Vishweshwarayya Abhiyantriki Padvika Mahavidyalaya, Almala

Department of Information Technology

Remote Meeting Web Application

Team Member	Enrollment No.
1.Raghav Sunil Deshpande	2210950223

1. Background of the Project

In today's digital age, remote communication has become a necessity. With the rise of global teams, remote work, and online education, the need for reliable and efficient meeting software has never been greater. Existing solutions like Zoom have shown the potential of virtual collaboration, but there is always room for innovation and customization. This project aims to develop a remote meeting software that addresses common challenges like user interface simplicity, real-time communication, and data security.

2. Working of the Project

The project is structured as a web-based application that facilitates remote meetings through video, audio, and chat functionalities. The following components outline the working mechanism:

- **Frontend:** Built using React.js, the frontend manages the user interface, allowing users to create or join meetings, manage their profiles, and communicate in real time.

- **Backend:** The backend is powered by Node.js and Express, handling server-side logic, user authentication, and API requests. Socket.io is utilized for real-time communication between clients, ensuring seamless audio and video transmission.

- **Database:** MongoDB is employed for storing user data, meeting records, chat logs, and other relevant information. The database is structured to support quick retrieval and storage of data to maintain the application's performance.

- **Real-time Communication:** WebRTC is leveraged to establish peer-to-peer connections for video and audio communication. This ensures low latency and high-quality streams during meetings.

- **Security:** The application includes user authentication (using JWT tokens) and data encryption to ensure secure communication and data protection.

3. Advantages

- Real-Time Communication: The use of WebRTC and Socket.io allows for low-latency, real-time video, and audio communication, providing a smooth user experience.

- Scalability: With the combination of Node.js and MongoDB, the application is designed to handle a large number of simultaneous users without performance degradation.

- User-Friendly Interface: React.js ensures a responsive and intuitive user interface, making it easy for users to navigate and use the platform.

-Security: The application incorporates robust security measures, including encrypted communication and secure authentication, to protect user data and privacy.

- Customization: Unlike off-the-shelf solutions, this software can be customized to meet specific organizational needs, making it a flexible option for various user groups.

4. Software Description

- Frontend:

-Technologies: React.js, HTML, CSS, JavaScript

-Features: User login/signup, dashboard, meeting scheduling, real-time chat, video, and audio controls.

-Backend:

-Technologies: Node.js, Express.js, WebRTC, Socket.io

-Features: User authentication, API for managing meetings, handling real-time communication, server-side logic.

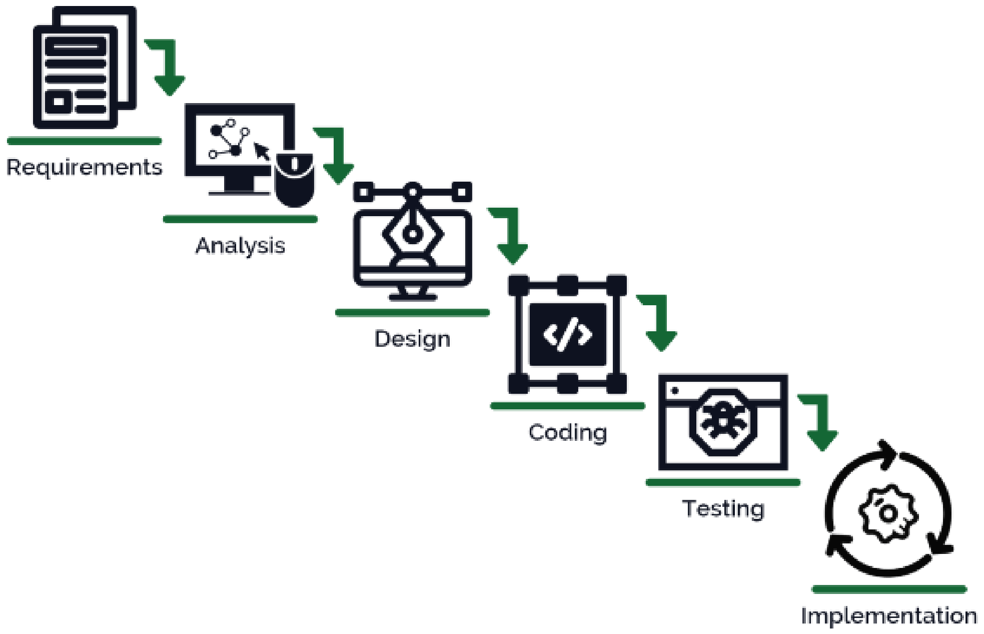
-Database:

-Technology: MongoDB

-Features: Stores user information, meeting data, chat logs, and other metadata.

-Deployment: The application can be deployed on cloud platforms such as AWS, Azure, or Heroku, ensuring scalability and reliability.

5. Project Life Cycle



-Requirement Analysis: Understanding the specific needs of users and defining the scope of the project.

-Design Phase: Creating wireframes and UI/UX design for the frontend, and planning the architecture for the backend and database.

-Development Phase: Writing code for the frontend, backend, and integrating the database. Implementing real-time communication using WebRTC and Socket.io.

-Testing Phase: Performing unit testing, integration testing, and user acceptance testing to ensure the application works as intended.

-Deployment: Deploying the application on a cloud platform, ensuring it is accessible and scalable.

-Maintenance and Updates: Continuously monitoring the application, fixing bugs, and implementing new features as required.

6. System Requirements

Hardware:

1. ***Server:*** A cloud server with at least 4 GB RAM, 2 CPU cores, and 100 GB storage (e.g., AWS EC2, DigitalOcean Droplet).
2. ***Client:*** Any modern computer or smartphone with internet access, a webcam, and a microphone for video and audio communication.

Software:

1. ***Operating System:*** Windows, macOS, Linux (for development and deployment).
2. ***Development Environment:*** Node.js, NPM, MongoDB, React.js, VS Code or any other IDE.
3. ***Browser:*** Chrome, Firefox, or any other modern browser supporting WebRTC.

7. Limitations/Disadvantages

-Bandwidth Dependency: The quality of video and audio communication heavily depends on the user's internet connection. Poor connectivity can result in lag or dropped calls.

-Scalability Challenges: While the system is designed to be scalable, managing a very high number of users simultaneously may require more sophisticated load balancing and resource management strategies.

-Data Privacy Concerns: Despite encryption, any online platform is susceptible to data breaches, which may concern users regarding the privacy of their communications.

-Complexity in Development: Implementing real-time communication with low latency and high reliability is technically challenging and requires expertise in WebRTC and related technologies.

8. Applications

-Remote Work: Organizations can use the software for team meetings, client interactions, and collaborative projects.

-Online Education: Educational institutions can leverage the platform for virtual classrooms, student-teacher interactions, and webinars.

-Telemedicine: Healthcare providers can use the software for remote consultations, follow-ups, and patient communication.

-Social Gatherings: The software can be adapted for personal use, allowing friends and families to connect via video calls, especially when physical meetings are not possible.

9. References

Documentation and resources for React.js, Node.js, and WebRTC from their official websites.

Online tutorials and guides on MongoDB for database management.

Research papers and articles on the challenges and advancements in real-time communication technologies.