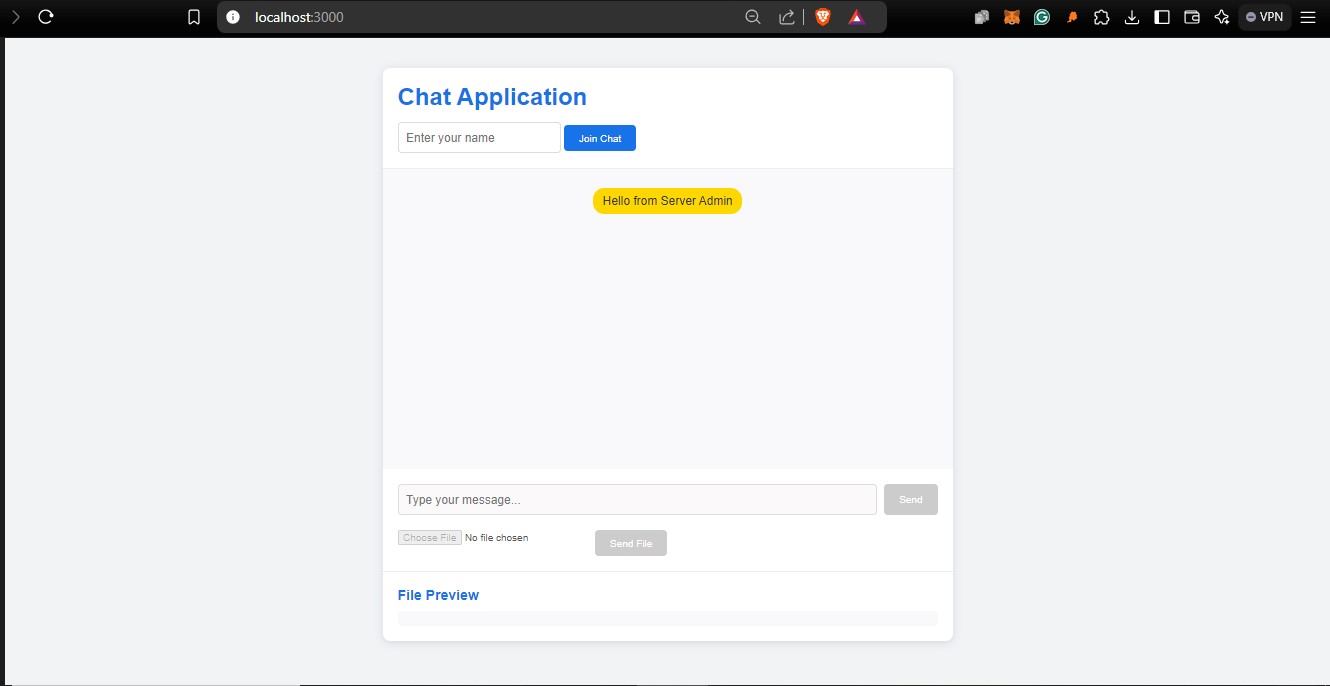
**NA PROJECT - REPORT  
 Anirudh Tallavajhula-16354364**

**Real-Time Chat Application with File Sharing Capability**

# 1. Introduction

This report details the development of a real-time chat application using Node.js, Socket.IO, and a web-based frontend. The application enables multiple users to communicate in real time and share files while implementing specific protocol requirements for client-server communication.



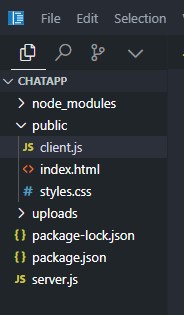
*Figure 1: Main Chat Application Interface*

## 2. System Architecture

### 2.1 Technology Stack

* Frontend: HTML5, CSS3, JavaScript
* Backend: Node.js, Express.js
* Real-time Communication: Socket.IO
* File System: Node.js fs module

### 2.2 System Components



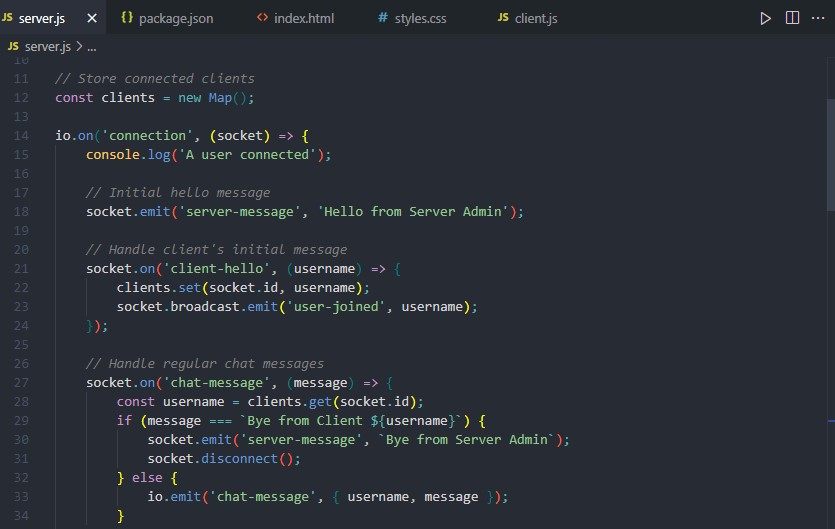
## 3. Implementation Details

### 3.1 Server-Side Implementation

The server implementation handles:

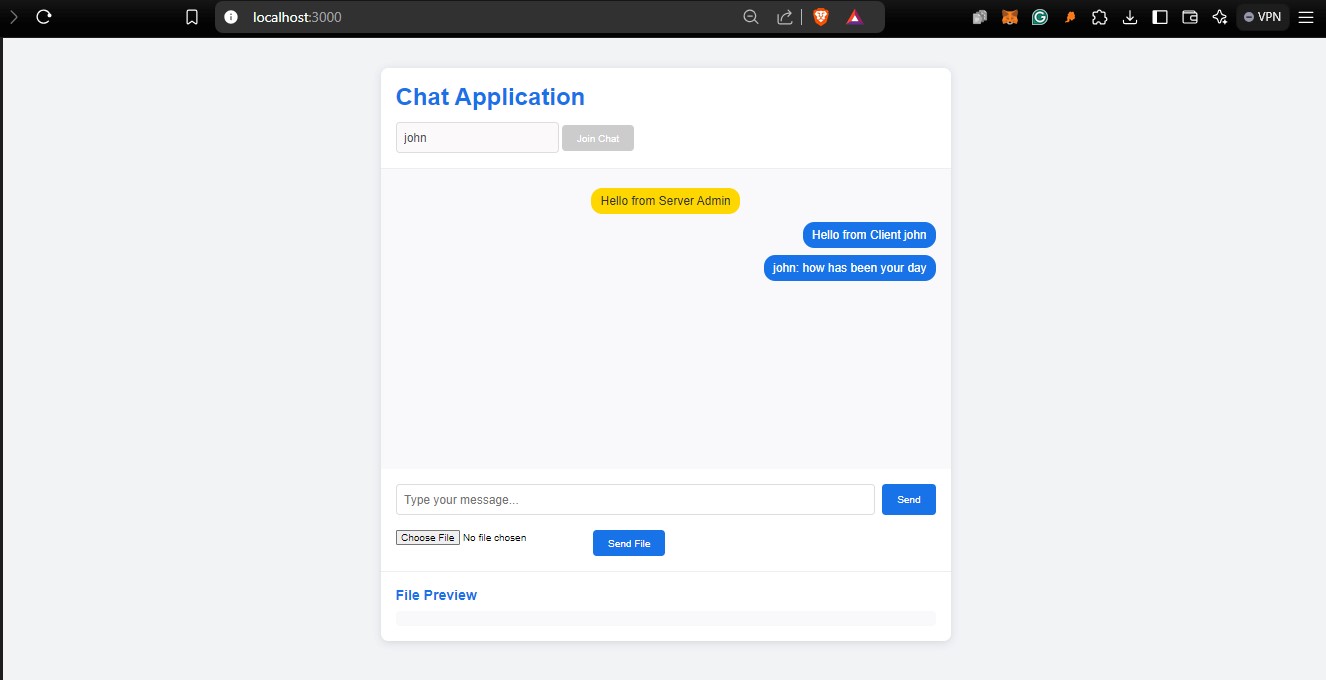
* Socket connection management
* Multiple client support
* File processing
* Message broadcasting

Key code snippet from server.js:



**3.2 Client-Side Implementation** The client-side features:

1. Real-time message handling
2. File upload interface
3. Dynamic UI updates
4. Connection status management

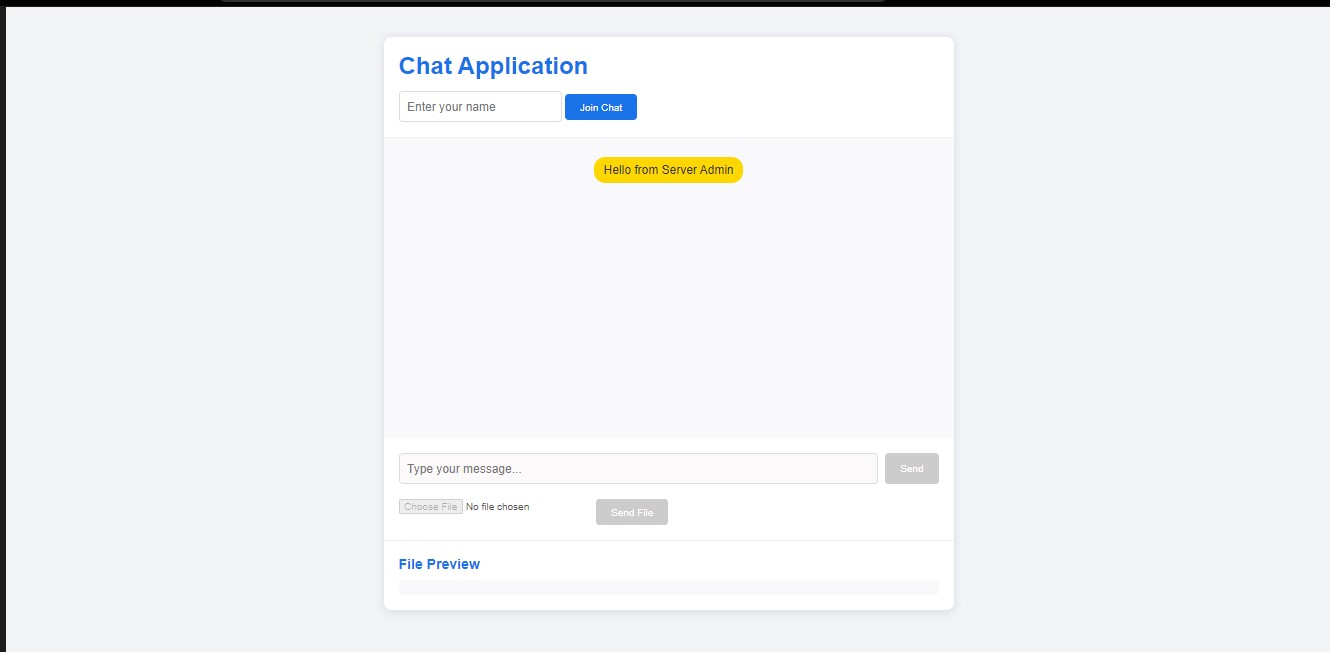


*Figure 4: Active Chat Session*

## 4. Features and Functionality

### 4.1 Chat Functionality

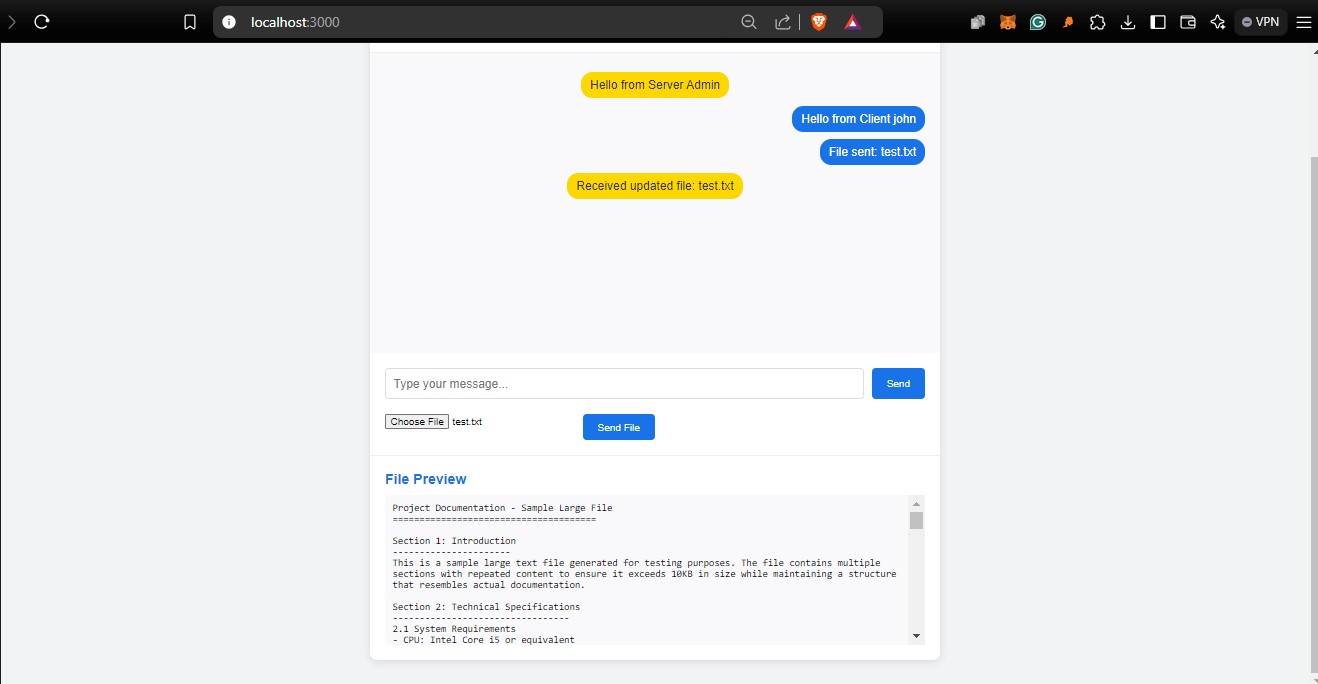
1. Initial handshake protocol implementation
2. Real-time message delivery
3. User join/leave notifications
4. Proper disconnection handling



*Figure 5: Chat Protocol Implementation*

### 4.2 File Sharing

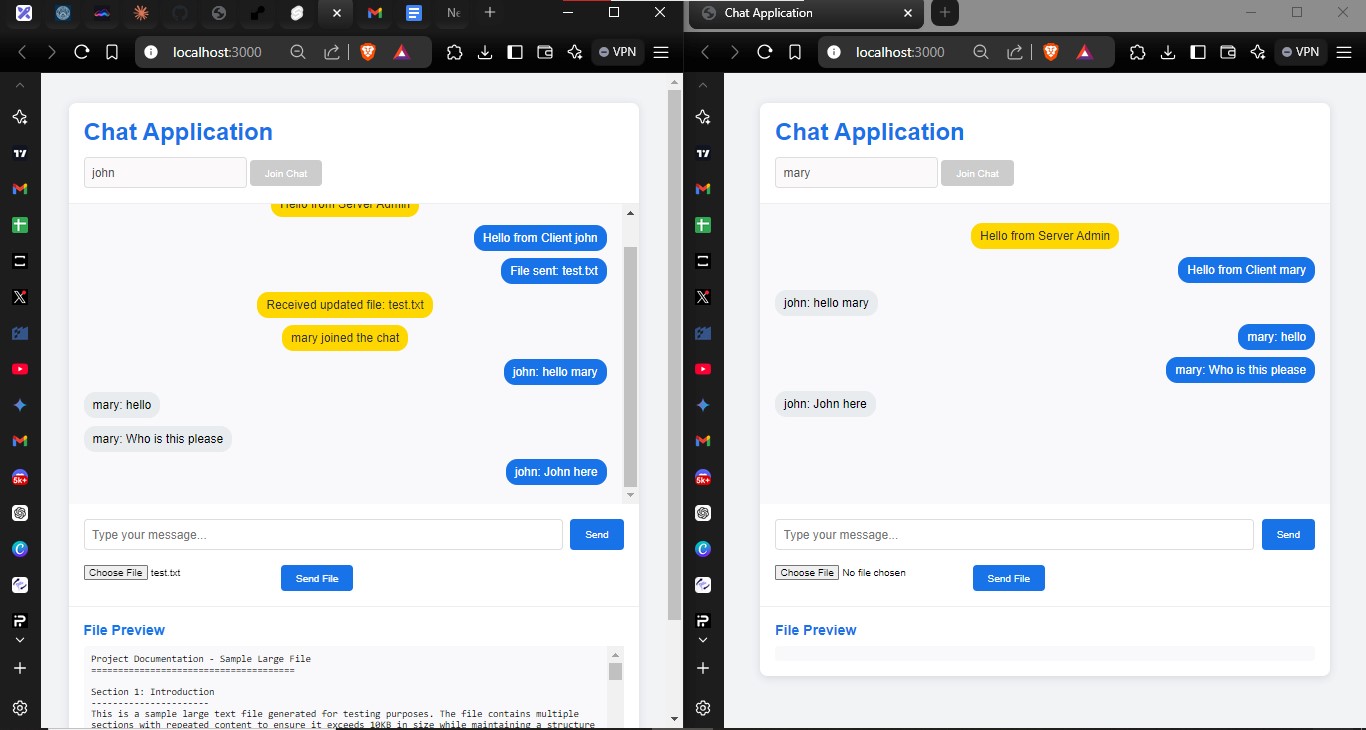
1. Support for files larger than 10KB
2. Server-side file processing
3. File preview functionality
4. Automatic file reception



*Figure 6: File Sharing Feature*

### 4.3 Multiple Client Support

1. Concurrent user connections
2. Broadcast messaging
3. Individual client tracking
4. Proper resource management



*Figure 7: Multiple Client Support*

## 5. Challenges and Solutions

### 5.1 Multiple Client Management

**Challenge**: Implementing proper handling of multiple concurrent connections.

**Solution**: Utilized Socket.IO's room feature and maintained a client map for user tracking.

## 6. Conclusion

The chat application successfully implements all the required features:

1. Real-time communication
2. Protocol-specific message handling
3. File transfer and processing
4. Multiple client support