



**RAJALAKSHMI  
ENGINEERING COLLEGE**

An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ACADEMIC YEAR 2024-2025**



**CS23432 SOFTWARE CONSTRUCTION**

**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 Use-case 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	AMD Ryzen 5 5600H CPU @ 3.30GHz 3.29GHz, 8 GB RAM 512 GB SSD, 64-bit OS, x64-based processor
Software	StarUML, Azure

**LAB PLAN**  
**CS23432 - 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: “-”

**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 completed.

**EX NO: 2**

## **PROBLEM STATEMENT**

### **AIM:**

To prepare the problem statement for the Regiflex Course Registration System.

### **Problem Statement:**

Regiflex – Smart Course Registration System

In educational institutions, course registration is often a time-consuming and confusing process for students. The lack of a unified, intuitive platform leads to poor planning, scheduling conflicts, and missed academic opportunities. Students frequently register without fully understanding course prerequisites, available electives, credit loads, or how their selected courses align with their academic goals and future careers.

Regiflex is a course registration and recommendation system designed to streamline the registration process, prevent conflicts, and guide students in making smarter academic decisions. The system allows students to browse and register for courses, view their academic timetable, track credit loads, and receive AI-powered recommendations based on their interests, academic history, and program structure.

By improving clarity and automation in the registration process, Regiflex not only enhances the student experience but also assists academic advisors in managing course demand and student progress more efficiently.

### **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

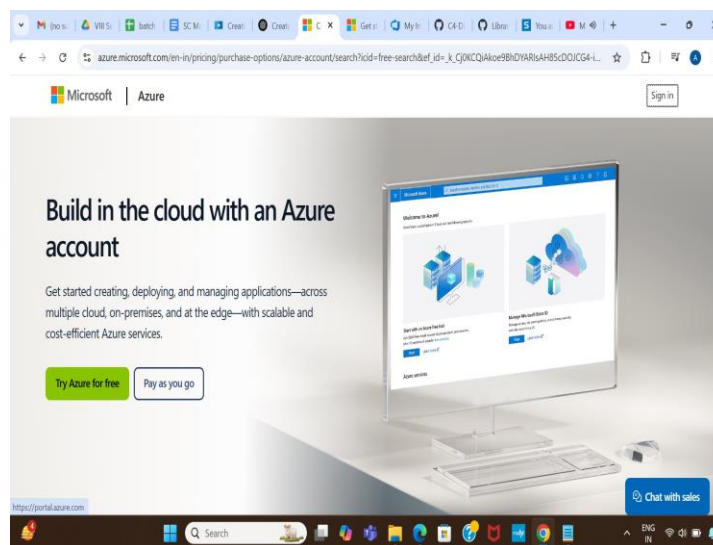
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

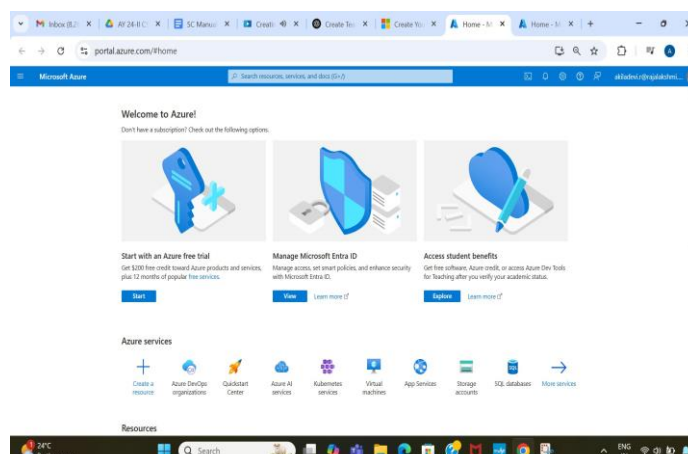
"As a [role], I [want to], [so that]."

#### Procedure:

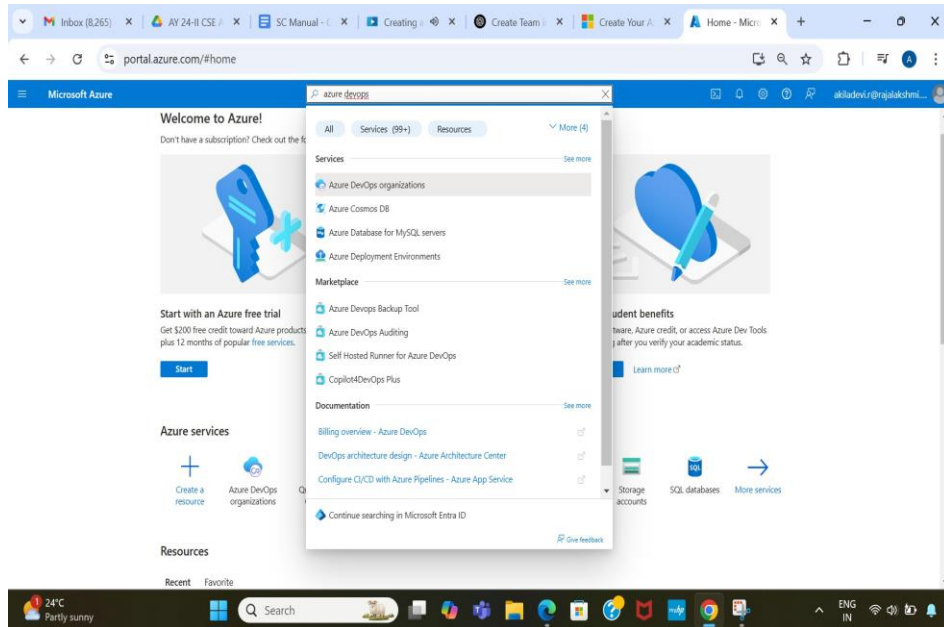
1. Open your web browser and go to the Azure website:  
<https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for  
<https://signup.live.com/?lic=1>



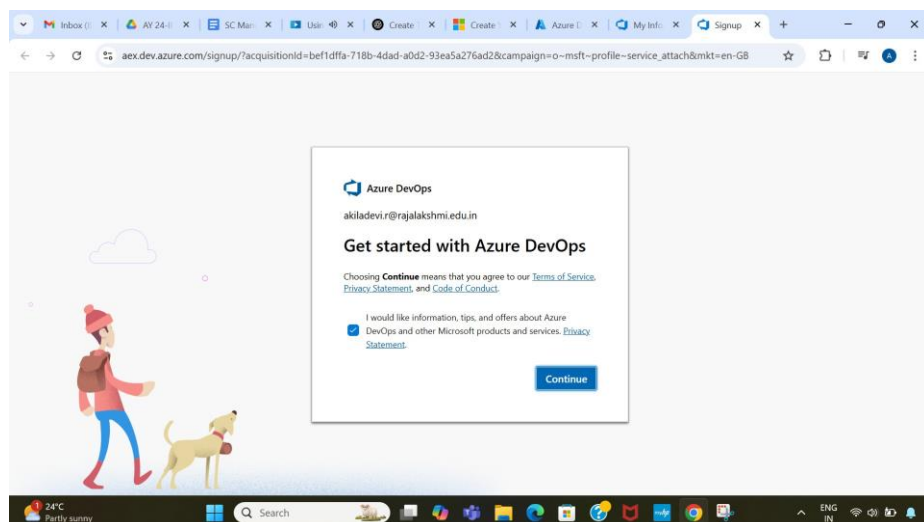
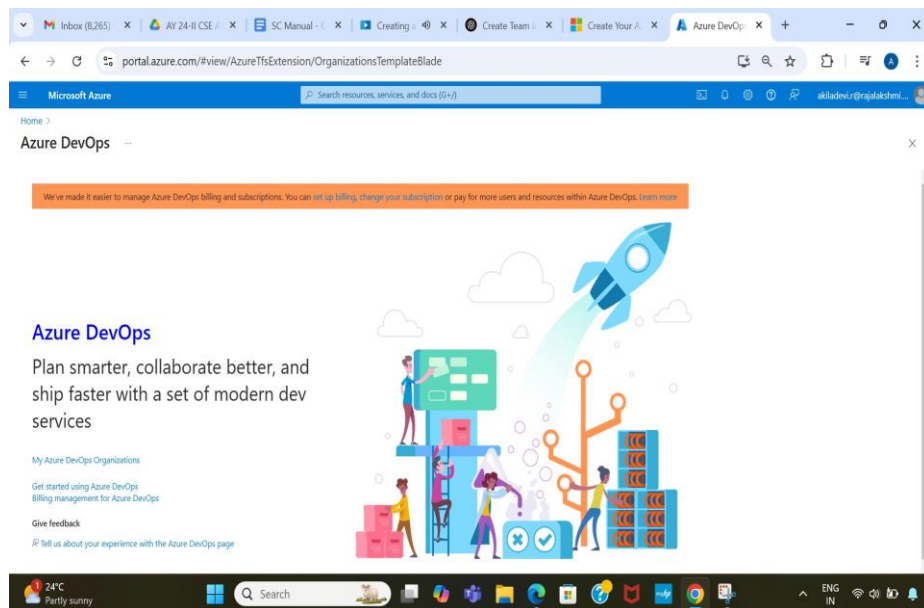
3. Azure home page

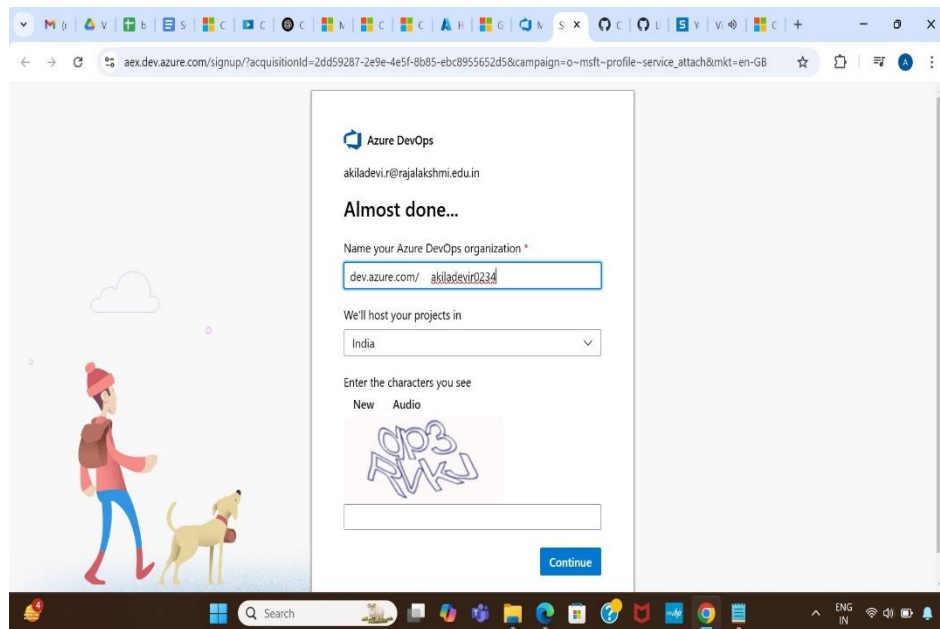


4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



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.



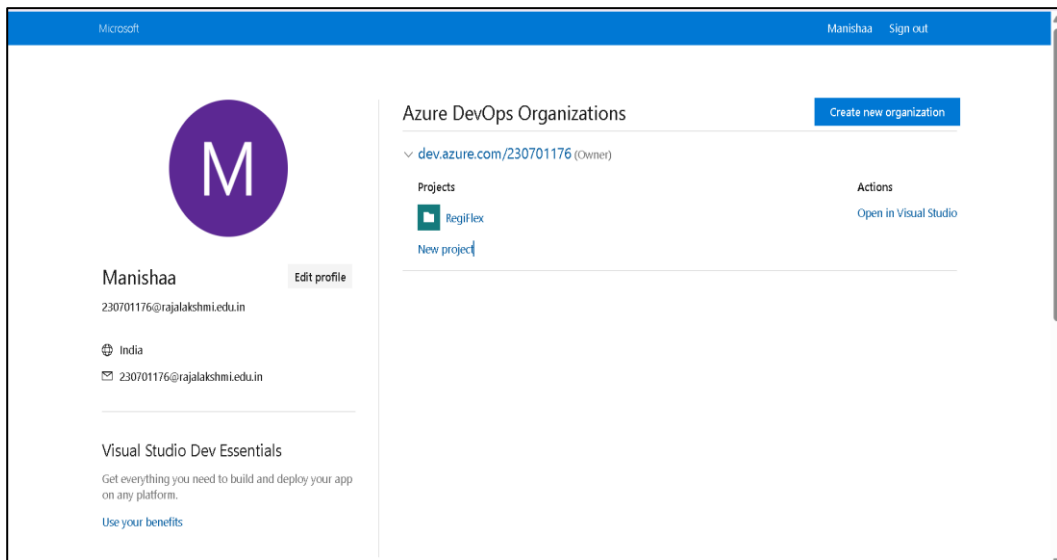


## 6. Create the First Project in Your Organization

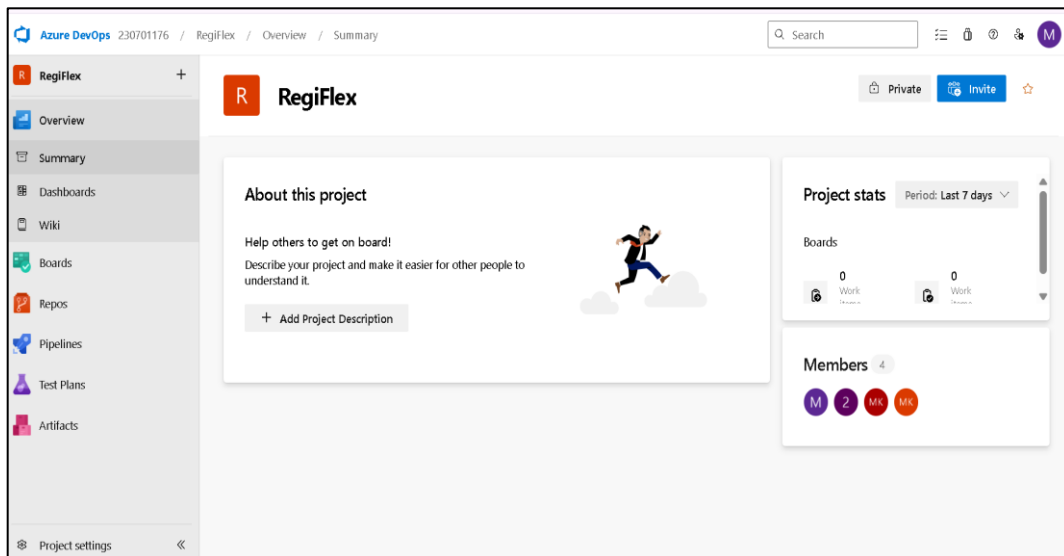
After the organization is set up, you'll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.

- i. On the organization's **Home page**, click on the **New Project** button.
- ii. Enter the project name, description, and visibility options:
  - 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.

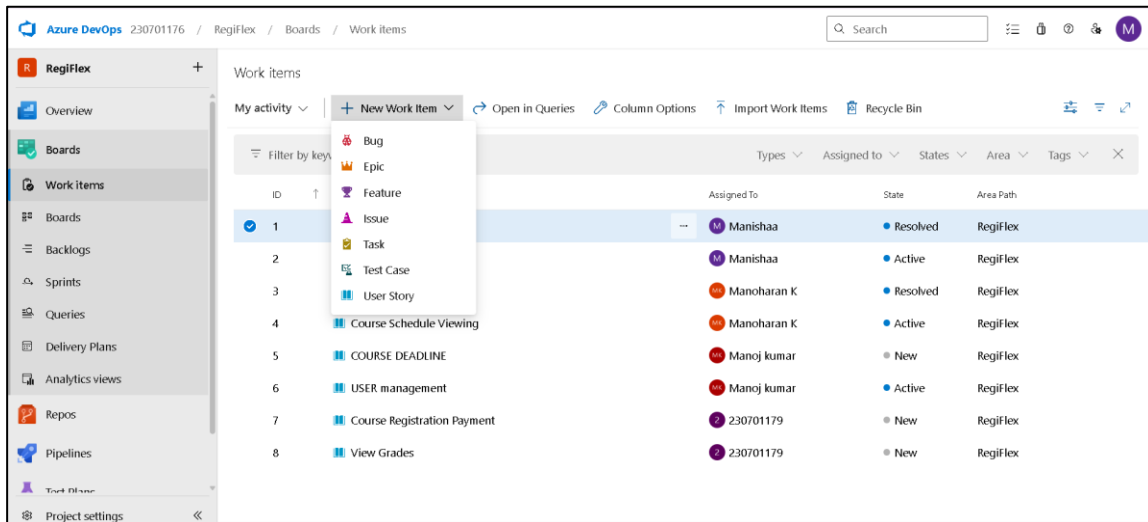


## 8. Project dashboard

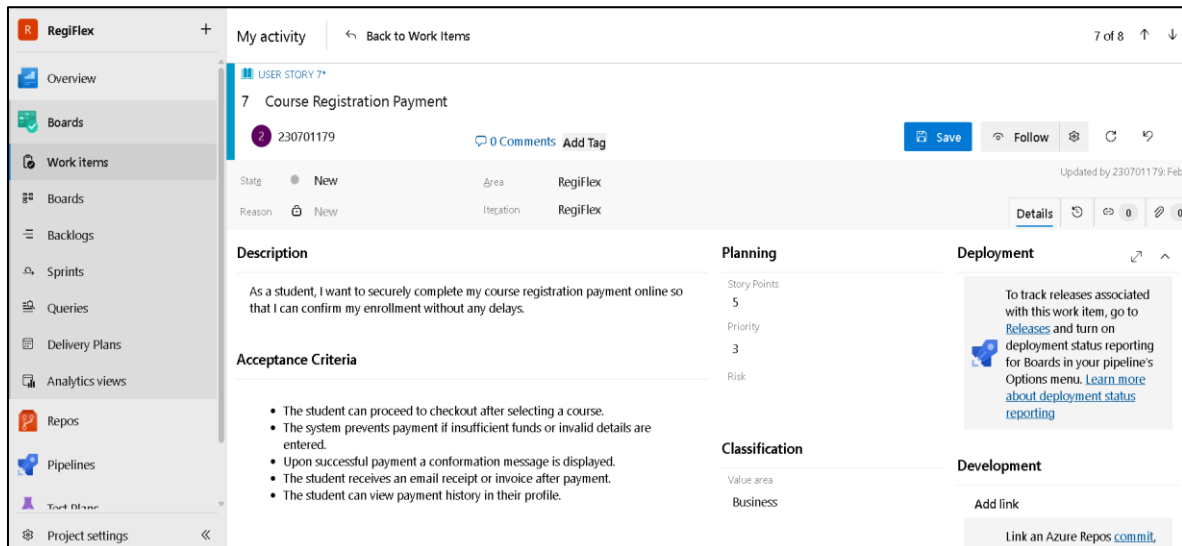


## 9. To manage user stories

- 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:

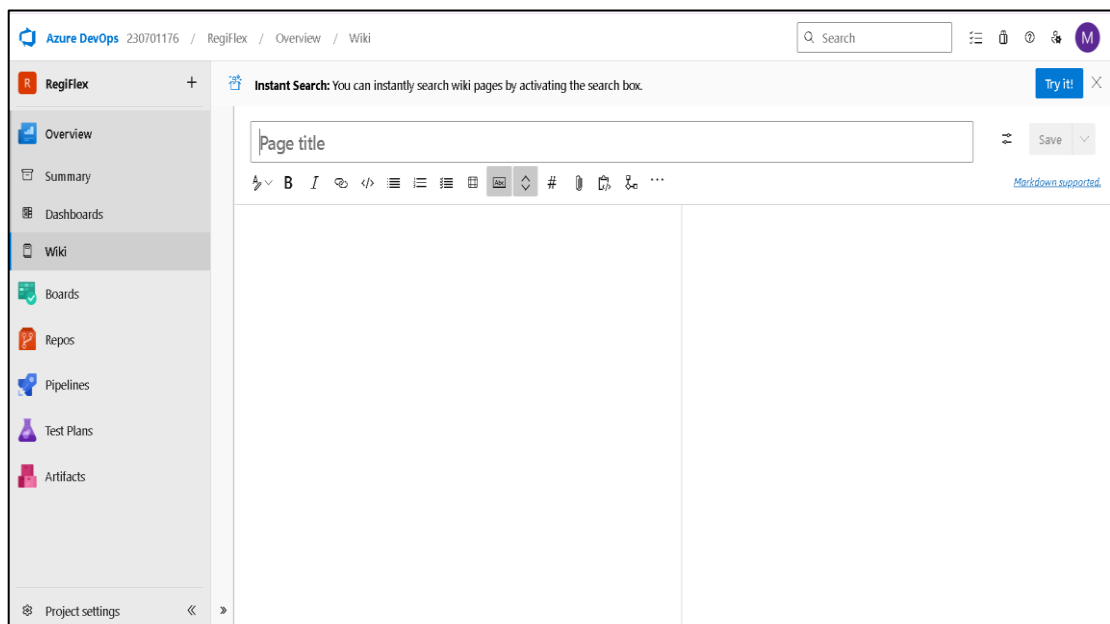
To design a Sequence Diagram by using Mermaid.js

#### 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 User
```

```
    participant App as Course Registration App
```

```
    participant DB as Database
```

```
    participant PG as Payment Gateway
```

```
    participant NS as Notification Service
```

User->>App: Open Course Registration App

App->>User: Redirect to Login Page

User->>App: Signup/Login with credentials

App->>DB: Validate credentials

DB-->>App: Credentials verified

App->>User: Login successful

User->>App: View available courses

App->>DB: Fetch course list

DB-->>App: Send course list

App->>User: Show course list

alt User searches for a specific course

User->>App: Search for courses

App->>DB: Fetch search results

DB-->>App: Send search results

App->>User: Show search results

end

User->>App: Select a course

App->>DB: Fetch course details (availability, fees, mentor names)

DB-->>App: Send course details

App->>User: Show course details

User->>App: Register in Course

App->>DB: Check seat availability

DB-->>App: Return seat status

alt Seats Available

App->>DB: Verify user eligibility

DB-->>App: Eligibility confirmed

App->>DB: Register user in course

DB-->>App: Confirm registration

App->>User: Registration successful

else Course Full

App->>DB: Fetch alternative courses

DB-->>App: Send list of alternative courses

App->>User: Show alternative courses

end

User->>App: Proceed to Payment

App->>PG: Make payment

PG-->>App: Payment successful

App->>NS: Send confirmation notification

NS-->>App: Notification sent  
App->>User: Notify Payment Success

User->>App: Access course materials and timetable  
App->>DB: Fetch course-related resources  
DB-->>App: Send materials  
App->>User: Provide access to course content:::

## 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+)

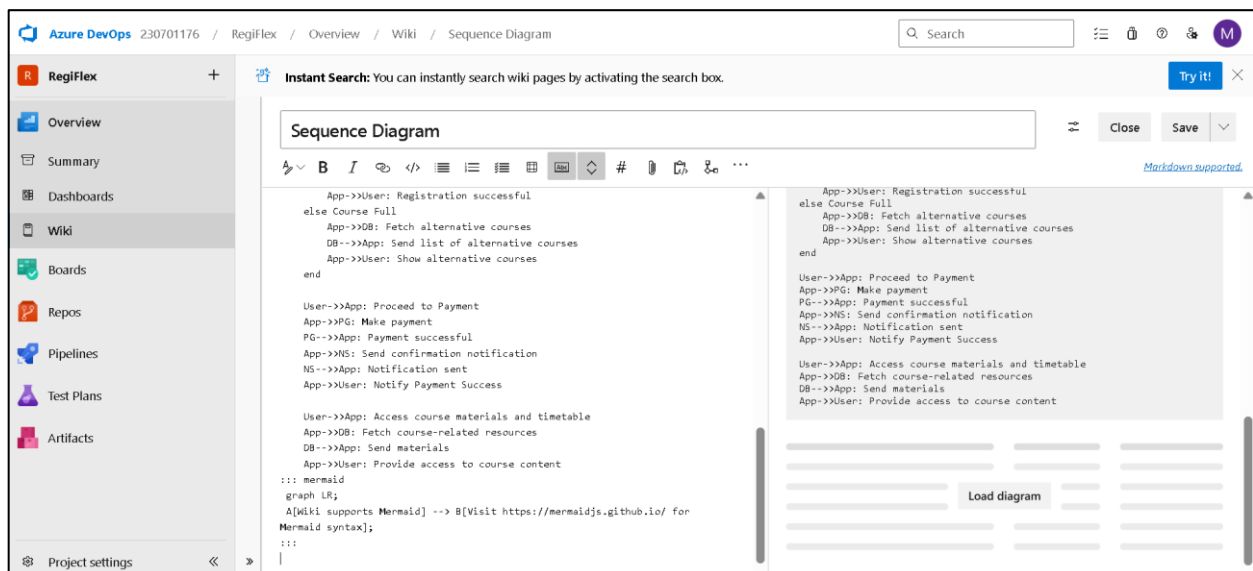
<<-->> Dotted line with bidirectional arrowheads (v11.0.0+)

-x Solid line with a cross at the end

--x Dotted line with a cross at the end

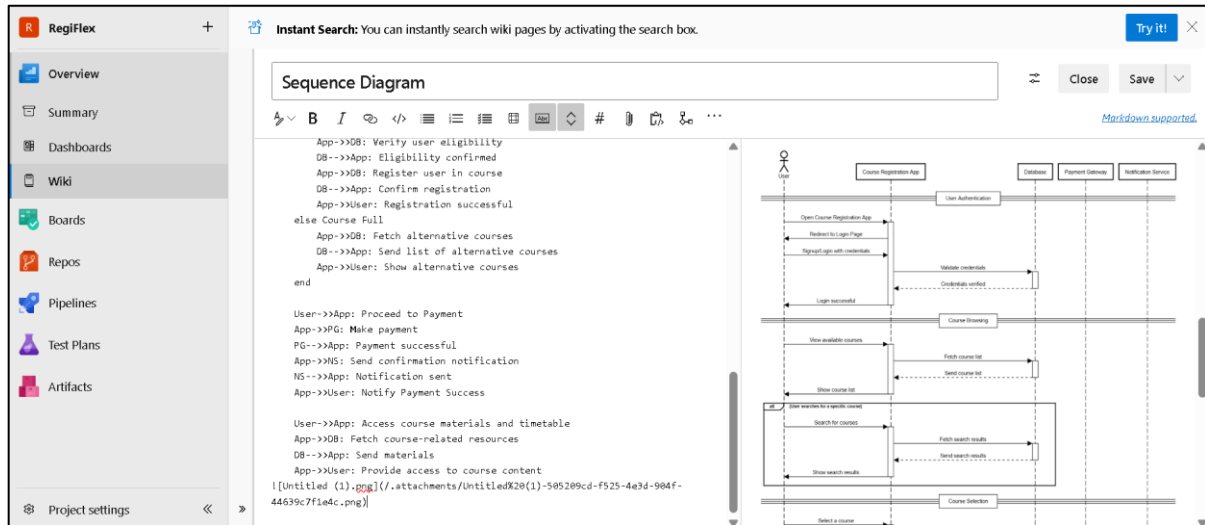
-) Solid line with an open arrow at the end (async)

--) Dotted line with an open arrow at the end (async)





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



#### Result:

The sequence diagram was drawn successfully.

## EX NO. 6

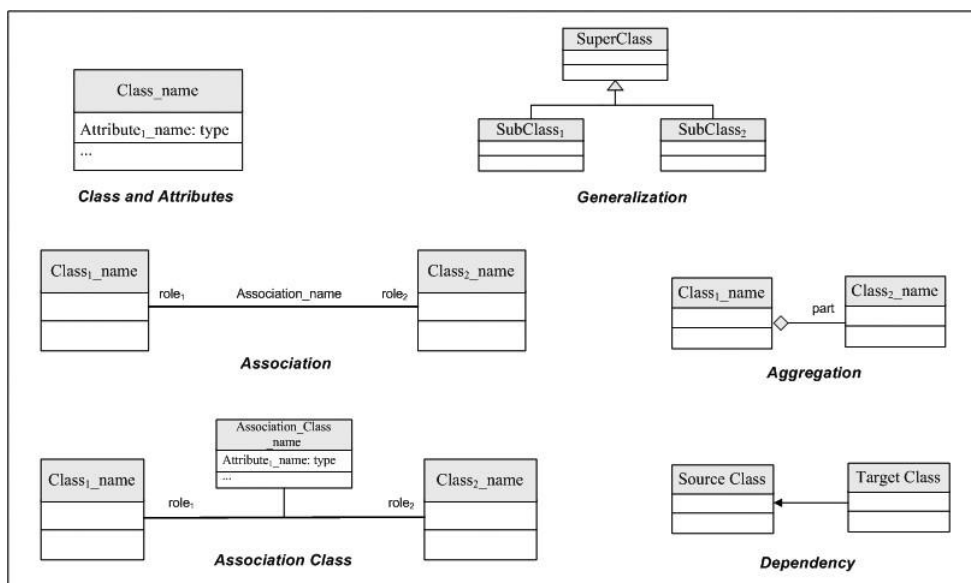
### CLASS DIAGRAM

#### AIM :-

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

#### THEORY:

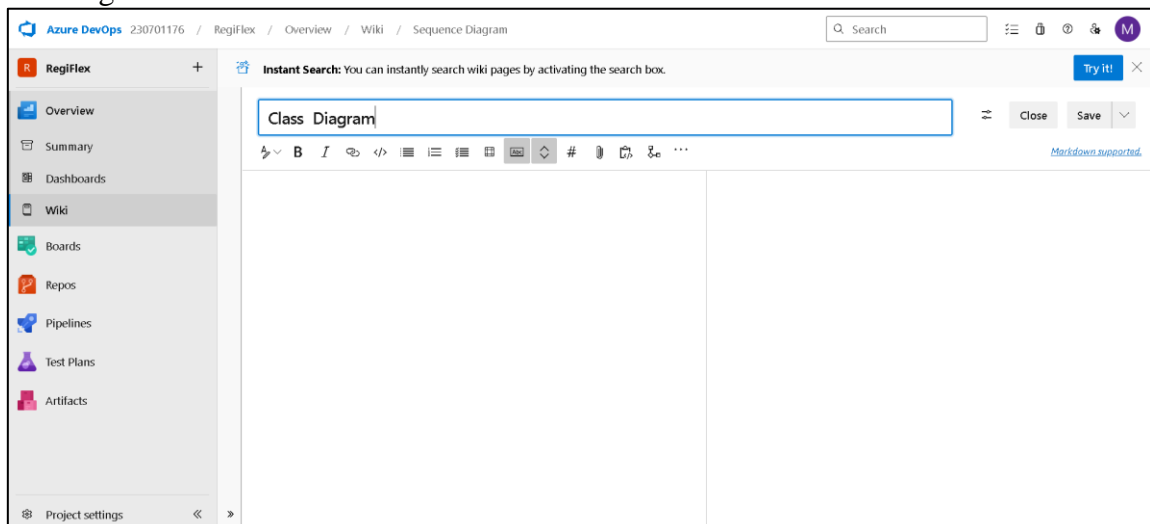
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



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

```
---
title: Class Diagram
---

classDiagram

class USERS {

    +int userID

    +string name

    +string email

    +string password

    +string role

    +EditProfile()

    +CreateProfile()

    +DeleteProfile()

}

class STUDENT {

    +int StudentID

    +int userID

    +int enrollmentID

    +DO_Enrollment()

    +Dropcourse()

}

class ADMIN {

    +int adminID

    +int userID

    +ManageCourse()

    +EditCourse()

}

class INSTRUCTOR {

    +int instructorID

    +int userID
```

```
+ViewCourse()

+ViewStudents()

}

class COURSES {

+int courseID

+string courseName

+string description

+int instructorID

+int availableSeats

+float fee

+ViewSchedule()

+SearchCourses()

+GetRecommendations()

}

class WAITLIST {

+int WaitlistID

+int userID

+int CourseID

+CheckWaitlist()

+UpdateWaitlist()

}

class PAYMENT {

+int paymentID

+float amount

+date paymentDate

+string paymentTo

+paymentGateway()

+CheckStatus()

+Transactions()

}
```

```

class NOTIFICATIONS {

    +int notificationID

    +string recipient

    +date notificationDate

    +int Time

    +Viewmessage()

    +Sendnotification()

}

```

```

class LEADERBOARD {

    +int courseID

    +int StudentID

    +int Rank

    +getTopStudents()

    +ViewRank()

}

```

USERS "1" --> "1" STUDENT : logs in

USERS "1" --> "1" ADMIN : logs in

USERS "1" --> "1" INSTRUCTOR : logs in

USERS "1" --> "\*" PAYMENT : pays

USERS "1" --> "\*" NOTIFICATIONS : receives

USERS "1" --> "\*" COURSES : enrolls

COURSES "1" --> "\*" LEADERBOARD : ranks

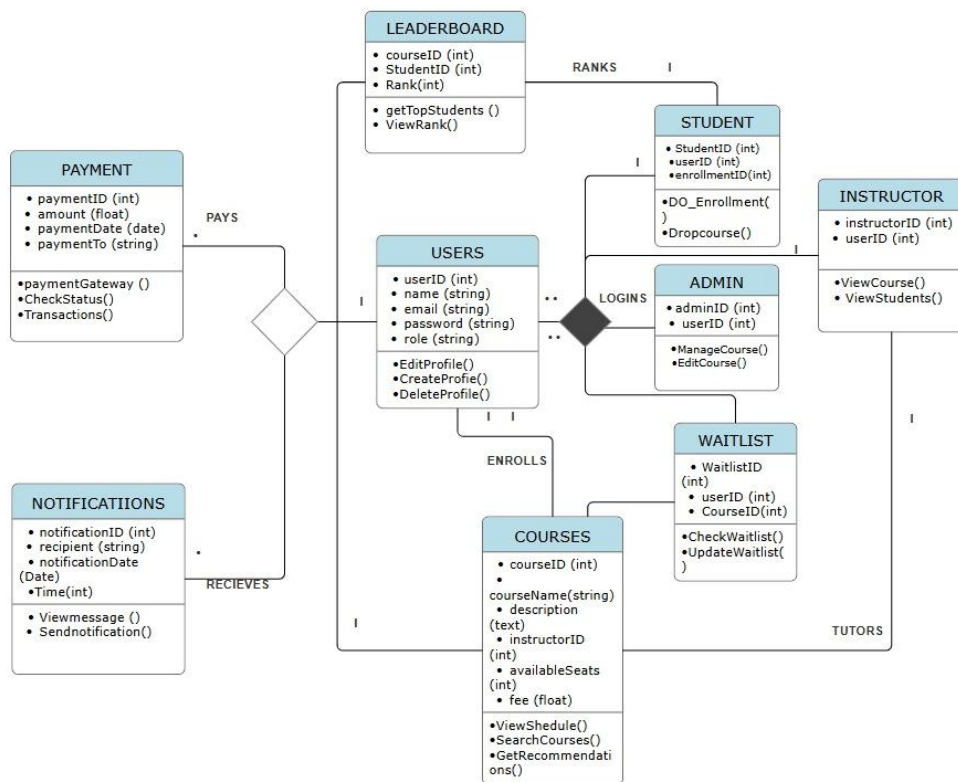
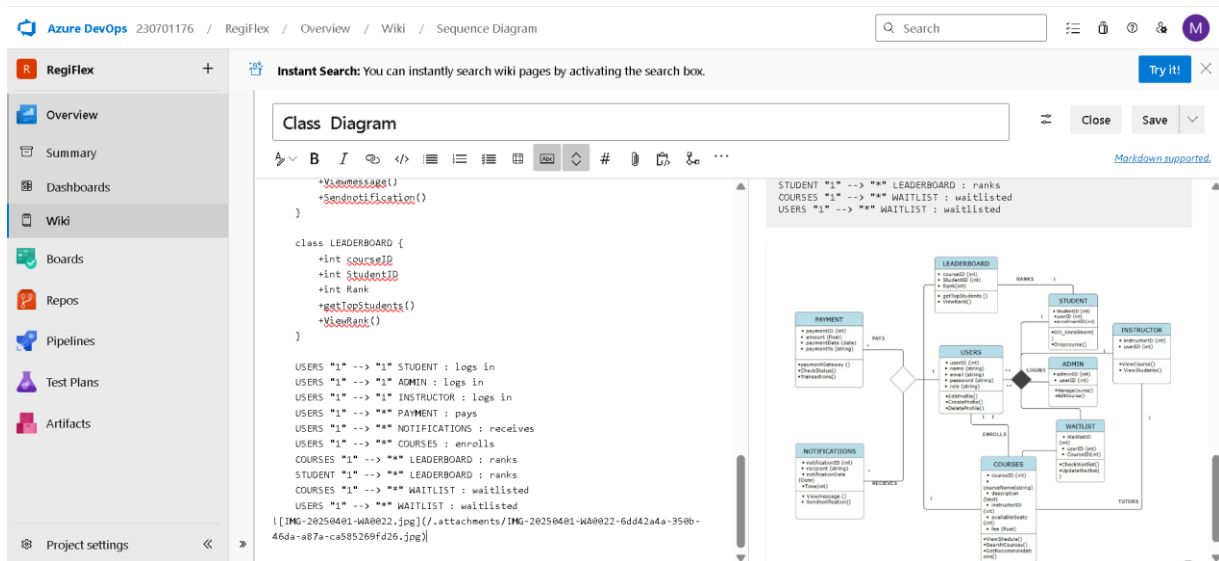
STUDENT "1" --> "\*" LEADERBOARD : ranks

COURSES "1" --> "\*" WAITLIST : waitlisted

USERS "1" --> "\*" WAITLIST : waitlisted

### 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**

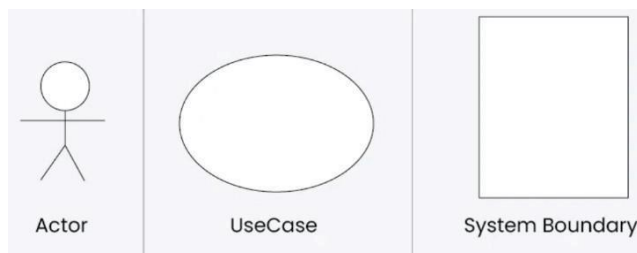
### **AIM:**

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

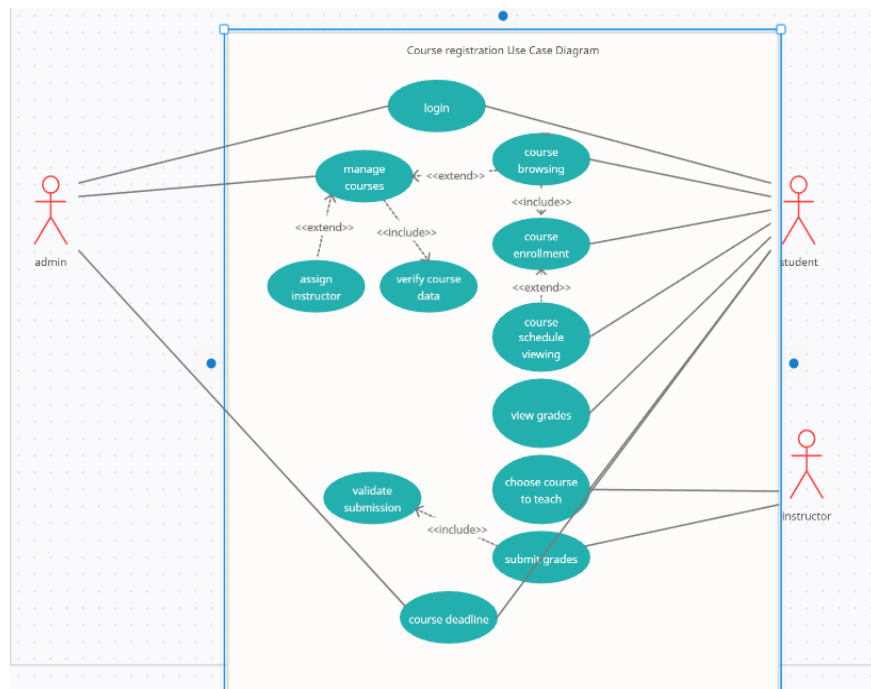
- Open Draw.io (diagrams.net).
- Click "Create New Diagram" and select "Blank" or "UML Use Case" template.
- Add Actors (Users, Admins, External Systems) from the UML section.
- Add Use Cases (Functionalities) using ellipses.
- Connect Actors to Use Cases with lines (solid for direct interaction, dashed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

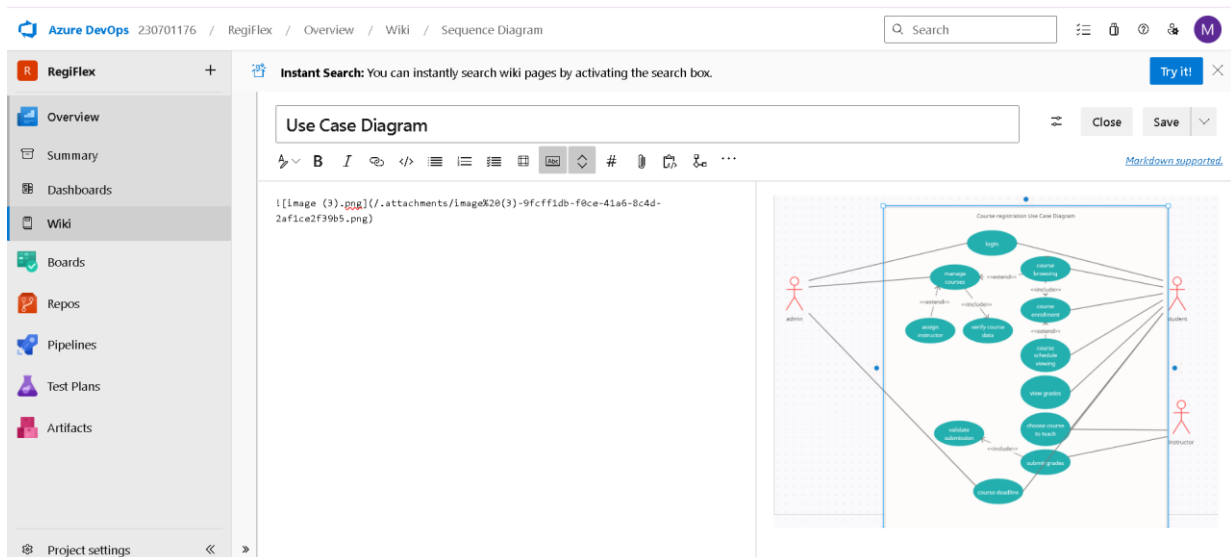
- Open Azure DevOps and go to your project.
- Navigate to Wiki (Project > Wiki).
- Click "Edit Page" or create a new page.
- Drag & Drop the exported PNG/JPG image.

- 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.



#### Result:

The use case diagram was designed successfully



## EX NO. 8



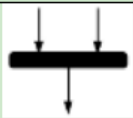








### ACTIVITY DIAGRAM

#### 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 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 230701176 / RegiFlex / Overview / Wiki / Sequence Diagram

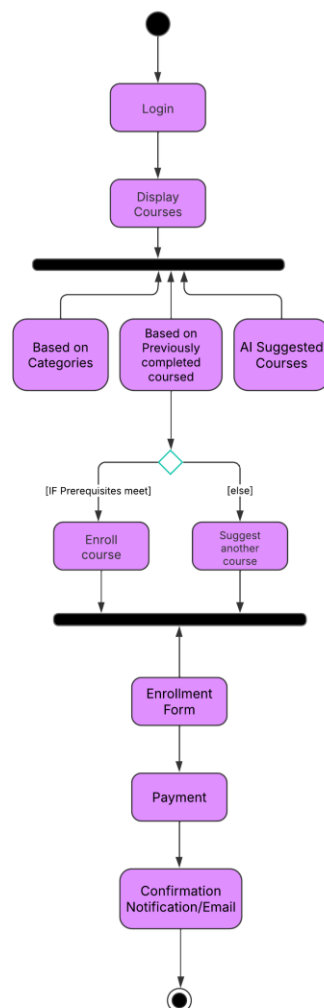
Instant Search: You can instantly search wiki pages by activating the search box. Try it!

Activity Diagram

! [Activity diagram (2).png] (/attachments/Activity%20diagram%20(2)-29541c22-122f-4786-b2b7-cf242b415c2f.png)

```

graph TD
    Start(( )) --> Login[Login]
    Login --> Display[Display Courses]
    Display --> Selection
    Selection --> BasedOnCategories[Based on Categories]
    Selection --> BasedOnPreviouslyCompleted[Based on Previously completed courses]
    Selection --> AISuggested[AI Suggested Courses]
    BasedOnCategories --> Selection
    BasedOnPreviouslyCompleted --> Selection
    AISuggested --> Selection
    Selection --> Decision{ }
    Decision -- "[IF Prerequisites meet]" --> Enroll[Enroll course]
    Decision -- "[else]" --> Suggest[Suggest another course]
    Enroll --> Selection
    Suggest --> Selection
    Selection --> EnrollmentForm[Enrollment Form]
    EnrollmentForm --> Payment[Payment]
    Payment --> Confirmation[Confirmation Notification/Email]
    Confirmation --> End(( ))
  
```



**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.

##### ARCHITECTURE SYMBOL OVERVIEW



#### Procedure

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

Azure DevOps 230701176 / RegiFlex / Overview / Wiki / Sequence Diagram

Instant Search: You can instantly search wiki pages by activating the search box. [Try it!](#)

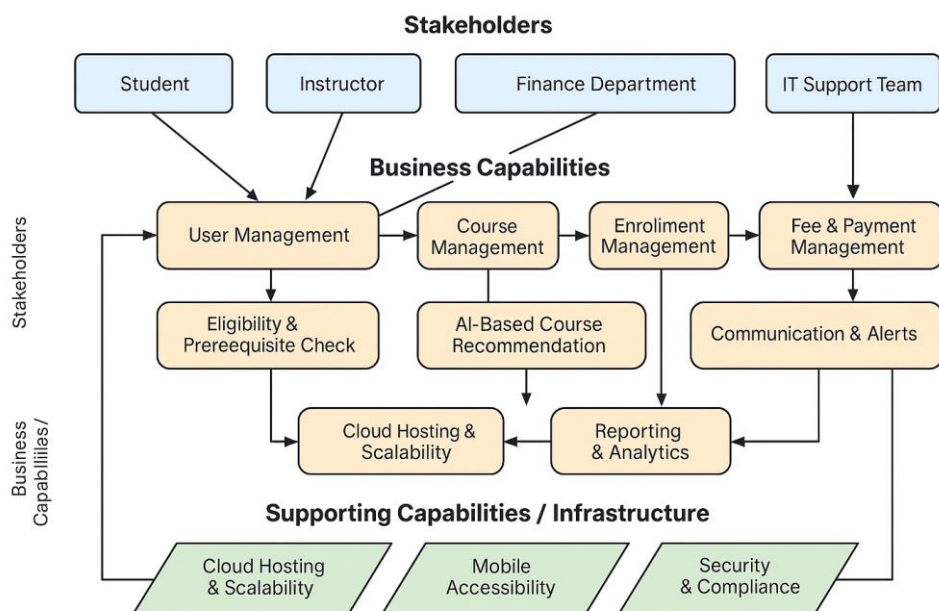
### Architecture Diagram

! [business\_architecture.png] (/attachments/business%20architecture-279d2d46-5948-4e18-a8bc-8d226af878f5.png)

The diagram illustrates the architecture of the RegiFlex system. It is organized into three main layers:

- Stakeholders:** Includes Student, Instructor, Finance Department, and IT Support Team.
- Business Capabilities:** Includes User Management, Course Management, Enrolment Management, Fee & Payment Management, Eligibility & Prerequisite Check, AI-Based Course Recommendation, and Communication & Alerts.
- Supporting Capabilities / Infrastructure:** Includes Cloud Hosting & Scalability, Mobile Accessibility, and Security & Compliance.

Arrows indicate the flow of data and interactions between these components. For example, Stakeholders interact with Business Capabilities, which in turn interact with the Supporting Capabilities / Infrastructure.



## Result:

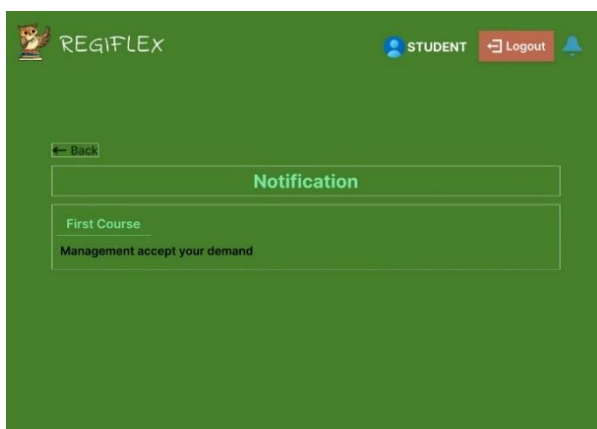
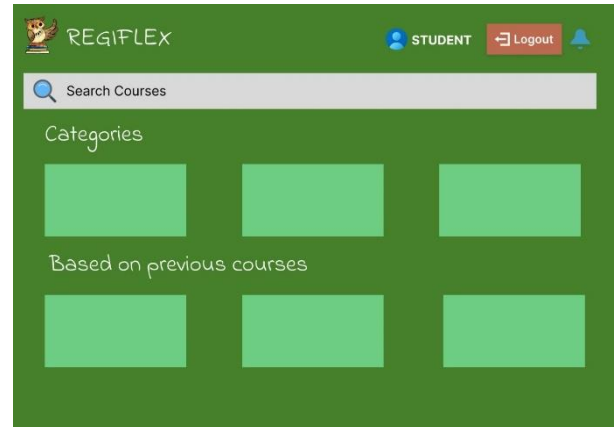
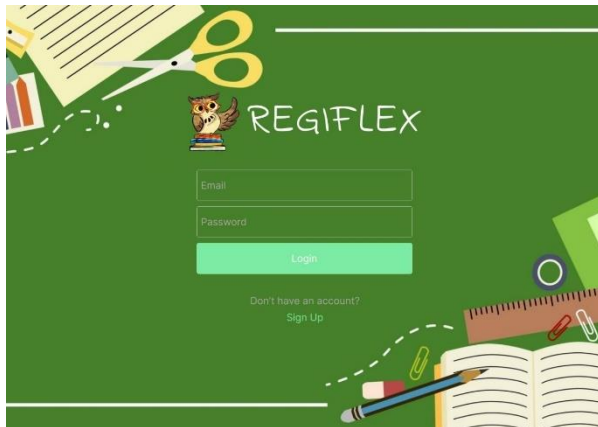
The architecture diagram was designed successfully

## EX NO. 10

### USER INTERFACE

#### AIM:

Design User Interface for our project.



#### 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.

