A Guide to Building a Full-Stack Live Audio Chat Room App

Alexis Ambriz, CMPE 297

August 23, 2024

Introduction

- Overview of the project
- Purpose of the live audio chat room app
- Technologies used: ReactJS, NodeJS, Stream API

Project Overview

Imagine a platform like Clubhouse or Twitter Spaces; Users can create and join audio chat rooms! This involves real-time audio interactions and discussions.

Technologies Used

Important frameworks and APIs used in the project:

- **ReactJS**: For building the frontend user interface
- **NodeJS**: For the backend server and API
- **Stream API**: For managing audio calls and user interactions

Setting Up the Project

Steps involved in setting up the project:

- Creating a new app on GetStream.io
- Installing necessary packages: React, NodeJS, Express, etc.
- Project directory structure

GetStream.io Setup

GetStream.io is a platform that provides APIs for building in-app chat, video, and audio features. To set up the project:

- Sign up for a free account on GetStream.io
- Create a new app and obtain API keys
- GetStream.io provides APIs for building in-app chat, video, and audio features

Package Installation

- Use npm or yarn to install required packages
- Examples: ‘react‘, ‘react-dom‘, ‘node-fetch‘, ‘express‘
- These packages provide essential functionalities for building the app

Project Structure

- Organize your project into clear folders
- Example: ‘frontend‘, ‘backend‘, ‘shared‘
- This structure helps keep the code organized and maintainable

Backend Development

- NodeJS server setup
- ExpressJS for handling HTTP requests
- Implementing routes for user authentication, room creation, and joining
- Integrating Stream API for audio calls and user interactions

NodeJS Server Setup

- Create a new NodeJS project
- Initialize the project using ‘npm init‘ or ‘yarn init‘
- Set up the ‘server.js‘ file as the entry point

ExpressJS for HTTP Requests

- Integrate the ExpressJS framework
- Define routes to handle different HTTP requests
- Example: ‘/api/users‘, ‘/api/rooms‘, ‘/api/join-room‘

Integrating Stream API

- Use the Stream API to manage audio calls, user interactions, and more
- Create users, channels (rooms), and manage audio streams
- Use the ‘node-fetch‘ package to make HTTP requests to the Stream API

Frontend Development

- Creating the React app
- User Context for managing user state
- Main component for authentication and navigation
- Room component for displaying room information and audio controls
- Integrating Stream JavaScript SDK for audio management

Creating the React App

- Use a tool like Create React App to generate a new React project
- Set up the basic structure and components

User Context

- Create a React Context to store and share user information
- Pass the context to components that need access to user data

Main Component

- The main component handles user authentication and navigation
- It renders different components based on the user's state

Room Component

- The Room component displays the content of a specific room
- It shows room information, participant list, and audio controls
- Uses the Stream JavaScript SDK to manage audio streams

Integrating Stream JavaScript SDK

- Use the Stream JavaScript SDK to connect to the Stream service
- Manage audio streams, participant list, and other functionalities

Conclusion

- Recap of the key steps involved
- Potential applications and future enhancements
- Encourage viewers to explore and build upon the project