System Programming

# Simple Chat Appplication

## Team Members

Ikromjon Sharipov - 220448

Islombek Abdurahmonov - 221236

Fayzulla Asatullaev - 221405

## Objectives

The primary objective of this project is to develop a real-time chat application that enables users to exchange messages instantly over a network. This application will serve as a practical demonstration of client-server architecture, socket programming, and GUI integration using modern web technologies.

Specifically, we aim to:

- Build a functional chat application with a user-friendly Graphical User Interface (GUI).
- Utilize WebSocket technology via Socket.IO for efficient and persistent connections.
- Create a client-side interface using Next.js for a responsive and interactive user experience.
- Apply styling with Tailwind CSS for a visually appealing and maintainable design.

## Features

The initial version of the Simple Chat Application will include the following core features:

- **Real-time Text Messaging:** Users will be able to send and receive text messages instantly.
- **User Input:** A text input area for users to compose messages.
- **Message Display:** A chat window to display the history of messages exchanged in real-time.
- **Username Identification:** Users will be able to identify themselves with a username when connecting to the chat.
- **Basic Connection Management:** Functionality to connect to and disconnect from the chat server.
- **User Presence (optional):** A simple indication of currently online users.

## Technology Stack

This project will be built using the following technologies, chosen for their suitability for real-time web applications and modern UI development:

**Frontend (GUI):**

- **Next.js:** A React framework for building user interfaces. We will use Next.js for its features in creating a dynamic and responsive client-side application.
- **TypeScript:** For writing type-safe and maintainable JavaScript code in the Next.js frontend.
- **Tailwind CSS:** A utility-first CSS framework to rapidly style the user interface components and ensure a consistent design.

**Backend:**

- **Node.js:** The server-side runtime environment for handling WebSocket connections and message broadcasting.

- **Socket.IO:** A JavaScript library for enabling real-time, bidirectional communication between the Next.js frontend and the Node.js backend using WebSockets.

## Development Approach

We will follow an iterative development approach:

**Phase 1: Backend Server Setup:** Implement the Node.js server with Socket.IO to handle basic connection management and message broadcasting.

**Phase 2: Frontend GUI Development:** Build the user interface in Next.js with input and display areas, integrating with the backend using WebSockets.

**Phase 3: Core Chat Functionality:** Implement real-time message sending and receiving, display message history.

**Phase 4: Enhancements and Refinements:** Add username handling, basic styling with Tailwind CSS.

**Phase 5: Testing and Deployment:** Thoroughly test the application and prepare for potential deployment.

**Phase 6: Version Control:** We will follow an iterative development approach, with all code changes continuously tracked and documented using Git version control.

## Potential Deployment

While the primary focus is on developing the core functionality, we will explore options for deploying the client application to make it accessible online. Potential deployment platform include:

**Vercel:** Vercel is a popular platform specifically designed for deploying Next.js applications. It offers easy deployment, automatic scaling, and serverless functions, which could be suitable for hosting the Next.js frontend of our chat application.

## Project Repository:

https://github.com/sharipovikromjon/chat-app-in-nextjs