



**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**

<b>Ex No</b>	<b>List of Experiments</b>
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

<b>Requirements</b>	
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 Usecase 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

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

### CO - PO – PSO matrices of course

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

Correlation levels 1, 2 or 3 are as defined below:

1: Slight (Low)    2: Moderate (Medium)    3: Substantial (High)    No correlation: “-”

## **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 was successfully completed.

**EX NO: 2**

## **PROBLEM STATEMENT**

### **AIM:**

To prepare PROBLEM STATEMENT for your given project.

### **Problem Statement:**

In the current digital education landscape, learners often struggle to stay motivated, follow a structured learning path, and measure their progress effectively across various tech courses. Traditional e-learning platforms lack personalized guidance, goal-oriented learning paths, and interactive motivation tools that encourage consistent engagement. This results in poor learning outcomes, high dropout rates, and low course completion rates.

Our system addresses these challenges by providing an intelligent e-learning platform with personalized roadmaps, streak tracking, and achievement badges to boost motivation and help learners reach their goals effectively.

### **Result:**

The problem statement was written successfully.

**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.



**Aim:**

To create User Stories

**THEORY**

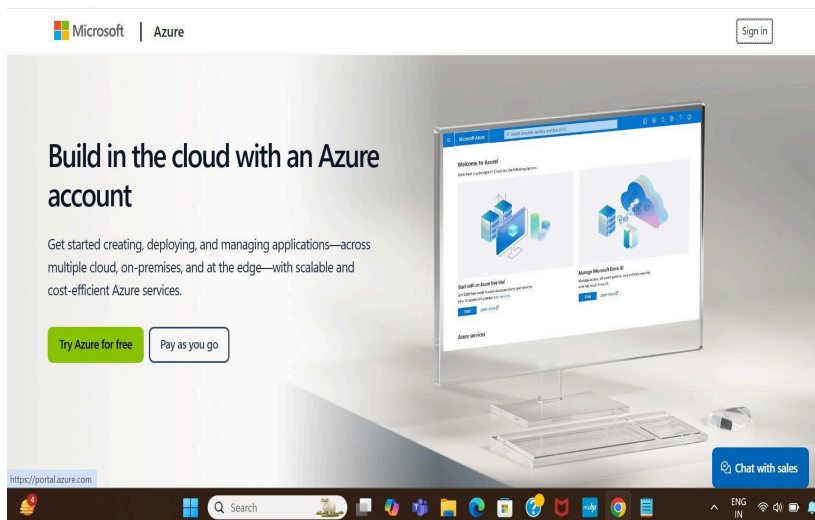
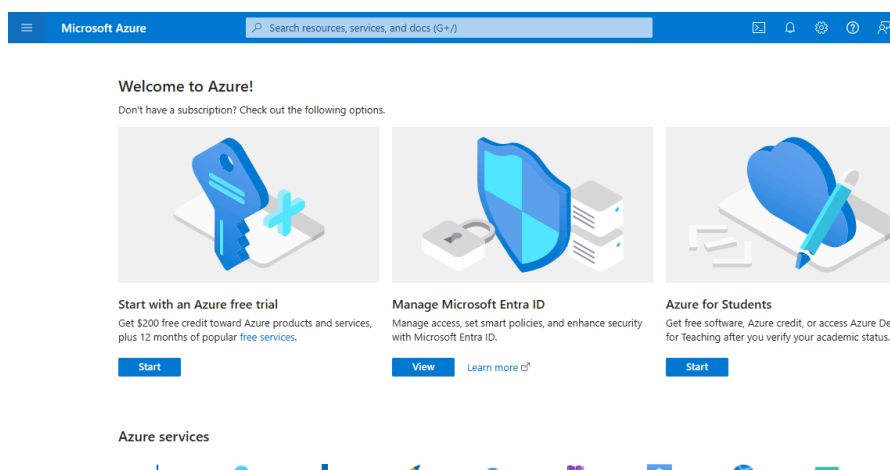
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

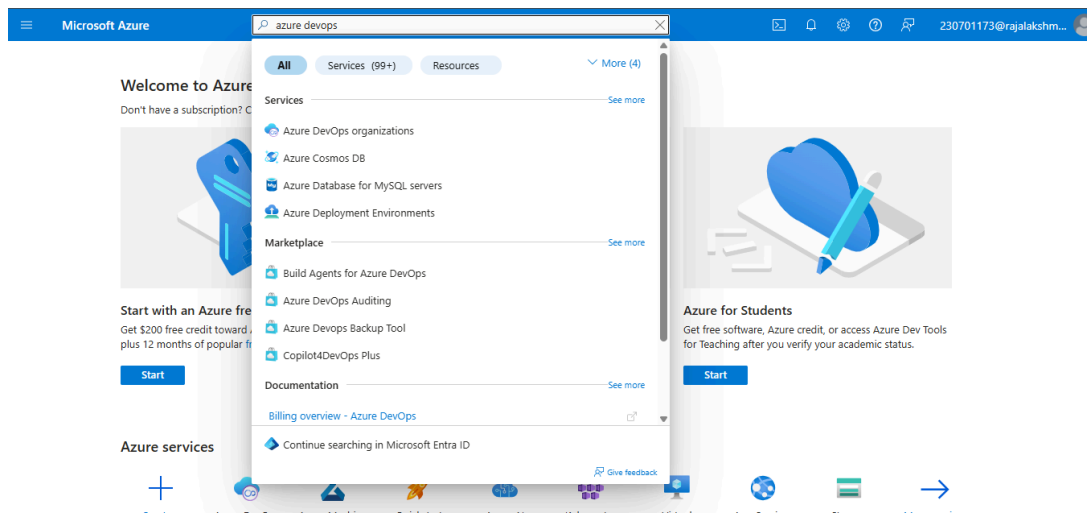
User story template

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

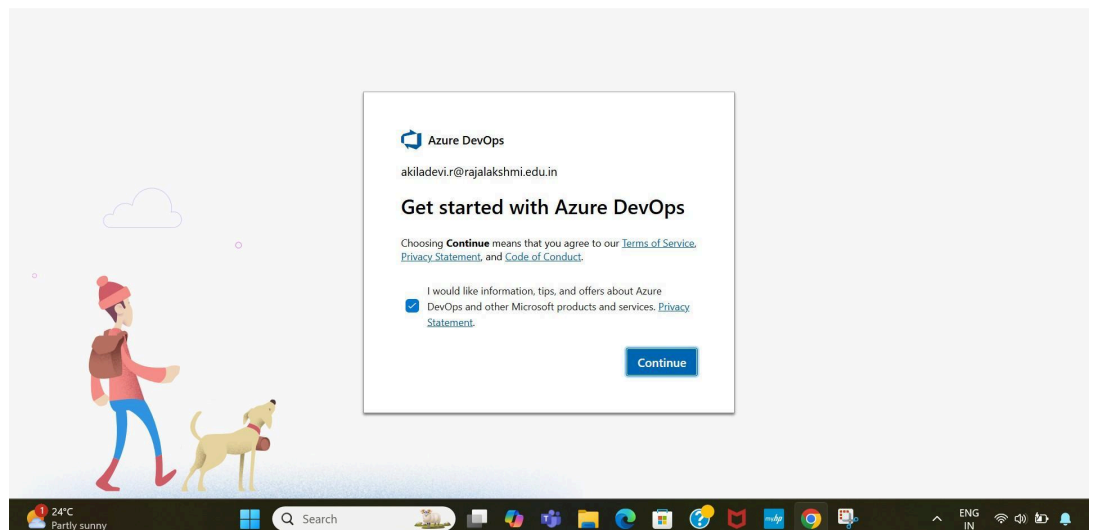
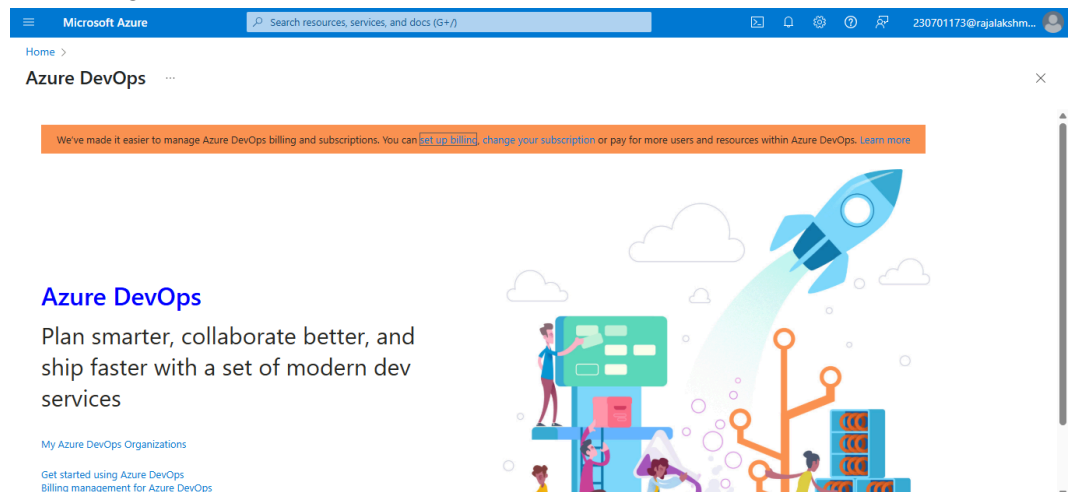
**Procedure:**

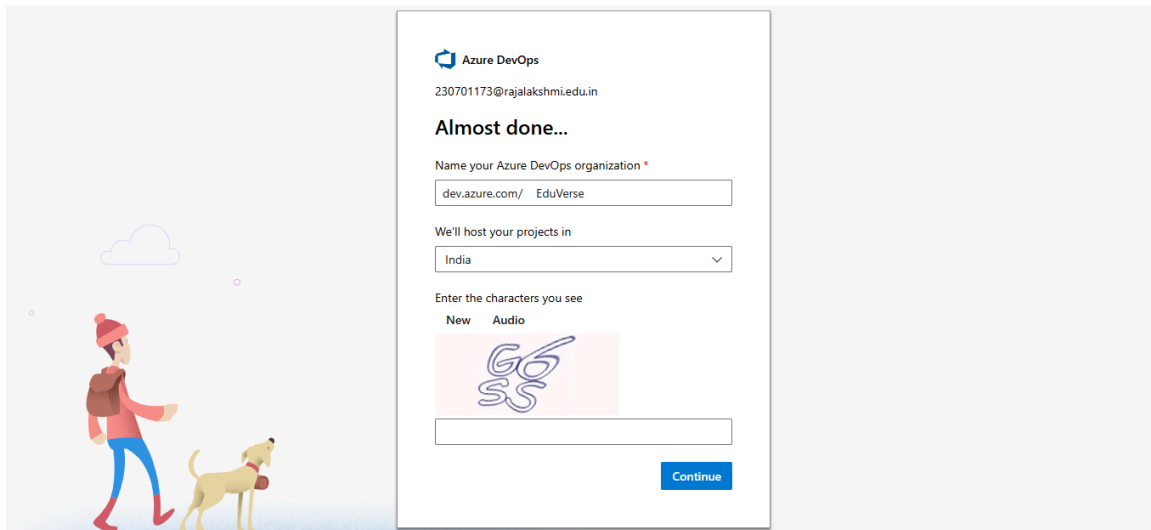
1. Open your web browser and go to the Azure website:  
<https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for  
<https://signup.live.com/?lic=1>

**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:
  - o **Name:** Choose a name for the project (e.g., **LMS**).
  - o **Description:** Optionally, add a description to provide more context about the project.
  - o **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.

Microsoft

Meenakshi SomuSign out

MS

Meenakshi Somu

230701173@rajalakshmi.edu.in

India

230701173@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

Create new organization

dev.azure.com/230701173 (Owner)

Projects

EduVerse

New project

Actions

Open in Visual Studio

dev.azure.com/230701168 (Member)

## 8. Project dashboard

Azure DevOps230701173 / EduVerse / Overview / Summary

Search

MS

EduVerse

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

Project settings

EduVerse

Private

Invite

+

About this project

Help others to get on board!

Describe your project and make it easier for other people to understand it.

+ Add Project Description

Project stats

Period: Last 7 days

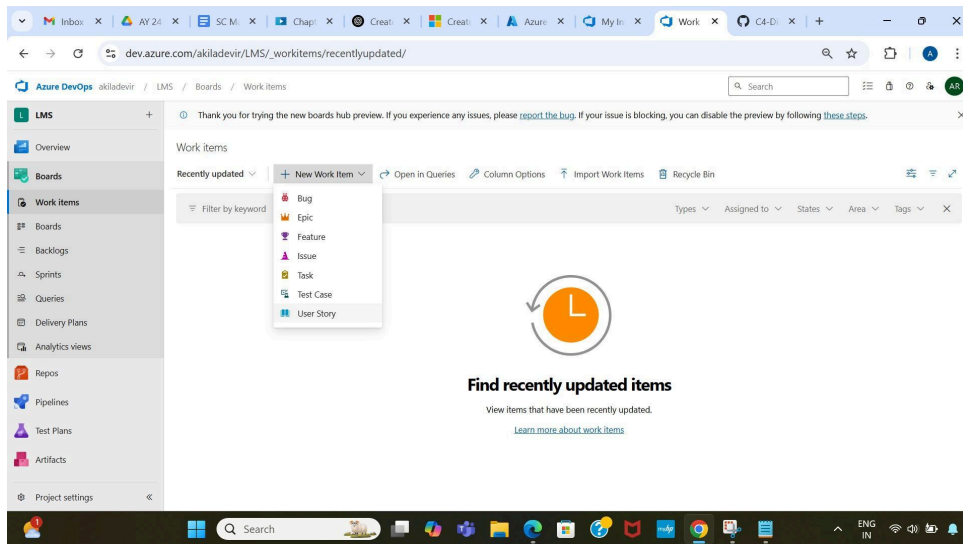
Boards

0 Work items created

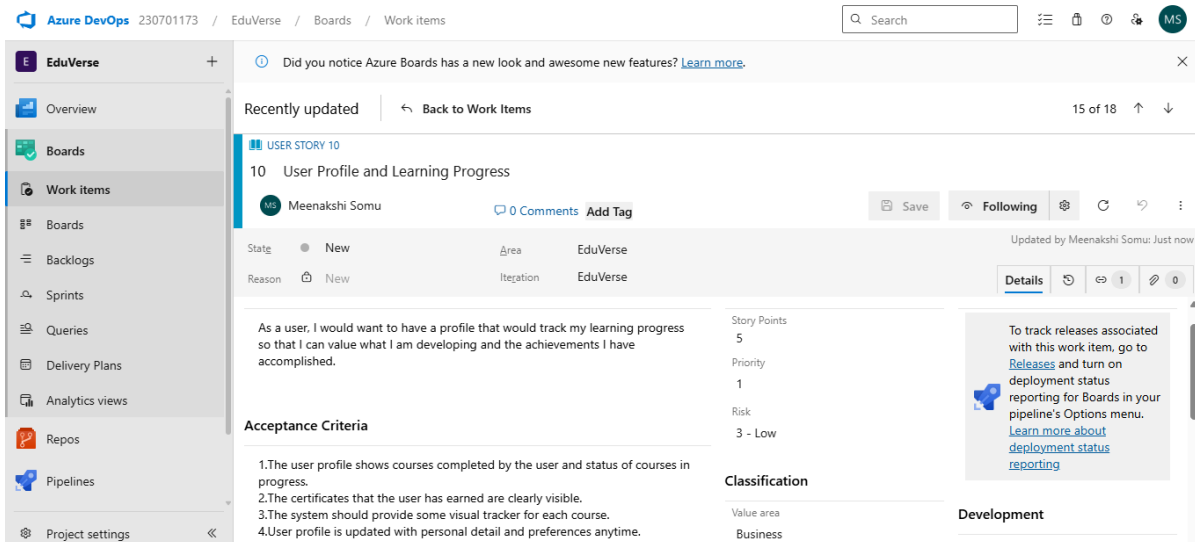
0 Work items completed

## 9. To manage user stories

- 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.
- 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.

**Aim:**

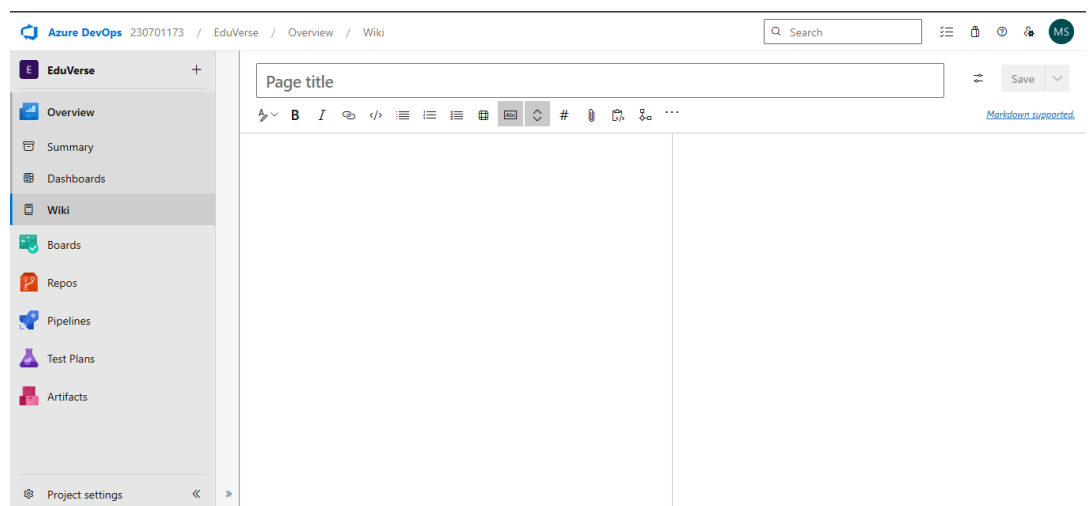
To design a Sequence Diagram by using Mermaid.js

**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
```

```
    CUSTOMER ->> BOOK-LOAN : borrows
```

```
    CUSTOMER ->> LIBRARY-MEMBER :
```

```
    registers CUSTOMER ->>{ FINE : incurs
```

```
    BOOK-LOAN ->> BOOK : contains
```

```
    BOOK-LOAN ->> LIBRARY-MEMBER : issued-to
```

```
    BOOK ->> BOOK-LOAN : "borrowed in"
```

```
    LIBRARY-MEMBER ->> BOOK-LOAN : "issued in"
```

```
    LIBRARY-MEMBER ->> LIBRARY-STAFF :
```

```
    manages BOOK-CATEGORY ->> BOOK : belongs-to
```

```
:::
```

**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 an open arrow at the end (async)

Sequence diagram V2

```
sequenceDiagram
    participant User
    participant WebApp
    participant CourseService
    participant PaymentService
    participant DB
    participant BadgeSystem

    User->>WebApp: Login/Register
    WebApp->>DB: Validate User
    DB-->>WebApp: Success/Fail

    User->>WebApp: Enroll in Course
    WebApp->>CourseService: Check Course Availability
    CourseService->>DB: Query Course
    DB-->>CourseService: Course Details

    WebApp->>PaymentService: Initiate Payment
    PaymentService->>DB: Save Payment Info
    DB-->>PaymentService: Payment Confirmed

    WebApp->>DB: Create Enrollment
    DB-->>WebApp: Enrollment Confirmed

    WebApp->>BadgeSystem: Update Badge Progress
    BadgeSystem->>DB: Update Badge Record
```

#### 4. click wiki menu and select the page

Sequence diagram V2

```
sequenceDiagram
    participant User
    participant WebApp
    participant CourseService
    participant PaymentService
    participant DB
    participant BadgeSystem

    User->>WebApp: Login/Register
    WebApp->>DB: Validate User
    DB-->>WebApp: Success/Fail

    User->>WebApp: Enroll in Course
    WebApp->>CourseService: Check Course Availability
    CourseService->>DB: Query Course
    DB-->>CourseService: Course Details

    WebApp->>PaymentService: Initiate Payment
    PaymentService->>DB: Save Payment Info
    DB-->>PaymentService: Payment Confirmed

    WebApp->>DB: Create Enrollment
    DB-->>WebApp: Enrollment Confirmed

    WebApp->>BadgeSystem: Update Badge Progress
    BadgeSystem->>DB: Update Badge Record
```

**Result:**

The sequence diagram was drawn successfully.



## EX NO. 6

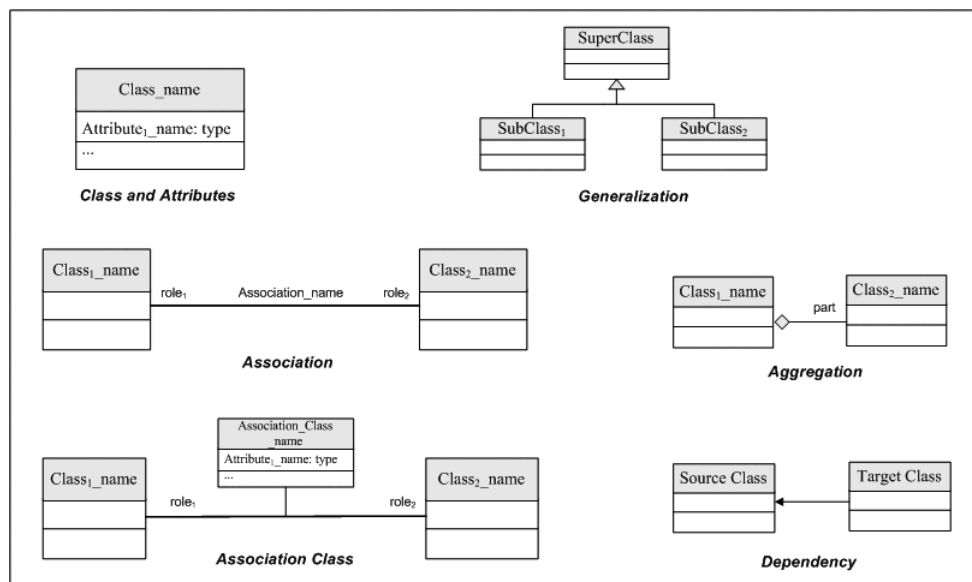
### CLASS DIAGRAM

#### AIM :-

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

#### THEORY

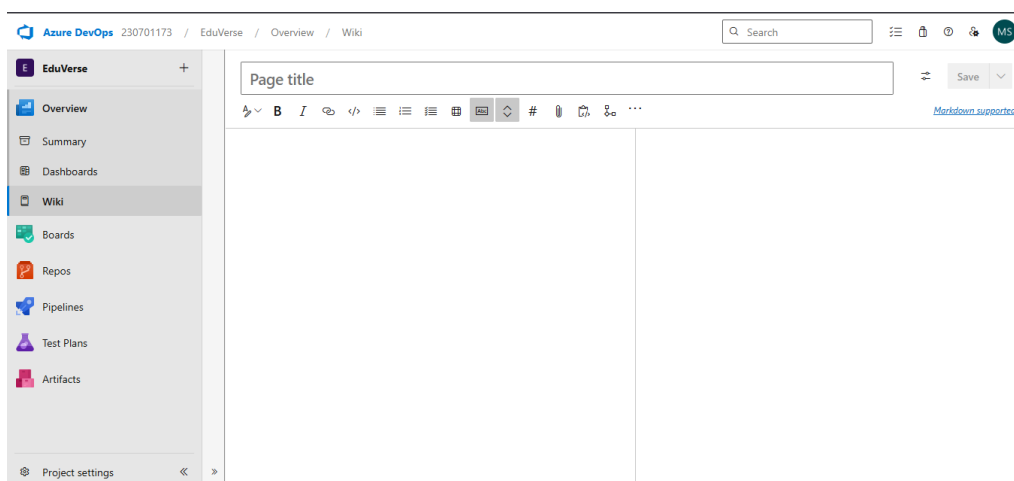
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



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
class Person {
+int userID
+string name
+string gender
+string email
+register()
+login()
+updateProfile()
+deleteAccount()
}

class Student {
+List registeredCourses
+List badges
+enroll()
+viewProgress()
+applyFinancialAid()
+requestCertificate()
+viewBadges()
}

class Instructor {
+List coursesTaught
+createCourse()
+updateCourse()
+viewStudents()
+gradeAssignments()
}

class Enrollment {
+int enrollmentID
+string status
+trackProgress()
+completeCourse()
+unenroll()
}

class GPSWorkshop {
+int workshopID
+string title
+string location
+string date
+findNearbyWorkshop()
+registerForWorkshop()
}

class Badge {
+int badgeID
+string name
+string description
+string criteria
+awardBadge(Student)
}

class Payment {
+int paymentID
+float amount
+string status
+processPayment()
+refund()
}
```

```

class StreakTracker {
    +int streakDays
    +increaseStreak()
    +resetStreak()
    +getStreakInfo()
}

class FinancialAid {
    +int aidID
    +string status
    +applyForAid()
    +approveAid()
    +rejectAid()
}

class Notification {
    +int notificationID
    +string message
    +datetime timestamp
    +sendNotification()
}

class Course {
    +int courseID
    +string title
    +string description
    +float price
    +string category
    +float rating
    +enroll(Student)
    +search(string)
    +addRecommendation(Student)
    +getCourseInfo()
}

class Review {
    +int reviewID
    +int rating
    +string comment
    +addReview()
    +viewReviews(courseID)
}

Person <|-- Student
Person <|-- Instructor

Student "1" --> "0..*" Enrollment : creates
Enrollment "1" --> "1" Course : to
Student "1" --> "0..*" GPSWorkshop : attends
Student "1" --> "0..*" Payment : makes
Student "1" --> "1" StreakTracker : tracks
Student "1" --> "0..*" FinancialAid : applies for
FinancialAid "1" --> "1" Course : applies to
Student "1" --> "0..*" Notification : receives
Student "1" --> "0..*" Review : writes
Review "1" --> "1" Course : reviews
Instructor "1" --> "0..*" Course : teaches
Student "1" --> "0..*" Badge : earns

```

```

%% Composition and Aggregation (symbolic only — visual only)
Course *-- Review : has
Student *-- StreakTracker : maintains
Student *-- Enrollment : owns

```

Course o-- FinancialAid : supports  
 Course o-- Payment : linked to  
 Course o-- Badge : awards

## Relationship Types

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

Azure DevOps 230701173 / EduVerse / Overview / Wiki / class diagram

Page title

```

+addReview()
+viewReviews(courseID)
}

Person <|-- Student
Person <|-- Instructor

Student "1" --> "0..*" Enrollment : creates
Enrollment "1" --> "1" Course : to
Student "1" --> "0..*" GPSWorkshop : attends
Student "1" --> "1" StreakTracker : tracks
Student "1" --> "0..*" FinancialAid : applies for
Student "1" --> "1" Course : receives
Student "1" --> "0..*" Review : writes
Review "1" --> "1" Course : reviews
Instructor "1" --> "0..*" Course : teaches
Student "1" --> "0..*" Badge : earns

%% Composition and Aggregation (symbolic only - visual only)
Course *-- Review : has
Student *-- StreakTracker : maintains
Student *-- Enrollment : owns
Course o-- FinancialAid : supports
Course o-- Payment : linked to
Course o-- Badge : awards
  
```

Markdown supported



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

## Result:

The class diagram was designed successfully.

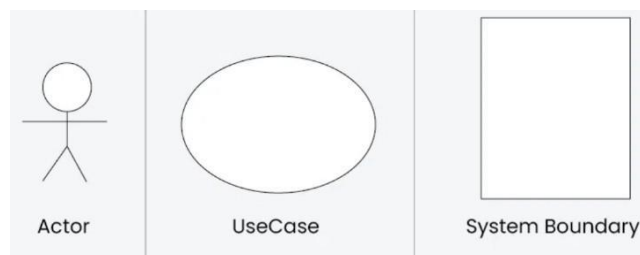
**Aim:**

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**

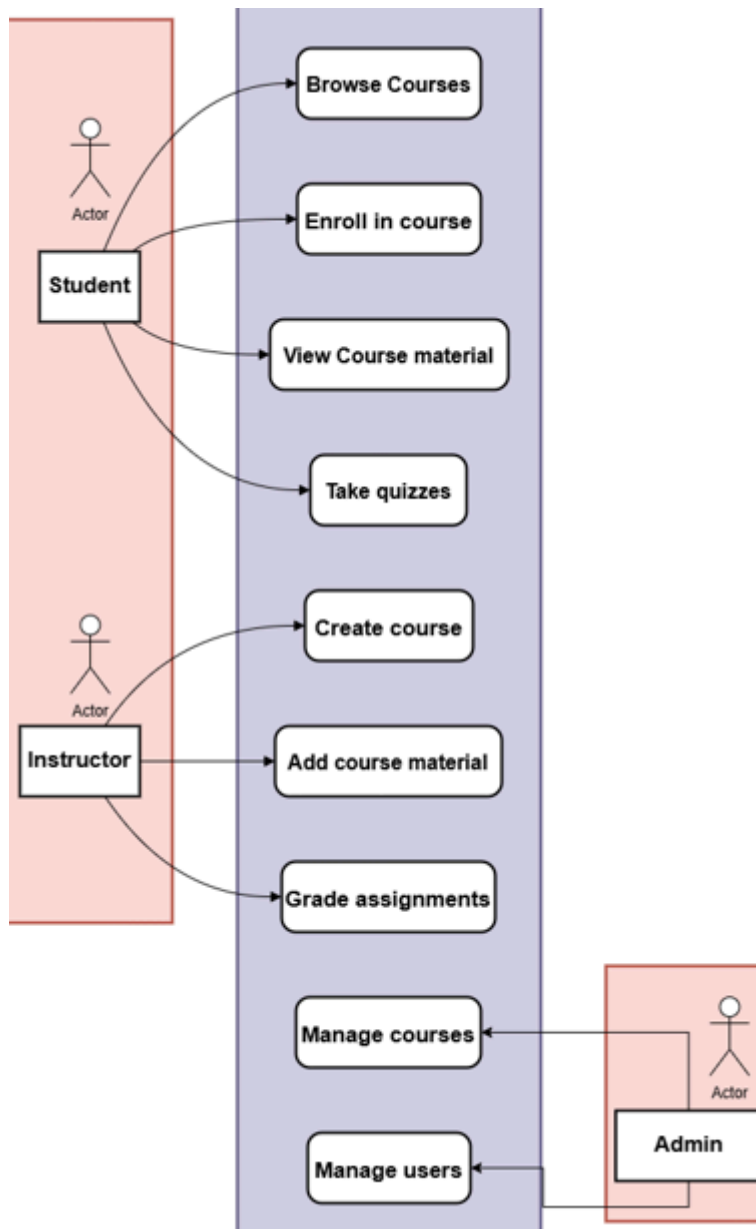
Step 1: Create the Use Case Diagram in Draw.io

- 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)



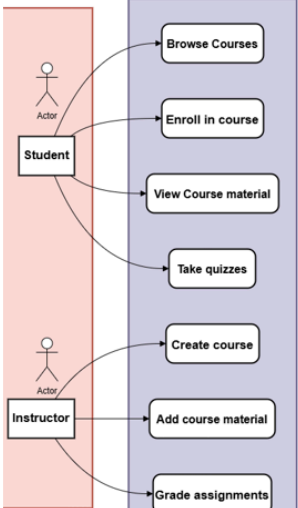
#### Option 2: Attach to Work Items in Azure Boards

- 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 diagram.

Azure DevOps 230701173 / EduVerse / Overview / Wiki / Sequence diagram

Page title

![[Screenshot 2025-05-04 125716.png](/.attachments/Screenshot%202025-05-04%20125716-160ce55a-6d93-4629-a4b3-74877996715a.png)]]



```
graph LR
    subgraph Actors
        Student
        Instructor
    end
    subgraph UseCases
        UC1[Browse Courses]
        UC2[Enroll in course]
        UC3[View Course material]
        UC4[Take quizzes]
        UC5[Create course]
        UC6[Add course material]
        UC7[Grade assignments]
    end
    Student --> UC1
    Student --> UC2
    Student --> UC3
    Student --> UC4
    Instructor --> UC5
    Instructor --> UC6
    Instructor --> UC7
```

The diagram is a UML Use Case Diagram for the EduVerse system. It features two actors, 'Student' and 'Instructor', each represented by a stick figure icon and a label box. These actors are positioned on the left side of the diagram within a light red vertical rectangular area. To the right of this area is a light blue vertical rectangular area containing seven use cases, each represented by a rounded rectangle. The use cases are: 'Browse Courses', 'Enroll in course', 'View Course material', 'Take quizzes', 'Create course', 'Add course material', and 'Grade assignments'. Arrows indicate the associations between the actors and the use cases: 'Student' is associated with 'Browse Courses', 'Enroll in course', 'View Course material', and 'Take quizzes'; 'Instructor' is associated with 'Create course', 'Add course material', and 'Grade assignments'.

## Result:

The use case diagram was designed success







## EX NO. 8



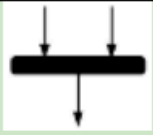








### ACTIVITY DIAGRAM

#### 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 to communicate additional messages
Send signal		Show that a signal is being sent to a receiving activity
Receive signal		Demonstrates the acceptance of an event
Flow final symbol		Represents the end of a specific process flow
Option loop		Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate		Represents a transition that invokes the last active state.
End		Marks the end state of an activity and represents the completion of all flows of a process

#### Procedure

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

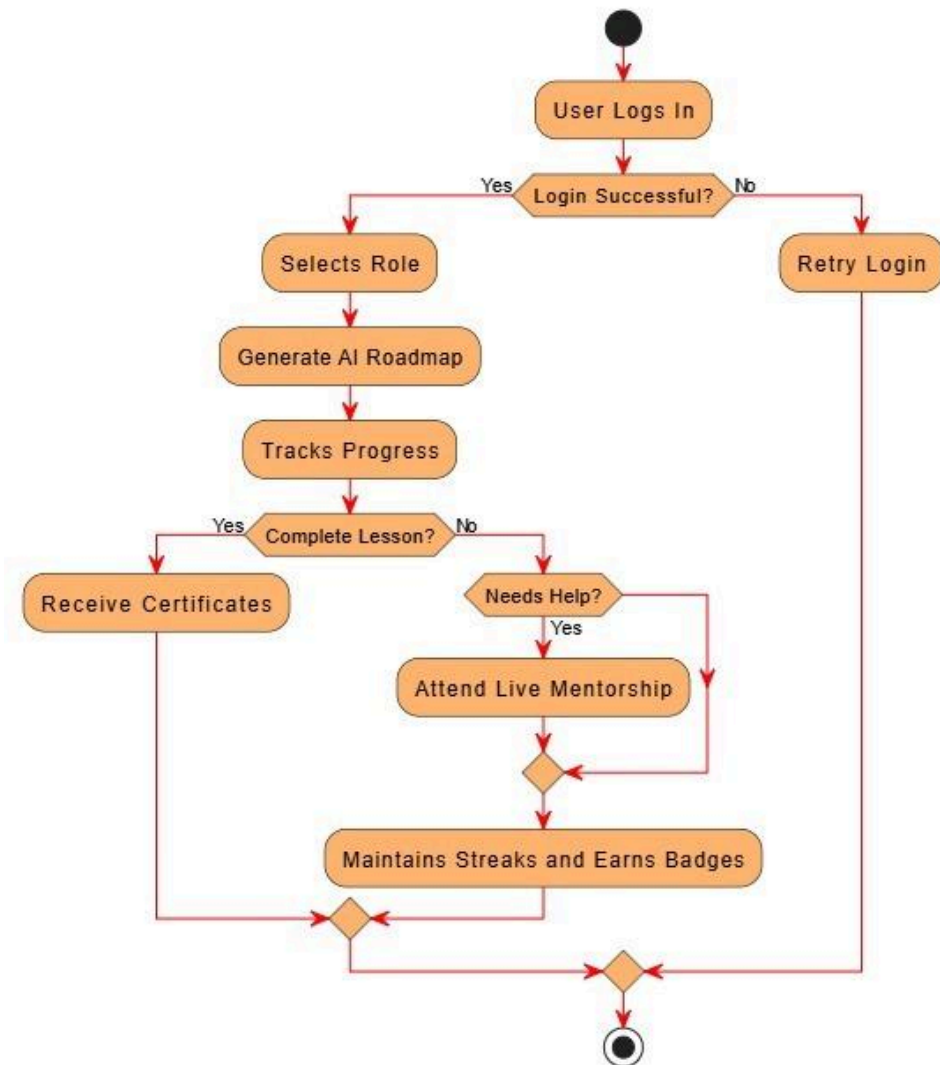
Azure DevOps 230701173 / EduVerse / Overview / Wiki / Sequence diagram

Activity diagram

```

graph TD
    Start(( )) --> UserLogsIn[User Logs In]
    UserLogsIn --> LoginSuccessful{Login Successful?}
    LoginSuccessful -- No --> RetryLogin[Retry Login]
    LoginSuccessful -- Yes --> SelectsRole[Selects Role]
    SelectsRole --> GenerateAIRoadmap[Generate AI Roadmap]
    GenerateAIRoadmap --> TracksProgress[Tracks Progress]
    TracksProgress --> CompleteLesson{Complete Lesson?}
    CompleteLesson -- No --> NeedsHelp{Needs Help?}
    NeedsHelp -- Yes --> AttendLiveMentorship[Attend Live Mentorship]
    AttendLiveMentorship --> MaintainsStreaks[Maintains Streaks and Earns Badges]
    CompleteLesson -- Yes --> ReceiveCertificates[Receive Certificates]
    ReceiveCertificates --> MaintainsStreaks
    MaintainsStreaks --> End(( ))
    RetryLogin --> End
  
```

Markdown supported.



## Result:

The activity diagram was designed successfully

**EX NO. 9**

## ARCHITECTURE DIAGRAM

**Aim:**

## Steps to draw the Architecture Diagram using draw.io.

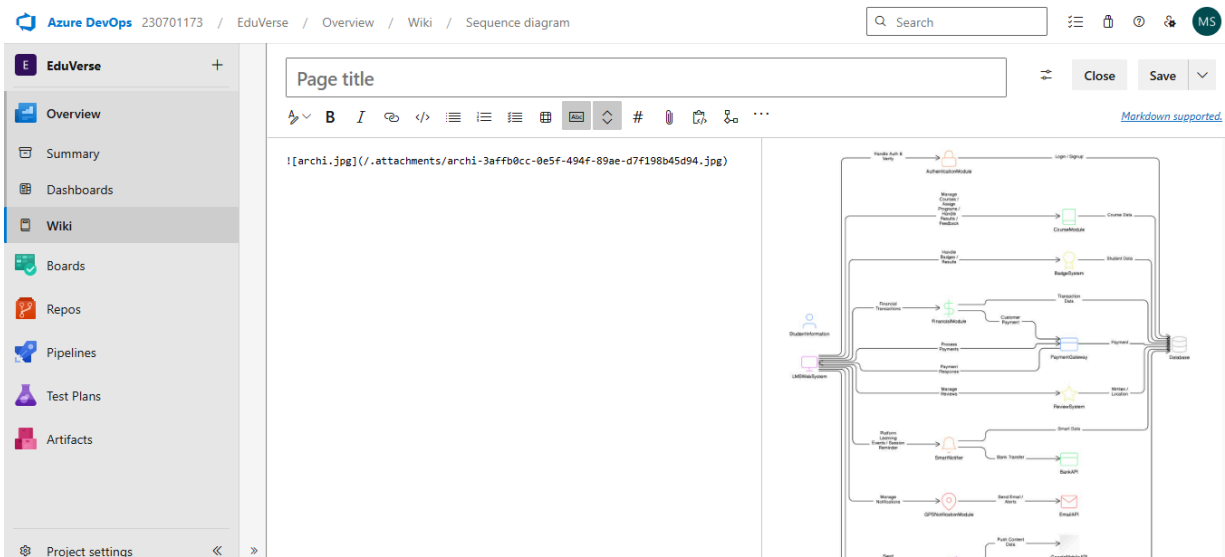
### Theory:

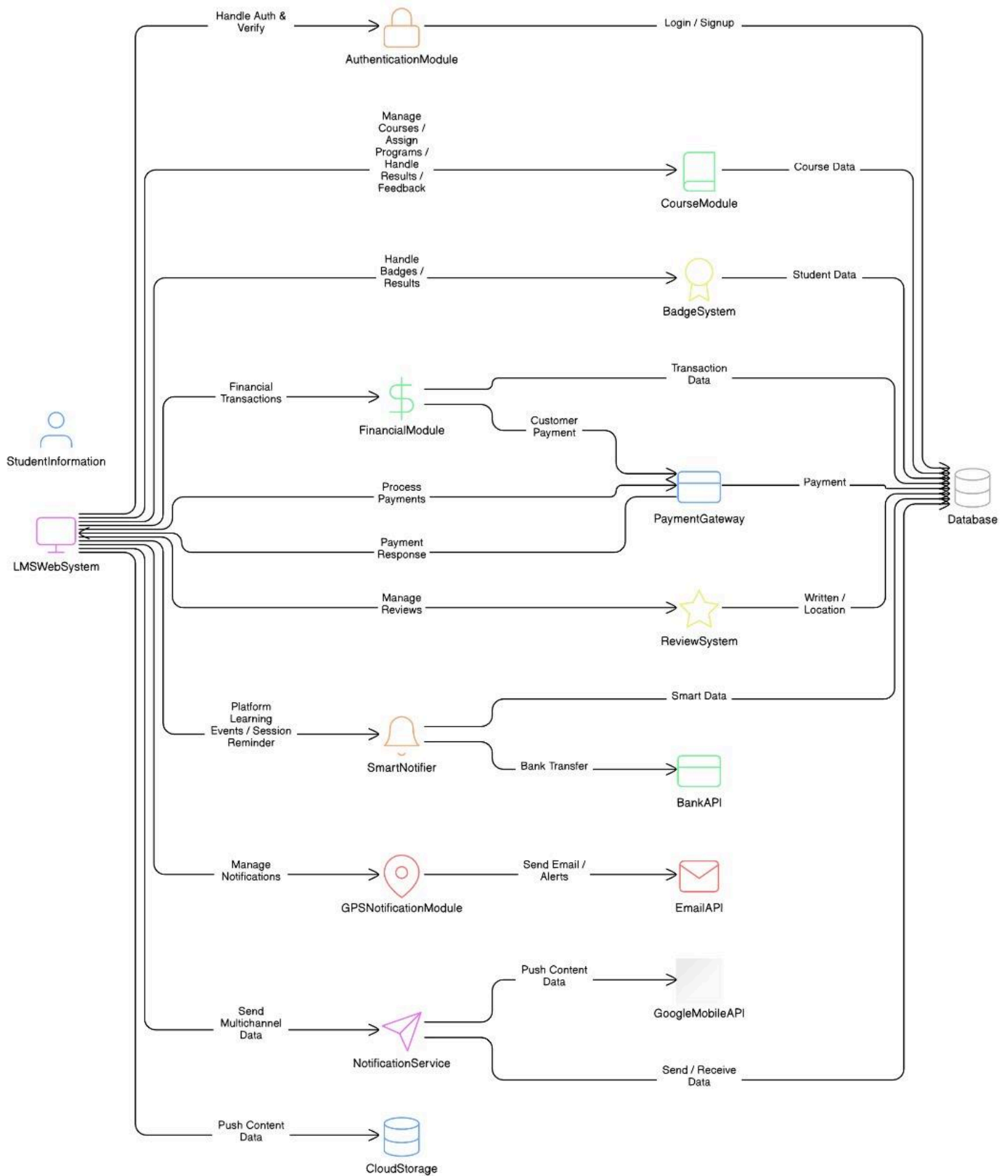
An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.



## Procedure

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



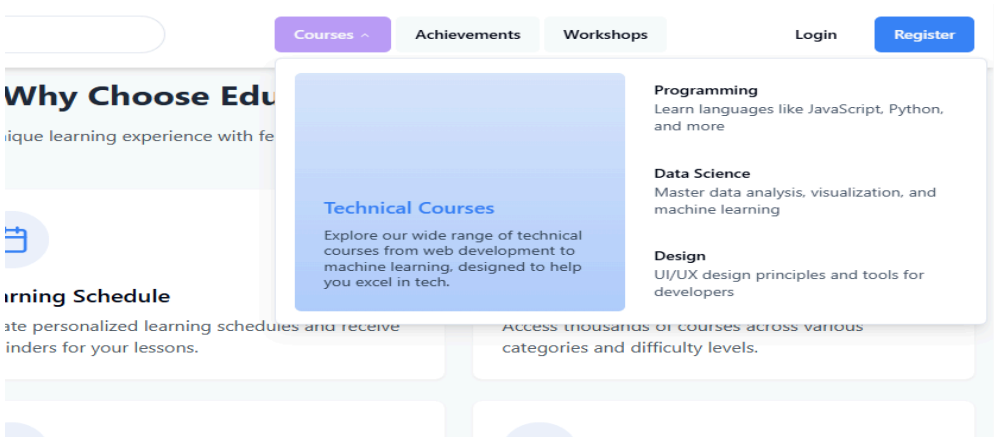
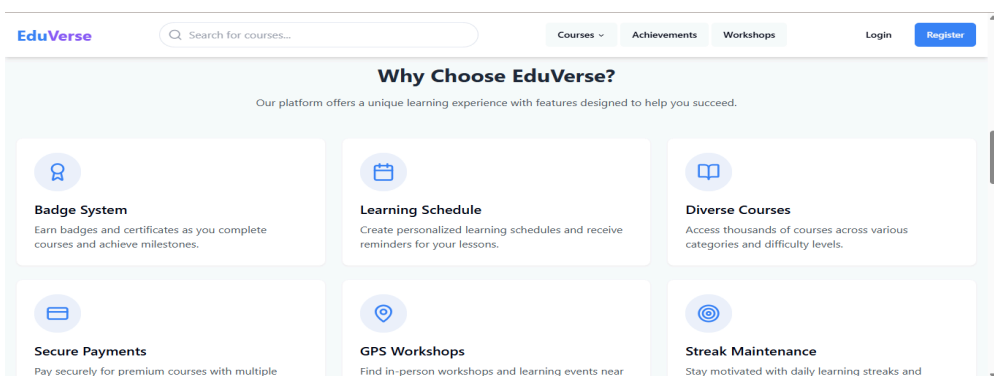
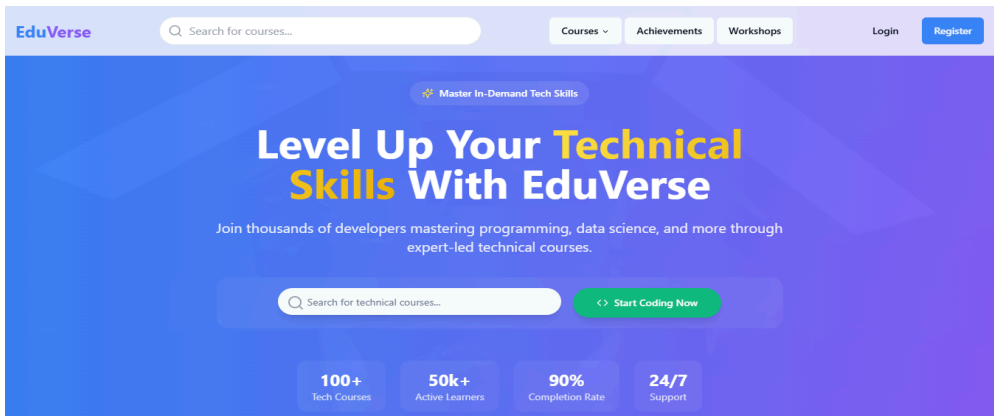


**Result:**

The architecture diagram was designed successfully

**Aim:**

Design User Interface for the given project



Learning Progress

Your Badges

**Fast Learner**

Complete a course within 7 days

**Knowledge Seeker**

Enroll in 5 different courses

**Consistent Scholar**

Maintain a 30-day learning streak

Locked

**Top Achiever**

Score 95% or higher on all assessments

Locked

Achievements

Track your learning journey milestones and showcase your accomplishments.

**First Course Completed**

Completed your first course on EduVerse

Achieved on 15/8/2023

Unlocked

**7-Day Streak**

Logged in for 7 consecutive days

Achieved on 20/8/2023

Unlocked

**Quiz Master**

Score 100% on 5 different quizzes

Complete the requirements to unlock this achievement.

Locked

**Community Contributor**

Help 10 other students in the community forum

Locked

**Web Development...**

Led by David Miller

Learn modern web development techniques with React, Node.js, and MongoDB.

15 May 2025

10:00 AM - 3:00 PM

Online

32/50 registered

Register Now

**Data Science for Beginners**

Led by Sophia Chen

Introduction to data analysis, visualization, and machine learning fundamentals.

20 May 2025

9:00 AM - 12:00 PM

Online

87/100 registered

Register Now

**UX/UI Design Workshop**

Led by Michael Roberts

Learn the principles of user-centered design and create stunning interfaces.

5 June 2025

11:00 AM - 4:00 PM

Tech Hub, San Francisco

18/30 registered

Register Now



404

Page Not Found

The page you're looking for doesn't exist or has been moved.

Back to Home

Browse Courses

**EduVerse**  
Your journey to technical excellence begins here

**Create Your Account**  
Join thousands of learners in our community

Sign In

Sign Up

Full Name

John Doe

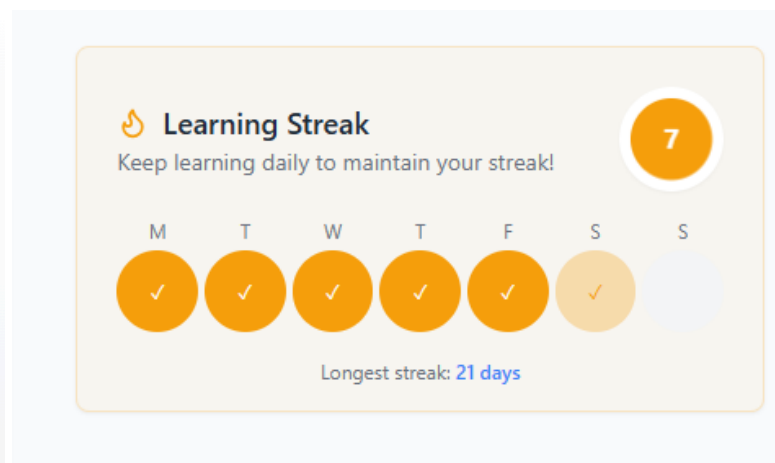
Username

johndoe

Email

your@email.com

Password



Result:  
The UI was designed successfully.



**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 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.