Video Chat Application Documentation

Overview

This documentation explains the process of creating, deploying, and testing a video chat application using WebRTC, PeerJS, and a TURN server for network traversal. The project allows users to join a roo and communicate via video and audio streams.

Technologies Used

- **HTML/CSS**: Frontend structure and styling
- **JavaScript**: Client-side logic
- **Node.js**: Backend server
- **Socket.IO**: Real-time communication between server and clients
- **PeerJS**: WebRTC abstraction for peer-to-peer communication
- **Xirsys**: TURN/STUN server for NAT traversal
- **Render**: For deployment

Folder Structure

- `index.html`: Main HTML file containing the video grid and basic layout.
- `index.js`: Client-side JavaScript for handling video streams and peer connections.
- `server.js`: Node.js server managing WebSocket connections and signaling.
- `package.json`: Dependencies and scripts.
- `public/`: Contains static assets like JavaScript files and CSS files.

Features

- Join a room using a unique room ID.
- Real-time video and audio streaming between participants.
- Handles multiple participants with dynamic video grid resizing.
- Works across different networks using a TURN server.

Setting Up the Application

Prerequisites

1. Node.js installed on your system.
2. A GitHub account for version control and deployment.
3. A free Xirsys account to get TURN/STUN credentials.

Step-by-Step Guide

1. Clone the Repository

```
git clone <repository_url>
cd <repository_folder>
```

2. Install Dependencies

```
npm install
```

3. Configure TURN/STUN Server

1. Sign up at Xirsys and create a new "ice" configuration.
2. Retrieve your TURN/STUN credentials from the Xirsys dashboard.
3. Update your `index.js` file with the credentials as follows:

```
iceServers: [
  { urls: ['stun:your_stun_server_url'] },
  {
    username: 'your_username',
    credential: 'your_credential',
    urls: [
      'turn:your_turn_server_url:80?transport=udp',
      'turn:your_turn_server_url:3478?transport=udp',
      'turn:your_turn_server_url:80?transport=tcp',
      'turn:your_turn_server_url:3478?transport=tcp',
      'turns:your_turn_server_url:443?transport=tcp',
      'turns:your_turn_server_url:5349?transport=tcp'
    ]
  }
]
```

4. Run the Application Locally

```
node server.js
```

Access the app in your browser at `http://localhost:3000`.

5. Deploy to Render

1. Push your project to GitHub.
   ```
   git add .
   git commit -m "Initial commit"
   git push origin main
   ```
2. Log in to Render and create a new web service.
3. Connect your GitHub repository and deploy.

4. Update the deployed app URL in your project for client-side use.

Testing the Application

1. Open the application on two different devices.
2. Use the same room ID to join the room.
3. Ensure both video and audio streams are visible.

Troubleshooting

Problem: Not working on different networks

- Ensure the TURN server credentials are valid.
- Check that your app is deployed correctly and accessible online.

Problem: Video grid not displaying

- Verify that the **`` `index.js` ``** file is correctly linked in **`` `index.html` ``**.
- Check the console for any JavaScript errors.

Future Improvements

- Add a chat feature for text communication.
- Implement screen sharing.
- Add user authentication and room security.
- Optimize UI for mobile devices.

Credits

- **PeerJS**: For simplifying WebRTC.
- **Xirsys**: For providing free TURN/STUN servers.
- **Render**: For deployment.

For additional help or questions, feel free to reach out!