Blog ->

Signup /login

Login -> create blog /update/delete

Without llogin:

Read blogs

In order to like or comment on blog user need to login

Admin -> can delete users

Can remove not appropriate blog

Can send warning to user

Can create blog s

////////////////////////////////////////////////////////////////////////////////////////////

Here's a **Software Requirements Specification (SRS)** for a **Real-time Chat System using Node.js**. This document includes sections such as introduction, system features, functional and non-functional requirements, and tech stack recommendations.

**Software Requirements Specification (SRS)**

**Project Title: Real-Time Chat System**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the requirements for a real-time chat system developed using Node.js, which allows users to send and receive messages instantly.

**1.2 Scope**

This system will:

* Support one-to-one
* Provide real-time communication using WebSockets (via Socket.io).
* Include features such as user authentication, message history, and online user indicators.

**1.3 Definitions**

* **Real-Time**: Instant message delivery without refresh.
* **Socket.io**: Library for real-time communication.

**2. Overall Description**

**2.1 Product Perspective**

This is a standalone system, possibly extendable with features like video calls or integration with other platforms.

**2.2 User Classes and Characteristics**

* **Registered Users**: Can log in, chat, and view message history.
* **Admins**: Can manage users and moderate chats.

**2.3 Assumptions and Dependencies**

* Internet connection is required.
* Node.js and MongoDB are installed.
* Frontend can be built using React or any modern JS framework.

**3. System Features**

**3.1 User Authentication**

* **Description**: Users can register and log in using email/password.
* **Input**: Email, password.

**3.2 Real-Time Messaging**

* **Description**: Messages are delivered instantly using Socket.io.
* **Input**: Message text.
* **Output**: Broadcast to intended recipient(s).

**3.3 Group Chats**

* **Description**: Users can create and join chat rooms.
* **Input**: Room name, list of users.
* **Output**: Room created; users notified.

**3.4 Chat History**

* **Description**: Past messages are stored in a database.
* **Input**: Chat room/user ID.
* **Output**: Fetched message history.

**3.5 Online User Indicator**

* **Description**: Shows which users are online.
* **Output**: Online users list.

**4. External Interface Requirements**

**4.1 User Interfaces**

* Responsive frontend with:
  + Login/Register page
  + Chat UI (user list, message panel

**4.2 Software Interfaces**

* Node.js backend
* MongoDB for data storage
* Socket.io for WebSocket communication

**6. Technology Stack**

* **Backend**: Node.js, Express.js
* **Real-Time Engine**: Socket.io
* **Database**: MongoDB