```markdown

# emedical

## Overview

### Introduction

The **emedical** project is a comprehensive e-medical platform designed to streamline healthcare services and improve patient-doctor interactions. Built using modern web technologies, this platform aims to provide a seamless experience for both patients and medical professionals. The system allows users to book appointments, manage medical records, and access telemedicine services.

### Purpose and Features

The primary purpose of emedical is to digitize and simplify healthcare services. Key features include:

- **User Authentication:** Secure login and registration for patients and doctors.
- **Appointment Management:** Easy scheduling and rescheduling of appointments.
- **Telemedicine:** Video consultation capabilities for remote patient care.
- **Medical Records:** Secure storage and access to patient medical records.
- **Payment Processing:** Integration with Stripe for secure online payments.
- **Notifications:** Real-time notifications for appointment reminders and updates.
- **Analytics:** Dashboard for doctors to view patient statistics and appointment trends.

### Intended Users and Use Cases

- **Patients:** Can book appointments, view medical records, and communicate with doctors.
- **Doctors:** Can manage appointments, view patient records, and provide telemedicine services.
- **Administrators:** Can manage users, appointments, and system settings.

The platform is ideal for healthcare providers looking to modernize their services and improve patient engagement.

---

## Installation

### Required Software

- Node.js (v16.14.2 or higher)
- npm (v8.5.0 or higher)
- MongoDB (v4.4 or higher)
- A modern web browser (Chrome, Firefox, etc.)
- Git (optional but recommended for version control)

### Step-by-Step Installation

1. **Clone the Repository** `bash git clone https://github.com/your-repository/emedical.git cd emedical`

2. **Install Dependencies** `bash npm install`

3. **Set Up MongoDB**

   - Install MongoDB on your system or use MongoDB Atlas for a cloud-based solution.
   - Create a `.env` file in the root directory and add the following: `MONGODB_URI=mongodb://localhost:27017/emedical`

4. **Start the Development Server** `bash npm start`

5. **Access the Application**

   - Open your browser and navigate to `http://localhost:3000`.

---

## Code Structure

### Directory Structure
```

```
. â”œâ”€â”€ .gitignore â”œâ”€â”€ README.md â”œâ”€â”€ build/ â”œâ”€â”€ package-lock.json â”œâ”€â”€ package.json
â”œâ”€â”€ public/ â”œâ”€â”€ src/ â”‚ â”œâ”€â”€ components/ â”‚ â”œâ”€â”€ pages/ â”‚ â”œâ”€â”€ styles/ â”‚
â”œâ”€â”€ utils/ â”‚ â”œâ”€â”€ App.js â”‚ â””â”€â”€ index.js
```

Key Files

| File/Directory | Description | |---------------------|--|| `.gitignore` | Specifies files and directories to ignore by Git. | | `README.md` | This documentation file. | | `package.json` | Contains project dependencies and scripts. | | `src/App.js` | Main application component. | | `src/index.js` | Entry point for the React application. | | `public/` | Contains static assets and the `index.html`. |

Features & Functionality

Main Features

1. **User Authentication**

   - Secure login and registration using Mantine and React.
   - Role-based access control for patients and doctors.

2. **Appointment Management**

   - Calendar integration for scheduling appointments.
   - Real-time notifications for appointment reminders.

3. **Telemedicine**

   - Video consultation using WebRTC or similar technologies.
   - Integrated chat system for patient-doctor communication.

4. **Medical Records**

   - Secure storage and retrieval of patient records.
   - PDF generation for medical reports.

5. **Payment Processing**

   - Integration with Stripe for online payments.
   - Secure payment gateway for appointment fees.

Code Snippets

User Authentication

```javascript // src/components/Auth/Login.js import { useState } from 'react'; import { TextInput, Button, Notification } from '@mantine/core';

function Login() { const [email, setEmail] = useState("); const [password, setPassword] = useState("); const [error, setError] = useState(");

const handleSubmit = async (e) => { e.preventDefault(); try { // Login logic here } catch (err) { setError('Invalid credentials'); } };

return (

Login {error && {error}} ); } ```

#### Payment Integration

```javascript // src/components/Payment/PaymentForm.js import { useStripe, useElements } from '@stripe/react-stripe-js'; import { CardElement } from '@stripe/react-stripe-js'; import { Button } from '@mantine/core';

function PaymentForm({ amount }) { const stripe = useStripe(); const elements = useElements();

const handleSubmit = async (e) => { e.preventDefault(); if (!stripe || !elements) return;

```
const { error } = await stripe.createPaymentMethod({
  type: 'card',
  card: elements.getElement(CardElement),
});

if (error) {
```

```
  console.error(error);
} else {
  // Process payment
}
```

};

return (

Pay ${amount}

); } ```

API Documentation

Backend API Endpoints

| Endpoint | Method | Description | |---------------------------|--------|--------------------------------| | `/api/auth/login` | POST | Authenticate user | | `/api/auth/register` | POST | Register new user | | `/api/appointments` | GET | Get all appointments | | `/api/appointments` | POST | Create new appointment | | `/api/appointments/:id` | PUT | Update appointment | | `/api/appointments/:id` | DELETE | Delete appointment |

Example Request/Response

GET `/api/appointments` ```javascript // Request { "headers": { "Authorization": "Bearer " } }

// Response [{ "id": "123", "patient": "John Doe", "doctor": "Dr. Smith", "date": "2023-10-01T10:00:00", "status": "pending" }] ```

Configuration

Environment Variables

Create a `.env` file in the root directory with the following variables:

```
MONGODB_URI=mongodb://localhost:27017/emedical STRIPE_PUBLIC_KEY=your_stripe_public_key
STRIPE_SECRET_KEY=your_stripe_secret_key JWT_SECRET=your_jwt_secret
```

Project Settings

- **Development Mode:** Run `npm start`.
- **Production Build:** Run `npm run build`.

Dependencies

| Package | Version | |-----------------------------|---------| | @emotion/react | ^11.9.0 | | @emotion/styled | ^11.9.0 | | @mantine/carousel | ^6.3.1 | | @mantine/charts | ^6.3.1 | | @mantine/code-highlight | ^6.3.1 | | @mantine/core | ^6.3.1 | | @mantine/dates | ^6.3.1 | | @mantine/dropzone | ^6.3.1 | | @mantine/form | ^6.3.1 | | @mantine/hooks | ^6.3.1 | | @mantine/modals | ^6.3.1 | | @mantine/notifications | ^6.3.1 | | @mantine/nprogress | ^6.3.1 | | @mantine/spotlight | ^6.3.1 | | @mantine/tiptap | ^6.3.1 | | @mui/icons-material | ^5.10.0 | | @mui/material | ^5.10.0 | | @mui/styled-engine-sc | ^5.10.0 | | @stripe/react-stripe-js | ^2.1.0 | | @stripe/stripe-js | ^1.49.0 | | @tabler/icons-react | ^1.0.0 | | @testing-library/jest-dom | ^5.16.5 | | @testing-library/react | ^13.4.0 | | @testing-library/user-event | ^13.5.0 | | @tiptap/extension-link | ^2.1.0 | | @tiptap/react | ^2.1.0 | | @tiptap/starter-kit | ^2.1.0 | | axios | ^1.3.4 | | cors | ^2.8.5 | | date-fns | ^2.29.3 | | dayjs | ^1.11.7 | | embla-carousel-react | ^8.3.0 | | express | ^4.18.2 | | framer-motion | ^10.16.0| | mongodb | ^4.10.3 | | mongoose | ^8.0.3 | | react | ^18.2.0 | | react-datepicker | ^4.13.0 | | react-dom | ^18.2.0 | | react-responsive-carousel | ^5.3.0 | | react-router-dom | ^6.4.3 | | react-scripts | ^5.0.1 | | recharts | ^2.7.0 | | stripe | ^8.107.0| | styled-components | ^6.0.5 | | web-vitals | ^2.1.4 |

Usage

Running the Application

1. Install dependencies: `bash npm install`

2. Start the development server: `bash npm start`

3. The application will be available at `http://localhost:3000`.

Key Commands

Command	Description
`npm start`	Start the development server
`npm run build`	Build the production version
`npm test`	Run tests
`npm run eject`	eject the app

Testing

Running Tests

To run the test suite, execute the following command: `bash npm test`

The tests are written using Jest and React Testing Library. The test files are located in the `src` directory, with each component having its corresponding test file.

Deployment

Deploying to Production

1. **Prepare for Deployment:** `bash npm run build`

2. **Deploy to a Cloud Provider:**

   - Use platforms like Heroku, AWS, or Vercel to deploy the application.
   - Ensure MongoDB is configured in the cloud (e.g., MongoDB Atlas).

3. **Set Environment Variables:**

   - Configure the `.env` file with production-specific variables.
   - Ensure sensitive information like API keys and database URIs are stored securely.

4. **Start the Server:** `bash npm start`

By following these steps, you can deploy the emedical platform to a production environment and make it accessible to users. ```