Hotel Booking Chatbot

This project implements a hotel booking chatbot with a React frontend and an Express.js backend. The chatbot uses OpenAI's API for natural language processing and maintains conversation history using SQLite and Sequelize.

Project Structure

```
hotel-booking-chatbot/
├── frontend/           # React frontend
│   ├── src/
│   │   ├── App.jsx
│   │   ├── App.css
│   │   └── main.jsx
│   ├── package.json
│   └── vite.config.js
├── backend/            # Express.js backend
│   ├── index.js
│   ├── emailService.js
│   ├── package.json
│   └── .env
└── README.md
```

Prerequisites

- Node.js (v14 or later)
- npm
- OpenAI API key

Setup

PROF

Backend Setup

1. Navigate to the backend directory:
   cd hotel-booking-chatbot/backend

2. Install dependencies:
   npm install

3. Create a `.env` file in the backend directory and add your OpenAI API key:
   OPENAI_API_KEY=your_api_key_here

4. Start the server:
   npm start

The server will start running on http://localhost:3000.

Frontend Setup

1. Navigate to the frontend directory:
```
cd hotel-booking-chatbot/frontend
```

2. Install dependencies:
```
npm install
```

3. Start the development server:
```
npm run dev
```

The frontend will be accessible at http://localhost:5173.

Usage

Once both the backend and frontend are running, you can interact with the chatbot through the web interface. The chatbot can handle queries about room bookings, provide room options, and simulate the booking process.

API Endpoints

Chat Endpoint

- URL: /chat
- Method: POST
- Body:
```
{
"message": "I'd like to book a room",
"userId": "user123"
}
```
- Response:
```
{
"response": "Certainly! I'd be happy to help you book a room at Bot9 Palace. Could you please tell me what dates you're looking to stay with us?"
}
```

PROF

External API Endpoints

The chatbot interacts with the following external API endpoints:

1. List Hotel Room Options:
```
curl -X GET https://bot9assignement.deno.dev/rooms
```

2. Create a Booking:
```
curl -X POST https://bot9assignement.deno.dev/book
-H "Content-Type: application/json"
-d '{
"roomId": 2,
"fullName": "John Doe",
"email": "john.doe@example.com",
"nights": 3
}'
```

Testing

You can test the chatbot using the web interface or by sending POST requests to the /chat endpoint using tools like cURL or Postman.

Example cURL command:

```
curl -X POST http://localhost:3000/chat
-H "Content-Type: application/json"
-d '{"message": "I want to book a room", "userId": "testUser123"}'
```

Error Handling

The application includes basic error handling for invalid user inputs and API failures. Check the server logs for any error messages.

Technologies Used

- Backend: Express.js, OpenAI API, SQLite, Sequelize
- Frontend: React, Vite, Axios, react-markdown

Future Improvements

- Add more sophisticated error handling and recovery mechanisms
- Enhance the chatbot's natural language understanding capabilities
- Implement actual email sending functionality for booking confirmations