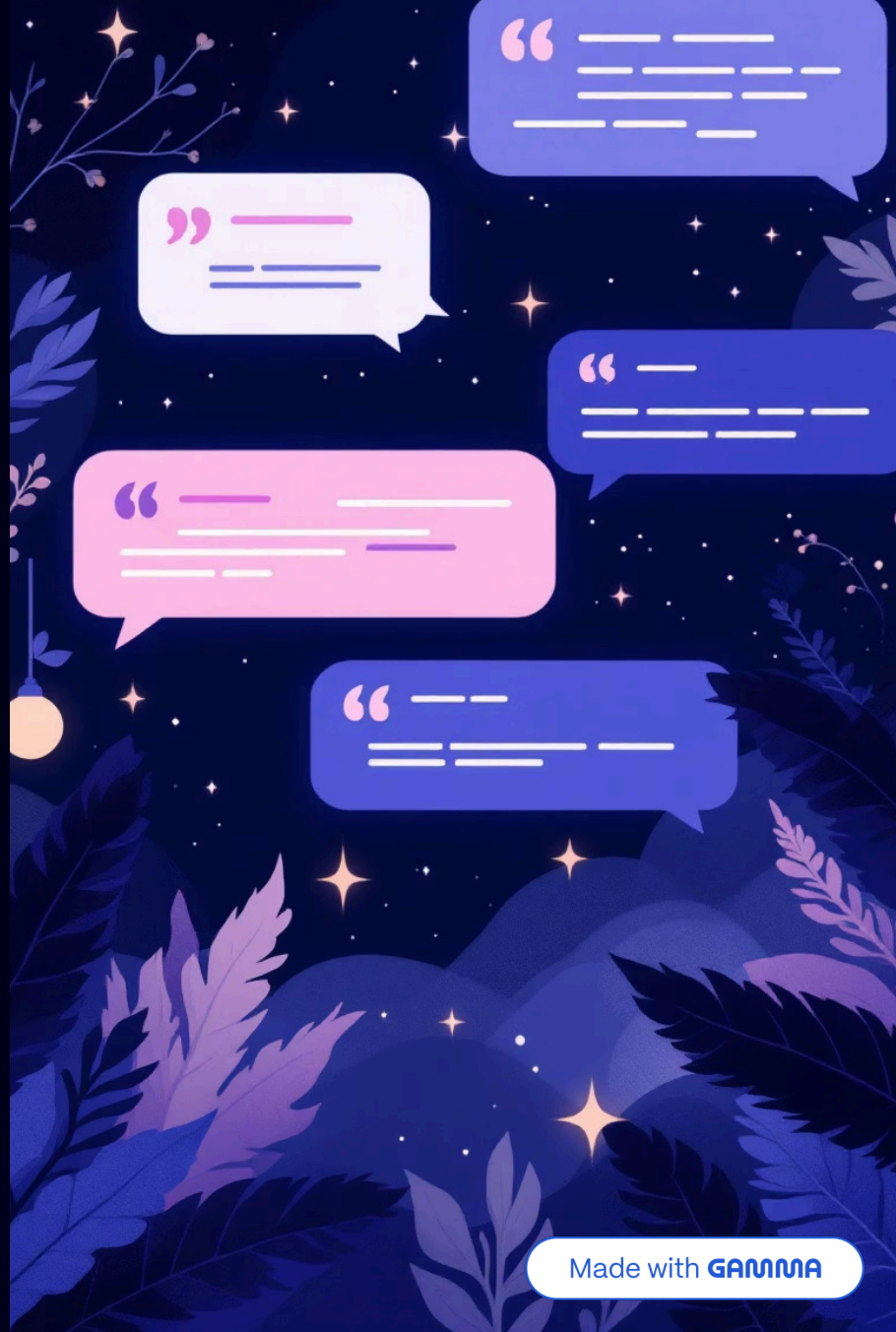
Operating System Course Project

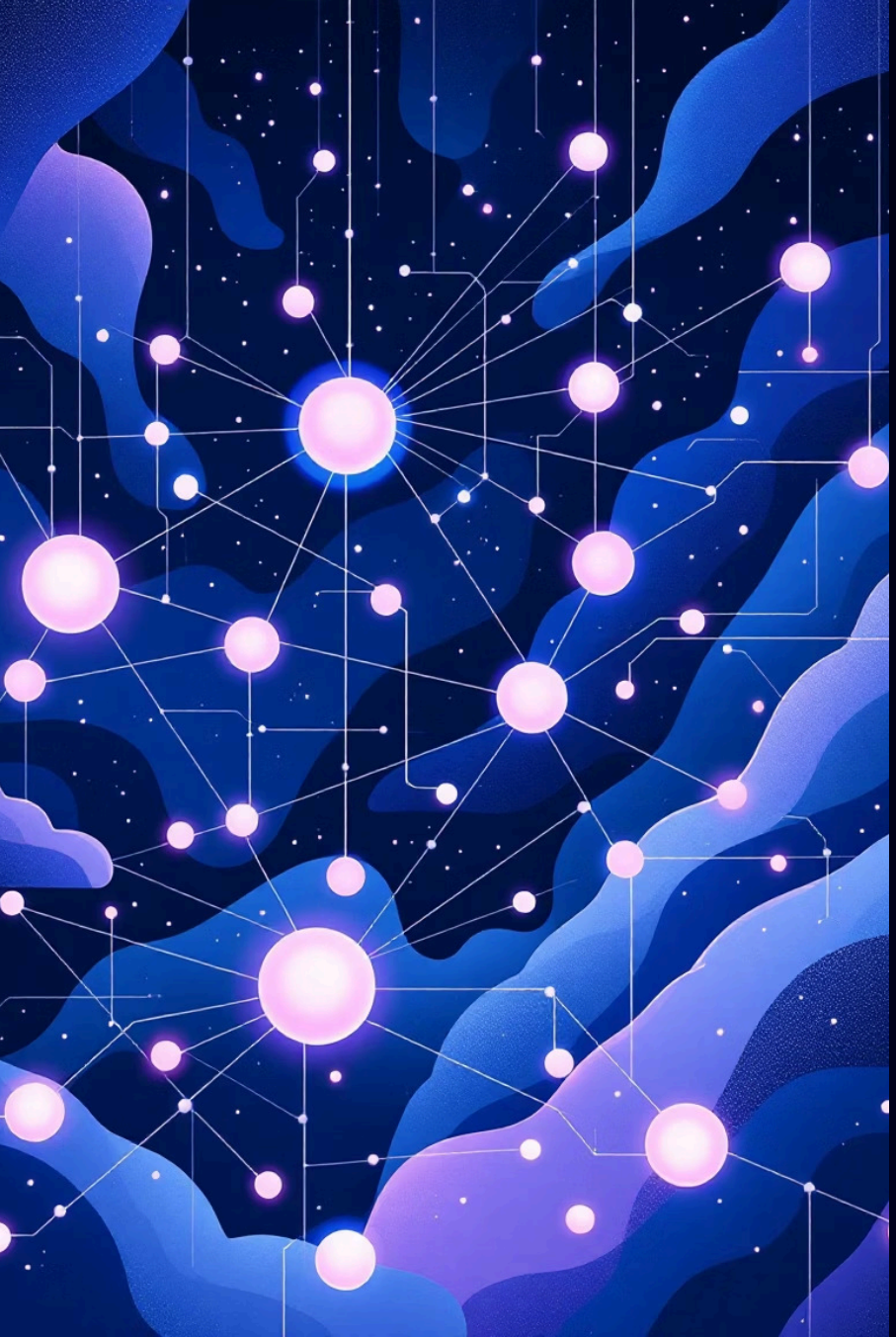
Presented by Bilal Naeem (F23CSC008) & Shahjahan (F23CSC021)

Threaded Chat Application in C

Leveraging Multithreading and Socket Programming

This project demonstrates the practical implementation of multithreading and socket-based interprocess communication in a Linux environment.





Introduction to Core Concepts

Multithreading

Allows multiple tasks to execute concurrently within a single process, enhancing efficiency.

Interprocess Communication (IPC)

Enables processes to exchange data and coordinate actions, crucial for distributed systems.

Project Objectives

Multithreading Mastery

Understand and implement multithreading using the pthread library in C.

Socket Programming

Study TCP-based client-server communication and network sockets.

Real-time Chat System

Build a system supporting multiple simultaneous clients for efficient communication.

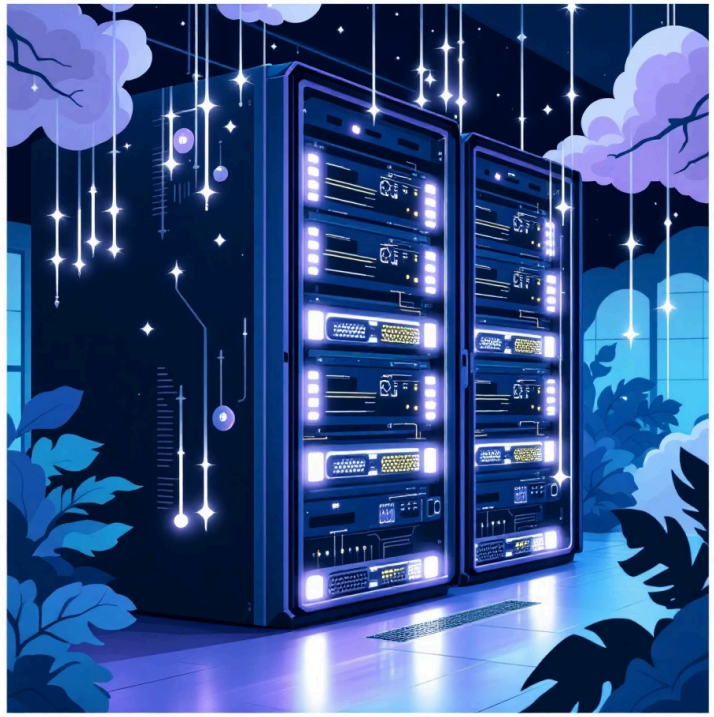
OS Concepts

Demonstrate thread scheduling and concurrent execution management.

Literature Review: Concurrent Servers

Traditional single-threaded servers handle one client at a time, leading to delays and poor performance. Multithreaded servers significantly improve performance, responsiveness, and resource utilization.

Socket programming with TCP/IP is widely adopted for network communication. POSIX threads provide a standardized API for thread creation and synchronization in Unix-like OS.



This project draws inspiration from classical client-server models and modern concurrent programming techniques.

Problem Statement: Single-Threaded Limitations

> In a single-threaded communication system, the server can handle only one client at a time. When multiple clients try to connect simultaneously, the server becomes unresponsive or introduces delays, making real-time communication impossible.

The challenge is to design a system for simultaneous client communication without performance degradation.

Proposed Solution: Multithreaded Architecture

1

Server Setup

Creates TCP socket, binds to port, listens for connections.

2

Client Threads

New thread created for each client using `pthread_create()`.

3

Message Broadcast

Messages from one client broadcast to all others.

This approach ensures concurrent execution, efficient resource utilization, and real-time communication.

Modules & Components

Server Initialization

Socket creation, binding, and listening for connections.

Client Handler

Thread for each client, message reception, disconnection handling.

Client Program

Connects to server, sends/receives messages.

Message Broadcasting

Distributes messages to all clients, uses synchronization.

Expected Outcomes

1 **Simultaneous Connections**

Multiple clients connect and exchange messages in real time.

2 **No Performance Lag**

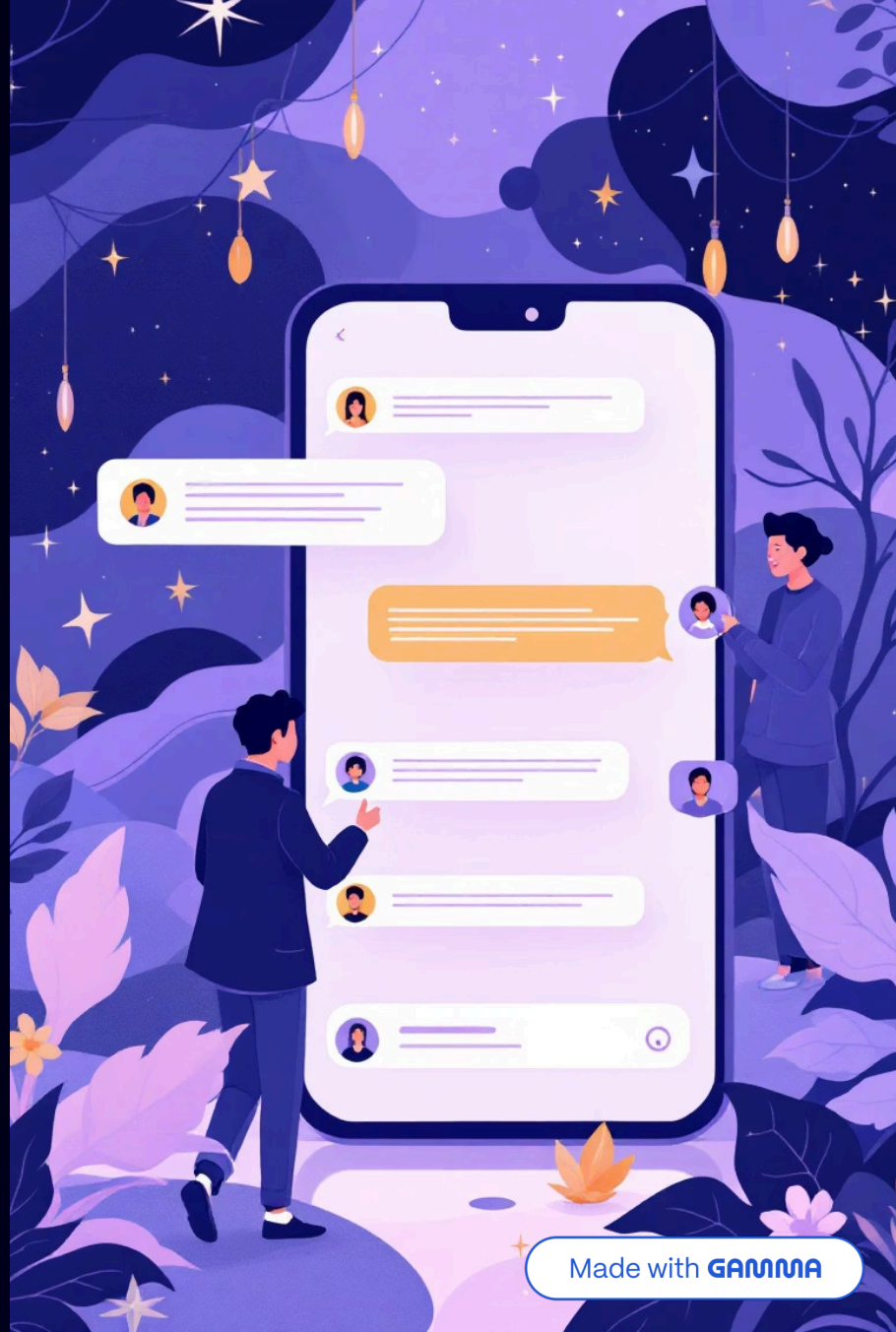
Blocking or delays are eliminated through concurrent thread execution.

3 **Resource Efficiency**

Optimal utilization of system resources.

4 **Concept Demonstration**

Clear illustration of multithreading, thread scheduling, and IPC.



Tools & Technologies

Programming Language

C

Operating System

Linux (Ubuntu / Kali Linux)

Compiler

GCC

Libraries Used

- `pthread.h` (multithreading)
- `sys/socket.h` & `arpa/inet.h` (socket programming)
- `unistd.h` & `string.h` (system calls & string operations)

Development Environment

Linux Terminal / VS Code