

**RAJALAKSHMI ENGINEERING COLLEGE**  
**RAJALAKSHMI NAGAR, THANDALAM - 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CS23432  
SOFTWARE CONSTRUCTION**

**Laboratory Record Note Book**

Name : .....

Year / Branch / Section : .....

Register No. : .....

Semester : .....

Academic Year : .....



**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**  
**RAJALAKSHMI NAGAR, THANDALAM – 602 105**

**BONAFIDE CERTIFICATE**

NAME \_\_\_\_\_ REGISTER NO. \_\_\_\_\_

ACADEMIC YEAR 2024-25 SEMESTER- IV BRANCH: B. Tech Information Technology [AD/AE]. This Certification is the Bonafide record of work done by the above student in the **CS23432- Software Construction** Laboratory during the year 2024-2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on \_\_\_\_\_

Internal Examiner

External Examiner

<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: “-“

## **EX NO: 1**

### **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.

---

### **PROBLEM STATEMENT**

**Aim:**

To prepare PROBLEM STATEMENT for Student data management system.

**Problem Statement:**

**Student Data Management System**

Managing student data manually is time-consuming, error-prone, and inefficient, especially as the number of students increases. Institutions face difficulties in maintaining accurate records of student profiles, enrollments, course details, performance tracking, and document storage. A centralized Student Data Management System is needed to automate these processes, improve data accuracy, ensure secure access, and make information easily available to students, faculty, and administrators.

**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.

---

## 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]."

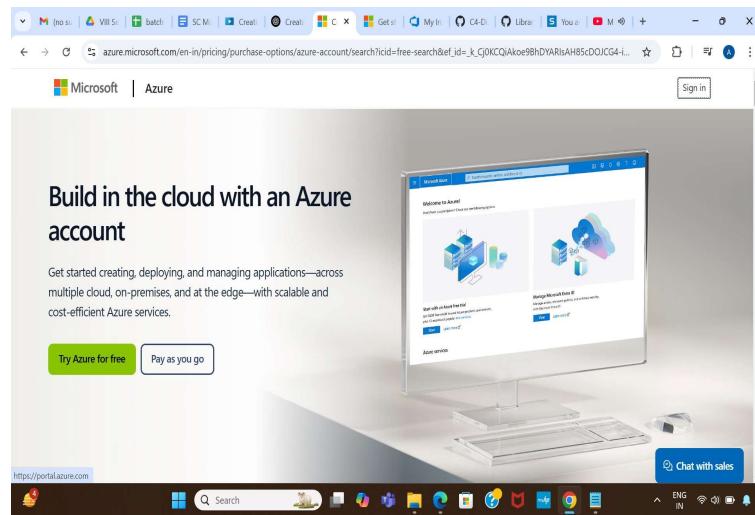
### Procedure:

1. Open your web browser and go to the Azure website:

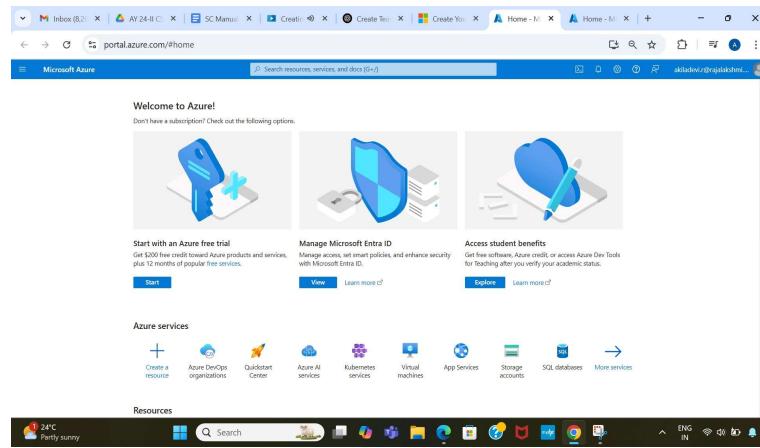
<https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.

2. If you don't have a Microsoft account, you can sign up for

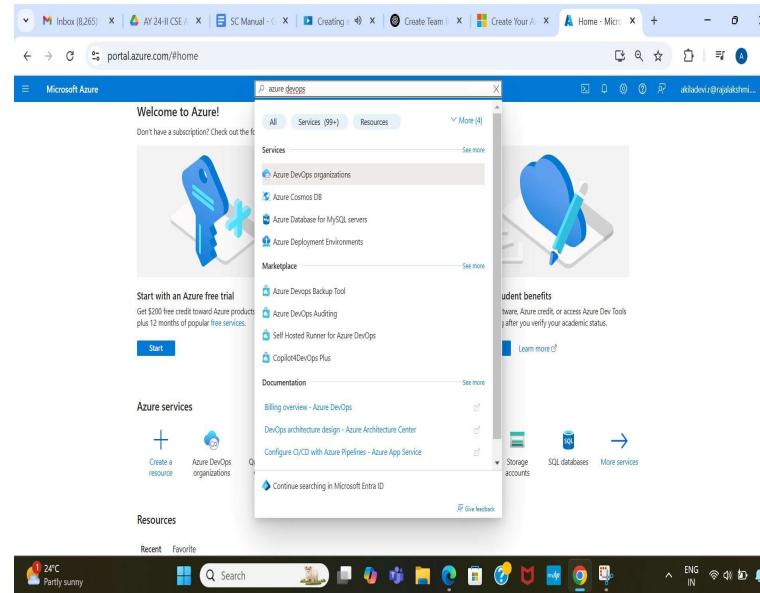
<https://signup.live.com/?lic=1>



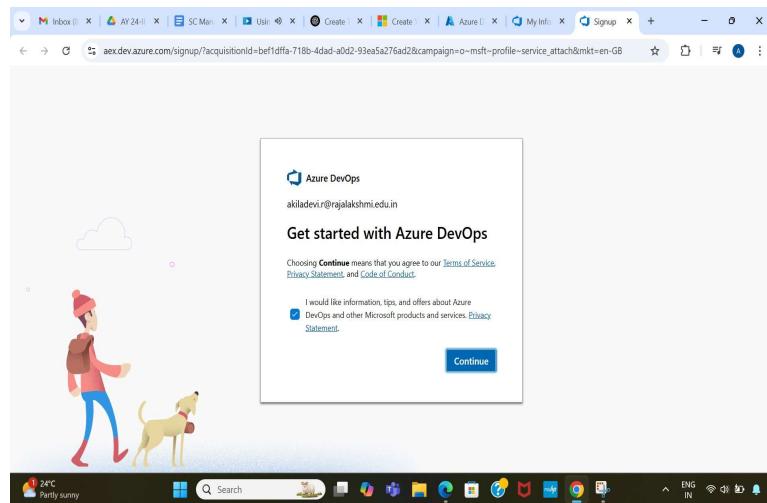
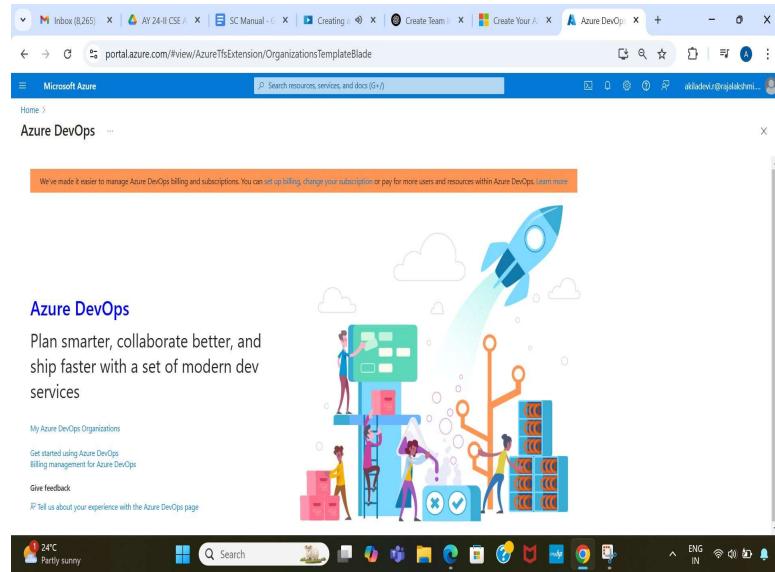
### 3. Azure home page

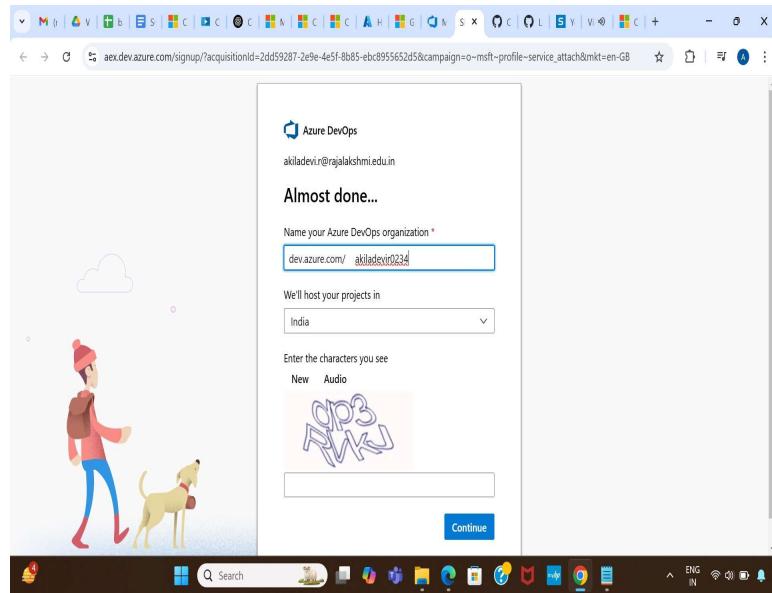


### 4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



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.





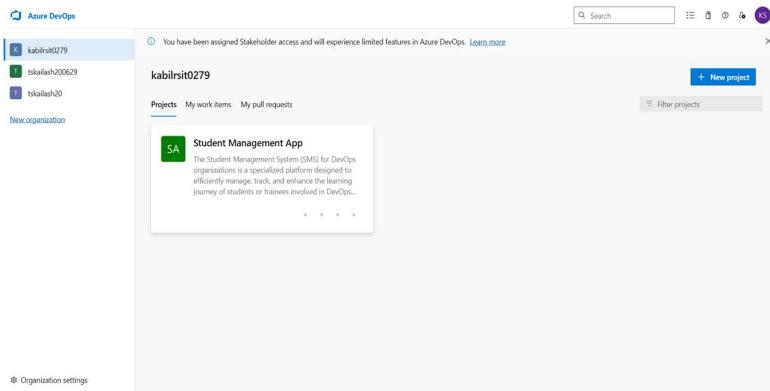
## 6. Create the First Project in Your Organization

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.

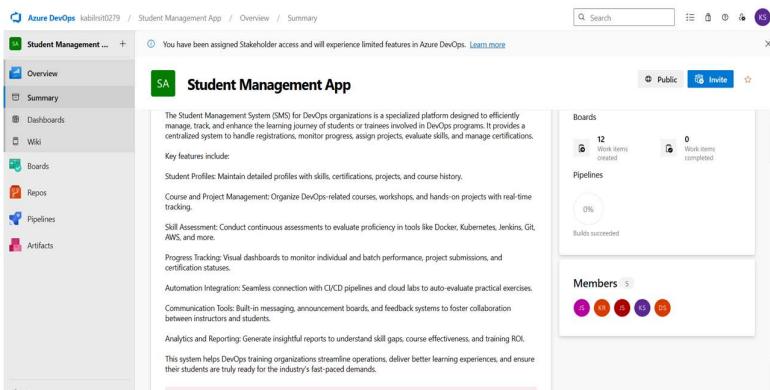
- i. On the organization's **Home page**, click on the **New Project** button.
- ii. Enter the project name, description, and visibility options:
  - **Name:** Choose a name for the project (e.g., **LMS**).
  - **Description:** Optionally, add a description to provide more context about the project.
  - **Visibility:** Choose whether you want the project to be **Private** (accessible only to those invited) or **Public** (accessible to anyone).
- iii. Once you've filled out the details, click **Create** to set up your first project.

A screenshot of the 'Create new project' dialog box. It has fields for 'Project name \*' and 'Description'. Below these, there's a 'Visibility' section with two options: 'Public' and 'Private'. The 'Public' option is described as allowing anyone on the internet to view the project. The 'Private' option is described as allowing only people given access to view the project. A note at the bottom states that public projects are disabled for the organization. At the bottom right are 'Cancel' and 'Create' buttons.

- 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.

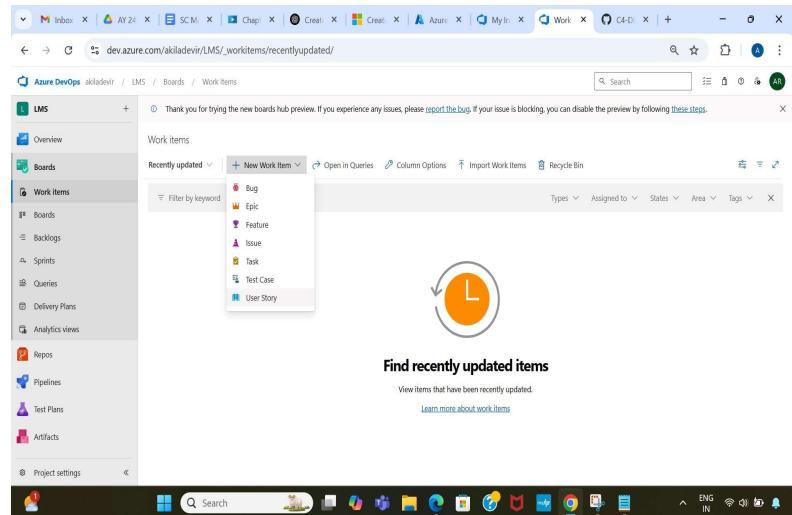


## 8. Project dashboard



## 9. To manage user stories

- 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

The screenshot shows the Azure DevOps Work items screen for a user story titled '21 Able to receive automatic notifications'. The story is in the 'New' state and is assigned to the 'Student Management App' area. The 'Acceptance Criteria' section lists three points: 1. Notifications are viewable in the in-app notification center. 2. Students can enable or disable email alerts in their settings. 3. All past notifications are accessible under "Notification History". The 'Planning' section shows story points (2), priority (2), and risk (R). The 'Deployment' section includes a tooltip about tracking releases and deployment status reporting. The 'Classification' section shows value area (Business) and the 'Development' section shows add link and GitHub integration options.

## Result:

The user story was written successfully.

## EX NO: 5

### SEQUENCE DIAGRAM

#### Aim:

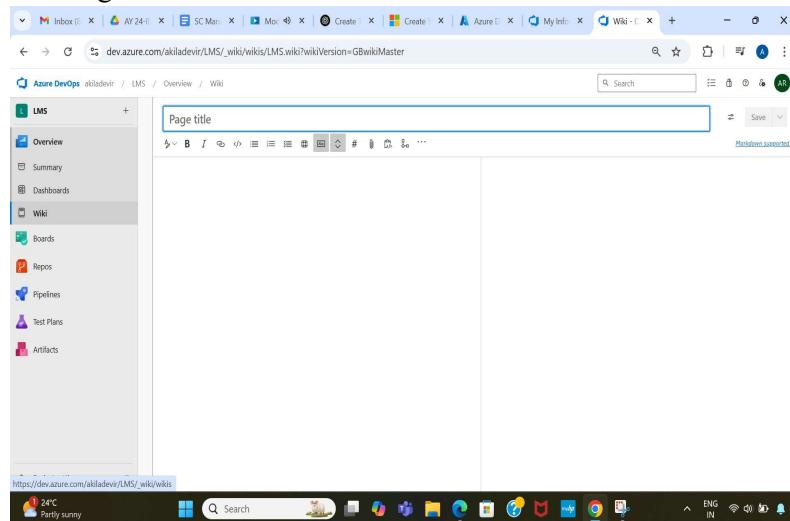
To design a Sequence Diagram by using Mermaid.js

#### 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

```
sequenceDiagram
    participant Instructor
    participant System
    participant StudentDatabase
    participant CourseDatabase
```

Instructor->>System: Login

System->>Instructor: Authenticate & Show Dashboard

Instructor->>System: Select Course & Student

System->>CourseDatabase: Fetch Course Details

System->>StudentDatabase: Fetch Student Record

Instructor->>System: Enter Grade

System->>StudentDatabase: Save Grade  
StudentDatabase-->>System: Confirm Grade Saved

System-->>Instructor: Grade Submission Success:::

### 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+)

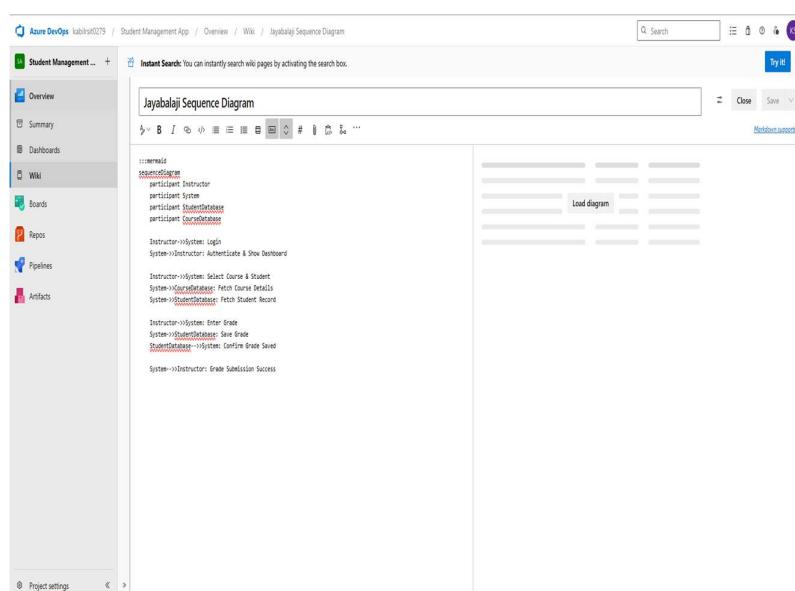
<<->>> Dotted line with bidirectional arrowheads (v11.0.0+)

-x Solid line with a cross at the end

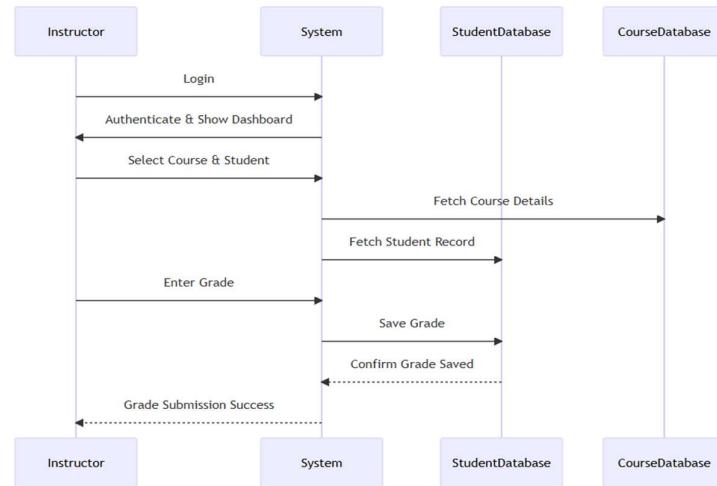
--x Dotted line with a cross at the end

-) 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

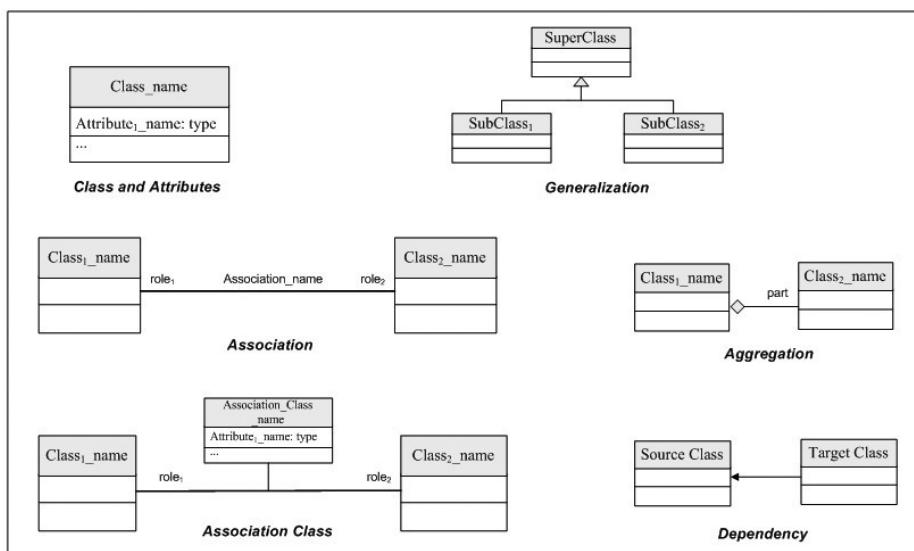
### CLASS DIAGRAM

#### Aim:

To draw a sample class diagram for Student Management 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.

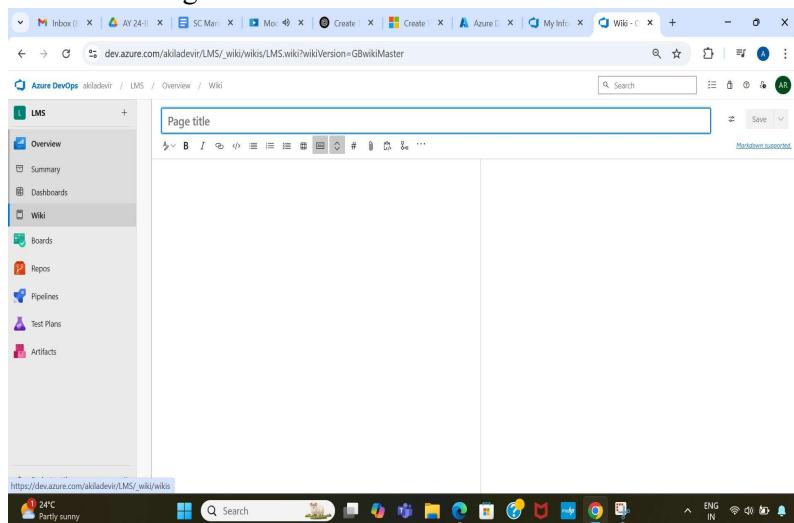


#### Notations in class diagram

#### Procedure:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu



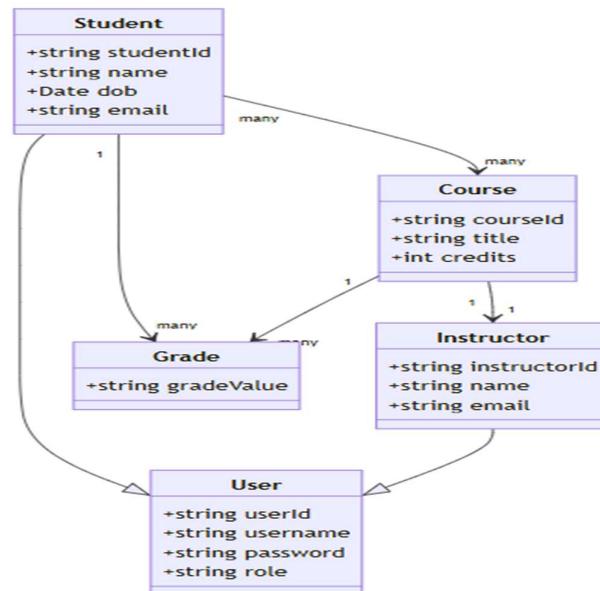
3. Write code for drawing class diagram and save the code

Class Diagram

```

class Student {
    +string studentId
    +string name
    +Date dob
    +string email
}
class Course{
    +string courseId
    +string title
    +int credits
}
class Instructor {
    +string instructorId
    +string name
    +string email
}
class User {
    +string userId
    +string username
    +string password
    +string role
}
%% Relationships
Student "1" --> "many" Grade
Course "1" --> "many" Grade
Student "many" --> "many" Course
Course "1" --> "1" Instructor
Student --> User
Instructor --> User

```



**Result:**

The use case diagram was designed successfully.

---

## EX NO: 7

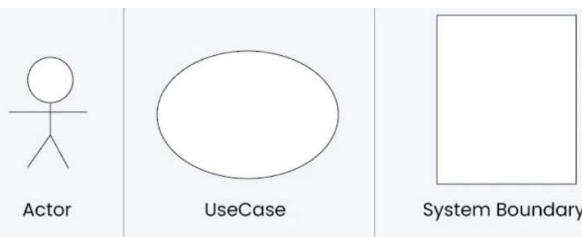
### USECASE DIAGRAM

#### 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](https://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

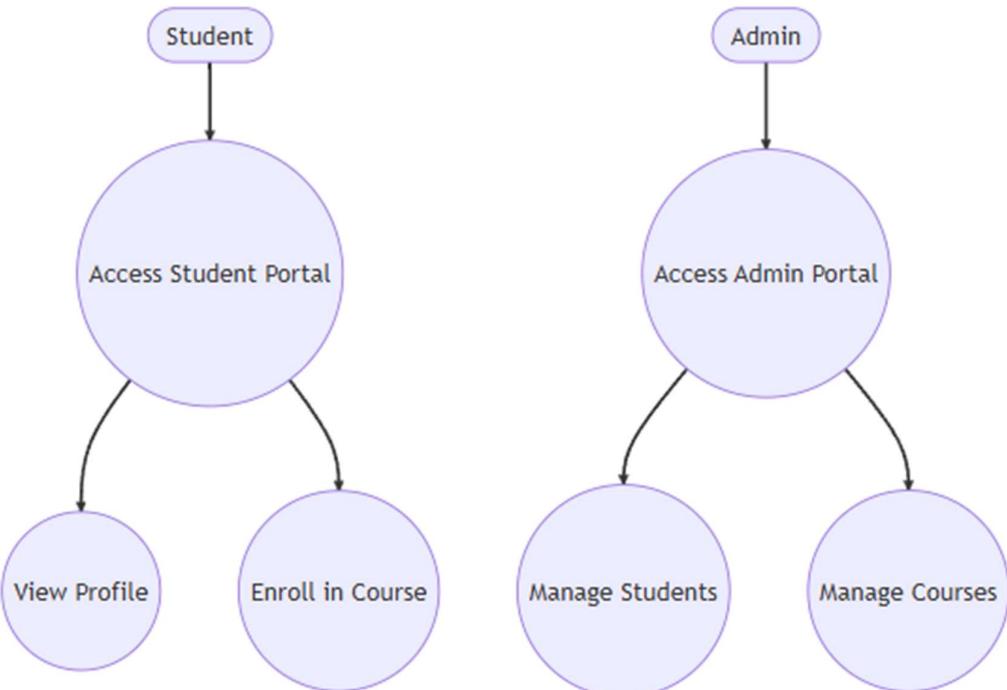
Option 1: Add to Azure DevOps Wiki

- 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.

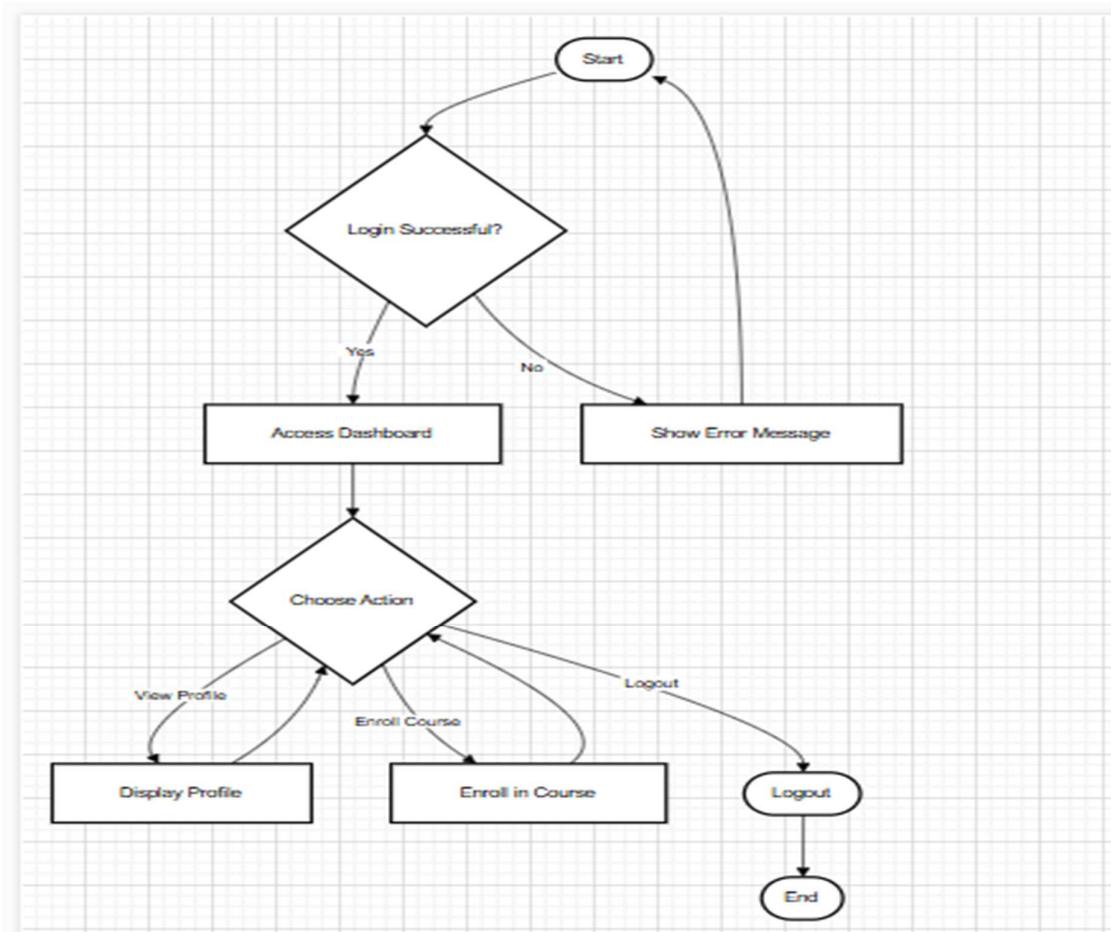
#### Theory:

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



**Result:**

The activity diagram was designed successfully

## EX NO. 9

### ARCHITECTURE DIAGRAM

#### Aim:

Steps to draw the Architecture Diagram using draw.io.

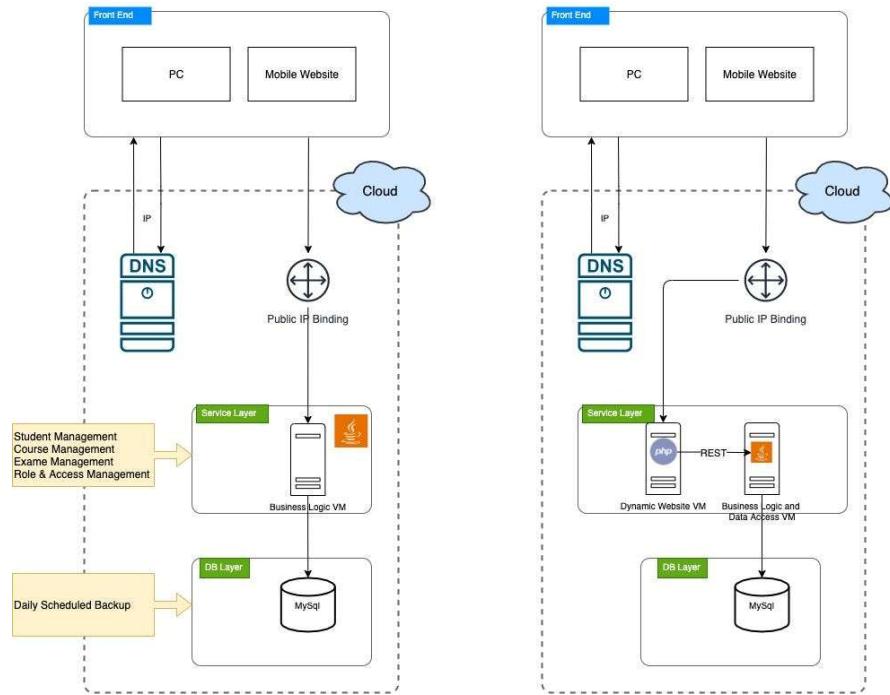
#### Theory:

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

## **EX NO. 10**

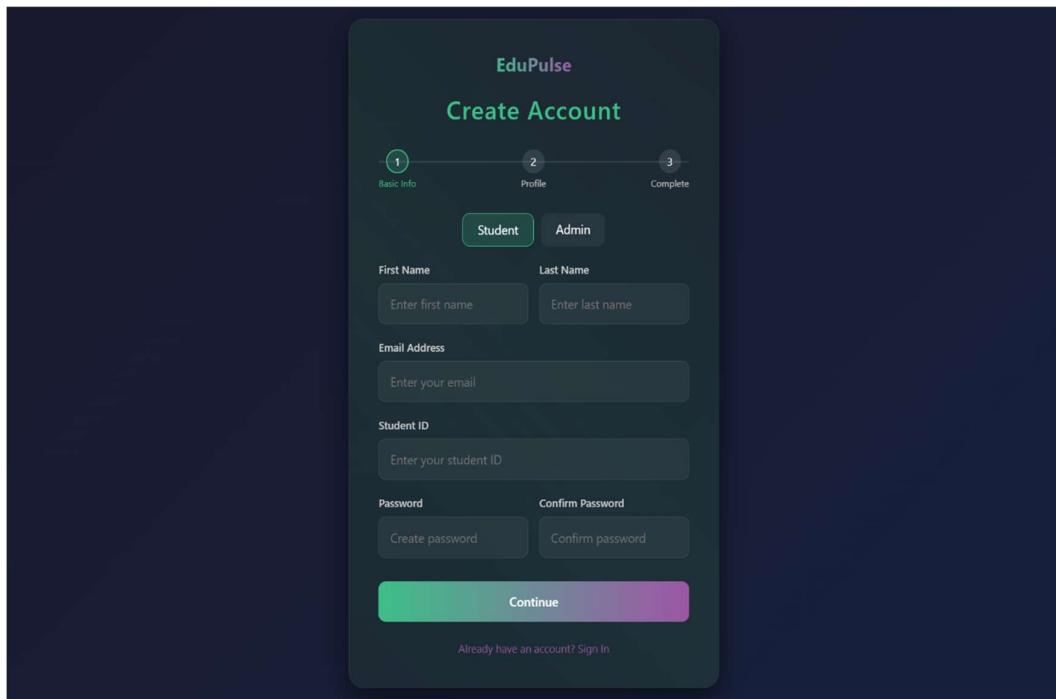
### **USER INTERFACE**

#### **Aim:**

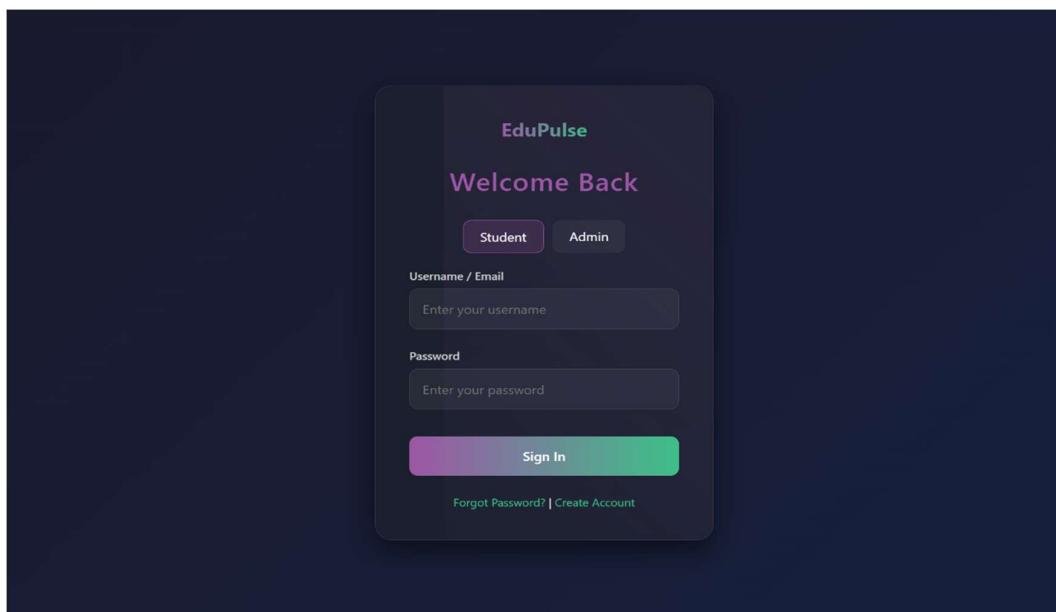
Design User Interface for the Student data Management System

#### **User interface:**

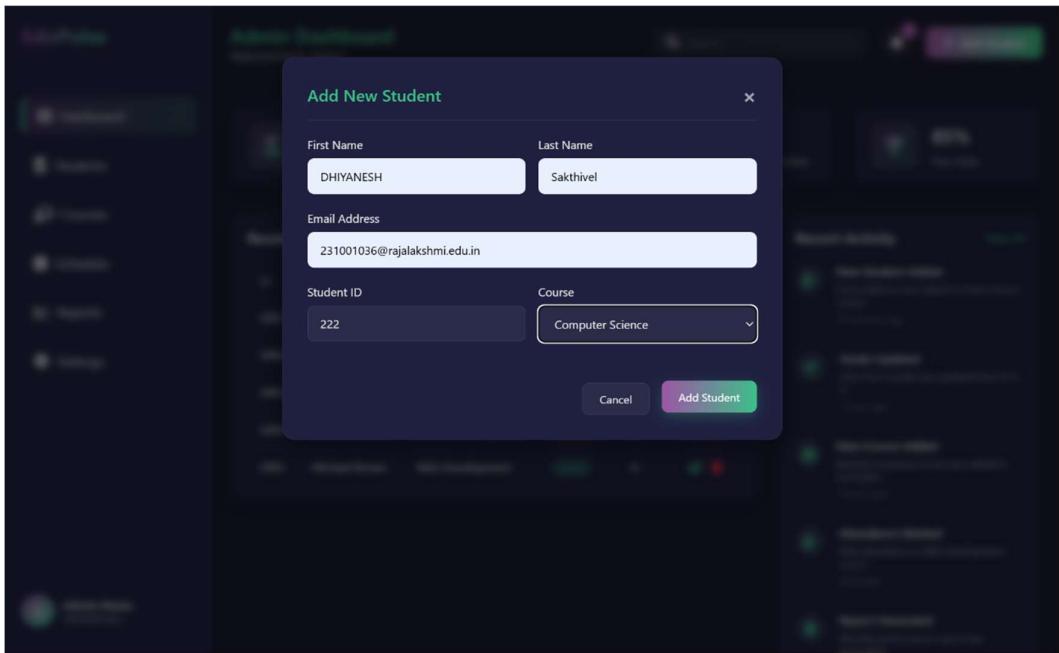
##### **1. Signup**



##### **2. Login**



### 3. Add Student



### 4. Admin Dashboard

A screenshot of the EduPulse Admin Dashboard. The sidebar includes links for Dashboard, Students, Courses, Schedule, Reports, and Settings. The main dashboard shows statistics: 1,250 Total Students, 42 Total Courses, 92% Attendance Rate, and 85% Pass Rate. Below these are sections for Recent Students (listing S001-S005) and Recent Activity (listing events like New Student Added, Grade Updated, New Course Added, Attendance Marked, and Report Generated).

## 5. Directory

```
task5
└── backend
    ├── models
    │   └── user.js
    ├── node_modules
    └── routes
        ├── auth.js
        └── protected.js
    ├── .env
    ├── package-lock.json
    ├── package.json
    └── server.js
└── frontend
    ├── home.html
    ├── index.html
    ├── login.html
    └── script.js
```

### 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):

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.

---

## **EXP NO : 12**

### **TESTING – TEST PLAN, TEST CASE, LOAD TESTING**

#### **Aim:**

To design and manage structured test plans and test cases in Azure DevOps for validating core user stories through both happy path and error scenarios and evaluate the performance of the application's endpoint by creating and executing load tests using Azure Load Testing.

#### **Procedure:**

##### **TEST CASE DESIGN PROCEDURE**

###### **1. Understand Core Features of the Application**

- Review requirement documents and user stories.
- Identify all main functionalities of the application.
- Ensure complete coverage of modules before test case creation.

###### **2. Define User Interactions**

- Determine common user behaviors based on application flow.
- Translate user actions into testable scenarios.
- Ensure each test case mimics a real user operation.

###### **3. Design Happy Path Test Cases**

- Create test cases for expected and correct user actions.
- Ensure each functionality works under normal conditions.
- Add these cases under the relevant Test Suite in Azure DevOps.

###### **4. Design Error Path Test Cases**

- Identify edge cases, invalid inputs, and system failures.
- Test how the system handles incorrect or unexpected behavior.
- Add these test cases to the same or a separate Test Suite in Azure DevOps.

###### **5. Break Down Steps and Expected Results**

- Write step-by-step instructions in the "Steps" section of the test case.
- Provide expected results for each action.
- Ensure clarity for both manual execution and automation mapping.

## 6. Use Clear Naming and IDs

- Name test cases clearly using a defined naming convention (e.g., TC01, TC02, etc.).
- Ensure titles reflect the purpose of the test case.
- Azure DevOps auto-generates test case IDs for tracking.

## 7. Separate Test Suites

- Group test cases based on functionality (e.g., Login, Playlist, Recommendations).
- Use Static, Requirement-based, or Query-based suites in Azure DevOps.
- Improves traceability and execution flow.

## 8. Prioritize and Review

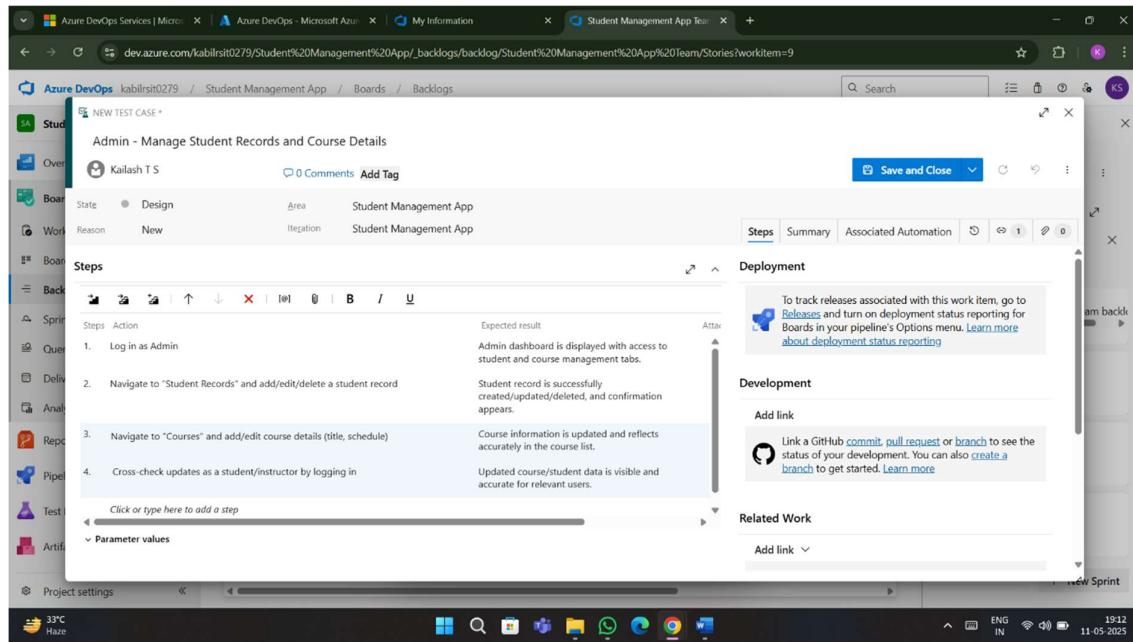
- Mark test cases with priority (High, Medium, Low).
- Review test cases for completeness and correctness.
- Ensure alignment with associated user stories or features.

### 1. New test plan

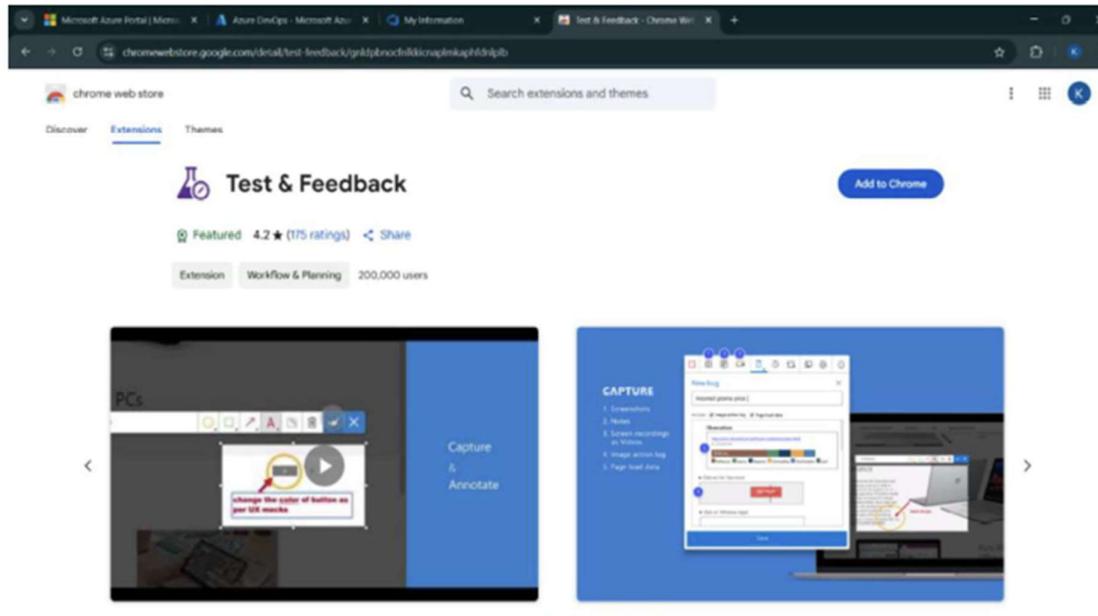
The screenshot shows the Azure DevOps Test Plans interface. On the left, there's a sidebar with options like Overview, Boards, Repos, Pipelines, Test Plans, Test plans, Progress report, Parameters, Configurations, Runs, and Artifacts. Under 'Test Plans', 'Test plans' is selected. In the center, a card for 'dummy test' is displayed, showing it was created on May 11 and 18, with a 100% run and 100% passed status. Below this, a 'Test Suites' section lists 'dummy test (5)'. On the right, the 'dummy test (ID: 24)' details page is shown, featuring tabs for Define, Execute, and Chart. The 'Define' tab is active, displaying a table titled 'Test Cases (5 items)' with columns for Title, Order, Test Case Id, Assigned To, and Status. The table contains five entries:

Title	Order	Test Case Id	Assigned To	Status
Student Registration - Create Account with Personal & Contact Information	1	22	Kabil RS	Defined
Student Browses and Enrolls in a Course Successfully	2	25	Kabil RS	Defined
Verify Admin Can Assign Instructor to Course	3	26	Kailash T S	Defined
Verify Admin Can Manage Student Records and Course Details	4	28	Kailash T S	Defined
Admin - Manage Student Records and Course Details	5	29	Kailash T S	Defined

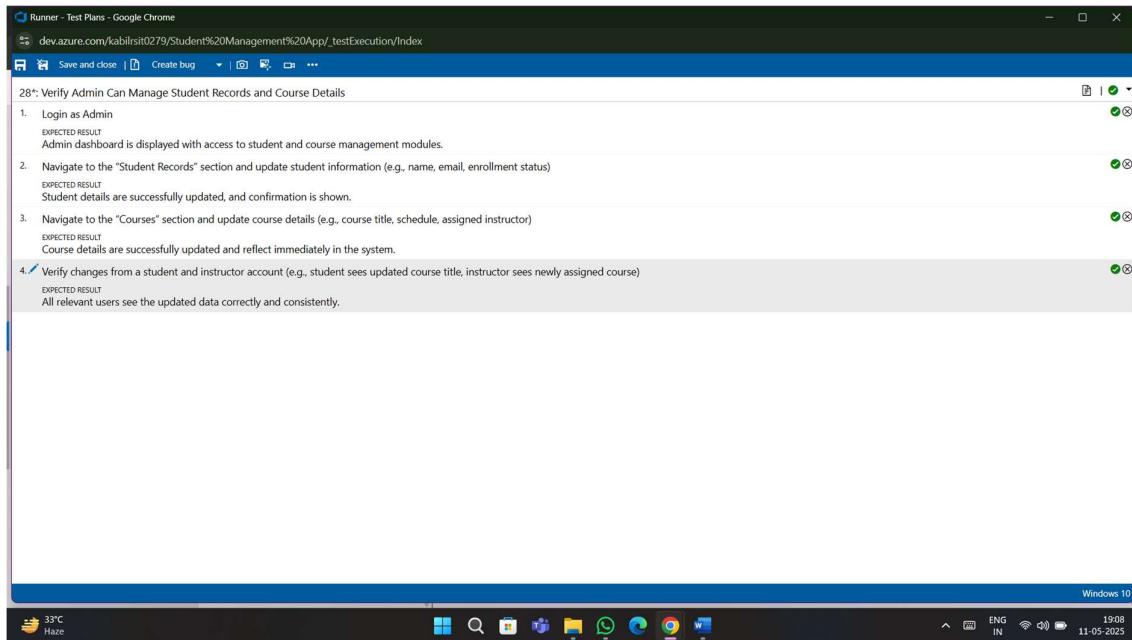
## 2. Test case



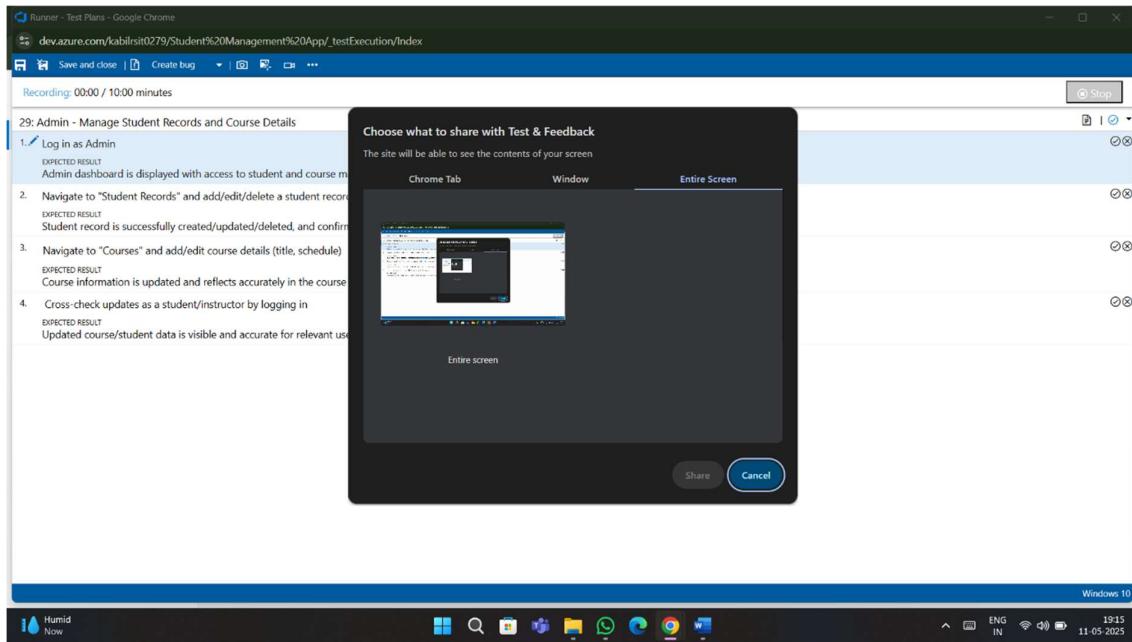
## 3. Installation of Test



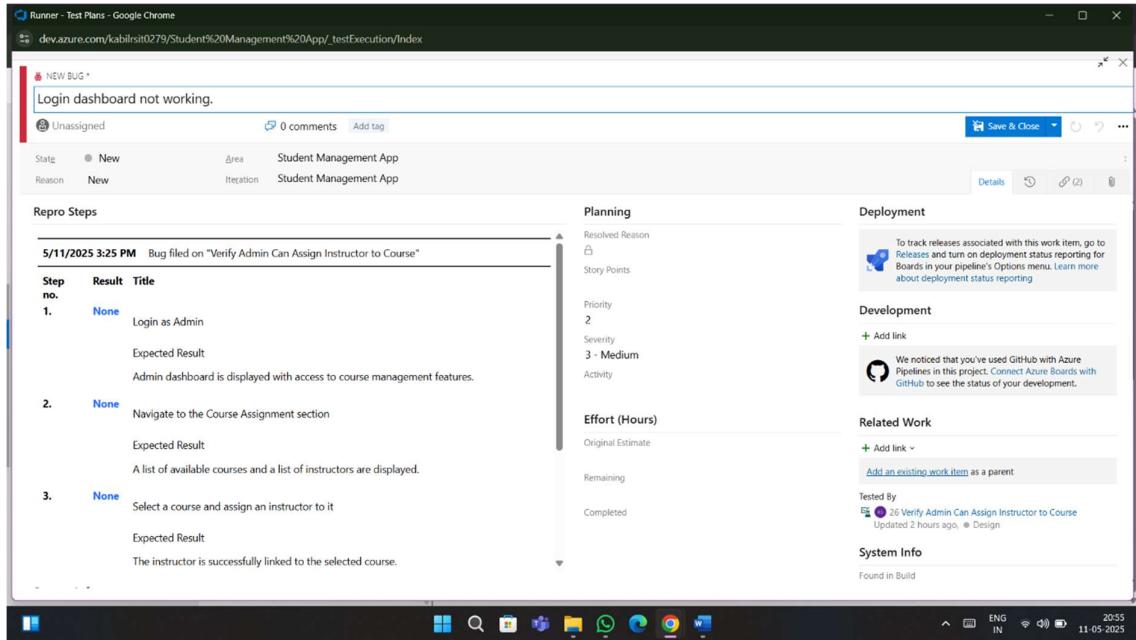
## 4. Running the Test Cases



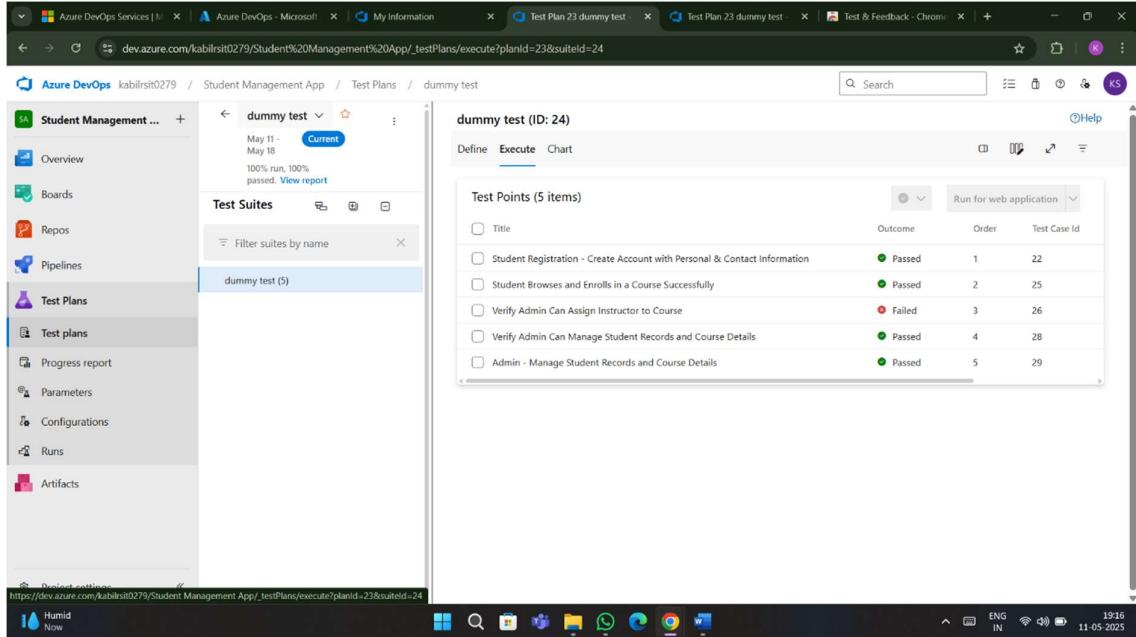
## 5. Recording the Test Cases



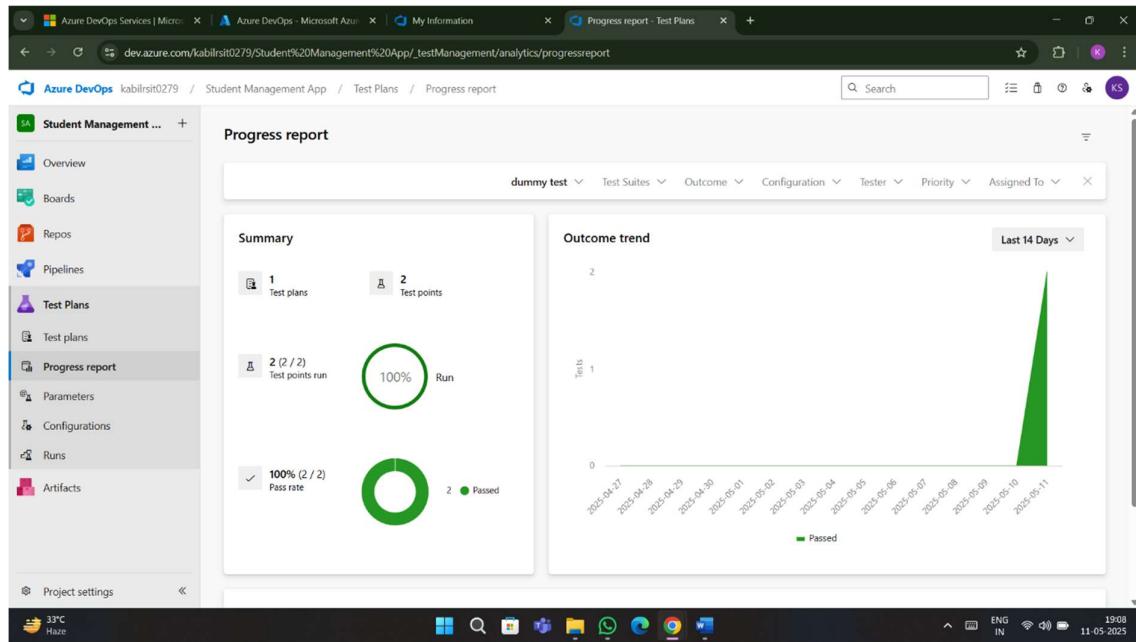
## 6. Creating Bugs



## 7. Test Case Results



## 8. Progress Report



## LOAD TESTING PROCEDURE :

### Steps to Create an Azure Load Testing Resource:

Before you run your first test, you need to create the Azure Load Testing resource:

#### 1. Sign in to Azure Portal

Go to <https://portal.azure.com> and log in.

#### 2. Create the Resource

- Go to Create a resource — Search for “Azure Load Testing”.
- Select Azure Load Testing and click Create.

#### 3. Fill in the Configuration Details

- Subscription: Choose your Azure subscription.
- Resource Group: Create new or select an existing one.
- Name: Provide a unique name (no special characters).
- Location: Choose the region for hosting the resource.

#### 4. (Optional) Configure tags for categorization and billing.

#### 5. Click Review + Create, then Create.

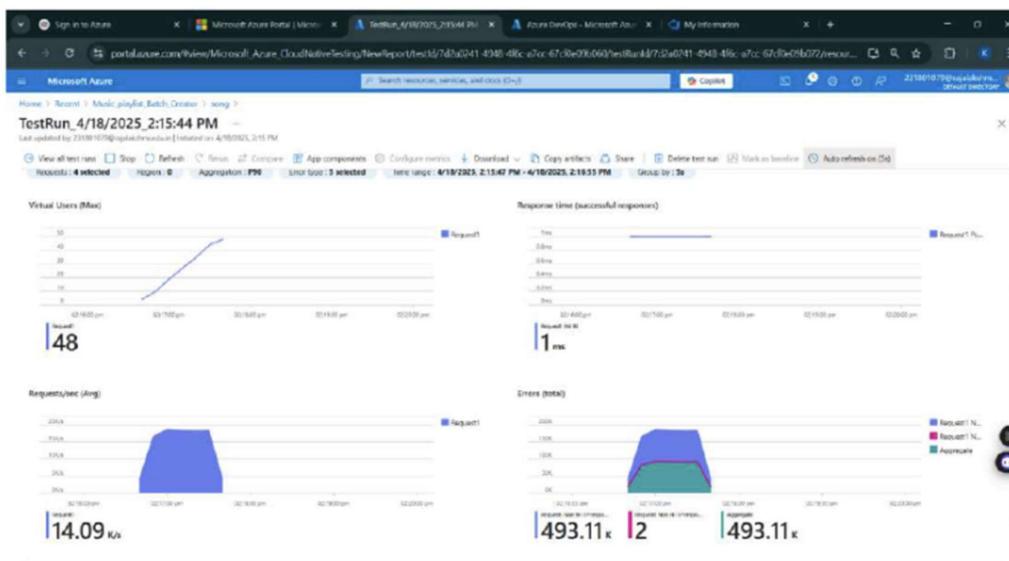
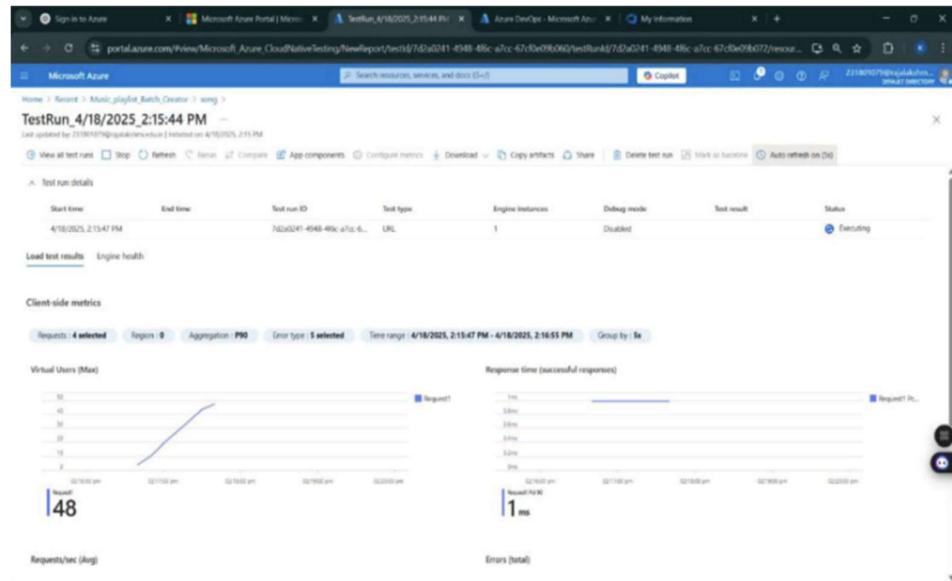
#### 6. Once deployment is complete, click Go to resource.

## **Steps to Create and Run a Load Test:**

Once your resource is ready:

1. Go to your Azure Load Testing resource and click Add HTTP requests > Create.
2. Basics Tab
  - Test Name: Provide a unique name.
  - Description: (Optional) Add test purpose.
  - Run After Creation: Keep checked.
3. Load Settings
  - Test URL: Enter the target endpoint (e.g., <https://yourapi.com/products>).
4. Click Review + Create — Create to start the test.

## Load Testing:



## **RESULT:**

Test plans and test cases for selected user stories were created in Azure DevOps, covering both happy and error paths and an Azure Load Testing resource was also set up, and a load test was successfully run to evaluate the performance of the target endpoint.

---