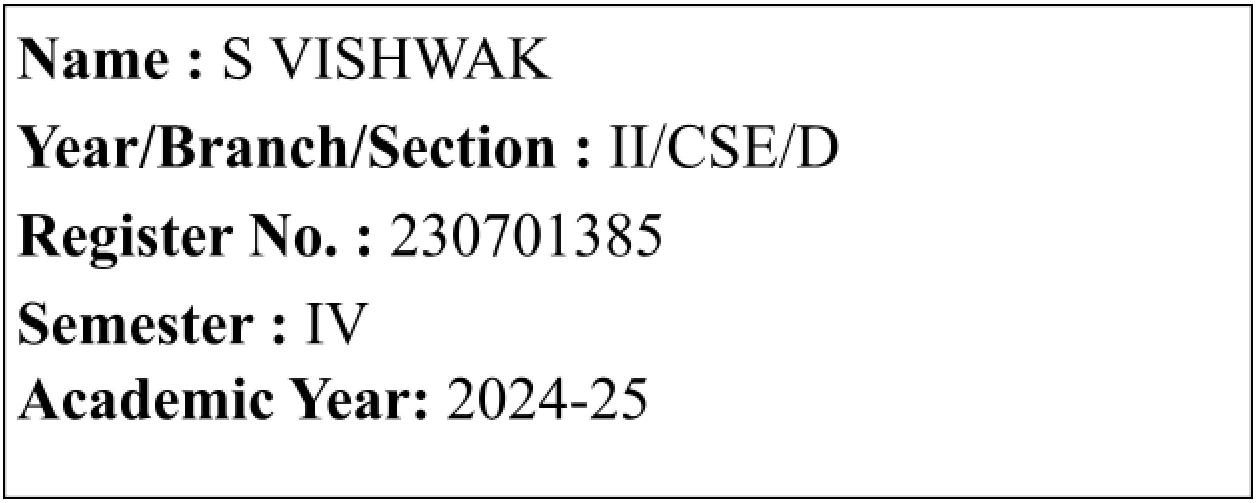
RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23432
SOFTWARE ENGINEERING LAB

LAB MANUAL



Name : S VISHWAK

Year/Branch/Section : II/CSE/D

Register No. : 230701385

Semester : IV

Academic Year: 2024-25

Exp No.	List of Experiments
1	Study of Azure DevOps
2	Designing Project using AGILE-SCRUM Methodology.
3	Agile Planning
4	User stories – Creation
5	Architecture Diagram Using AZURE
6	Designing Usecse and Class Diagram
7	Designing Interaction Diagrams
8	Design Interface
9	Implementation – Design a Web Page based on Scrum Methodology
10	Testing using Azure.
11	Deployment

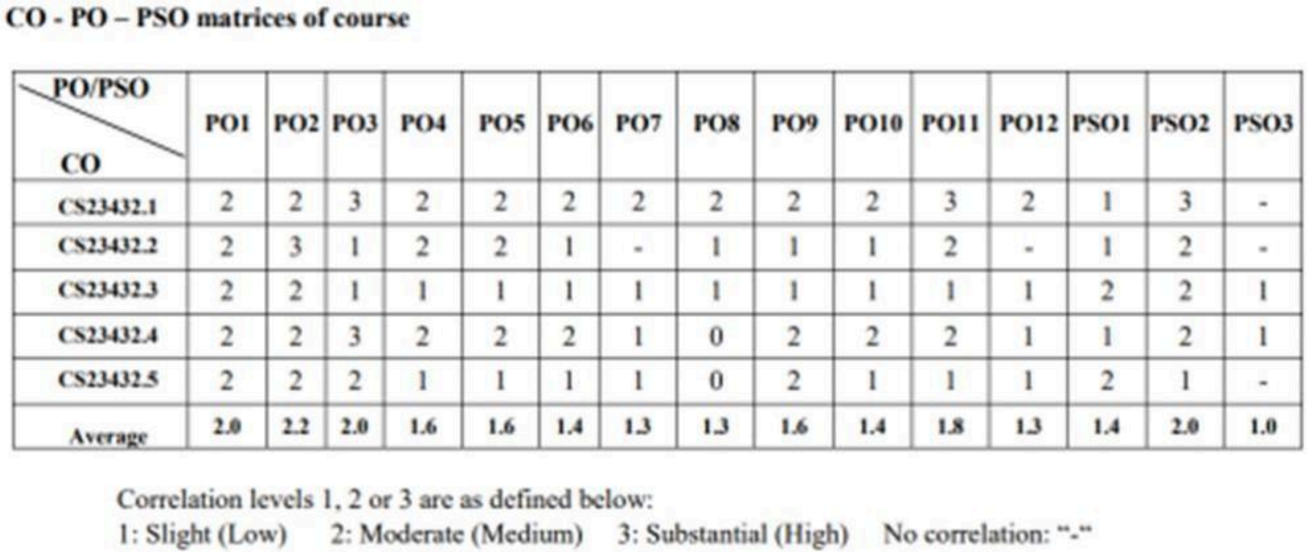
Requirements	
Hardware	Intel i3, CPU @ 1.20GHz 1.19 GHz, 4 GB RAM, 32 Bit Operating System
Software	StarUML , Azure

Course Outcomes (COs)

Course Name: Software Engineering

Course Code: CS23432

CO 1	**Understand the software development process models.**
CO 2	**Determine the requirements to develop software**
CO 3	**Apply modeling and modeling languages to design software products**
CO 4	**Apply various testing techniques and to build a robust software products**
CO 5	**Manage Software Projects and to understand advanced engineering concepts**



CO - PO – PSO matrices of course

PO/PSO CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CS23432.1	2	2	3	2	2	2	2	2	2	2	3	2	1	3	-
CS23432.2	2	3	1	2	2	1	-	1	1	1	2	-	1	2	-
CS23432.3	2	2	1	1	1	1	1	1	1	1	1	1	2	2	1
CS23432.4	2	2	3	2	2	2	1	0	2	2	2	1	1	2	1
CS23432.5	2	2	2	1	1	1	1	0	2	1	1	1	2	1	-
Average	**2.0**	**2.2**	**2.0**	**1.6**	**1.6**	**1.4**	**1.3**	**1.3**	**1.6**	**1.4**	**1.8**	**1.3**	**1.4**	**2.0**	**1.0**

Correlation levels 1, 2 or 3 are as defined below:
1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High) No correlation: "-"

INDEX

Ex. No. : 1 **Date : 23-01-2025**

Register No. : 230701385 **Name : S VISHWAK**

Study of Azure DevOps

AIM:

To study how to create an agile project in the 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:

The study has been successfully completed.

Ex. No. : 2 **Date : 30-01-2025**

Register No. : 230701385 **Name : S VISHWAK**

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

Hospital Management System:

The increasing demand for efficient healthcare services highlights significant limitations in traditional hospital management practices. These outdated methods often result in data inconsistency, scheduling errors, delays in patient care, and administrative inefficiencies. Furthermore, the lack of a secure and centralized system compromises data privacy and compliance with healthcare regulations. In response to these challenges, the development of a Hospital Management System is essential. The proposed system aims to digitize and streamline core hospital operations by enabling patients to register, schedule appointments, access medical records, and receive timely reminders. Simultaneously, it allows doctors to view and update patient information in real-time, ensuring faster and more informed decision-making. With a focus on data security, regulatory compliance, and user-friendly design, the system seeks to improve hospital efficiency, enhance patient experience, and support better healthcare outcomes.

RESULT:

The problem statement has been successfully written.

Ex. No. : 3 **Date : 06-02-2025**

Register No. : 230701385 **Name : S VISHWAK**

AGILE PLANNING

AIM:

To prepare an Agile Plan.

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 ultimate 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 Agile plan has been successfully completed.

Ex. No. : 4 **Date : 13-02-2025**

Register No. : 230701385 **Name : S VISHWAK**

CREATE USER STORIES

AIM:

To create User Stories

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.

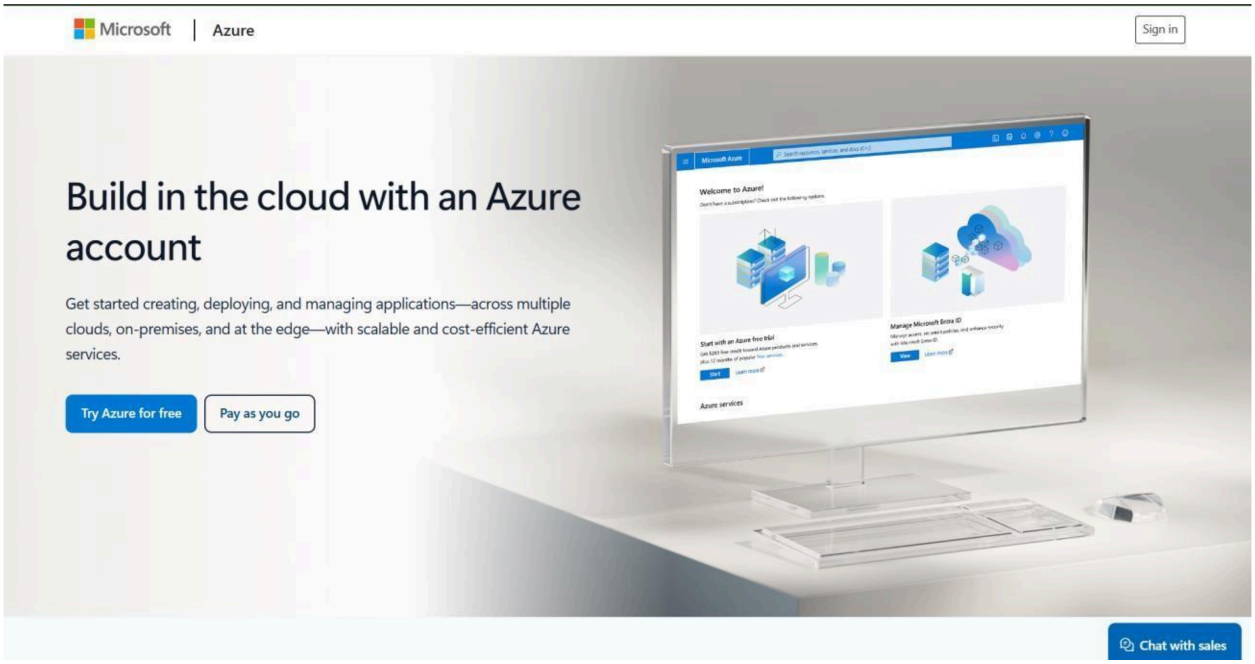
User story template

"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.

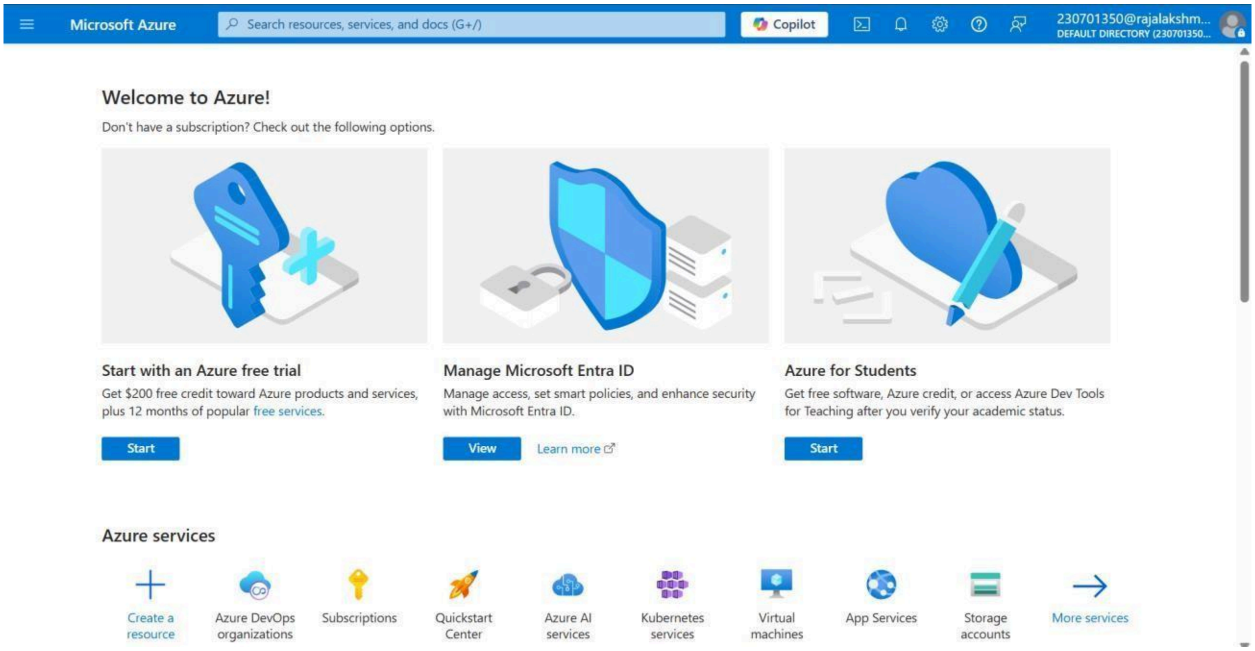
2. If you don’t have a Microsoft account, you can sign up for *https://signup.live.com/?lic=1*



Build in the cloud with an Azure account

Get started creating, deploying, and managing applications—across multiple clouds, on-premises, and at the edge—with scalable and cost-efficient Azure services.

3. Azure home page


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
230701350@rajalakshm...
DEFAULT DIRECTORY (230701350...

Welcome to Azure!

Don't have a subscription? Check out the following options.

Start with an Azure free trial

Get $200 free credit toward Azure products and services, plus 12 months of popular free services.

Manage Microsoft Entra ID

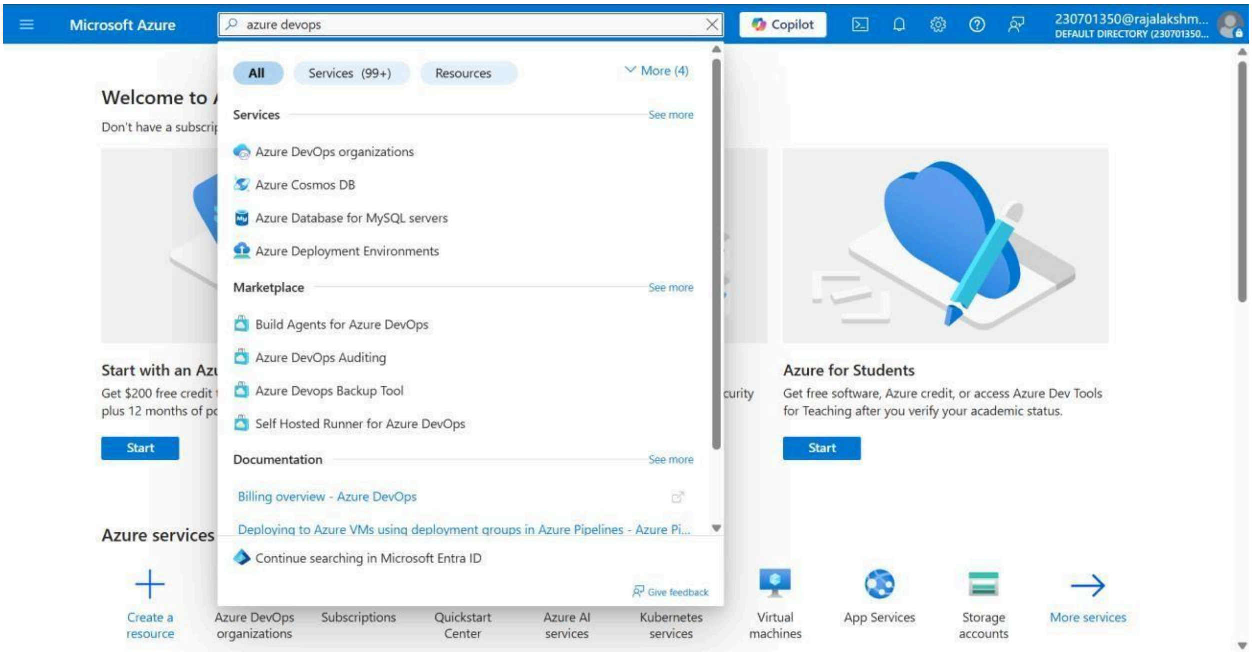
Manage access, set smart policies, and enhance security with Microsoft Entra ID.

Azure for Students

Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.

Azure services

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



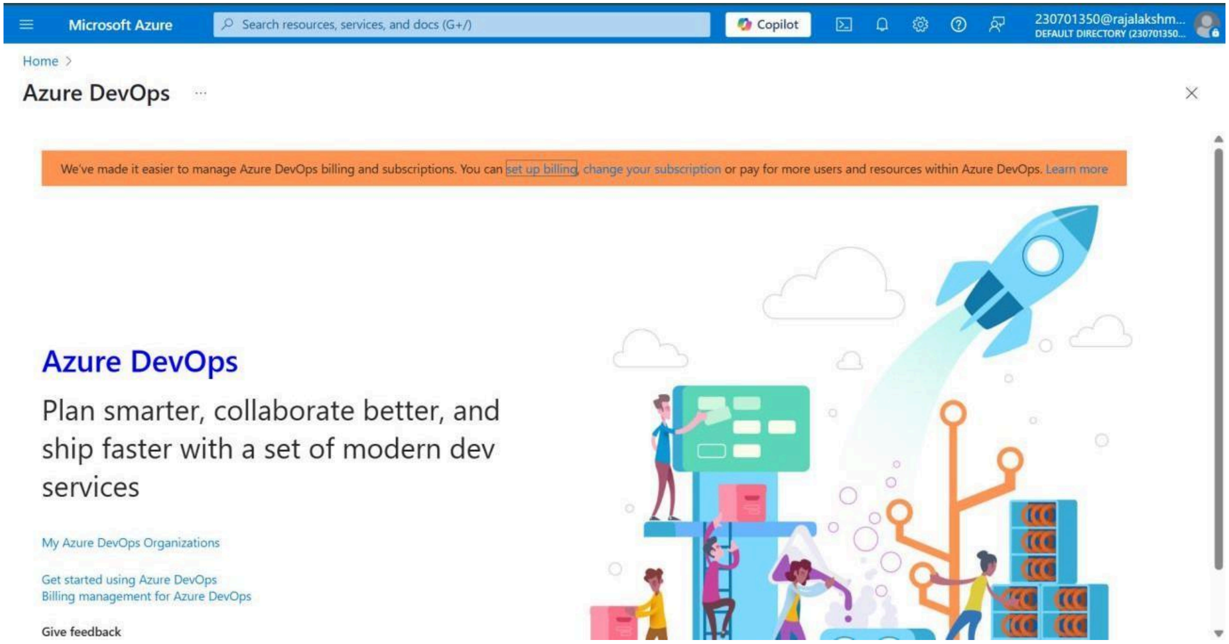
Don't have a subscrip

Get $200 free credit
plus 12 months of pc

curity

Get free software, Azure credit, or access Azure Dev Tools for Teaching after you verify your academic status.

5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home Page.


Microsoft Azure
Search resources, services, and docs (G+/)
Copilot
230701350@rajalakshm...
DEFAULT DIRECTORY (230701350...
Home >
Azure DevOps ...
My Azure DevOps Organizations
Get started using Azure DevOps
Billing management for Azure DevOps
Give feedback

We've made it easier to manage Azure DevOps billing and subscriptions. You can set up billing, change your subscription or pay for more users and resources within Azure DevOps. Learn more

Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services

6. Create the First Project in Your Organization

After the organization is set up, you’ll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.

i. On the organization’s **Home page**, click on the **New Project** button.

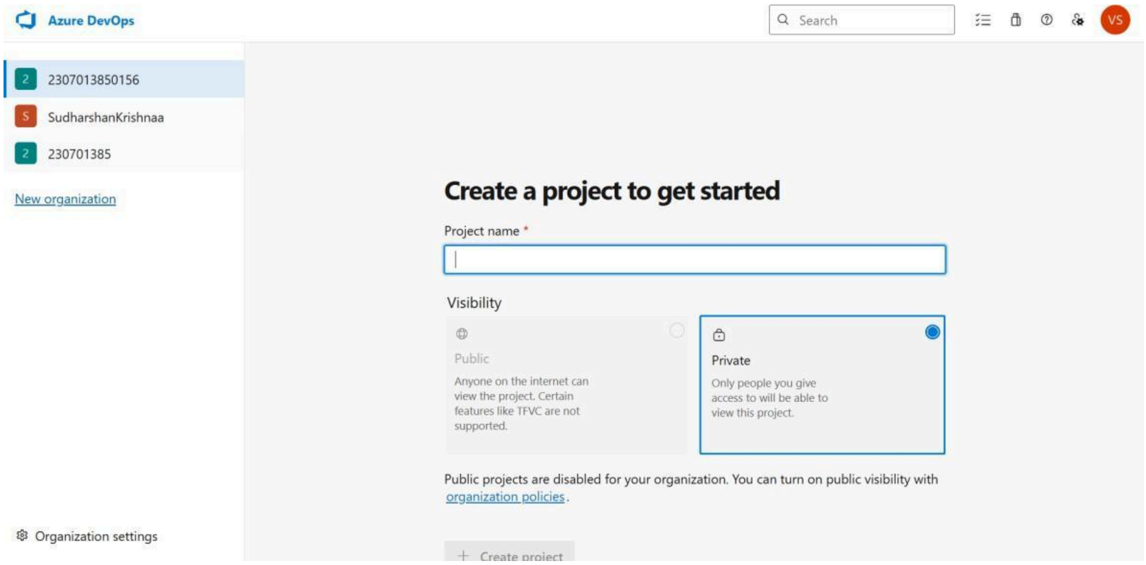
ii. Enter the project name, description, and visibility options:

○ **Name**: Choose a name for the project (e.g., LMS).

○ **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).

iii. Once you’ve filled out the details, click **Create.**



Create a project to get started

Project name *

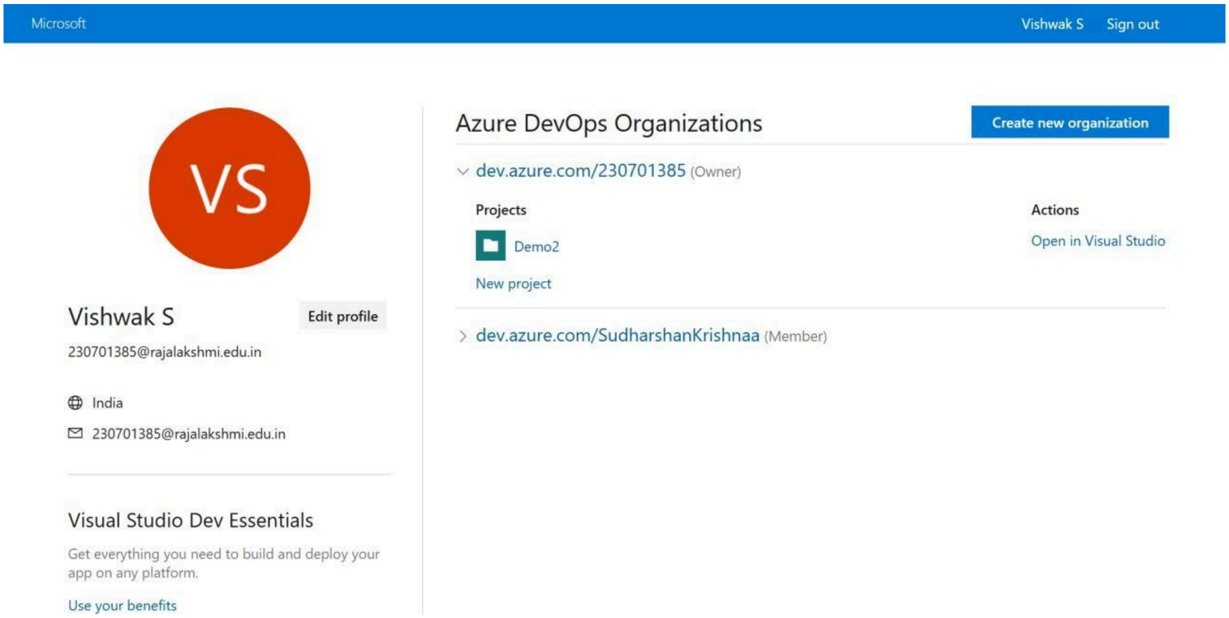
Visibility

Public
Anyone on the internet can view the project. Certain features like TFVC are not supported.

● Private
Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with organization policies.

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.



Vishwak S

230701385@rajalakshmi.edu.in

India

230701385@rajalakshmi.edu.in

Visual Studio Dev Essentials

Get everything you need to build and deploy your app on any platform.

Use your benefits

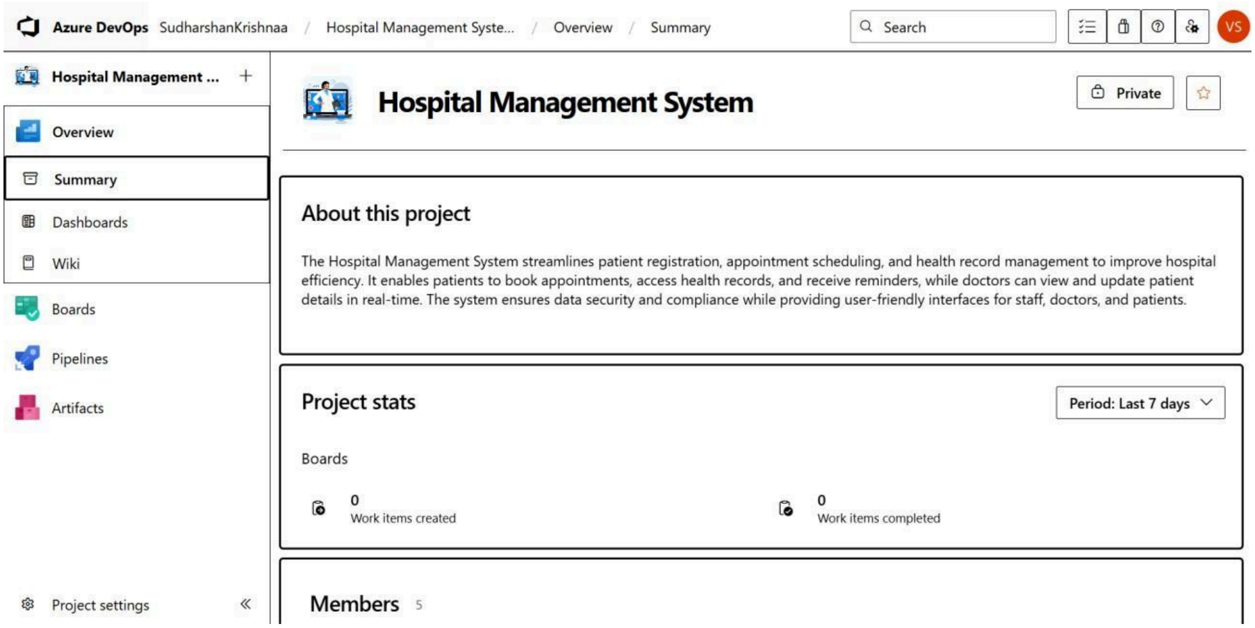
Azure DevOps Organizations

dev.azure.com/230701385 (Owner)

Projects	Actions
Demo2	Open in Visual Studio
New project	

dev.azure.com/SudharshanKrishnaa (Member)

8. Project dashboard



Hospital Management System

About this project

The Hospital Management System streamlines patient registration, appointment scheduling, and health record management to improve hospital efficiency. It enables patients to book appointments, access health records, and receive reminders, while doctors can view and update patient details in real-time. The system ensures data security and compliance while providing user-friendly interfaces for staff, doctors, and patients.

Project stats

Period: Last 7 days

Boards

0 Work items created

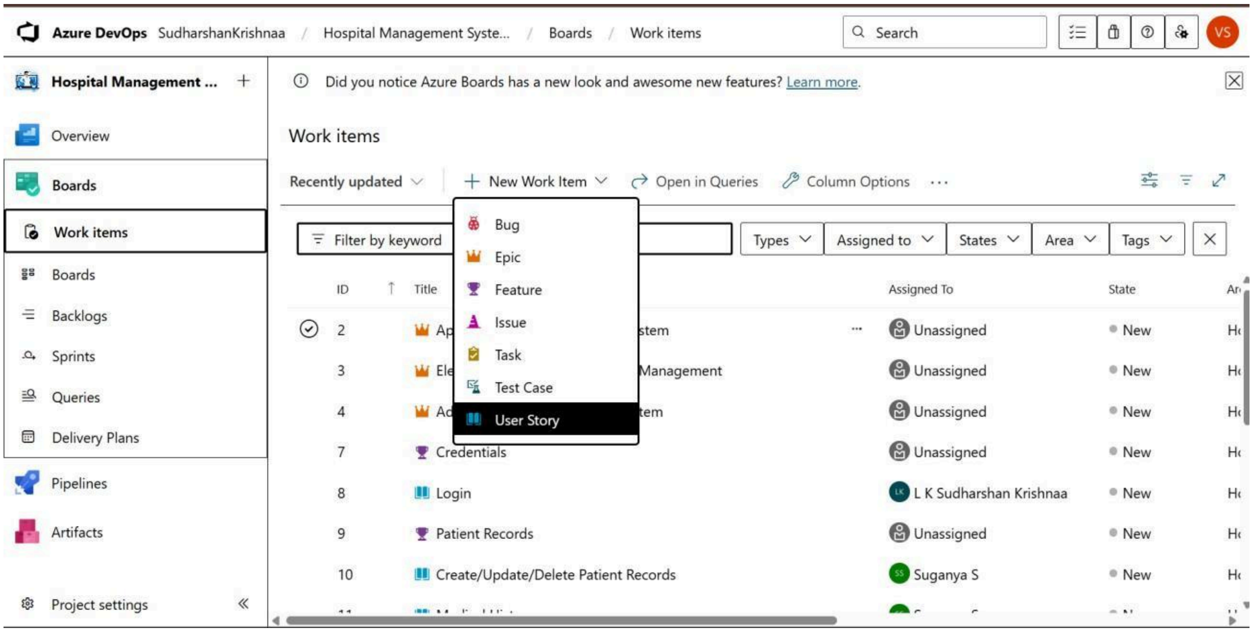
0 Work items completed

Members 5

9. To manage user stories

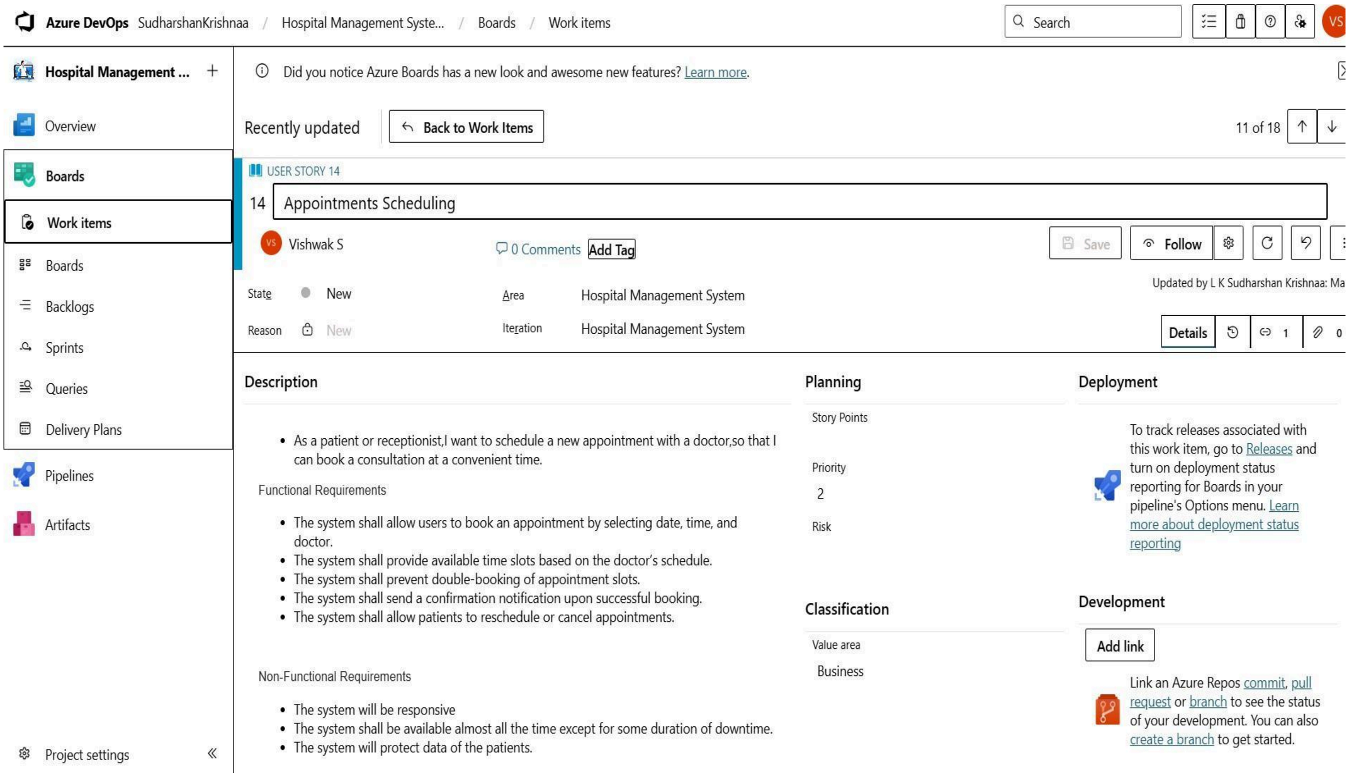
a. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards** page, where you can manage work items, backlogs, and sprints.

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.



ID	Title	Assigned To	State
2	Ap stem	Unassigned	New
3	Ele Management	Unassigned	New
4	Ad tem	Unassigned	New
7	Credentials	Unassigned	New
8	Login	L K Sudharshan Krishnaa	New
9	Patient Records	Unassigned	New
10	Create/Update/Delete Patient Records	Suganya S	New

10. Fill in User Story Details



USER STORY 14

14 Appointments Scheduling

Vishwak S

Updated by L K Sudharshan Krishnaa: Ma

State: New
Reason: New
Area: Hospital Management System
Iteration: Hospital Management System

Description

- As a patient or receptionist,I want to schedule a new appointment with a doctor,so that I can book a consultation at a convenient time.

Functional Requirements

- The system shall allow users to book an appointment by selecting date, time, and doctor.
- The system shall provide available time slots based on the doctor's schedule.
- The system shall prevent double-booking of appointment slots.
- The system shall send a confirmation notification upon successful booking.
- The system shall allow patients to reschedule or cancel appointments.

Non-Functional Requirements

- The system will be responsive
- The system shall be available almost all the time except for some duration of downtime.
- The system will protect data of the patients.

Planning

Story Points

Priority: 2

Risk

Classification

Value area: Business

Deployment

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Development

Add link

Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started.

My Assigned User Stories:

- As a patient or receptionist, I want to schedule a new appointment with a doctor, so that I can book a consultation at a convenient time
- As a patient, I can cancel or reschedule my appointment, so that I can manage my schedule and avoid missed visits.

RESULT:

The user story was written successfully.

Ex. No. : 5 **Date : 20-02-2025**

Register No. : 230701385 **Name : S VISHWAK**

SEQUENCE DIAGRAM

AIM:

To design a Sequence Diagram by using Mermaid.js

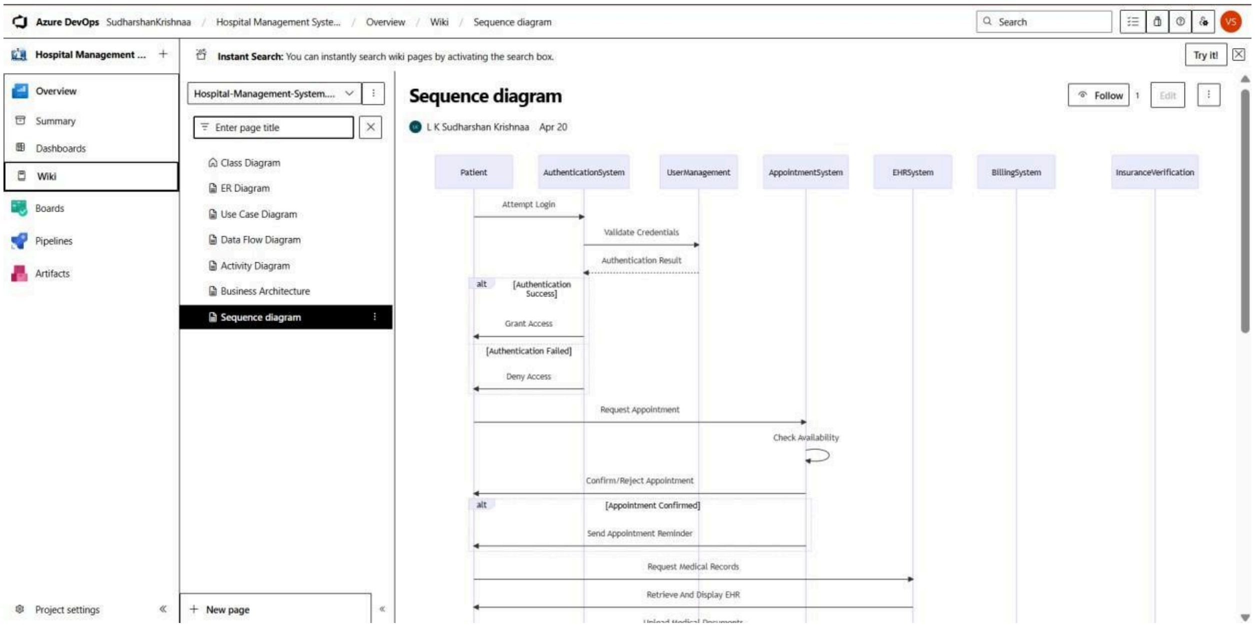
THEORY:

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 behaviour in a system.

PROCEDURE:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu



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

```
::: mermaid
sequenceDiagram
    participant Patient
    participant AuthenticationSystem
    participant UserManagement
    participant AppointmentSystem
    participant EHRSystem
    participant BillingSystem
    participant InsuranceVerification

    Patient->>AuthenticationSystem: Attempt Login
    AuthenticationSystem->>UserManagement: Validate Credentials
    UserManagement-->>AuthenticationSystem: Authentication
    Result alt Authentication Success
        AuthenticationSystem->>Patient: Grant Access
    else Authentication Failed
        AuthenticationSystem->>Patient: Deny Access
    end
    Patient->>AppointmentSystem: Request Appointment
    AppointmentSystem->>AppointmentSystem: Check Availability
    AppointmentSystem->>Patient: Confirm/Reject Appointment
    alt Appointment Confirmed
```

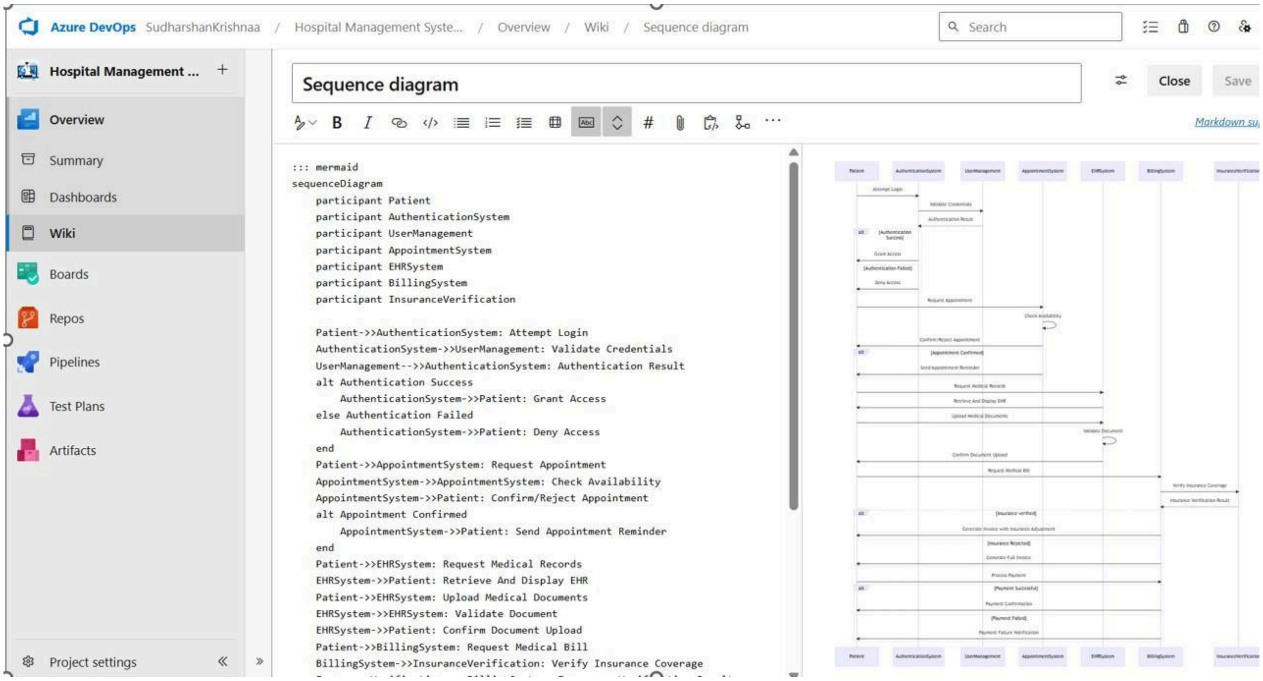
```
        AppointmentSystem->>Patient: Send Appointment Reminder
    end
    Patient->>EHRSystem: Request Medical Records
    EHRSystem->>Patient: Retrieve And Display EHR
    Patient->>EHRSystem: Upload Medical Documents
    EHRSystem->>EHRSystem: Validate Document
    EHRSystem->>Patient: Confirm Document Upload
    Patient->>BillingSystem: Request Medical Bill
    BillingSystem->>InsuranceVerification: Verify Insurance Coverage
    InsuranceVerification-->>BillingSystem: Insurance Verification Result
    alt Insurance verified
        BillingSystem->>Patient: Generate Invoice with Insurance Adjustment
    else Insurance Rejected
        BillingSystem->>Patient: Generate Full Invoice
    end
    Patient->>BillingSystem: Process Payment
    alt Payment Successful
        BillingSystem->>Patient: Payment Confirmation
    else Payment Failed
        BillingSystem->>Patient: Payment Failure Notification
    end
:::
```

EXPLANATION:

- participant defines the entities involved:
    - Patient, AuthenticationSystem, UserManagement, AppointmentSystem, EHRSystem, BillingSystem, and InsuranceVerification are the participants in the system.
- ->> represents a direct message from one participant to another:
    - Patient->>AuthenticationSystem: Attempt Login means the patient sends a login attempt request to the authentication system.
- -->> represents a response message:
    - UserManagement-->>AuthenticationSystem: Authentication Result means the user management system sends back the result of validation to the authentication system.
- alt is used for conditional flows (if/else logic):
    - alt Authentication Success - if the authentication is successful, access is granted to the patient.
    - else Authentication Failed - if authentication fails, access is denied.
    - Similarly used for appointment confirmation, insurance verification, and payment confirmation.
- loop can be used for repeated actions.

- Arrows explained:
    - -> Solid line without arrow
    - --> Dotted line without arrow
    - ->> Solid line with arrowhead: Sending a request/message.
    - -->> Dotted line with arrowhead: Sending a response.
    - <<->> Solid line with bidirectional arrowheads (v11.0.0+)
    - <<-->> Dotted line with bidirectional arrowheads (v11.0.0+):
    - -x Solid line with a cross at the end
    - --x Dotted line with a cross at the end

    - -) Solid line with an open arrow at the end (async)
    - --) Dotted line with a open arrow at the end (async)
- Special Notes:
    - After ->>, a participant is activated (ready to process).
    - After -->>, a participant is deactivated (finished the current process).



Sequence diagram

```
::: mermaid
sequenceDiagram
    participant Patient
    participant AuthenticationSystem
    participant UserManagement
    participant AppointmentSystem
    participant EHRSystem
    participant BillingSystem
    participant InsuranceVerification

    Patient->>AuthenticationSystem: Attempt Login
    AuthenticationSystem->>UserManagement: Validate Credentials
    UserManagement-->>AuthenticationSystem: Authentication Result
    alt Authentication Success
        AuthenticationSystem->>Patient: Grant Access
    else Authentication Failed
        AuthenticationSystem->>Patient: Deny Access
    end
    Patient->>AppointmentSystem: Request Appointment
    AppointmentSystem->>AppointmentSystem: Check Availability
    AppointmentSystem->>Patient: Confirm/Reject Appointment
    alt Appointment Confirmed
        AppointmentSystem->>Patient: Send Appointment Reminder
    end
    Patient->>EHRSystem: Request Medical Records
    EHRSystem->>Patient: Retrieve And Display EHR
    Patient->>EHRSystem: Upload Medical Documents
    EHRSystem->>EHRSystem: Validate Document
    EHRSystem->>Patient: Confirm Document Upload
    Patient->>BillingSystem: Request Medical Bill
    BillingSystem->>InsuranceVerification: Verify Insurance Coverage
```

RESULT:

The sequence diagram has been drawn successfully.

Ex. No. : 6 **Date : 27-02-2025**

Register No. : 230701385 **Name : S VISHWAK**

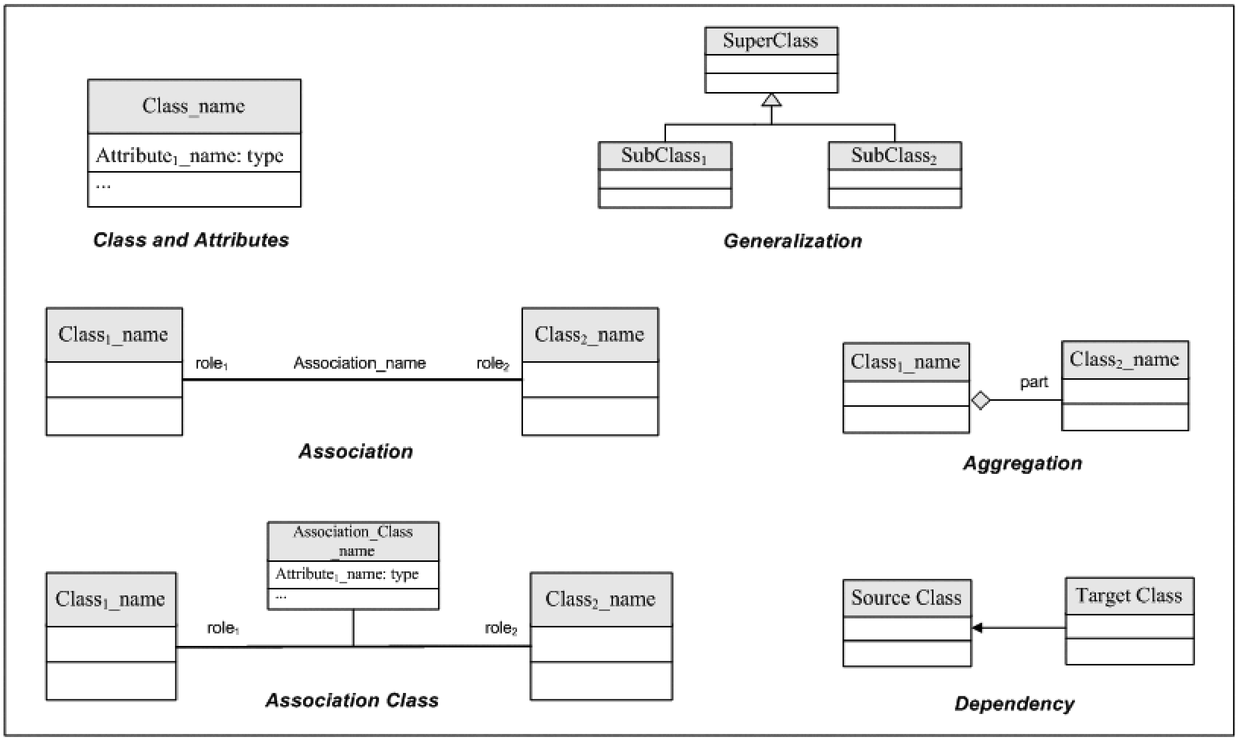
CLASS DIAGRAM

AIM:

To design a Class Diagram for your project.

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.

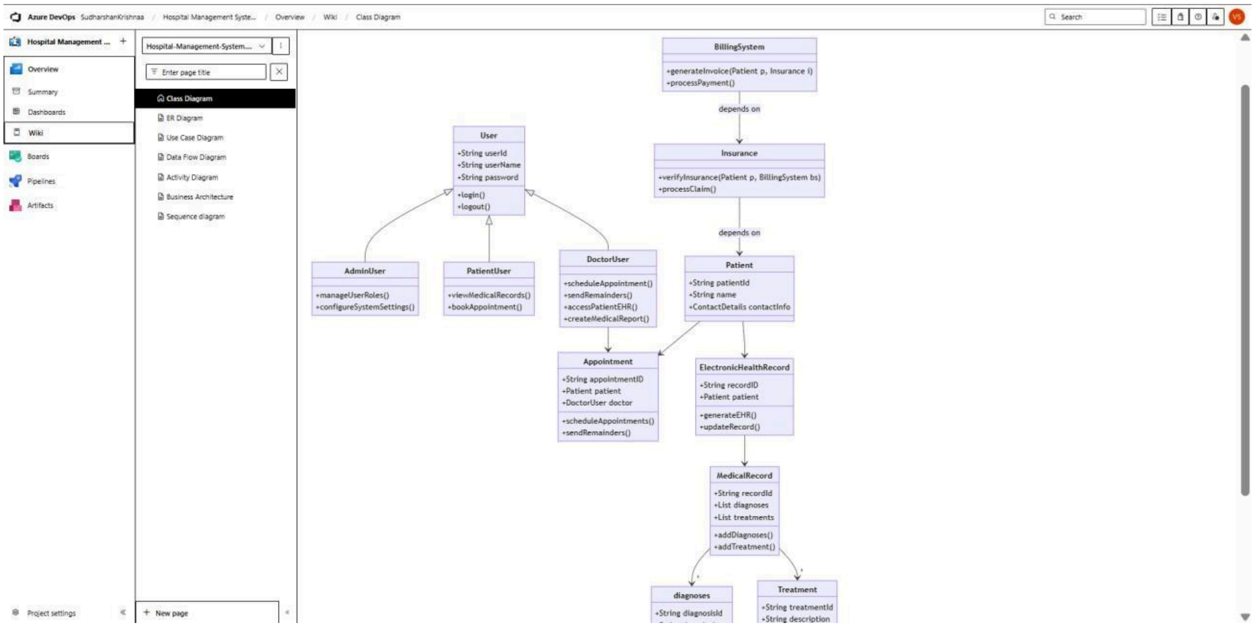


Notations in Class Diagram.

PROCEDURE:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu



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

```
:::mermaid

classDiagram

%% Base Classes

class User {

  +String userId

  +String userName

  +String password

  +login()
  +logout()}
class Patient {
  +String patientId
  +String name
  +ContactDetails contactInfo
}
class Appointment {
  +String appointmentID
  +Patient patient
  +DoctorUser doctor
  +scheduleAppointments()
  +sendRemainders()}
class ElectronicHealthRecord {
  +String recordID
  +Patient patient
  +generateEHR()
  +updateRecord()}
class MedicalRecord {
  +String recordId
  +List diagnoses
  +List treatments
  +addDiagnoses()
  +addTreatment()}
class diagnoses {
  +String diagnosisId
  +String description
  +Date diagnosisDate
}
class Treatment {
  +String treatmentId
  +String description
  +Date startDate
  +Date endDate
}
class Insurance {
  +verifyInsurance(Patient p, BillingSystem bs)
  +processClaim()
}
class BillingSystem {
  +generateInvoice(Patient p, Insurance i)
  +processPayment()
}
%% Inheritance
User <|--
AdminUser
```

```
User <|--
PatientUser User
<|-- DoctorUser
%% Admin
class AdminUser {
  +manageUserRoles()
  +configureSystemSettings()
}
%% Patient
class PatientUser {
  +viewMedicalRecords()
  +bookAppointment()
}
%% Doctor
class DoctorUser {
  +scheduleAppointment()
  +sendRemainders()
  +accessPatientEHR()
  +createMedicalReport()
}
%% Associations
Patient --> Appointment
DoctorUser --> Appointment
```

Patient --> ElectronicHealthRecord

ElectronicHealthRecord --> MedicalRecord

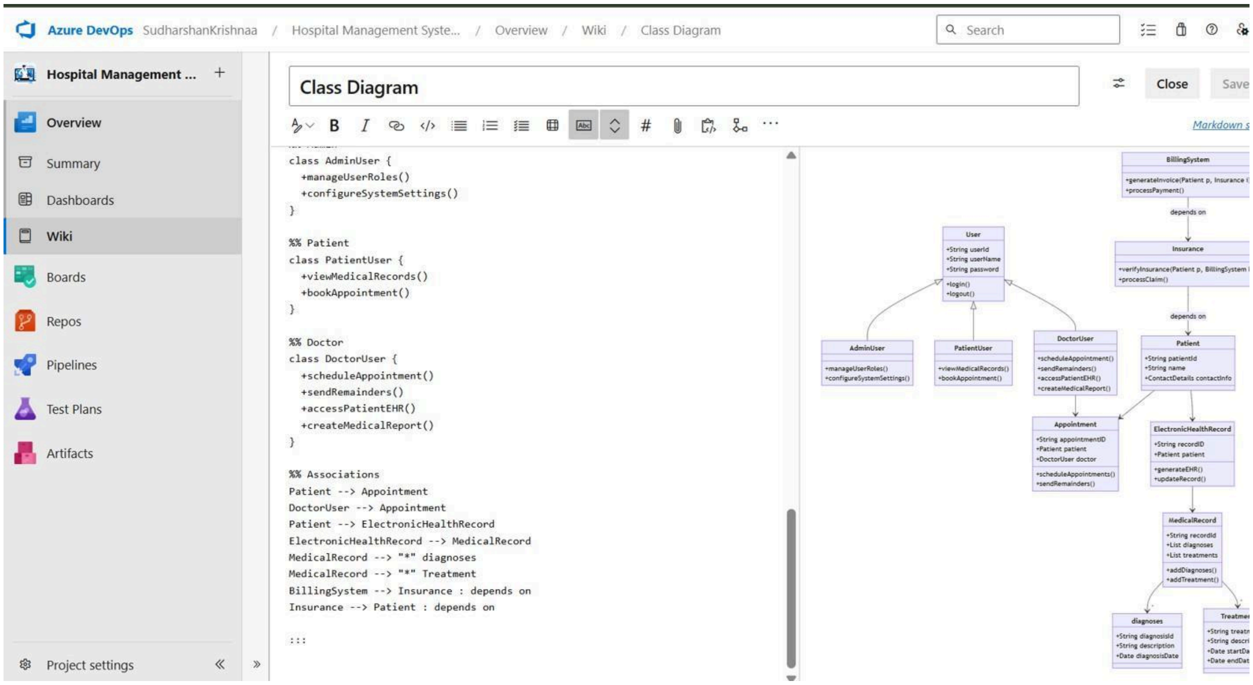
MedicalRecord --> "*" diagnoses

MedicalRecord --> "*" Treatment

BillingSystem --> Insurance : depends on

Insurance --> Patient : depends on

:::



```
class AdminUser {
  +manageUserRoles()
  +configureSystemSettings()
}

%% Patient
class PatientUser {
  +viewMedicalRecords()
  +bookAppointment()
}

%% Doctor
class DoctorUser {
  +scheduleAppointment()
  +sendRemainders()
  +accessPatientEHR()
  +createMedicalReport()
}

%% Associations
Patient --> Appointment
DoctorUser --> Appointment
Patient --> ElectronicHealthRecord
ElectronicHealthRecord --> MedicalRecord
MedicalRecord --> "*" diagnoses
MedicalRecord --> "*" Treatment
BillingSystem --> Insurance : depends on
Insurance --> Patient : depends on

:::
```

RESULT:

The Class diagram has been drawn successfully.

Ex. No. : 7 **Date : 06-03-2025**

Register No. : 230701385 **Name : S VISHWAK**

USE CASE DIAGRAM

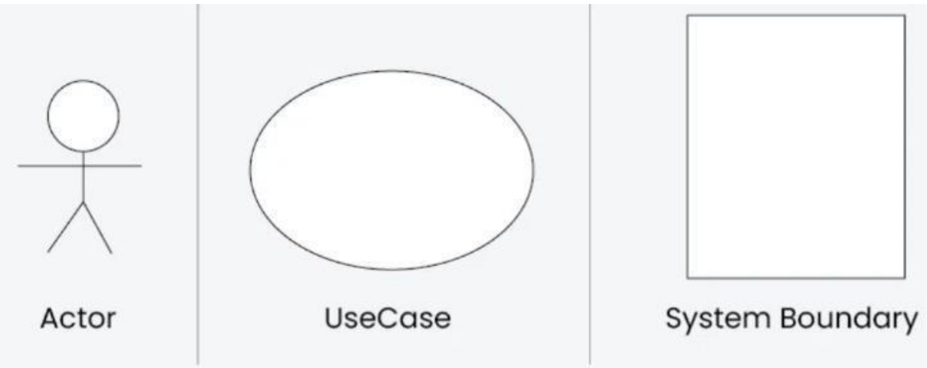
AIM:

Steps to draw the Use Case Diagram using draw.io

THEORY:

● UCD 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 .drawio 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 CaseDiagram](attachments/use_case_diagram.png) Option

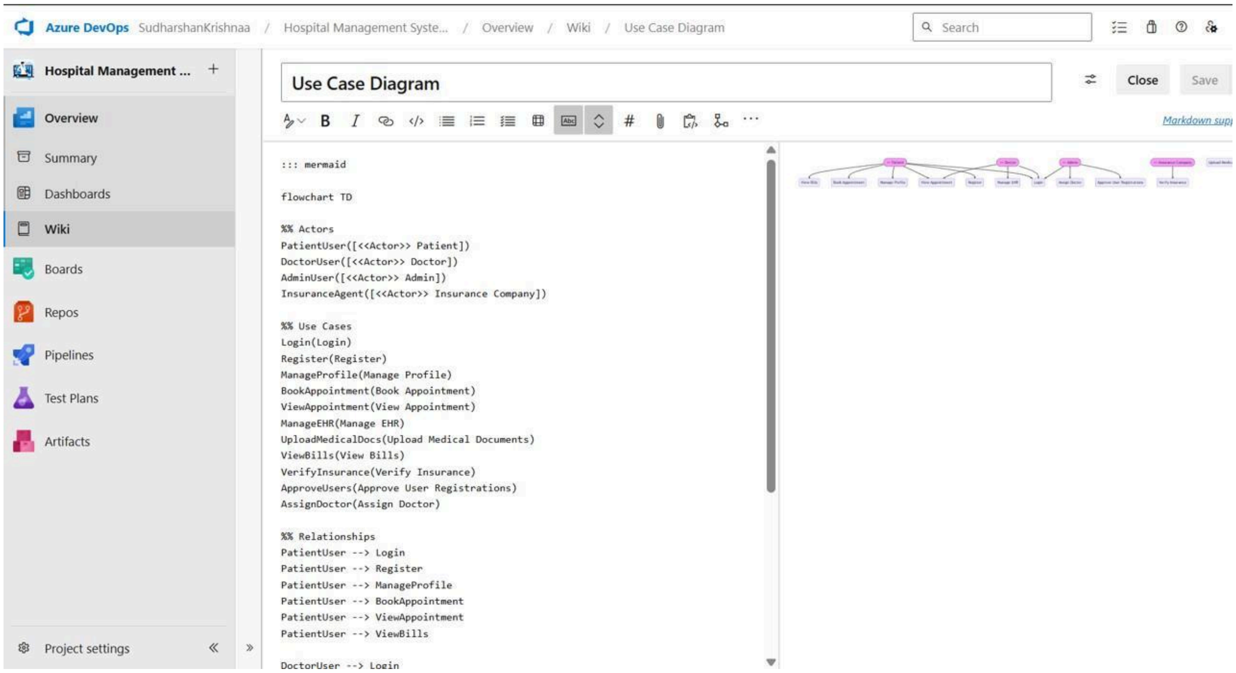
2: Attach to Work Items in Azure Boards

- 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.

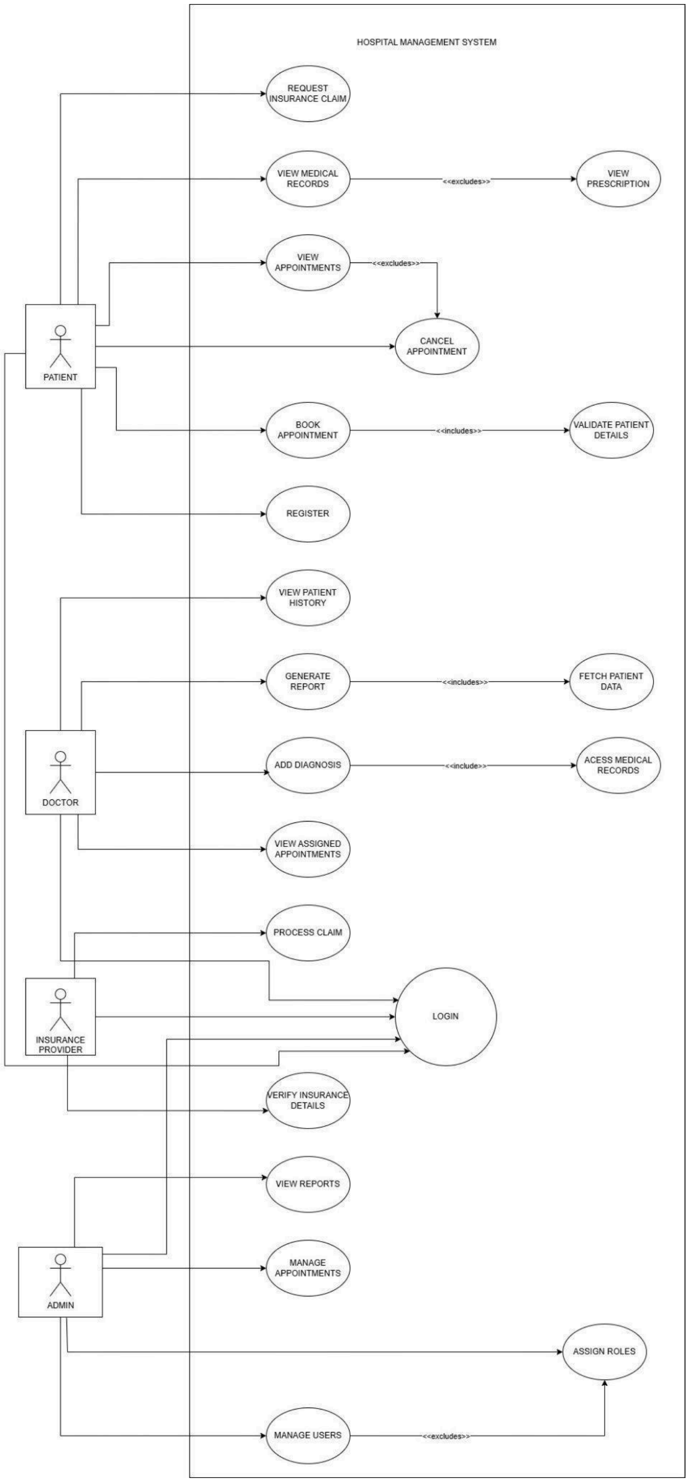
CODE:

```
::: mermaid
flowchart TD

%% Actors
PatientUser([<<Actor>> Patient])
DoctorUser([<<Actor>> Doctor])
AdminUser([<<Actor>> Admin])
InsuranceAgent([<<Actor>> Insurance Company])

%% Use Cases
Login(Login)
Register(Register)
ManageProfile(Manage Profile)
BookAppointment(Book Appointment)
ViewAppointment(View Appointment)
ManageEHR(Manage EHR)
UploadMedicalDocs(Upload Medical Documents)
ViewBills(View Bills)
VerifyInsurance(Verify Insurance)
ApproveUsers(Approve User Registrations)
AssignDoctor(Assign Doctor)

%% Relationships
PatientUser --> Login
PatientUser --> Register
PatientUser --> ManageProfile
PatientUser --> BookAppointment
PatientUser --> ViewAppointment
PatientUser --> ViewBills
DoctorUser --> Login
DoctorUser --> ManageEHR
DoctorUser --> ViewAppointment
AdminUser --> Login
AdminUser --> ApproveUsers
AdminUser --> AssignDoctor

InsuranceAgent --> VerifyInsurance

style PatientUser fill:#f9f,stroke:#333,stroke-width:1px
style DoctorUser fill:#f9f,stroke:#333,stroke-width:1px
style AdminUser fill:#f9f,stroke:#333,stroke-width:1px
style InsuranceAgent fill:#f9f,stroke:#333,stroke-width:1px
:::
```



Use Case Diagram

```
::: mermaid

flowchart TD

%% Actors
PatientUser([<<Actor>> Patient])
DoctorUser([<<Actor>> Doctor])
AdminUser([<<Actor>> Admin])
InsuranceAgent([<<Actor>> Insurance Company])

%% Use Cases
Login(Login)
Register(Register)
ManageProfile(Manage Profile)
BookAppointment(Book Appointment)
ViewAppointment(View Appointment)
ManageEHR(Manage EHR)
UploadMedicalDocs(Upload Medical Documents)
ViewBills(View Bills)
VerifyInsurance(Verify Insurance)
ApproveUsers(Approve User Registrations)
AssignDoctor(Assign Doctor)

%% Relationships
PatientUser --> Login
PatientUser --> Register
PatientUser --> ManageProfile
PatientUser --> BookAppointment
PatientUser --> ViewAppointment
PatientUser --> ViewBills

DoctorUser --> Login
```



RESULT:

Hence, the use case diagram has been successfully created.

Ex. No. : 8 **Date : 27-03-2025**

Register No. : 230701385 **Name : S VISHWAK**

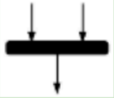
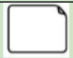
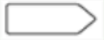
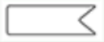
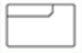
ACTIVITY DIAGRAM

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 o 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

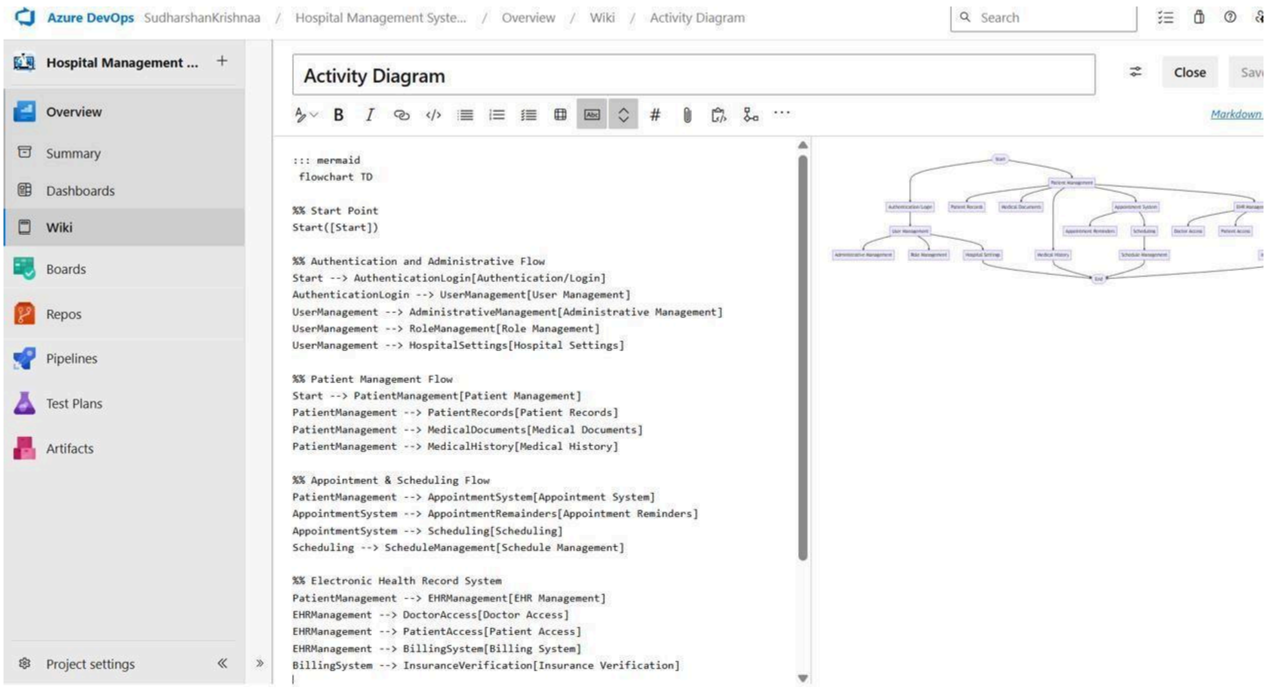
PROCEDURE:

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

CODE:

```
::: mermaid

 flowchart TD

%% Start Point

Start([Start])


%% Authentication and Administrative Flow

Start --> AuthenticationLogin[Authentication/Login]

AuthenticationLogin --> UserManagement[User Management]

UserManagement --> AdministrativeManagement[Administrative Management]

UserManagement --> RoleManagement[Role Management]

UserManagement --> HospitalSettings[Hospital Settings]


%% Patient Management Flow

Start --> PatientManagement[Patient Management]

PatientManagement --> PatientRecords[Patient Records]

PatientManagement --> MedicalDocuments[Medical Documents]

PatientManagement --> MedicalHistory[Medical History]
```

```
%% Appointment & Scheduling Flow
PatientManagement --> AppointmentSystem[Appointment System]
AppointmentSystem --> AppointmentRemainders[Appointment Reminders]
AppointmentSystem --> Scheduling[Scheduling]
Scheduling --> ScheduleManagement[Schedule Management]

%% Electronic Health Record System
PatientManagement --> EHRManagement[EHR Management]
EHRManagement --> DoctorAccess[Doctor Access]
EHRManagement --> PatientAccess[Patient Access]
EHRManagement --> BillingSystem[Billing System]
BillingSystem --> InsuranceVerification[Insurance Verification]

%% Ending
InsuranceVerification --> End([End])
ScheduleManagement --> End
MedicalHistory --> End
HospitalSettings --> End
:::
```



```
::: mermaid
 flowchart TD

%% Start Point
Start([Start])

%% Authentication and Administrative Flow
Start --> AuthenticationLogin[Authentication/Login]
AuthenticationLogin --> UserManagement[User Management]
UserManagement --> AdministrativeManagement[Administrative Management]
UserManagement --> RoleManagement[Role Management]
UserManagement --> HospitalSettings[Hospital Settings]

%% Patient Management Flow
Start --> PatientManagement[Patient Management]
PatientManagement --> PatientRecords[Patient Records]
PatientManagement --> MedicalDocuments[Medical Documents]
PatientManagement --> MedicalHistory[Medical History]

%% Appointment & Scheduling Flow
PatientManagement --> AppointmentSystem[Appointment System]
AppointmentSystem --> AppointmentRemainders[Appointment Reminders]
AppointmentSystem --> Scheduling[Scheduling]
Scheduling --> ScheduleManagement[Schedule Management]

%% Electronic Health Record System
PatientManagement --> EHRManagement[EHR Management]
EHRManagement --> DoctorAccess[Doctor Access]
EHRManagement --> PatientAccess[Patient Access]
EHRManagement --> BillingSystem[Billing System]
BillingSystem --> InsuranceVerification[Insurance Verification]
```

RESULT:

Hence, the Activity diagram has been successfully created.

Ex. No. : 9 **Date : 03-04-2025**

Register No. : 230701385 **Name : S VISHWAK**

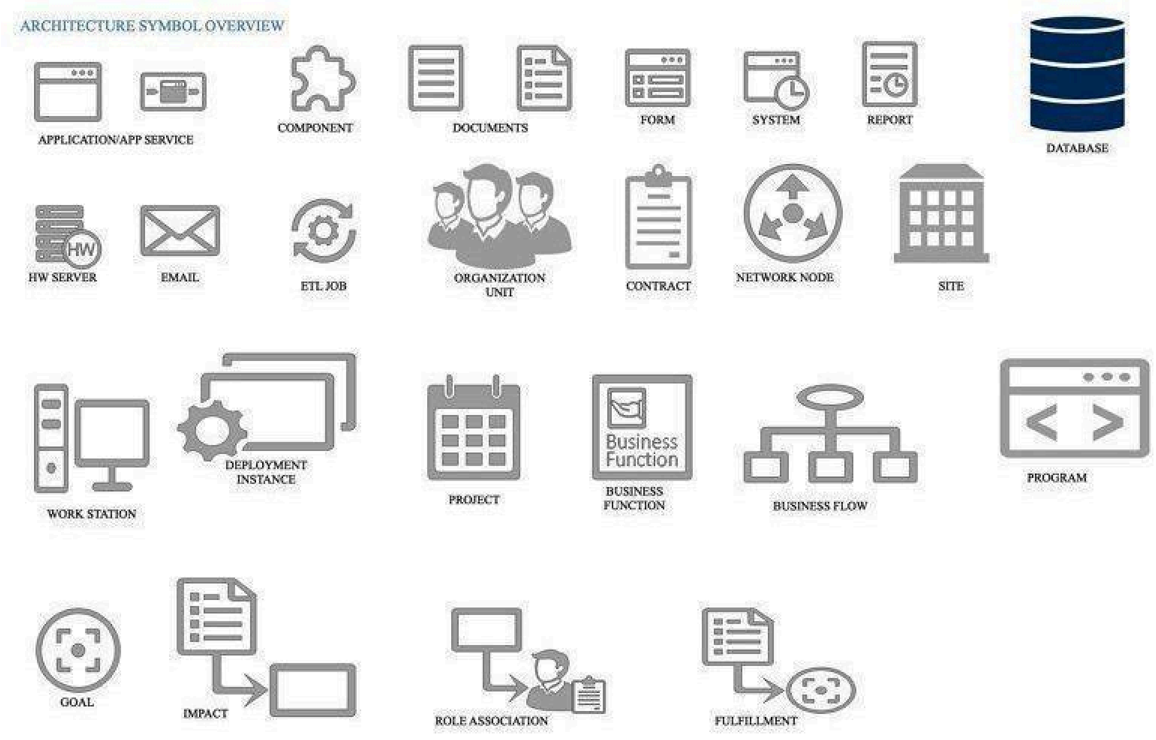
ARCHITECTURE DIAGRAM

AIM:

To draw an architecture diagram for your project or system.

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.



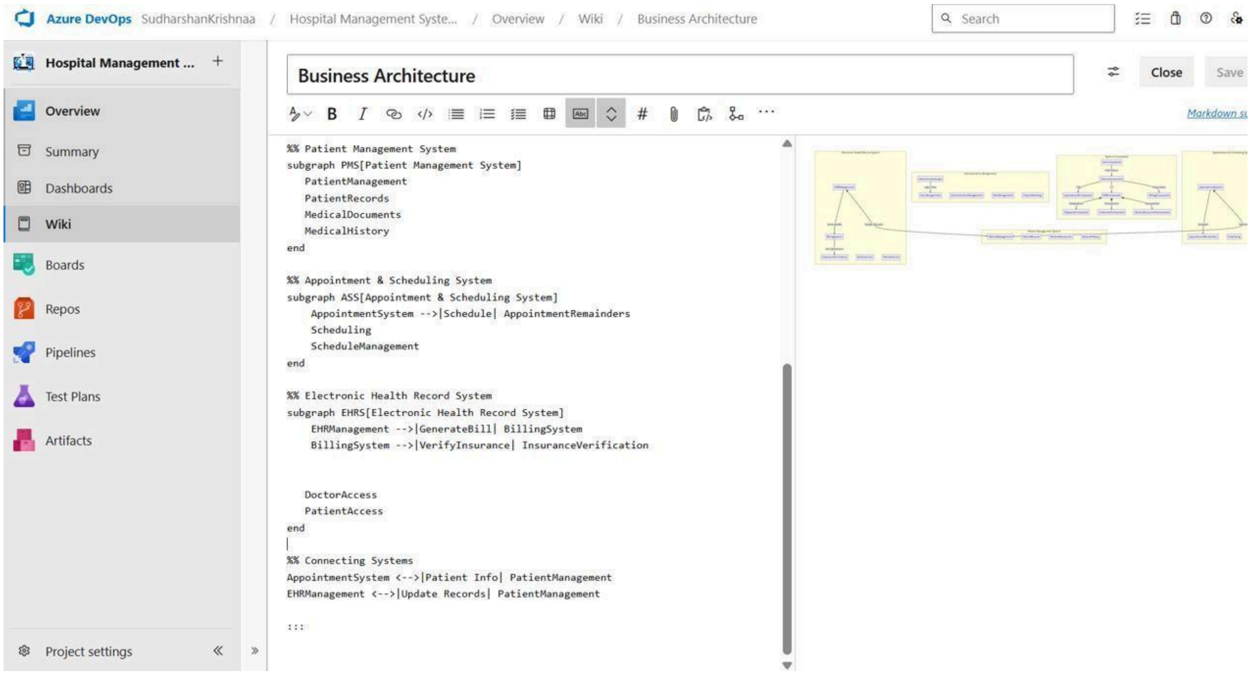
PROCEDURE:

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

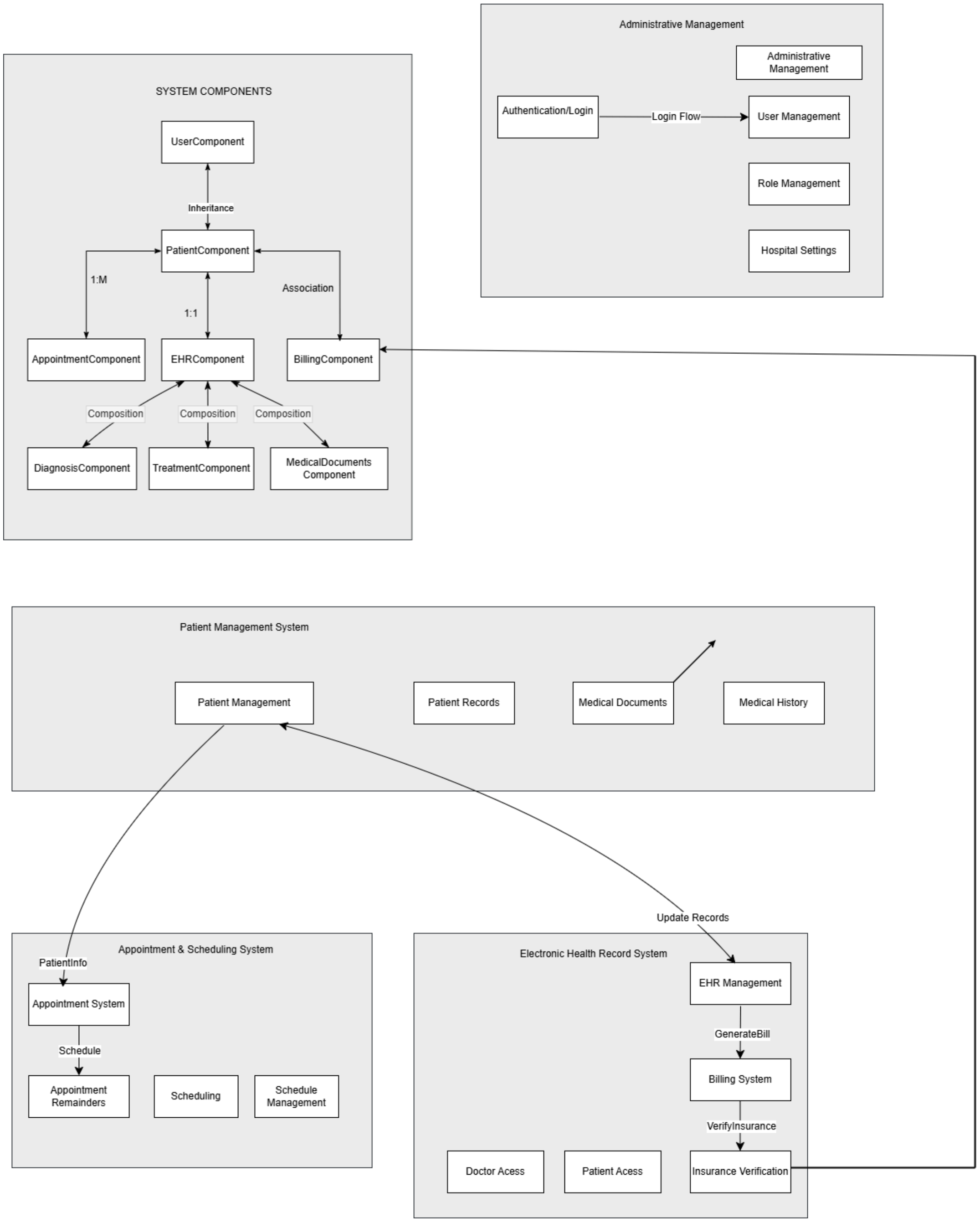
CODE:

```
::: mermaid
 flowchart TD

%% System Components
subgraph SC[System Components]
   UserComponent -->|Inheritance| PatientComponent
   PatientComponent -->|1:M| AppointmentComponent
   PatientComponent -->|1:1| EHRComponent
   PatientComponent -->|Association| BillingComponent

   EHRComponent <--> |Composition|DiagnosisComponent
   EHRComponent <--> |Composition|TreatmentComponent
   EHRComponent <--> |Composition|MedicalDocumentsComponent
end

%% Administrative Management
subgraph AM[Administrative Management]
   AuthenticationLogin -->|Login Flow| UserManagement
   AdministrativeManagement
   RoleManagement
   HospitalSettings
end
```

```
%% Patient Management System
subgraph PMS[Patient Management System]
  PatientManagement
  PatientRecords
  MedicalDocuments
  MedicalHistory
end

%% Appointment & Scheduling System
subgraph ASS[Appointment & Scheduling System]
   AppointmentSystem -->|Schedule| AppointmentRemainders
   Scheduling
   ScheduleManagement
end

%% Electronic Health Record System
subgraph EHRS[Electronic Health Record System]
   EHRManagement -->|GenerateBill| BillingSystem
   BillingSystem -->|VerifyInsurance| InsuranceVerification


  DoctorAccess
  PatientAccess
end

%% Connecting Systems
AppointmentSystem <-->|Patient Info| PatientManagement
EHRManagement <-->|Update Records| PatientManagement

:::
```



```
%% Patient Management System
subgraph PMS[Patient Management System]
   PatientManagement
   PatientRecords
   MedicalDocuments
   MedicalHistory
end

%% Appointment & Scheduling System
subgraph ASS[Appointment & Scheduling System]
    AppointmentSystem -->|Schedule| AppointmentRemainders
    Scheduling
    ScheduleManagement
end

%% Electronic Health Record System
subgraph EHRS[Electronic Health Record System]
    EHRManagement -->|GenerateBill| BillingSystem
    BillingSystem -->|VerifyInsurance| InsuranceVerification


   DoctorAccess
   PatientAccess
end

%% Connecting Systems
AppointmentSystem <-->|Patient Info| PatientManagement
EHRManagement <-->|Update Records| PatientManagement

:::
```



RESULT:

The architecture diagram was designed successfully.

Ex. No. : 10 **Date : 17-04-2025**

Register No. : 230701385 **Name : S VISHWAK**

USER INTERFACE

AIM:

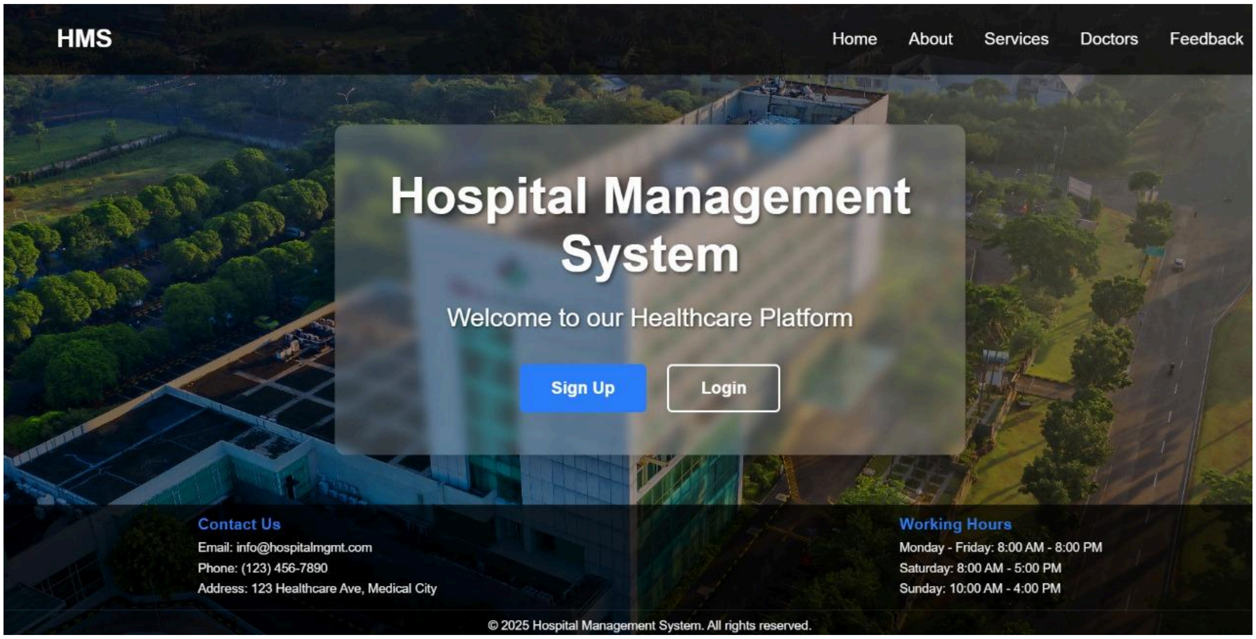
To design a User Interface for your project or system.

THEORY:

- UCD 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**

OUTPUT:



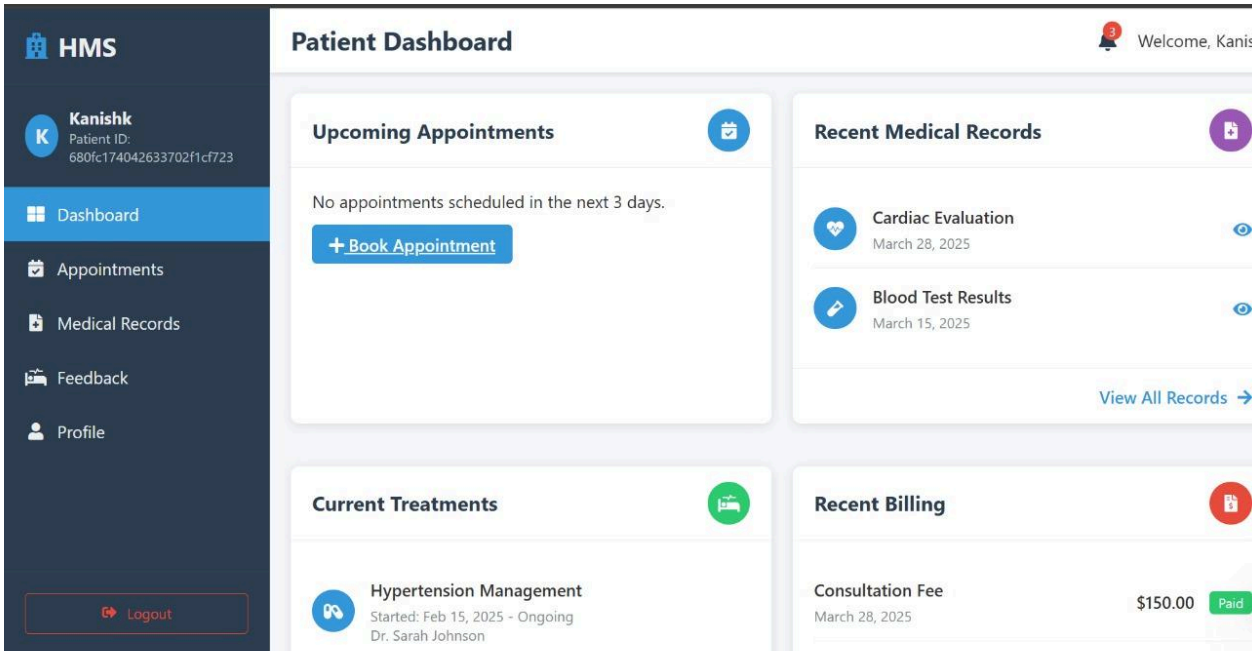
Hospital Management System

Welcome to our Healthcare Platform

Contact Us
Email: info@hospitalmgmt.com
Phone: (123) 456-7890
Address: 123 Healthcare Ave, Medical City

Working Hours
Monday - Friday: 8:00 AM - 8:00 PM
Saturday: 8:00 AM - 5:00 PM
Sunday: 10:00 AM - 4:00 PM

© 2025 Hospital Management System. All rights reserved.



Kanishk
Patient ID:
680fc174042633702f1cf723

Patient Dashboard

Welcome, Kanis

Upcoming Appointments

No appointments scheduled in the next 3 days.

Recent Medical Records

Cardiac Evaluation
March 28, 2025

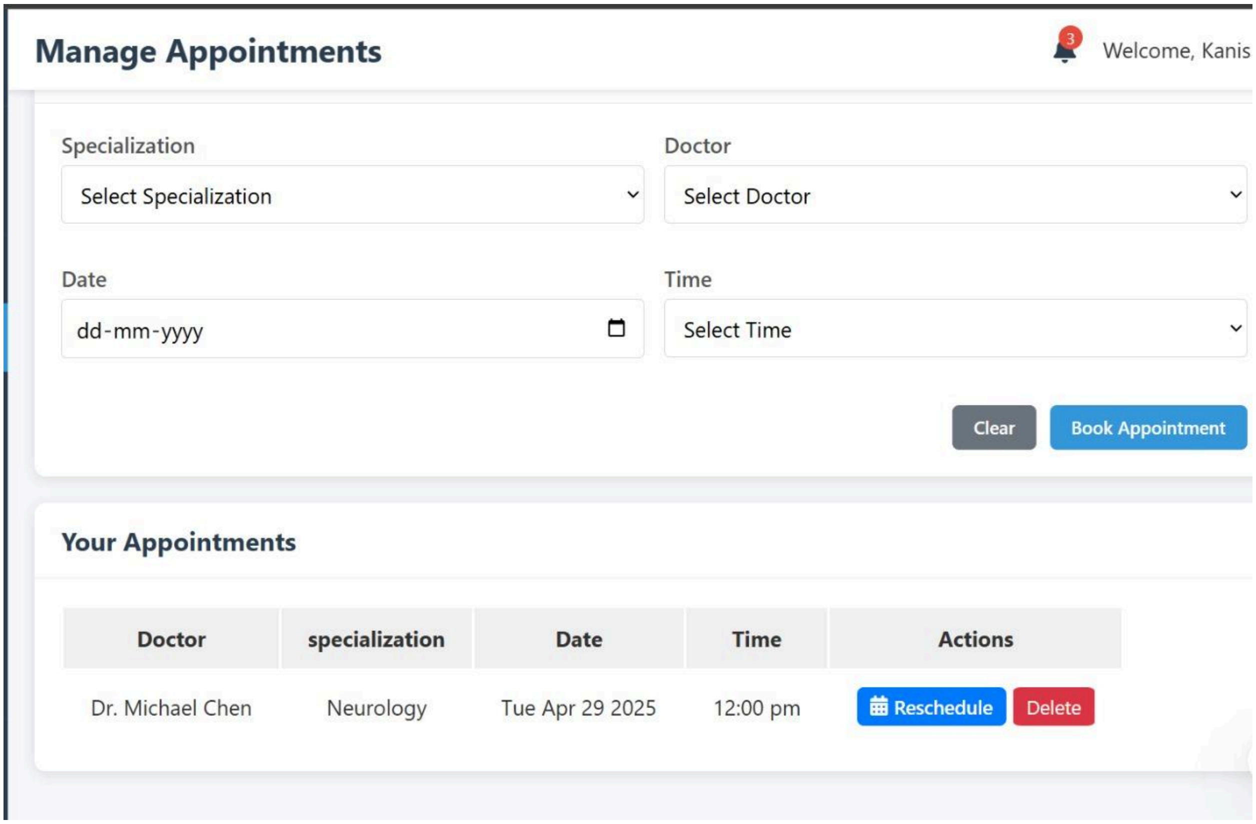
Blood Test Results
March 15, 2025

Current Treatments

Hypertension Management
Started: Feb 15, 2025 - Ongoing
Dr. Sarah Johnson

Recent Billing

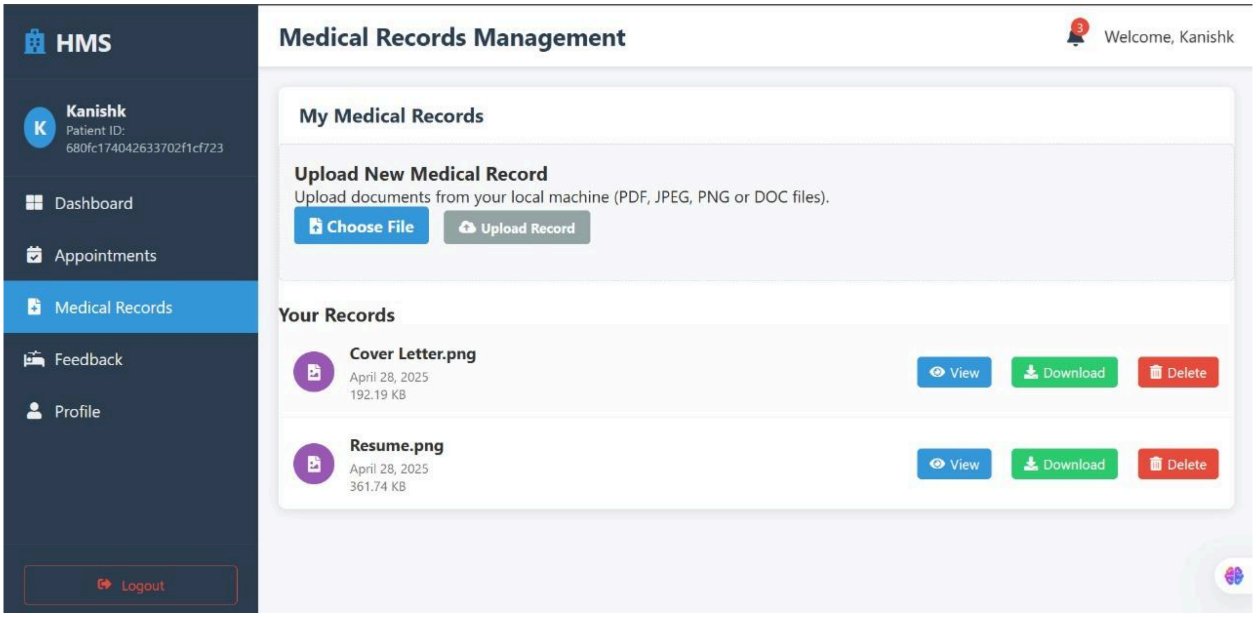
Consultation Fee
March 28, 2025
$150.00



Manage Appointments

Your Appointments

Doctor	specialization	Date	Time	Actions
Dr. Michael Chen	Neurology	Tue Apr 29 2025	12:00 pm	Reschedule Delete



Medical Records Management

Kanishk
Patient ID: 680fc174042633702f1cf723

My Medical Records

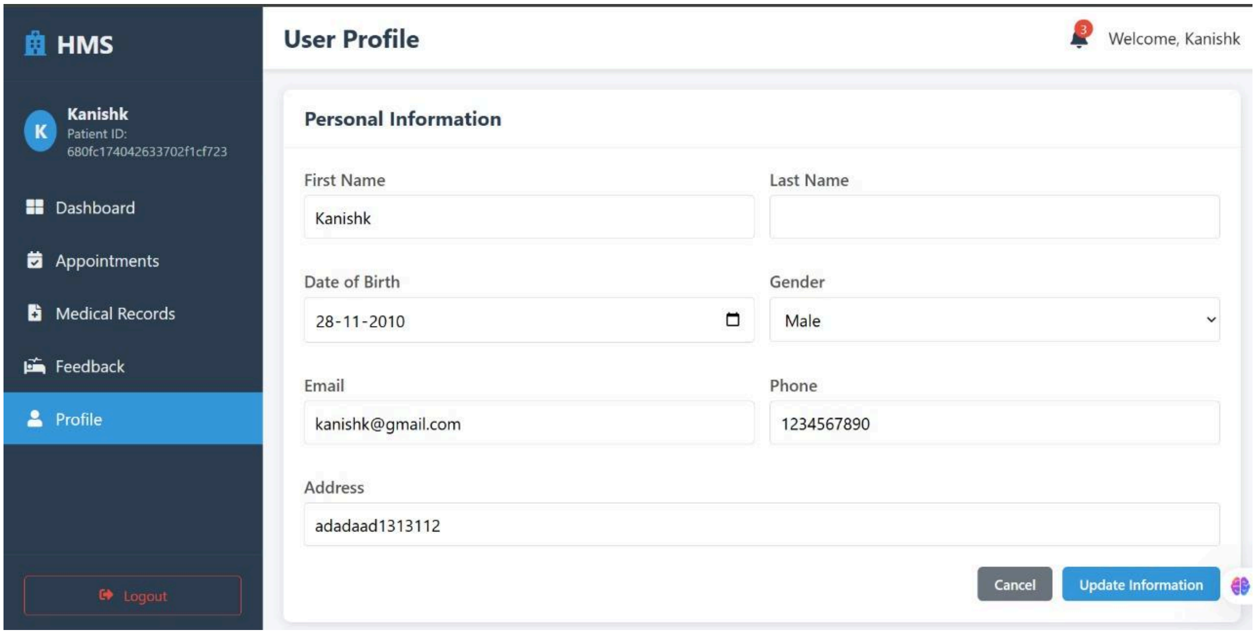
Upload New Medical Record

Upload documents from your local machine (PDF, JPEG, PNG or DOC files).

Your Records

Cover Letter.png
April 28, 2025
192.19 KB

Resume.png
April 28, 2025
361.74 KB



User Profile

Welcome, Kanishk

Kanishk
Patient ID:
680fc174042633702f1cf723

Personal Information

First Name: Kanishk

Last Name:

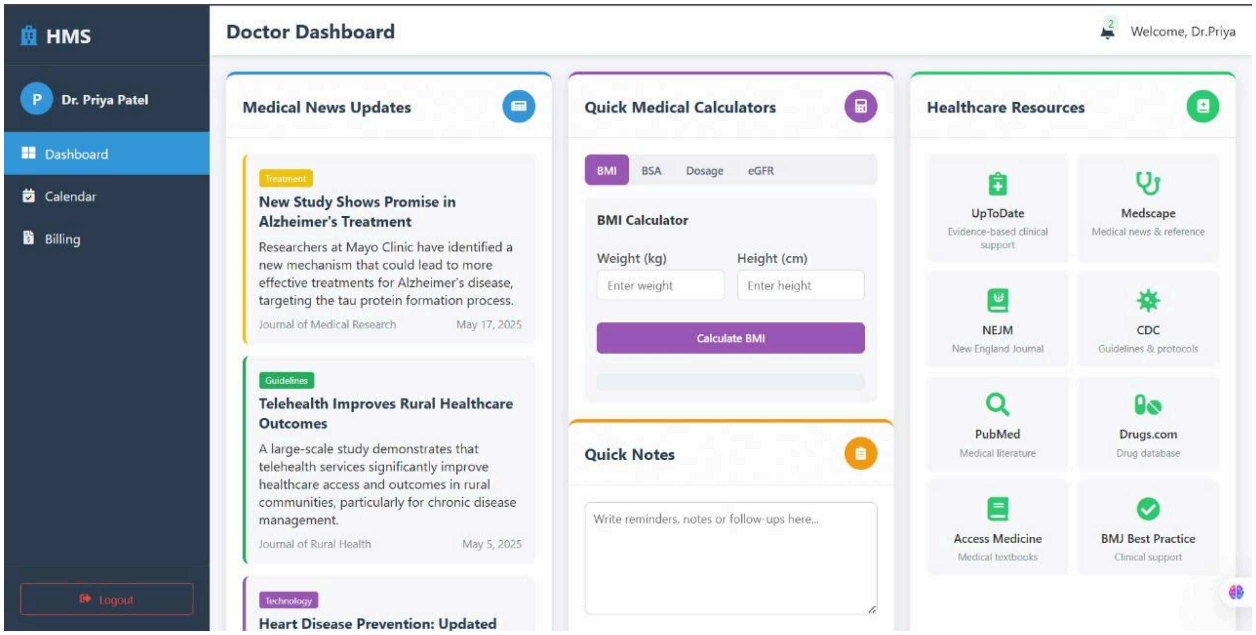
Date of Birth: 28-11-2010

Gender: Male

Email: kanishk@gmail.com

Phone: 1234567890

Address: adadaad1313112



Doctor Dashboard

Welcome, Dr.Priya

Dr. Priya Patel

Medical News Updates

Treatment

New Study Shows Promise in Alzheimer's Treatment

Researchers at Mayo Clinic have identified a new mechanism that could lead to more effective treatments for Alzheimer's disease, targeting the tau protein formation process.

Journal of Medical Research May 17, 2025

Guidelines

Telehealth Improves Rural Healthcare Outcomes

A large-scale study demonstrates that telehealth services significantly improve healthcare access and outcomes in rural communities, particularly for chronic disease management.

Journal of Rural Health May 5, 2025

Technology

Heart Disease Prevention: Updated

Quick Medical Calculators

BMI Calculator

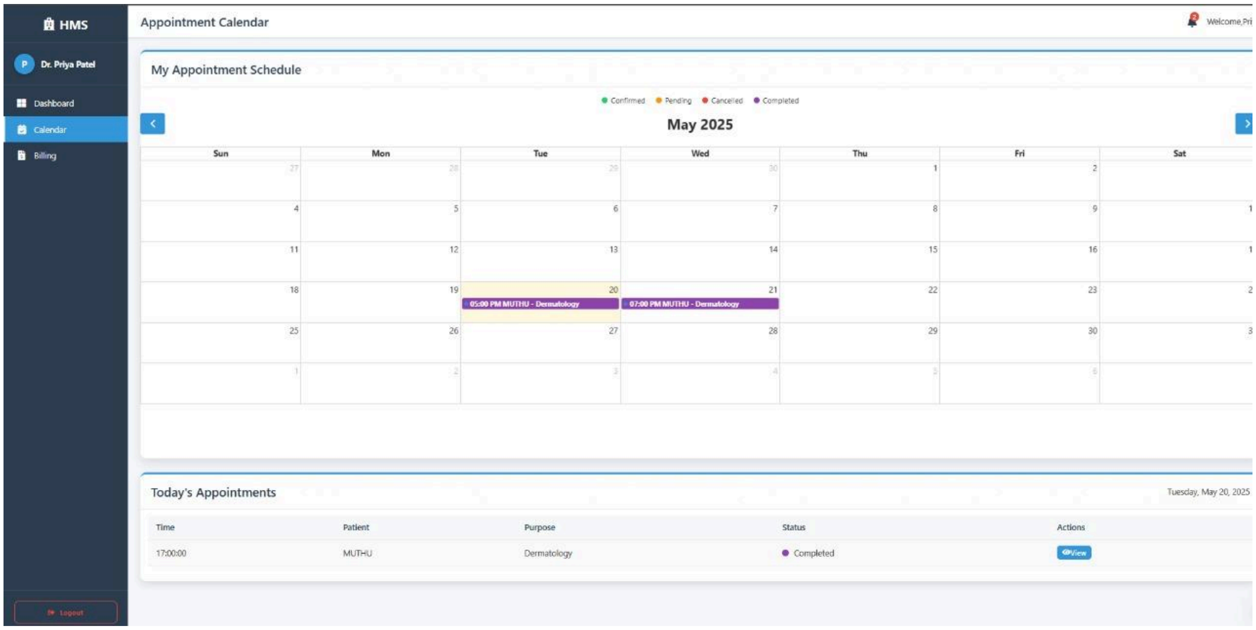
Weight (kg): Enter weight

Height (cm): Enter height

Quick Notes

Write reminders, notes or follow-ups here...

Healthcare Resources



Appointment Calendar

My Appointment Schedule

● Confirmed ● Pending ● Cancelled ● Completed

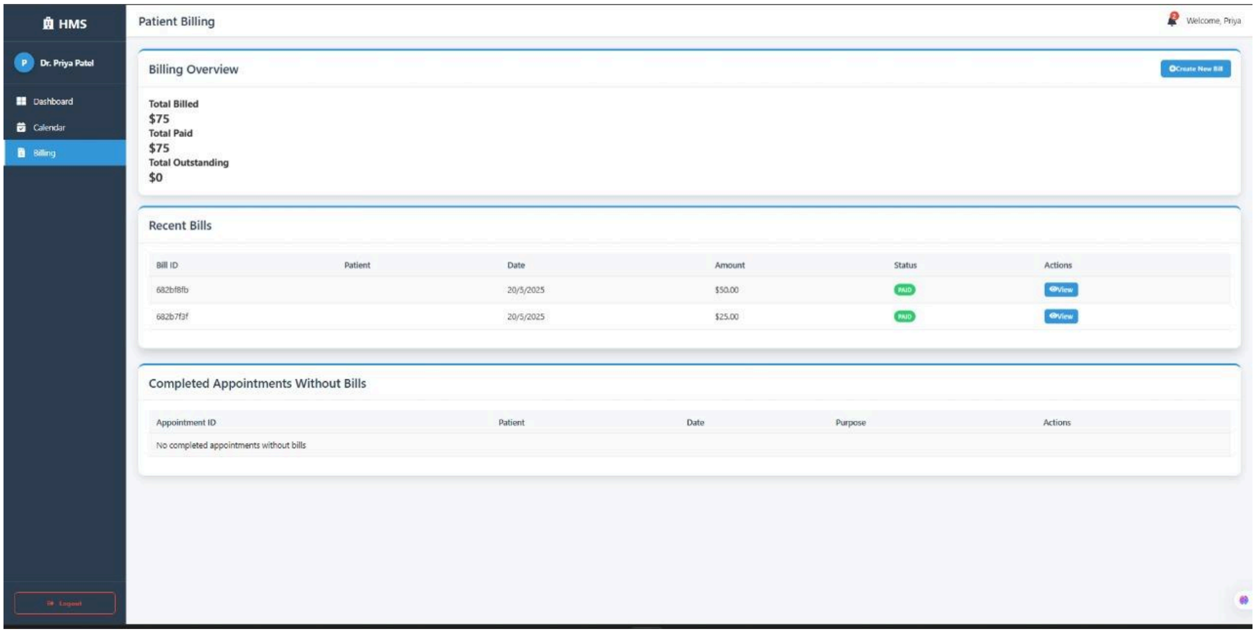
May 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2	
4	5	6	7	8	9	1
11	12	13	14	15	16	1
18	19	20 05:00 PM MUTHU - Dermatology	21 07:00 PM MUTHU - Dermatology	22	23	2
25	26	27	28	29	30	3
1	2	3	4	5	6	

Today's Appointments

Tuesday, May 20, 2025

Time	Patient	Purpose	Status	Actions
17:00:00	MUTHU	Dermatology	● Completed	View



Patient Billing

Billing Overview

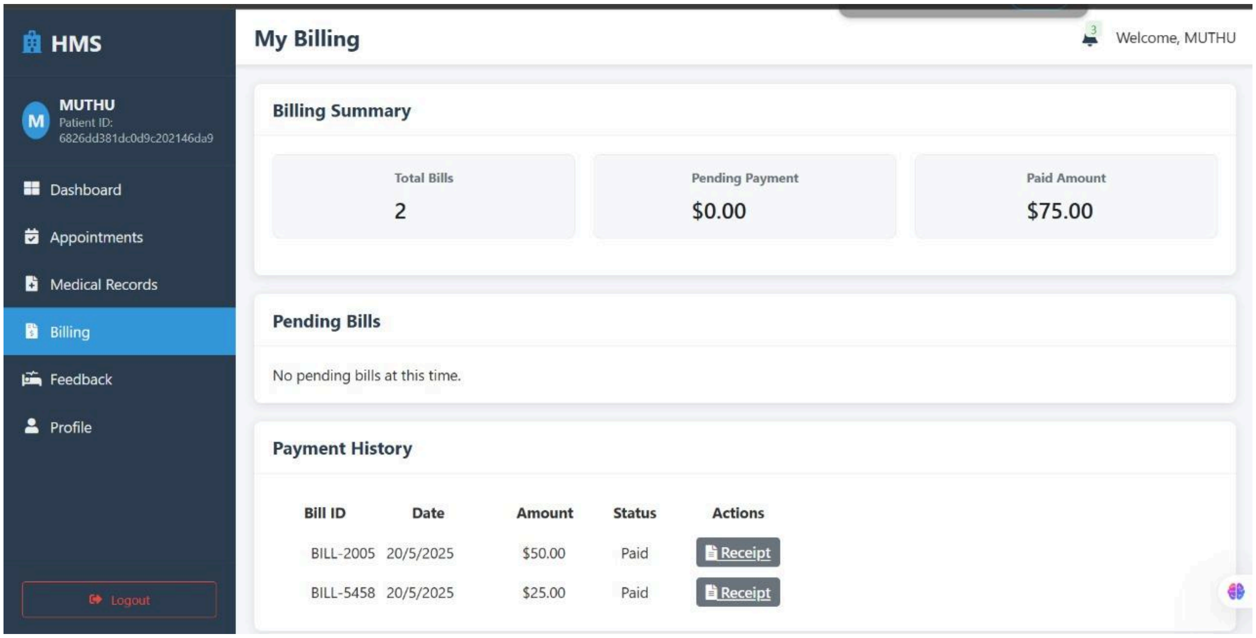
Total Billed
$75
Total Paid
$75
Total Outstanding
$0

Recent Bills

Bill ID	Patient	Date	Amount	Status	Actions
682bf8fb		20/5/2025	$50.00	PAID	View
682b7f3f		20/5/2025	$25.00	PAID	View

Completed Appointments Without Bills

Appointment ID	Patient	Date	Purpose	Actions
No completed appointments without bills				



My Billing

Billing Summary

Total Bills	Pending Payment	Paid Amount
2	$0.00	$75.00

Pending Bills

No pending bills at this time.

Payment History

Bill ID	Date	Amount	Status	Actions
BILL-2005	20/5/2025	$50.00	Paid	Receipt
BILL-5458	20/5/2025	$25.00	Paid	Receipt

RESULT:

The UI was designed successfully.

Ex. No. : 11 **Date : 24-04-2025**

Register No. : 230701385 **Name : S VISHWAK**

IMPLEMENTATION

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 <repo_url>

  cd <repo_folder>

- 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 the 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 application was successfully implemented.

Ex. No. : 12 **Date : 08-05-2025**

Register No. : 230701350 **Name : SUDHARSHAN KRISHNAA L K**

CI/CD PIPELINE

AIM:

To set up CI/CD pipelines.

PROCEDURE:

Step 1: Set Up the Azure DevOps Project and Repository

1. Sign in to Azure DevOps.
2. Create a new project or open an existing one.
3. Navigate to Repos > Files.
4. Upload your source code or clone the repository from GitHub or another remote location.
5. Make sure your project includes required build configuration files (e.g., `package.json` for Node.js, `pom.xml` for Java, or `.csproj` for .NET).

Step 2: Create a Build Pipeline (CI - Continuous Integration)

1. Go to Pipelines > Create Pipeline.
2. Choose the source repository where your code is hosted.
3. Select a pipeline configuration method:

   - Use the YAML template (recommended) or
   - Use the classic editor for a GUI-based setup.

4. Define build steps depending on your project type:
   - For Node.js: install dependencies and run build script.
   - For Python: set up an environment and run test cases.
5. Save and run the pipeline to verify that the code builds successfully.

Step 3: Set Up a Release Pipeline (CD - Continuous Deployment)

1. Navigate to Pipelines > Releases and click New Release Pipeline.
2. Start with an empty job or use a template (e.g., Azure App Service deployment).
3. Add an artifact by linking it to the output from the build pipeline created in Step 2.
4. Name and configure the deployment stages such as Dev, QA, and Production.
5. Add deployment tasks to the stage (e.g., deploy to Azure App Service or Virtual Machine).

Step 4: Configure Deployment Settings

1. In each stage, click Tasks and select the deployment method (e.g., Azure App Service Deploy).

2. Choose the appropriate Azure subscription and select the target App Service or VM.

3. Specify deployment package or folder (usually taken from the build artifact).

4. Configure additional settings like environment variables or deployment slots if needed.

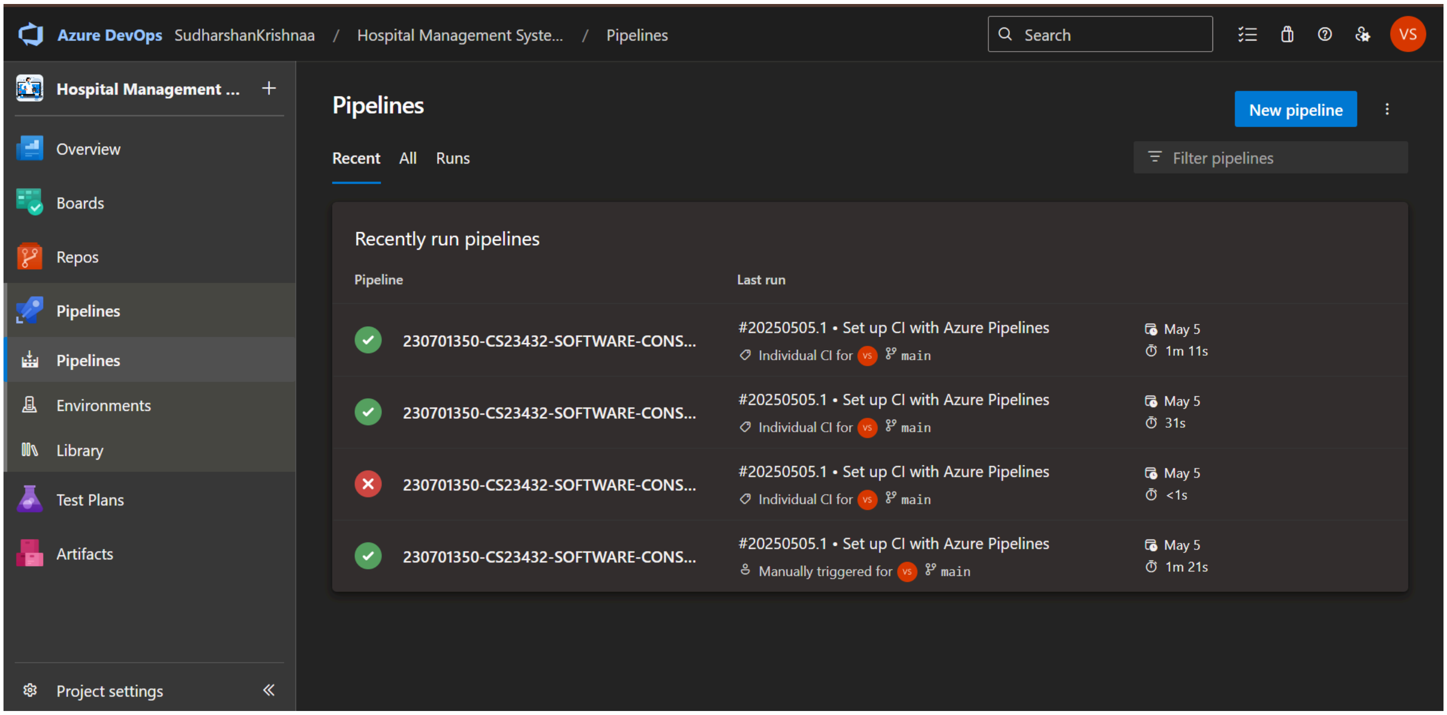
Step 5: Trigger and Monitor the Pipeline

1. Set up automatic deployment triggers by enabling Continuous Deployment trigger on the artifact.

2. Save the release pipeline and click Create Release.

3. Monitor the deployment in the Releases section.

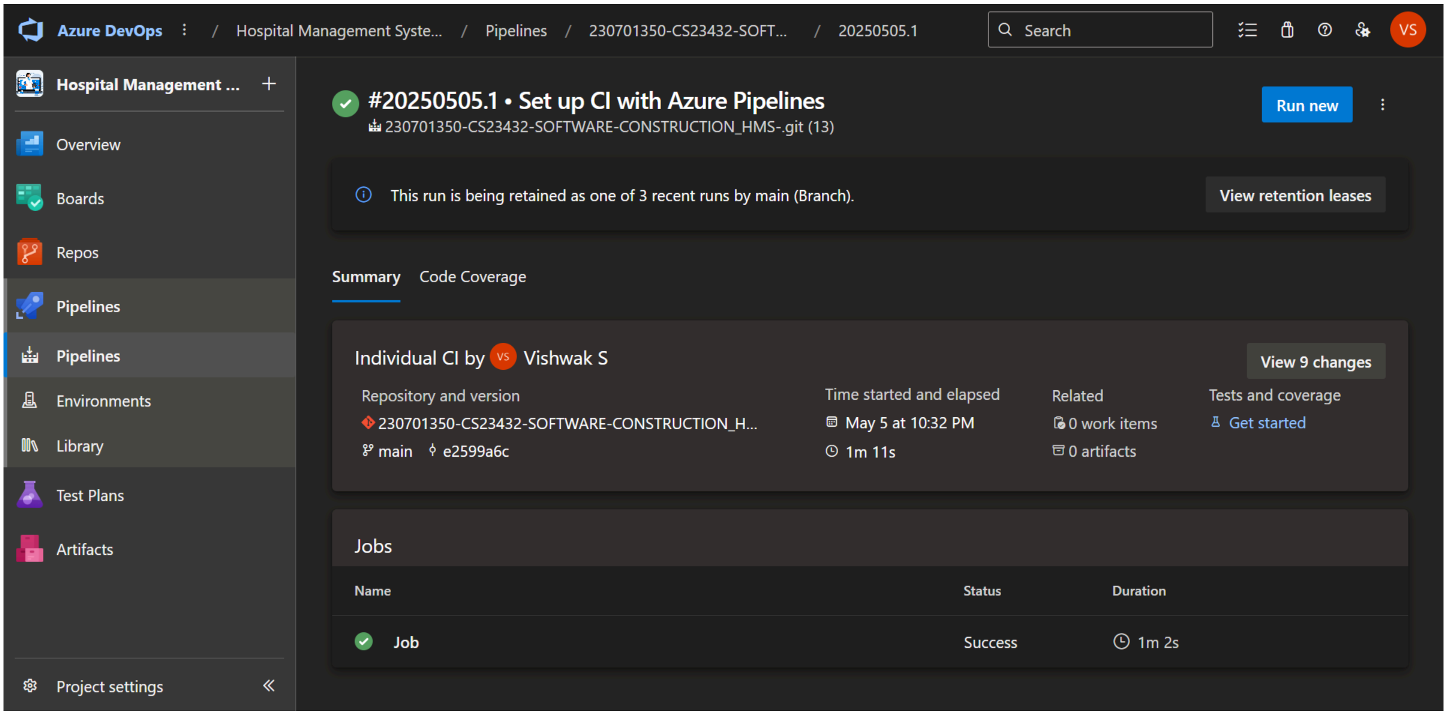
4. Once deployed, verify the application on the respective environment (Dev, QA, or Production)



Pipelines

Recently run pipelines

Pipeline	Last run	
✓ 230701350-CS23432-SOFTWARE-CONS...	#20250505.1 • Set up CI with Azure Pipelines Individual CI for VS main	May 5 1m 11s
✓ 230701350-CS23432-SOFTWARE-CONS...	#20250505.1 • Set up CI with Azure Pipelines Individual CI for VS main	May 5 31s
✗ 230701350-CS23432-SOFTWARE-CONS...	#20250505.1 • Set up CI with Azure Pipelines Individual CI for VS main	May 5 <1s
✓ 230701350-CS23432-SOFTWARE-CONS...	#20250505.1 • Set up CI with Azure Pipelines Manually triggered for VS main	May 5 1m 21s



✓ #20250505.1 • Set up CI with Azure Pipelines

230701350-CS23432-SOFTWARE-CONSTRUCTION_HMS-.git (13)

This run is being retained as one of 3 recent runs by main (Branch).

Individual CI by VS Vishwak S

Repository and version: 230701350-CS23432-SOFTWARE-CONSTRUCTION_H... main e2599a6c

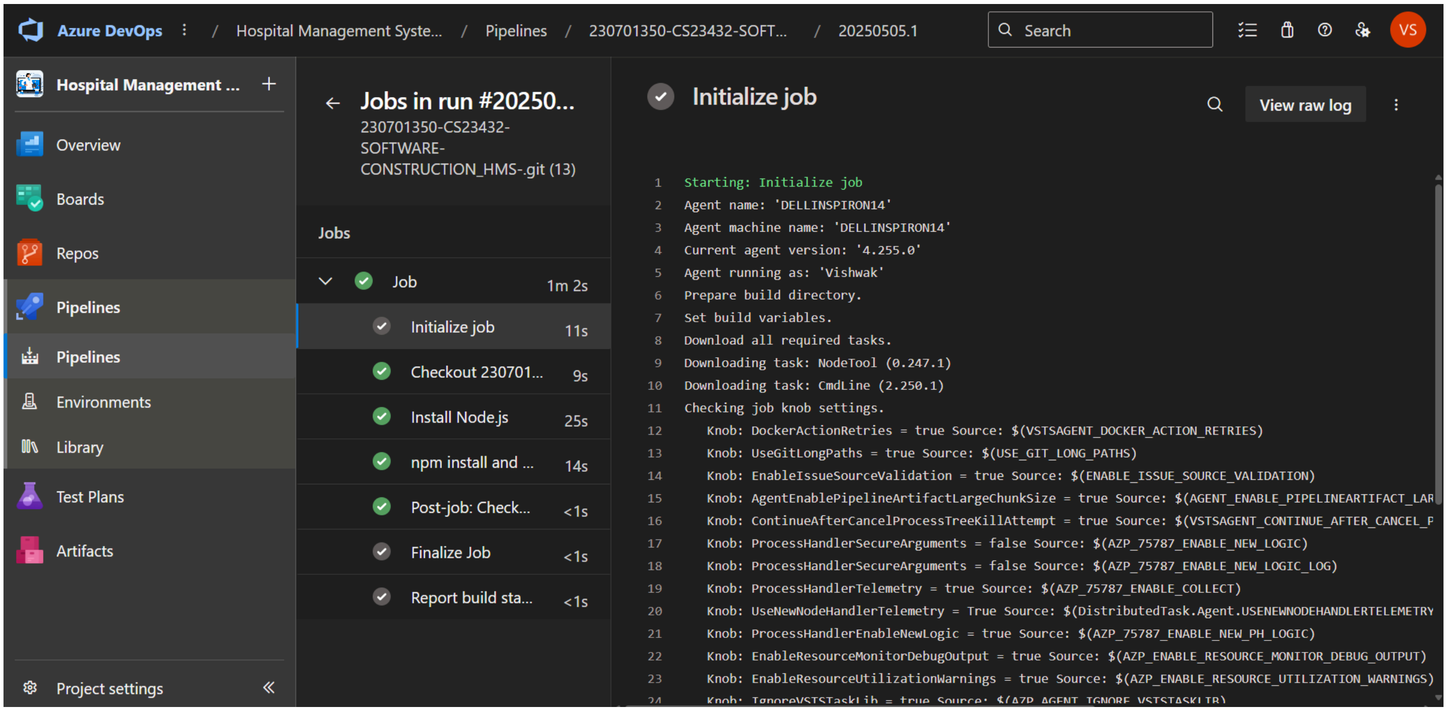
Time started and elapsed: May 5 at 10:32 PM 1m 11s

Related: 0 work items 0 artifacts

Tests and coverage: Get started

Jobs

Name	Status	Duration
✓ Job	Success	1m 2s



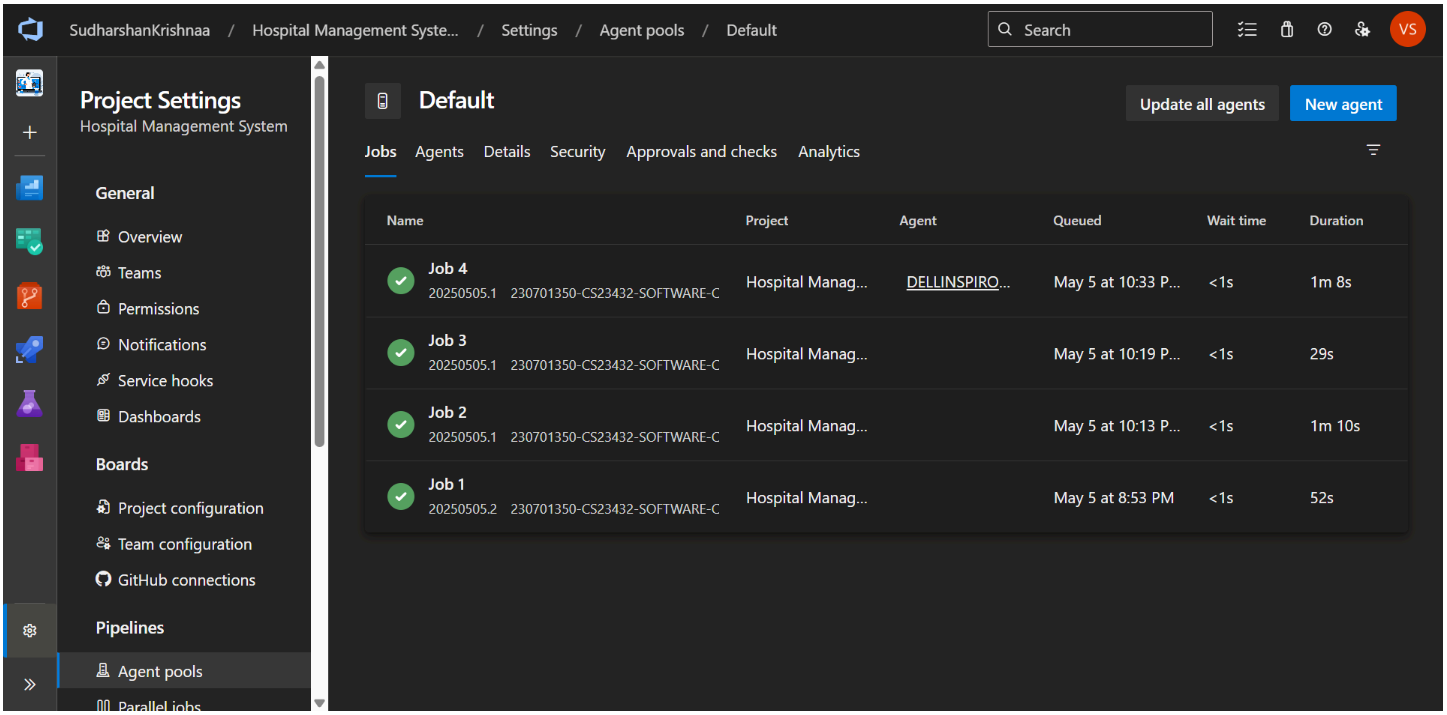
Jobs in run #20250...
230701350-CS23432-SOFTWARE-CONSTRUCTION_HMS-.git (13)

Jobs

Job	1m 2s
Initialize job	11s
Checkout 230701...	9s
Install Node.js	25s
npm install and ...	14s
Post-job: Check...	<1s
Finalize Job	<1s
Report build sta...	<1s

Initialize job

```
Starting: Initialize job
Agent name: 'DELLINSPIRON14'
Agent machine name: 'DELLINSPIRON14'
Current agent version: '4.255.0'
Agent running as: 'Vishwak'
Prepare build directory.
Set build variables.
Download all required tasks.
Downloading task: NodeTool (0.247.1)
Downloading task: CmdLine (2.250.1)
Checking job knob settings.
   Knob: DockerActionRetries = true Source: $(VSTSAGENT_DOCKER_ACTION_RETRIES)
   Knob: UseGitLongPaths = true Source: $(USE_GIT_LONG_PATHS)
   Knob: EnableIssueSourceValidation = true Source: $(ENABLE_ISSUE_SOURCE_VALIDATION)
   Knob: AgentEnablePipelineArtifactLargeChunkSize = true Source: $(AGENT_ENABLE_PIPELINEARTIFACT_LAR
   Knob: ContinueAfterCancelProcessTreeKillAttempt = true Source: $(VSTSAGENT_CONTINUE_AFTER_CANCEL_P
   Knob: ProcessHandlerSecureArguments = false Source: $(AZP_75787_ENABLE_NEW_LOGIC)
   Knob: ProcessHandlerSecureArguments = false Source: $(AZP_75787_ENABLE_NEW_LOGIC_LOG)
   Knob: ProcessHandlerTelemetry = true Source: $(AZP_75787_ENABLE_COLLECT)
   Knob: UseNewNodeHandlerTelemetry = True Source: $(DistributedTask.Agent.USENEWNODEHANDLERTELEMETRY
   Knob: ProcessHandlerEnableNewLogic = true Source: $(AZP_75787_ENABLE_NEW_PH_LOGIC)
   Knob: EnableResourceMonitorDebugOutput = true Source: $(AZP_ENABLE_RESOURCE_MONITOR_DEBUG_OUTPUT)
   Knob: EnableResourceUtilizationWarnings = true Source: $(AZP_ENABLE_RESOURCE_UTILIZATION_WARNINGS)
   Knob: IgnoreVSTSTaskLib = true Source: $(AZP_AGENT_IGNORE_VSTSTASKLIB)
```



Default

Name	Project	Agent	Queued	Wait time	Duration
Job 4 20250505.1 230701350-CS23432-SOFTWARE-C	Hospital Manag...	DELLINSPIRO...	May 5 at 10:33 P...	<1s	1m 8s
Job 3 20250505.1 230701350-CS23432-SOFTWARE-C	Hospital Manag...		May 5 at 10:19 P...	<1s	29s
Job 2 20250505.1 230701350-CS23432-SOFTWARE-C	Hospital Manag...		May 5 at 10:13 P...	<1s	1m 10s
Job 1 20250505.2 230701350-CS23432-SOFTWARE-C	Hospital Manag...		May 5 at 8:53 PM	<1s	52s

RESULT:

Thus the CI/CD pipeline was successfully implemented.