Product Requirements & Specification Document

Project Name

MediTrack - Patient Record API

Description

MediTrack is a backend API for managing patient records, supporting CRUD operations, secure user authentication, and file uploads for medical reports. Built with Node.js, Express, and MongoDB, it is designed for educational use and easy integration.

---

1. Goals & Objectives

Goal	Description
Patient Record Management	Enable CRUD operations for patient records
Secure Access	Implement user authentication and authorization
Medical Report Uploads	Support file uploads (PDF, images) for patient reports
Educational Simplicity	Maintain clear, easy-to-understand code and structure

---

2. Functional Requirements

ID	Requirement
FR1	Users can register and log in (JWT-based authentication)
FR2	Authenticated users can create, read, update, and delete patient records
FR3	Each patient record includes personal info and a list of medical reports
FR4	Users can upload, retrieve, and delete medical report files (PDF, images)
FR5	Only authenticated users can access or modify records
FR6	API returns appropriate status codes and error messages

---

3. Non-Functional Requirements

ID	Requirement
NFR1	Use Node.js, Express, and MongoDB
NFR2	RESTful API design
NFR3	Input validation and error handling
NFR4	Secure file storage (local or cloud optional)

NFR5	Basic rate limiting to prevent abuse
NFR6	Clear API documentation (OpenAPI/Swagger)

---

4. User Roles

Role	Permissions
User	Register, log in, manage own patient records and reports

---

5. Data Model Overview

Patient

```
{
  "id": "ObjectId",
  "name": "string",
  "dob": "date",
  "gender": "string",
  "contact": "string",
  "reports": [
    {
      "filename": "string",
      "url": "string",
      "uploadedAt": "date"
    }
  ],
  "createdBy": "UserId"
}
```

User

```
{
  "id": "ObjectId",
  "username": "string",
  "passwordHash": "string"
}
```

---

6. API Endpoints

Method	Endpoint	Description	Auth Required
POST	/auth/register	Register new user	No
POST	/auth/login	User login, returns JWT	No
GET	/patients	List all patient records	Yes

POST	/patients	Create new patient record	Yes
GET	/patients/:id	Get patient record by ID	Yes
PUT	/patients/:id	Update patient record	Yes
DELETE	/patients/:id	Delete patient record	Yes
POST	/patients/:id/reports	Upload medical report file	Yes
GET	/patients/:id/reports/:fileId	Download medical report file	Yes
DELETE	/patients/:id/reports/:fileId	Delete medical report file	Yes

---

7. Security & Compliance

- JWT authentication for all protected endpoints
- Passwords hashed using bcrypt
- Input validation and sanitization
- File type and size restrictions for uploads
- Access control: users can only access their own records

---

8. Error Handling

Code	Description
400	Bad Request (validation errors)
401	Unauthorized (auth required/failed)
403	Forbidden (access denied)
404	Not Found (resource missing)
500	Internal Server Error

---

9. Technology Stack

Component	Technology
Language	Node.js
Framework	Express
Database	MongoDB
Auth	JWT, bcrypt
File Upload	Multer
Docs	Swagger/OpenAPI

---

10. Out of Scope

- No frontend/UI implementation
- No advanced user roles (admin, etc.)
- No third-party integrations (e.g., email, SMS)
- No production-grade deployment setup

---

11. Milestones

Milestone	Description
1. Project Setup	Repo, dependencies, basic structure
2. Auth Implementation	Register, login, JWT
3. Patient CRUD	Endpoints for patient records
4. File Uploads	Medical report upload/download/delete
5. Validation & Security	Input checks, access control, error handling
6. Documentation	API docs, usage examples

---

12. Acceptance Criteria

- All endpoints function as specified
- Only authenticated users can access/modify data
- File uploads are secure and retrievable
- API returns correct status codes and errors
- Documentation is clear and complete

End of Document