# **Author: Oh Hyun Wo (ohyunwoods663@gmail.com) | https://github.com/punixcorn**

# Hospital Backend System Documentation

This document outlines the complete backend architecture of our hospital system, covering the data models, API endpoints, dynamic scheduling logic, and deployment considerations.

**Note:**

* Our backend is built with Node.js and Express, written in TypeScript, and uses MongoDB (via Mongoose) for data storage.
* Authentication is handled via JWT tokens (often stored as HTTP-only cookies) with role-based access controls.
* The system is modularized into separate domains for Auth, Doctor, Patient, and Notes functionality.
* This website is not production ready.
* Most of the Api’s listed here may or may not be present or and functionality may have not been fully confirmed, due to lack of time this was rushed.

[**Author: Oh Hyun Wo (ohyunwoods663@gmail.com) | https://github.com/punixcorn**](#_Toc1)

[Hospital Backend System Documentation](#_Toc2)

[The Expose endpoints:](#_Toc3)

[Patient doctor selection.](#_Toc4)

[Doctor retrieval of their patient list.](#_Toc5)

[Submitting doctor notes and processing actionable steps.](#_Toc6)

[Retrieving actionable steps and reminders.](#_Toc7)

[1. Models](#_Toc8)

[**User Model**](#_Toc9)

[**Assignment Model**](#_Toc10)

[**Note Model**](#_Toc11)

[**Session Model**](#_Toc12)

[**Verification Model**](#_Toc13)

[2. API Endpoints](#_Toc14)

[**Authentication Endpoints**](#_Toc15)

[**POST /api/auth/signup**](#_Toc16)

[**POST /api/auth/login**](#_Toc17)

[**Doctor Endpoints**](#_Toc18)

[**GET /api/doctor/get-patients**](#_Toc19)

[**POST /api/doctor/select-patient**](#_Toc20)

[**Patient Endpoints**](#_Toc21)

[**GET /api/patient/available-doctors**](#_Toc22)

[**POST /api/patient/select-doctor**](#_Toc23)

[**Notes Endpoints**](#_Toc24)

[**POST /api/notes/add\_note\_task**](#_Toc25)

[**GET /api/notes/get\_my\_tasks**](#_Toc26)

[3. Dynamic Scheduling & Task Lifecycle](#_Toc27)

[**Dynamic Scheduling**](#_Toc28)

REQUIREMENTS :  
As per requirements the following endpoints

## The Expose endpoints:

User signup and authentication.

* /api/auth/signup
* /api/auth/login
* /api/auth/logout (out of scope)

### Patient doctor selection.

* /api/patient/available-doctors
* /api/patient/select-doctor

### Doctor retrieval of their patient list.

* /api/doctor/get\_patients

### Submitting doctor notes and processing actionable steps.

* /api/notes/add\_note\_task

### Retrieving actionable steps and reminders.

* /api/notes/get\_my\_tasks

# 1. Models

### **User Model**

**File:** src/models/user.model.ts

**Purpose:**Stores all user data, including both patients and doctors. The boolean field role distinguishes between them:

* role: true → Doctor
* role: false → Patient

**Schema Fields:**

* **email:** String (unique, required) – User’s email address.
* **password:** String (required) – Password (hashed via a pre-save bcrypt hook).
* **verified:** Boolean (default: false) – Indicates if the email is verified.
* **role:** Boolean (required) – Indicates user type.
* **Timestamps:** Automatically maintained createdAt and updatedAt.

**Instance Methods:**

* comparePassword(password: string): Promise<boolean> – Compares a candidate password with the stored hash.
* omitPassword(): object – Returns a user object with sensitive data (password) omitted.

### **Assignment Model**

**File:** src/models/assignment.model.ts

**Purpose:**Maps doctors to their patients. When a patient selects a doctor, their ID is added to the doctor’s Assignment document.

**Schema Fields:**

* **doctor:** ObjectId (ref: "User", required) – ID of the doctor.
* **patients:** Array of ObjectId (ref: "User", required) – IDs of patients who selected this doctor.
* **Timestamps:** createdAt and updatedAt are maintained automatically.

### **Note Model**

**File:** src/models/notes.model.ts

**Purpose:**Holds a doctor’s note for a patient along with the LLM-processed actionable steps.

**Schema Fields:**

* **doctor\_id:** ObjectId (ref: "User", required) – Doctor who created the note.
* **patient\_id:** ObjectId (ref: "User", required) – Patient who will see the note.
* **original\_note:** String (required) – The original note text.
* **extracted\_actions:** Object – Contains LLM-processed data following the interface:
* typescript
* CopyEdit
* export interface LLMdata { actionable\_steps: { checklist: string; plan: string; number\_of\_days: number; interval\_between\_days: number; }; }
* **number\_of\_days\_left:** Number (required) – Initialized to number\_of\_days from LLMdata; decrements daily.
* **date\_of\_creation:** Date (default: Date.now) – The creation date.
* **is\_done:** Boolean (default: false) – Set to true when number\_of\_days\_left reaches 0.
* **remind\_patient\_today:** Boolean (default: false) – Flag indicating if the patient has been reminded on the current day.
* **Timestamps:** createdAt and updatedAt are automatically maintained.

### **Session Model**

**File:** src/models/session.models.ts

**Purpose:**Tracks active user sessions. Even if JWTs are used for stateless authentication, session data may be stored server-side to allow for immediate revocation or additional security checks.

**Typical Schema Fields:**

* **user:** ObjectId (ref: "User", required) – The authenticated user’s ID.
* **token:** String (required) – The session token.
* **createdAt:** Date (default: Date.now) – Session creation timestamp.
* *(Optional: Set an expiry on sessions via Mongoose’s expires option.)*

### **Verification Model**

**File:** src/models/verification.model.ts

**Purpose:**Manages user verification (e.g., email confirmation). When a user registers, a verification record is created, which is later marked as verified upon successful confirmation.

**Schema Fields:**

* **user:** ObjectId (ref: "User", required) – Associated user.
* **code:** String (required) – Verification code.
* **expiresAt:** Date (required) – When the code expires.
* **verified:** Boolean (default: false) – Whether verification has completed.
* **Timestamps:** Automatically includes createdAt and updatedAt.

# 2. API Endpoints

The API is organized by module. All endpoints assume the presence of authentication middleware that validates a JWT (and populates req.user).

### **Authentication Endpoints**

*(Located in src/api/auth)*

#### **POST /api/auth/signup**

* **Purpose:** Registers a new user.
* **Request Body Example:**
* json
* CopyEdit
* { "email": "user@example.com", "password": "yourPassword", "role": false }
* **Response:**
* **201 Created:** Returns user details (without the password).
* **Errors:** 400 for missing fields; 500 on error.

#### **POST /api/auth/login**

* **Purpose:** Authenticates a user.
* **Request Body Example:**
* json
* CopyEdit
* { "email": "user@example.com", "password": "yourPassword" }
* **Response:**
* **200 OK:** JWT is returned and stored as an HTTP-only cookie.
* **Errors:** 401 for invalid credentials; 500 on error.

### **Doctor Endpoints**

*(Located in src/api/doctor)*

#### **GET /api/doctor/get-patients**

* **Purpose:** Retrieves a list of patients who have chosen the doctor.
* **Authentication:** Requires a valid JWT for a doctor (role: true).
* **Response Example:**
* json
* CopyEdit
* { "patients": [ { "\_id": "patientId1", "email": "patient1@example.com", ... }, { "\_id": "patientId2", "email": "patient2@example.com", ... } ] }

#### **POST /api/doctor/select-patient**

* **Purpose:** Allows a doctor to send a note to a specific patient. The note is processed (via Gemini) to extract actionable steps (checklist & plan).
* **Authentication:** Requires a valid doctor token.
* **Request Body Example:**
* json
* CopyEdit
* { "patientId": "60f8a8e2c3a1b34d2c8d9f1a", "note": "Patient needs to buy drug X and take it daily for 7 days." }
* **Response Example:**
* json
* CopyEdit
* { "actionableSteps": { "checklist": ["Buy required medication"], "plan": ["Schedule daily reminder for 7 days"] } }

### **Patient Endpoints**

*(Located in src/api/patient)*

#### **GET /api/patient/available-doctors**

* **Purpose:** Lists all doctors available for selection.
* **Authentication:** Requires a valid JWT (applies to all authenticated users).
* **Response Example:**
* json
* CopyEdit
* { "doctors": [ { "\_id": "doctorId1", "name": "Dr. Smith", "email": "drsmith@example.com" }, { "\_id": "doctorId2", "name": "Dr. Jones", "email": "drjones@example.com" } ] }

#### **POST /api/patient/select-doctor**

* **Purpose:** Lets a patient choose a doctor, updating the Assignment model.
* **Authentication:** Must be an authenticated patient (role: false).
* **Request Body Example:**
* json
* CopyEdit
* { "doctorId": "60f8a8e2c3a1b34d2c8d9f1a" }
* **Response Example:**
* json
* CopyEdit
* { "message": "Doctor selected successfully", "assignment": { "doctor": "60f8a8e2c3a1b34d2c8d9f1a", "patients": ["patientId"], ... } }

### **Notes Endpoints**

*(Located in src/api/notes)*

#### **POST /api/notes/add\_note\_task**

* **Purpose:** Allows a doctor to create a note for a patient. The note is processed via an LLM to extract actionable steps.
* **Fields:**
* doctor\_id, patient\_id, original\_note
* extracted\_actions (includes checklist, plan, number\_of\_days, interval\_between\_days)
* number\_of\_days\_left is initialized from the LLM data.
* is\_done is false initially; it becomes true when number\_of\_days\_left reaches 0.
* remind\_patient\_today tracks if the patient has been reminded for the day.
* **Authentication:** Accessible only to doctors.
* **Request Body Example:**
* json
* CopyEdit
* { "patientId": "patientId123", "originalNote": "Patient must take drug X daily for 7 days." }
* **Response Example:**
* json
* CopyEdit
* { "note": { "\_id": "noteId", "doctor\_id": "doctorId", "patient\_id": "patientId123", "original\_note": "Patient must take drug X daily for 7 days.", "extracted\_actions": { "actionable\_steps": { "checklist": "Buy drug X", "plan": "Take drug X daily", "number\_of\_days": 7, "interval\_between\_days": 1 } }, "number\_of\_days\_left": 7, "is\_done": false, "remind\_patient\_today": false, "date\_of\_creation": "2025-02-14T05:01:17.000Z", "createdAt": "...", "updatedAt": "..." } }

#### **GET /api/notes/get\_my\_tasks**

* **Purpose:** Lets a patient retrieve all active (i.e. is\_done: false) note tasks.
* **Authentication:** Accessible only to patients.
* **Response Example:**
* json
* CopyEdit
* { "tasks": [ { "\_id": "noteId", "doctor\_id": "doctorId", "patient\_id": "patientId", "original\_note": "Patient must take drug X daily for 7 days.", "extracted\_actions": { ... }, "number\_of\_days\_left": 5, "is\_done": false, "remind\_patient\_today": false, "createdAt": "...", "updatedAt": "..." } ] }

# 3. Dynamic Scheduling & Task Lifecycle

### **Dynamic Scheduling**

* **Overview:**The system supports dynamic scheduling of patient reminders based on the actionable plan extracted from a doctor’s note.
* **Process:**

1. **Note Creation:**When a doctor creates a note (via add\_note\_task), the LLM processing returns a plan (e.g., “Take medication daily for 7 days”) and a corresponding number of days.
2. **Initialization:**The number\_of\_days\_left field is set to the LLM’s number\_of\_days.
3. **Daily Decrement:**A background job (using a scheduler like Bull or node-cron) runs daily, decrementing number\_of\_days\_left by 1 for each active note.
4. **Completion:**Once number\_of\_days\_left reaches 0, the note’s is\_done flag is set to true, preventing it from being retrieved by the patient.
5. **Reminder Flag:**The remind\_patient\_today flag tracks whether the patient has been reminded on a particular day. It is reset each day by the scheduling system.

* **Cancellation:**If a new note is added for the patient, any previously scheduled actionable tasks for that patient are canceled to ensure that only the most recent note is active.