PROJECT REPORT

PEER-PEER MULTI-USER VIDEO CALLING

NAME: GEEVARGHESE A K

FS BETA

INTRODUCTION

VI.mate is a cutting-edge video conferencing application designed to facilitate seamless and high-quality peer-to-peer communication. Leveraging the power of React.js for the frontend and Vite for rapid development and building, VI.mate aims to revolutionize the way users connect and communicate in real-time. The application utilizes WebRTC technology to provide unparalleled video streaming capabilities, enabling users to share their video feeds directly without the need for a centralized server. This approach not only enhances the user experience by reducing latency but also significantly reduces data storage requirements on the server.

VI.mate is built with a focus on simplicity and professionalism, offering a user-friendly interface that is both intuitive and visually appealing. The application supports real-time chat, screen sharing, and authorization management, ensuring a secure and engaging environment for all users. The project is open-source, inviting contributions from the developer community to further enhance its capabilities and reach.

The application's architecture is organized into distinct components, including a StreamProvider for managing streams, a set of styled components for the UI, and a routing system for navigating between different screens such as the Lobby and Room. The backend, implemented using Node.js and Express.js, handles signaling purposes for the WebRTC technology, ensuring smooth communication between peers.

VI.mate stands out for its innovative approach to video conferencing, eliminating the need for a centralized server and providing a platform for direct, peer-to-peer communication. With its robust features and open-source nature, VI.mate is poised to become a go-to solution for high-quality video conferencing needs.

Frontend Structure:

The frontend of VI.mate is built using React.js, a popular JavaScript library for building user interfaces. It is organized into a modular structure, facilitated by Vite, a build tool that offers fast development and building capabilities. The application is structured around several key components:

* Routers: Located in the src/router directory, the Routers component is responsible for managing the application's routing logic. It ensures that users are directed to the appropriate screens based on their actions and the application's state.
* Screens: The src/screens directory contains components for different screens within the application, such as the Lobby and Room. These components are responsible for rendering the UI for each screen and managing the user interactions specific to that screen.
* Context: The src/context directory includes the StreamProvider component, which is crucial for managing streams in the application. This component likely handles the creation, management, and destruction of media streams, ensuring that video and audio data is efficiently handled.
* Styled Components: Throughout the application, styled-components are used to define the UI elements' styles. This approach allows for the creation of reusable, styled React components with CSS written directly within JavaScript files.
* Utilities and Hooks: The application likely utilizes custom hooks and utility functions to manage state, handle side effects, and perform other common tasks in a React application.

Backend Structure:

The backend of VI.mate is implemented using Node.js and Express.js, providing a robust server environment for handling signaling purposes for the WebRTC technology. The server-side logic is organized into a structured format, with the following key components:

* Server Logic: The core server logic is contained within the server directory. This includes the setup for the Express.js server, middleware configurations, and route handlers for managing requests and responses.
* Signaling: The server is primarily responsible for handling signaling between peers. This involves exchanging information necessary for establishing and maintaining peer-to-peer connections, such as offer, answer, and ICE candidate messages.
* Dependencies: The package.json file lists all the npm packages used by the server, including those for WebRTC, Socket.io for real-time communication, and any other necessary libraries.
* Profiles: The profiles directory contains images of female and male profiles for the application. These are likely used for user avatars or profile pictures within the application.

This structure ensures a clear separation of concerns between the frontend and backend, with each component playing a specific role in the overall functionality of the VI.mate application.

# REQUIREMENTS

I. System Requirements

* Operating System: Windows

• node

II. Dependencies

* React.js
* Vite
* Socket.io
* webRTC

# FEATURES

* Peer-to-Peer Video Conferencing: At the heart of VI.mate is its ability to facilitate direct, peer-to-peer video conferencing. This is achieved through the use of WebRTC technology, which allows for real-time video and audio communication without the need for a centralized server.
* Real-Time Chat: VI.mate includes a real-time chat feature, enabling users to communicate textually during their video calls. This feature is designed to enhance the interactive experience, allowing users to discuss topics, share ideas, or simply chat with each other.
* Screen Sharing: Users have the ability to share their screens with others during a video call. This feature is particularly useful for presentations, demonstrations, or when collaborating on documents or code.
* Authorization Management: VI.mate implements a secure authorization system to manage access to video calls. This ensures that only authorized users can join specific calls, providing a level of security and privacy for users.
* User-Friendly Interface: The application boasts a user-friendly interface that is both intuitive and visually appealing. It is designed to be easy to navigate, with clear instructions and intuitive controls for managing video calls and chats.
* Open-Source and Community-Driven: VI.mate is open-source, inviting contributions from the developer community. This approach not only enhances the application's capabilities but also fosters a sense of community among its users.
* Cross-Platform Compatibility: VI.mate is designed to be accessible across various platforms, including desktop and mobile devices. This ensures that users can join video calls from anywhere, at any time.
* High-Quality Video Streaming: Utilizing WebRTC technology, VI.mate provides high-quality video streaming capabilities. It optimizes video quality and performance to ensure a smooth and clear video experience for all users.

## FUTURE ENHANCEMENT

* AI-Powered Features: Leveraging artificial intelligence, future expansions could introduce AI-powered features such as automatic language translation, intelligent chat moderation, and personalized video call recommendations based on user preferences and behavior.
* Customizable User Experience: Future expansions could include the ability for users to customize their video call experience. This could range from choosing from a variety of video call themes and backgrounds to customizing the layout and arrangement of video feeds during a call.
* Support for Larger Meetings: As the demand for video conferencing grows, future expansions could focus on enhancing VI.mate's ability to support larger meetings. This could involve optimizing the application for better performance under high load and introducing features to manage and organize larger groups of participants.
* Educational and Training Tools: Recognizing the growing need for remote learning and training, future expansions could include educational and training tools within VI.mate. This could include interactive whiteboards, quizzes, and the ability to create and share educational content directly within the application.