**Software Requirement Specification (SRS)**

**Project Name**: WebChat **Version**: 1.0  
**Prepared by**: [Your Name]  
**Date**: [Insert Date]

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to define the software requirements for "WebChat", a simple real-time chat web application. It is intended to guide an intern-level developer through each step required to design, build, and test the application.

**1.2 Scope**

The WebChat application will:

* Allow user registration and login
* Enable one-on-one real-time messaging
* Show online/offline user status
* Store and retrieve chat history
* Support basic security (e.g., password hashing)

**1.3 Intended Audience**

* Intern developers
* Project mentors
* QA testers
* Deployment engineers

**2. System Overview**

The system will be built using:

* **Frontend**: HTML, CSS, JavaScript (React optional)
* **Backend**: Node.js (Express.js) or Spring Boot
* **Database**: MongoDB or MySQL
* **WebSocket**: Socket.IO (Node.js) or Spring WebSocket

Deployment will be done using services like Render, Vercel, or localhost for testing.

**3. Functional Requirements & Development Steps**

**3.1 User Registration & Login**

**Features:**

* Register with name, email, password
* Secure login
* JWT session token (optional)

**Steps:**

1. Create registration and login forms (HTML/CSS).
2. Backend API routes: /register and /login.
3. Validate inputs.
4. Hash passwords using bcrypt.
5. Store in database.
6. Return success or error messages.

**3.2 User Dashboard & Contact List**

**Features:**

* Display list of other users
* Highlight online users

**Steps:**

1. Create a dashboard page.
2. Fetch all registered users except the current user.
3. Show online status using WebSocket connection status.

**3.3 One-on-One Chat**

**Features:**

* Real-time messaging
* Message delivery and display

**Steps:**

1. Set up Socket.IO (or Spring WebSocket) on frontend and backend.
2. When user clicks another user, open chat window.
3. Emit messages via socket.
4. Listen and display incoming messages.

**3.4 Message History**

**Features:**

* Load past conversations

**Steps:**

1. Create API endpoint to fetch message history between two users.
2. Retrieve from database and display in chat window.

**3.5 Logout**

**Features:**

* Clear session/token
* Disconnect from WebSocket

**Steps:**

1. Add logout button.
2. Clear token/localStorage.
3. Redirect to login page.

**4. Non-Functional Requirements**

**4.1 Performance**

* Should support 10-20 users with minimal delay

**4.2 Usability**

* Interface should be clean, intuitive, and responsive

**4.3 Security**

* Passwords hashed (bcrypt)
* Secure WebSocket connection (wss://)

**4.4 Maintainability**

* Follow proper folder structure:
  + Frontend: /public, /src/components, /src/pages
  + Backend: /routes, /controllers, /models

**5. Interface Design**

**5.1 Pages to Develop**

* Register Page
* Login Page
* Dashboard (with user list)
* Chat Window

**5.2 Example UI Flow**

Login/Register → Dashboard → Select User → Chat Window → Logout

**6. Database Design**

**6.1 Users**

| **Field** | **Type** |
| --- | --- |
| user\_id | String (UUID) |
| name | String |
| email | String |
| password\_hash | String |
| online\_status | Boolean |

**6.2 Messages**

| **Field** | **Type** |
| --- | --- |
| message\_id | String (UUID) |
| sender\_id | String |
| receiver\_id | String |
| text | String |
| timestamp | DateTime |

**7. Development Tools Required**

* VS Code
* Node.js / Spring Boot
* MongoDB Atlas or MySQL
* Postman (API testing)
* Git/GitHub (version control)

**8. Testing Plan**

**8.1 Unit Testing**

* Test registration and login APIs
* Test message sending/receiving

**8.2 Manual Testing**

* Register multiple users and test chat flow
* Check message delivery timing
* Test error handling and UI

**9. Future Enhancements**

* Group chats
* File sharing (images, docs)
* Read receipts
* Emojis
* Push notifications

**10. Conclusion**

This SRS aims to guide a beginner/intermediate developer step-by-step through building a real-time chat web application. It balances simplicity with essential features, and offers a solid foundation for future growth.

**End of Document**

Aashish Kumar