RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM – 602 105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025

RAJALAKSHMI
ENGINEERING COLLEGE

CS23432

SOFTWARE CONSTRUCTION

Lab Manual

2024-2025

Name: Suganya S

Year/Branch/Section : II /CSE/ D

Register No. : 230701351

Semester : IV

Academic Year: 2024-25

Ex 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 Usecase 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

LAB PLAN

CS19442-SOFTWARE ENGINEERING LAB

Ex No	Date	Topic	Page No	Sign
1		Study of Azure DevOps		
2		Writing Problem Statement		
3		Designing Project using AGILE-SCRUM Methodology by using Azure.		
4		Agile Planning		
5		User stories – Creation		
6		Architecture Diagram Using AZURE		
7		Designing Usecse Diagram using StarUML		
8		Designing Activity Diagrams using StarUML		
9		Designing Sequence Diagrams using StarUML		
10		Design Class Diagram		
11		Design User Interface		

11		Implementation – Design a Web Page based on Scrum Methodology		
12		Testing		
13		Deployment		

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: "-"

EX NO : 1 STUDY OF AZURE DEVOPS

AIM:

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

STUDY:

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 was successfully completed.

EX NO : 2 PROBLEM STATEMENT

AIM :

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

In modern healthcare environments, managing patient information, scheduling appointments, and maintaining accurate health records manually or using outdated systems often leads to inefficiencies, delays, and medical errors. Hospitals and clinics need a centralized and efficient solution to streamline administrative and medical workflows.

The **Hospital Management System (HMS)** aims to address these challenges by providing a unified digital platform that supports:

- **Patient Registration**: Simplifying the onboarding process by allowing front desk staff to register new patients with mandatory details and generate a unique patient ID. This ensures consistency and helps maintain a centralized database for easy retrieval and updates.
- **Appointment Scheduling**: Enabling patients to book appointments with doctors based on their availability and receive automated reminders via SMS or email. It also helps doctors manage their daily schedule efficiently and view relevant patient information ahead of consultations.
- **Electronic Health Records (EHR)**: Creating and maintaining digital records of patient histories, prescriptions, diagnoses, and test results. Doctors can access and update these records securely during consultations, while patients can view their medical data through a secure portal, promoting transparency and improved care coordination.

By implementing this system, hospitals can improve operational efficiency, reduce administrative workload, enhance patient experience, and ensure accurate and accessible medical data across departments.

RESULT:

The Problem statement is written successfully.

EX NO : 3 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 was completed successfully.

EX NO: 4 CREATE USER STORY

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.

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. Go to Azure Home Page.

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

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.

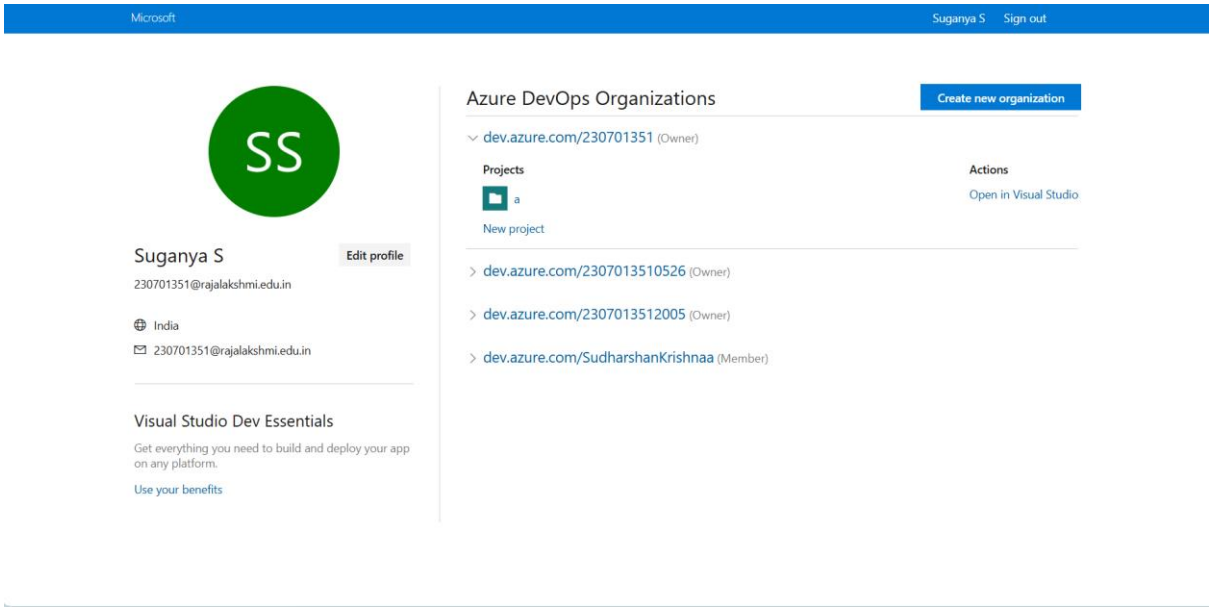
6. Create the First Project in Your Organization. After the organization is set up, you’ll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.

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

(ii)Enter the project name, description, and visibility options:

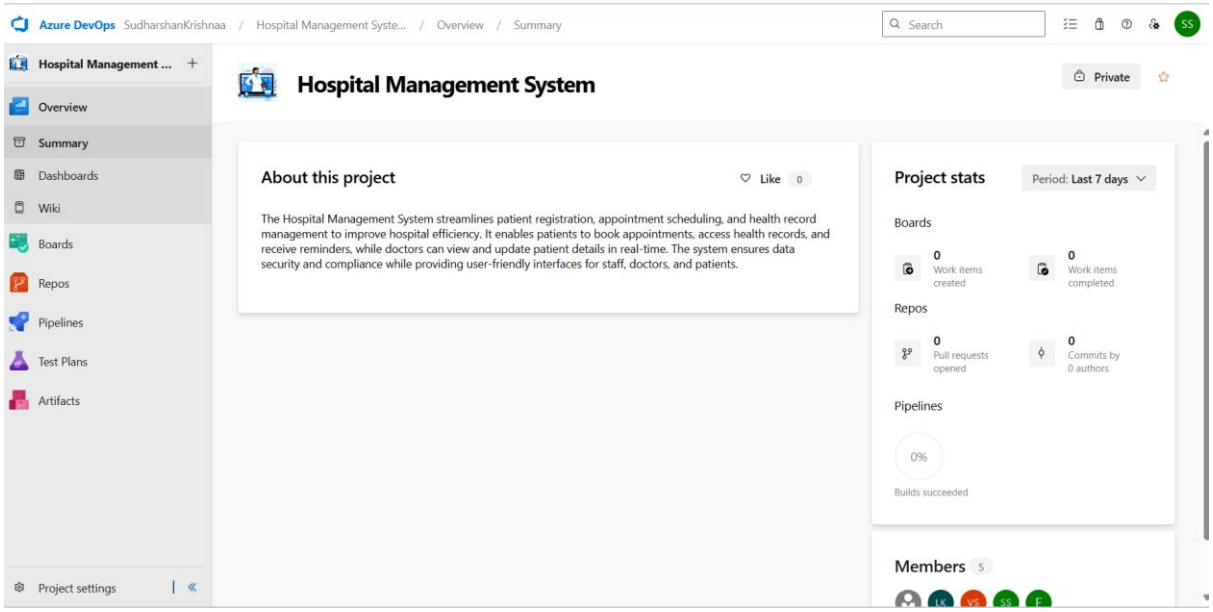
- Name: Choose a name for the project (e.g., LMS).
- Description: Optionally, add a description to provide more context about the project.
- Visibility: Choose whether you want the project to be Private
- (accessible only to those invited) or Public (accessible to anyone).

(iii) Once you’ve filled out the details, click Create 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. Open project's dashboard.

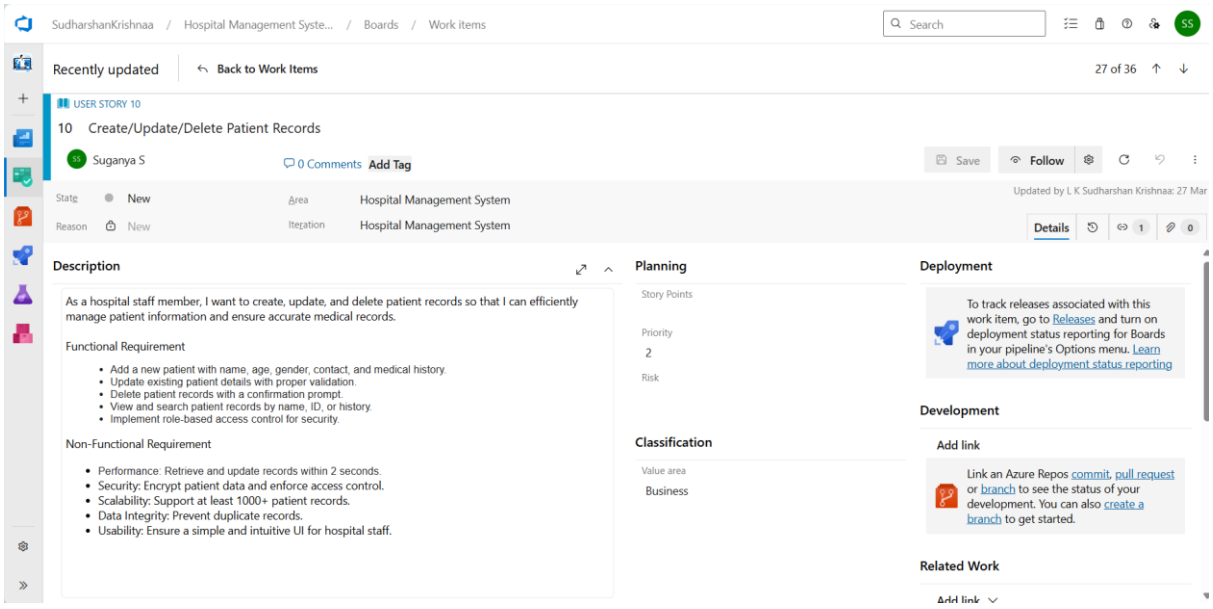


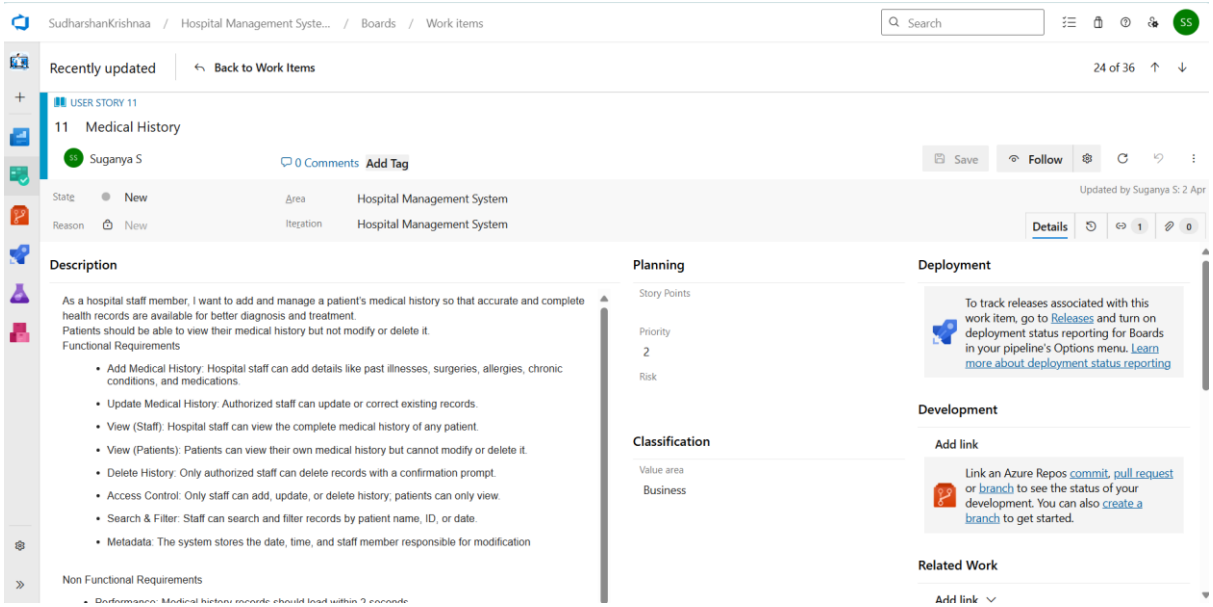
9. To manage user stories:

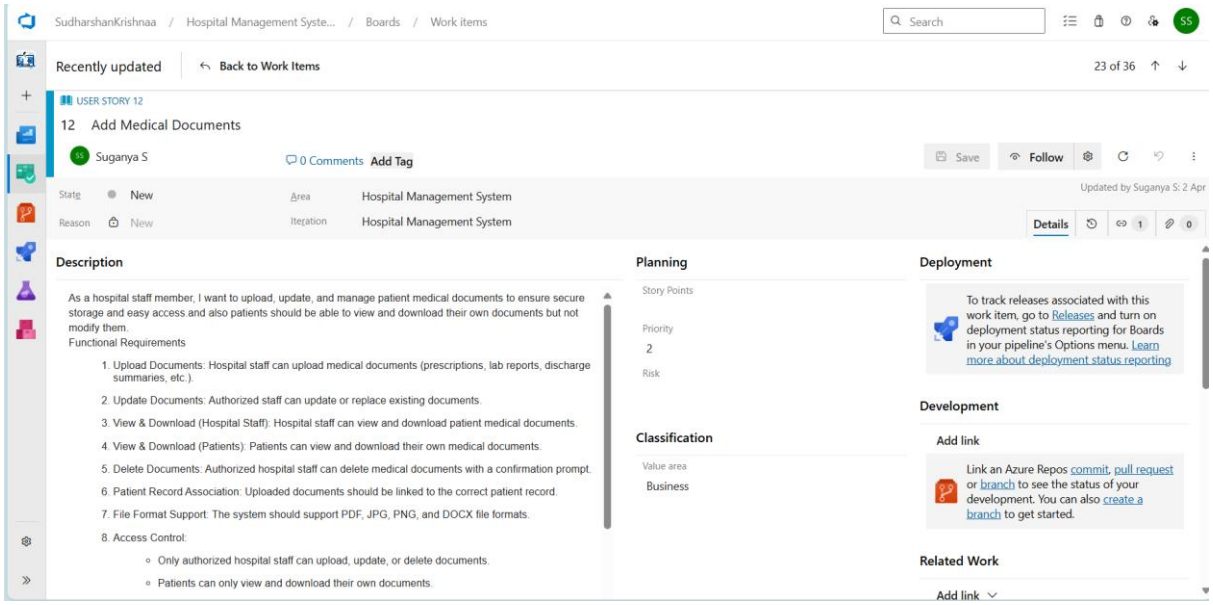
a. From the left-hand navigation menu, click on Boards. This will take you to the main Boards page, where you can manage work items, backlogs, and sprints.

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 the User Story.







EPIC: PATIENT MANAGEMENT SYSTEM

The Patient Management System is a key component of the Hospital Management System that focuses on handling patient data, medical history, and related documents securely and efficiently. Hospital staff can create, update, and delete patient records, upload and manage medical documents, and maintain detailed medical histories with proper access control. Patients can securely view and download their own medical information but cannot modify or delete any data. The system ensures quick access through search and filter options, maintains data integrity, and supports scalability to handle thousands of records. Designed with security and usability in mind, it improves hospital workflows and enhances the quality of patient care.

FEATURES:

1. Create/update/Delete Medical Document:

The system allows hospital staff to securely upload, update, and manage patient medical documents such as prescriptions, lab reports, and discharge summaries, while ensuring patients can only view and download their own documents.

Key Functionalities:
• Hospital staff can upload documents linked to specific patient records.
• Authorized staff can update or delete documents with a confirmation

prompt.
• Patients can view and download their own medical documents through a secure portal.
• The system supports multiple file formats including PDF, JPG, PNG, and DOCX.
• Each document entry includes metadata such as upload date, document type, and uploader's identity.
• Role-based access ensures only authorized staff can manage documents, maintaining data privacy.
• All documents are encrypted to protect patient confidentiality and ensure secure storage.

2. Medical History:

The system enables hospital staff to add, update, and manage detailed medical histories for each patient, ensuring accurate and complete records to support effective diagnosis and treatment. Patients can securely view their medical history but cannot modify or delete any entries.

Key Functionalities:
• Hospital staff can add details such as past illnesses, surgeries, allergies, chronic conditions, and medications.
• Authorized staff can update or delete records with proper validation and confirmation.
• Patients can view their own medical history via a secure interface but have no editing rights.
• The system supports search and filter options by patient name, ID, or date for easy access.
• All changes are tracked with metadata including timestamp and staff identity.
• Role-based access control ensures only authorized staff can make changes.
• Medical history data is encrypted to ensure confidentiality and loaded within 2 seconds for optimal performance.

3. Adding Medical Document :

The system allows hospital staff to securely upload, update, and manage patient medical documents such as prescriptions, lab reports, and discharge summaries. Patients can view and download their own

documents but cannot modify or delete them, ensuring data integrity and privacy.

Key Functionalities:
• Hospital staff can upload, update, or delete medical documents with confirmation.
• Documents are linked to the correct patient record and support formats like PDF, JPG, PNG, and DOCX.
• Patients can securely view and download their own documents.
• Role-based access control restricts editing privileges to authorized staff only.
• Each document includes metadata such as upload date, uploader name, and document type.
• System ensures encryption, fast retrieval (within 3 seconds), and prevents data loss or duplication

USER STORY 1:

As a hospital staff member, I want to create, update, and delete patient records so that I can efficiently manage patient information and ensure accurate medical records.

Acceptance criteria:

- Hospital staff can create a new patient record by entering required details.
- Hospital staff can update existing patient records with valid information.
- Hospital staff can delete patient records with a confirmation prompt.
- The system prevents duplicate patient records.
- Only authorized hospital staff can create, update, or delete records.
- Hospital staff can search and filter patient records for quick access.
- The system displays an error message if mandatory fields are missing.

USER STORY 2:

As a hospital staff member, I want to add and manage a patient's medical history so that accurate and complete health records are available for better diagnosis and treatment. Patients should be able to view their medical history but not modify or delete it.

Acceptance criteria:

- **Hospital staff** can add and update medical history records.
- **Patients** can view their own medical history but cannot modify or delete it.
- The system **restricts unauthorized users** from accessing or modifying records.
- The system **displays an error message** if required fields are missing.
- **Staff can search and filter** patient medical history by name, ID, or date.
- The system stores **metadata** for every modification (e.g., staff name and timestamp).

USER STORY 3:

As a hospital staff member, I want to upload, update, and manage patient medical documents to ensure secure storage and easy access.and also patients should be able to view and download their own documents but not modify them.

Acceptance Criteria:

- Hospital staff can upload, update, and manage patient medical documents.
- Patients can view and download their own medical documents.
- Hospital staff can delete medical documents with a confirmation prompt.
- The system restricts unauthorized users from modifying or deleting documents.
- The system prevents patients from modifying or deleting documents but allows them to view and download.
- The system supports multiple file formats (PDF, JPG, PNG, DOCX).
- The system displays an error message if an invalid file format is uploaded.

RESULT:

The assigned user story for my project has been written successfully.

EX NO: 5 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.

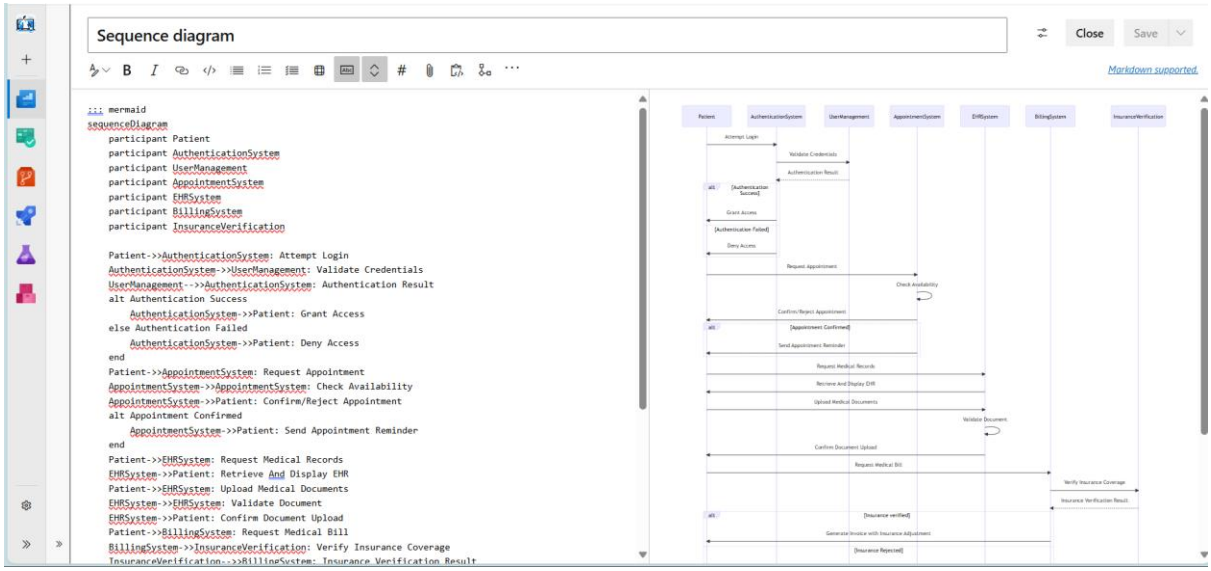
2. To design select wiki from menu.

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

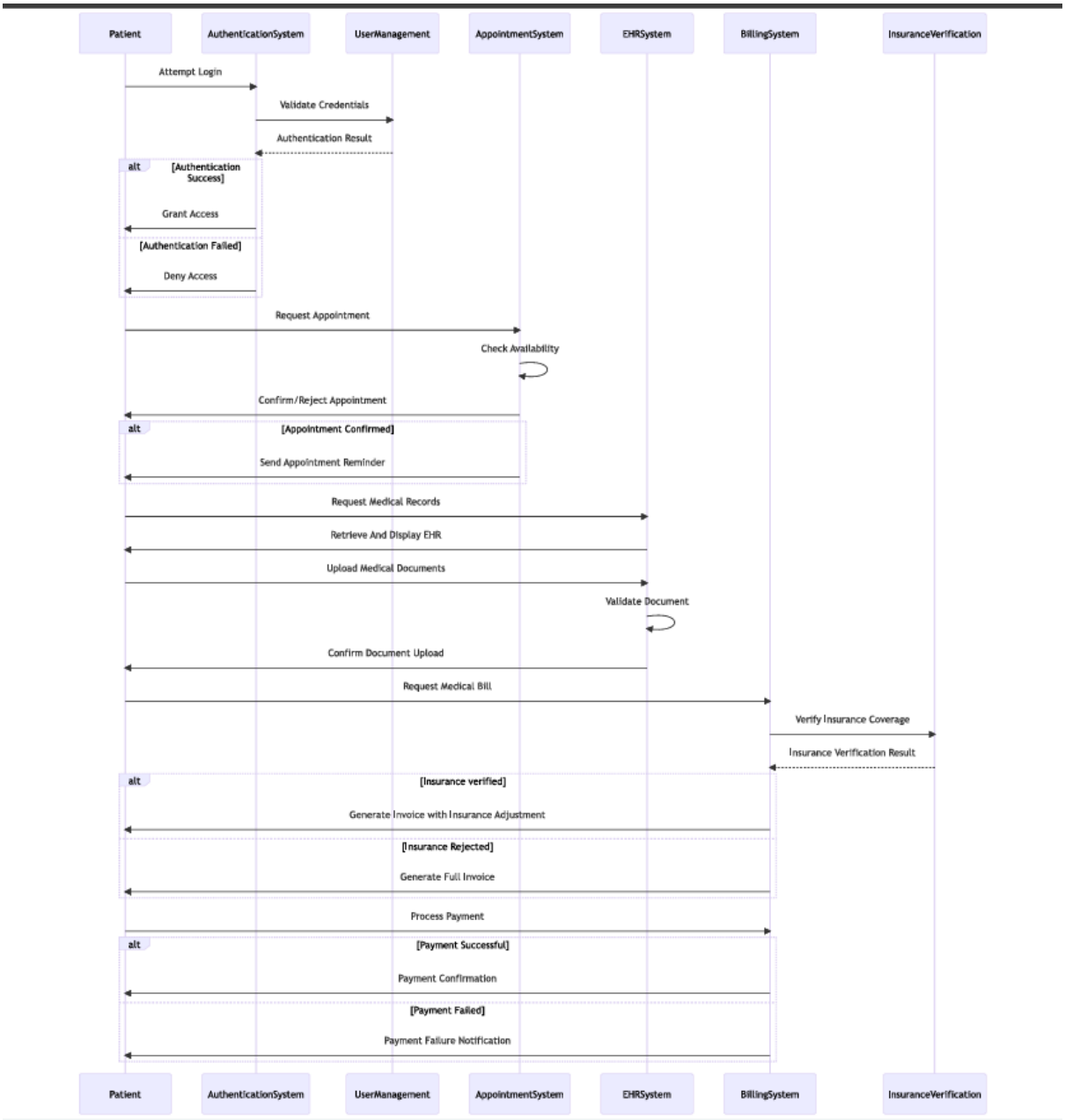
```
::: mermaid
sequenceDiagram
    participant Patient
    participant AuthenticationSystem
    participant UserManagement
    participant AppointmentSystem
    participant EHRSystem
    participant BillingSystem
    participant InsuranceVerification

    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
```

:::



4. Click wiki menu and select the page.



RESULT:

The sequence diagram is drawn successfully.

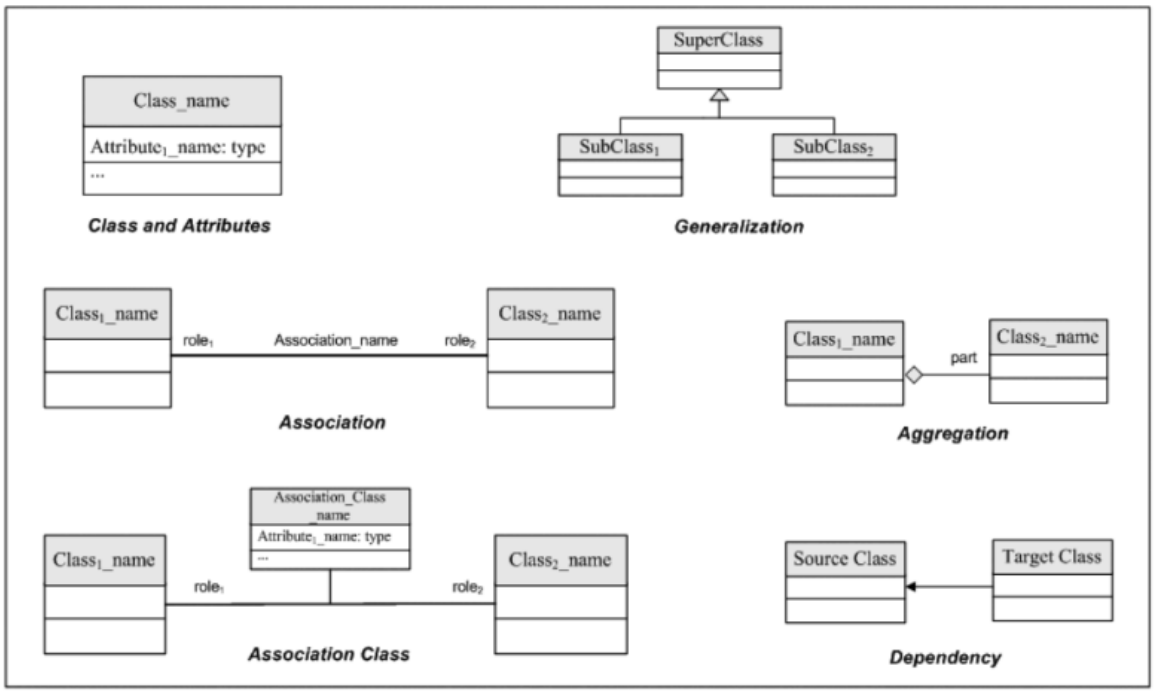
EX NO: 6 CLASS DIAGRAM

AIM:

To draw a simple class diagram.

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 the code for drawing Class Diagram and save the code.

:::mermaid

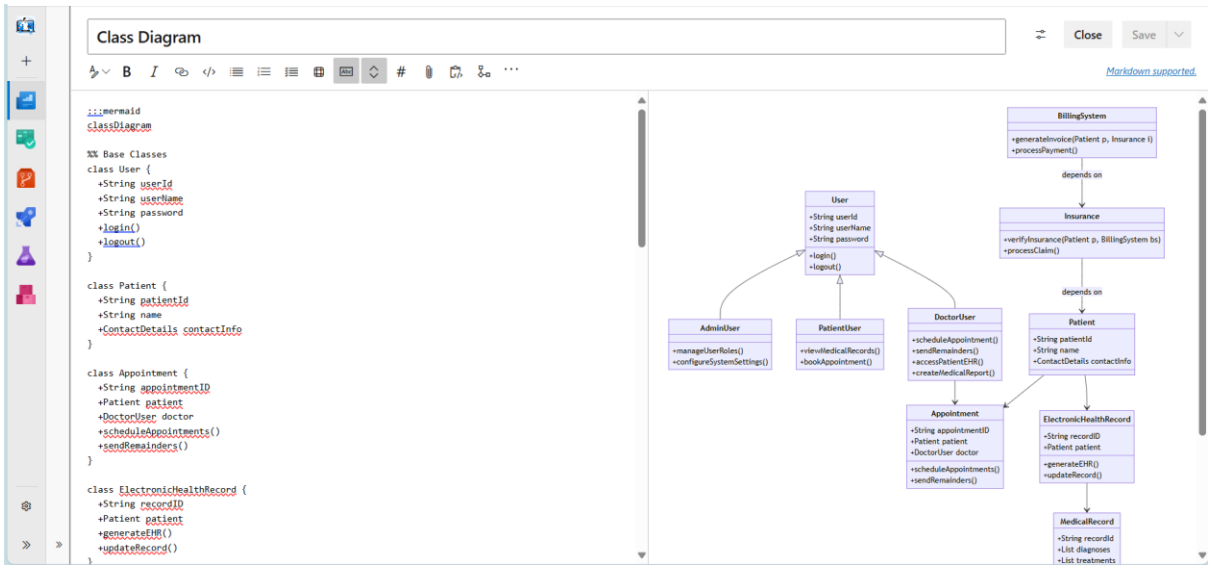
classDiagram

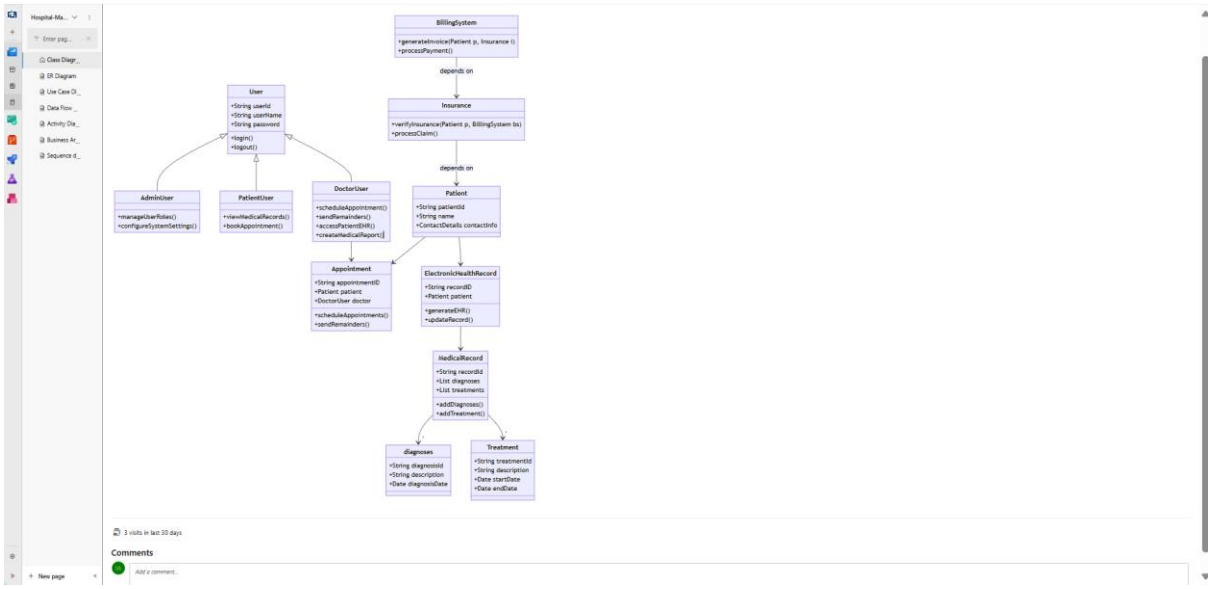
```
%% Base Classes
class User {
  +String userId
  +String userName
  +String password
  +login()
  +logout()
}

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

:::
```





RESULT:

Thus the class diagram has been designed successfully.

EX NO:7 USE CASE DIAGRAM

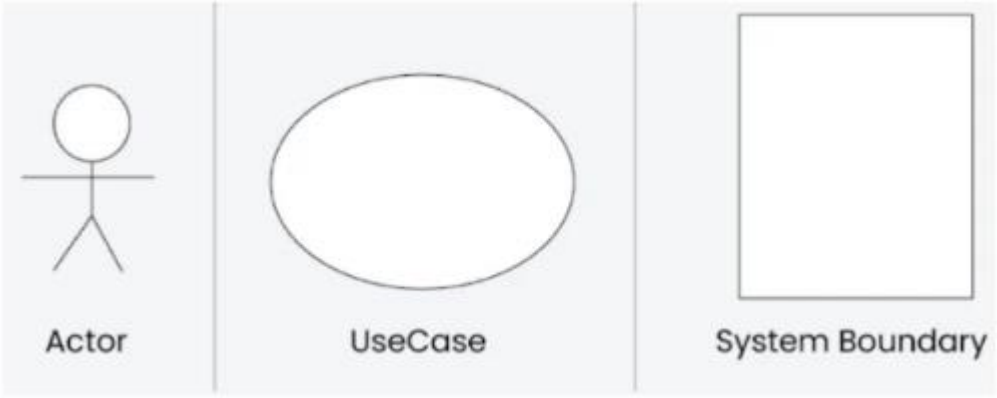
AIM:

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

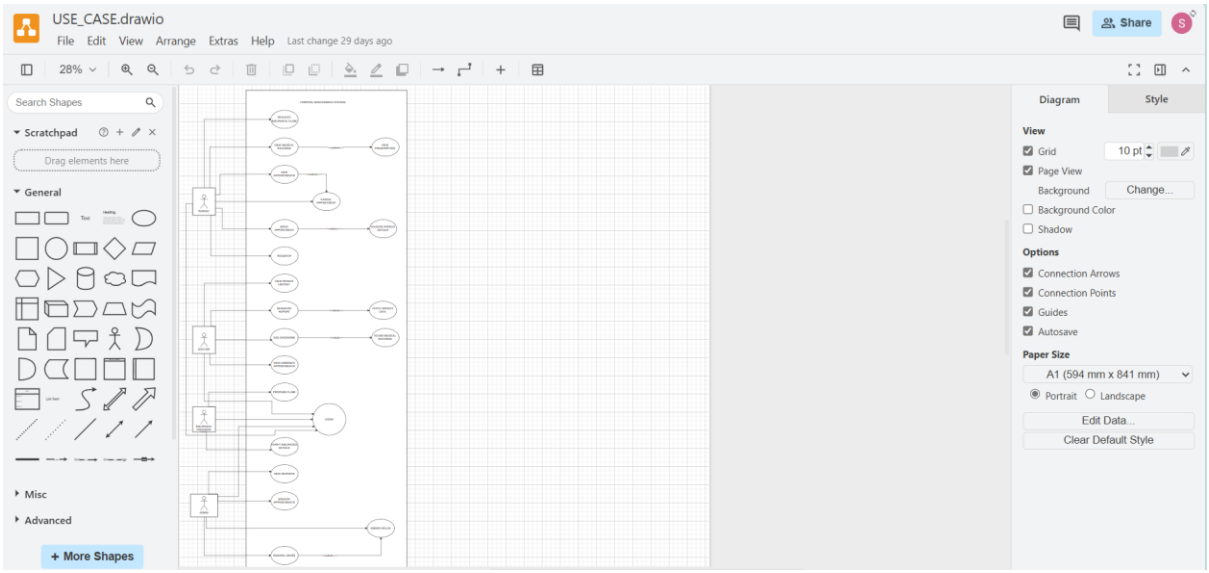
- Use Cases
- Actors
- Relationships
- System Boundary

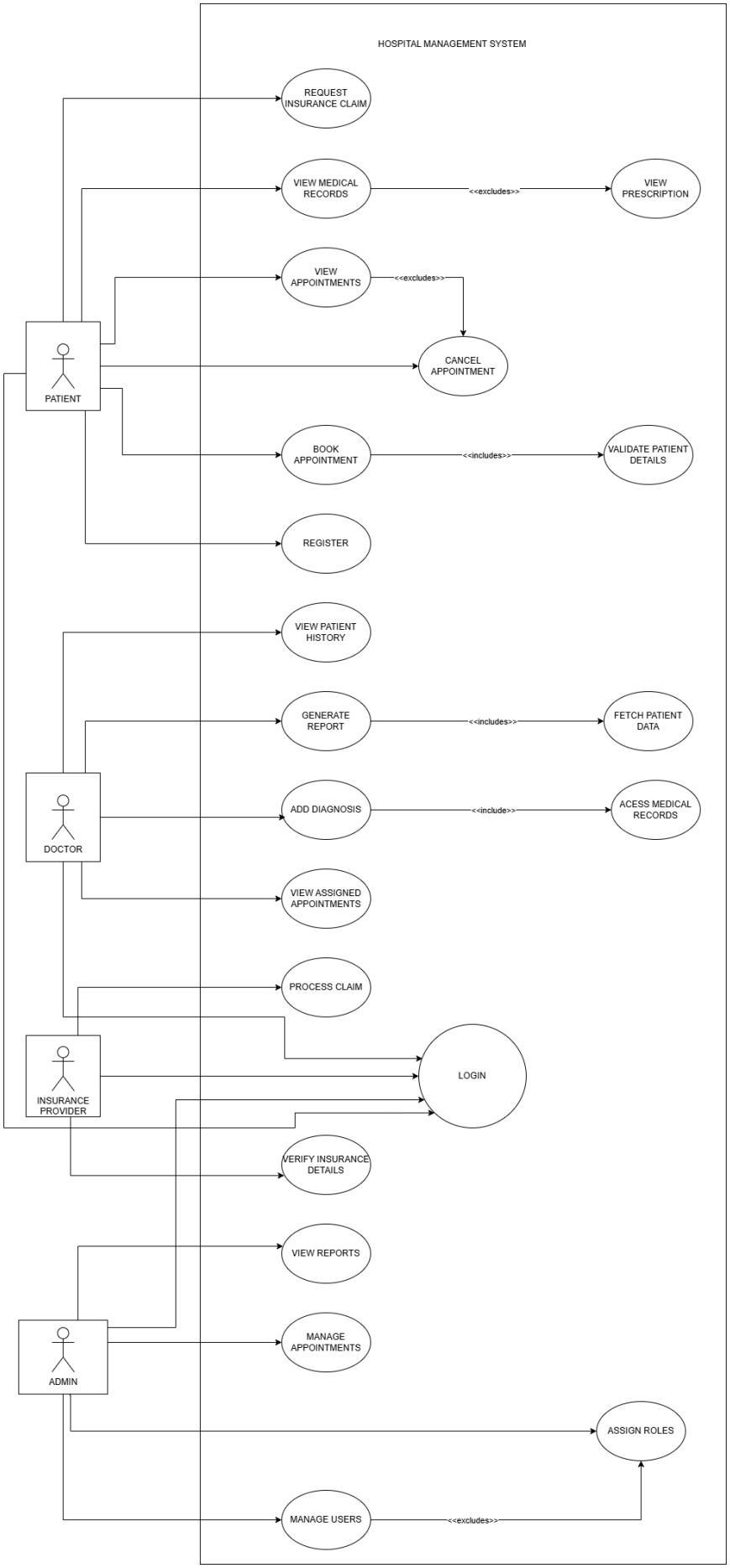


PROCEDURE :

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.


USE_CASE.drawio
File Edit View Arrange Extras Help Last change 29 days ago
Share
Search Shapes
Scratchpad
Drag elements here
General
Misc
Advanced
+ More Shapes
Diagram
Style
View
Grid
10 pt
Page View
Background
Change...
Background Color
Shadow
Options
Connection Arrows
Connection Points
Guides
Autosave
Paper Size
A1 (594 mm x 841 mm)
Portrait
Landscape
Edit Data...
Clear Default Style


HOSPITAL MANAGEMENT SYSTEM
REQUEST INSURANCE CLAIM
VIEW MEDICAL RECORDS
<<excludes>>
VIEW PRESCRIPTION
VIEW APPOINTMENTS
<<excludes>>
CANCEL APPOINTMENT
PATIENT
BOOK APPOINTMENT
<<includes>>
VALIDATE PATIENT DETAILS
REGISTER
VIEW PATIENT HISTORY
GENERATE REPORT
<<includes>>
FETCH PATIENT DATA
DOCTOR
ADD DIAGNOSIS
<<include>>
ACESS MEDICAL RECORDS
VIEW ASSIGNED APPOINTMENTS
PROCESS CLAIM
INSURANCE PROVIDER
LOGIN
VERIFY INSURANCE DETAILS
VIEW REPORTS
MANAGE APPOINTMENTS
ADMIN
ASSIGN ROLES
MANAGE USERS
<<excludes>>

Step 2: Upload the Diagram to Azure DevOps

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu.

3. Write the code for drawing USE CASE Diagram and save the 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)

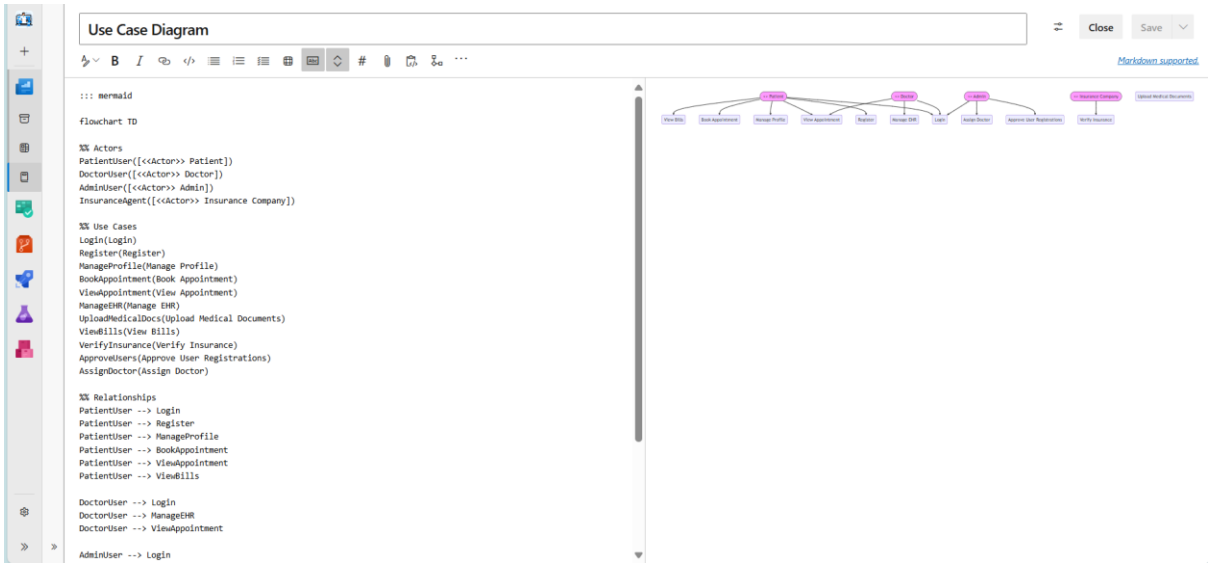
```
VerifyInsurance(Verify Insurance)

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
```

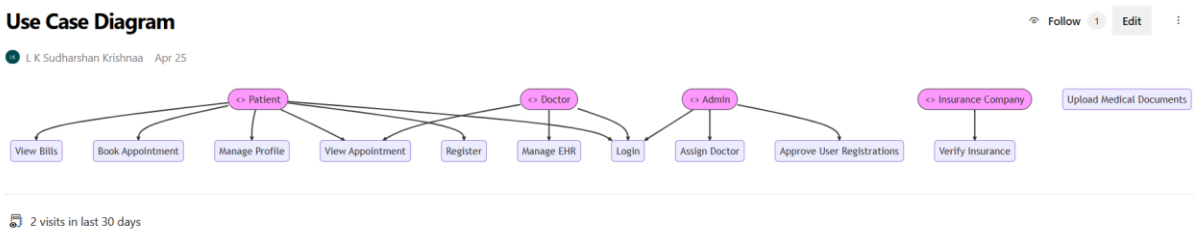
style DoctorUser fill:#f9f,stroke:#333,stroke-width:1px

style AdminUser fill:#f9f,stroke:#333,stroke-width:1px

style InsuranceAgent fill:#f9f,stroke:#333,stroke-width:1px

:::





RESULT:

The use case diagram was designed successfully.

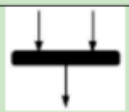
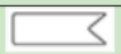
EX NO: 8 ACTIVITY DIAGRAM

AIM :

To draw a sample activity diagram for the Weather Application.

THEORY:

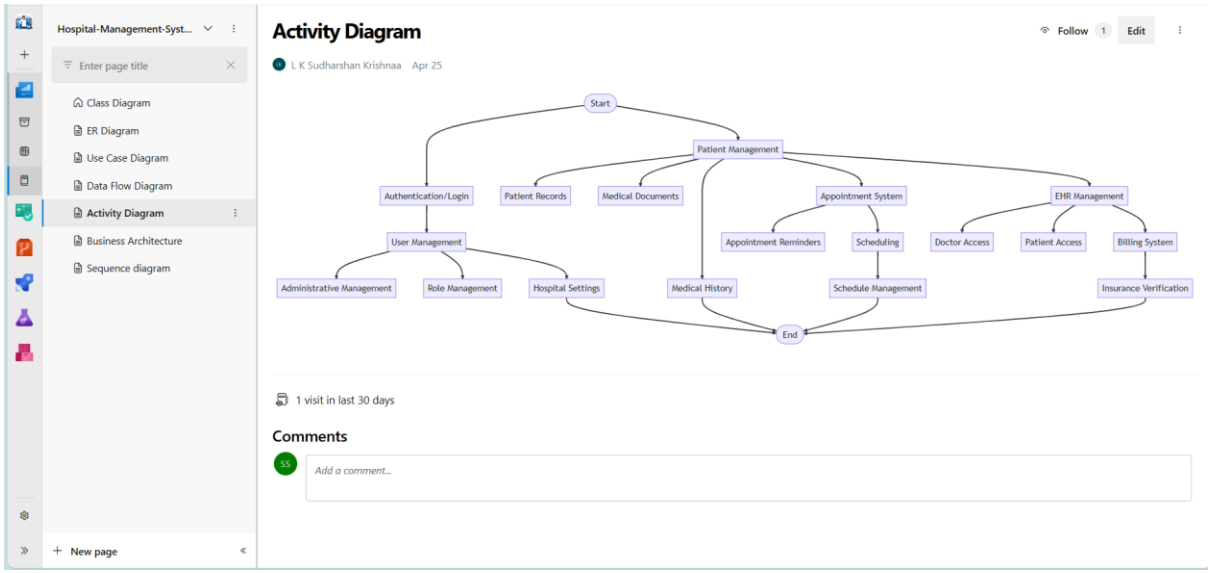
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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:

Step 1. Draw diagram in draw.io.

Step 2. Upload the diagram in Azure DevOps wiki.



RESULT:

The activity diagram was designed successfully.

EX NO:9 ARCHITECTURE DIAGRAM

AIM:

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:

Step 1. Draw diagram in draw.io

Step 2. Upload the diagram in Azure DevOps wiki.

Step 3.Write the code for drawing ARCHITECTURE Diagram and save the code

::: mermaid

flowchart TD

```
subgraph Client
    User["User (Patient / Staff)"]
    App["Mobile/Web App (React.js / Flutter)"]
end

subgraph Backend [Node.js + Express.js Backend]
    API["Express.js API"]
    Auth["Auth & Role Access"]
    Patient["Patient Service"]
    History["Medical History Service"]
    Documents["Medical Documents Service"]
    Appointments["Appointment Service"]
    Notify["Notification System"]
end

subgraph Database
    DB[(MongoDB)]
end


User --> App
App --> API
```

API --> Auth

API --> Patient

API --> History

API --> Documents

API --> Appointments

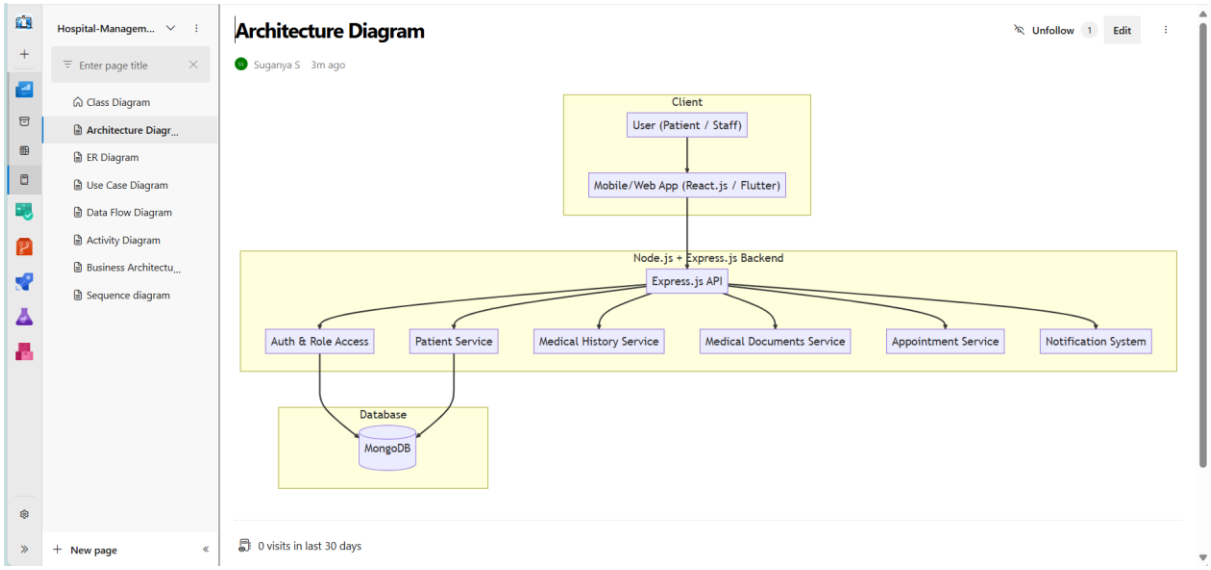
API --> Notify

Auth --> DB

Patient --> DB

History

:::



RESULT:

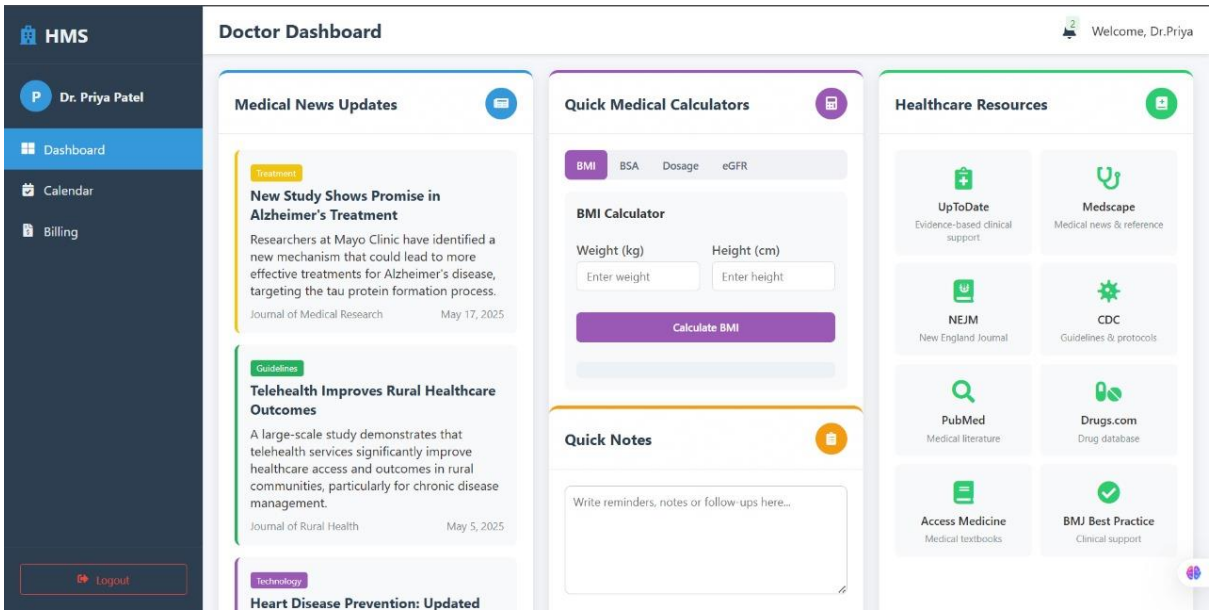
The architecture diagram was designed successfully

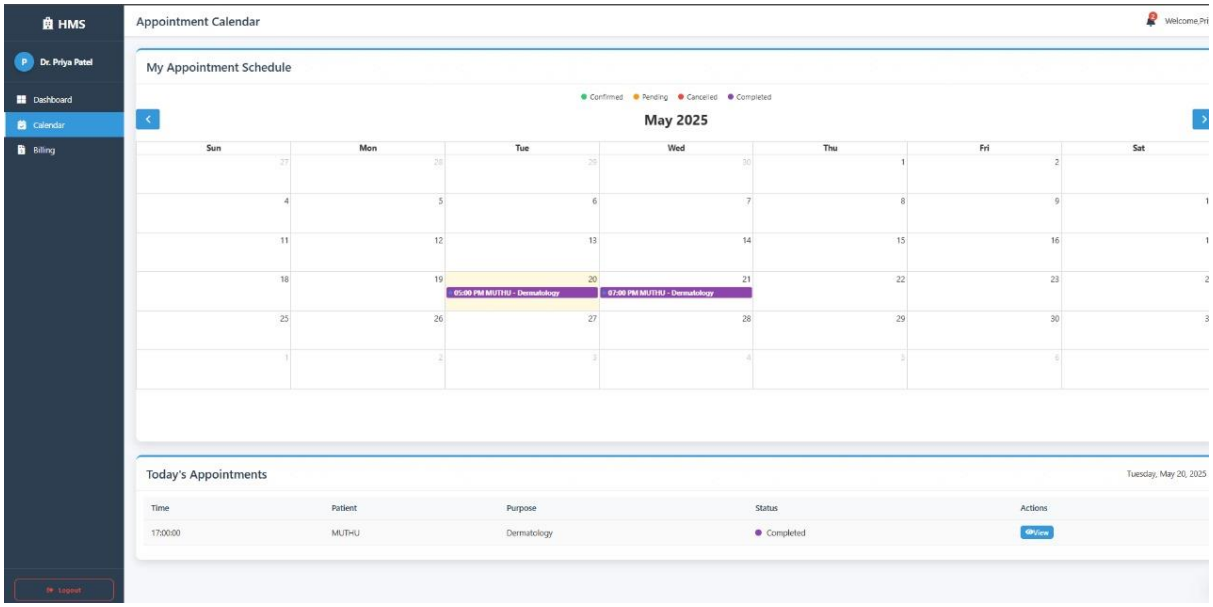
EX NO:10 USER INTERFACE

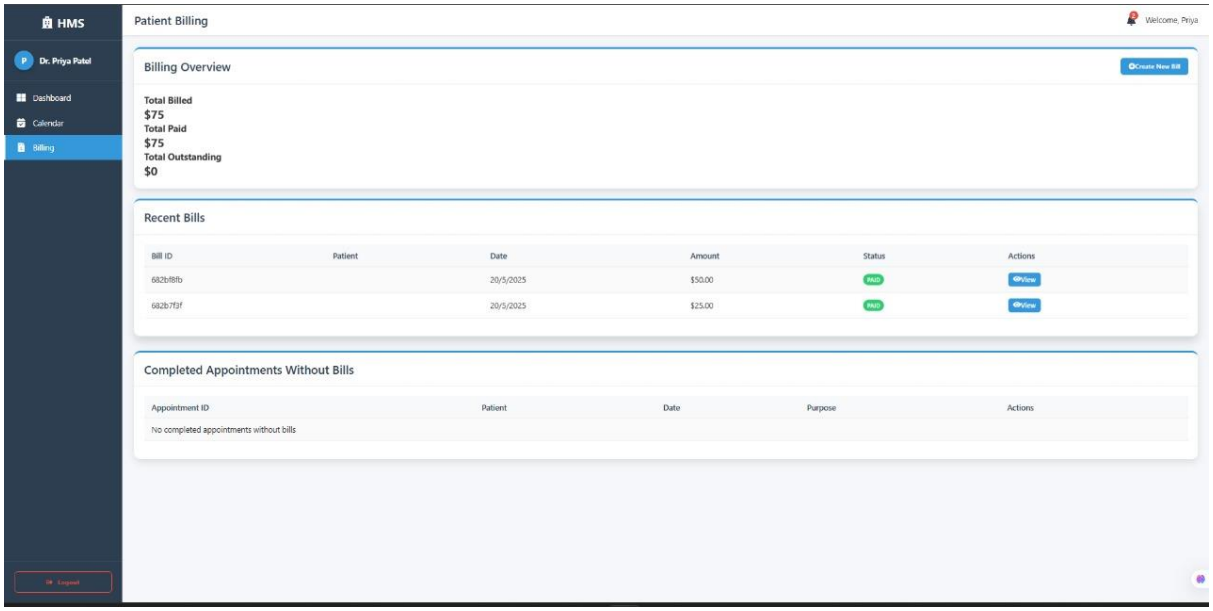
AIM:

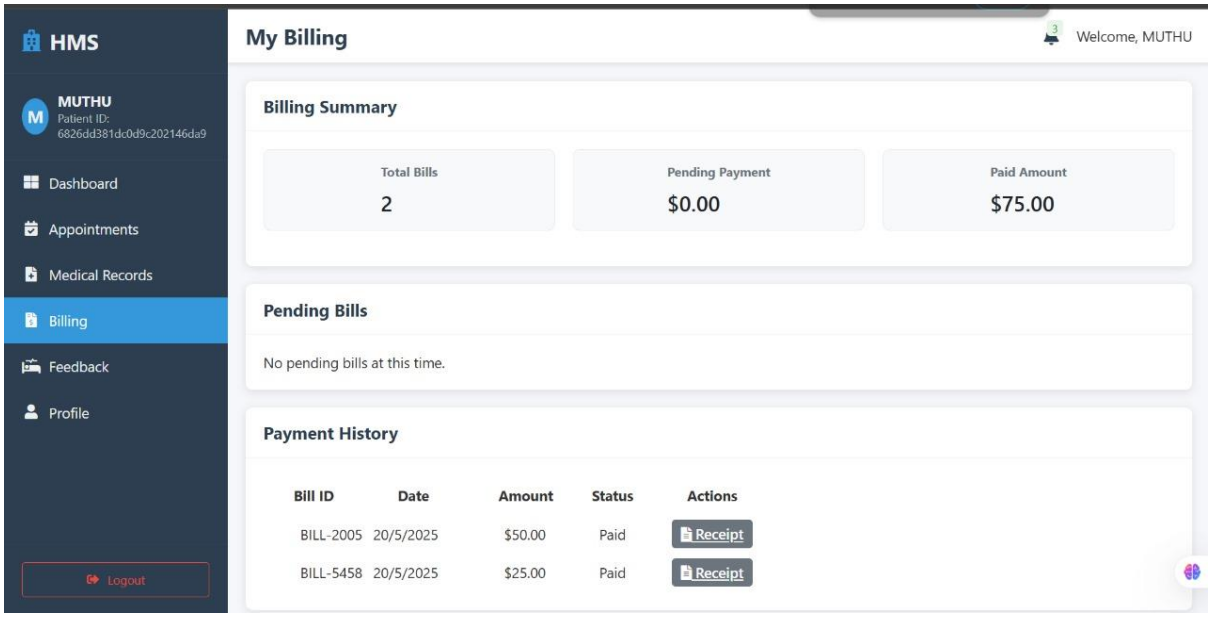
Design User Interface for the HMS Web APP.

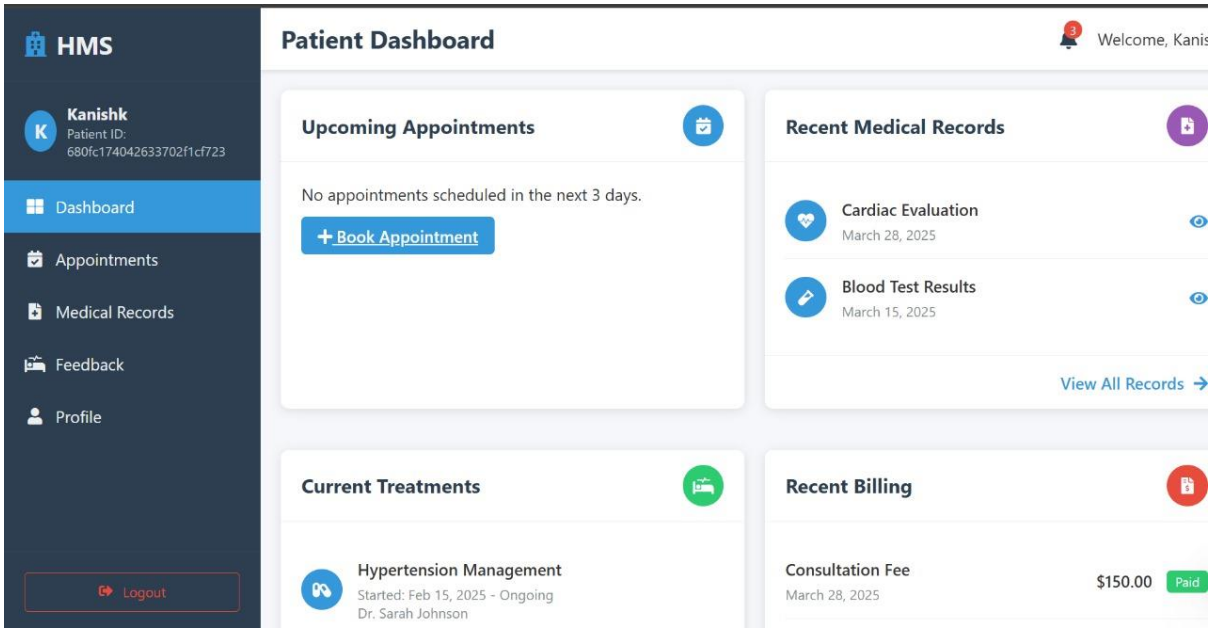
UI DESIGNS OF HMS APP:

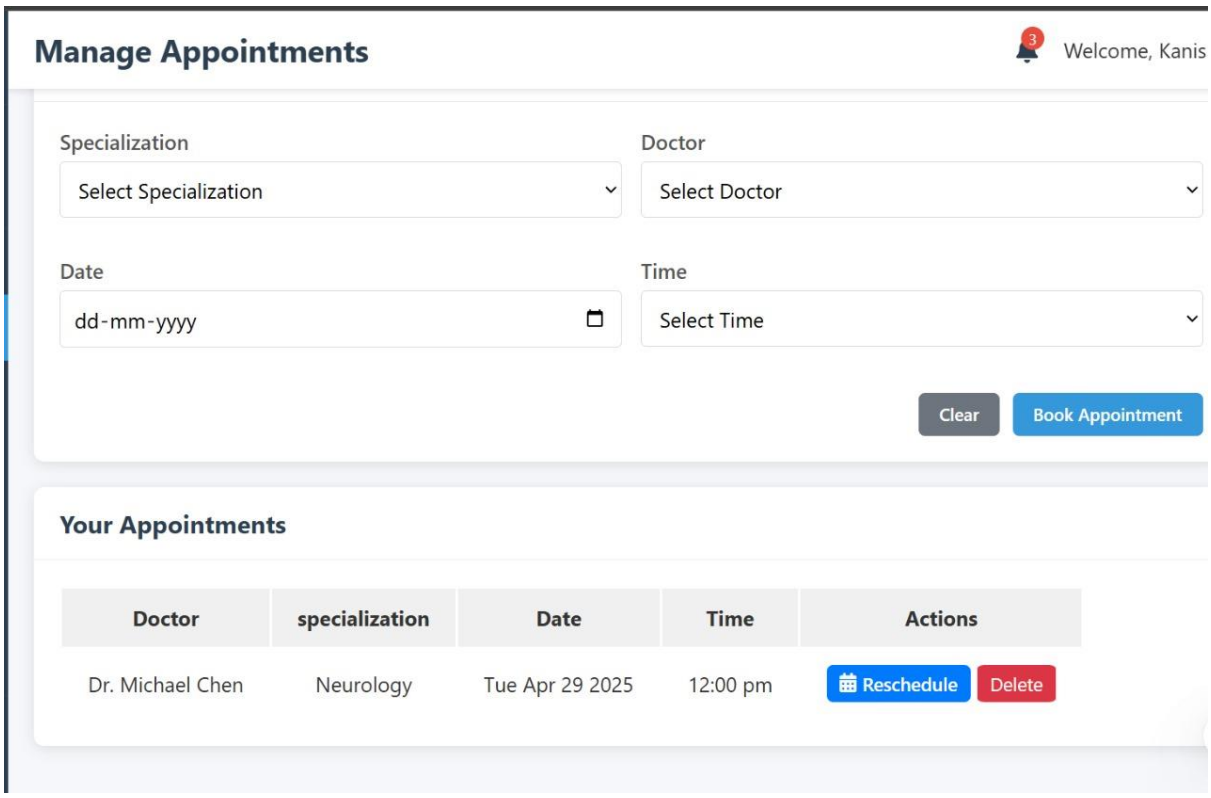


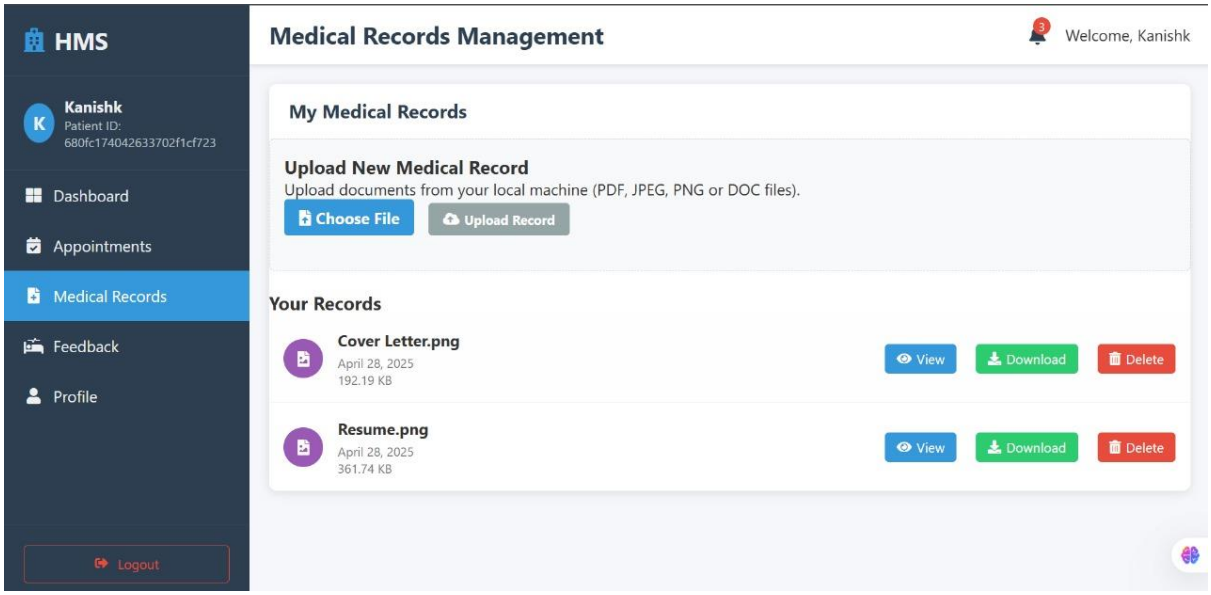


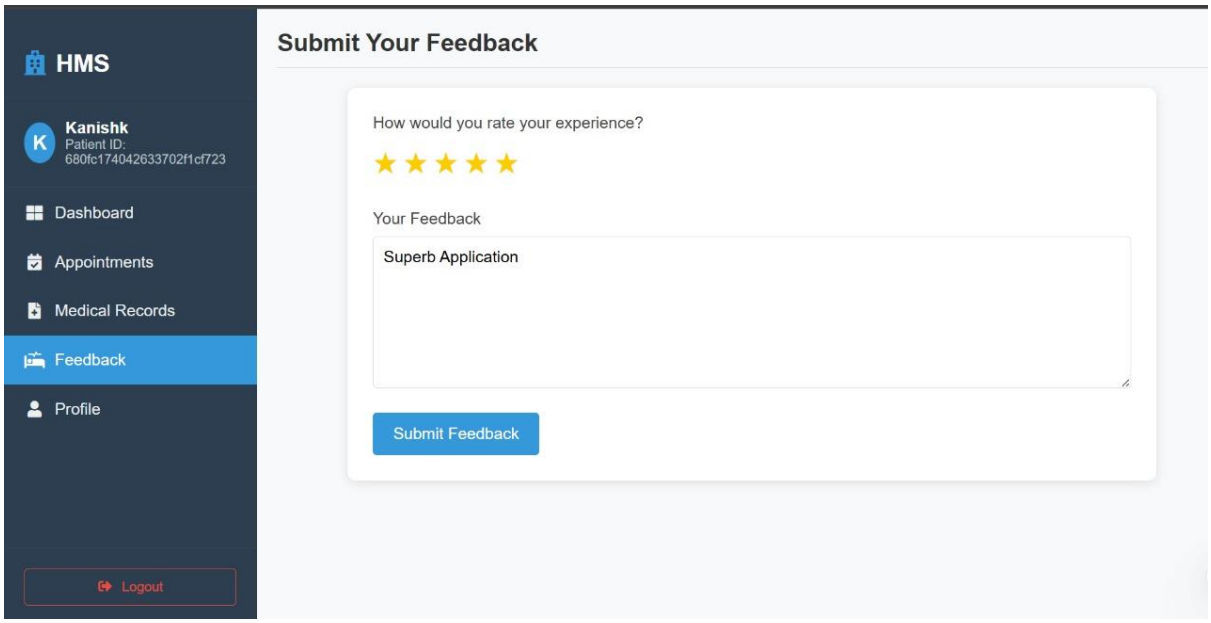

HMS
Patient Billing
Welcome, Priya
Dr. Priya Patel
Dashboard
Calendar
Billing
Billing Overview
Create New Bill
Total Billed
$75
Total Paid
$75
Total Outstanding
$0
Recent Bills
Bill ID
Patient
Date
Amount
Status
Actions
682bf8fb
20/5/2025
$50.00
PAID
View
682b7f3f
20/5/2025
$25.00
PAID
View
Completed Appointments Without Bills
Appointment ID
Patient
Date
Purpose
Actions
No completed appointments without bills
Logout

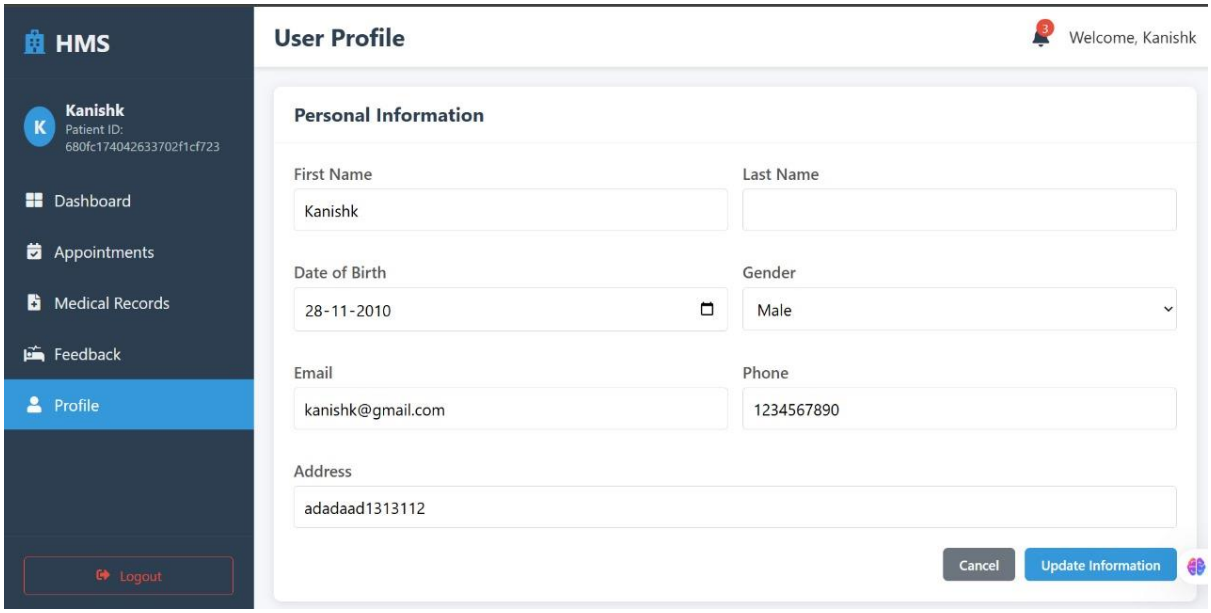

HMS
MUTHU
Patient ID: 6826dd381dc0d9c202146da9
Dashboard
Appointments
Medical Records
Billing
Feedback
Profile
Logout
My Billing
Welcome, MUTHU
Billing Summary
Total Bills
2
Pending Payment
$0.00
Paid Amount
$75.00
Pending Bills
No pending bills at this time.
Payment History
Bill ID
Date
Amount
Status
Actions
BILL-2005
20/5/2025
$50.00
Paid
Receipt
BILL-5458
20/5/2025
$25.00
Paid
Receipt


HMS
Kanishk
Patient ID:
680fc174042633702f1cf723
Dashboard
Appointments
Medical Records
Feedback
Profile
Logout
Patient Dashboard
Welcome, Kanis
Upcoming Appointments
No appointments scheduled in the next 3 days.
+ Book Appointment
Recent Medical Records
Cardiac Evaluation
March 28, 2025
Blood Test Results
March 15, 2025
View All Records
Current Treatments
Hypertension Management
Started: Feb 15, 2025 - Ongoing
Dr. Sarah Johnson
Recent Billing
Consultation Fee
March 28, 2025
$150.00
Paid


Manage Appointments
Welcome, Kanis
Specialization
Select Specialization
Doctor
Select Doctor
Date
dd-mm-yyyy
Time
Select Time
Clear
Book Appointment
Your Appointments
Doctor
specialization
Date
Time
Actions
Dr. Michael Chen
Neurology
Tue Apr 29 2025
12:00 pm
Reschedule
Delete


HMS
Kanishk
Patient ID:
680fc17404263702f1cf723
Dashboard
Appointments
Medical Records
Feedback
Profile
Logout
Medical Records Management
3
Welcome, Kanishk
My Medical Records
Upload New Medical Record
Upload documents from your local machine (PDF, JPEG, PNG or DOC files).
Choose File
Upload Record
Your Records
Cover Letter.png
April 28, 2025
192.19 KB
View
Download
Delete
Resume.png
April 28, 2025
361.74 KB
View
Download
Delete


HMS
Kanishk
Patient ID:
680fc17404263702f1cf723
Dashboard
Appointments
Medical Records
Feedback
Profile
Logout
Submit Your Feedback
How would you rate your experience?
Your Feedback
Superb Application
Submit Feedback



RESULT :

The UI was Designed successfully.

EX NO: 11 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:

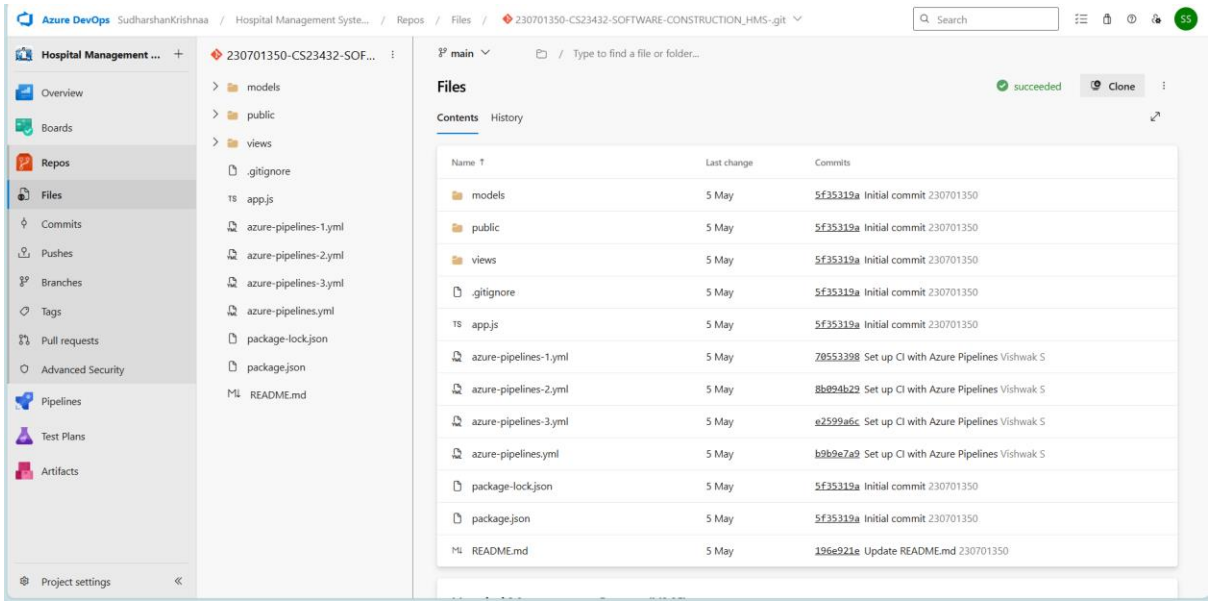
  git add .

  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):



RESULT :

Thus the application was successfully implemented.

EX NO: 12 TESTING USING AZURE

AIM:

To perform testing of the Weather App project using Azure DevOps Test Plans, ensuring that all user stories meet their acceptance criteria as defined in Agile methodology.

PROCEDURE:

Step 1: Open Azure DevOps Project

- Log in to your Azure DevOps account. Open your project.

Step 2: Navigate to Test Plans

- In the left menu, click on Test Plans.
- Click on "New Test Plan" and give it a name (e.g., *Emergency Alerts*).

Step 3: Create Test Suites

- Under the test plan, click "+ New Suite" to add test suites for each Epic or Feature.
  - o Suite 1: *Medical Records*

Step 4: Add Test Cases

- Click on a suite → + New Test Case.
- Enter the following details for each test case:
  - Test Case ID: (e.g., TC001)
  - Title: (e.g., *Receive Notification in Time*)
  - Scenario: Describe the situation being tested.
  - Steps:
    - Launch the app
    - Trigger extreme weather API event
    - Observe time taken for notification

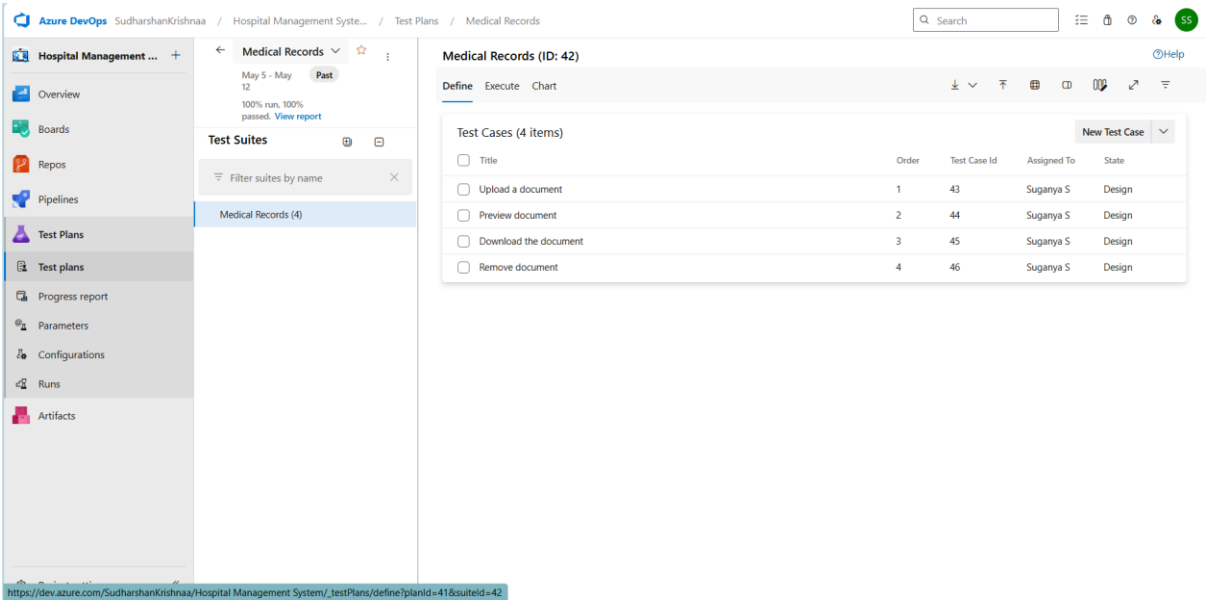
- Expected Result: Notification should appear within 10 seconds.

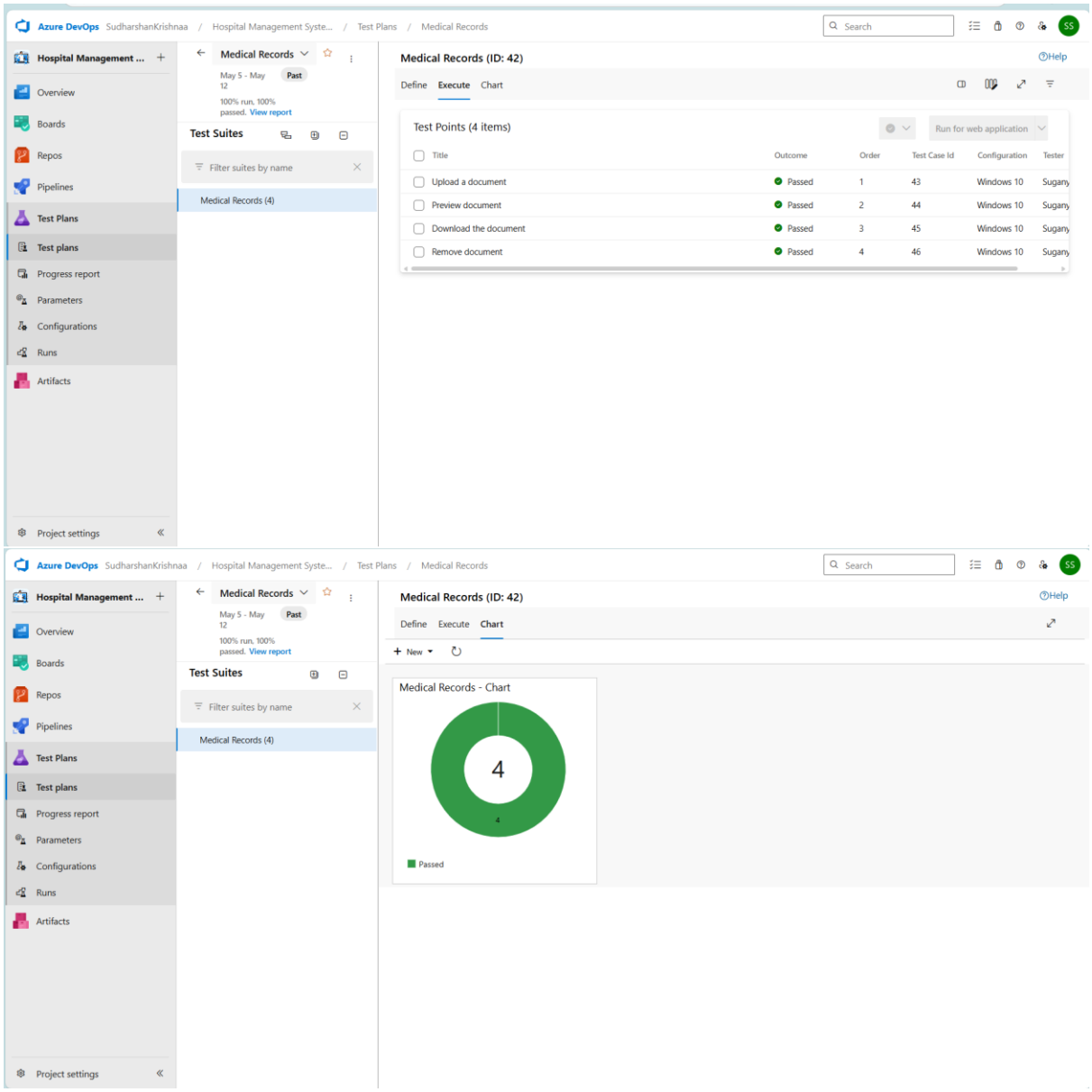
Step 5: Execute Test Cases

- Click on each test case and select Run for web application.
- Follow the steps, mark results as Pass/Fail, and provide Actual Result and Remarks.

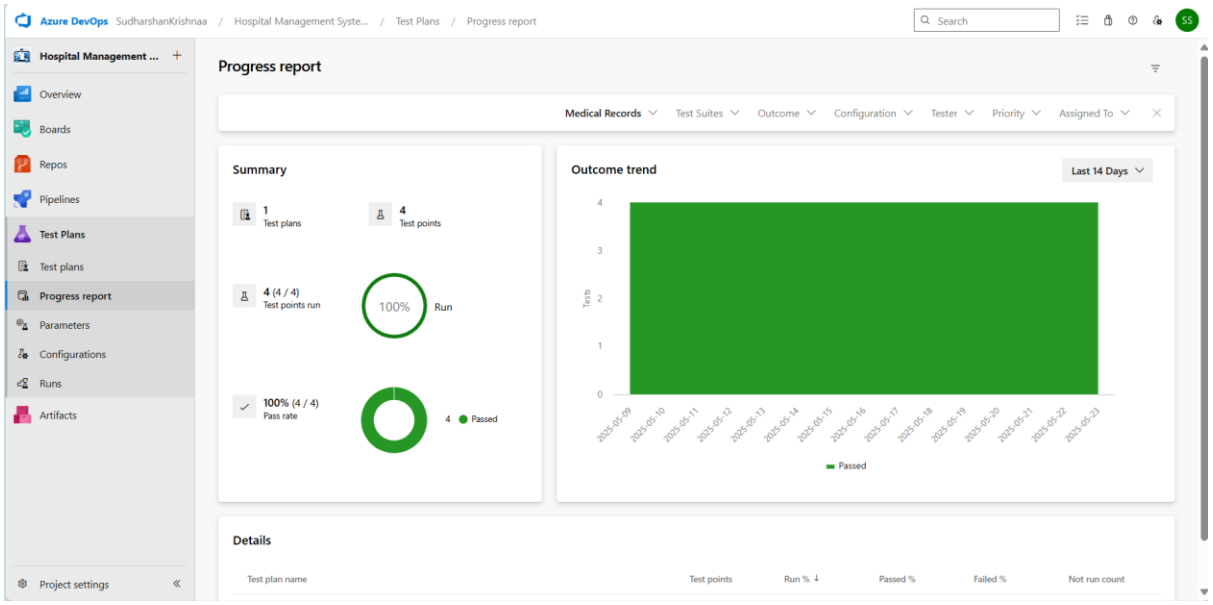
Step 6: Track and Report

- Go to Test Plans → Charts to view test progress.
- Use filters to track:
  - Passed/Failed test cases
  - Test coverage per user story
  - Bugs linked to failed test cases




Azure DevOps
SudharshanKrishnaa / Hospital Management Syste... / Test Plans / Medical Records
Search
Hospital Management ...
Overview
Boards
Repos
Pipelines
Test Plans
Test plans
Progress report
Parameters
Configurations
Runs
Artifacts
Project settings
Medical Records
May 5 - May 12
Past
100% run, 100% passed. View report
Test Suites
Filter suites by name
Medical Records (4)
Medical Records (ID: 42)
Help
Define
Execute
Chart
Test Points (4 items)
Run for web application
Title
Outcome
Order
Test Case Id
Configuration
Tester
Upload a document
Passed
1
43
Windows 10
Sugany
Preview document
Passed
2
44
Windows 10
Sugany
Download the document
Passed
3
45
Windows 10
Sugany
Remove document
Passed
4
46
Windows 10
Sugany

Azure DevOps
SudharshanKrishnaa / Hospital Management Syste... / Test Plans / Medical Records
Search
Hospital Management ...
Overview
Boards
Repos
Pipelines
Test Plans
Test plans
Progress report
Parameters
Configurations
Runs
Artifacts
Project settings
Medical Records
May 5 - May 12
Past
100% run, 100% passed. View report
Test Suites
Filter suites by name
Medical Records (4)
Medical Records (ID: 42)
Help
Define
Execute
Chart
New
Medical Records - Chart
4
Passed



RESULT:

Thus the application was successfully tested in Azure.

EX NO: 13 CI/CD PIPELINE

AIM:

To implement a Continuous Integration and Continuous Deployment (CI/CD) pipeline for the Weather App using Azure DevOps, ensuring automated build, test, and deployment of the application.

PROCEDURE:

Step 1: Create a Build Pipeline (CI)

- Go to Pipelines → Create Pipeline.
- Select Azure Repos Git → Choose your repository.
- Choose Starter pipeline or YAML file.

azure-pipelines-1.yml

```
# Node.js
# Build a general Node.js project with npm.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/javascript

trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- task: NodeTool@0
```

```
 inputs:
   versionSpec: '20.x'
   displayName: 'Install Node.js'
- script: |
   npm install
   npm run build
   displayName: 'npm install and build'
```

azure-pipeline-2.yml

```
# Node.js
# Build a general Node.js project with npm.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/javascript
trigger:
- main
pool:
  name: Default
steps:
- task: NodeTool@0
  inputs:
    versionSpec: '20.x'
  displayName: 'Install Node.js'
- script: |
```

```
    npm install
    npm run build
  displayName: 'npm install and build'
```

azure-pipelines-3.yml

```
  # Node.js
# Build a general Node.js project with npm.
# Add steps that analyze code, save build artifacts, deploy, and more:
# https://docs.microsoft.com/azure/devops/pipelines/languages/javascript
trigger:
- main
pool:
name: Default
steps:
- task: NodeTool@0
 inputs:
versionSpec: '20.x'
displayName: 'Install Node.js'
- script: |
```

npm install

npm run build

displayName: 'npm install and build'

azure-pipelines.yml

 # Node.js

 # Build a general Node.js project with npm.

 # Add steps that analyze code, save build artifacts, deploy, and more:

 # https://docs.microsoft.com/azure/devops/pipelines/languages/javascript

trigger:

- main

pool:

name: Default

steps:

- task: NodeTool@0

inputs:

versionSpec: '20.x'

displayName: 'Install Node.js'

- script: |

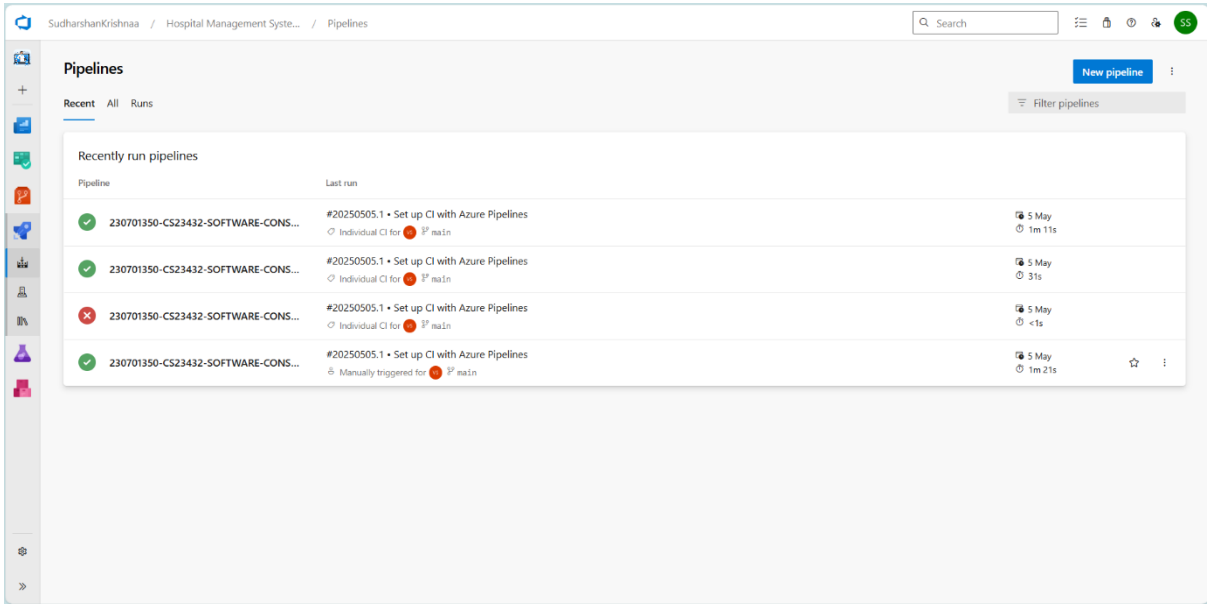
npm install

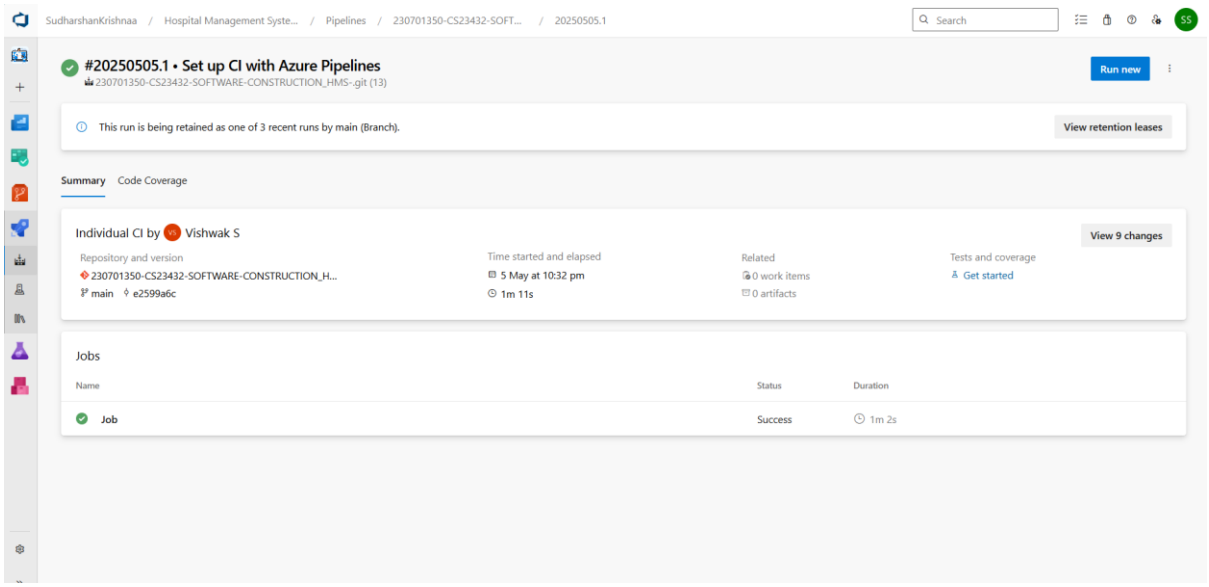
npm run build

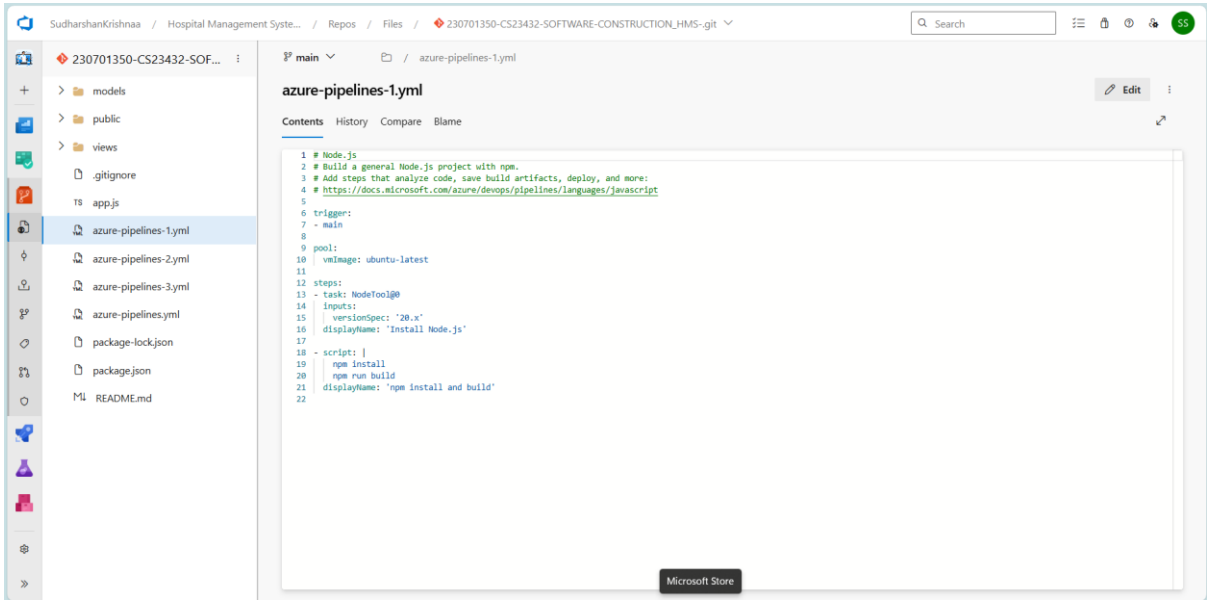
displayName: 'npm install and build'

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 CI/CD pipeline has been successfully implemented