🤖 AI Hackathon Task – MERN Stack Students (Final & Mid)

Project Title:

AI Clinic Management + Smart Diagnosis SaaS

🌍 Problem Statement

Small and medium-sized clinics still rely heavily on:

- Paper-based prescriptions
- Manual patient records
- No digital appointment tracking
- No analytics or reporting
- No AI support for diagnosis

This leads to:

- Data loss
- Time waste
- Inefficient patient handling
- No performance visibility

Your mission is to build a **modern AI-powered Clinic Management SaaS** that digitizes the entire workflow and intelligently assists doctors.

This is not just a hackathon submission.

This can be your **first real startup idea**.

🎯 Vision

Build a scalable SaaS platform that:

- Digitizes clinic operations
- Improves efficiency
- Provides intelligent AI assistance
- Can realistically be sold to local clinics

👥 User Roles (Mandatory – 4 Roles)

1️⃣ Admin

- Manage doctors
- Manage receptionists
- View analytics
- Manage subscription plans (simulation allowed)
- Monitor system usage

2️⃣ Doctor

- View appointments
- Access patient history
- Add diagnosis
- Write prescriptions
- Use AI assistance
- View analytics (personal stats)

3️⃣ Receptionist

- Register new patients
- Book appointments
- Update patient info
- Manage daily schedule

4️⃣ Patient

- Login securely
- View profile
- View appointment history
- View prescriptions
- Download prescription PDF
- See AI-generated explanation (if enabled)

🛠 Tech Stack Requirements

🏆 FINAL HACKATHON – Advanced MERN Track

Required Tech Stack:

- MongoDB
- Express.js
- React.js
- Node.js
- JWT Authentication
- Role-Based Access Control
- Chart.js / Recharts (for analytics)
- Cloudinary / Supabase Storage (for file uploads)

AI Integration (Required for full marks):

- Gemini / OpenAI API (or any other AI Tool which ever is free)
- Backend AI endpoint handling
- Graceful fallback if API fails

🚀 **MID HACKATHON – Intermediate Level (Batch 16 & 17)**

Required Tech Stack:

Option 1:

- MERN (If you know React)
- Simple REST APIs if you about NodeJS
- Basic JWT

Option 2:

- HTML, CSS, JavaScript
- Firebase Auth / Supabase Auth
- Firestore / Supabase DB
- Supabase Storage / Cloudinary
- Basic CRUD
- Optional AI

🔥 **Core Features (Mandatory for Both Levels)**

1️⃣ **Authentication & Authorization**

- Secure login
- Role-based dashboard
- Protected routes
- Input validation

2️⃣ Patient Management

- Add patient
- Edit patient
- View patient profile
- View medical history timeline

3️⃣ Appointment Management

- Book appointment (by receptionist or patient)
- Cancel appointment
- Update status (pending / confirmed / completed)
- Doctor schedule view

4️⃣ Prescription System

- Add medicines
- Add dosage
- Add notes
- Generate PDF
- Patient can download prescription

5️⃣ Medical History Timeline

Each patient should have:

- Appointment history
- Diagnosis history
- Prescription history
- Timestamp tracking

🤖 AI Features (Advanced Layer)

AI must enhance experience, not block system.

If AI fails, the system must still function normally.

🧠 AI Feature 1 – Smart Symptom Checker

Doctor enters:

- Symptoms
- Age
- Gender
- History

AI returns:

- Possible conditions
- Risk level
- Suggested tests

📄 AI Feature 2 – Prescription Explanation

AI generates:

- Simple explanation for patient
- Lifestyle recommendations
- Preventive advice

Optional: Urdu explanation mode.

⚠ AI Feature 3 – Risk Flagging

System detects:

- Repeated infection patterns
- Chronic symptoms

- High-risk combinations

📊 AI Feature 4 – Predictive Analytics (Final Hackathon Only)

- Most common disease this month
- Patient load forecast
- Doctor performance trends

📊 Analytics Dashboard (Mandatory for Final)

Admin Dashboard:

- Total patients
- Total doctors
- Monthly appointments
- Revenue (simulated)
- Most common diagnosis

Doctor Dashboard:

- Daily appointments
- Monthly stats
- Prescription count

💳 SaaS Layer (Final Hackathon only)

Simulate subscription plans:

Free Plan

- Limited patients
- No AI features

Pro Plan

- Unlimited patients

- AI features enabled
- Advanced analytics

Feature-based access control required.

🗃️ Suggested Database Structure

Users

- id
- name
- email
- password
- role
- subscriptionPlan

Patients

- id
- name
- age
- gender
- contact
- createdBy

Appointments

- id
- patientId
- doctorId
- date
- status

Prescriptions

- id
- patientId
- doctorId
- medicines[]
- instructions
- createdAt

DiagnosisLogs

- id
- symptoms
- aiResponse
- riskLevel
- createdAt

🎨 **UI Requirements**

- Clean medical theme
- Sidebar navigation
- Responsive layout
- Proper error messages
- Loading states
- Form validation

🚀 **Deployment Requirement (Final Hackathon)**

- Must be deployed
- Live demo required
- GitHub repository required

- Proper README required

Mid-level hackathon deployment optional but encouraged.

💡 **Startup Opportunity**

This project is highly commercial.

You are encouraged to:

- Approach nearby clinics
- Offer live demo
- Customize features
- Add SMS reminders
- Add WhatsApp integration
- Add billing module
- Convert into real SaaS

If you launch this as a startup, we would be extremely proud of you.

✅ Submission Requirements (Mandatory)

1. **Deployed URL (Live App)**

- A working deployed link where the project can be tested.
- Example platforms: Vercel / Netlify (Frontend), Render / Railway / Cyclic / Firebase / Supabase (Backend/DB hosting as applicable)

2. **GitHub Repository URL**

- Public repo preferred (or provide collaborator access if private).
- Must include clean commit history (avoid uploading zip as 1 commit).

3. **Project Demo Video URL (LinkedIn or YouTube)**

- A 3–7 minute demo video showing:
    - Login + role dashboards

- Patient management
- Appointment booking
- Prescription generation (PDF)
- Medical history timeline
- AI features (if implemented)
- Admin analytics (final hackathon)