

RAJALAKSHMI ENGINEERING COLLEGE

**RAJALAKSHMI NAGAR, THANDALAM – 602
105**



CS23432 SOFTWARE CONSTRUCTION LABORATORY

Laboratory Note Book

Name: Ramprasath N

Year / Branch / Section: 2nd YEAR / AIML/AC

University Register No. :2116231501129

College Roll No: 231501129

Semester: IV SEMESTER

Academic Year: 2024-2025

EX NO: 1 STUDY OF AZURE DEVOPS

AIM:

To study how to create an agile project in Azure DevOps environment.

STUDY:

Azure DevOps is a cloud-based platform by Microsoft that provides tools for DevOps practices, including CI/CD pipelines, version control, agile planning, testing, and monitoring. It supports teams in automating software development and deployment.

1. Understanding Azure DevOps

Azure DevOps consists of five key services:

1.1 Azure Repos (Version Control)

- Supports Git repositories and Team Foundation Version Control (TFVC).
- Provides features like branching, pull requests, and code reviews.

1.2 Azure Pipelines (CI/CD)

- Automates build, test, and deployment processes.
- Supports multi-platform builds (Windows, Linux, macOS).
- Works with Docker, Kubernetes, Terraform, and cloud providers (Azure, AWS, GCP).

1.3 Azure Boards (Agile Project Management)

- Manages work using Kanban boards, Scrum boards, and dashboards.
- Tracks user stories, tasks, bugs, sprints, and releases.

1.4 Azure Test Plans (Testing)

- Provides manual, exploratory, and automated testing.
- Supports test case management and tracking.

1.5 Azure Artifacts (Package Management)

- Stores and manages NuGet, npm, Maven, and Python packages
- Enables versioning and secure access to dependencies.

Getting Started with Azure DevOps:

Step 1: Create an Azure DevOps Account Visit Azure DevOps.

- Sign in with a Microsoft Account.
- Create an Organization and a Project.

Step 2: Set Up a Repository (Azure Repos) Navigate to Repos.

- Choose Git or TFVC for version control.

- Clone the repository and push your code.

Step 3: Configure a CI/CD Pipeline (Azure Pipelines) Go to Pipelines→ New Pipeline.

- Select a source code repository (Azure Repos, GitHub, etc.).
- Define the pipeline using YAML or the Classic Editor.
- Run the pipeline to build and deploy the application.

Step 4: Manage Work with Azure Boards Navigate to Boards.

- Create work items, user stories, and tasks.
- Organize sprints and track progress.

Step 5: Implement Testing (Azure Test Plans) Go to Test Plans. • Create and run test cases

- View test results and track bugs.

RESULT:

Thus, the study for the given problem statement was successfully completed.

EX NO: 2 WRITING PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

Patients

- Secure login to access personal health records.
- Book, reschedule, or cancel appointments.
- View past consultations, prescriptions, and uploaded reports.

Doctors

- Login to view daily appointment schedules.
- Access assigned patients' medical histories.
- Record diagnosis, prescribe medication, upload reports.
- Add follow-up recommendations.

Appointment Lifecycle

Each appointment progresses through the following statuses:

- **Booked:** Patient selects an available slot with a doctor.
- **Confirmed:** System notifies both parties.
- **Completed:** Doctor records consultation details.
- **Cancelled:** Appointment is voided before it occurs.

Medical Record Management

After each appointment, doctors update the patient's record with:

- Diagnosis notes
- Prescription details
- Treatment plan
- Uploaded lab/test reports
- Follow-up instructions

RESULT:

Thus, the problem statement for the given problem is successfully written

EX NO: 3 DESIGNING PROJECT USING AGILE-SCRUM METHODOLOGY BY USING AZURE.

AIM:

To plan an agile model for the given problem statement.

THEORY:

Agile planning is a part of the Agile methodology, which is a project management style with an incremental, iterative approach. Instead of using an in-depth plan from the start of the project—which is typically product-related—Agile leaves room for requirement changes throughout and relies on constant feedback from end users.

With Agile planning, a project is broken down into smaller, more manageable tasks with the goal of having a defined image of a project's vision. Agile planning involves looking at different aspects of a project's tasks and how they'll be achieved, for example:

- Roadmaps to guide a product's release ad schedule

- Sprints to work on one specific group of tasks at a time
- A feedback plan to allow teams to stay flexible and easily adapt to change

User stories, or the tasks in a project, capture user requirements from the end user's perspective Essentially, with Agile planning, a team would decide on a set of user stories to action at any given time, using them as a guide to implement new features or functionalities in a tool. Looking at tasks as user stories is a helpful way to imagine how a customer may use a feature and helps teams prioritize work and focus on delivering value first.

STEPS IN AGILE PLANNING PROCESS:

1. Define vision
2. Set clear expectations on goals
3. Define and break down the product roadmap
4. Create tasks based on user stories
5. Populate product backlog
6. Plan iterations and estimate effort

7. Conduct daily stand-ups

8. Monitor and adapt

RESULT:

Thus, the designing project using agile-scrum methodology by using azure was completed successfully.

EX NO: 4 AGILE PLANNING

Aim:

To develop a system this speaks between a doctor and a patient. Where the patient book appointment to see their doctors.

Scope:

The scope of the Hospital Management System is that the system has two different login one is doctor and the another is the patient. Then make a patient able to book appointment in the system. And the doctors able to make changes in their patients report

Epic 1: Doctor Access

Objective: To make the doctor can able to access the system.

User Stories:

- As a doctor I can able to update my patients record.
- As a doctor I need my separate Id for the login.
- As a doctor I need to give prescriptions for my patients.

Epic 2: Patient Access

Objective: To make patient use the system effectively.

User Stories:

- As a patient I can able to see my previous record.
- As a patient I need my separate Id for the login.
- As a patient I need to see prescriptions given by doctors.

Epic 3: Appointment Scheduling

Objective: This is about how the appointment of the patient will be mailed to him/her with the check of doctor's availability and they have their checkup.

User Stories:

- As a patient I need to make sure the doctor is free for my time.
- As a doctor I will meet the patient when I was free.
- As a system I send the alert for the patient about the availability.

Sprint Plan – Hospital Management System

Sprint 1: User Management and Authentication

Duration: 2 weeks

Focus Area: Role-based access and login system

Covered Epics: Epic 1

User Stories:

- Implement secure login functionality for patients and doctors.
- Set up role-based dashboards (Patient / Doctor).
- Enable profile creation and updates for both user types.

Sprint 2: Appointment Scheduling Module

Duration: 2 weeks

Focus Area: Appointment booking and management

Covered Epics: Epic 2

User Stories:

- Allow patients to view doctor availability and book appointments.
- Enable doctors to view their daily schedule.
- Implement appointment status flow (Booked, Confirmed, Completed, Cancelled).

Sprint 3: Medical Record Management

Duration: 2 weeks

Focus Area: Consultation data and health record tracking

Covered Epics: Epic 3

User Stories:

- Enable doctors to enter diagnosis, prescriptions, and treatment plans.
- Allow doctors to upload lab/test reports.
- Allow patients to view their complete medical history.

Sprint 4: Notification and Follow-up System

Duration: 2 weeks

Focus Area: Communication and reminders

Covered Epics: Epic 4

User Stories:

- Send appointment confirmations and reminders to patients.
- Notify patients when new reports or prescriptions are uploaded.
- Alert doctors for critical cases or required follow-ups.

Sprint 5: Final Testing and Deployment

Duration: 1 week

Focus Area: System testing and go-live

Covered Epics: All

User Stories:

- Conduct integration and end-to-end workflow testing.
- Perform user acceptance testing with sample users.
- Deploy the HMS and monitor for feedback or bug reports.

RESULT:

Thus, the agile plan for the fitness app has been successfully created using Epics, User Stories, and a Sprint-based approach for structured development.

EX NO: 5 USER STORIES – CREATION

AIM:

To create User Stories for the given problem statement.

THEORY:

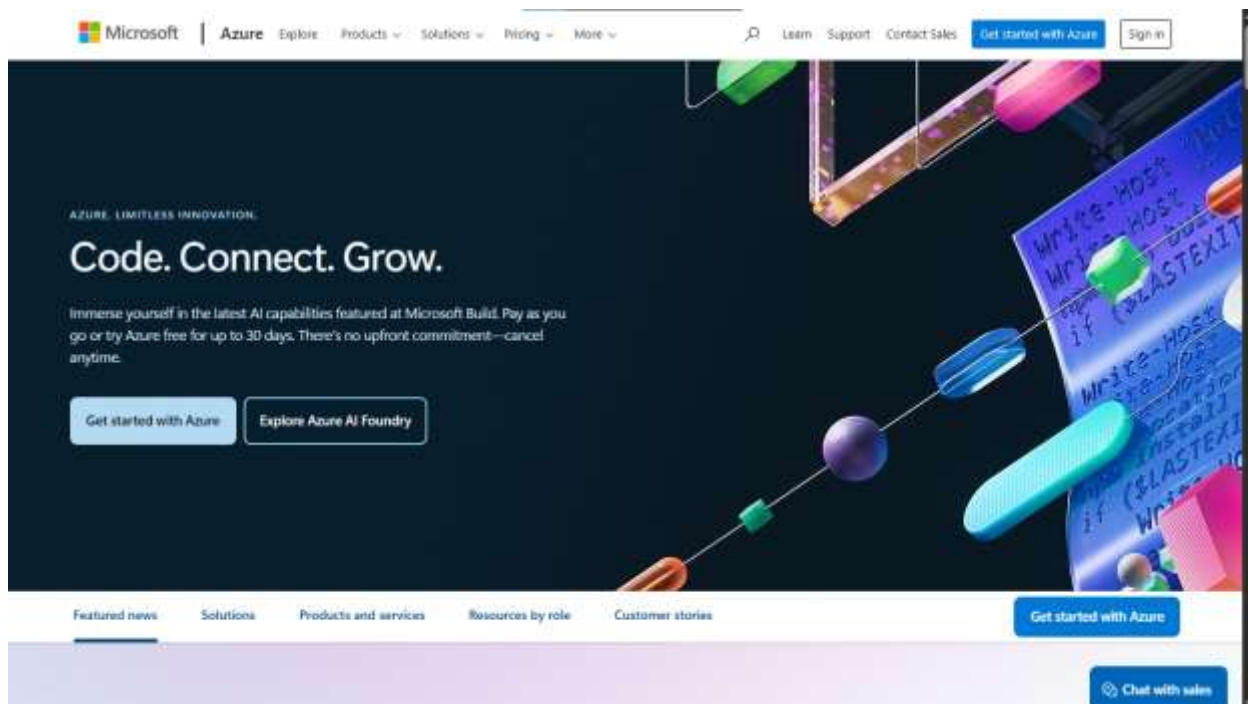
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

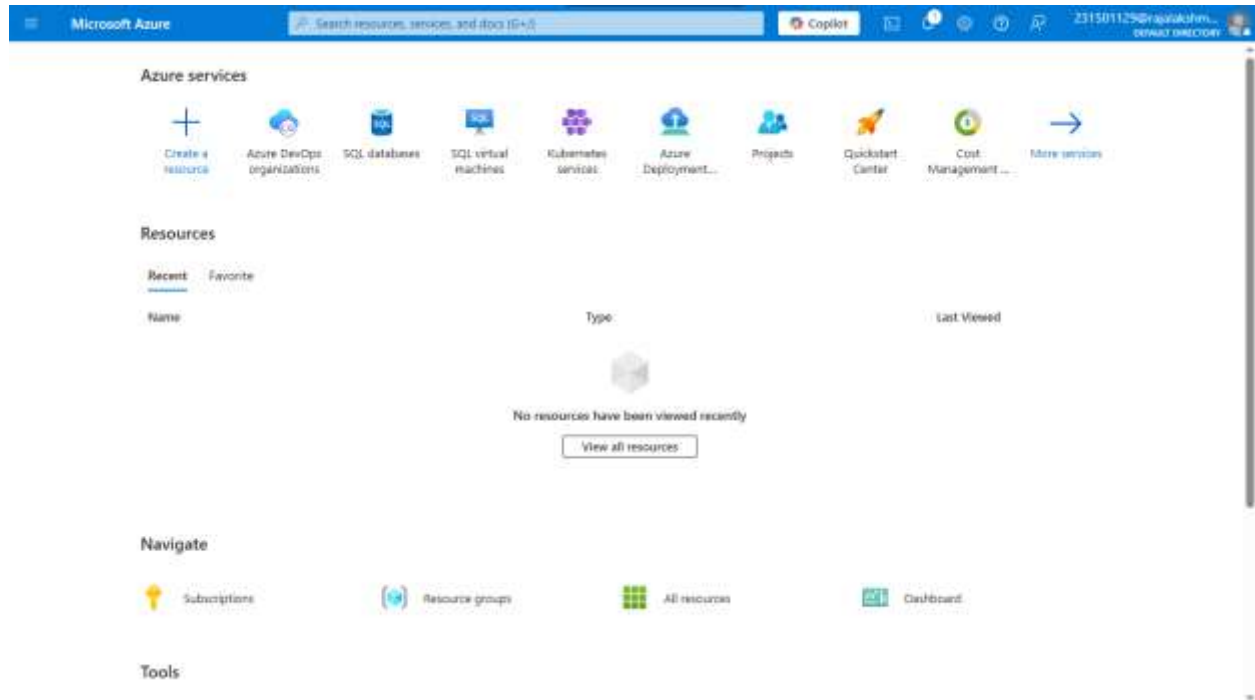
"As a [role], I [want to], [so that]."

PROCEDURE:

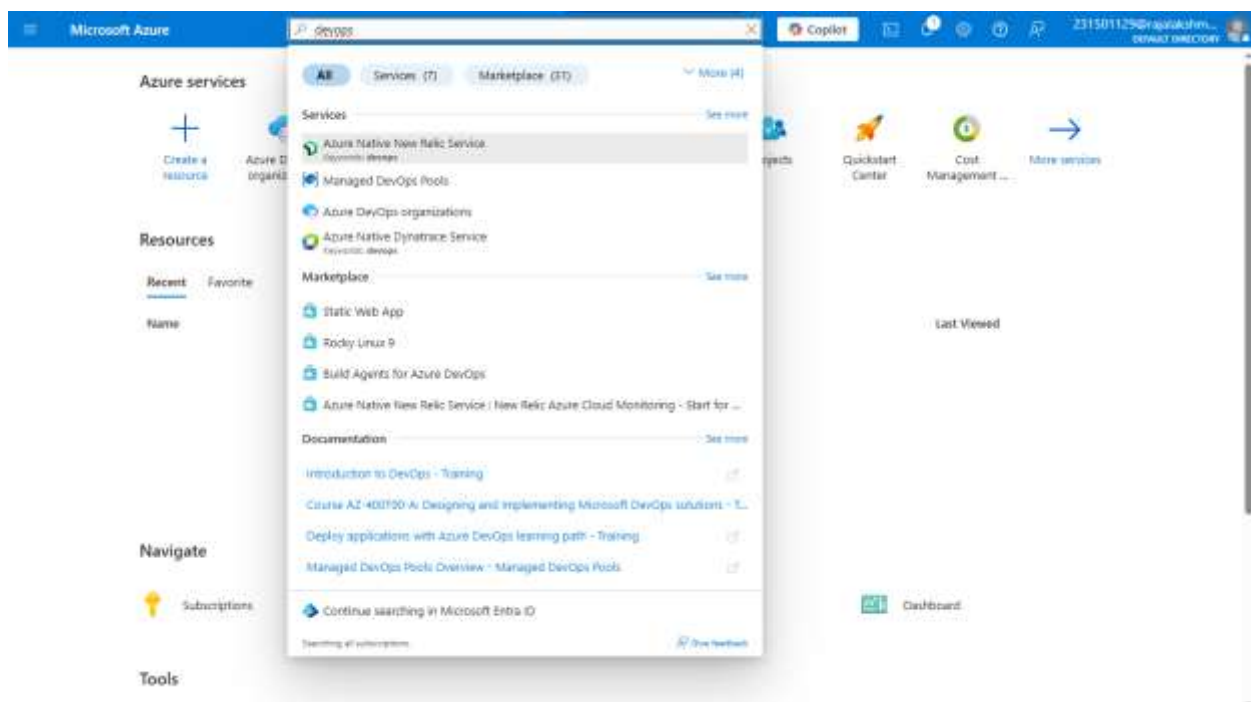
1. Open your web browser and go to the Azure website: <https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for <https://signup.live.com/?lic=1>



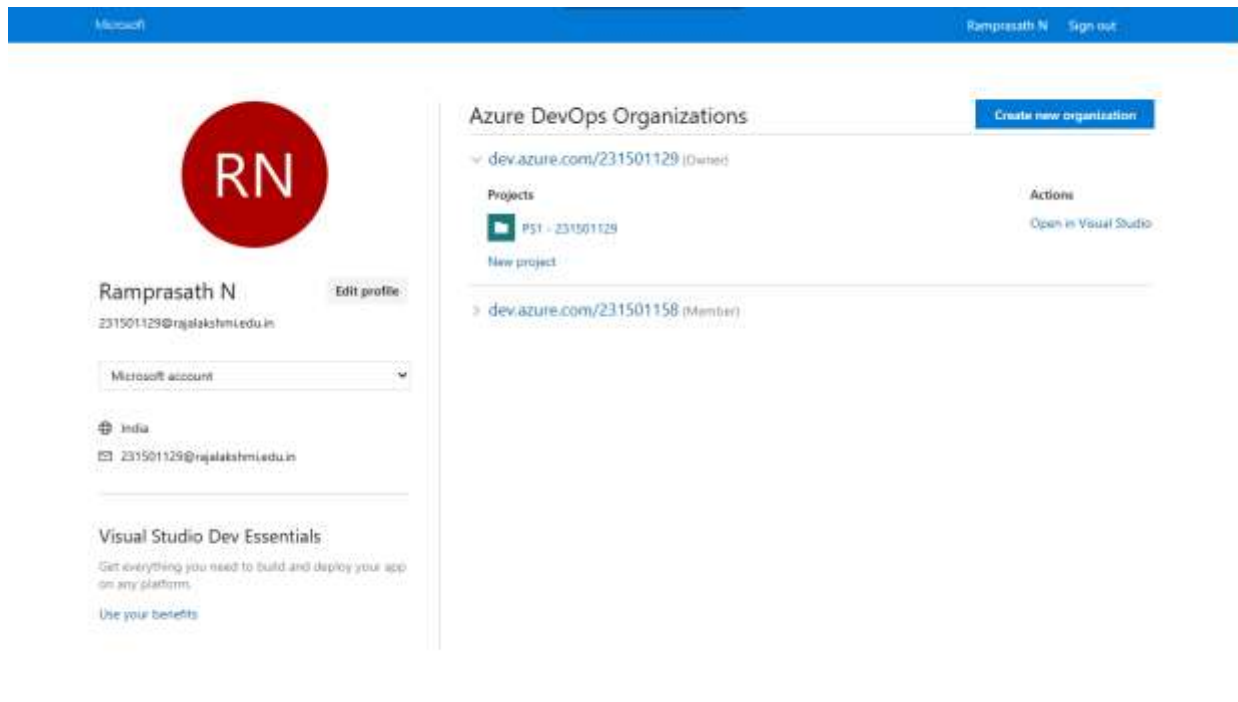
3. Azure Home



4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



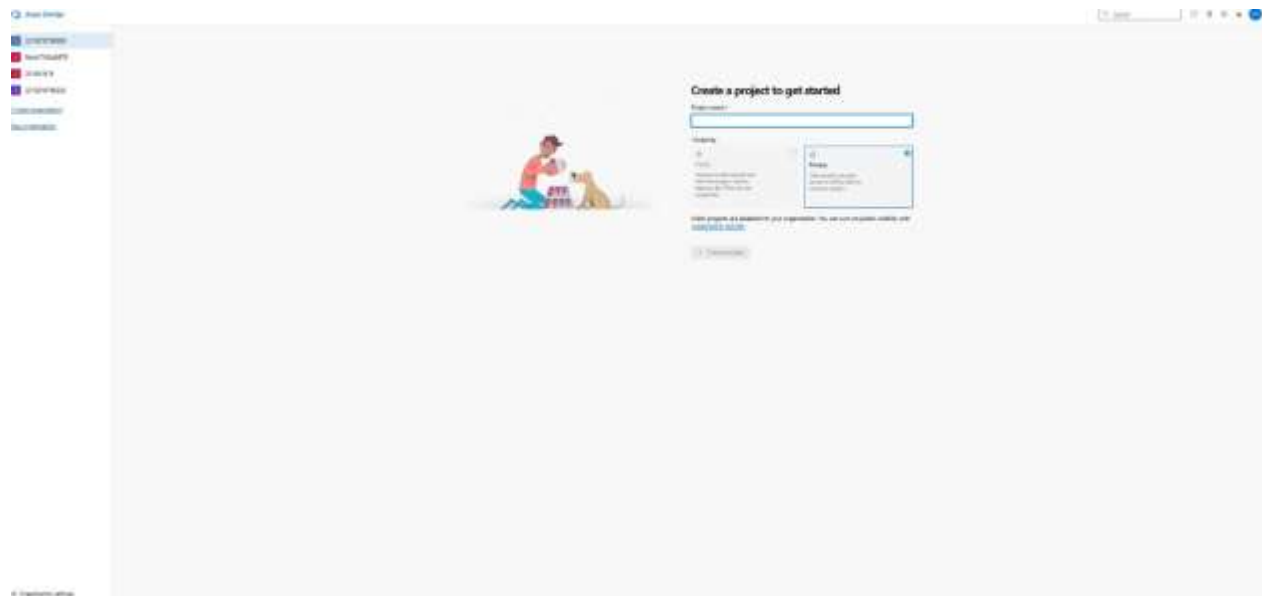
My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.



6. Create the First Project in Your Organization

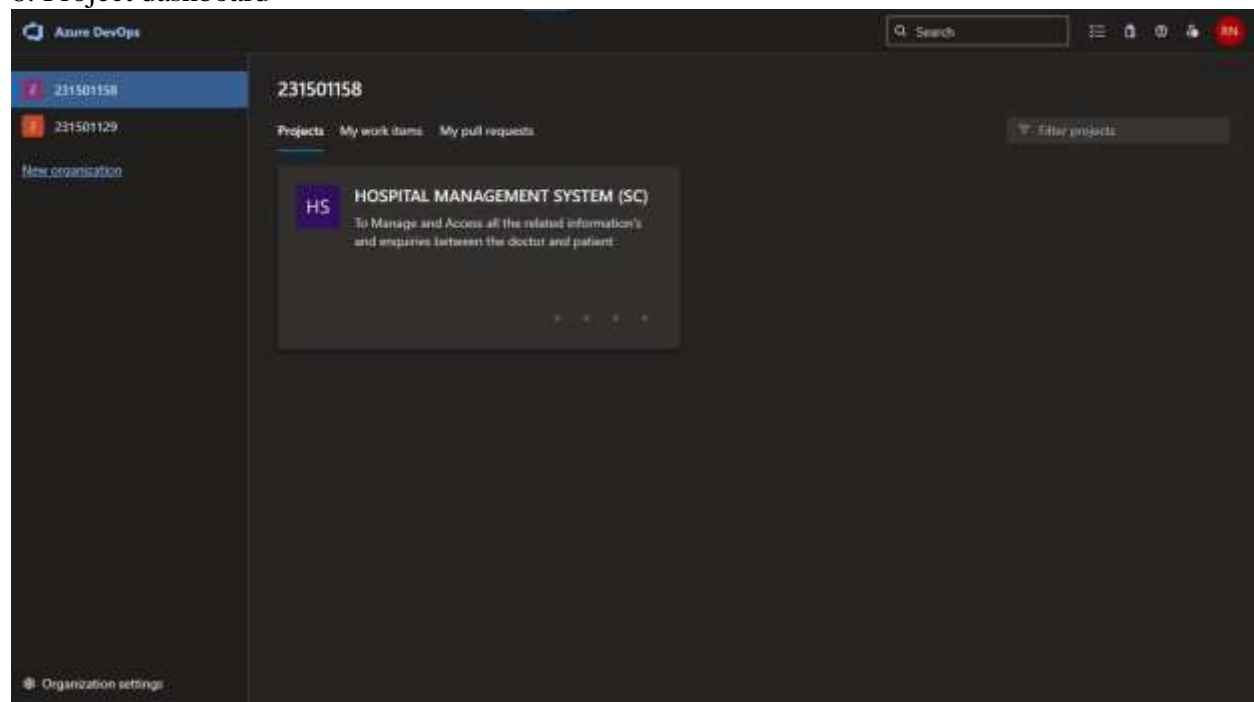
- i. **Home page**, click on the **New Project** button.
- ii. Enter the project name, description, and visibility options:
 - **Name**: Choose a name for the project (e.g., **HMS**).
 - **Description**: Optionally, add a description to provide more context about the project.
 - **Visibility**: Choose whether you want the project to be **Private** (accessible only to those invited) or **Public** (accessible to anyone).

Once you've filled out the details, click **Create** to set up your first project.



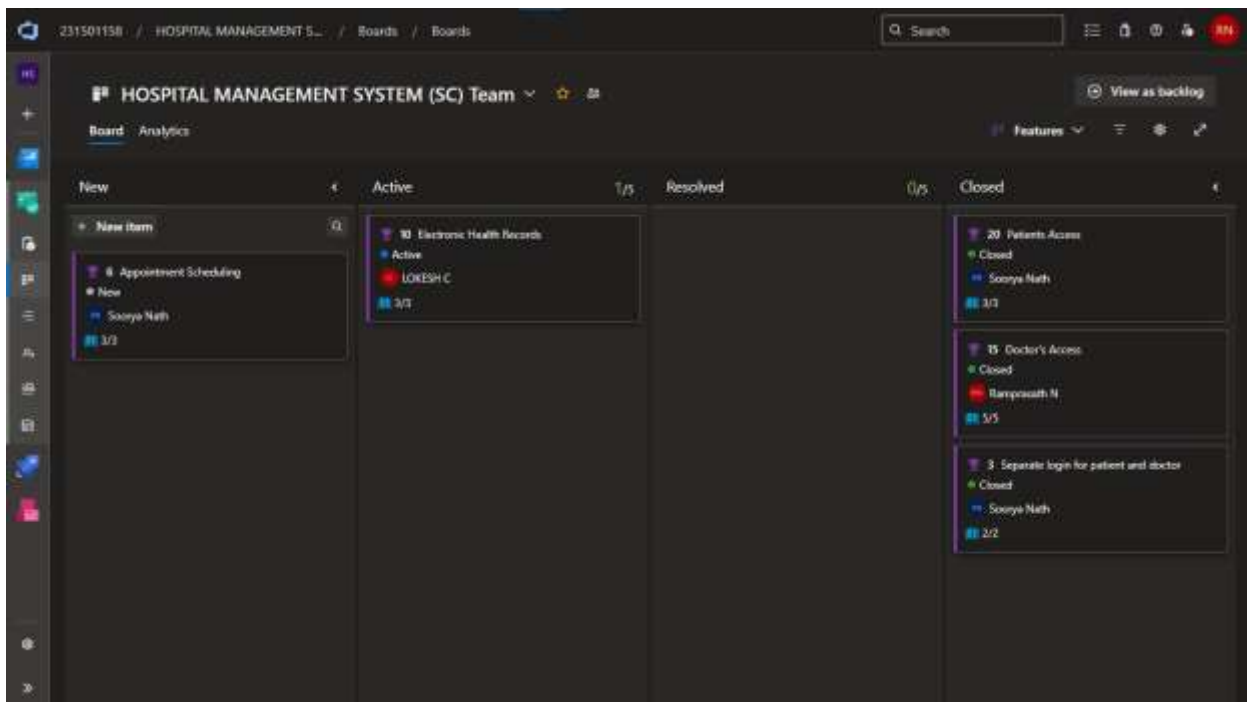
7. Once logged in, ensure you are in the correct organization. If you're part of multiple organizations, you can switch between them from the top left corner (next to your user profile). Click on the Organization name, and you should be taken to the Azure DevOps Organization Home page.

8. Project dashboard

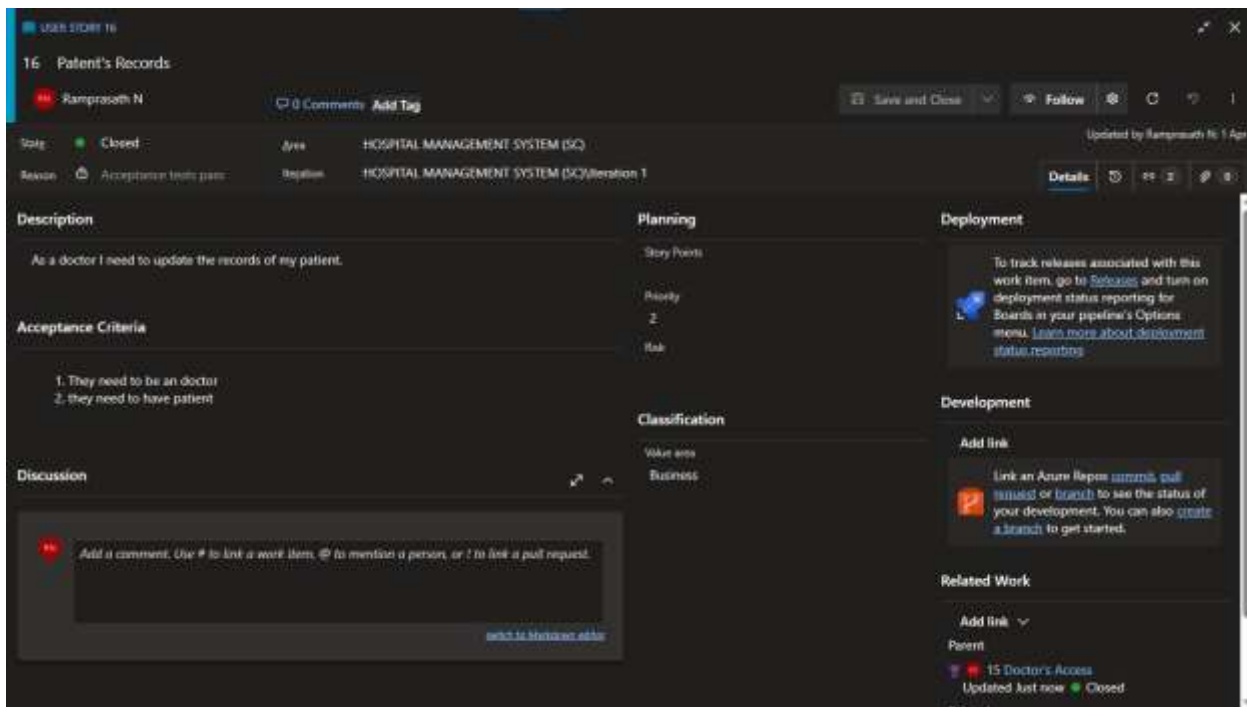


9. To manage user stories

a. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main Boards page, where you can manage work items, backlogs, and sprints. b. On the **work items** page, you'll see the option to **Add a work item** at the top. Alternatively, you can find a + button or **Add New Work Item** depending on the view you're in. From the **Add a work item** dropdown, select **User Story**. This will open a form to enter details for the new User Story.



10. Fill in User Story Details



Result:

The user story for the given problem statement was written successfully.

EX NO: 6 SEQUENCE DIAGRAM

AIM:

To design a Sequence Diagram by using Mermaid.js for the given problem statement.

THEORY:

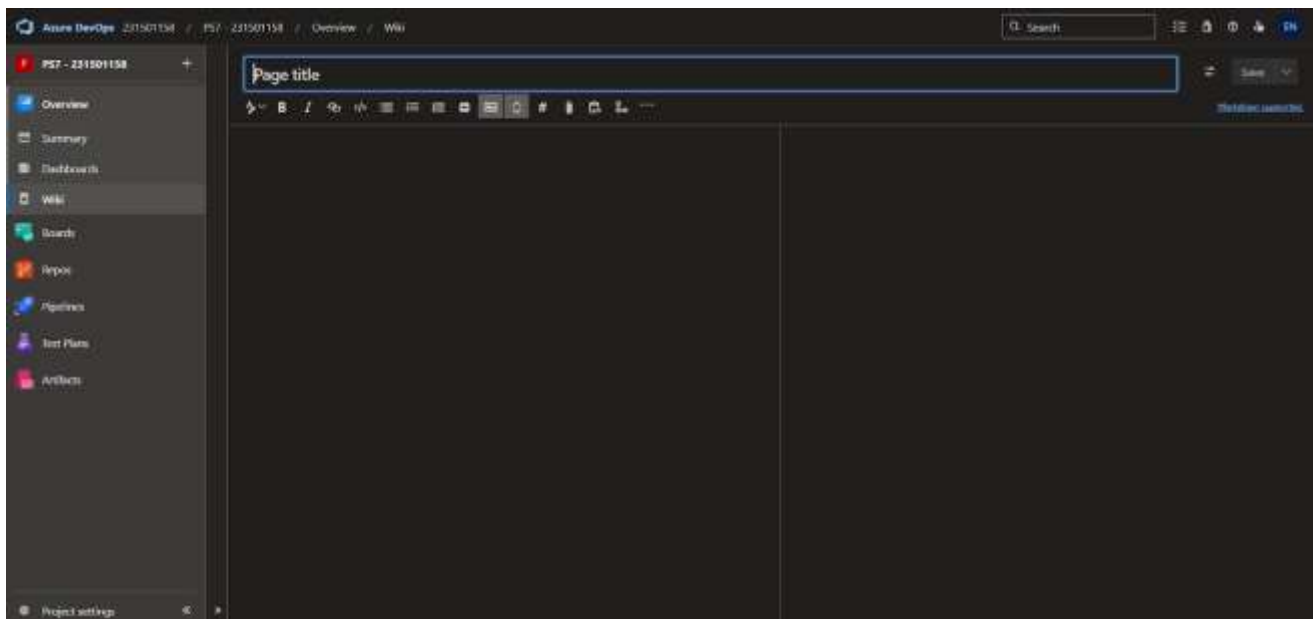
A Sequence Diagram is a key component of Unified Modelling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modelling dynamic behavior in a system.

PROCEDURE:

1. Open a project in Azure DevOps Organizations.
2. To design select wiki from menu.

Write code for drawing sequence diagram and save the code :::: mermaid
sequence sequence Diagram:
::: mermaid

sequenceDiagram



Patient->>+HMS: New User
HMS-->>-Patient: You are added as User
Doctor->>+HMS:New Doctor
Doctor-->>+HMS: you are added as Doctor
Doctor->>+HMS: Updates Records
Patient->>+HMS: View Records
HMS-->>-Patient: Records
Patient->>+HMS: Book appointment
HMS->>+Doctor: Free Dates
Doctor-->>-HMS: Available Dates
HMS-->>-Patient: Appointment Booked
Doctor->>+HMS: EHR
Patient->>+HMS: View EHR
HMS-->>-Patient: EHR results

:::

EXPLANATION:

Hospital Management System – Interaction Flow Description

1. PATIENT → HMS (Hospital Management System):

The process begins when a new patient registers or logs into the system to access healthcare services such as viewing records or booking appointments.

2. HMS → PATIENT:

The system verifies the patient details and confirms that the patient has been successfully added as a registered user.

3. DOCTOR → HMS:

A doctor registers into the system, entering their credentials and specialization for administrative approval.

4. HMS → DOCTOR:

The doctor is successfully added to the system and gains access to their dashboard to manage appointments and patient data.

5. **DOCTOR → HMS:**

The doctor updates patient records, which may include diagnosis details, prescriptions, treatment plans, or uploaded reports.

6. **PATIENT → HMS:**

A patient accesses the system to view their medical records including past diagnoses and uploaded test reports.

7. **HMS → PATIENT:**

The system retrieves and displays the requested medical records to the patient.

8. **PATIENT → HMS:**

The patient initiates the process to book an appointment with a doctor by selecting a preferred date or time.

9. **HMS → DOCTOR:**

The system queries the doctor's schedule to find available time slots for the requested appointment.

10. **DOCTOR → HMS:**

The doctor responds with their available dates and times for appointments.

11. **HMS → PATIENT:**

The system confirms the appointment has been successfully booked based on the doctor's availability.

12. **DOCTOR → HMS:**

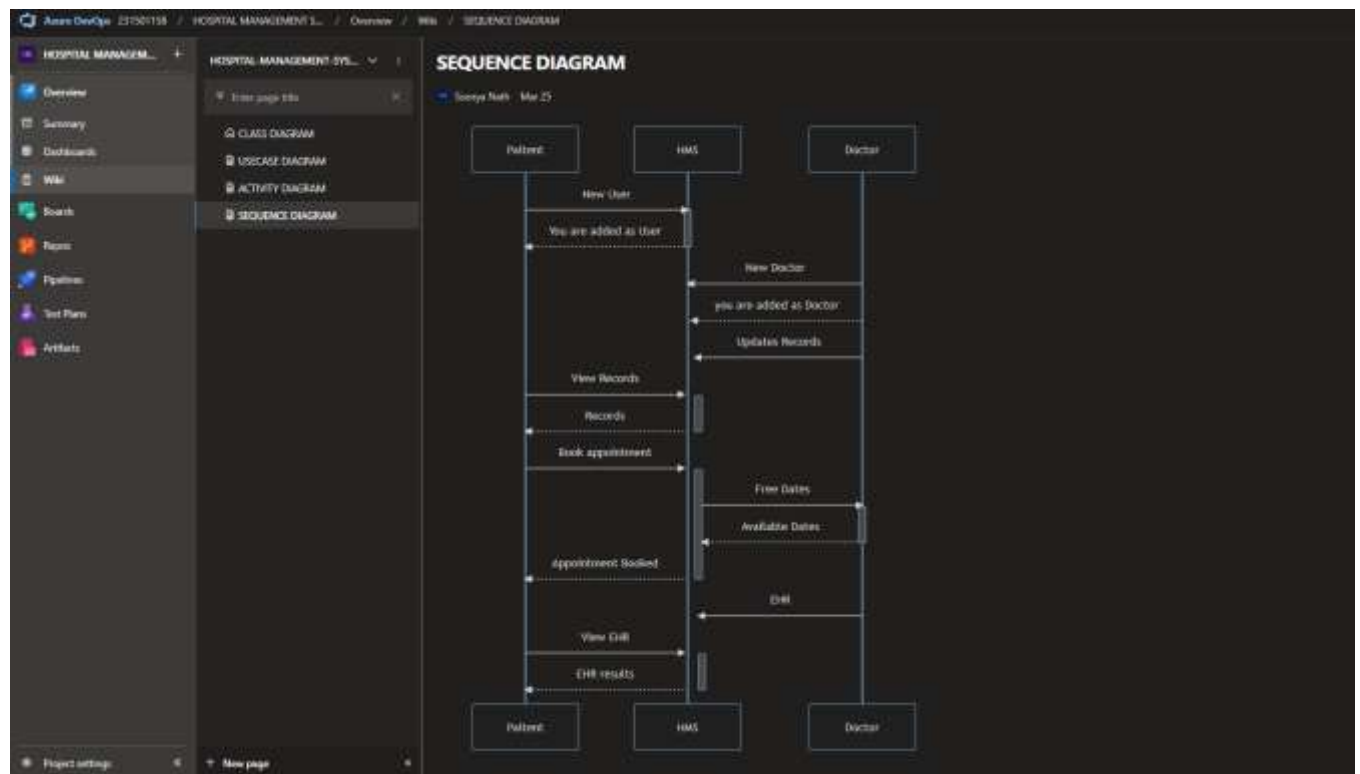
During or after the consultation, the doctor inputs Electronic Health Record (EHR) data for the patient.

13. PATIENT → HMS:

The patient logs in to view their Electronic Health Record and check the outcome or follow-up instructions.

14. HMS → PATIENT:

The system displays the EHR details and any attached documents (e.g., prescriptions or lab results) to the patient.



RESULT:

Thus, the sequence diagram for the given problem statement was drawn successfully.

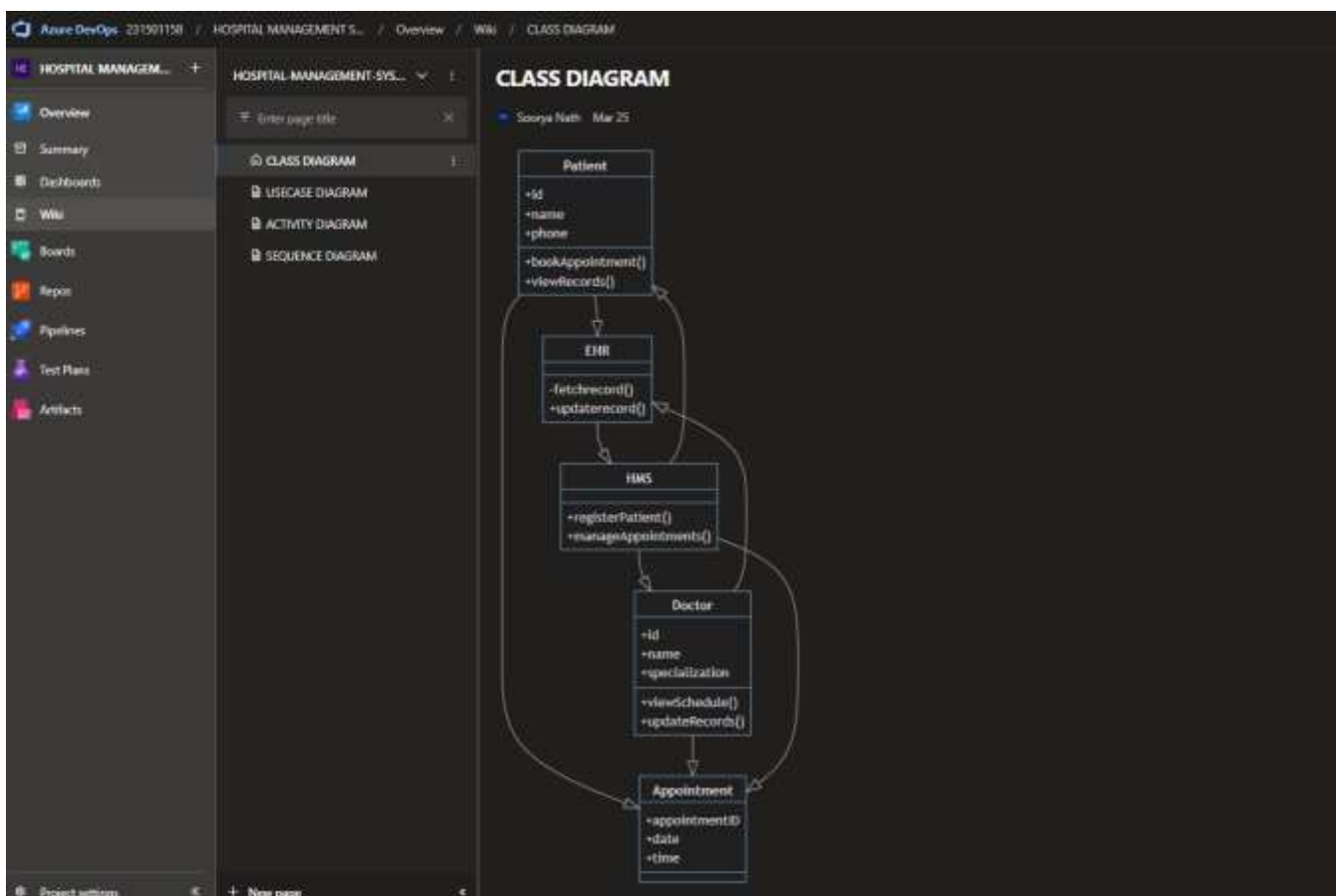
EX NO: 7 CLASS DIAGRAM

AIM:

To draw a sample class diagram for your project or system.

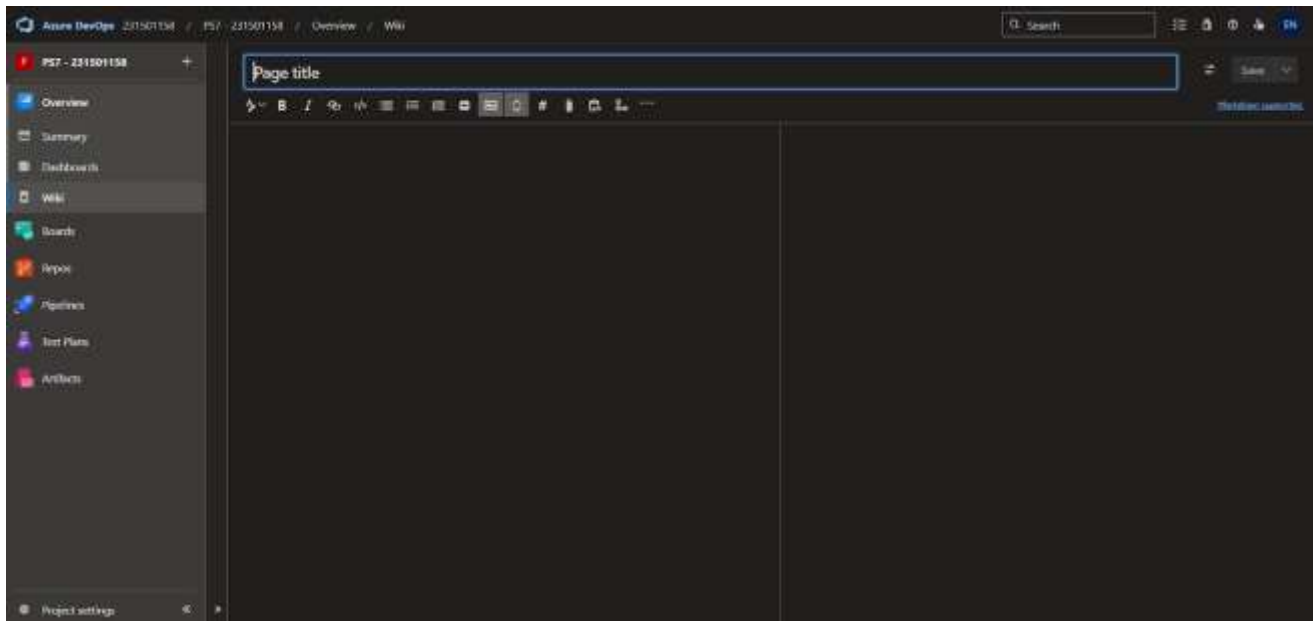
THEORY:

A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



PROCEDURE:

1. Open a project in Azure DevOps Organizations.
2. To design select wiki from menu.



3. Write code for drawing class diagram and save the code

```mermaid

```mermaid

classDiagram

class Patient {

+id

+name

+phone

+bookAppointment()

+viewRecords()

}

class Doctor {

+id

+name

+specialization

+viewSchedule()

+updateRecords()

}

class Appointment {

+appointmentID

+date

+time

}

class HMS {

+registerPatient()

+manageAppointments()

}

class EHR{

```
-fetchrecord()
+updaterecord()
}
Patient --|> Appointment
Doctor --|> Appointment
HMS --|> Patient
HMS --|> Doctor
HMS --|> Appointment
EHR --|> HMS
Patient --|> EHR
Doctor --|> EHR
```  

::
```

## **RESULT:**

Thus, the use case diagram for the given problem statement was designed successfully.

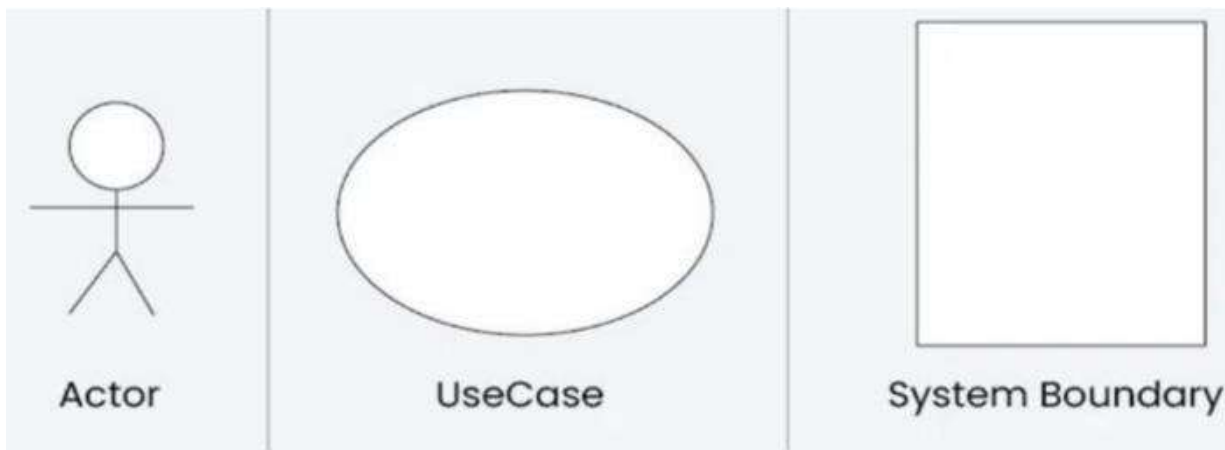
## EX NO: 8 USE CASE DIAGRAM

### AIM:

Steps to draw the Use Case Diagram using draw.io

### THEORY:

- UCD(USER CASE DIAGRAM) shows the relationships among actors and use cases within a system which Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model and communicate the scope of a development project
- Use Cases
- Actors
- Relationships
- System Boundary Boxes



### PROCEDURE:

Step 1: Create the Use Case Diagram in Draw.io

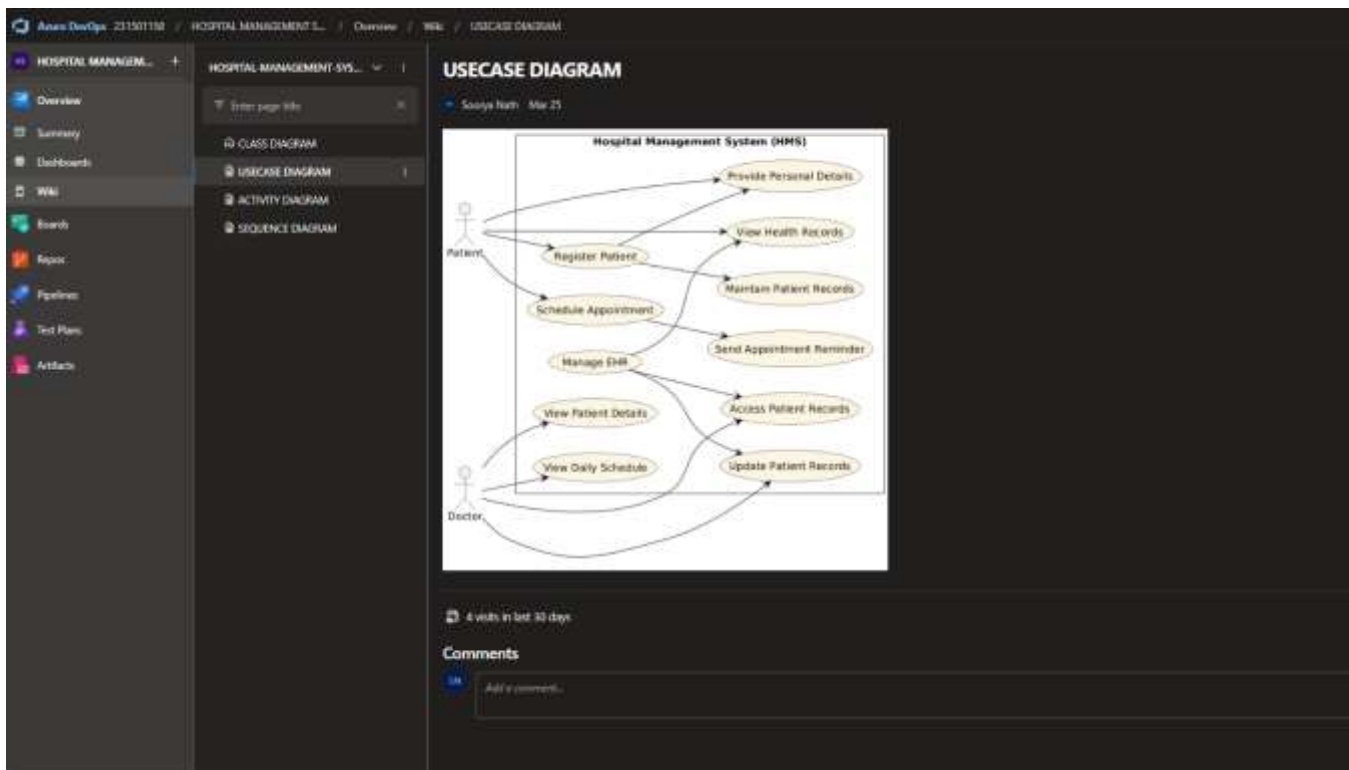
- Open Draw.io (diagrams.net).
- Click "Create New Diagram" and select "Blank" or "UML Use Case" template. • Add Actors (Users, Admins, External Systems) from the UML section.
- Add Use Cases (Functionalities) using ellipses.
- Connect Actors to Use Cases with lines (solid for direct interaction, dashed for

<<include>> and <<extend>>).

- Save the diagram as .draw io or export as PNG/JPG/SVG. Step 2: Upload the Diagram to Azure DevOps

#### Option 1: Add to Azure DevOps Wiki

- Open Azure DevOps and go to your project.
- Navigate to Wiki (Project > Wiki).
- Click "Edit Page" or create a new page.
- Drag & drop the exported PNG/JPG image.
- Use Markdown to embed the diagram:  
! [Use Case Diagram]  
(attachments/use\_case\_diagram.png) Option 2: Attach to Work Items in Azure Boards
- Open Azure DevOps → Navigate to Boards (Project > Boards). • Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram
- Add comments or descriptions to explain the use case Diagram.





**RESULT:**

The use case diagram for the given problem statement was designed successfully.



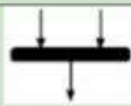



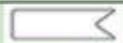




## EX NO: 9 ACTIVITY DIAGRAM

### AIM:

To draw a sample activity diagram for your project or system.

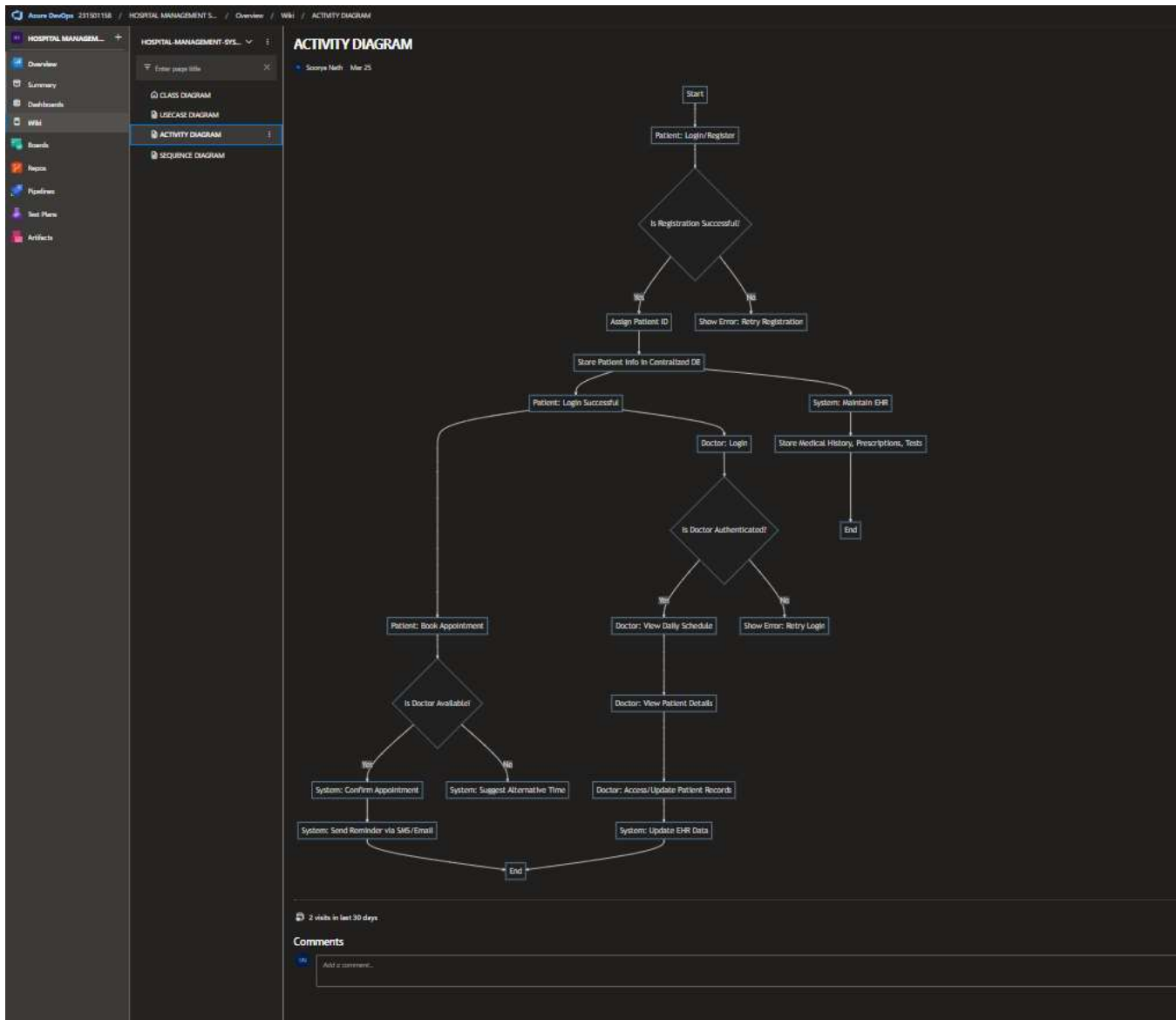
### THEORY:

Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

Notations	Symbol	Meaning
Start		Shows the beginning of a process
Connector		Shows the directional flow, or control flow, of the activity
Joint symbol		Combines two concurrent activities and re-introduces them to a flow where one activity occurs at a time
Decision		Represents a decision
Note		Allows the diagram creators to communicate additional messages
Send signal		Show that a signal is being sent to a receiving activity
Receive signal		Demonstrates the acceptance of an event
Flow final symbol		Represents the end of a specific process flow
Option loop		Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate		Represents a transition that invokes the last active state.
End		Marks the end state of an activity and represents the completion of all flows of a process

### PROCEDURE:

1. Draw diagram in draw.io
2. Upload the diagram in the Azure Wiki



## Diagram Code:

::: mermaid

flowchart TD

%% Patient Registration Flow

A[Start] --> B[Patient: Login/Register]

B --> C{Is Registration Successful?}

C -->|Yes| D[Assign Patient ID]

C -->|No| E[Show Error: Retry Registration]

D --> F[Store Patient Info in Centralized DB]

F --> G[Patient: Login Successful]

%% Appointment Booking Flow

G --> H[Patient: Book Appointment]

H --> I{Is Doctor Available?}

I -->|Yes| J[System: Confirm Appointment]  
I -->|No| K[System: Suggest Alternative Time]  
J --> L[System: Send Reminder via SMS/Email]  
L --> M[End]

%% Doctor Interaction Flow

G --> N[Doctor: Login]  
N --> O{Is Doctor Authenticated?}  
O -->|Yes| P[Doctor: View Daily Schedule]  
O -->|No| Q[Show Error: Retry Login]  
P --> R[Doctor: View Patient Details]  
R --> S[Doctor: Access/Update Patient Records]  
S --> T[System: Update EHR Data]  
T --> M

%% System: EHR Maintenance

F --> U[System: Maintain EHR]  
U --> V[Store Medical History, Prescriptions, Tests]  
V --> W[End]

...

## RESULT:

Thus, the Activity diagram for the above problem statement done successfully.

## EX NO: 10 ARCHITECTURE DIAGRAM

### AIM:

Steps to draw the Architecture Diagram using draw.io.

### THEORY:

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.



### PROCEDURE:

1. Draw diagram in draw.io
2. Upload the diagram in Azure DevOps wiki

### RESULT:

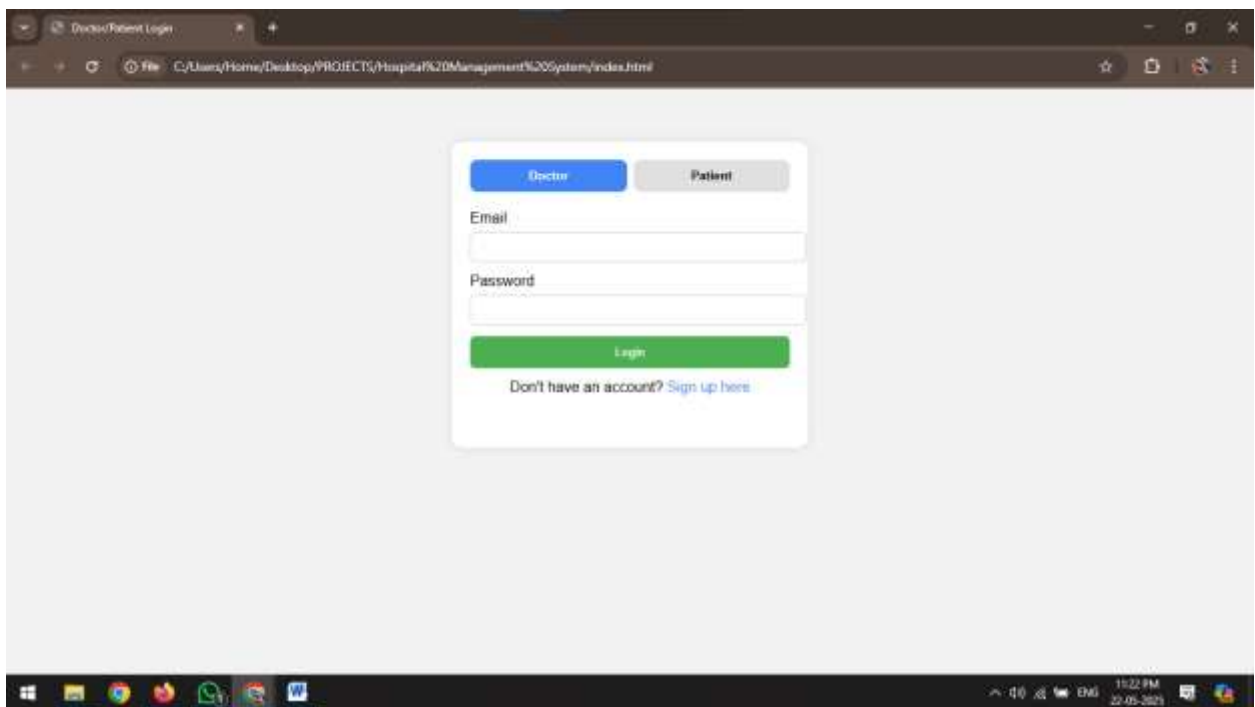
Thus, the architecture diagram for the given problem statement was designed successfully.

## EX NO: 11 USER INTERFACE

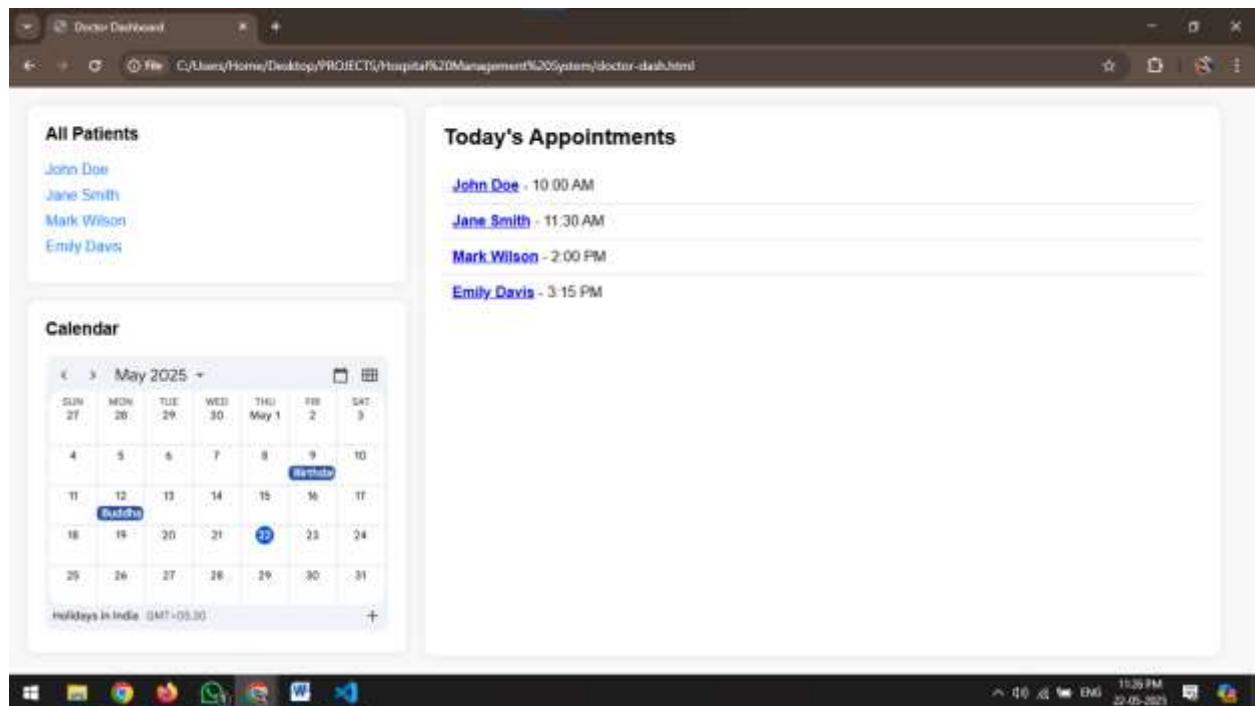
### AIM:

Design User Interface for the given project.

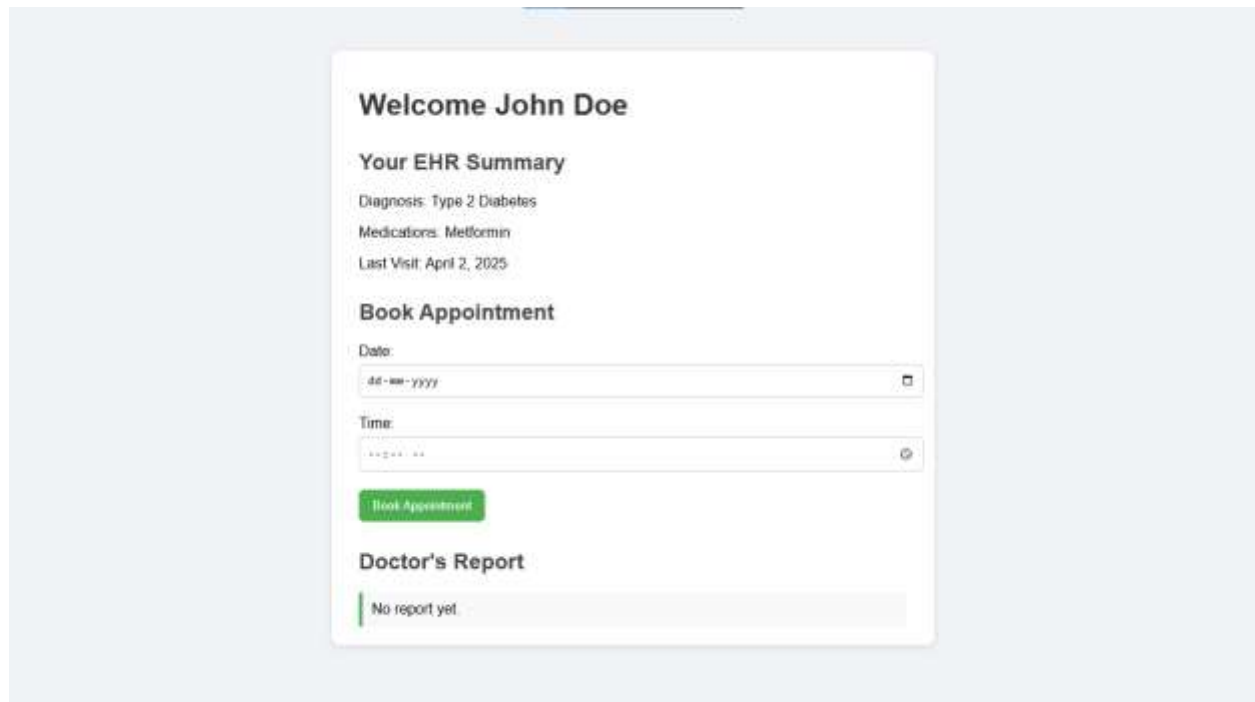
### LOGIN PAGE



### Doctors Dashboard



## Patients Dashboard



**RESULT:**

Thus, the UI for the given problem statement is completed successfully.



## **EX NO: 12 IMPLEMENTATIONS**

### **AIM:**

To implement the given project based on Agile Methodology.

### **PROCEDURE:**

#### **Step 1:** Set Up an Azure DevOps Project

- Log in to Azure DevOps.
- Click "New Project" → Enter project name → Click "Create".
- Inside the project, navigate to "Repos" to store the code.

#### **Step 2:** Add Your Web Application Code

- Navigate to Repos → Click "Clone" to get the Git URL.
- Open Visual Studio Code / Terminal and run: `git clone cd`
- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.).
- Commit & push: `git add. git commit -m "Initial commit" git push origin main`

#### **Step 3:** Set Up Build Pipeline (CI/CD - Continuous Integration)

- Navigate to Pipelines → Click "New Pipeline".
- Select Git Repository (Azure Repos, GitHub, or Bitbucket).
- Choose Starter Pipeline or a pre-configured template for your framework.

- Modify the azure-pipelines.yml file (Example for a Node.js app):

trigger:

- main

pool:

vmImage: 'ubuntu-latest'

steps:

task: UseNode@1

inputs:

version: '16.x'

-script: npm install

displayName: 'Install dependencies'

-script: npm run build

displayName: 'Build application'

-task: PublishBuildArtifacts@1

inputs:

pathToPublish: 'dist'

artifactName: 'drop'

Click "Save and Run" → The pipeline will start building app.

**Step 4:** Set Up Release Pipeline (CD - Continuous

Deployment) • Go to Releases → Click "New

Release Pipeline".

- Select Azure App Service or Virtual Machines (VMs) for deployment.
- Add an artifact (from the build pipeline).
- Configure deployment stages (Dev, QA, Production).

- Click "Deploy" to push your web app to Azure.

**RESULT:**

Thus, the implementation of the given problem statement is done successfully.