**Full Stack Development with MERN**

**Smart Meet Project Documentation (FSD)**

**1. Introduction**

1. **Project Title:** Smart Meet
2. **Team Members:** *Vishal Sharma .*
3. **Date of Documentation:** October 6, 2025

**2. Project Overview**

1. **Purpose:** The purpose of **Smart Meet** is to create a seamless video conferencing platform enabling live audio, video, screen sharing, text chat, and secure user authentication. It addresses key gaps in the current market, such as user fatigue, low engagement, and security concerns. The goal is to improve meeting productivity and outcomes through smart, integrated features.
2. **Problem Statement:** Many remote teams, students, and professionals face challenges like high levels of screen fatigue, scattered communication, security/privacy concerns, and poor audio/video quality on unstable networks.
   1. **Customer Pain:** App freezes at critical moments, complicated interfaces lead to confusion, background noise and echo problems, and difficulty managing multi-user calls.
3. **Solution (Smart Meet):** Smart Meet is a full-stack web application (React front-end + Node.js/WebRTC back-end) that provides reliable, secure, and engaging video conferencing.
   1. **Uniqueness/Novelty:** AI-powered real-time transcription and summaries, interactive virtual whiteboards, optimized low-bandwidth mode, and end-to-end encryption by default.
4. **Features:**
   1. **User & Account Management:** Sign up, Login, Password Reset, Profile Management, and Auth login using Google/Microsoft.
   2. **Meeting Management:** Schedule a new meeting, start an instant meeting, and generate/share unique meeting links.
   3. **In-Meeting Experience:** High-quality audio/video streaming, screen sharing, real-time chat, and participant controls (mute/unmute, start/stop video).
   4. **Post-Meeting Features:** Record meetings to the cloud or locally, and access/view past meeting recordings.

**3. Architecture**

The Smart Meet platform is designed with a real-time, microservices architecture using WebRTC to ensure low latency, high availability, and scalability.

1. **Frontend (Client Layer):**
   1. **Technology:** **React.js**.
   2. **Role:** Web-based interface for video calls, screen sharing, chat, and scheduling. Provides a dynamic and responsive UI.
2. **Backend (Application Logic Layer):**
   1. **Technology:** **Node.js** and **Express.js**, with **Socket.IO** for signaling.
   2. **Role:** Manages business logic, API requests, user authentication, meeting management, real-time signaling, and chat handling.
3. **Database:**
   1. **Technology:** **MongoDB**.
   2. **Role:** Stores user profiles, meeting data, chat logs, and recording metadata. Provides scalability and quick data retrieval for live sessions.

**4. Setup Instructions**

1. **Prerequisites:**
   1. Node.js (LTS version)
   2. MongoDB installed and running or access to a cloud MongoDB service.
   3. Git
   4. A modern web browser.
2. **Installation:**
   1. Clone the repository: git clone [repository URL]
   2. Navigate to the project root: cd smart-meet-project
   3. Set up environment variables for the Server (e.g., MONGO\_URI, JWT\_SECRET) and Client (e.g., REACT\_APP\_API\_URL).
   4. Install backend dependencies: cd server && npm install
   5. Install frontend dependencies: cd ../client && npm install

**5. Folder Structure**

1. **Client:** (React Frontend)
   1. src/components/ (Reusable UI components)
   2. src/pages/ (Top-level views like Dashboard, MeetingRoom, Login)
   3. src/context/ (State management)
   4. src/services/ (API calls)
2. **Server:** (Node.js/Express Backend)
   1. routes/ (API endpoints: user.js, meeting.js)
   2. controllers/ (Business logic for routes)
   3. models/ (MongoDB schemas: User, Meeting)
   4. config/ (Database connection, environment variables)
   5. middleware/ (Authentication, error handling)

**6. Running the Application**

1. **Frontend:**
   1. From the client/ directory: npm start (Starts the React development server)
2. **Backend:**
   1. From the server/ directory: npm start (Starts the Node.js server)

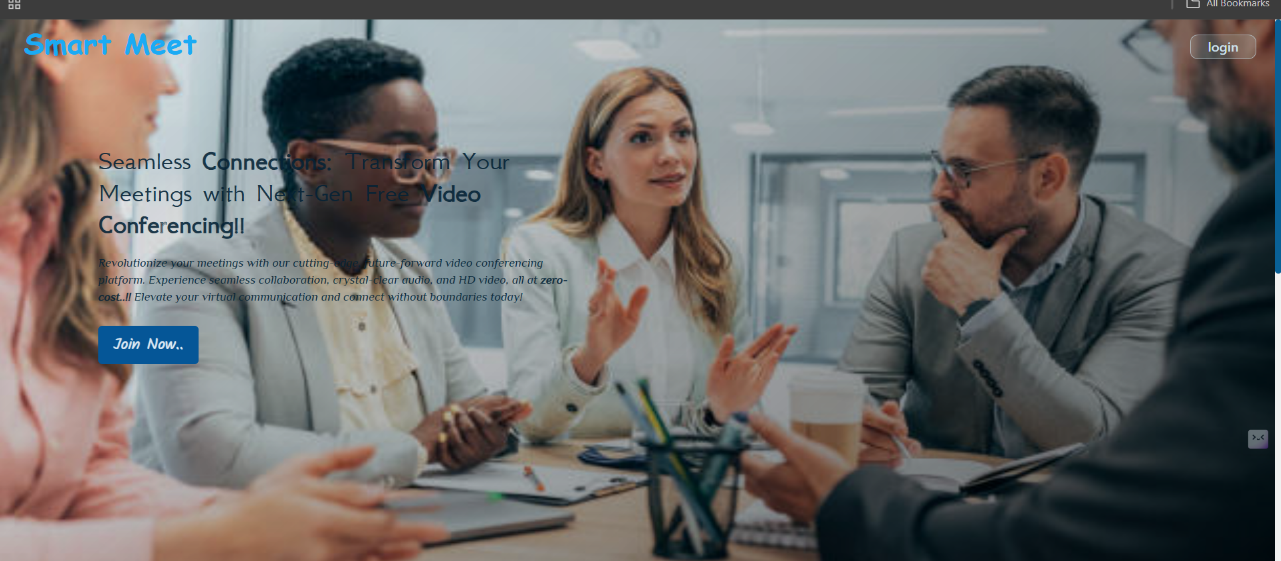
**7. API Documentation**

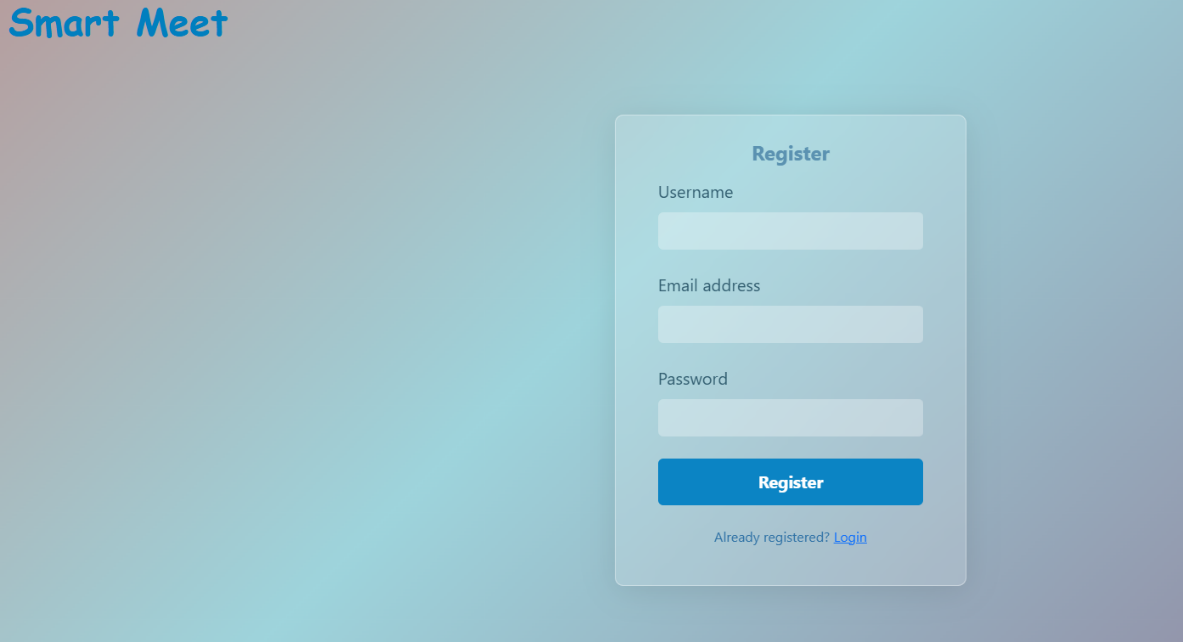
| Functional Area | Request Method | Endpoint Example | Description |
| --- | --- | --- | --- |
| **Authentication** | POST | /api/auth/login | Authenticates a user and returns a JWT. |
|  | POST | /api/auth/register | Creates a new user account. |
| **Meetings** | POST | /api/meetings/create | Creates and schedules a new meeting. |
|  | GET | /api/meetings/:id | Retrieves meeting details for joining. |
| **User Profile** | PUT | /api/users/profile | Updates the logged-in user's profile information. |

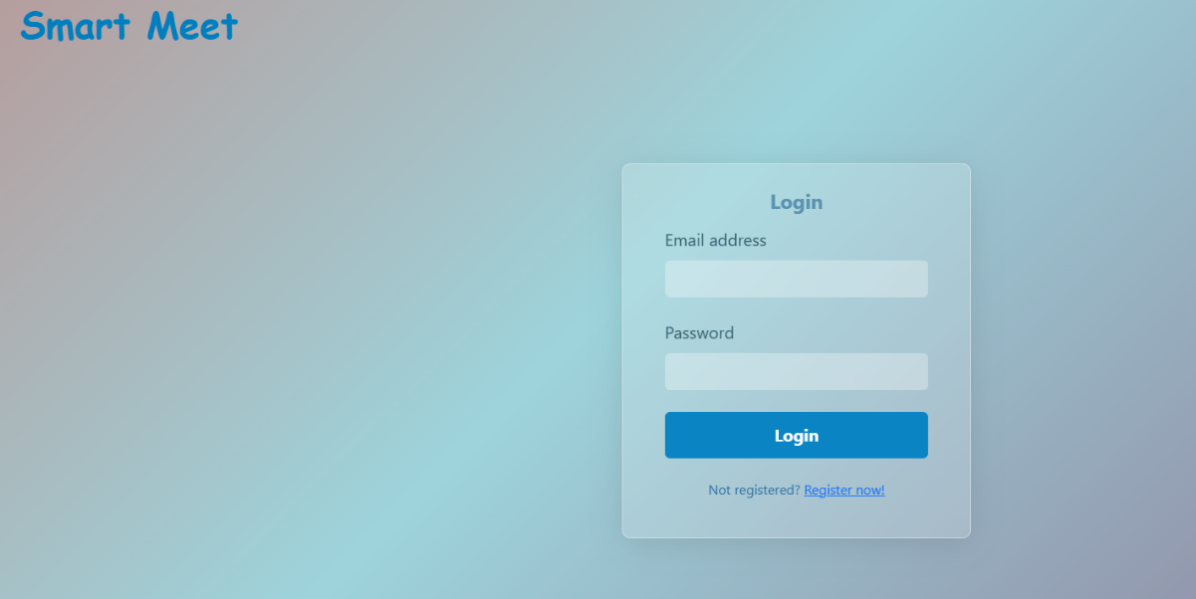
**8. Authentication**

1. **Mechanism:** Secure login is handled via user credentials and may include third-party authentication (Google/Microsoft).
2. **Authorization:** JWT (JSON Web Token) is used for secure authentication and session management.
3. **Security:** Role-based access (Host, Participant, Admin) is implemented.

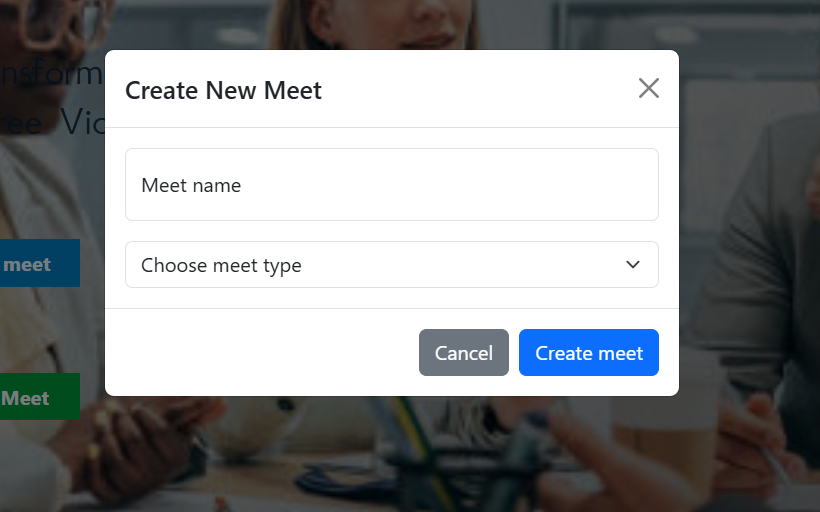
**9. User Interface**











**10. Testing**

1. **Strategy:** User Acceptance Testing (UAT) was conducted to ensure the platform addresses key user requirements.
2. **Scope:** Testing covered User registration/login, meeting creation/joining, in-meeting controls (audio/video, screen sharing, chat), and recording functionality.
3. **Test Environment:** URL: https://freelancefinder.example.com.
4. **Known Bug:** **BG-001** - Screen sharing fails to start (High Severity, Open Status). The screen share button should prompt the user to select a window, but currently nothing happens.

**11. Screenshots or Demo**

**URL :-** [**smart-meet-omega.vercel.app**](https://smart-meet-omega.vercel.app/)

**12. Known Issues**

1. Screen sharing functionality is currently non-responsive (Bug ID: BG-001).
2. Further performance testing may be required to guarantee audio/video latency stays under the target of 150ms and join time under 5 seconds, as per NFRs.

**13. Future Enhancements**

1. **Advanced Collaboration:** Implement advanced AI moderation and interactive tools like virtual whiteboards and live polls/Q&A sessions.
2. **AI Features:** Integration of AI-powered real-time transcription and automated meeting summaries.
3. **Scalability:** Develop future native apps for iOS and Android.
4. **Business Model:** Introduce Enterprise plans with advanced security, custom branding, and pay-per-use features like advanced AI analytics.
5. **Integration:** Integration with calendar and project management tools