J-Doc: AI-Enabled Medical Consultant

Documentation

SOFTWARE ENGINEERING AND MANAGEMENT SS G562
in
Master of Engineering (Software Systems)

by

Team 11

Name	**Email**
Shreedhar Soni (2024H1120179P)	**h20240179@pilani.bits-pilani.ac.in**
Rohit Singhee (2024H1120258P)	**h20240258@pilani.bits-pilani.ac.in**
Anmol Sharma(2024H1120191P)	**h20240191@pilani.bits-pilani.ac.in**
Kuldeep Chaudhary (2024H1120184P)	**h20240184@pilani.bits-pilani.ac.in**

Github Repository URL

https://github.com/RohitSinghee228/Jdoc-AI-Medical-Consultant/tree/master/

Under the supervision of

Dr. Tanmaya Mahapatra

Assistant Professor
Department of Computer Science and Information Systems

Birla Institute of Technology and Science
Pilani, Pilani Campus, Rajasthan (India) November,2024

Textual Description Of J-Doc:

Github Repository URL:

https://github.com/RohitSinghee228/Jdoc-AI-Medical-Consultant/tree/master/

J-Doc: AI-Enabled Medical Consultant is a transformative web-based platform developed to address global challenges in healthcare accessibility. By combining advanced AI models and telemedicine features, J-Doc offers users a seamless and efficient pathway from initial symptom identification to expert medical consultation. The platform's primary goal is to streamline the diagnostic process, enabling faster responses and helping doctors focus on complex cases by reducing repetitive initial consultations.

Users interact with J-Doc through an intuitive interface, where they can input symptoms from a curated list covering diverse medical conditions. Leveraging finely-tuned Large Language Models (LLMs) enhanced by prompt engineering, the AI system interprets these symptoms and predicts potential diseases. The use of advanced Natural Language Processing (NLP) and Deep Learning allows J-Doc to deliver accurate, context-aware preliminary diagnoses, empowering users with actionable health insights before formal consultation.

The system also integrates a secure telemedicine network, which connects users with qualified healthcare providers for remote consultations. This feature is especially valuable in underserved or remote areas, where in-person access to healthcare may be limited.

SRS Document of J-Doc

Table of Contents

1. Introduction

1.1 Purpose

This SRS document offers a comprehensive summary of J-Doc. The primary objective is to outline the project requirements for J-Doc. The document provides a detailed description of both functional and non-functional requirements as proposed by the client. The project aims to improve healthcare accessibility by combining AI-driven diagnostics and telemedicine. Its central purpose is to provide a user-friendly environment where patients can enter symptoms, receive preliminary disease predictions, and access remote consultations with healthcare providers. Additionally, the project includes distinct dashboards for Admin, Doctor, and Patient, supporting patient management and facilitating appointments. The document describes hardware and software interface requirements and includes relevant UML diagrams.

1.2 Product Scope

The product scope of "J-Doc" is to provide a responsive web platform that leverages AI to predict potential diseases based on patient-reported symptoms, helps in better diagnosis, reduces cognitive load of Doctors, facilitate seamless connections with qualified healthcare providers for remote consultations, and support efficient patient management through specialized dashboards for Admin, Doctor, and Patient roles. The platform aims to improve healthcare accessibility, particularly in underserved regions, by integrating telemedicine features and ensuring secure handling of user data, thereby making healthcare more accessible, efficient, and user-friendly**.**

1.3 Intended Audience

The intended users of the “J-Doc” system include individuals registering as patients, licensed healthcare providers, and hospital administrators.

1.4 References

1.IEEE 830 Template-1993

.

2. Approach

2.1 Methodology

This project will employ an Agile Scrum methodology to facilitate iterative development, ensure timely feedback, and support flexibility and adaptability to evolving requirements.

2.2 Technology Stack

- **Backend:** Django
- **Frontend:** HTML, CSS, JavaScript
- **Database:** SQLite
- **API Architecture:** REST API
- **LLM Deployment**: Ollama Dockerized Service
- **Model Used:** llama3.1 for AI chat functionality

2.3 Product Functions

The "J-Doc: AI-Enabled Medical Consultant " system will be developed to provide the following essential functions:

Admin Functions:

1 **User & Access Management**:
- Approve or deny registration requests for new doctors and patients.
- Manage user records (view, update, delete).

2 **Appointment & Admission Management**:
- Approve, deny, or modify patient-doctor appointment requests.
- Admit or discharge patients and generate and send discharge bills in PDF format.

3 **Dashboard Overview**:
- Access a dashboard displaying statistics and recent records (e.g., pending doctor and patient registration, pending appointments, recent activity summaries).

Doctor Functions:

1 **Registration & Profile**:
- Submit a registration request to the admin for access to the platform.
- After approval, log in and access the doctor dashboard.

2 **Appointment & Patient Management**:
- View and manage their appointments, including pending and completed cases.
- Access AI-assisted diagnostic suggestions for each appointment and finalize prescriptions.

3 **Patient History & Records**:
- View records of both current and discharged patients, with details like symptoms, treatment history, and contact information.

Patient Functions:

1 **Registration & Login**:

- Submit a registration request to the admin, and upon approval, log in to the platform.

2 **Appointment Booking & Management**:

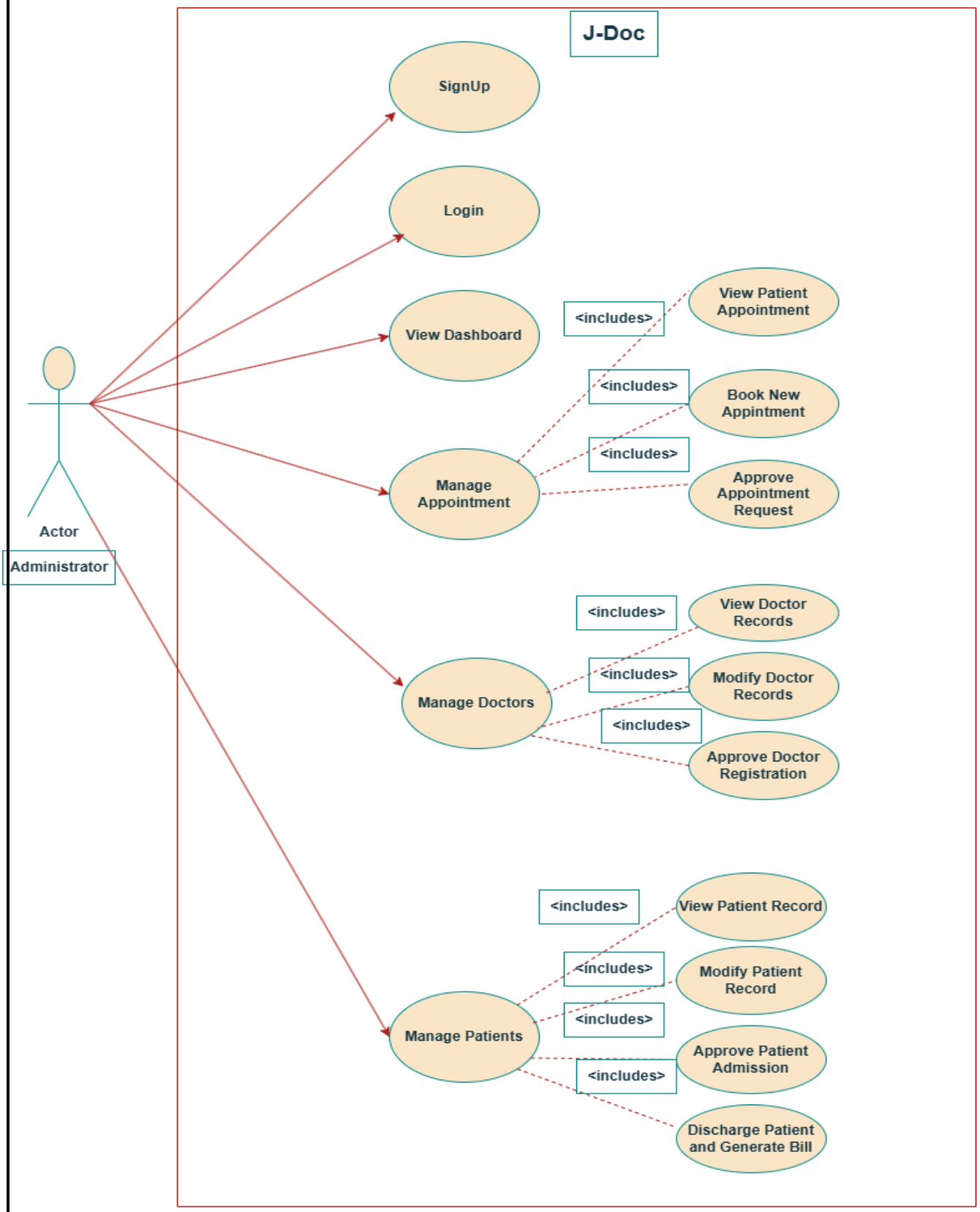
- Book new appointments with doctors and receive AI-assisted preliminary diagnoses through a chatbot.
- Check appointment status and access the details of confirmed bookings.

3 **Discharge & Prescription Access**:

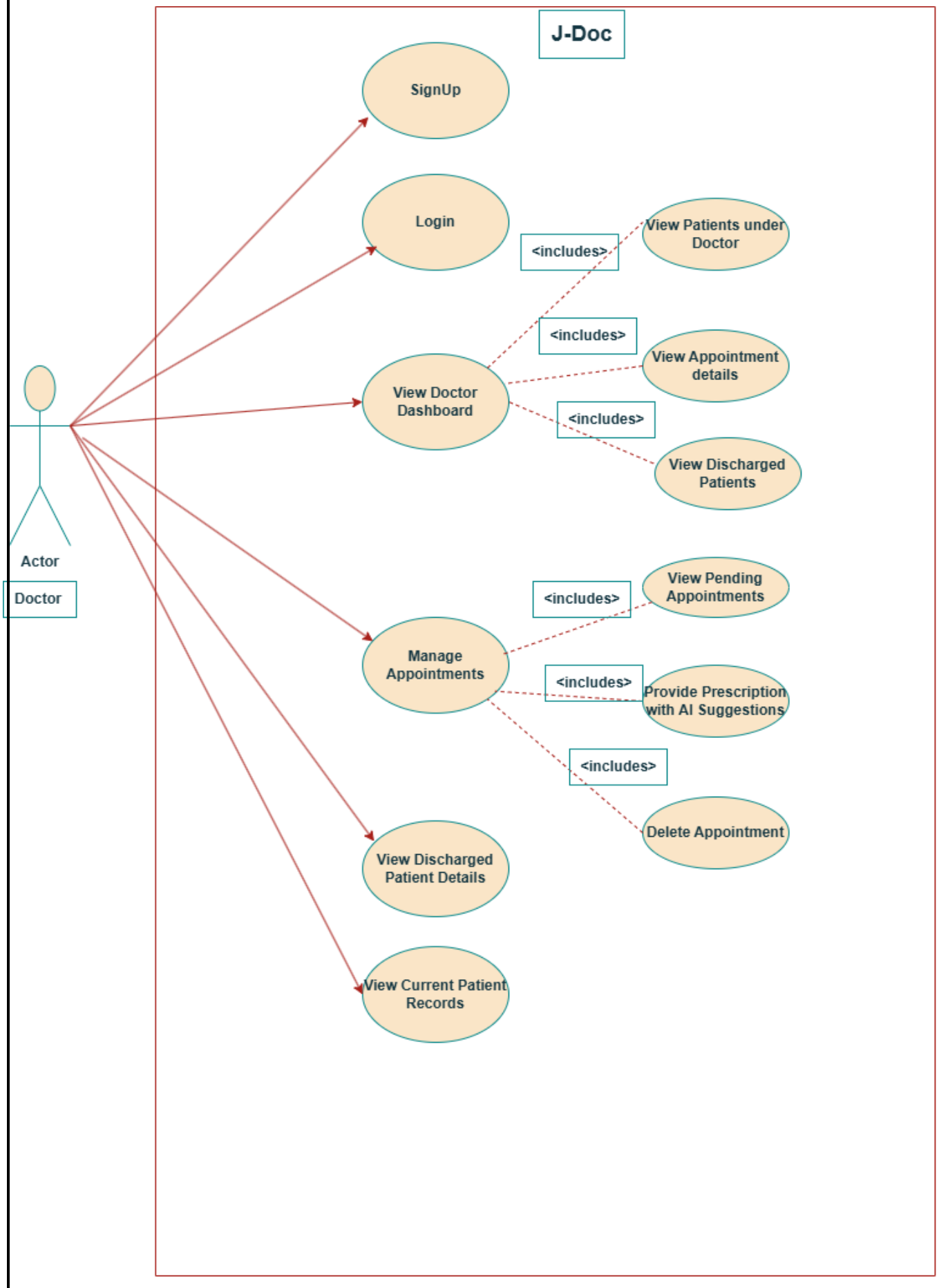
After discharge, view and download prescriptions and other related documents provided by the doctor

2.4 Use Case Diagrams

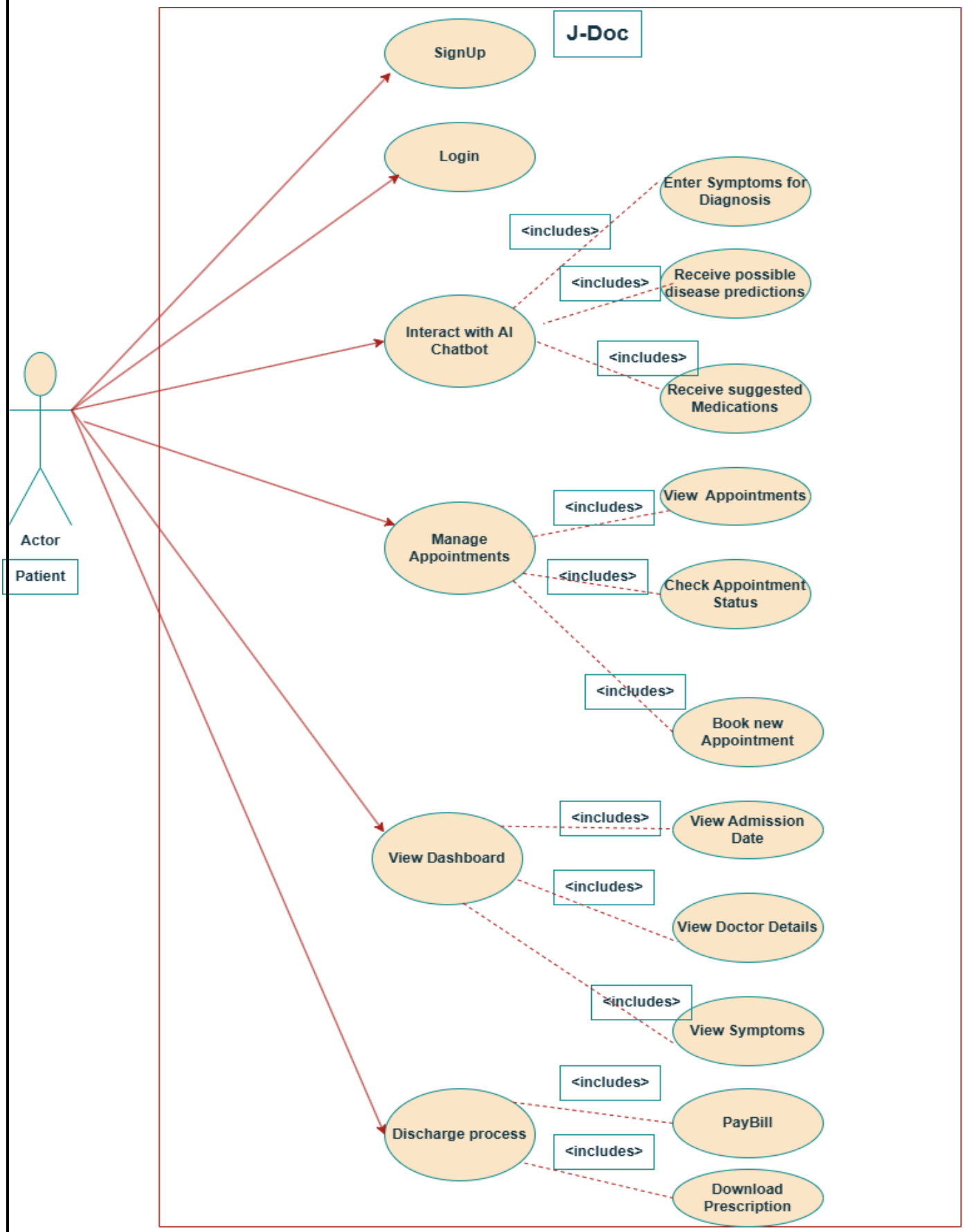
2.4.1 Administrator



2.4.2 Doctor :


J-Doc
SignUp
Login
View Doctor Dashboard
Manage Appointments
View Discharged Patient Details
View Current Patient Records
<includes>
View Patients under Doctor
<includes>
View Appointment details
<includes>
View Discharged Patients
<includes>
View Pending Appointments
<includes>
Provide Prescription with AI Suggestions
<includes>
Delete Appointment
Actor
Doctor

2.4.3 Patient


J-Doc
SignUp
Login
Interact with AI Chatbot
Enter Symptoms for Diagnosis
Receive possible disease predictions
Receive suggested Medications
<includes>
<includes>
<includes>
Manage Appointments
View Appointments
Check Appointment Status
Book new Appointment
<includes>
<includes>
<includes>
View Dashboard
View Admission Date
View Doctor Details
View Symptoms
<includes>
<includes>
<includes>
Discharge process
PayBill
Download Prescription
<includes>
<includes>
Actor
Patient

3. Functional Requirements:

R1: Admin Functional Requirements

Requirement 1.1

Initiator	Admin
Date	10/11/2024
Requirement	Admin will be able to view and approve new registration requests from Doctors and Patients.
Importance	High
Description	The Admin dashboard should display all pending Doctor and Patient registration requests for approval.
Rationale	Approval by the Admin ensures only authorized healthcare professionals and patients can access the system..
Use Case	Admin accesses the registration requests section, reviews details, and either approves or denies requests.
Dependencies	Admin Dashboard Module.
Conflicts	None identified.
Supporting Material	Mockup of the Admin dashboard showing pending requests.
Comments	Essential for maintaining security and control over who accesses the platform.

Requirement 1.2

Initiator	Admin
Date	10/11/2024
Requirement	Admin can manage Doctor and Patient records.
Importance	High
Description	Admin should be able to view, update, and delete records of both Doctors and Patients.
Rationale	Accurate records are essential for smooth operation and regulatory compliance..
Use Case	Admin selects a record to view details, edit information, or delete the record if necessary.
Dependencies	Doctor and Patient record modules.
Conflicts	None identified.
Supporting Material	Table of Doctor and Patient records for Admin access.
Comments	Record management helps keep data up-to-date and organized

Requirement 1.3

Initiator	Admin
Date	10/11/2024
Requirement	Admin can manage appointments between Doctors and Patients.
Importance	High
Description	Admin should be able to book, view, approve, or deny appointment requests.
Rationale	Streamlines appointment scheduling and ensures patient needs are addressed in a timely manner.
Use Case	Admin accesses the appointment module, reviews details, and either approves or denies the request.
Dependencies	: Appointment scheduling and notification modules
Conflicts	None identified
Supporting Material	Appointment management screen.
Comments	Key feature for ensuring efficient appointment scheduling.

R2 : Doctor Functional Requirements

Requirement 2.1

Initiator	Doctor
Date	10/11/2024
Requirement	Doctors will be able to register and request approval from Admin..
Importance	High
Description	Doctors register by providing required credentials; access is granted upon Admin approval.
Rationale	Ensures only authorized doctors can access patient information and system features.
Use Case	Doctor completes registration with personal and professional details; request is sent to Admin
Dependencies	Registration and Authentication modules.
Conflicts	None identified.
Supporting Material	Registration form mockup
Comments	Doctor registration maintains system integrity.

Requirement 2.2

Initiator	Doctor
Date	10/11/2024
Requirement	Doctors can manage their patient appointments.
Importance	High
Description	Doctors can view pending appointments and provide prescriptions, aided by AI suggestions.
Rationale	Facilitates efficient patient care and treatment.
Use Case	Doctor views appointments, uses AI suggestions, and provides a prescription.
Dependencies	Appointment and AI modules.
Conflicts	None identified.
Supporting Material	Appointment and prescription screens.
Comments	Enhances diagnostic accuracy and speeds up patient treatment.

R3: Patient Functional Requirement

Requirement 3.1

Initiator	Patient
Date	10/11/2024
Requirement	Patients will be able to register and request admission
Importance	High
Description	Patients fill in registration details; admission requires Admin approval.
Rationale	Validates patient eligibility and admission.
Use Case	Patient provides necessary personal information and submits registration request.
Dependencies	Registration and Admission modules..
Conflicts	None identified.
Supporting Material	Patient registration form..
Comments	Registration is essential for patient access..

Requirement 3.2

Initiator	Patient
Date	10/11/2024
Requirement	Patients can manage their appointments.
Importance	High
Description	Patients should be able to book, view, and manage their appointments with doctors.
Rationale	Simplifies appointment scheduling and updates.
Use Case	Patient selects a doctor, fills in symptom details, and requests an appointment.
Dependencies	Appointment scheduling and LLM modules.
Conflicts	None identified.

Supporting Material	Appointment booking screen.
Comments	Facilitates access to healthcare services.

Requirement 3.3

Initiator	Patient
Date	10/11/2024
Requirement	Patients can view and download their prescription post-discharge.
Importance	Medium
Description	Patients can download the prescription provided by the doctor after discharge..
Rationale	Enables easy access to post-care instructions.
Use Case	Patient logs into their account, navigates to the prescriptions section, and downloads the document.
Dependencies	Prescription and Discharge modules.
Conflicts	None identified.
Supporting Material	Prescription view and download screens.
Comments	Enhances patient post-discharge care.

R4: LLM Functional Requirement

Requirement 4.1

Initiator	System
Date	10/11/2024
Requirement	The LLM will interact with patients to capture and interpret symptom information.
Importance	Medium
Description	The LLM will act as a chatbot, prompting patients to enter symptoms in natural language. It will interpret these symptoms to provide possible diagnoses, suggested treatments, and medication options.
Rationale	Provides patients with an AI-driven initial assessment, allowing healthcare professionals to focus on complex cases and streamline diagnosis
Use Case	Patients interact with the chatbot, entering symptoms. The LLM suggests potential diagnoses, medication, and treatment recommendations based on symptoms provided.
Dependencies	NLP model, patient interface, appointment module.
Conflicts	None identified.
Supporting Material	Wireframe of chatbot interface
Comments	Improves the efficiency of patient-doctor interactions by gathering initial data before consultation.

Requirement 4.2

Initiator	Doctor
Date	10/11/2024
Requirement	The LLM will generate suggestions for the doctor's review based on patient symptoms.
Importance	Medium
Description	For each patient appointment, the LLM will generate potential diagnoses, medications, and recommended tests. Doctors can review and either accept or modify these suggestions when formulating a final prescription
Rationale	Supports doctors in quickly assessing patient needs, enhancing diagnostic accuracy and minimizing time on routine assessments.
Use Case	Upon reviewing a patient's symptoms, the LLM outputs a list of probable diagnoses, medications, and suggested tests, which the doctor can use as part of their diagnostic process.
Dependencies	Appointment management module, patient records module.
Conflicts	None identified.
Supporting Material	Sample output from the LLM for symptoms input
Comments	Assists doctors in preliminary diagnosis, making consultation more efficient.

Requirement 4.3

Initiator	Patient
Date	10/11/2024
Requirement	The LLM will provide health insights and educational information to patients based on the diagnosis.
Importance	Medium
Description	After preliminary diagnosis, the LLM provides patients with an overview of the condition, potential lifestyle changes, and treatment options related to their symptoms.
Rationale	Educates patients about their health conditions, promoting informed decision-making.
Use Case	Following symptom input, the LLM displays basic information about probable diagnoses and general lifestyle or dietary recommendations.
Dependencies	Patient dashboard, symptom input module.
Conflicts	None identified.
Supporting Material	Prescription view and download screens.
Comments	Enhances patient understanding and engagement in their healthcare.

4. Non Functional Requirements

4.1 Performance Requirement

1.The J-Doc platform should efficiently handle a significant number of users simultaneously
.
2.The system must maintain fast response times for essential functions, particularly for disease prediction and video consultations, to ensure smooth user experience.

4.2 Reliability Requirement

1 The platform should consistently provide accurate and reliable disease predictions to ensure patients receive the correct referrals for further consultation. The prediction algorithm must be dependable in delivering precise results based on symptom input.

2 High availability is essential, ensuring that patients, doctors, and pharmacies can access the system at all times.

3 The system should reflect real-time updates for Doctor Availability and Medicine Delivery Status, supporting timely and accurate information.

4 Data integrity is critical, with all patient data, prescriptions, and medical records being securely maintained to prevent any data loss or corruption.

4.3 Security Requirement

1 Access to the system dashboard should be restricted to users with valid, authenticated credentials.

2 Online payments within the platform should adhere to industry standards, supporting secure methods such as UPI.

3 Patient, Doctor, and Pharmacy data should be stored in compliance with established security protocols to protect sensitive information.

5. Formal and Semi Formal Specifications

5.1 Formal Specification

Type:

Defines Symptom, Disease, Doctor, Prescription, Patient, Appointment, Consultation, Location, AdminApproval

Uses Boolean, String, Date

Syntax:

RegisterDoctor: Doctor → AdminApproval
RegisterPatient: Patient → AdminApproval
Login: UserCredentials → Boolean
ApproveDoctor: AdminApproval → Boolean
ApprovePatient: AdminApproval → Boolean
GetLocation: String → Location
PredictDisease: Symptom × Patient → Disease
SearchDoctor: Disease × Location → Doctor
CreateConsultation: Doctor × Patient → Consultation
CreatePrescription: Consultation → Prescription
ViewAppointmentStatus: Appointment → String
DoctorAvailable: Doctor → Boolean
DischargePatient: Patient → Boolean
DownloadPrescription: Patient → PrescriptionFile

Equations:

RegisterDoctor(Doctor) =
IF Doctor's Details are Verified THEN Return AdminApproval
ELSE Return False

RegisterPatient(Patient) =
IF Patient's Details are Verified THEN Return AdminApproval
ELSE Return False

ApproveDoctor(AdminApproval) =
IF AdminApproval is True THEN Return True
ELSE Return False

ApprovePatient(AdminApproval) =
IF AdminApproval is True THEN Return True
ELSE Return False

CreateConsultation(Doctor, Patient) =
IF DoctorAvailable(Doctor) = True THEN Return Consultation
ELSE Return Not Defined

CreatePrescription(Consultation) =

```
        IF Consultation is Confirmed THEN Return Prescription
        ELSE Return Not Defined


ViewAppointmentStatus(Appointment) =
        IF Appointment is Confirmed THEN Return “Confirmed”
        ELSE Return “Pending”

DoctorAvailable(Doctor) =
      IF Doctor’s Schedule is Open THEN Return True
      ELSE Return False

DischargePatient(Patient) =
     IF All Prescriptions Issued AND Bill Paid THEN Return True
     ELSE Return False

DownloadPrescription(Patient) =
   IF DischargePatient(Patient) = True THEN Return PrescriptionFile
   ELSE Return Not Defined
```

Function Description:

RegisterDoctor():Accepts a Doctor object containing their registration details. It triggers an AdminApproval process where the admin verifies the details before approval.

RegisterPatient():Takes Patient details as input. Similar to doctor registration, the details must be verified and approved by an admin.

Login():Accepts UserCredentials and validates them to return True if authentication is successful, granting access to the respective dashboard.

ApproveDoctor()&ApprovePatient():Processes AdminApproval status for doctor or patient registration requests, returning True if the admin approves the registration.

GetLocation():Accepts a String input (address or zip code) and returns the Location associated with the patient.

PredictDisease():Utilizes AI to process Symptom and Patient data, outputting a probable Disease based on AI predictions.

SearchDoctor():Searches for a Doctor who specializes in treating the identified Disease within a defined Location. The function considers doctor availability and distance from the patient.

CreateConsultation():Checks if a Doctor is available for consultation with the given Patient, initiating a Consultation if the doctor is free.

CreatePrescription():Generates a Prescription following a successful Consultation, with details of recommended treatments based on the symptoms.

ViewAppointmentStatus():Retrieves the Appointment status, which can either be “Confirmed” or “Pending,” based on the admin’s decision.

DoctorAvailable():Checks if a specific Doctor is currently available for consultation, returning True if they are free to take appointments.

DischargePatient():Confirms if a Patient is ready for discharge based on their bill payment and

prescription issuance.

DownloadPrescription():
Allows patients to download a PrescriptionFile after discharge, containing their treatment and medication details for reference.

5.2 Decision Table

Decision Table								
Conditions								
Patient Registered	NO	YES	YES	YES	YES	YES	YES	YES
Valid Login Credentials	NO	YES	YES	YES	YES	YES	YES	YES
Symptoms Provided		NO	YES	YES	YES	YES	YES	YES
Location Provided			NO	YES	YES	YES	YES	YES
Doctor Available				NO	YES	YES	YES	YES
Consultation Approved					NO	YES	YES	YES
Payment Completed						NO	YES	YES
LLM Diagnosis Provided							YES	YES
Prescription Generated							YES	YES
Actions								
Display Registration Error	EXC							
Display Login Error		EXC						
Display Symptom Input Prompt			EXC					
Display Location Input Prompt				EXC				
Display Doctor Not Available Message				EXC				
Display Payment Required Message					EXC			
Conduct LLM Diagnosis						EXC		
Generate Prescription							EXC	
Complete Consultation								EXC

5.3 Decision Tree

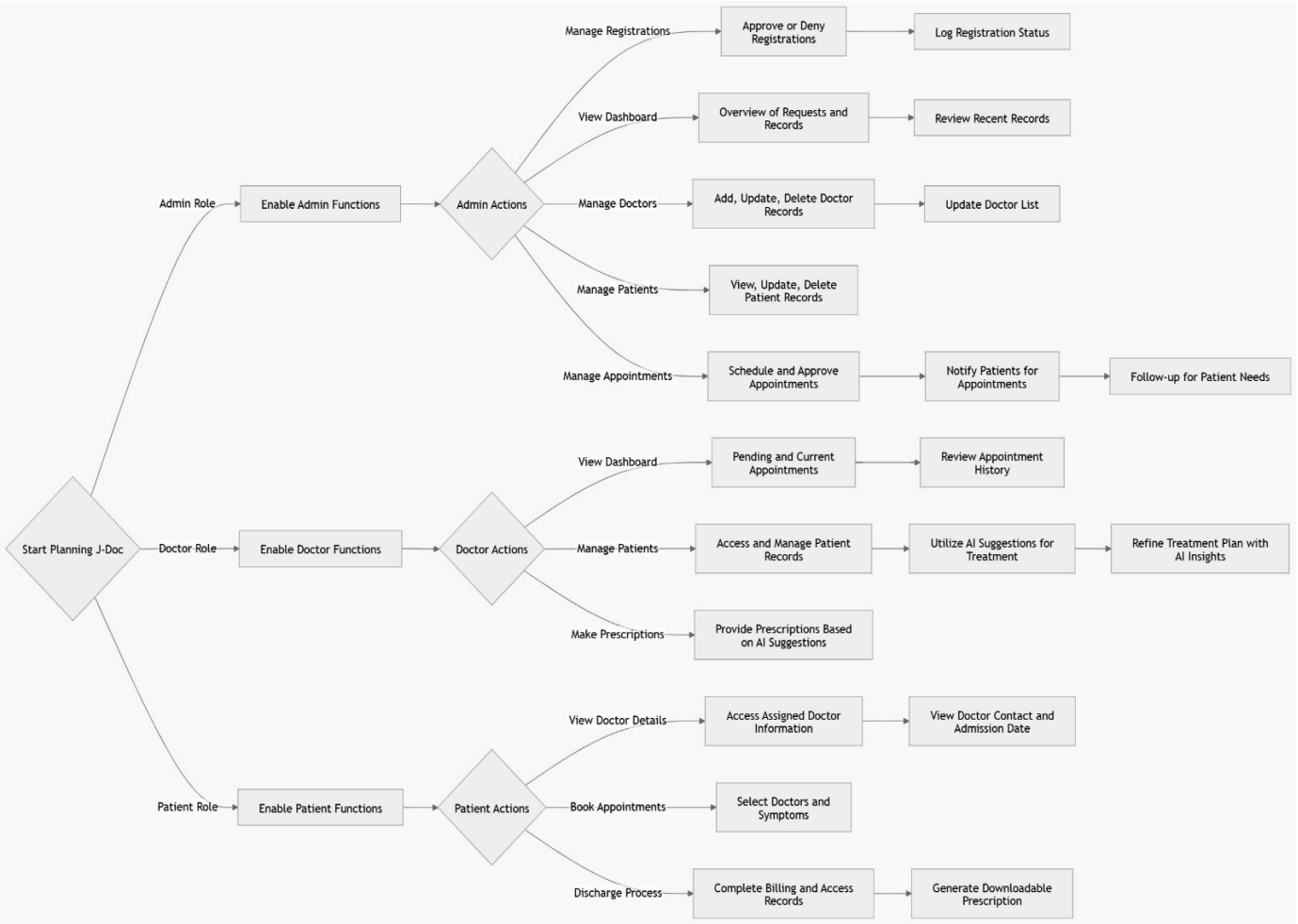


Fig: Decision Tree of J-Doc

6. Modelling and Project Planning

We require the customer consent to read the GPS location and know the address. So, we must ask first the consent.

6.1 Data Flow Diagram

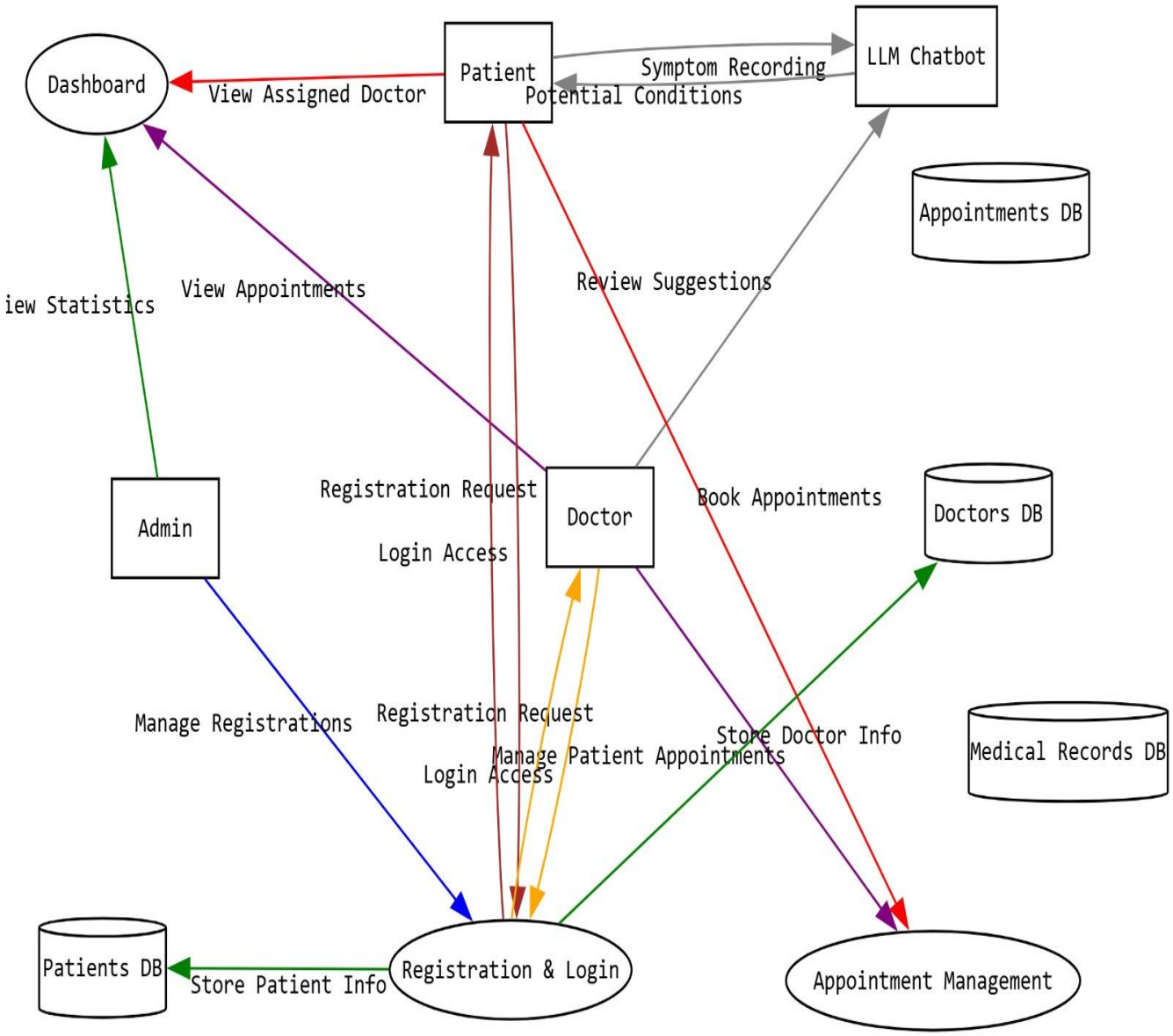


Figure: Data Flow Diagram

6.2 Activity Network

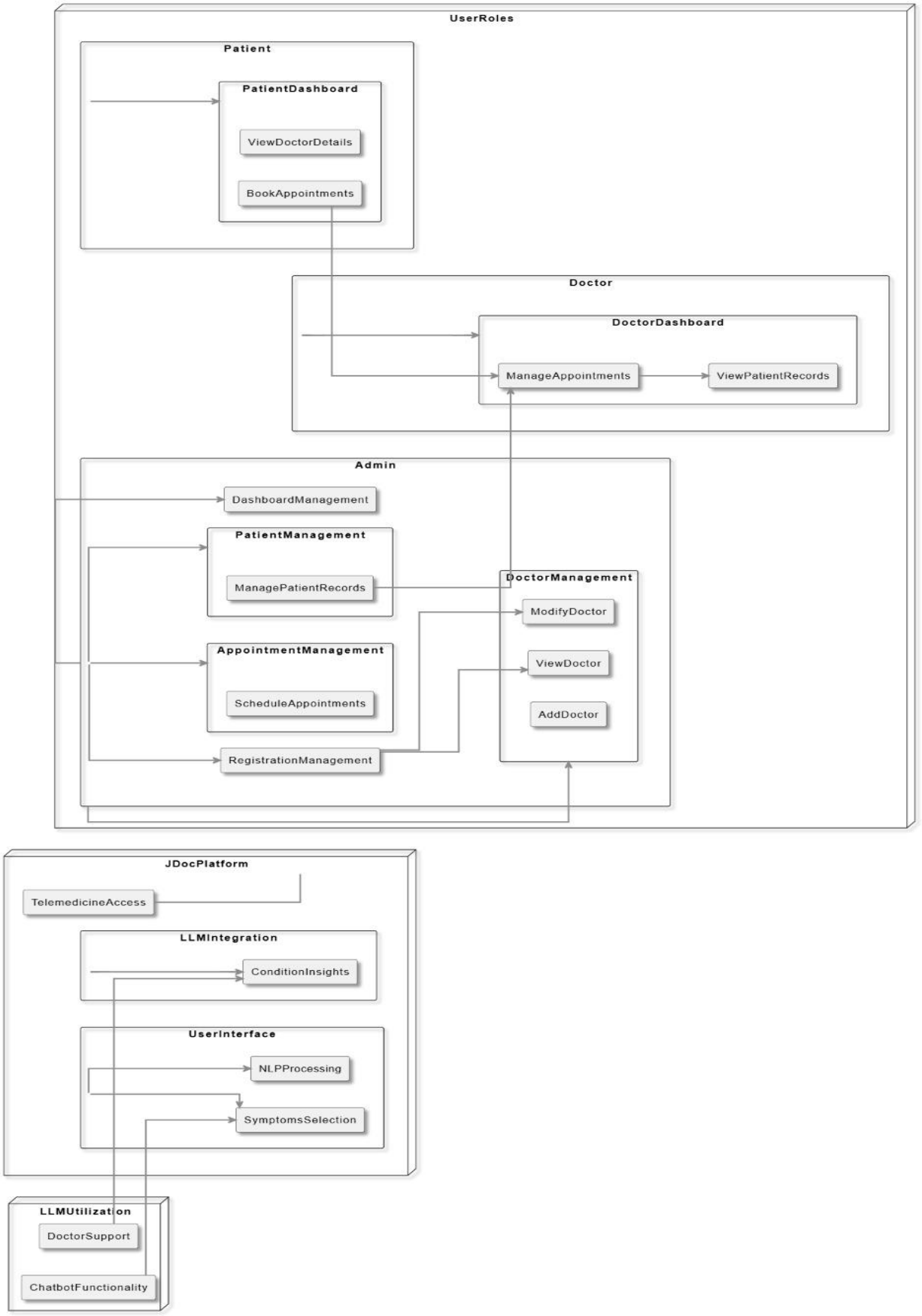


Figure: Activity Network

6.3 Work Breakdown Structure

Figure: Work Breakdown Structure

6.4 PERT Chart

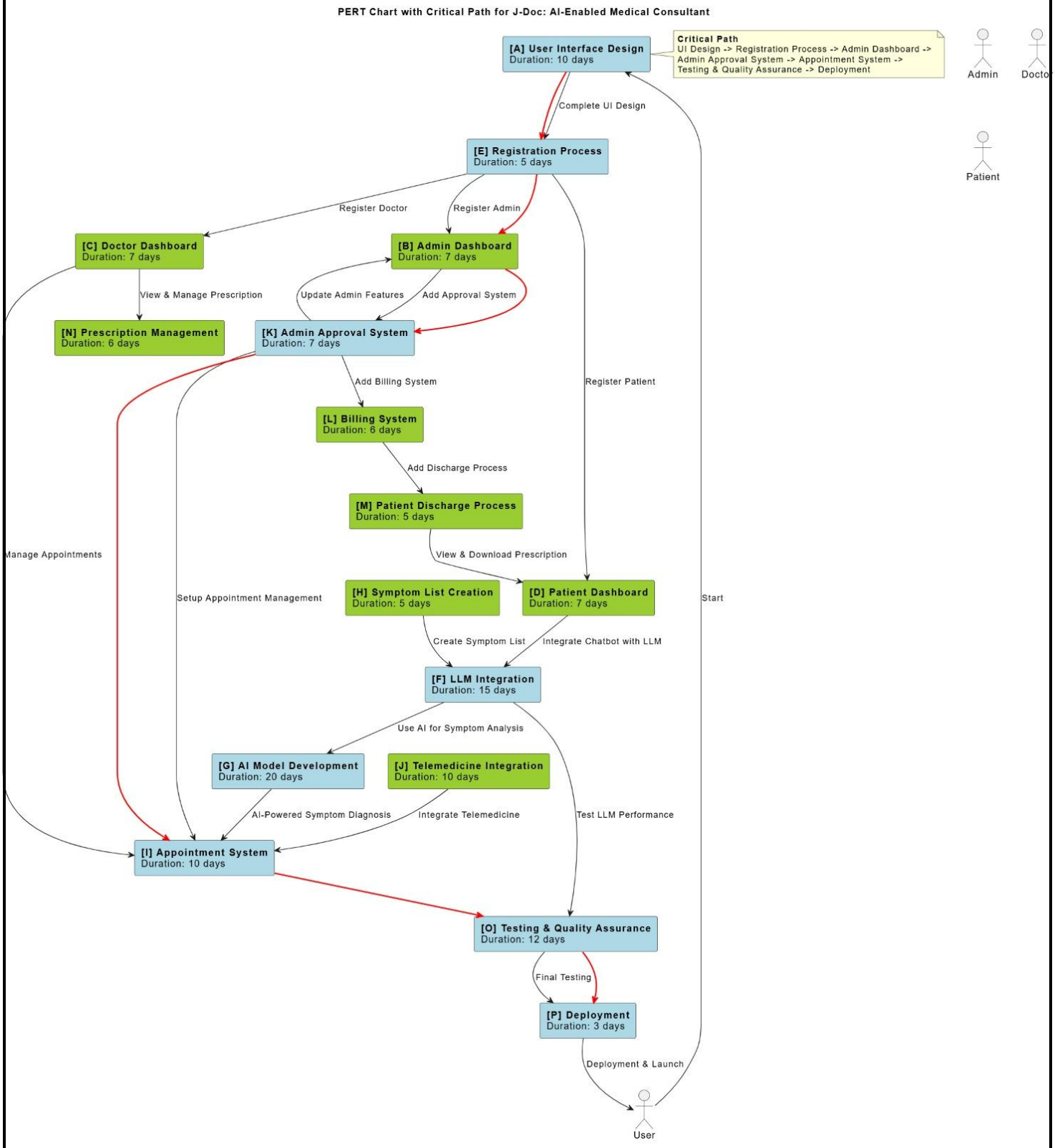


Figure: PERT Chart

7. Testing and Quality Assurance

This section will provide a clear and concise reference for each of the test cases, covering their names, expected outcomes, IDs, descriptions, inputs, and expected outputs**.**

1. **Test Case Name**: Test Patient User Form Valid
   - **Expected Result**: Form should be valid with all required fields provided.
   - **Test Case ID**: test_forms_1
   - **Description**: Tests that the PatientUserForm is valid when all required fields are filled correctly.
   - **Input**: Form data containing first_name as "John," last_name as "Doe," username as "johndoe," and password as "password123."
   - **Expected Output**: True, indicating the form is valid

2. **Test Case Name**: Test Patient User Form Invalid
   - **Expected Result**: Form should be invalid when required fields are missing.
   - **Test Case ID**: test_forms_2
   - **Description**: Tests that the PatientUserForm is invalid when no data is provided.
   - **Input**: Empty form data.
   - **Expected Output**: False, indicating the form is invalid.

3. **Test Case Name**: Test Doctor String Representation
   - **Expected Result**: Doctor model string should return the correct format.
   - **Test Case ID**: test_models_1
   - **Description**: Verifies that the __str__ method of the Doctor model returns a string in the correct format: "first_name (department)."
   - **Input**: Doctor instance with first_name as "testdoctor" and department as "Cardiologist."
   - **Expected Output**: "testdoctor (Cardiologist)".

4. **Test Case Name**: Test Patient String Representation
   - **Expected Result**: Patient model string should return the correct format.
   - **Test Case ID**: test_models_2
   - **Description**: Verifies that the __str__ method of the Patient model returns a string in the correct format: "first_name (symptoms)."
   - **Input**: Patient instance with first_name as "testpatient" and symptoms as "Test symptoms."
   - **Expected Output**: "testpatient (Test symptoms)".

5. **Test Case Name**: Test Patient Dashboard URL Resolution
   - **Expected Result**: URL should resolve to the correct view function.
   - **Test Case ID**: test_urls_1
   - **Description**: Tests that the URL for the patient dashboard resolves to the correct view function.
   - **Input**: URL pattern name patient-dashboard.
   - **Expected Output**: patient_dashboard_view, indicating that the URL resolves to the correct view.

6. **Test Case Name**: Test Home View Status and Template
   - **Expected Result**: Home view should return a 200 status code and use the correct template.
   - **Test Case ID**: test_view_1
   - **Description**: Tests that the home_view returns a 200 status code and renders the expected template (hospital/index.html).
   - **Input**: HTTP GET request to the URL for home_view.
   - **Expected Output**: 200 status code and template hospital/index.html is used.

8. Pre-implementation Size Estimation:

8.1 Basic COCOMO

- LOC is possibly the simplest among all metrics available to measure project size. This metric measures the size of a project by counting the number of source instructions in the developed program. While counting the number of source instructions, comment lines, and header lines are ignored.
- Based on experience and looking at similar applications, we estimated the size of the project.

  Estimated size = 17 KLOC

- Effort (E) = a1 × (KLOC)^a2 person-month
  $Time_{dev}$ = b1 × (Effort)^b2 months
  Project Cost (C) = E * Manpower cost per month
  Productivity (P) = KLOC / E (KLOC per person-month)
  Average Staffing Size (SS) = E / Timedev (persons)

- Because the Project size is less than 50 KLOC and deadline of the project is not tight we use,
  Organic Mode => **a1 = 2.4, a2 = 1.05, b1 = 2.5, b2 = 0.38**

  Effort = a1 * (KLOC)^a2 = 2.4 * (4.5)^1.05
  = 47 Person-Month

  $Time_{dev}$ = b1 * (Effort)^b2
  = 2.5 * (17)^0.38
  = 7.33 Months

- Manpower cost per month is ₹10,0000
  Project Cost (C) = E * Manpower cost per month
  = 47* 10000 = ₹470000

  Productivity (P) = KLOC / E (KLOC per person-month)
  = 17/47 = 0.361KLOC per person-month

  Average Staffing Size (SS) = E / $Time_{dev}$ (persons)
  = 47 / 7.33 = 6.41persons

8.2 Function Point

This uses No. of external inputs(EIs), No. of external outputs(EOs), No. of external inquires(EINs), No. of internal logic files(ILFs) and No. of external interface files (EIFs).

S.No	EIs	EOs	EINs	ILFs	EIFs
1.	Email, password, name, phone number, location, profile type (Simple)	-		Registered Users (Simple)	
2.	Symptoms (Average)	Predicted Disease (Complex)	Predict Disease (Average)	Patient Symptoms, Prediction History (Simple)	
3.	Doctor Availability (Simple)	Consultation Status, Consultation Link (Simple)		Consultation History (Simple)	
4.	Prescription (Simple)	Medicine Delivery Status (Average)		Patient Prescriptions, Order History (Average)	
Total	4	4	1	6	0

Measurement Parameter	Weighting Factor		
	Simple	**Average**	**Complex**
No. of external inputs(EIs)	3	4	6
No. of external outputs(EOs)	4	5	7
No. of external inquires(EINs)	3	4	6
No. of internal logic files(ILFs)	7	10	15
No. of external interface files (EIFs)	5	7	10

- Calculation of unadjusted function points:

Measurement Parameter	Simple	Average	Complex	Total
No. of external inputs(EIs)	9	4	6	19
No. of external outputs(EOs)	8	5	7	20
No. of external inquires(EINs)	0	4	0	4
No. of internal logic files(ILFs)	28	20	0	48
No. of external interface files (EIFs)	0	0	0	0
				92

UFP = 92

- Technical Complexity Factor Calculation:
  Degree of Influence (DI) = 14 * 3 = 42 (All factors considered average)

  TCF = 0.65 + 0.01*DI
  = 0.65 + 0.01*42
  = 1.07

  FP = TCF * UFP
  Adjusted Function Point (FP) = 1.07 * 92 = 98.44

- Javascript Code on average produces 47 Lines of Code per FP.
  Total LOC = 98.44 * 47
  = 4627 = 4.6 KLOC

8.3 Effort and Time Estimation using Intermediate COCOMO

- Organic Mode => **a1 = 2.4, a2 = 1.05, b1 = 2.5, b2 = 0.38**
- Effort Adjustment Factor(EAF):

Complexity of the product(CPLX):- High(1.15)
Programming Language experience(LEXP):- Low(1.07)
Use of software tools(TOOL):- High(0.91)
Application of modern programming methods(MODP):- High(0.91)
Programmer Capability(PCAP):- High (0.87)

All other factors are nominal

EAF = 1.15 * 1.07 * 0.91 * 0.91 * 0.87
= 0.88

Effort = a1 * (KLOC)^a2 * EAF = 2.4 * (11.64)^1.05 * 0.88
= 27.79 person-month

$Time_{dev}$ = b1 * (Effort)^b2
= 2.5 * (27.79)^0.38
= 8.84 Months

8.4 Detailed COCOMO

Since the size of code and mode are small and organic,
Here μ_p , phase sensitive effort multiplier for different phases are (0.06, 0.16, 0.26, 0.42, 0.16)

$E_p = \mu_p \times E$
$Time_{dev}(p) = \mu_p \times Time_{dev}$

$E_{Plan\ and\ Requirements} = 0.06 * 10.4 = 0.62$ Person-Month
$Time_{dev}$(Plan and Requirements) = 0.06 × 6.08 = 0.36 Months

$E_{System\ Design} = 0.16 * 10.4 = 1.66$ Person-Month
$Time_{dev}$(System Design) = 0.16 × 6.08 = 0.97 Months

$E_{Detailed\ Design} = 0.26 * 10.4 = 2.70$ Person-Month
$Time_{dev}$(Detailed Design) = 0.26 × 6.08 = 1.58 Months

$E_{Module\ Code\ and\ Test} = 0.42 * 10.4 = 4.36$ Person-Month
$Time_{dev}$(Module Code and Test) = 0.42 × 6.08 = 2.55 Months

$E_{Plan\ and\ Requirements} = 0.16 * 10.4 = 1.66$ Person-Month
$Time_{dev}$(Plan and Requirements) = 0.16 × 6.08 = 0.97 Months

8.5 COCOMO Ⅱ

No. of Screen = 10,
No. of Views = 7, No. of Tables = 5 => Complexity = Medium

No. of Report = 8,
No. of Sections = 3, No. of Tables = 9 => Complexity = Difficult

No. of 3GL Components = 1
Code Reuse = 15%

Object Points = (10*2) + (8*8) + (10*1) = 94
NOP = (Object Points * (100 - %Reuse)) / 100
= (94 * 85)/100
= 80

Productivity Table

Developers' experience	Very Low	Low	Nominal	High	Very High
CASE maturity	Very Low	Low	Nominal	High	Very High
PRODUCTIVITY	4	7	13	25	50

6. Finally, the estimated effort in person-months is computed as

E = NOP/PROD

Stage 2:

$$PM_{nominal} = A * (Size)^{B}$$ **where A = 2.5**

B = 0.91 + 0.01 * (Sum of rating on scaling factors for the project)

$$PM_{adjusted} = PM_{nominal} \times \left(\prod_{i=1}^{7} EM_i \right)$$

where, PM – Effort Estimate

- Developer's Experience and Case Maturity in our case is nominal so,

  E = 80/13 = 6.15 Person-Month

- For Stage 2: Taking PREC and FLEX to be Low and other factors to be nominal,Assuming the early drivers to be constant

  B = 0.91 + 0.01 * (21.22) = 1.12
  $PM_{Nominal}$ = 2.5 * (4.6)^1.12 = 13.81

$$PM_{nominal} = A * (Size)^{B}$$ where A = 2.5

$$B = 0.91 + 0.01 * (\text{Sum of rating on scaling factors for the project})$$

$$PM_{adjusted} = PM_{nominal} \times \left(\prod_{i=1}^{17} EM_i \right)$$ where, PM – Effort Estimate

$$TDEV_{nominal} = \left[\emptyset * (PM_{adjusted})^{(0.28+0.2(B-0.091))} \right] * \frac{SCED\%}{100}$$

Where

$\emptyset$ = constant, provisionally set to 3.67, ***B*** = Scaling factor
$TDEV_{nominal}$ = calendar time in months with a scheduled constraint
$PM_{adjusted}$ = Estimated effort in Person months (after adjustment)

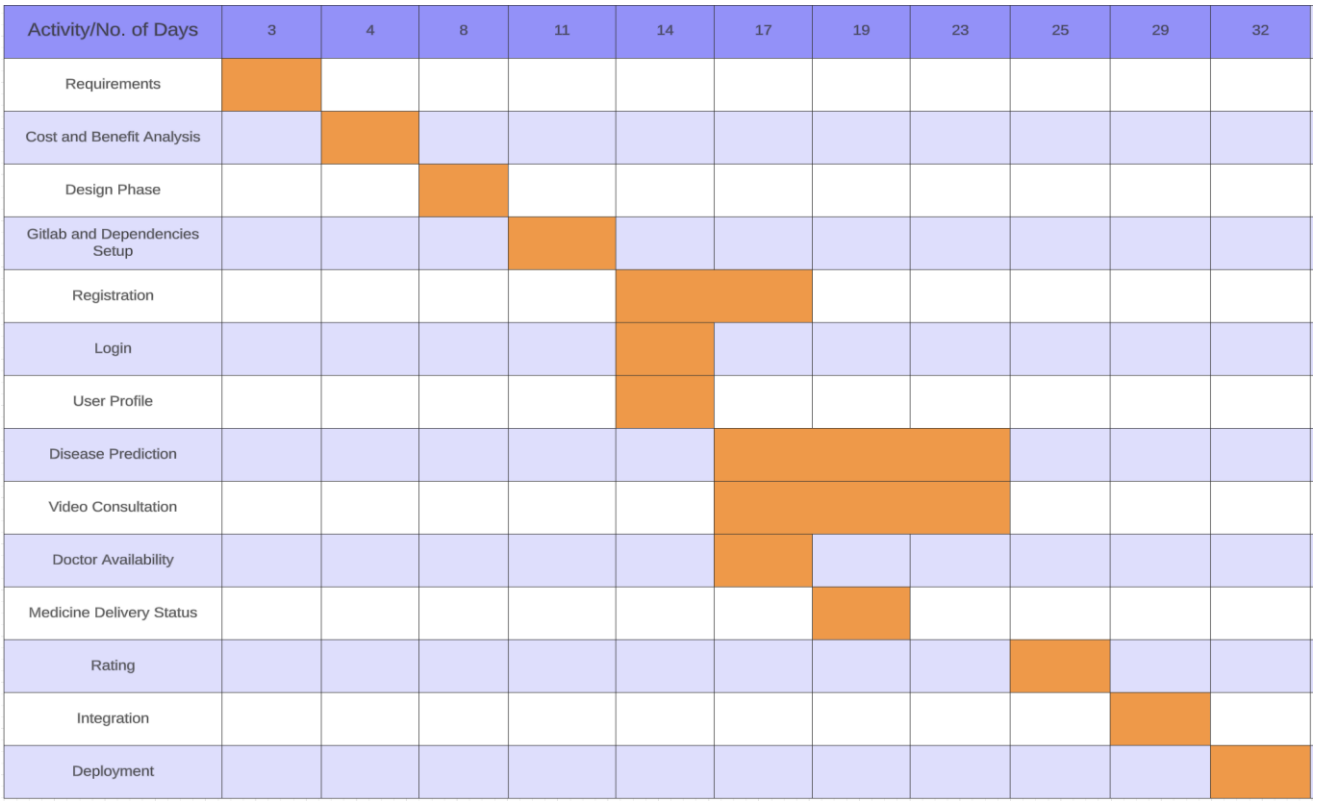
- For Stage 3: Taking DATA and CPLX to be Low as per the project.
- RELY, DOCU and TOOL to be High as per the project.

$PM_{Adjusted}$ = 13.81 * (0.85) = 11.73
Taking SCED to be Low,
$TDEV_{Nominal}$ = [3.67 * $(11.73)^{(0.28 + 0.2(1.12 - 0.091)}$] * 85/100 = 10.16 Months

8.6 Earned Value Analysis and Gantt Chart

Activities with their schedules:

Activity name (Modules)	**Predecessor**	**Days**	**Hours/day (Cost)**	**Total hours (total cost)**
Requirements	-	3	2	6
Cost and BenefitAnalysis	Requirements	1	2	2
Design Phase	Cost and Benefit Analysis	4	2	8
Gitlab and Dependencies Setup (GDS)	Design Phase	3	1	3
Registration	GDS	6	2	12
Login	GDS	3	3	9
User Profile	GDS	3	1	3
Disease Prediction	User Profile	9	4	36
Video Consultation	User Profile	9	2	18
Doctor Availability	User Profile	3	2	6
Medicine Delivery Status	Doctor Availability	2	2	4
Rating	Video Consultation	2	1	2
Integration	Rating	4	2	8
Deployment	Integration	3	3	9



GANTT CHART

- **Task -** Develop and test 8 Modules of MediPro
- **Budget -** ₹1,20,000 (₹15,000 per Module)
- **Time -** 4 Months (2 Modules per Month)

At Month 3:

- 6 Modules Developed
- ₹1,05,000 spent to date

- **Planned Value =** 6 Modules * ₹15,000 = ₹90,000
- **Actual Cost =** ₹1,05,000
- **Earned Value =** 6 Modules * ₹15,000 = ₹90,000

Derived Metrics

- **Cost Variance = EV - AC =** ₹90,000 - ₹1,05,000 = - ₹15,000
- **Schedule Variance = EV - PV =** 90,000 - 90,000 = 0
- *Cost Performance Index = EV/AC = 0.85*
- **Schedule Performance Index = EV/PV =** 90000/90000 = 1

Our project is Over-Budget and is On-Schedule.

8.7 Post Code Implementation Analysis

Halstead's Software Science

- **Program length (N):** This is the total number of operator and operand occurrences in the program.
- **Vocabulary size (n):** This is the total number of distinct operators and operands in the program.
- **Program volume (V):** This is the product of program length (N) and the logarithm of vocabulary size (n), i.e., V = N*log2(n).
- **Program level (L):** This is the ratio of the number of operator occurrences to the number of operand occurrences in the program, i.e., L = n1/n2, where n1 is the number of operator occurrences and n2 is the number of operand occurrences.
- **Program difficulty (D):** This is the ratio of the number of unique operators to the total number of operators in the program, i.e., D = (n1/2) * (N2/n2).
- **Program effort (E):** This is the product of program volume (V) and program difficulty (D), i.e., E = V*D.
- **Time to implement (T):** This is the estimated time required to implement the program, based on the program effort (E) and a constant value that depends on the programming language and development environment.

- For our frontend code,

Total Program Length (N): 35448 Total
Program Vocabulary (n): 6265 Total
Program Volume (V): 281716.39Total
Program Difficulty (D): 397.5 Total
Program Effort (E): 4071150.66

Total Development Time (T): 3769.58 minutes

- For Backend Code,

Total Program Length (N): 6947 Total
Program Vocabulary (n): 1493 Total
Program Volume (V): 47185.65Total
Program Difficulty (D): 169.0 Total
Program Effort (E): 529310.63

Total Development Time (T): 490.10 minutes