**Patient Booking System** using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**

**Patient Booking System – Features**

**Users**

1. **Patient** – can register/login, book appointments, see appointment history.
2. **Admin / Doctor** – can view appointments, approve/reject, manage slots.

**📐 Architecture Overview**

Frontend (React) ↔ Backend (Node.js + Express) ↔ Database (MongoDB)

**📦 Tech Stack**

* **Frontend**: React, Tailwind CSS or Bootstrap
* **Backend**: Node.js + Express.js
* **Database**: MongoDB
* **Authentication**: JWT + bcrypt
* **Date & Time Handling**: Moment.js or Day.js
* **Optional**: Redux for state management

**🛠️ Functional Modules**

**1. Authentication**

* Register/Login (Patient & Admin)
* Password encryption (bcrypt)
* Token-based auth (JWT)

**2. Patient Dashboard**

* Book Appointment (select doctor, date, time)
* View Appointment History
* Cancel Appointment

**3. Admin/Doctor Dashboard**

* View all Appointments
* Approve/Reject bookings
* View patient list

**4. Appointment Slot Management**

* Admin creates available time slots
* Patients can only choose from available slots

**🗂️ MongoDB Schema Examples**

**User Schema**

{

name: String,

email: String,

password: String,

role: { type: String, enum: ['patient', 'admin'], default: 'patient' }

}

**Appointment Schema**

{

patientId: ObjectId,

doctorId: ObjectId,

date: Date,

timeSlot: String,

status: { type: String, enum: ['pending', 'approved', 'rejected'], default: 'pending' }

}

**🌐 API Endpoints (Express.js)**

**Auth**

* POST /api/auth/register
* POST /api/auth/login

**Appointments**

* POST /api/appointments/book
* GET /api/appointments/mine
* GET /api/appointments/all (Admin)
* PUT /api/appointments/:id/status

**Time Slots**

* POST /api/slots/create
* GET /api/slots/available?date=...

**🖥️ UI Pages (React.js)**

1. **Home**
2. **Register/Login**
3. **Patient Dashboard**
   * Book Appointment Form
   * Appointment List
4. **Admin Dashboard**
   * View All Appointments
   * Approve/Reject Buttons
5. **404 / Unauthorized Pages**

**📁 Folder Structure**

client/

├── src/

│ ├── components/

│ ├── pages/

│ ├── services/ (API calls)

│ └── App.js

server/

├── models/

├── routes/

├── controllers/

├── middleware/

└── index.js

**✅ Bonus Features**

* Email confirmation/reminders (Nodemailer)
* Role-based route protection (private routes)
* Appointment rescheduling
* Search/filter appointments

**USERS**

[ Patient ] [ Admin / Doctor ]

| |

| |

v v

[ React Frontend (UI)] [React Admin Panel]

| |

+------------+---------------+

|

v

[ Node.js + Express Backend ]

|

+------------+-------------+

| |

v v

[ MongoDB: Users Collection ] [ MongoDB: Appointments Collection ]

[ MongoDB: Slots Collection ]

[ MongoDB: Doctor Profile Collection ]

**MODULE FLOW (Functional Flow Description)**

[Patient Registers/Login]

|

v

[Token Created (JWT) + Stored in LocalStorage]

|

v

[Book Appointment Form]

|

v

[Fetch Available Slots from Backend]

|

v

[Select Doctor + Date + Time]

|

v

[Submit Booking -> POST to /api/appointments]

|

v

[Store in MongoDB with status = 'pending']

**Admin / Doctor Flow**

[Admin Login]

|

v

[Dashboard View]

|

v

[GET /api/appointments]

|

v

[List All Appointments (Pending, Approved, Rejected)]

|

v

[Approve or Reject -> PUT /api/appointments/:id/status]

**DATABASE STRUCTURE (Collections & Relationships)**

#### Users Collection (users)

json

CopyEdit

{

\_id: ObjectId,

name: "John Doe",

email: "john@example.com",

password: "<hashed>",

role: "patient" | "admin"

}

#### Appointments Collection (appointments)

json

CopyEdit

{

\_id: ObjectId,

patientId: ObjectId, // from users

doctorId: ObjectId, // from users

date: "2025-08-01",

timeSlot: "10:00 AM",

status: "pending" | "approved" | "rejected",

reason: "Fever"

}

#### Slots Collection (slots)

json

CopyEdit

{

\_id: ObjectId,

doctorId: ObjectId,

date: "2025-08-01",

availableSlots: ["09:00 AM", "10:00 AM", "11:00 AM"]

}

#### Doctor Profiles Collection (doctors)

json

CopyEdit

{

\_id: ObjectId,

userId: ObjectId, // Linked to users

specialization: "Cardiology",

experience: "10 years"

}

**API STRUCTURE (Express.js Routes)**

#### Auth Routes

bash

CopyEdit

POST /api/auth/register

POST /api/auth/login

GET /api/auth/profile

#### Appointment Routes

bash

CopyEdit

POST /api/appointments → Book appointment

GET /api/appointments/mine → Get logged-in patient’s appointments

GET /api/appointments → Admin/doctor gets all appointments

PUT /api/appointments/:id → Approve/Reject appointment

#### Slot Routes

bash

CopyEdit

POST /api/slots → Admin creates time slots

GET /api/slots?date=... → Patient fetches available slots

**FRONTEND COMPONENTS (React)**

#### 🖥️ Patient Panel

* Register/Login Page
* Book Appointment Form
* My Appointments Page

#### ⚙️ Admin Panel

* Login Page
* Appointments Dashboard (with Approve/Reject)
* Slot Management Page

**Patient Flowchart (Left Branch)**

Start

|

v

[Register / Login]

|

v

[Token Created (JWT) and Stored in LocalStorage]

|

v

[Open Book Appointment Form]

|

v

[Fetch Available Slots from Server]

|

v

┌─────────────────────────────┐

│ Are Slots Retrieved? │

└─────────────────────────────┘

| Yes | No

v v

[Select Doctor, Date & Time] [Show Error / Retry]

|

v

[Submit Booking Request]

|

v

[Store Appointment with Status = "Pending"]

|

v

End

**Admin / Doctor Flowchart (Right Branch)**

Start

|

v

[Login]

|

v

[View Appointment Requests]

|

v

┌─────────────────────────────┐

│ Approve or Reject Booking? │

└─────────────────────────────┘

| Approve | Reject

v v

[Update Appointment [Update Appointment

Status = Approved] Status = Rejected]

| |

v v

End End

## Interaction Between Flows

* When the patient submits a booking, the backend:
  + Saves the appointment as "pending" in MongoDB.
  + Notifies or lists the booking on the Admin panel.
* When the Admin/Doctor approves or rejects:
  + The appointment status is updated accordingly.
  + Optionally, a notification/email can be sent back to the patient.