RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF COMPUTER SCIENCE &

ENGINEERING ACADEMIC YEAR 2024-2025

EVEN SEMESTER

CS23432 SOFTWARE ENGINEERING LAB

LAB MANUAL

SECOND YEAR

FOURTH SEMESTER

2024- 2025

EVEN SEMESTER

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 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

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		
10		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:

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 was successfully .

EX NO: 2

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

Online Course Portal System

In the rapidly evolving educational landscape, there is a growing demand for flexible and accessible learning platforms that can support self-paced learning, instructor-led sessions, and continuous skill development.Traditional classroom-based methods are no longer sufficient to meet the dynamic needs of learners and educators in the digital era.

The **Online Course Portal System** is designed to provide a centralized platform where students can access a variety of online courses, view course content, take quizzes, and track their progress. Instructors can upload course materials, manage assessments, and interact with students through forums or announcements. Administrators can oversee user registrations, course offerings, and platform analytics.

This system aims to bridge the gap between educators and learners by offering a user-friendly interface, efficient content management, and performance tracking. By digitizing the learning environment, the portal supports personalized learning paths and ensures that users have 24/7 access to educational resources, thus enhancing the overall quality and reach of education.

RESULT:

The problem statement was 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 STORIES

AIM:

To create User Stories

THEORY

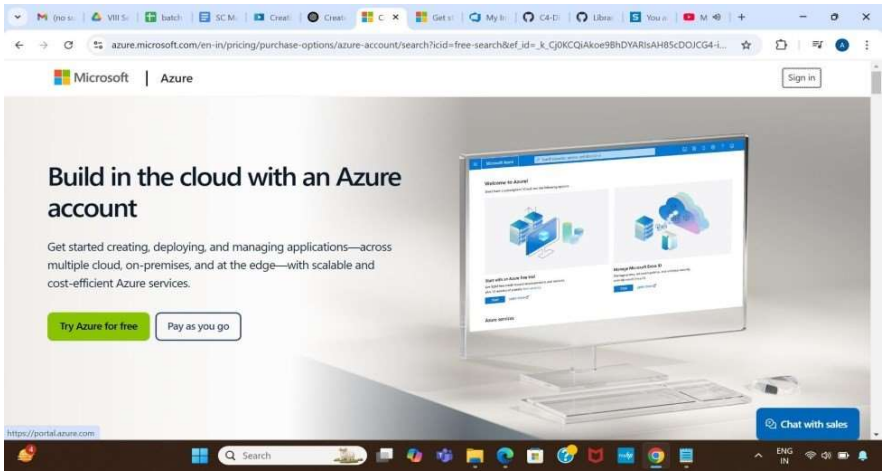
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

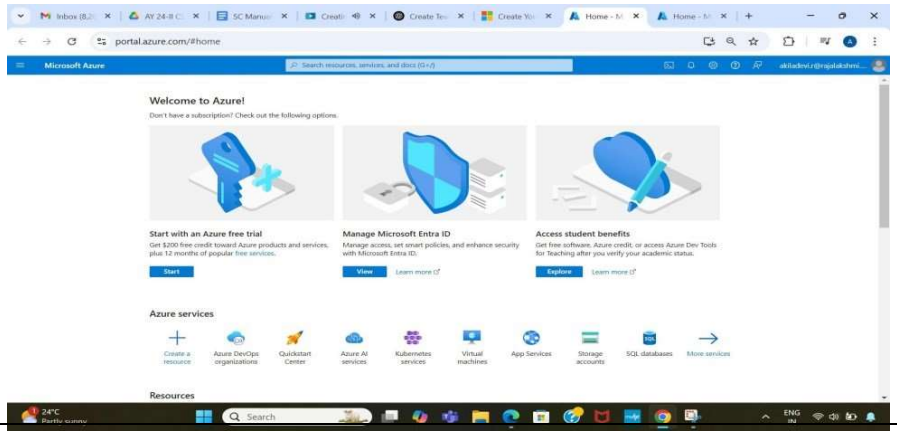
"As a [role], I [want to], [so that]."

PROCEDURE:

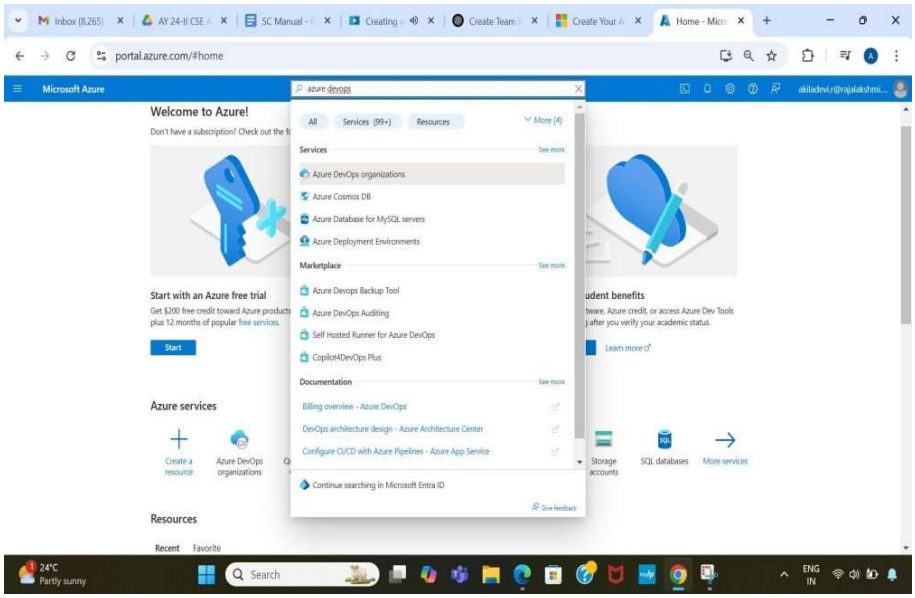
1. Open your web browser and go to the Azure website: *https://azure.microsoft.com/en-in* Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for *https://signup.live.com/?lic=1*



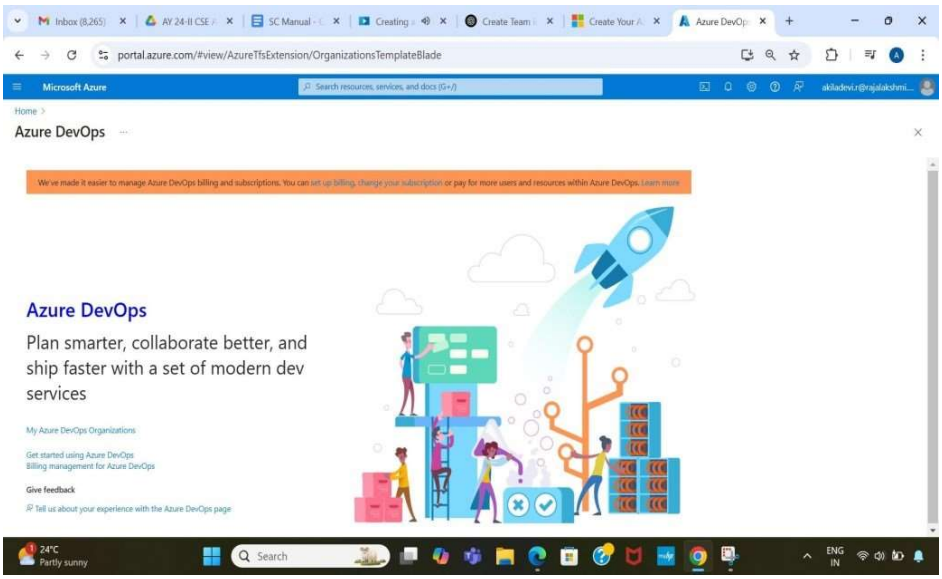
3. Azure home page

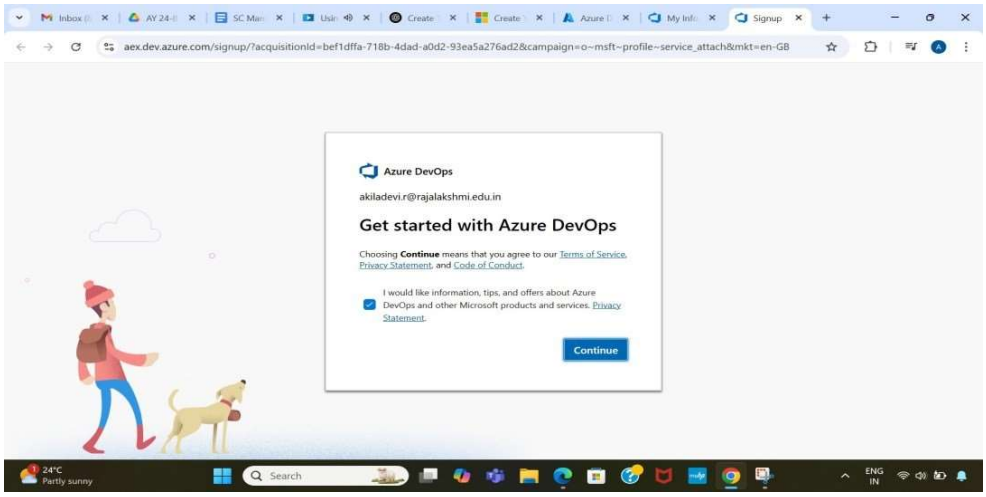


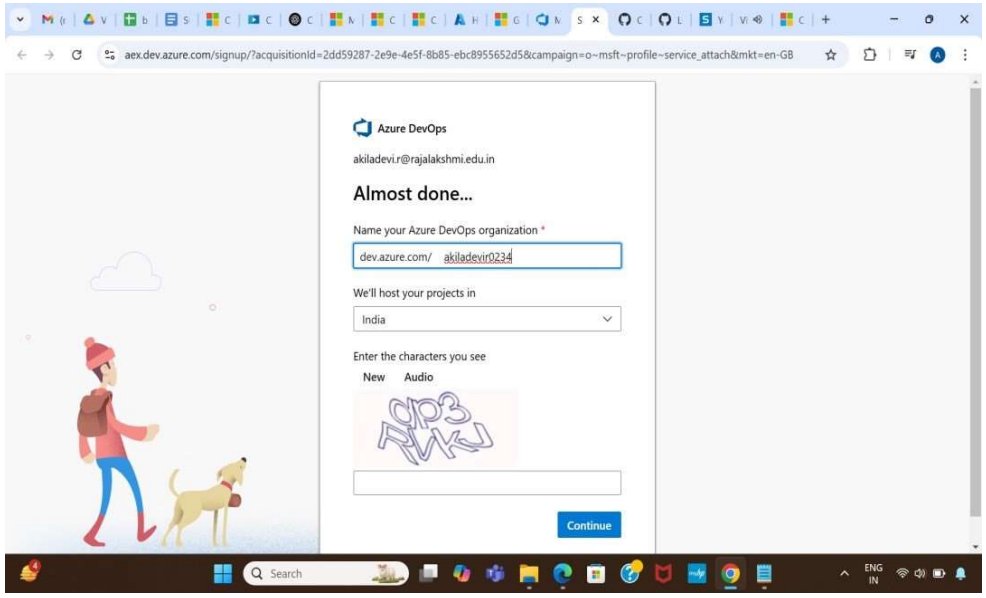
4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



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

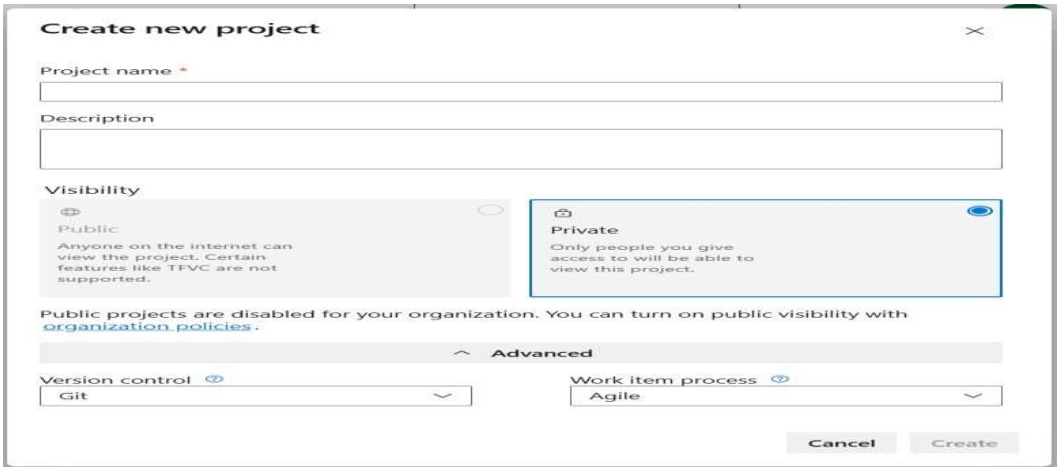
After the organization is set up, you'll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.

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

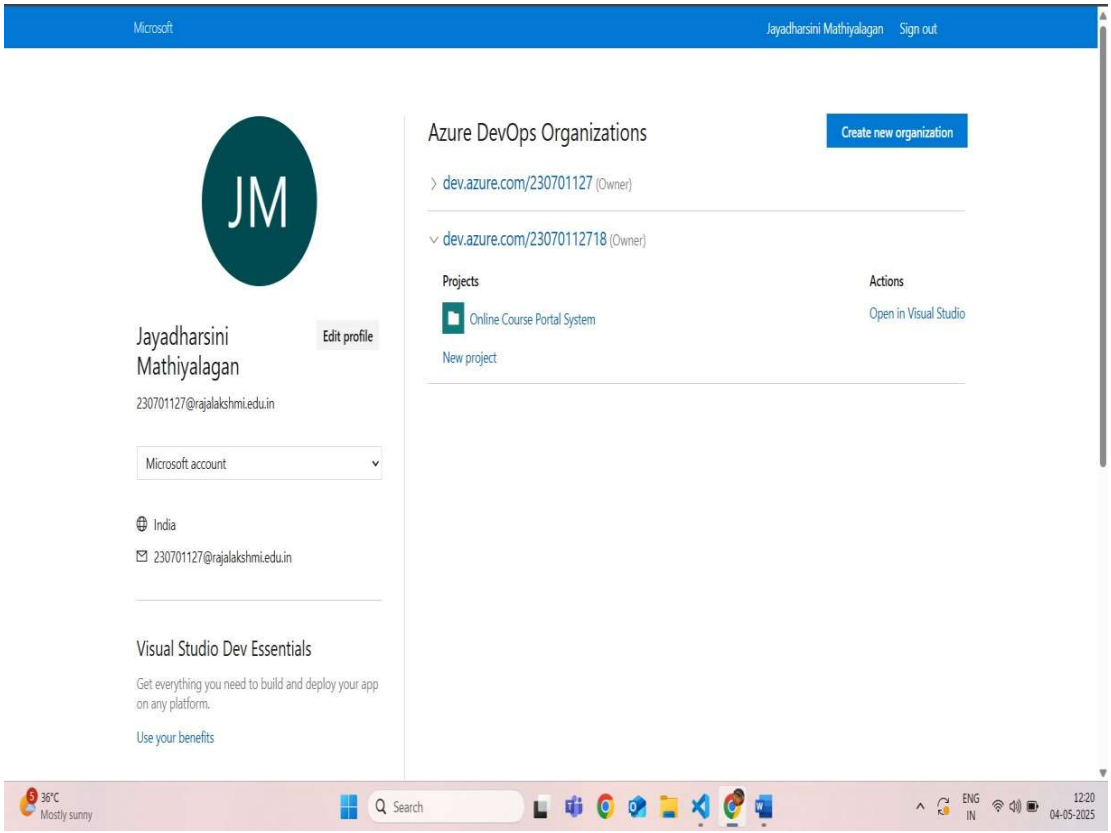
ii. Enter the project name, description, and visibility options:

- **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).

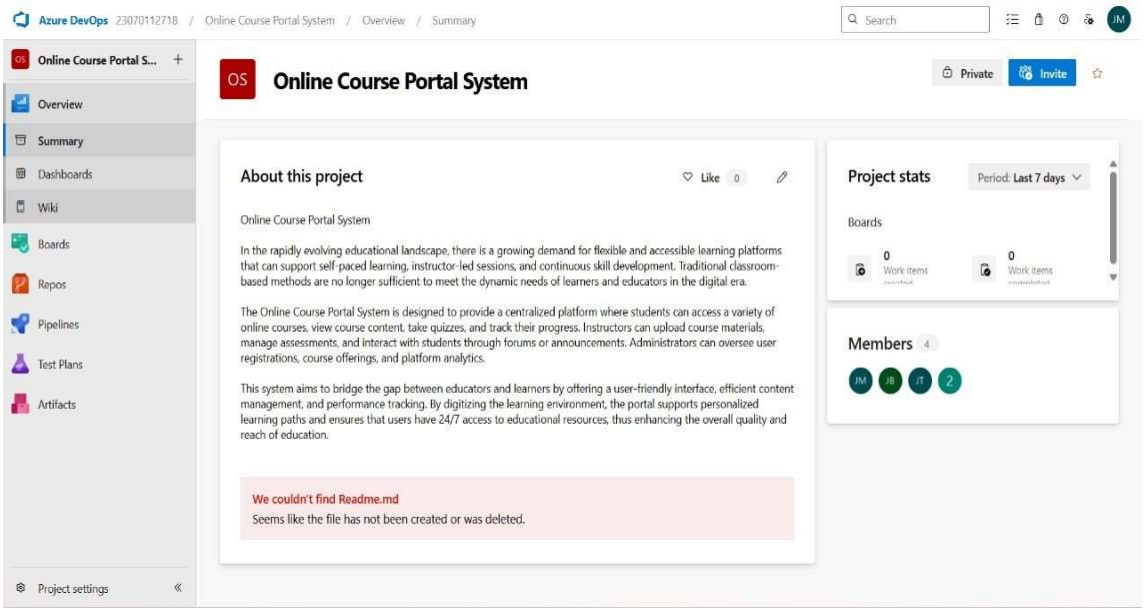
- Once you've filled out the details, click **Create** to set up your first project.



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



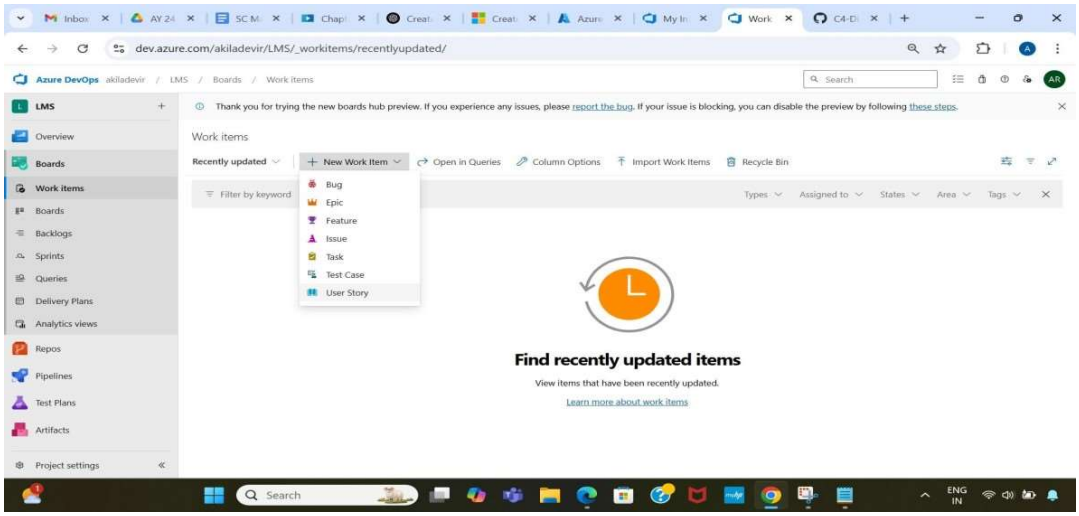
8. Project dashboard



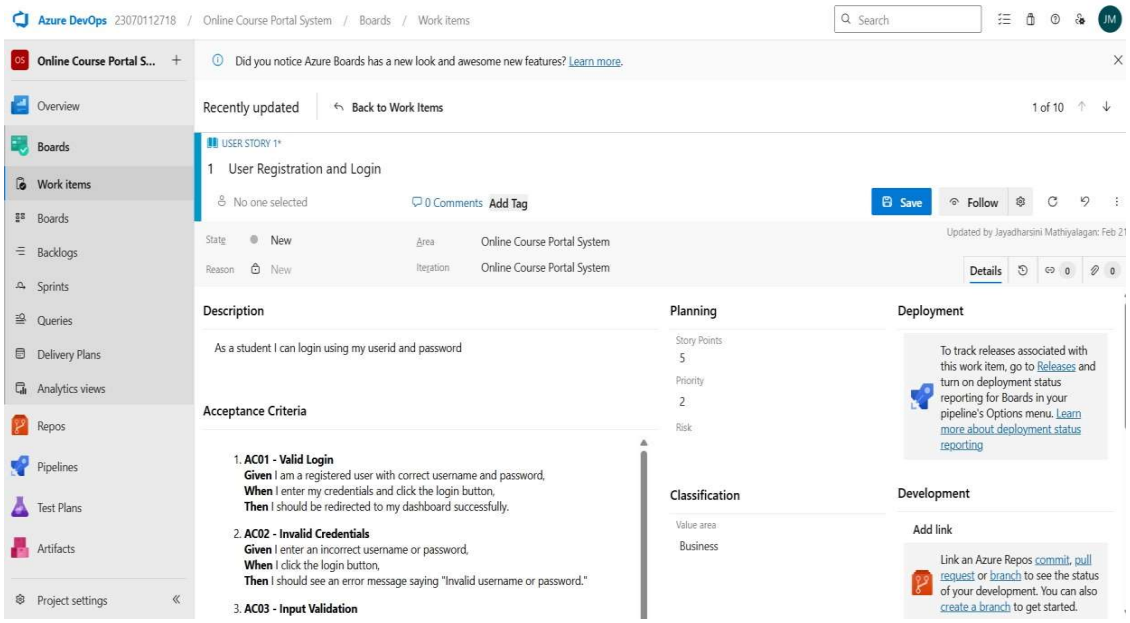
9.To manage user stories

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

b. On the **work items** page, you'll see the option to **Add a work item** at the top. Alternatively, you can find a + button or **Add New Work Item** depending on the view you're in. From the **Add a work item** dropdown, select **User Story**. This will open a form to enter details for the new User Story.



10.Fill in User Story Details



RESULT :

The user story was written successfully.

EX.NO:5

SEQUENCE DIAGRAM

Aim:

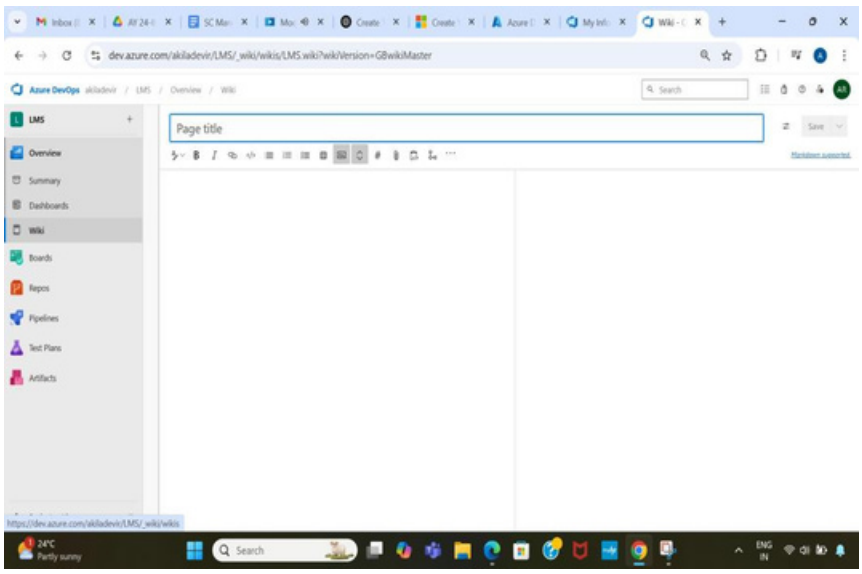
To design a Sequence Diagram by using Mermaid.js

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

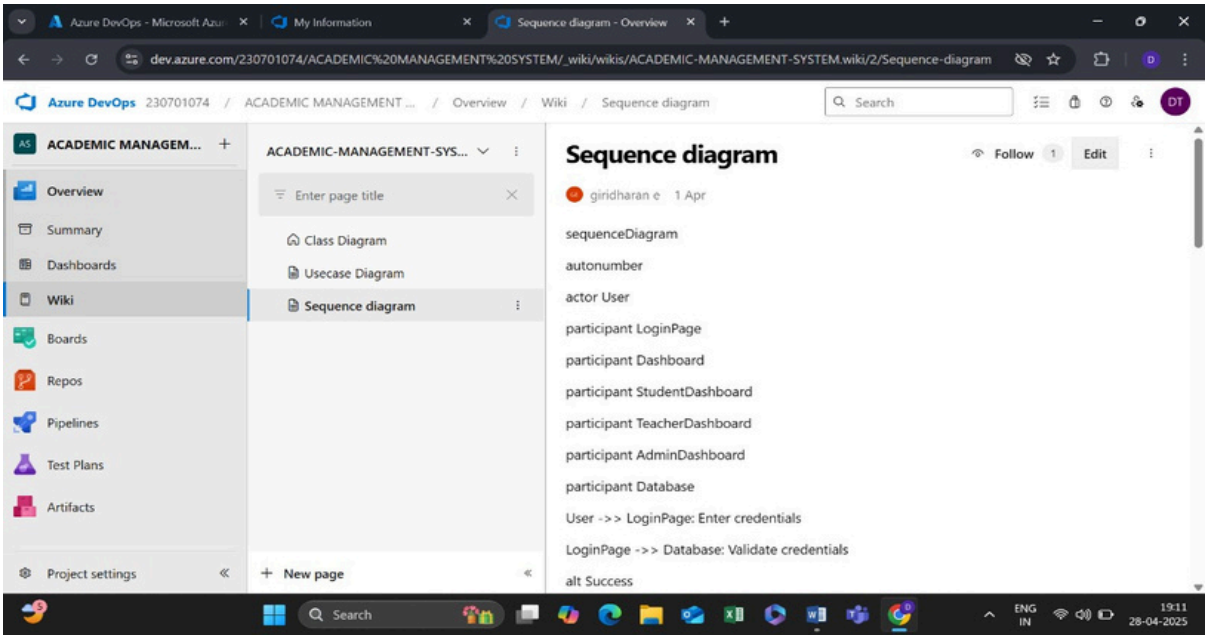
```
sequenceDiagram
    participant Student
    participant System
    participant Instructor
    %% Student Flow
    Student->>System: Login with username and password
    System-->>System: Validate credentials

    alt New User

        System-->>Student: [New User]
        Student->>System: Enter area of interest and current status
        System-->>Student: Recommend suitable courses
```

```
else Old User
  System-->>Student: [Old User]
  Student->>System: Profile → My Courses
  System-->>Student: Display enrolled courses
  Student->>System: Select enrolled course
  System-->>Student: Display progress or completion status
  System-->>Student: Display pending modules (Quizzes, Assignments, Badges)
  Note right of System: 75% - 5 courses – Bronze (5% off) 80% -
  15 courses – Silver (15% off) 85% - 25 courses – Gold (25% off)
  end

System-->>Student: Assessments contribute 50% of final score
System-->>Student: Live doubt clearing sessions held weekly
System-->>Student: Video doubt solutions uploaded throughout the week
System-->>Student: Reminders, Feedback, Recommendations, Instructor & Payment details
%% Instructor Flow
Instructor->>System: Login with User ID and password
System-->>System: Validate credentials
System-->>Instructor: Access instructor features
System-->>Instructor: Display instructor details & bank account
Instructor->>System: Upload video
Instructor->>System: Prepare assignment questions and answers
Instructor->>System: Monitor student doubts
Instructor->>System: Conduct live sessions (weekly)
Instructor->>System: Manage courses
System-->>Student: Display course recommendations, Duration, Fee, Rating, Instructor info,
Syllabus/modules, Filters
```

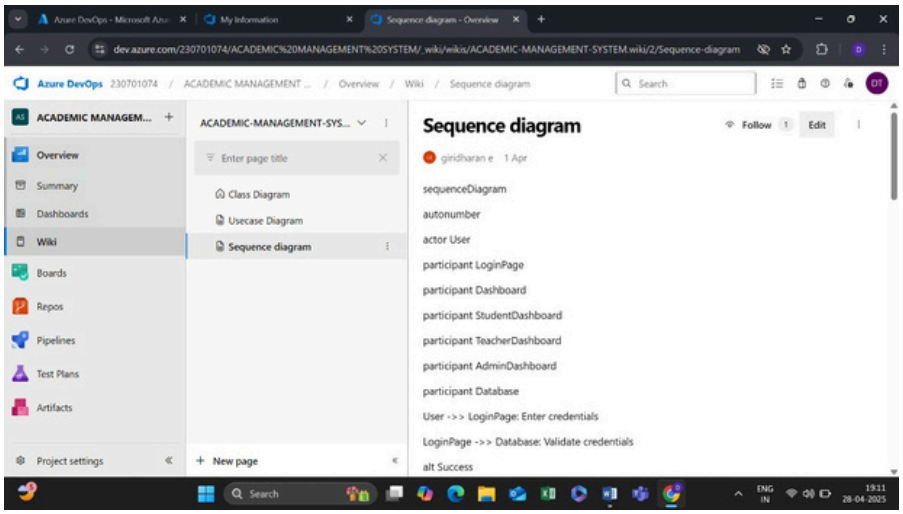
Explanation:

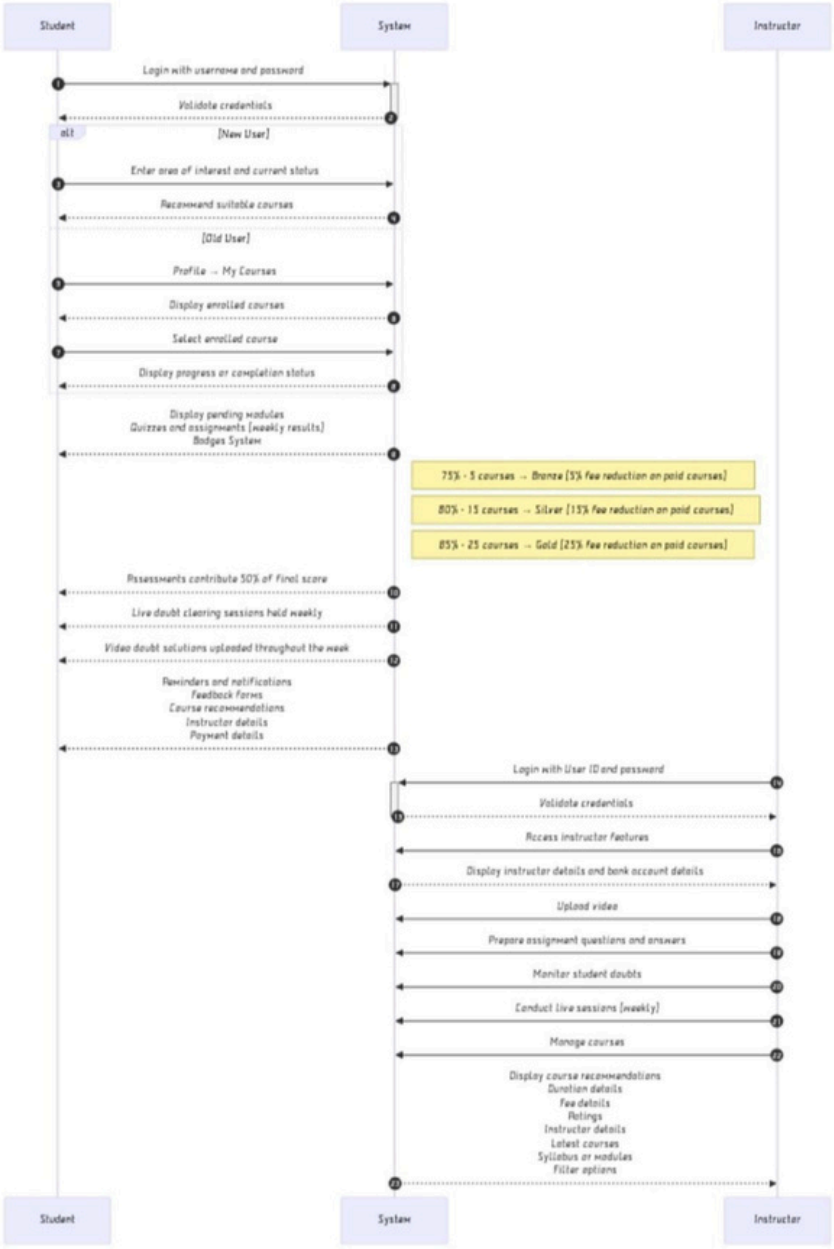
participant defines the entities involved.
->> represents a direct message.
-->> represents a response message.
+ after ->> activates a participant.
- after -->> deactivates a participant.
alt / else for conditional flows.
loop can be used for repeated actions.

->	Solid line without arrow
-->	Dotted line without arrow
->>	Solid line with arrowhead
-->>	Dotted line with arrowhead
<<->>	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)



4.click wiki menu and select the page





Result:

The sequence diagram was drawn successfully.

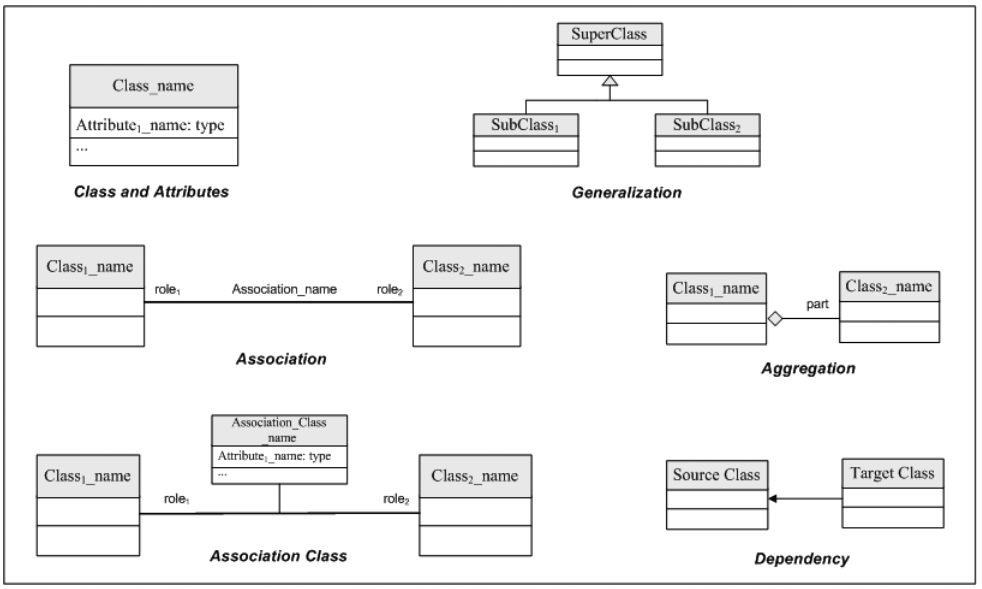
EX NO. 8 **CLASS DIAGRAM**

AIM :-

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

THEORY

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

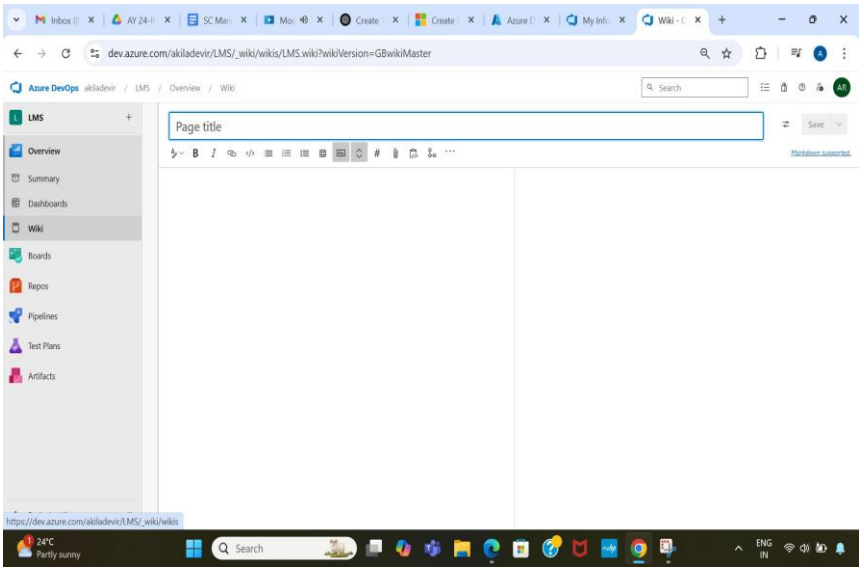


Notations in class diagram

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

```
classDiagram

%% Base Class
class User {
  - userID: String
  - password: String
  + login(): void
  + logout(): void
}

%% Inherited Classes
class Student {
  - areaOfInterest: String
  - currentStatus: String
  + enrolCourse(): void
  + viewProgress(): void
  + attemptQuiz(): void
  + submitAssignment(): void
  + viewBadges(): void
}

class Instructor {
  - courseDetails: String
  + uploadVideo(): void
  + createAssignment(): void
  + conductLiveSession(): void
  + monitorDoubts(): void
}

%% Related Classes
class Course {
  - courseID: String
  - courseName: String
  - duration: String
  - fee: Float
  - rating: Float
  - syllabus: String
  + getDetails(): void
  + viewModules(): void
  + recommendCourse(): void
}

class Module {
  - moduleID: String
  - moduleName: String
  - content: String
  + viewModule(): void
  + viewPendingModules(): void
}
```

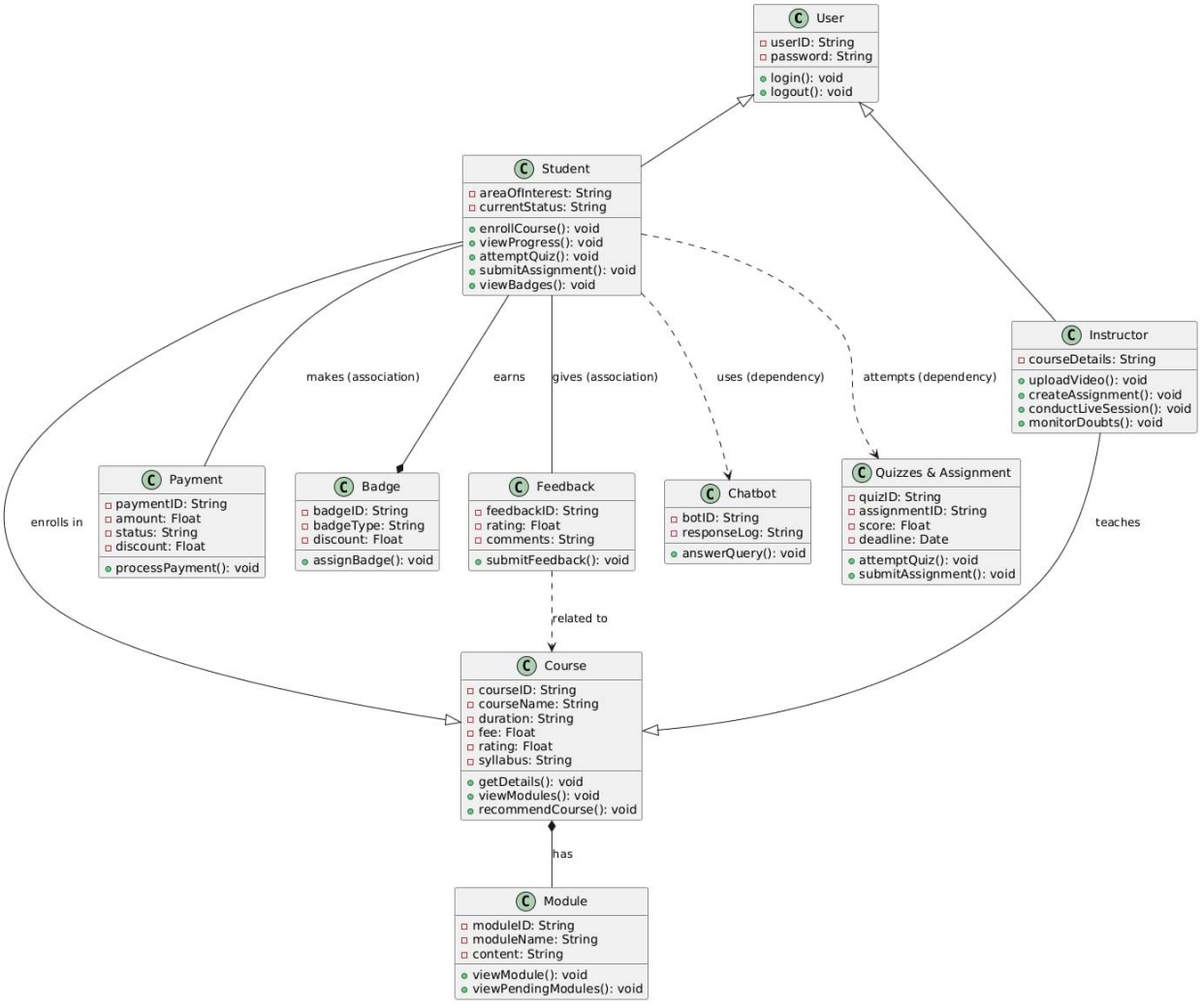
```
class QuizzesAssignment {
  - quizID: String
  - assignmentID: String
  - score: Float
  - deadline: Date
  + attemptQuiz(): void
  + submitAssignment(): void
}

class Payment {
  - paymentID: String
  - amount: Float
  - status: String
  - discount: Float
  + processPayment(): void
}

class Badge {
  - badgeID: String
  - badgeType: String
  - discount: Float
  + assignBadge(): void
}

class Feedback {
  - feedbackID: String
  - rating: Float
  - comments: String
  + submitFeedback(): void
}

class Chatbot {
  - botID: String
  - responseLog: String
  + answerQuery(): void
}

%% Inheritance
Student --|> User
Instructor --|> User

%% Associations and Dependencies
Student --> Course : enrolls
Student --> Payment : makes
Student --> Badge : earns
Student --> Feedback : gives
Student ..> Chatbot : uses
Student ..> QuizzesAssignment : attempts

Instructor --> Course : teaches
Instructor ..> QuizzesAssignment : prepares
```

Feedback --> Course : related to
Course --> Module : has

Relationship Types

Type	Description
<|	Inheritance
*	Composition
o	Aggregation
>	Association
<	Association
|>	Realization

Visit : https://mermaid.js.org/syntax/classDiagram.html



Result:

The use case diagram was designed successfully.

EX NO: 7

USECASE DIAGRAM

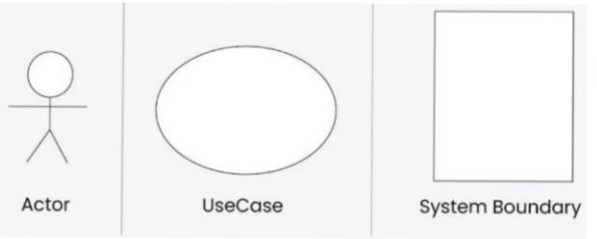
AIM:

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

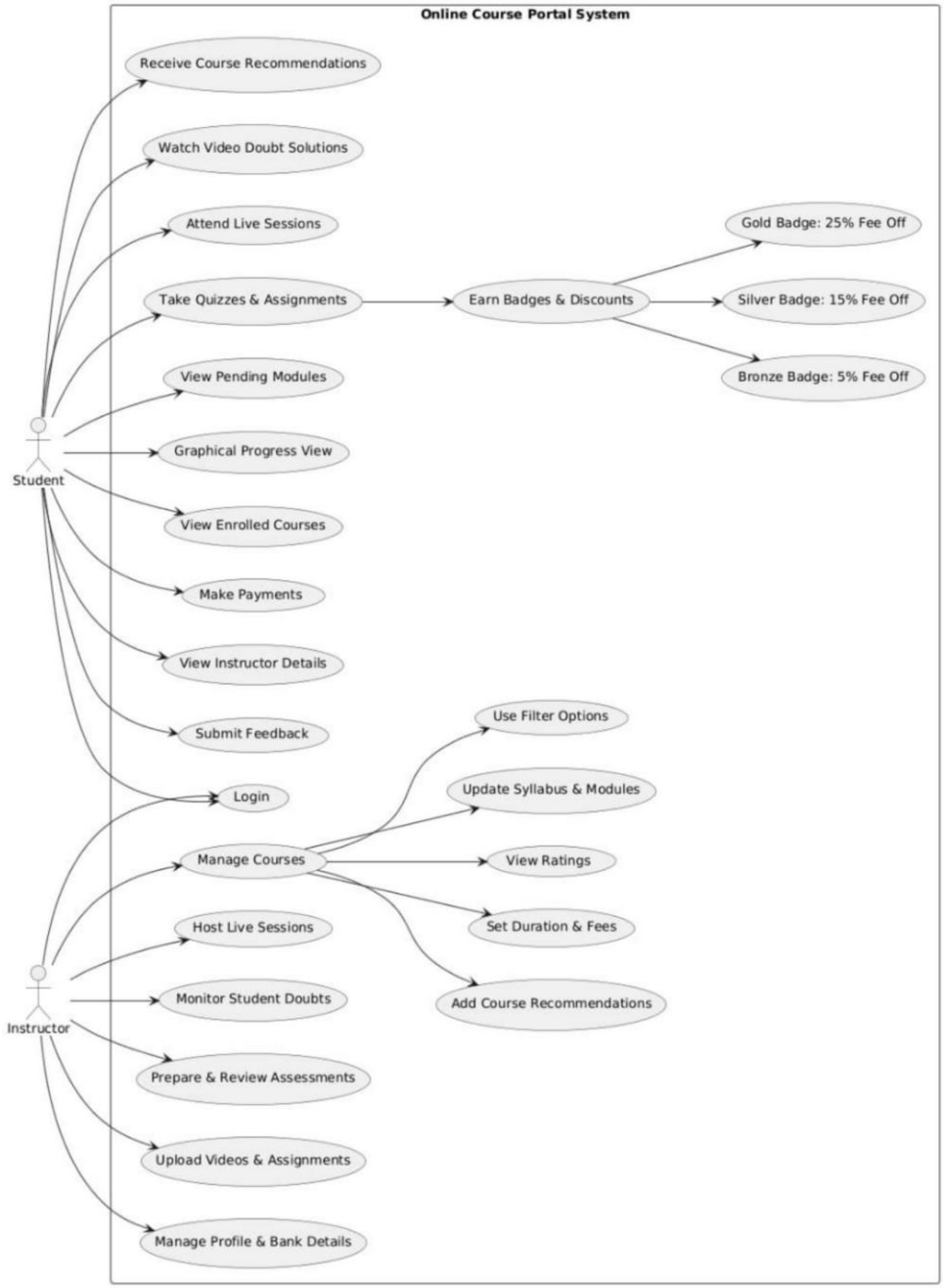
Option 1: Add to Azure DevOps Wiki

- 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 Case Diagram](attachments/use_case_diagram.png)

Option 2: Attach to Work Items in Azure Boards

- Open Azure DevOps → Navigate to Boards (Project > Boards).
- Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram.
- Add comments or descriptions to explain the use case.

USE CASE DIAGRAM – ONLINE COURSE PORTAL SYSTEM



RESULT:

The use case diagram was designed successfully

EX NO. 8

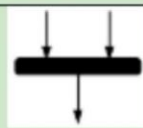
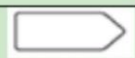
ACTIVITY DIAGRAM

AIM :-

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

THEORY:-

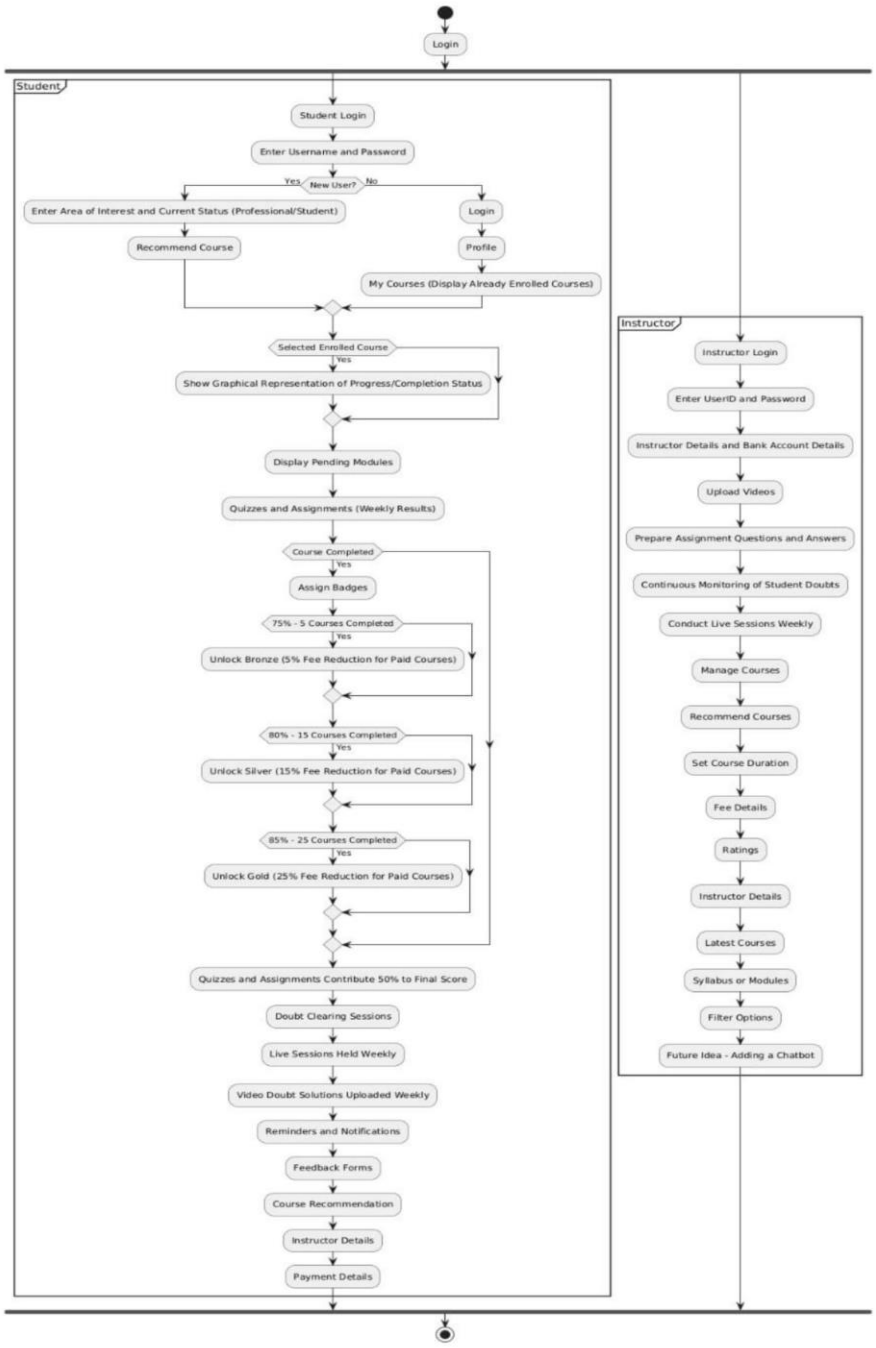
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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

ACTIVITY DIAGRAM – ONLINE COURSE PORTAL SYSTEM



Result:
The activity diagram was designed successfully

EX NO. 11

ARCHITECTURE DIAGRAM

Aim:

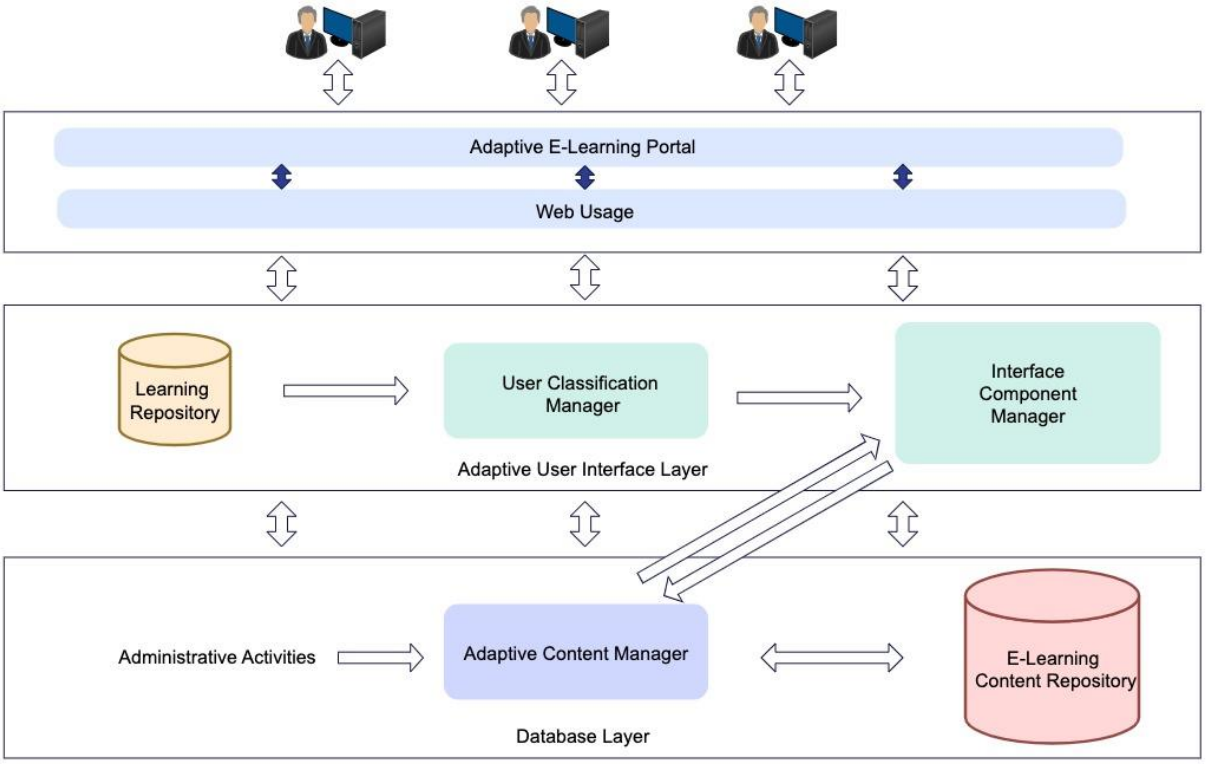
Steps to draw the Architecture Diagram using draw.io.

Theory:

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

Procedure

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



Result:

The architecture diagram was designed successfully

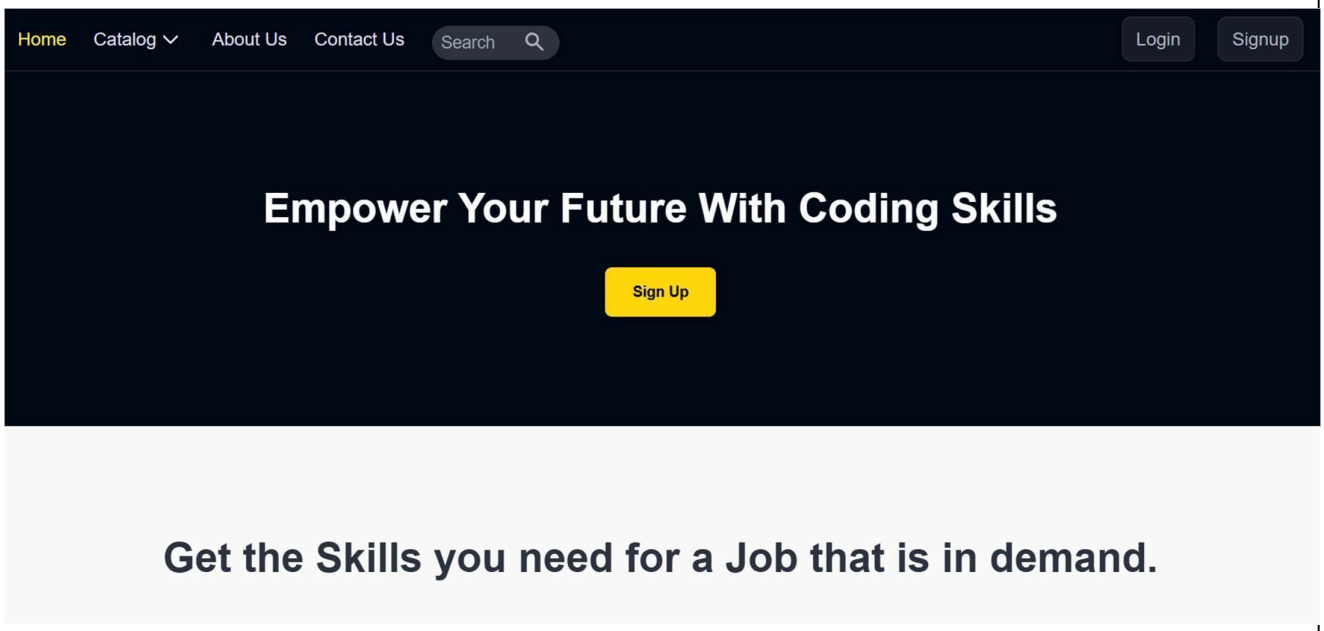
EX NO. 10

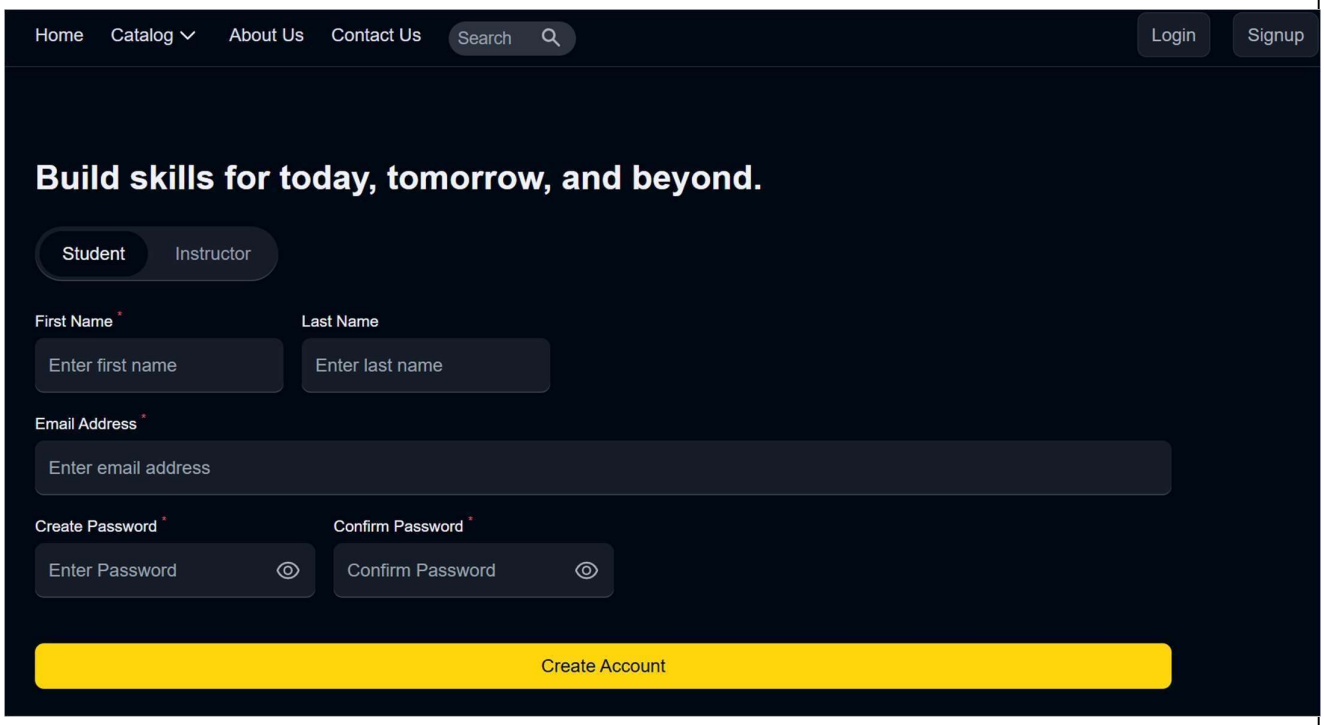
USER INTERFACE

Aim:

Design User Interface for the given project

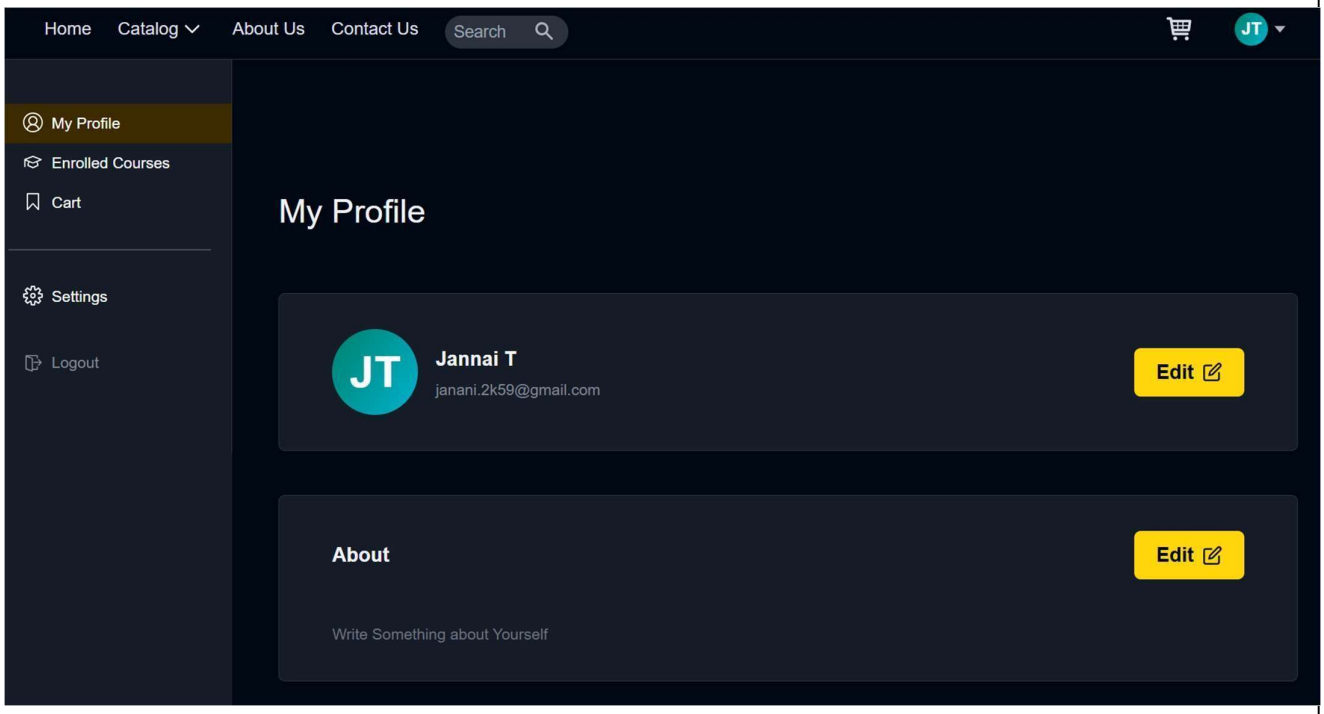
1] Login page



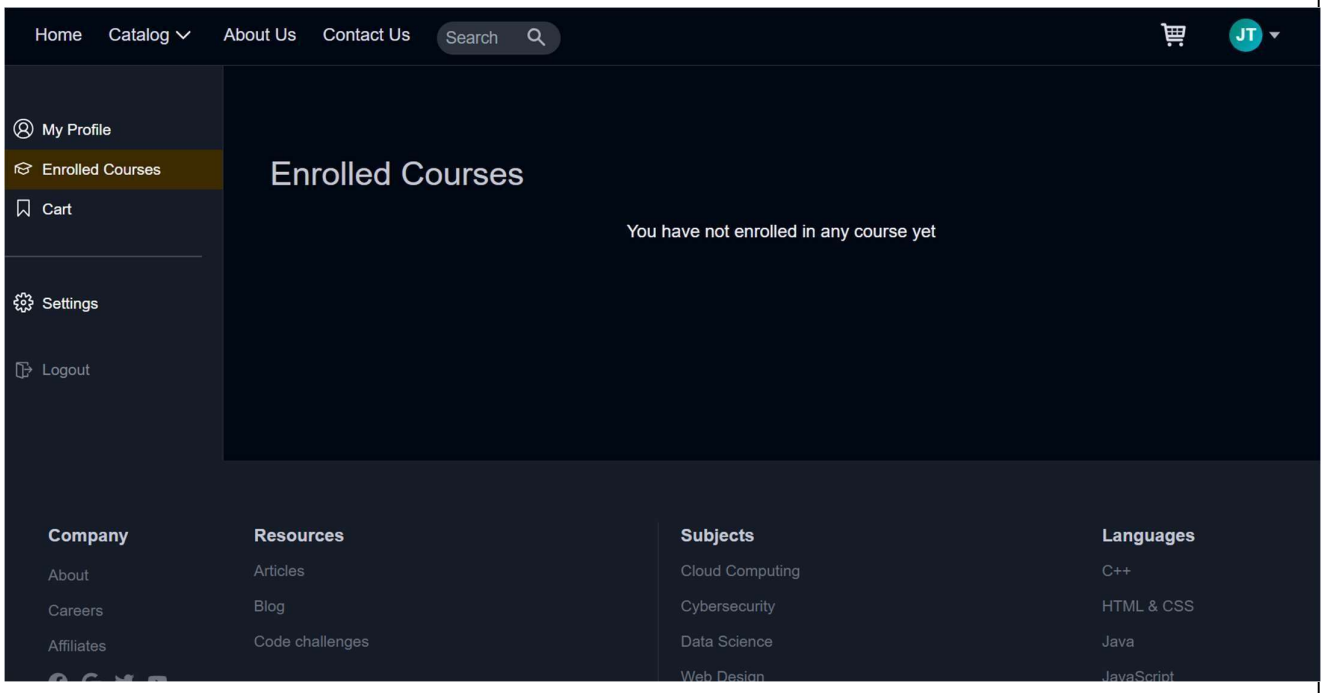


2] Student Login

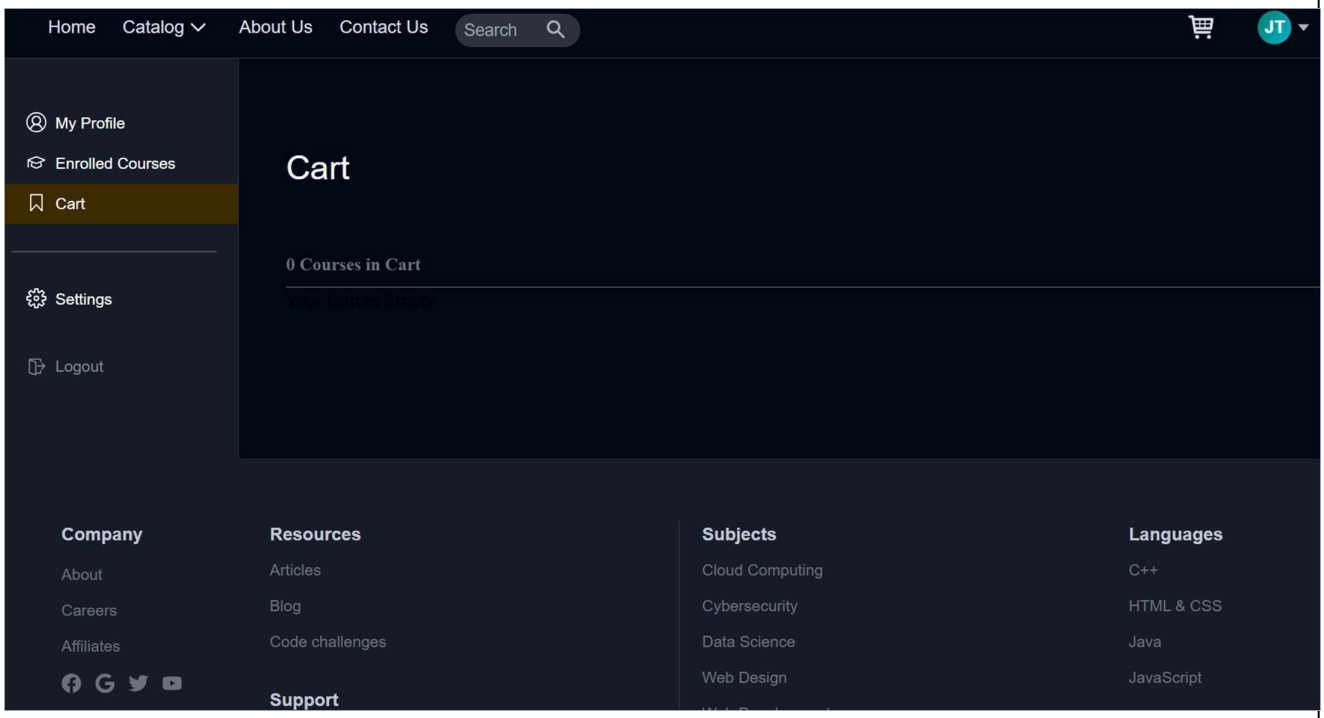
- My Profile



- Enrolled Courses

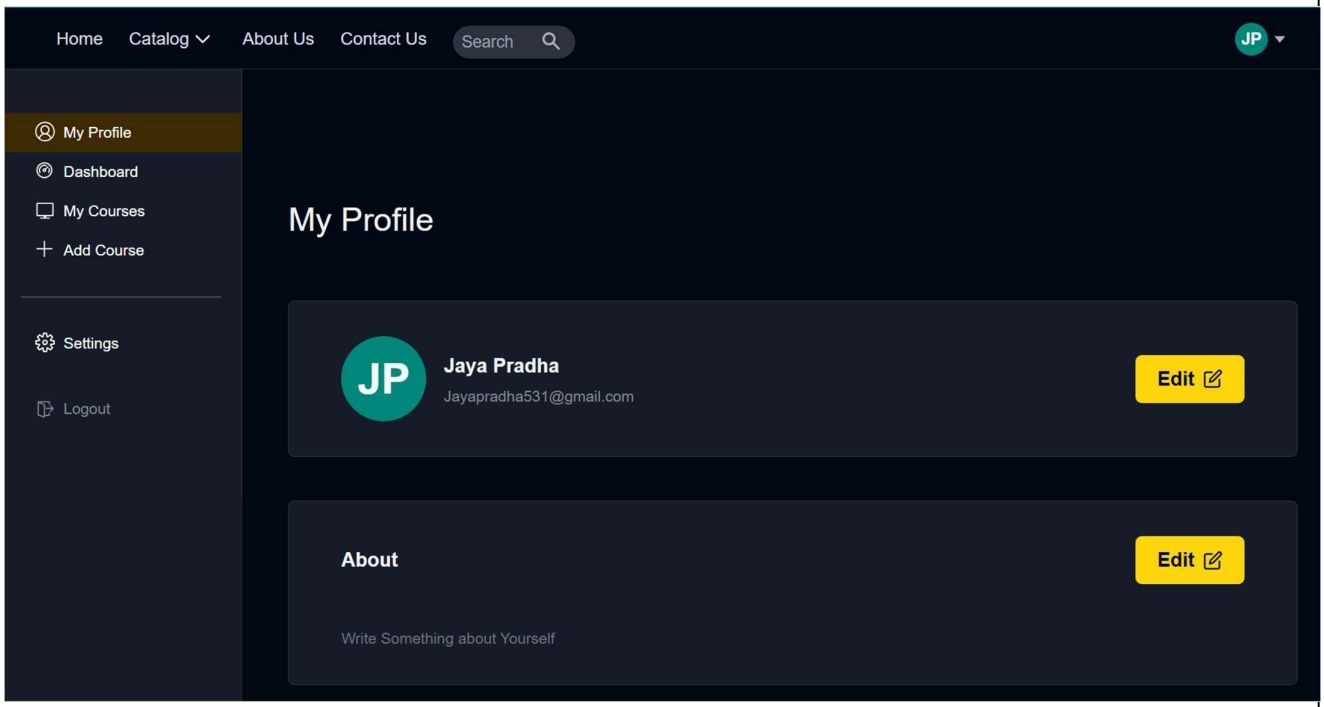


- Cart

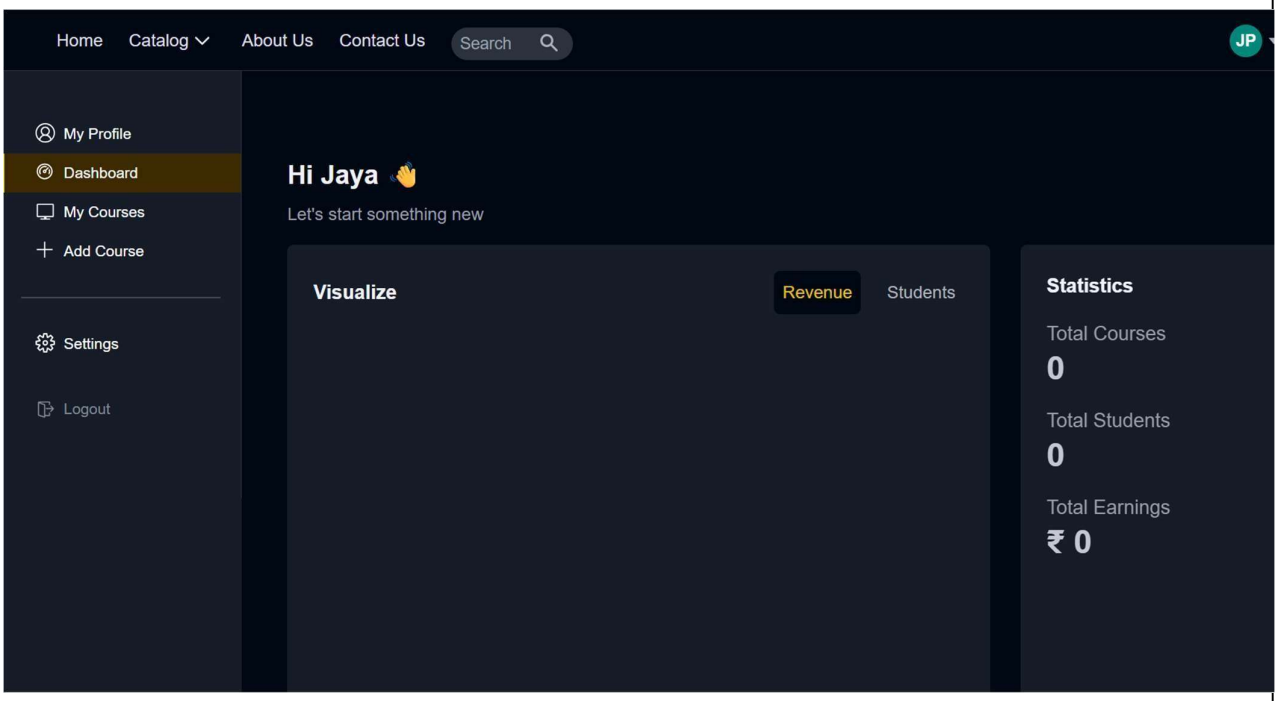


3] Instructor Login

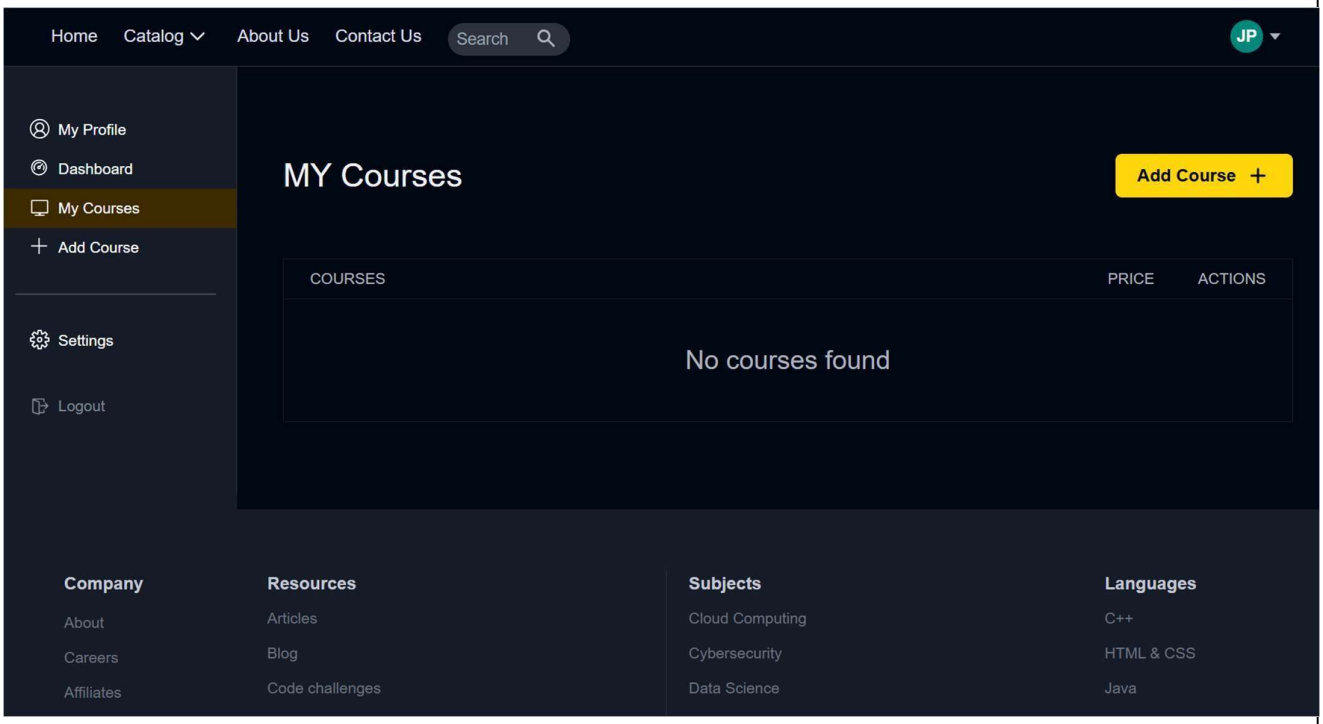
- My Profile



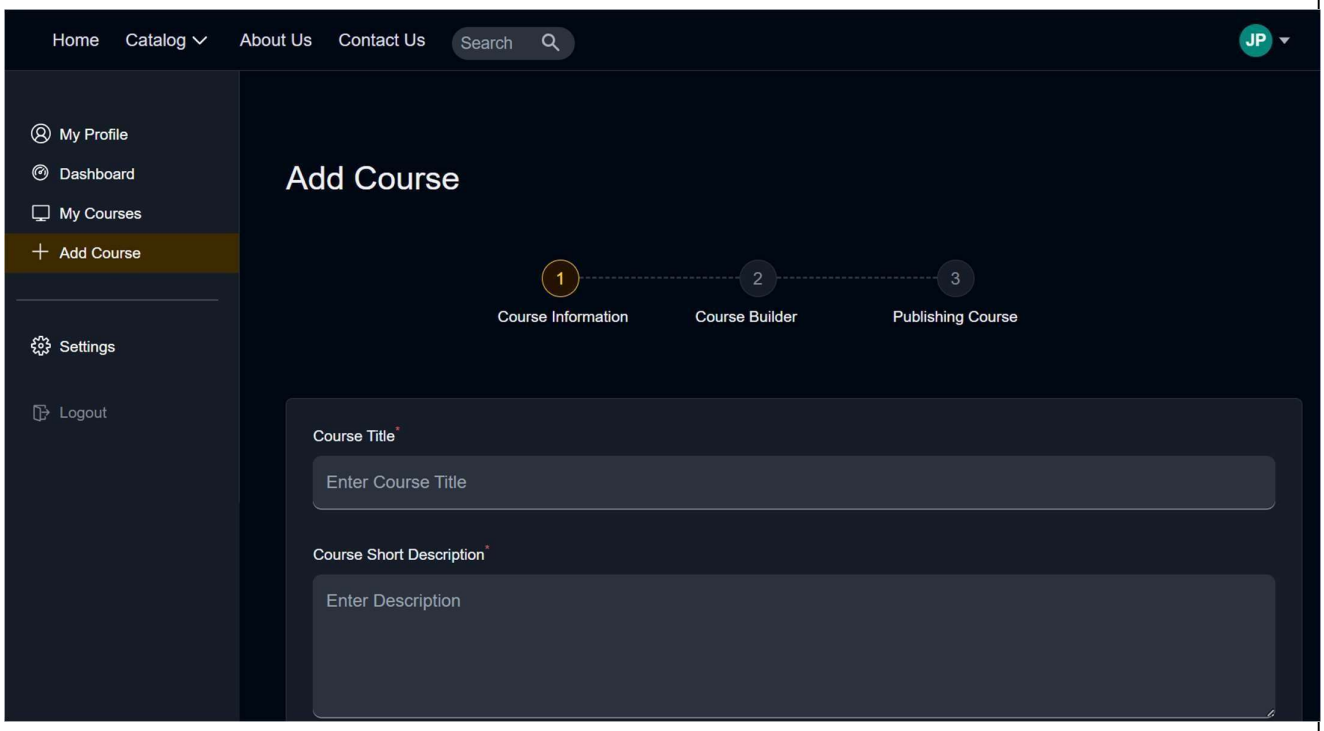
- Dashboard



- My courses



- Add Courses



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):

- service-worker.js:-

```
let cachedData  = "appV1"
self.addEventListener("install", (event) => {
event.waitUntil(        caches.open(cachedData
).then((cache) => {       const urls = [
"/static/js/main.*.js",
      "/static/css/main.*.css",
      "/index.html",
      "/index.css",
      "/static/media/banner.8e687823b1422880cc3f.mp4",
      "/",]
      for(let
i=0;i<urls.length;i++){
try{            cache.add(urls[i])
          console.log("cached",urls[i])
          }
catch(err){
console.log(err)
          }
      }
      }
      )
```

```
    )
});
```

- index.js

```
const express = require("express");
const app = express();

//routes import const userRoutes =
require("./routes/User"); const paymentRoutes =
require("./routes/Payments"); const profileRoutes =
require("./routes/Profile"); const CourseRoutes =
require("./routes/Course");

const database = require("./config/database");
const cookieParser = require("cookie-parser");

const cors = require("cors"); const fileUpload =
require("express-fileupload"); const { cloudnairyconnect }
= require("./config/cloudinary");

const dotenv = require("dotenv");
dotenv.config();

const PORT = process.env.PORT || 4000;
database.connect();

//middlewares
app.use(express.json());
app.use(cookieParser());

app.use(
cors({
   origin:
JSON.parse(process.env.CORS_ORIGIN),
credentials: true,     maxAge: 14400,
  })
);

app.use(
fileUpload({
useTempFiles: true,
tempFileDir: "/tmp",
  })
);

cloudnairyconnect();
```

```
//routes app.use("/api/v1/auth",
userRoutes);

app.use("/api/v1/payment", paymentRoutes);

app.use("/api/v1/profile", profileRoutes);

app.use("/api/v1/course", CourseRoutes);

app.use("/api/v1/contact", require("./routes/ContactUs"));

app.get("/", (req, res) => {
res.status(200).json({
   message: "Welcome to the API",
 });
});

app.listen(PORT, () => {
 console.log(`Server is running on port ${PORT}`);
});
self.addEventListener("fetch", (event) => {
const requestUrl = new URL(event.request.url);

   if (requestUrl.origin === 'https://api.dicebear.com' && requestUrl.pathname
=== '/5.x/initials/svg') {        event.respondWith(
caches.open('user-image-cache').then(function(cache) {                return
cache.match(event.request).then(function (response) {                  return
response || fetch(event.request).then(function(response) {
cache.put(event.request, response.clone());                      return response;
                });
              });
          })
       );
   }

   if(!navigator.onLine){       event.respondWith(
caches.match(event.request).then((resp)    =>    {
if(resp){                return resp
          }
          let requestUrl = event.request.clone();
return fetch(requestUrl)
       })
   )
   }
})
```

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.