

BOOK A DOCTOR APPLICATION

Team Members :

- 1. Dinesh M - Frontend , Backend**
- 2. Hari Krishnan D - Backend**
- 3. Vignesh M – Admin**
- 4. Sareesh K G - Testing**

1) PROJECT OVERVIEW:

The **Book a Doctor Appointment Platform** is also called as “DocOnCall” is an innovative solution built using the MERN stack, designed to simplify and enhance the process of connecting patients with healthcare providers. It offers a seamless, user-friendly experience for patients and robust management tools for doctors and administrators. The platform focuses on scalability, responsiveness, and secure data handling to ensure an efficient and trustworthy service.

- **Frontend(React.js):**
Provides a responsive and intuitive interface for browsing doctor profiles, booking appointments, managing schedules, and ensuring accessibility across all devices.
- **Backend(Node.js&Express.js):**
Powers API endpoints for user authentication, appointment booking, notifications, and secure data handling. Supports real-time updates for doctor availability, and appointment statuses to deliver a smooth user experience.
- **DoctorDashboard:**
Simplifies schedule management with tools for tracking appointments, accessing and handling feedback.
- **PatientDashboard:**
Enables easy appointment tracking, doctor search, and access to medical history while providing reminders for upcoming consultations.
- **AdminPanel:**
Facilitates platform management with tools for overseeing user accounts, managing appointments, analyzing system usage, and handling doctor-patient feedback.
- **Database(MongoDB):**
A scalable and secure solution for managing doctor profiles, patient records, appointments, and reviews, ensuring data integrity and reliability.

I. Purpose of the Project :

The Book a Doctor Appointment Platform is designed to bridge the gap between patients and healthcare providers, offering a streamlined, efficient, and user-friendly experience. The platform aims to:

- Simplify the process of booking appointments with healthcare professionals.
- Enhance accessibility to healthcare services with real-time availability and scheduling.
- Provide doctors with tools to manage appointments, schedules, and patient history effectively.
- Deliver a secure, scalable, and responsive system for managing healthcare interactions online.
- Offer administrators the ability to oversee and optimize platform usage, ensuring quality service for both patients and doctors.

II. Key Features:

For Patients:

- **User Registration & Login:** Secure authentication system for creating and managing accounts.
- **Doctor Search:** Search and filter doctors by specialization, location, ratings, and availability.
- **Appointment Booking:** Real-time booking with instant confirmation for available slots.
- **Appointment Management:** View, cancel, or reschedule appointments conveniently.
- **Notifications:** Receive email or SMS alerts for booking confirmations and appointment reminders.

For Doctors:

- **Profile Management:** Manage personal profiles, including qualifications, specialization, and fees.
- **Schedule Management:** Update availability and view upcoming appointments in real-time.
- **Income Tracking:** Track revenue from completed appointments.

For Administrators:

- **User Management:** Oversee and manage patient and doctor accounts.

- **Appointment Monitoring:** Track and analyze bookings, cancellations, and system usage.
- **Platform Analytics:** Gain insights into user behavior, growth, and revenue.

General Features:

- **Responsive Design:** Accessible on all devices, ensuring usability and convenience.
- **Secure Payments:** Integration with payment gateways for online consultation fees.
- **Data Security:** End-to-end encryption and secure handling of sensitive medical information.

2) ARCHITECTURE :

Frontend: React.js Architecture

The frontend is built using **React.js** to ensure a dynamic, responsive, and user-friendly interface. The key components and architecture include:

- **Component-Based Design:**
 - Reusable components like Header, Footer, Banner, Navbar, RelatedDoctor etc.
 - Structured component hierarchy for better maintainability and scalability.
- **State Management:**
 - **React Context API** to manage global states like user authentication, appointment data, and theme preferences (light/dark mode).
- **Routing:**
 - **React Router** for SPA navigation, enabling features like protected routes for patient and doctor dashboards.
- **API Integration:**
 - Uses **Axios** or **Fetch API** to interact with backend APIs for fetching doctor profiles, booking appointments, and managing user sessions.
- **Responsive Design:**
 - Styled using **CSS frameworks** like Tailwind CSS for ensuring seamless usability across devices.

Backend: Node.js and Express.js Architecture

The backend is powered by **Node.js** and **Express.js**, providing a robust and secure API layer. Key elements include:

- **RESTful API Design:**
 - Endpoints like /api doctors, /api appointments, /api users, etc....
 - Follows REST principles for modular and scalable API structure.
- **Authentication & Authorization:**
 - Implemented using **JWT (JSON Web Tokens)** for secure token-based authentication.
 - Role-based access control (RBAC) for separating patient, doctor, and admin functionalities.
- **Middleware:**
 - Custom middleware for request validation, error handling, and logging.
 - Third-party middleware like **Multer** for handling file uploads (e.g., profile images).
- **Routes:**
 - Maps API endpoints to controller functions.
- **Payment Integration :**
 - Secure payment gateway APIs like **Stripe** or **RazorPay** for consultation fees.

Database: MongoDB Schema & Interactions

The database schema is designed in **MongoDB** to ensure scalability and flexibility, with well-structured collections and relations:

- **Collections & Schemas:**
 1. **Users Collection:**
 - userId: Unique identifier
 - name, email, password, role (patient/doctor/admin)
 - contact, profileImage, etc.
 2. **Doctors Collection:**

- doctorId: Unique identifier
- name, specialization, experience, availability (schedule), fees

3. **Appointments Collection:**

- appointmentId: Unique identifier
- patientId, doctorId (references to Users and Doctors)
- date, time, status (e.g., booked, cancelled, completed)

4. **Admin Collection:**

- Handles data related to platform analytics and user management.

• **Database Interactions:**

○ **CRUD Operations:**

- Use of **Mongoose** for defining schemas and performing database operations.

○ **Indexing:**

- Index on doctorId, patientId, and date fields for fast query performance.

○ **Data Validation:**

- Use of **Mongoose Validators** for ensuring data integrity (e.g., valid email format, unique IDs).

3) **SETUP INSTRUCTIONS :**

I. Prerequisites

Before you begin, ensure you have the following software installed:

1. **Node.js** (v16 or higher)
 - [Download and install Node.js](#)
2. **MongoDB** (latest version)

- Install MongoDB locally or use a cloud-based service like [MongoDB Atlas](#).

3. Git

- [Download and install Git](#).

4. Package Manager

- **npm** (bundled with Node.js) or **yarn**.

5. Code Editor

- Recommended: [VS Code](#).

II. Installation Steps

i. Clone the Repository:

```
git clone <repository-url>
cd <project-folder>
```

ii. Install Dependencies:

- Navigate to the **frontend** folder and install dependencies:

```
cd frontend
npm install
```

- Navigate to the **backend** folder and install dependencies:

```
cd backend
npm install
```

iii. Set Up Environment Variables:

- Create a .env file in the **backend** folder and add the following variables:

```
PORT=5000
MONGO_URI=<Your MongoDB Connection String>
JWT_SECRET=<Your JWT Secret>
NODE_ENV=development
```

iv. Run the Application:

- Start the backend server:

```
cd backend
```

npm run server

- Start the frontend server:

cd frontend

npm run dev

v. Access the Application:

- Open your browser and navigate to:
 - Frontend: <http://localhost:5173>
 - Backend (API): <http://localhost:4000>
 - Admin: <http://localhost:5174>

4) FOLDER STRUCTURE :

- Frontend (Client):
 - node_modules: npm packages for the frontend.
 - public: Static files like index.html and assets.
 - Components: Reusable UI components (e.g., Navbar, Footer).
 - Pages: Specific pages (e.g., About, Appointment, Contact, etc....).
 - Context: Manages state (e.g., AppContext.jsx).
 - Css: Styling files (e.g., index .css).
 - .gitignore: Specifies files to ignore by Git.
 - package.json: Lists dependencies and scripts.
 - package-lock.json: Ensures consistent dependency versions.
- Backend (Server):
 - node_modules: npm packages for the backend.
 - index.js: Server entry point with config, middleware, and routing.
 - package.json: Lists backend dependencies and scripts.
 - package-lock.json: Ensures consistent backend dependency installation.
- Admin :
 - public: Static files like index.html and assets.
 - Components: Reusable UI components (e.g., Navbar, Sidebar).
 - Pages: Specific pages (e.g., Admin, Doctor, etc....).
 - Context: Manages state (e.g., AppContext.jsx, etc....).
 - Css: Styling files (e.g., index .css).
 - .gitignore: Specifies files to ignore by Git.
 - package.json: Lists dependencies and scripts.

- package-lock.json: Ensures consistent dependency versions.

5) **RUNNING THE APPLICATION :**

- To run the application locally, you'll need to start both the frontend and backend servers. Follow the commands below to launch each part of the application:

Frontend :

1. Navigate to the client directory:
`cd frontend`
2. Start the React development server:
`npm start`

This will run the frontend application on : <http://localhost:5173>

Backend (Node.js) :

1. Navigate to the server directory:
`cd backend`
2. Start the Node.js server:
`npm run server`

This will run the backend server on : <http://localhost:4000>

ADMIN :

1. Navigate to the server directory:
`cd admin`
2. Start the Node.js server:
`npm run dev`

This will run the backend server on: <http://localhost:5174>

6) **API DOCUMENTATION :**

- **General Routes**

1. Root Endpoint

Endpoint: /

Method: GET

Description: Verifies the API is working.

Example Response:


```
{  
  "message": "API Working"  
}
```

- **Admin Routes**

These routes handle admin-related functionalities.

- **Base Path:** /api/admin

2. GET /api/admin

- **Method:** GET
- **Description:** Retrieves all admin-related data. (Example functionality placeholder)

```
{  
  "admins": [  
    {  
      "id": "admin123",  
      "name": "Admin User",  
      "email": "admin@example.com"  
    }  
  ]  
}
```

3. POST /api/admin

- Method: POST
- Description: Creates a new admin.
- Request Body :

```
{  
  "name": "Admin User",
```

```
"email": "admin@example.com",  
"password": "securepassword123"  
}
```

- **Example Response:**

```
{  
  "message": "Admin created successfully",  
  "admin": {  
    "id": "admin123",  
    "name": "Admin User",  
    "email": "admin@example.com"  
  }  
}
```

4.PUT /api/admin/:id

- Method: PUT
- Description: Updates admin details by ID.
- Request Parameters:
 - id (string): Admin ID to update.
- Request Body:

```
{  
  "name": "Updated Admin Name",  
  "email": "updatedadmin@example.com"  
}
```

- **Example Response:**

```
{  
  "message": "Admin updated successfully",
```

```
"admin": {  
  "id": "admin123",  
  "name": "Updated Admin Name",  
  "email": "updatedadmin@example.com"  
}  
}
```

5. DELETE /api/admin/:id

- Method: DELETE
- Description: Deletes an admin by ID.
- Request Parameters:
 - id (string): Admin ID to delete.
- Example Response:

```
{  
  
  "message": "Admin deleted successfully"  
}
```

• Doctor Routes

These routes handle doctor-related functionalities.

- Base Path: /api/doctor

1. GET /api/doctor

- Method: GET
- Description: Retrieves all registered doctors.
- Example Response:

```
{  
  
  "doctors": [  
  
    {  
  
      "id": "doc123",
```

```
"name": "Dr. Alice Smith",  
"specialization": "Cardiology",  
"availability": ["Monday", "Wednesday", "Friday"]  
}  
]  
}
```

2. POST /api/doctor

- Method: POST
- Description: Adds a new doctor profile.
- Request Body:

```
{  
"name": "Dr. Alice Smith",  
"specialization": "Cardiology",  
"availability": ["Monday", "Wednesday", "Friday"]  
}
```

- **Example Response:**

```
{  
"message": "Doctor added successfully",  
"doctor": {  
  "id": "doc123",  
  "name": "Dr. Alice Smith",  
  "specialization": "Cardiology",  
  "availability": ["Monday", "Wednesday", "Friday"]  
}
```

```
}
```

3. GET /api/doctor/:id

- Method: GET
- Description: Retrieves details of a specific doctor by ID.
- Request Parameters:
 - id (string): Doctor ID.

```
{
```

```
"id": "doc123",
```

```
"name": "Dr. Alice Smith",
```

```
"specialization": "Cardiology",
```

```
"availability": ["Monday", "Wednesday", "Friday"]
```

```
}
```

4. PUT /api/doctor/:id

- Method: PUT
- Description: Updates a doctor's details.
- Request Parameters:
 - id (string): Doctor ID to update.
- Request Body:

```
{
```

```
"specialization": "General Medicine",
```

```
"availability": ["Tuesday", "Thursday"]
```

```
}
```

- **Example Response:**

```
{
```

```
"message": "Doctor updated successfully",
```

```
"doctor": {
```

```
"id": "doc123",  
  
"name": "Dr. Alice Smith",  
  
"specialization": "General Medicine",  
  
"availability": ["Tuesday", "Thursday"]  
  
}  
  
}
```

5. DELETE /api/doctor/:id

- Method: DELETE
- Description: Deletes a doctor's profile.
- Request Parameters:
 - id (string): Doctor ID.
- Example Response:

```
{  
  
"message": "Doctor deleted successfully"  
  
}
```

• User Routes

These routes handle user-related functionalities.

- **Base Path:** /api/user

1. GET /api/user

- Method: GET
- Description: Retrieves all registered users.
- Example Response:

```
{  
  
"users": [  
  
  {
```

```
    "id": "user123",  
    "name": "John Doe",  
    "email": "johndoe@example.com"  
  }  
]  
}
```

2. POST /api/user

- Method: POST
- Description: Registers a new user.
- Request Body:

```
{  
  "name": "John Doe",  
  "email": "johndoe@example.com",  
  "password": "password123"  
}
```

- **Example Response:**

```
{  
  "message": "User registered successfully",  
  "user": {  
    "id": "user123",  
    "name": "John Doe",  
    "email": "johndoe@example.com"  
  }  
}
```

3. GET /api/user/:id

- Method: GET
- Description: Retrieves details of a specific user by ID.
- Request Parameters:
 - id (string): User ID.

```
{  
  
  "id": "user123",  
  
  "name": "John Doe",  
  
  "email": "johndoe@example.com"  
}
```

4. PUT /api/user/:id

- Method: PUT
- Description: Updates user details by ID.
- Request Body:

```
{  
  
  "name": "John Doe Updated",  
  
  "email": "updatedjohn@example.com"  
}
```

- **Example Response:**

```
{  
  
  "message": "User updated successfully",  
  
  "user": {  
  
    "id": "user123",  
  
    "name": "John Doe Updated",  
  
    "email": "updatedjohn@example.com"
```



```
}
```

```
}
```

5. DELETE /api/user/:id

- Method: DELETE
- Description: Deletes a user profile by ID.
- Request Parameters:
 - id (string): User ID.
- Example Response:

```
{
```

```
  "message": "User deleted successfully"
```

```
}
```

- **Error Responses**

All endpoints return a standardized error response in case of invalid requests or server errors:

```
{
```

```
  "error": "Resource not found",
```

```
  "status": 404
```

```
}
```

7. AUTHENTICATION :

Registration:

Whenever a user signs up, their password is hashed using bcrypt and securely stored in the database.

Login:

After successful login, a JWT is generated for the user, containing their user ID, and signed with a secret key.

TokenUsage:

For protected requests, the JWT is sent in the Authorization header as a Bearer token (Bearer <token>).

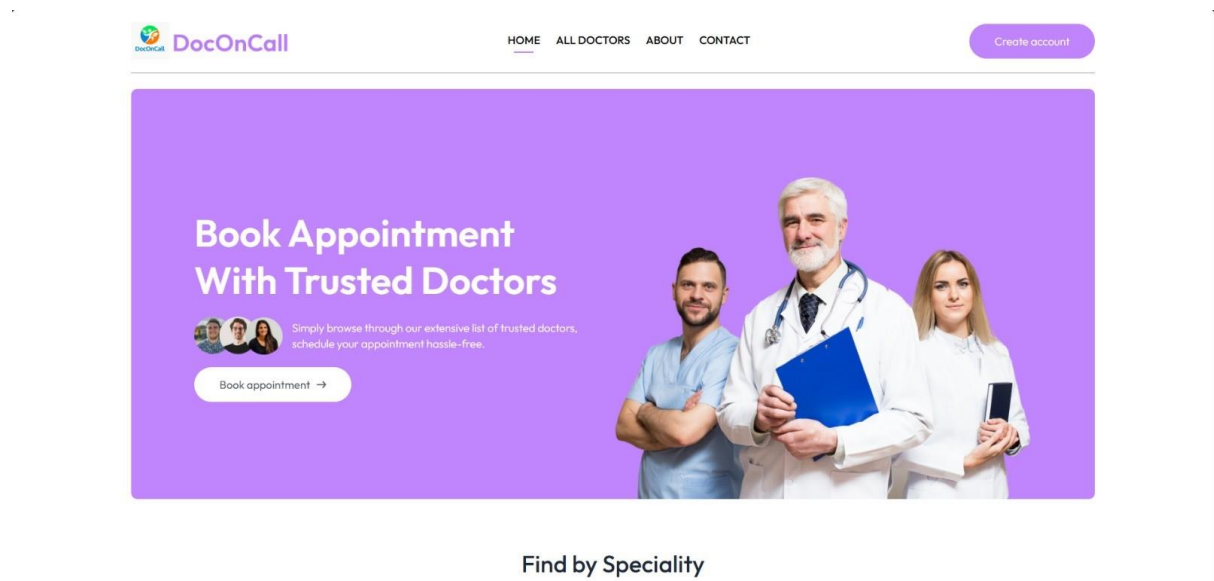
AuthenticationMiddleware:

This middleware function validates the JWT token and ensures that only authenticated users can access protected routes.

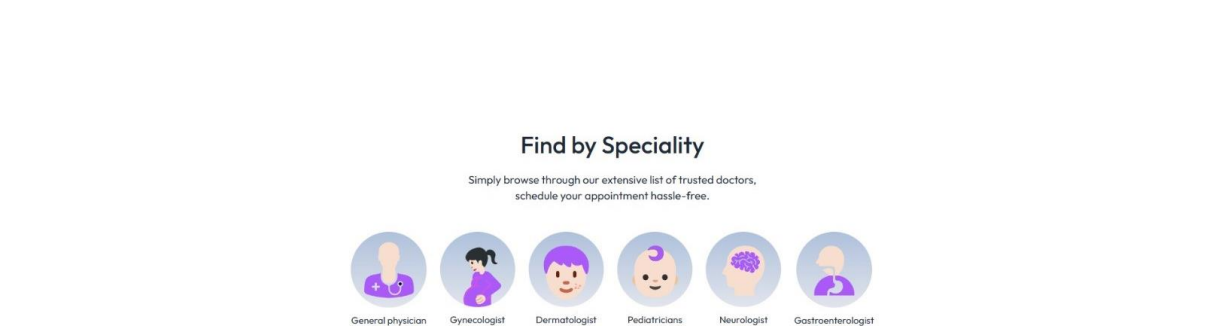
8. USER INTERFACE:

Screenshots of UI:


Home Page: Displays all available doctors with speciality and category filter options.




Speciality: Detailed view of the doctors with an Speciality Navigation button.





Detail of the Doctor: When the doctor is selected, his details and availability will be displayed.

 **DocOnCall**


HOME ALL DOCTORS ABOUT CONTACT





Dr. Vishnu 

MBBS - General physician 4 Year


About 

Dr. Vishnu is a compassionate gynecologist dedicated to supporting womens health and well-being at every stage of life. With expertise in diagnosing and managing a range of gynecological conditions,he adopts a patient-centered approach, ensuring that each individual receives personalized care tailored to their unique needs. Dr. Vishnu fosters a supportive and open environment, encouraging women to engage actively in their health decisions. He commitment to preventive care and education empowers he patients to maintain optimal reproductive health and overall wellness, enhancing their quality of life

Appointment fee: ₹600

Select a date:

18-11-2024



Available Time Slots:

09:30 am

10:00 am

10:30 am

11:00 am

11:30 am

12:00 pm

12:30 pm

01:00 pm


01:30 pm

Book an appointment

Related Doctors

Here are some of our extensive list of trusted doctors.

Sign Up: Here new users can register themselves.

 **DocOnCall**

HOME ALL DOCTORS ABOUT CONTACT

Create account

Create Account

Please sign up to book appointment

Full Name

Email

Password

Create Account

Already have an account? [Login here](#)

Login Page: Here already registered users can use their login credentials.

Login

Please log in to book appointment

Email

Password











Login

Create a new account? [click here](#)

All Doctors List View: Here all the doctors list who are available will be displayed.

Top Doctors to Book

Here are some of our extensive list of trusted doctors.

 • Available Dr. Vishnu General physician	 • Available Dr. Yumeko Gynecologist	 • Available Dr. Yuvan Dermatologist	 • Available Dr. Aarav Pediatricians	 • Available Dr. Preetha Gastroenterologist
 • Available Dr. Dhruv	 • Available Dr. Joseph Vijay	 • Available Dr. Nobita	 • Available Dr. Diya	 • Available Dr. Rudra

9. TESTING:

The testing strategy for the Book a Doctor platform includes the following components:

1. Unit Testing:

- Mocha and Chai are used to verify individual modules/functions.

2. Static Analysis:

- ESLint ensures code adheres to style and quality guidelines.

3. Code Coverage:

- NYC tracks code coverage during testing to measure test coverage.

4. Script-Based Testing Workflow:

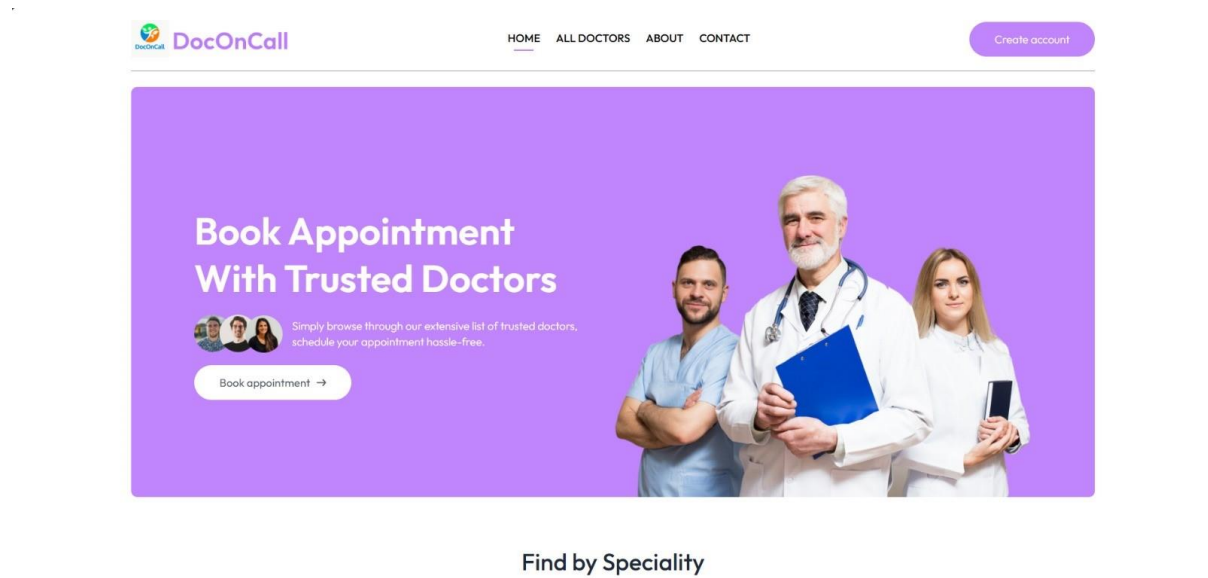
- "test": Runs test cases (via Mocha).
- "coverage": Generates code coverage reports (via NYC).
- "lint": Runs ESLint for code linting.

Tools Used:

- **Mocha:** JavaScript test framework for asynchronous testing.
- **Chai:** Assertion library for tests.
- **ESLint:** Identifies and fixes JavaScript code issues.
- **nyc:** Tracks code coverage.

10. SCREENSHOTS OR DEMO

Home:



Sign Up:

Create Account

Please sign up to book appointment

Full Name

Email

Password

Create Account

Already have an account? [Login here](#)

Login:

Login

Please log in to book appointment

Email

Password

Login

Create an new account? [click here](#)

Speciality View:

Find by Speciality

Simply browse through our extensive list of trusted doctors,
schedule your appointment hassle-free.



General physician



Gynecologist



Dermatologist



Pediatricians



Neurologist




Gastroenterologist


All Doctors View:

Top Doctors to Book


Here are some of our extensive list of trusted doctors.




● Available
Dr. Vishnu
General physician




● Available
Dr. Yumeko
Gynecologist




● Available
Dr. Yuvan
Dermatologist




● Available
Dr. Aarav
Pediatricians




● Available
Dr. Preetha
Gastroenterologist




● Available
Dr. Dhruv




● Available
Dr. Joseph Vijay



● Available
Dr. Nobita




● Available
Dr. Diya





● Available
Dr. Rudra

Doctor's Portfolio:

 **DocOnCall**

HOME ALL DOCTORS ABOUT CONTACT



Dr. Vishnu 


MBBS - General physician 4 Year

About ⓘ

Dr. Vishnu is a compassionate gynecologist dedicated to supporting women's health and well-being at every stage of life. With expertise in diagnosing and managing a range of gynecological conditions, he adopts a patient-centered approach, ensuring that each individual receives personalized care tailored to their unique needs. Dr. Vishnu fosters a supportive and open environment, encouraging women to engage actively in their health decisions. His commitment to preventive care and education empowers his patients to maintain optimal reproductive health and overall wellness, enhancing their quality of life.

Appointment fee: ₹600

Select a date:

18-11-2024 

Available Time Slots:

09:30 am

10:00 am

10:30 am

11:00 am

11:30 am

12:00 pm

12:30 pm

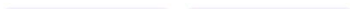
01:00 pm

01:30 pm

Book an appointment

Related Doctors

Here are some of our extensive list of trusted doctors.



About Us:

ABOUT US



Welcome to DocOnCall, your dedicated ally in healthcare accessibility and convenience. At DocOnCall, we understand the complexity of managing health amidst today's fast-paced lifestyle. That's why we're committed to delivering a platform that seamlessly connects you to trusted healthcare providers, allowing you to schedule appointments, access medical records, and consult with doctors, all from the comfort of your home.

We prioritize your health and well-being, continuously enhancing our platform with cutting-edge technology to ensure a smooth, efficient, and reliable experience. Whether you're booking an initial consultation, following up on treatment, or managing chronic care, DocOnCall is here to empower and support your healthcare journey.

Our Vision

At DocOnCall, our vision is to transform healthcare accessibility by bridging the gap between patients and medical professionals. We aim to create a future where healthcare is as accessible as possible, giving everyone the tools and support to prioritize their health without barriers. Through our innovative platform, we strive to redefine healthcare convenience and be a trusted partner in every user's wellness journey.

WHY CHOOSE US

Efficiency:

Streamlined appointment scheduling that fits into your busy lifestyle.

Convenience:

Access to a network of trusted healthcare professionals in your area.

Personalization:

Tailored recommendations and reminders to help you stay on top of your health.

Contact Us:

CONTACT US



Our OFFICE

DocOnCall Headquarters
2021 HealthTech Avenue,
Chennai, TamilNadu,
India

Tel: +1800-202-7031
Email: support@doconcall.com

Careers at DocOnCall

Learn more about our teams and job openings.

[Explore Jobs](#)

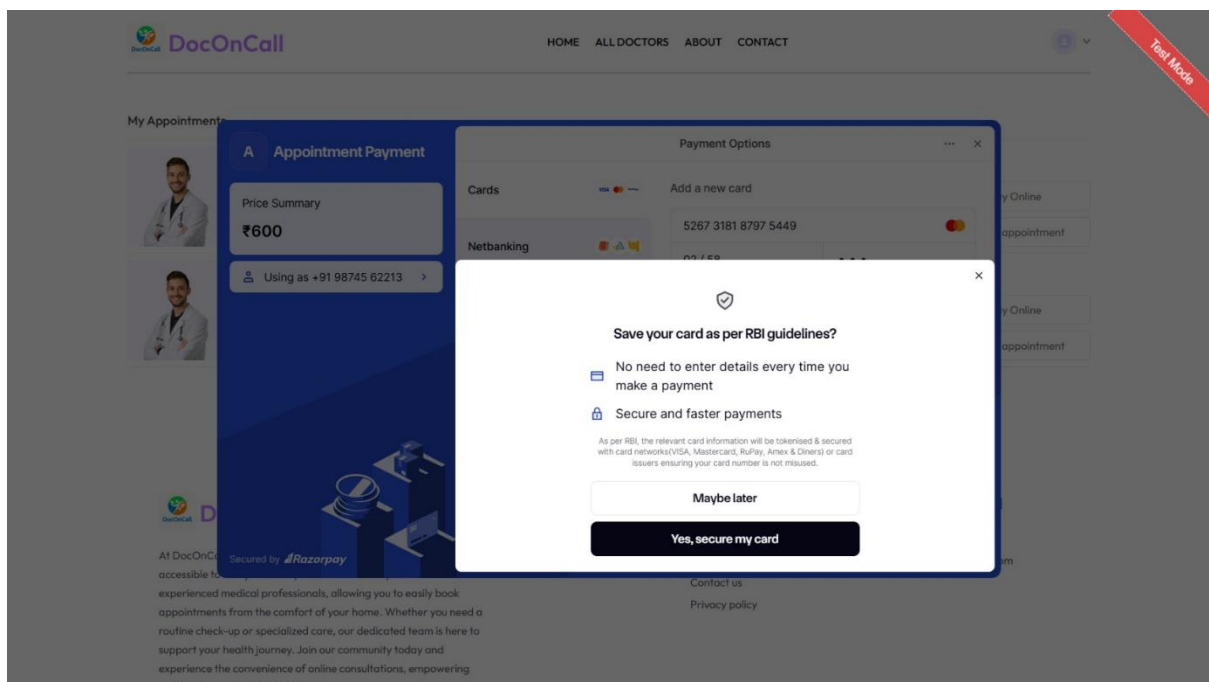
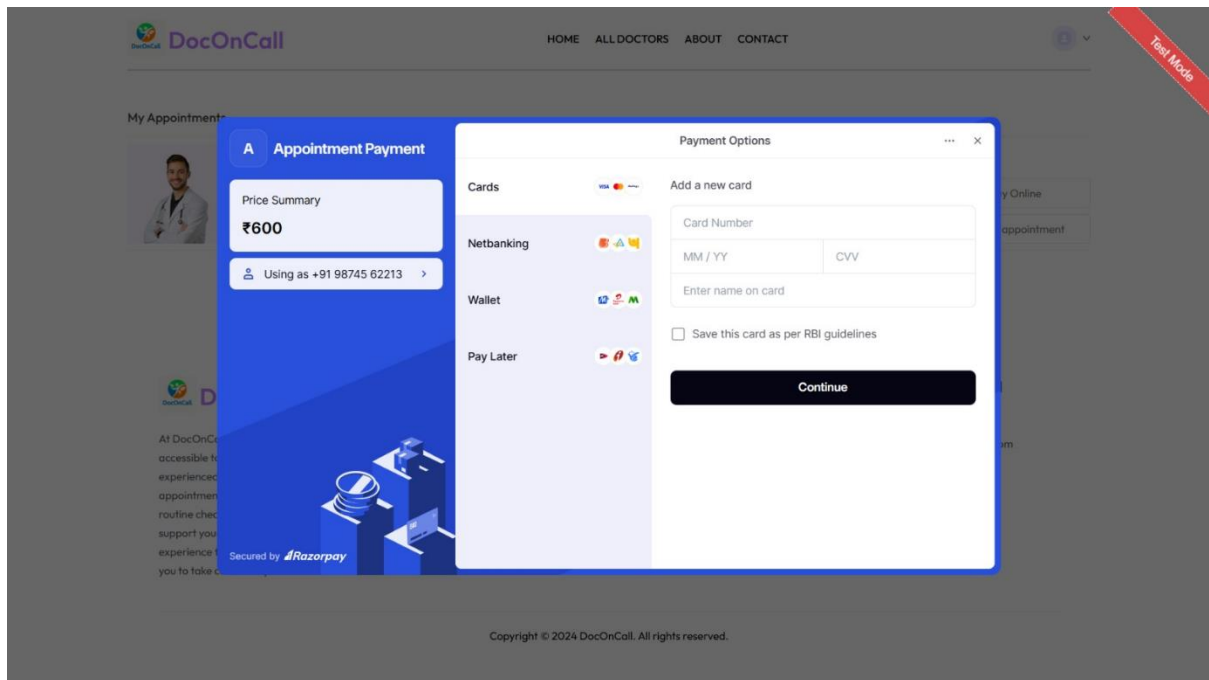
COMPANY

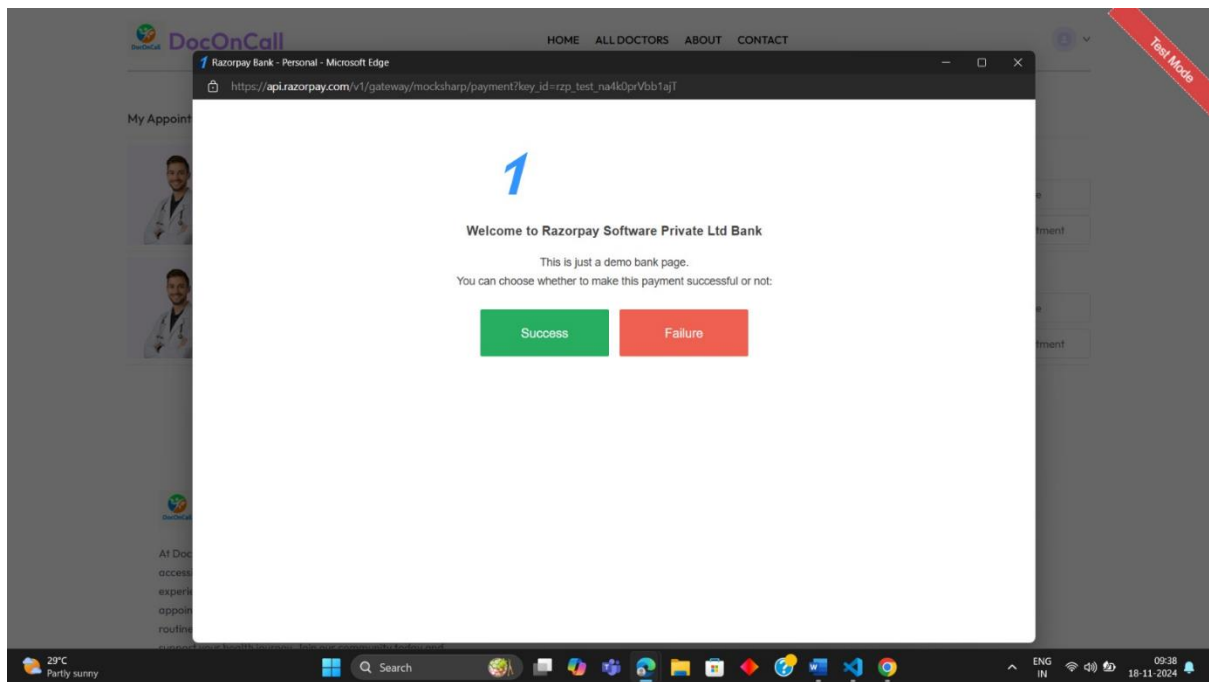
[Home](#)
[About us](#)
[Contact us](#)
[Privacy policy](#)

GET IN TOUCH

+1800-202-7031
support@doconcall.com

Payment Panel:





Admin Panel:

Admin Login

Email

Password

Login

Doctor Login? [Click here](#)

Doctor Login

Email

Password

Login

Admin Login? [Click here](#)

DocOnCall

Dashboard Panel

Doctor

Logout

Dashboard

Appointments

Profile

₹0

Earnings

0

Appointments

0

Patients

Latest Bookings

DocOnCall

Dashboard Panel

Admin

Logout

Dashboard

Appointments

Add Doctor

Doctors List

25

Doctors

2

Appointments

4

Patients

Latest Bookings

Dr. Vishnu

17 Nov 2024

Dr. Laya

7 Nov 2024

Cancelled

DocOnCall

Dashboard Panel

Admin

Logout

Dashboard

Appointments

Add Doctor

Doctors List

All Appointments

#	Patient	Age	Date & Time	Doctor	Fees	Actions
1	<div><div></div>Dinesh M</div>	21	7 Nov 2024, 11:30 am	<div><div></div>Dr. Laya</div>	₹500	Cancelled
2	<div><div></div>Dinesh</div>	NaN	17 Nov 2024, 09:30 pm	<div><div></div>Dr. Vishnu</div>	₹600	

DocOnCall

Dashboard Panel

Admin

Logout

Dashboard

Appointments

Add Doctor

Doctors List

Add Doctor

Doctor name

Name

Speciality

General physician

Doctor Email

Email

Education

Education

Doctor Password

Password

Address

address 1

address 2

Experience

1 Year

Fees

fees

About Doctor

write about doctor

Add doctor

11. KNOWN ISSUES:

Patient History: While registering in the application, patient history is not required and when diagnosing the problem it might be an issue.

Ratings and Review: Rating and review is not present for both patients and the doctors, when choosing a doctor it might be an issue.

JWT Token Expiry Handling: Expired tokens not detected, causing authorization errors.

12. FUTURE ENHANCEMENTS

- **User Experience:** AI-driven doctor recommendations, improved appointment search filters, and personalized health tips for users.
- **Security:** Two-factor authentication (2FA) for enhanced account security, encrypted medical records, and additional protection against DDoS attacks.
- **Scalability & Performance:** Implementing server-side rendering (SSR) for faster page loads, database sharding for large datasets, and Redis caching for optimized performance during peak usage.
- **Advanced Features:** Patient health record history, doctor reviews and ratings, in-app video consultations, and appointment reminders via SMS/Email.
- **Mobile & Analytics:** PWA support for offline functionality, dedicated mobile app development, and real-time admin dashboards for tracking patient bookings and system health.