

# **RAJALAKSHMI ENGINEERING COLLEGE**

**THANDALAM – 602 105**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ACADEMIC YEAR 2024-2025**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CS23432**

**SOFTWARE CONSTRUCTION**

**Lab Manual**

**2024-2025**

**Name : THARUNRAJ**

**Year/Branch/Section : II /CSE/ D**

**Register No. : 230701362**

**Semester : IV**

**Academic Year: 2024-25**

## INDEX

Ex No	List of Experiments
1	Study of Azure DevOps
2	Designing Project using AGILE-SCRUM Methodology.
3	Agile Planning
4	User stories – Creation
5	Architecture Diagram Using AZURE
6	Designing Usecase and Class Diagram
7	Designing Interaction Diagrams
8	Design Interface
9	Implementation – Design a Web Page based on Scrum Methodology
10	Testing using Azure.
11	Deployment

## **EX NO : 1**

## **STUDY OF AZURE DEVOPS**

### **AIM:**

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

### **STUDY:**

#### **1. Understanding Azure DevOps**

Azure DevOps consists of five key services:

##### **1.1 Azure Repos (Version Control)**

- Supports Git repositories and Team Foundation Version Control (TFVC).
- Provides features like branching, pull requests, and code reviews.

##### **1.2 Azure Pipelines (CI/CD)**

- Automates build, test, and deployment processes.
- Supports multi-platform builds (Windows, Linux, macOS).
- Works with Docker, Kubernetes, Terraform, and cloud providers (Azure, AWS, GCP).

##### **1.3 Azure Boards (Agile Project Management)**

- Manages work using Kanban boards, Scrum boards, and dashboards.
- Tracks user stories, tasks, bugs, sprints, and releases.

##### **1.4 Azure Test Plans (Testing)**

- Provides manual, exploratory, and automated testing.
- Supports test case management and tracking.

##### **1.5 Azure Artifacts (Package Management)**

- Stores and manages NuGet, npm, Maven, and Python packages.
- Enables versioning and secure access to dependencies.

### **Getting Started with Azure DevOps**

#### **Step 1: Create an Azure DevOps Account**

- Visit Azure DevOps.
- Sign in with a Microsoft Account.
- Create an Organization and a Project.

## Step 2: Set Up a Repository (Azure Repos)

- Navigate to Repos.
- Choose Git or TFVC for version control.
- Clone the repository and push your code.

## Step 3: Configure a CI/CD Pipeline (Azure Pipelines)

- Go to Pipelines → New Pipeline.
- Select a source code repository (Azure Repos, GitHub, etc.).
- Define the pipeline using YAML or the Classic Editor.
- Run the pipeline to build and deploy the application.

## Step 4: Manage Work with Azure Boards

- Navigate to Boards.
- Create work items, user stories, and tasks.
- Organize sprints and track progress.

## Step 5: Implement Testing (Azure Test Plans)

- Go to Test Plans.
- Create and run test cases.
- View test results and track bugs.

## **RESULT:**

The study was successfully completed.

**EX NO : 2****PROBLEM STATEMENT****AIM :**

To prepare PROBLEM STATEMENT for your given project.

**PROBLEM STATEMENT:**

In many educational institutions, tracking and understanding student academic performance remains a fragmented and manual process. Teachers and administrators often struggle to identify trends in student marks, deliver timely interventions, or securely manage student data. Meanwhile, students rarely get clear, visual insights into their own academic ups and downs across subjects, making it harder for them to take informed steps toward improvement.

To address these challenges, we have developed an intuitive academic progress tracking tool. It allows administrators to securely add and manage student records, visualize academic trends through clear graphs and dashboards, and send targeted alerts to students via email based on performance categories. Students, in turn, gain a personalized view of their academic journey, enabling them to recognize their strengths and areas needing attention.

**RESULT:**

The Problem statement is written successfully.

## **EX NO : 3**

## **AGILE PLANNING**

### **AIM:**

To prepare an Agile Plan.

### **THEORY:**

Agile planning is a part of the Agile methodology, which is a project management style with an incremental, iterative approach. Instead of using an in-depth plan from the start of the project—which is typically product-related—Agile leaves room for requirement changes throughout and relies on constant feedback from end users. With Agile planning, a project is broken down into smaller, more manageable tasks with the ultimate goal of having a defined image of a project's vision. Agile planning involves looking at different aspects of a project's tasks and how they'll be achieved, for example:

- Roadmaps to guide a product's release ad schedule
- Sprints to work on one specific group of tasks at a time
- A feedback plan to allow teams to stay flexible and easily adapt to change

User stories, or the tasks in a project, capture user requirements from the end user's perspective Essentially, with Agile planning, a team would decide on a set of user stories to action at any given time, using them as a guide to implement new features or functionalities in a tool. Looking at tasks as user stories is a helpful way to imagine how a customer may use a feature and helps teams prioritize work and focus on delivering value first.

Steps in Agile planning process:

1. Define vision
2. Set clear expectations on goals
3. Define and break down the product roadmap
4. Create tasks based on user stories
5. Populate product backlog
6. Plan iterations and estimate effort
7. Conduct daily stand-ups
8. Monitor and adapt

### **RESULT:**

Thus the Agile plan was completed successfully.

## **EX NO: 4**

## **CREATE USER STORY**

### **AIM:**

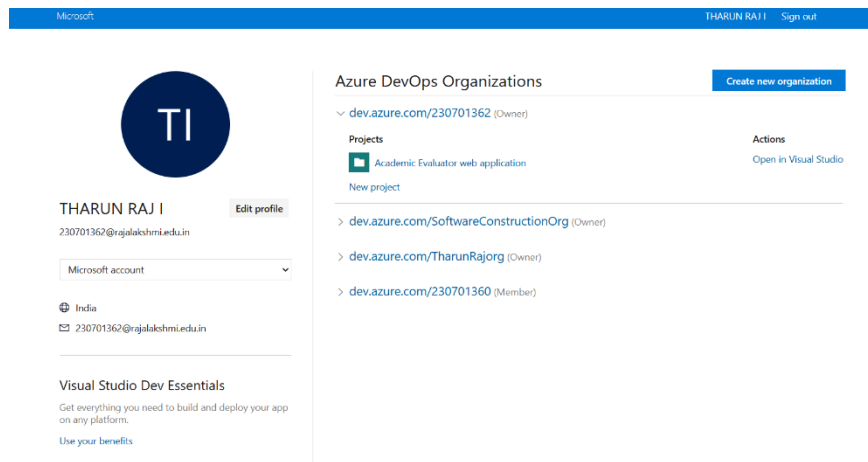
To create User Stories.

### **THEORY:**

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

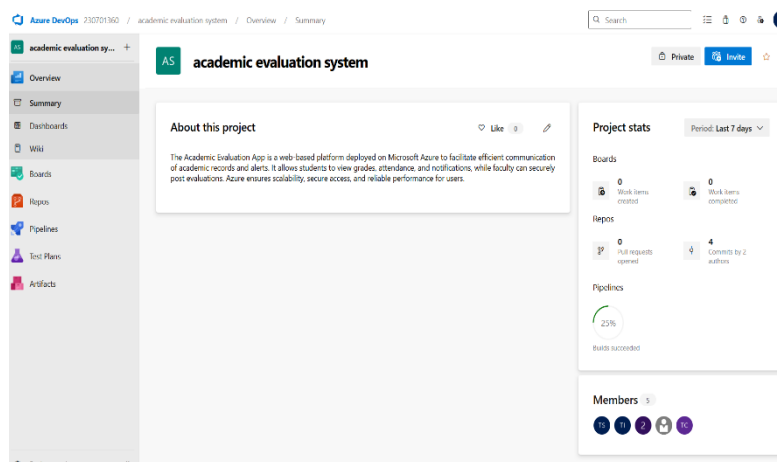
### **PROCEDURE:**

1. Open your web browser and go to the Azure website:  
<https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for  
<https://signup.live.com/?lic=1>
3. Go to Azure Home Page.
4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.
5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.
6. Create the First Project in Your Organization. After the organization is set up, you'll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.
  - (i) On the organization's Home page, click on the New Project button.
  - (ii) Enter the project name, description, and visibility options:
    - Name: Choose a name for the project (e.g., LMS).
    - Description: Optionally, add a description to provide more context about the project.
    - Visibility: Choose whether you want the project to be Private (accessible only to those invited) or Public (accessible to anyone).
  - (iii) Once you've filled out the details, click Create to set up your first project.



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

8. Open project's dashboard.



9. To manage user stories:

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

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

10. Fill in the User Story.



230701360 / academic evaluation system / Boards / Work items

Search

20 of 29

academic evaluation sy...

Overview

Boards

Work Items

Boards

Backlogs

Sprints

Queries

Delivery Plans

Analytics views

Repos

Pipelines

Test Plans

Artifacts

Recently updated

Back to Work Items

2 Dashboard

THARUN RAJ I

0 Comments

Add Tag

Save

Follow

Updated by "tejashee carpenkumar" May

Details

Item

Active

Area

academic evaluation system

Reason

Implementation started

Description

1)As a student, I want to view my academic and extracurricular performance on a dashboard so that I can track my progress easily.  
2)As a faculty member, I want to see a list of students and their performance metrics on a dashboard so that I can quickly analyze their progress.  
3)As an admin, I want to see an overview of student and faculty activity so that I can monitor the platform's performance and ensure smooth operations.  
4)As a parent, I want to access my child's performance dashboard so that I can support their academic and extracurricular growth.

Planning

Story Points

Priority

3

Risk

Classification

Value area

Business

Acceptance Criteria

1)The dashboard should display academic scores and extracurricular ratings in a visually appealing manner (charts, graphs, or tables).  
The dashboard should show the latest evaluation updates from authorized personnel.  
The student should be able to filter data by subject, activity, or semester.  
The dashboard should provide a section for feedback and improvement suggestions.  
2)The dashboard should display a list of students with their academic and extracurricular ratings.  
The faculty should be able to search for and filter students based on performance level, subject, or activity.  
The dashboard should allow faculty to select a student and add/update their evaluation.  
A notification system should alert faculty about students who need attention.  
3)The dashboard should provide an overview of student evaluations, faculty activity, and system analytics.  
The admin should be able to add or remove users (students, faculty).  
The dashboard should display insights on trends in student performance and engagement.  
The system should generate downloadable reports for academic and extracurricular performance.  
4)The parent should be able to log in and view their child's academic scores and activity ratings.  
The dashboard should provide improvement suggestions and faculty feedback.  
Alerts should notify parents if their child is underperforming in any area.  
A communication section should allow parents to reach out to faculty for discussions.

Discussion

Add a comment. Use # to link a work item, @ to mention a person, or / to link a pull request.

Switch to Markdown editor

Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Development

Add link

Build

Build academic evaluation system (2)\_20250520.1

Updated May 20

Succeeded

Related Work

Add link

Add an existing work item as a parent

## Epic: Enhance User Interface

### Purpose:

To improve the overall user experience for both students and administrators by making the interface more intuitive, interactive, and informative. This epic focuses on visual clarity, accessibility, and ease of navigation to ensure users can efficiently engage with key functionalities such as academic tracking and communication.

### Feature 1: Dynamic Subject Graphs

#### Purpose:

To help students and admins visualize academic performance over time in a subject-wise manner.

#### Key Functionalities:

- Each subject has its own interactive graph, reducing clutter and increasing clarity.
- Even small tests or quizzes are plotted to provide a complete academic picture.
- Students can easily identify trends and performance dips or spikes in each subject.

### Feature 2: Mode Toggle Button

#### Purpose:

To improve accessibility and personalization of the user interface.

#### Key Functionalities:

- Allows users to switch between light and dark mode based on preference or comfort.
- Enhances readability in different lighting conditions.
- Creates a more modern and user-friendly environment.

### Feature 3: Send Alerts (Admin Panel)

#### Purpose:

To simplify and streamline the communication process from admins to students.

#### Key Functionalities:

- Alert All: Send announcements or reminders to all students instantly.
- Alert Department: Target messages to a specific department, ensuring relevance.
- Alert Specific: Communicate with selected individual students for personal guidance or alerts.
- The interface is cleanly organized for quick selection and message delivery.

### Feature 4: Separate Graphs per Subject

#### Purpose:

To prevent information overload and allow detailed performance analysis.

#### Key Functionalities:

- Each subject's performance is displayed on a dedicated graph.
- Users can focus on one subject at a time to better understand their academic journey.
- Includes performance data for major exams and minor tests alike.

#### Feature 5: Readable Mark Display

##### Purpose:

To make performance data both visual and text-based for broader accessibility.

##### Key Functionalities:

- Alongside graphs, numerical marks are shown clearly.
- Ensures that students who prefer numbers to visuals can still access their performance data easily.
- Adds transparency and clarity to the graphical insights.

#### USER STORY 1:

***As a student, I want to view my academic and extracurricular performance on a dashboard so that I can track my progress easily.***

##### Acceptance Criteria :

1. The dashboard should display academic scores and extracurricular ratings in a visually appealing manner (charts, graphs, or tables).
2. The dashboard should show the latest evaluation updates from authorized personnel.
3. The student should be able to filter data by subject, activity, or semester.
4. The dashboard should provide a section for feedback and improvement suggestions.

#### USER STORY 2 :

***As a faculty member, I want to see a list of students and their performance metrics on a dashboard so that I can quickly analyze their progress.***

##### Acceptance Criteria :

1. The dashboard should display a list of students with their academic and extracurricular ratings.
2. The faculty should be able to search for and filter students based on performance level, subject, or activity..

3. The dashboard should allow faculty to select a student and add/update their evaluation.
4. A notification system should alert faculty about students who need attention

#### USER STORY 3 :

*As an admin, I want to see an overview of student and faculty activity so that I can monitor the platform's performance and ensure smooth operations.*

#### Acceptance Criteria :

1. The dashboard should provide an overview of student evaluations, faculty activity, and system analytics.
2. The admin should be able to add or remove users (students, faculty).
3. The dashboard should display insights on trends in student performance and engagement.
4. The system should generate downloadable reports for academic and extracurricular performance.

#### RESULT:

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

## EX NO: 5

## SEQUENCE DIAGRAM

### AIM:

To design a Sequence Diagram by using Mermaid.js

### THEORY:

A Sequence Diagram is a key component of Unified Modelling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modelling dynamic behaviour in a system.

### PROCEDURE:

1. Open a project in Azure DevOps Organisations.
2. To design select wiki from menu.
3. Write code for drawing sequence diagram and save the code.

sequenceDiagram

Code:

```
participant Student
participant Faculty
participant Admin
participant Parent

%% Student interactions
Student->>Student: Register and log in
Student->>Student: Access profile and performance
Student->>Student: View academic scores
Student->>Student: View extracurricular performance
Student->>Student: See feedback
Student->>Student: Compare ratings with peers
Student->>Student: Get recommendations

%% Faculty interactions
Faculty->>Faculty: Log in securely
Faculty->>Student: Assign academic scores
Faculty->>Student: Assign extracurricular ratings
Faculty->>Student: Add detailed feedback
Faculty->>Student: Generate performance reports
Faculty->>Student: Categorize students

%% Admin interactions
Admin->>Admin: Manage student accounts
Admin->>Admin: Manage faculty accounts
Admin->>Faculty: Oversee evaluation
Admin->>Admin: Generate analytics reports
```

%% Parent interactions (optional)

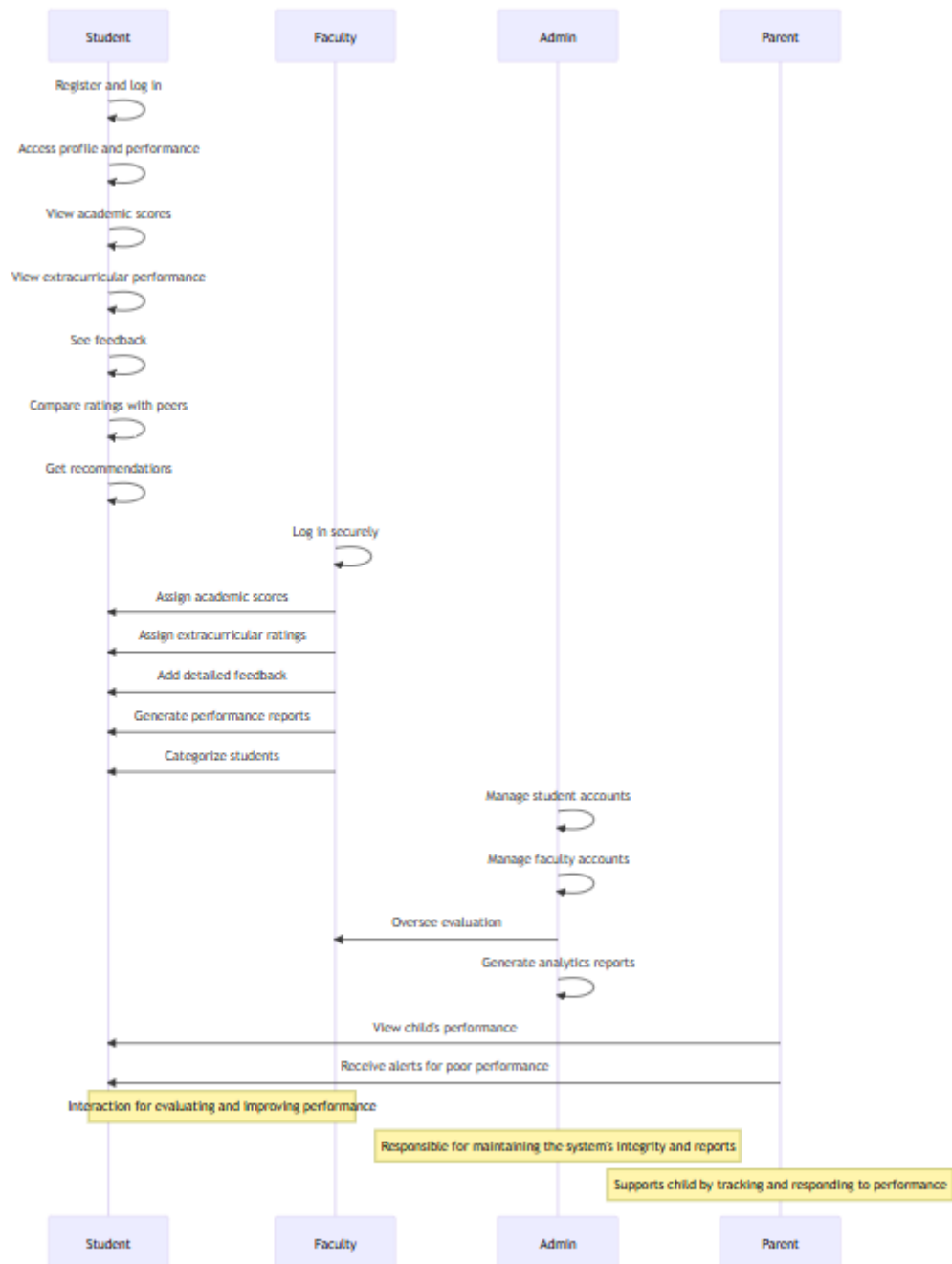
Parent->>Student: View child's performance

Parent->>Student: Receive alerts for poor performance

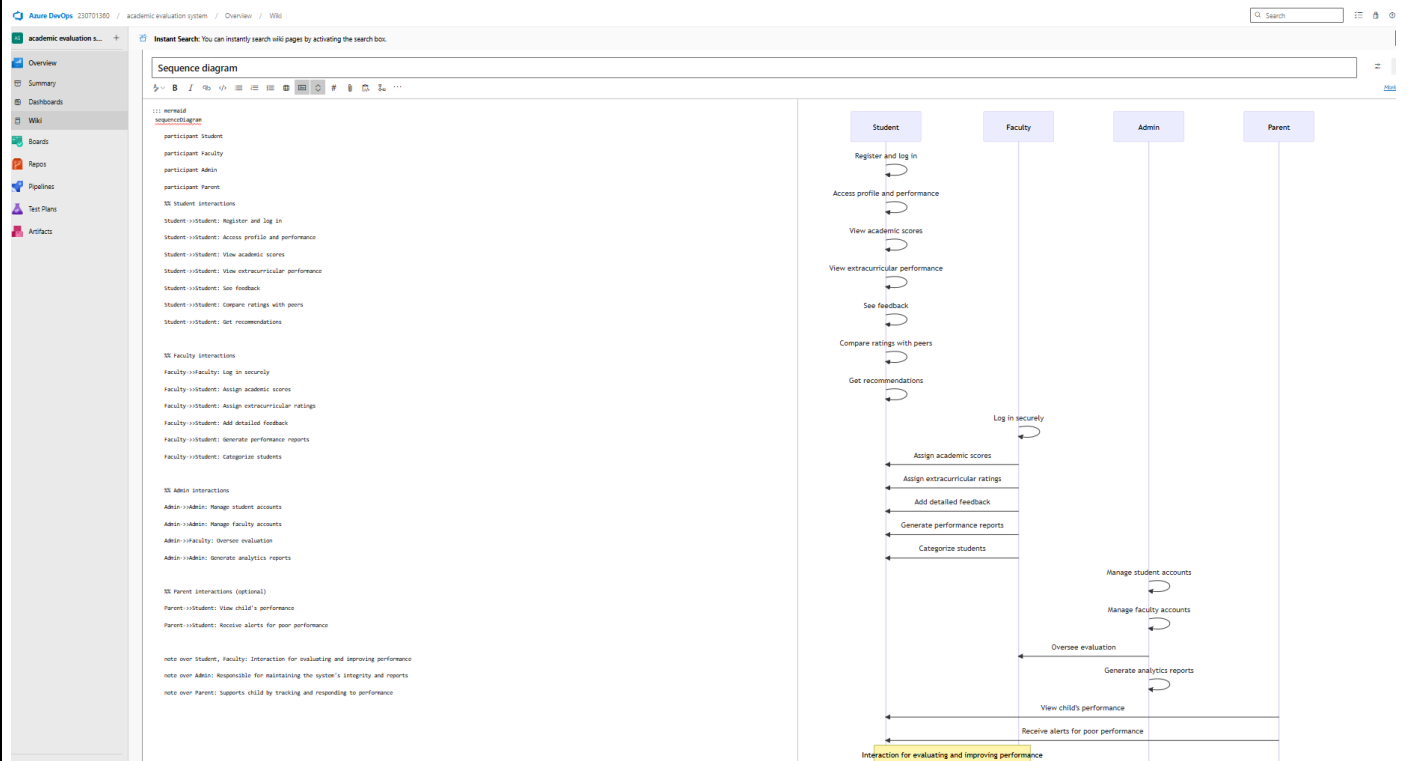
note over Student, Faculty: Interaction for evaluating and improving performance

note over Admin: Responsible for maintaining the system's integrity and reports

note over Parent: Supports child by tracking and responding to performance



4. Click wiki menu and select the page.



## RESULT:

The sequence diagram is drawn successfully.

## EX NO: 6

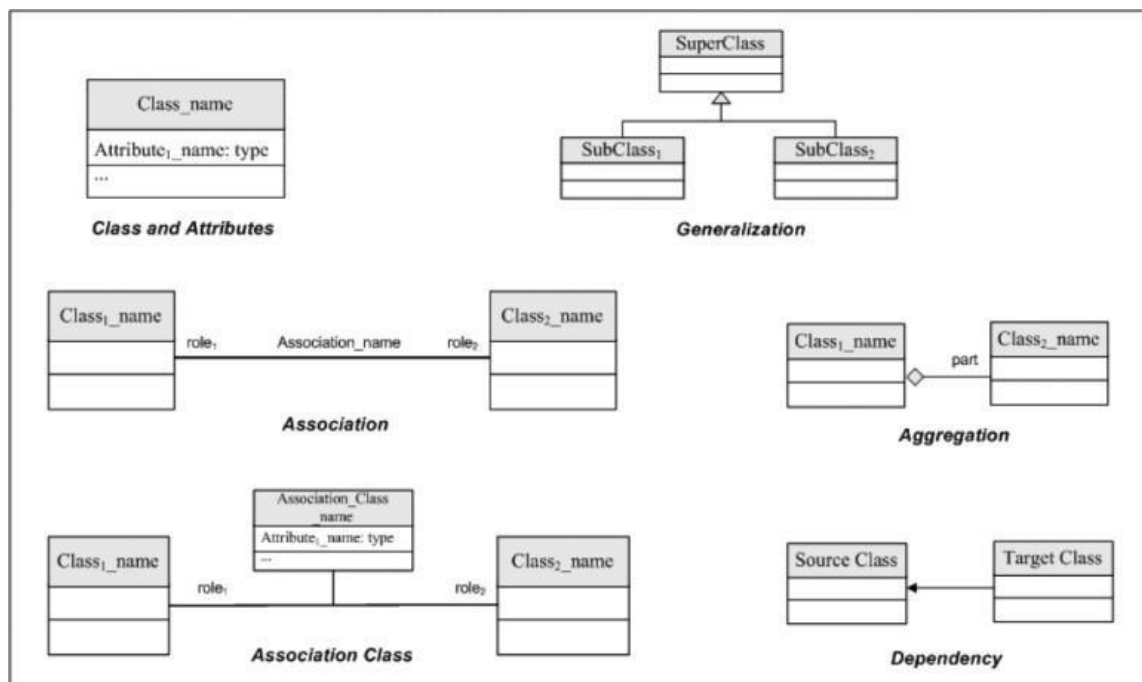
## CLASS DIAGRAM

### AIM:

To draw a simple class diagram.

### THEORY:

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



Notations in class diagram

### PROCEDURE:

1. Open a project in Azure DevOps Organisations.
2. To design select wiki from menu.
3. Write the code for drawing Class Diagram and save the code.

```
classDiagram
cl classDiagram
class User {
+int id
+String name
+String email
}
```



```
class Admin {
    +int adminId
    +addStudent(Object) : boolean
    +removeStudent(Object) : boolean
    +generateReport(int) : Object
}
```

```
class Student {
    +int studentId
    +String study
    +viewPerformance(int) : Object
    +viewRating(int) : int
}
```

```
class Rater {
    +int raterId
    +assignStudents Object
    +rate(Student, Skill, int)
    +viewRatings(Student)
}
```

```
class Skill {
    +int skillId
    +String name
    +String desc
    +getSkills()
}
```

%% Inheritance

Admin --|> User

Student --|> User

Rater --|> User

%% Relationships

Admin --> Student : administers

Rater --> Student : rates >

Student --> Rater : rated by <

Rater --> Skill : rates >

Rater --> Student : assigns >

Rater --> Skill : has >

academic evaluation sy... +

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

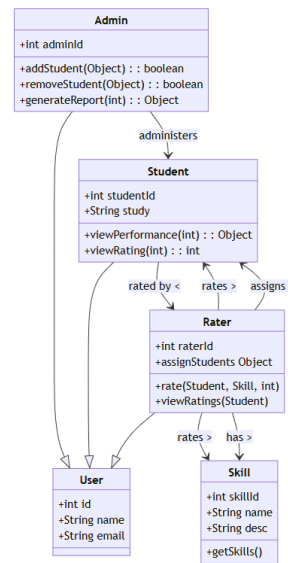
## Class Diagram

```

classDiagram
    class User {
        +int id
        +String name
        +String email
    }
    class Admin {
        +int adminId
        +addStudent(Object) : boolean
        +removeStudent(Object) : boolean
        +generateReport(int) : Object
    }
    class Student {
        +int studentId
        +String study
        +viewPerformance(int) : Object
        +viewRating(int) : int
    }
    class Rater {
        +int raterId
        +assignStudents Object
        +rate(Student, Skill, int)
        +viewRatings(Student)
    }
    class Skill {
        +int skillId
        +String name
        +String desc
        +getSkills()
    }

    %% Inheritance
    Admin --|> User
    Student --|> User
    Rater --|> User

    %% Relationships
    Admin --> Student : administers
    Rater --> Student : rates >
    Student --> Rater : rated by <
    Rater --> Skill : assigns >
    Rater --> Skill : rates >
    Skill --> Rater : has >
    
```



## RESULT:

Thus the class diagram has been designed successfully.

## EX NO: 7

## USE CASE DIAGRAM

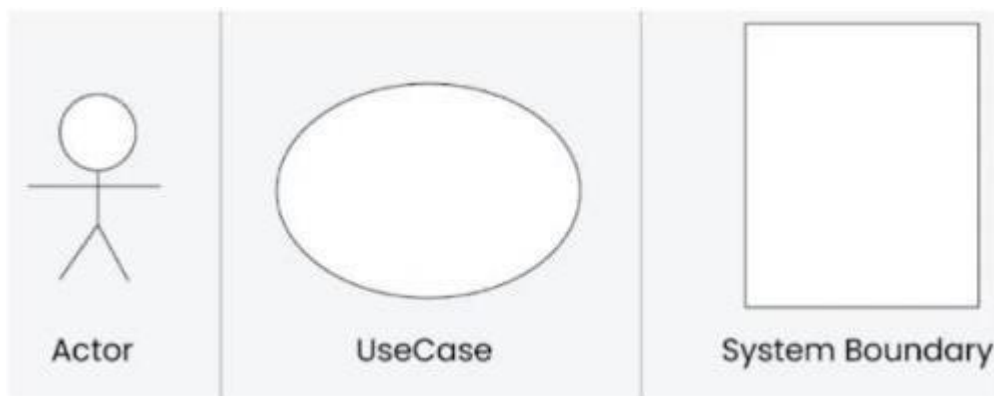
### AIM:

Steps to draw the Use Case Diagram using draw.io

### THEORY:

UCD shows the relationships among actors and use cases within a system which provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model and communicate the scope of a development project

- Use Cases
- Actors
- Relationships
- System Boundary



### PROCEDURE :

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

- Open Draw.io (diagrams.net).
- Click "Create New Diagram" and select "Blank" or "UML Use Case" template.
- Add Actors (Users, Admins, External Systems) from the UML section.
- Add Use Cases (Functionalities) using ellipses.
- Connect Actors to Use Cases with lines (solid for direct interaction, dashed for <<include>> and <<extend>>).
- Save the diagram as .drawio or export as PNG/JPG/SVG.

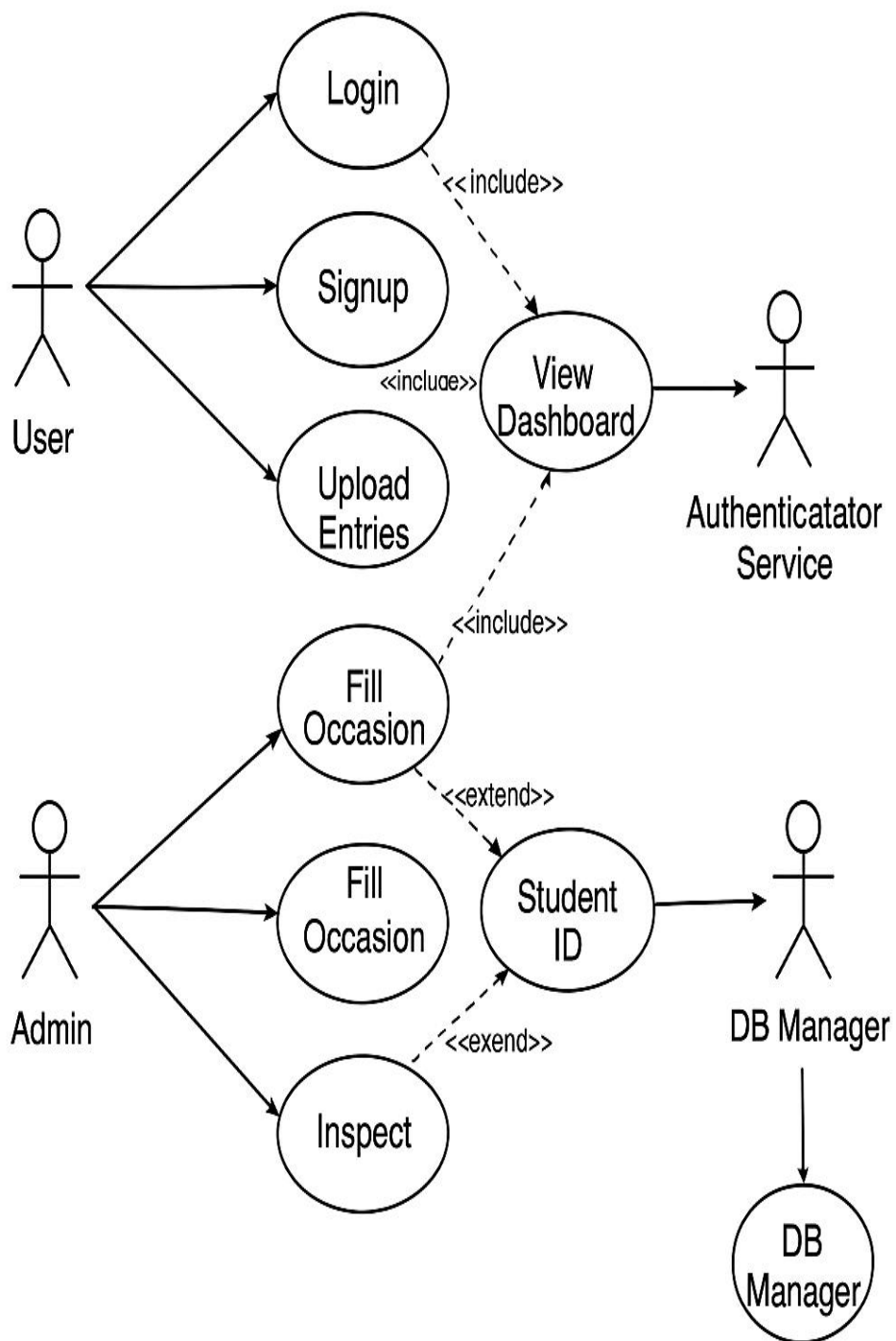
Step 2: Upload the Diagram to Azure DevOps

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



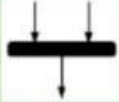








## ACTIVITY DIAGRAM

### AIM :

To draw a sample activity diagram for the Academic Progress Tracker.

### 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

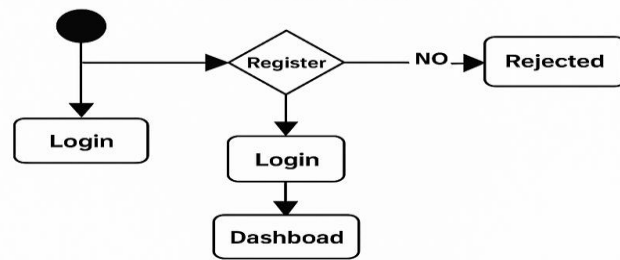
### PROCEDURE:

Step 1. Draw diagram in draw.io.

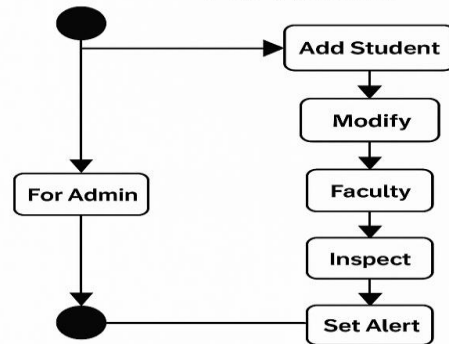
Step 2. Upload the diagram in Azure DevOps wiki.

## Activity Diagram

### For Student



### For Admin



## RESULT:

The activity diagram was designed successfully.

## EX NO: 9

## ARCHITECTURE DIAGRAM

### AIM:

To draw the Architecture Diagram using draw.io.

### THEORY:

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



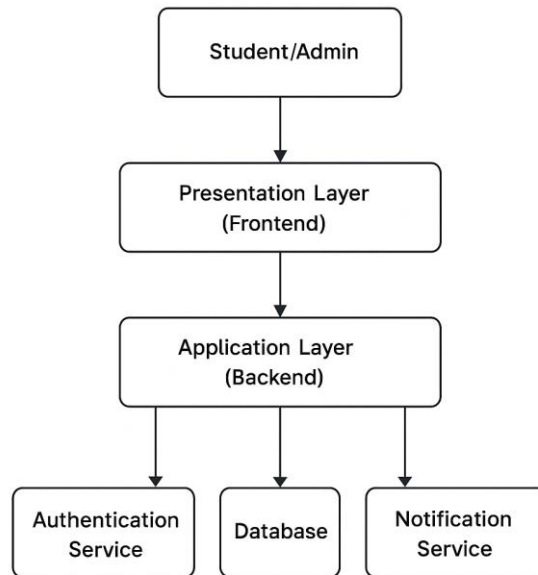
### PROCEDURE:

Step 1. Draw diagram in draw.io

Step 2. Upload the diagram in Azure DevOps wiki.



## Architecture Diagram



### RESULT:

The architecture diagram was designed successfully

**EX NO: 10**

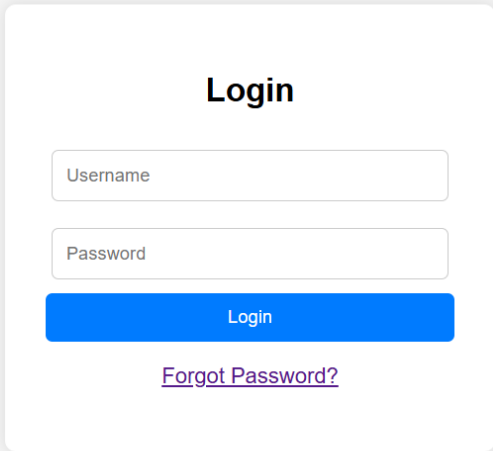
## **USER INTERFACE**

### **AIM:**

Design User Interface for the Academic Progress Tracker

### **UI DESIGNS OF WEATHER APP:**

#### **LOGIN PAGES:**



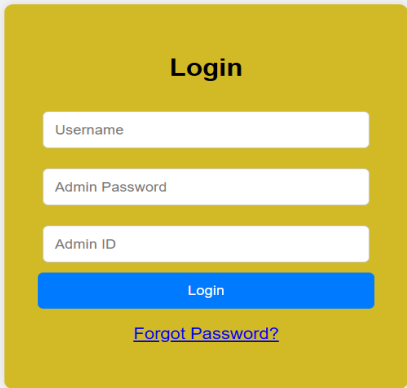
**Login**

Username

Password

Login

[Forgot Password?](#)



**Login**

Username

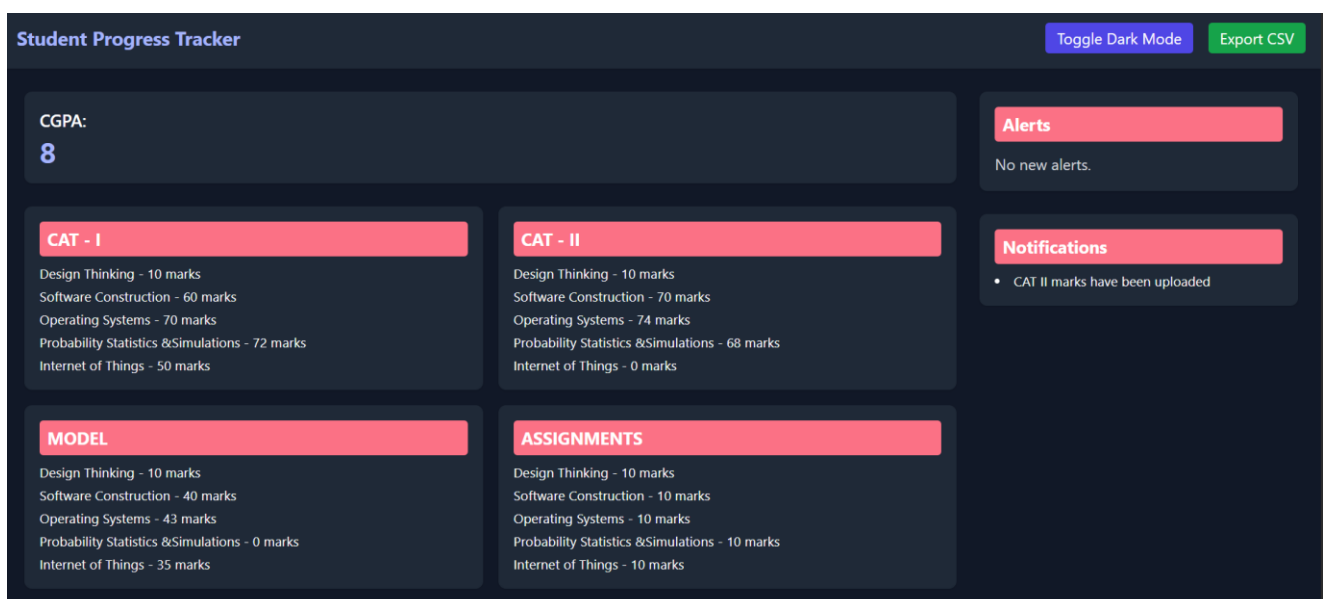
Admin Password

Admin ID

Login

[Forgot Password?](#)

#### **STUDENT DASHBOARD:**



**Student Progress Tracker**

Toggle Dark Mode Export CSV

**CGPA:**  
8

**CAT - I**

Design Thinking - 10 marks  
Software Construction - 60 marks  
Operating Systems - 70 marks  
Probability Statistics & Simulations - 72 marks  
Internet of Things - 50 marks

**CAT - II**

Design Thinking - 10 marks  
Software Construction - 70 marks  
Operating Systems - 74 marks  
Probability Statistics & Simulations - 68 marks  
Internet of Things - 0 marks

**MODEL**

Design Thinking - 10 marks  
Software Construction - 40 marks  
Operating Systems - 43 marks  
Probability Statistics & Simulations - 0 marks  
Internet of Things - 35 marks

**ASSIGNMENTS**

Design Thinking - 10 marks  
Software Construction - 10 marks  
Operating Systems - 10 marks  
Probability Statistics & Simulations - 10 marks  
Internet of Things - 10 marks

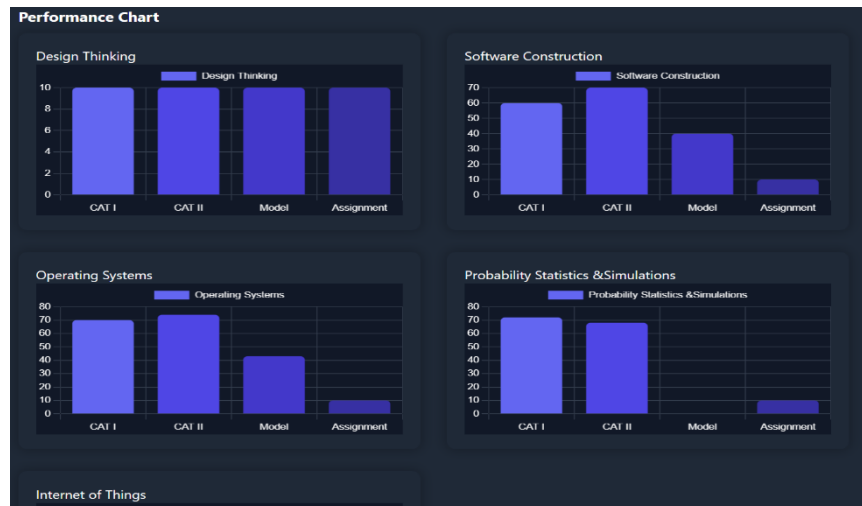
**Alerts**

No new alerts.

**Notifications**

- CAT II marks have been uploaded

## PROGRESS CHARTS:



## ADD STUDENT PAGE:

**Signup with Academic Eval**

Thrilok

230701366@rajalakshmi.edu.in

...

3423423324

230701366

CSE

4

8

8

Signup

## SEND ALERTS PAGE:

The image displays three distinct alert forms arranged horizontally on a light blue background. Each form is a white rounded rectangle with a blue button at the bottom.

- Alert to All Students:** Features a single text input field with the placeholder "Enter your message..." and a blue button labeled "Alert All".
- Alert by Dept & Semester:** Includes two text input fields for "Department (e.g., CSE)" and "Semester (e.g., 2)", followed by a larger text input field for the message, and a blue button labeled "Alert Department".
- Alert to Specific Students:** Includes a text input field for "Roll numbers (comma-separated)", a larger text input field for the message, and a blue button labeled "Alert Students".

## 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)

Azure DevOps

230701360 / academic evaluation system / Overview / Summary

academic evaluation sy... +

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

AS

academic evaluation system

Private

Invite

About this project

Like 0

The Academic Evaluation App is a web-based platform deployed on Microsoft Azure to facilitate efficient communication of academic records and alerts. It allows students to view grades, attendance, and notifications, while faculty can securely post evaluations. Azure ensures scalability, secure access, and reliable performance for users.

Project stats

Period: Last 7 days ▾

Boards

0 Work Items created

0 Work Items completed

Repos

0 Pull requests opened

4 Commits by 2 authors

Pipelines

25%

Builds succeeded

Members 5

1

1

2

### RESULT :

## EX NO: 12

## TESTING USING AZURE

### AIM:

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

### PROCEDURE:

#### Step 1: Open Azure DevOps Project

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

#### Step 2: Navigate to Test Plans

- In the left menu, click on Test Plans.
- Click on “New Test Plan” and give it a name (e.g. Admin Panel).

#### Step 3: Create Test Suites

- Under the test plan, click “+ New Suite” to add test suites for each Epic or Feature.
  - Suite 1: *Admin Panel*
  - Suite 2: *Dashboard*
  - Suite 3: *Student Alerts*

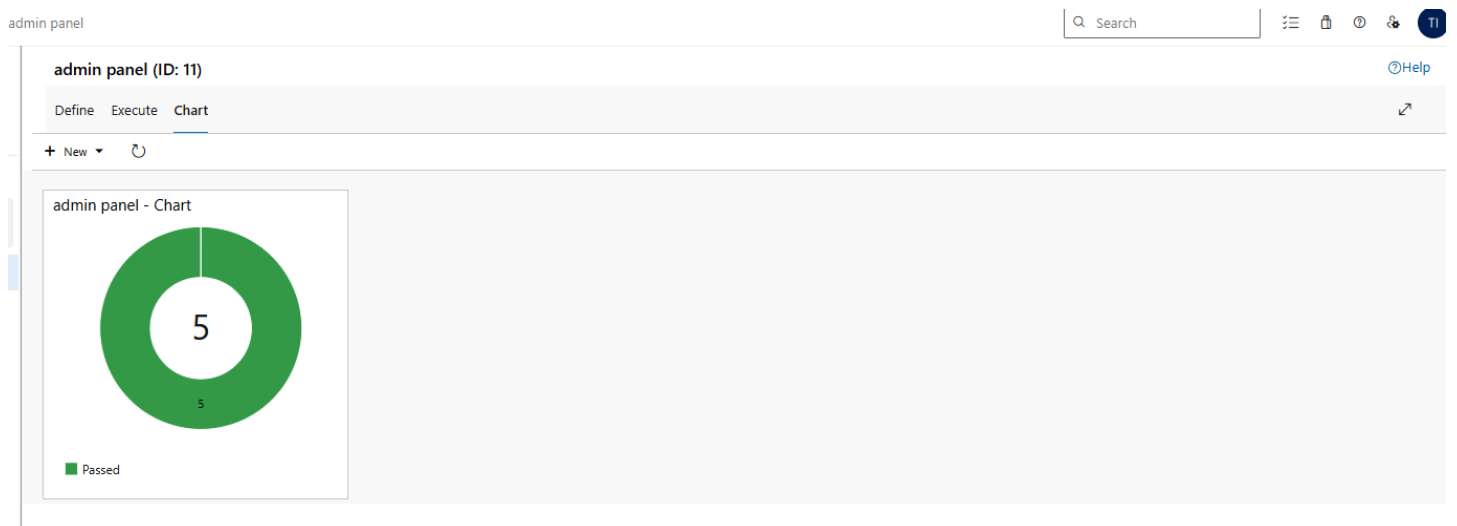
#### Step 4: Add Test Cases

- Click on a suite → + New Test Case.
- Enter the following details for each test case:
  - Test Case ID: (e.g., TC001)
  - Title: (e.g., *Unauthorized access prevented*)
  - Scenario: Describe the situation being tested.
  - Steps:
    - Launch the app
    - Trigger admin signUp event
    - Observe the authentication
  - Expected Result: Unauthorized personnel should never get

admin rights

- Step 5: Execute Test Cases

- Click on each test case and select Run for web application.





Azure DevOps230701360 / academic evaluation system / Test Plans

academic evaluation sy... +

Overview

Boards

Repos

Pipelines

Test Plans

Test plans

Progress report

Parameters

Configurations

Runs

Artifacts

Test Plans

MineAll

Filter by title

StateArea PathIterationAssigned To

Title	Test Plan ID	State	Area Path	Iteration	Assigned To	
academic alerts	14	Active	academic evaluation system	academic evaluation system	Thnikok Chander	
dashboard	12	Active	academic evaluation system	academic evaluation system	THARUN RAJ I	
admin panel	10	Active	academic evaluation system	academic evaluation system	230701363	
login	8	Active	academic evaluation system	academic evaluation system	Tejushree sanjeevikumar	

New Test Plan

## RESULT:

Thus the application was successfully tested in Azure.

## EX NO: 13

## CI/CD PIPELINE

### AIM:

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

### PROCEDURE:

#### Step 1: Create a Build Pipeline (CI)

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

trigger:

- main

pool:

name: Default

steps:

- task: UseNode@2

inputs:

version: '18.x'

- script: npm install

displayName: 'Install Dependencies'

- script: npm run build

displayName: 'Build Application'

- script: npm run test

displayName: 'Run Tests'

- Save and Run the pipeline to verify.

## Step 4: Set Up Release Pipeline (CD)

- Navigate to Pipelines → Releases → New pipeline.
- Add an Artifact (your build pipeline output).
- Add Stages like:
  - Development
  - Production
- Configure Deploy tasks in each stage:
  - For web apps: Use Azure Web App Deploy task.
  - For mobile: Use relevant deployment tools.

## Step 5: Add Approvals and Gates (Optional)

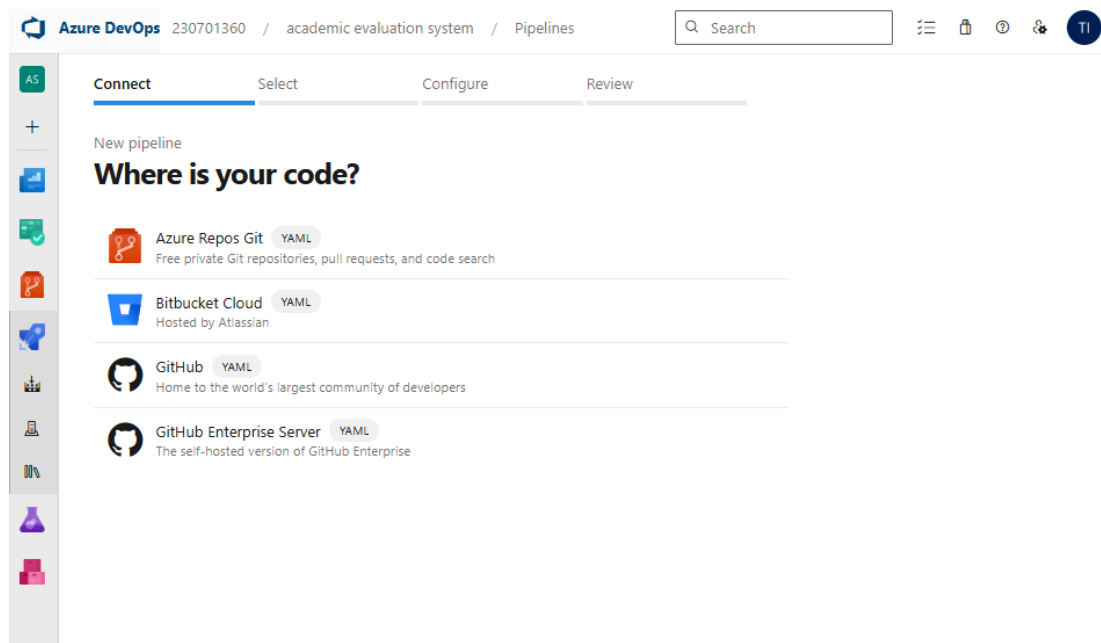
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

## Step 6: Automate Triggering

- Ensure the pipeline triggers:
  - On code push to main branch (CI)
  - After successful build for deployment (CD)

## Step 7: Monitor Pipeline

- Track pipeline status under Pipelines → Runs.
- Debug failures and download logs if necessary.
- Use Azure Boards to link builds with user stories and bugs.



Azure DevOps230701360 / academic evaluation system / Pipelines

Search

☰

🔒

🔗

🔧

👤


11

AS


+

## Individual CI by THARUN RAJ I

[View 5 changes](#)

Repository and version  academic evaluation system

 main  86756b7e

Time started and elapsed  Just now

 <1s

Related  0 work items

 0 artifacts

Tests and coverage  [Get started](#)

install node.js

### Jobs

Name	Status	Duration
 Job	Success	 4m 0s

## RESULT:

Thus the CI/CD pipeline has been successfully implemented.