



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

**DEPARTMENT OF COMPUTER SCIENCE &
ENGINEERING ACADEMIC YEAR 2024-2025
EVEN SEMESTER**



**CS23432 SOFTWARE ENGINEERING LAB
LAB MANUAL
SECOND YESR
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 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

Requirements	
Hardware	Intel i3, CPU @ 1.20GHz 1.19 GHz, 4 GB RAM, 32 Bit Operating System
Software	StarUML , Azure

LAB PLAN

CS19442-SOFTWARE ENGINEERING LAB

Ex No	Date	Topic	Page No	Sign
1		Study of Azure DevOps		
2		Writing Problem Statement		
3		Designing Project using AGILE-SCRUM Methodology by using Azure.		
4		Agile Planning		
5		User stories – Creation		
6		Architecture Diagram Using AZURE		
7		Designing 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

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

CO - PO – PSO matrices of course

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

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

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

EX NO: 1

STUDY OF AZURE DEVOPS

AIM:

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

STUDY:

Azure DevOps is a cloud-based platform by Microsoft that provides tools for DevOps practices, including CI/CD pipelines, version control, agile planning, testing, and monitoring. It supports teams in automating software development and deployment.

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:

Online Quiz System

The project aims to develop a comprehensive Online Quiz System that facilitates interactive quiz administration and assessment. The system will support user registration and role-based access (students and instructors), multiple question types (MCQs, True/False, Fill-in-the-Blanks), and image-based questions with CRUD functionality. Students can attempt quizzes with real-time scoring and performance feedback, while instructors can create quizzes, manage question banks, and analyze results through detailed reports and dashboards. The solution will be a secure, scalable web application featuring automated grading, progress tracking, and analytics to evaluate individual and group performance, enhancing the learning and assessment experience.

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:

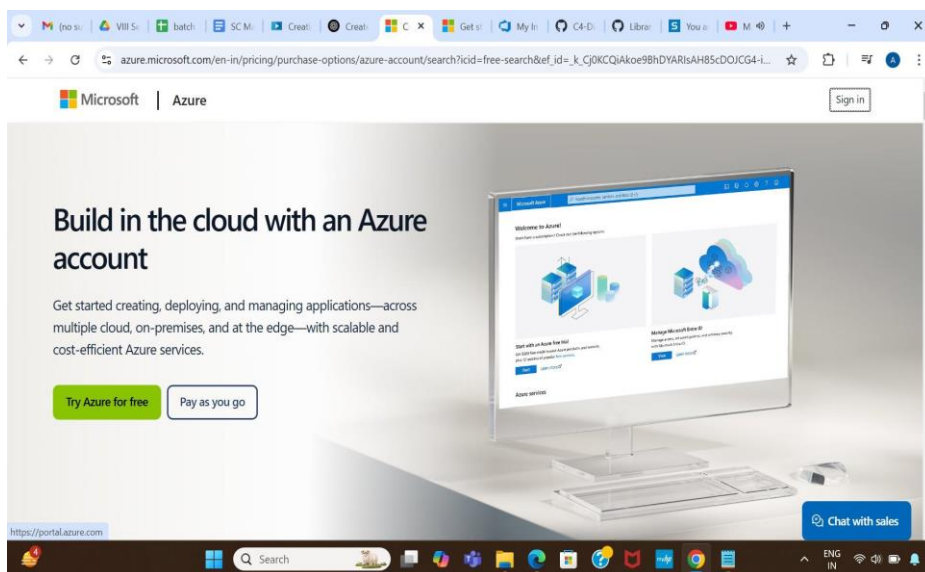
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

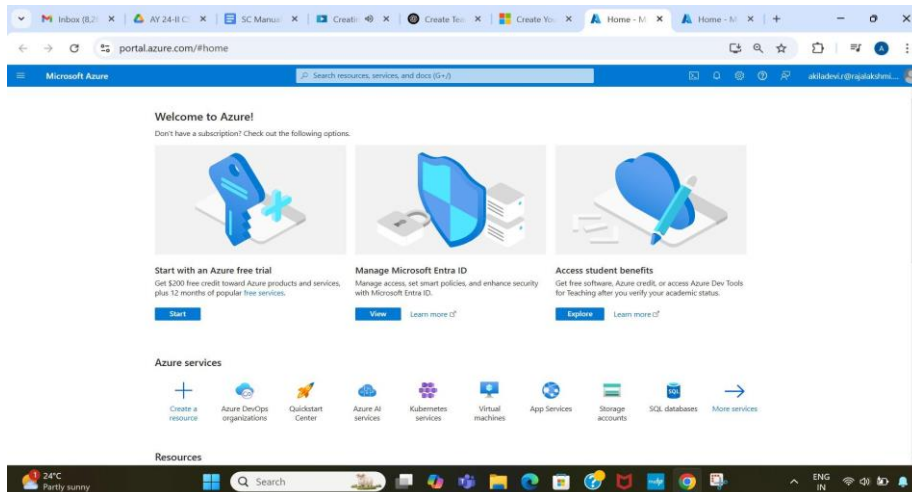
"As a [role], I [want to], [so that]."

PROCEDURE:

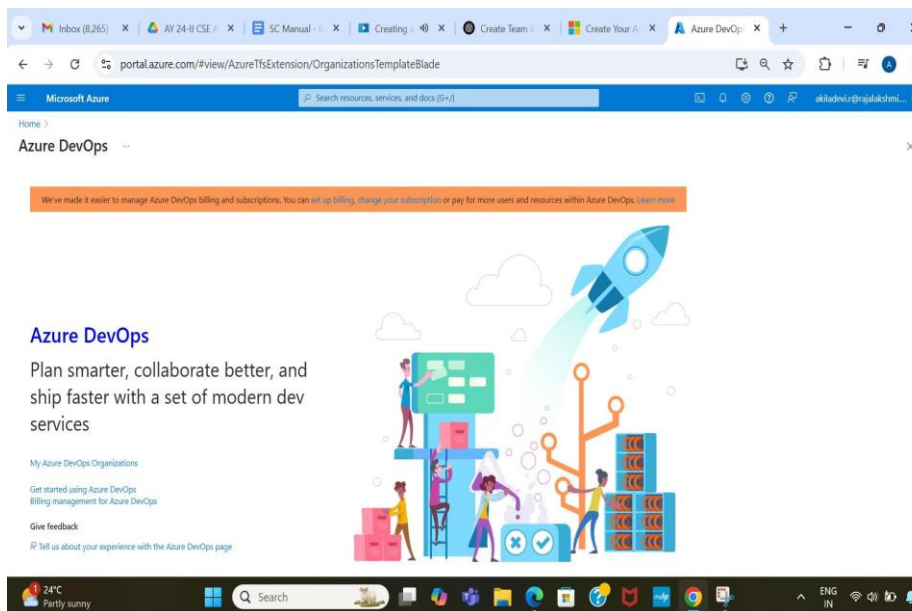
1. Open your web browser and go to the Azure website:
<https://azure.microsoft.com/en-in>
Sign in using your Microsoft account credentials. If you don't have an account, create one.
2. If you don't have a Microsoft account, sign up here:
<https://signup.live.com/?lic=1>



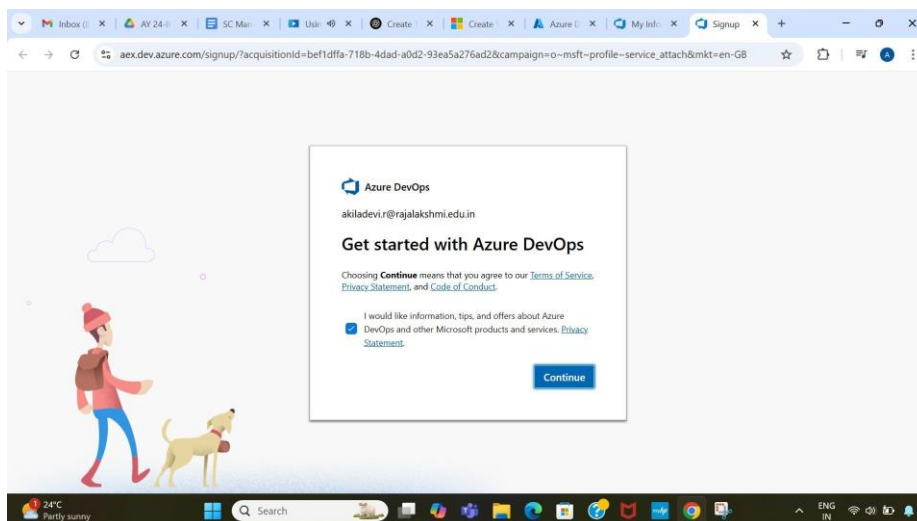
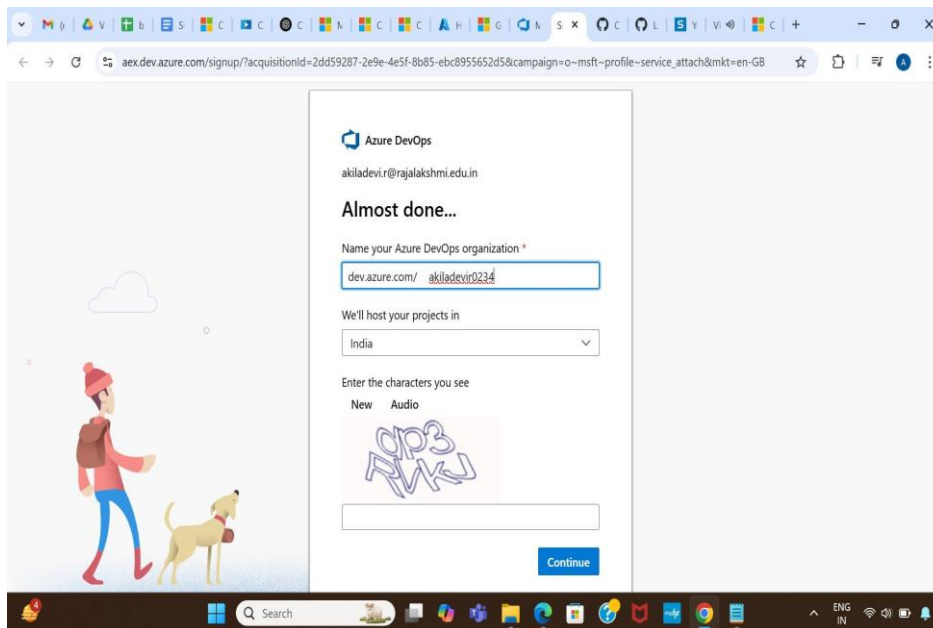
3. You will now reach the Azure home page.



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



5. Click on the "My Azure DevOps Organization" link and create an organization. You will be taken to the Azure DevOps Organization Home page.



6. Create the First Project in Your Organization
 - i. On the organization's Home page, click New Project.
 - ii. Enter the following details:
 - Name: (e.g., LMS)
 - Description: Add project context (optional)
 - Visibility: Choose Private or Public
 - iii. Click Create.

Create new project

Project name *

Description

Visibility

Public

Anyone on the internet can view the project. Certain features like TFVC are not supported.

Private

Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with [organization policies](#).

Advanced

Version control

Git

Work item process

Agile

Cancel

Create

7. Once logged in, ensure you're in the correct organization. If you're part of multiple organizations, switch between them from the top left corner near your profile.

SS

Shruthi S

230701312@rajalakshmi.edu.in

India

230701312@rajalakshmi.edu.in

Visual Studio Dev Essentials

Get everything you need to build and deploy your app on any platform.

Use your benefits

Azure DevOps Organizations

Create new organization

dev.azure.com/230701312 (Owner)

Projects

Online Quiz System

New project

Actions

Open in Visual Studio

dev.azure.com/230701314 (Member)

8. The Project Dashboard will be displayed.

Azure DevOps 230701314 / Trivius / Overview / Summary

Search

Trivius

Private

Trivius

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

Project settings

About this project

The Online Quiz System is a dynamic web-based application designed to simplify and enhance the assessment process for both students and instructors. The platform enables students to register, browse and take quizzes on a wide range of topics with support for multiple question types including Multiple Choice Questions (MCQ), True/False, and Fill in the Blanks. Instructors can create, manage, and schedule quizzes with precise control over timing, question formats, and content, including image uploads. Real-time score tracking, automatic result generation, and performance analytics make it easy for students to assess their knowledge and for instructors to monitor individual and overall class progress. The system includes secure login for both students and instructors, intuitive dashboards, classroom management tools, and detailed reporting features, making it an ideal solution for educational institutions and online learning environments.

Project stats

Period: Last 7 days

Boards

0

9. To manage user stories:
 - a. From the left menu, click on Boards.

- b. Click Add a Work Item → User Story.
- c. Enter the User Story details in the form provided.

10. Fill in User Story Details

Azure DevOps 230701314 / Trivius / Boards / Work items

Did you notice Azure Boards has a new look and awesome new features? [Learn more.](#)

Recently updated Back to Work Items 11 of 54

USER STORY 76

76 As an instructor, I want to create MCQs, True/False, and Fill-in-the-blank questions so that I can design varied quizzes for t

Shruthi S 0 Comments Add Tag Save Follow

Updated by 230701314: Apr 25

State: New Area: Trivius Reason: New Iteration: Trivius

Description

The system should provide an intuitive interface for instructors to create different types of questions including Multiple Choice Questions (MCQs), True/False, and Fill-in-the-blank. Instructors can input the question text, possible answers, correct answers, and optionally attach images or media. This feature ensures flexibility in designing quizzes suited for various learning objectives and assessment styles.

Planning

Story Points: 8
Priority: 2
Risk

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](#)

Acceptance Criteria

RESULT:

The user story was written successfully.

EX NO: 5

SEQUENCE DIAGRAM

AIM:

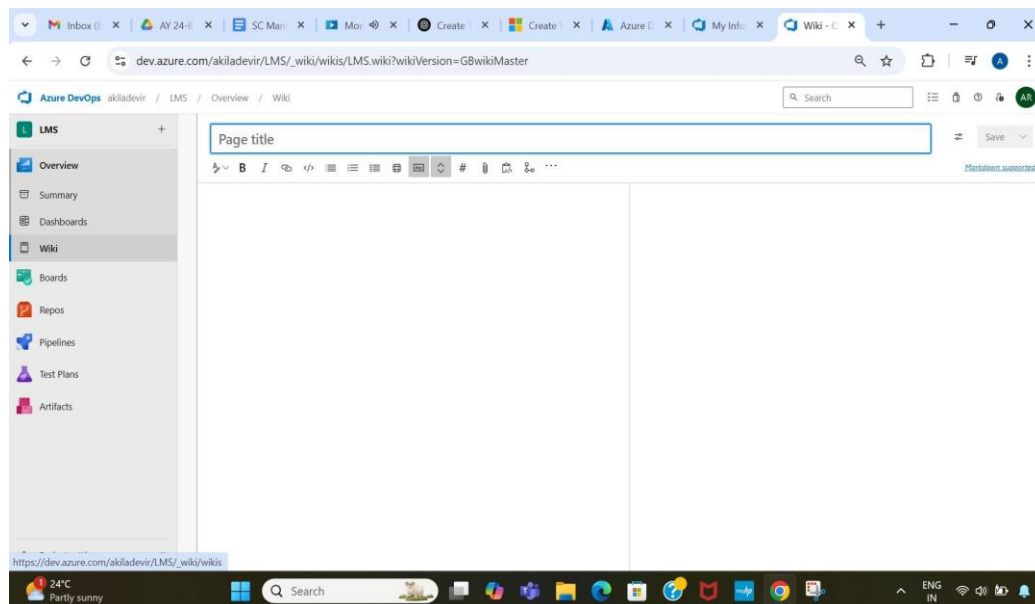
To design a Sequence Diagram by using Mermaid.js

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.

sequenceDiagram

participant Student

participant WebApp

participant AuthService

participant QuizService

participant QuestionBank

participant ResultService

participant Instructor

%% Student registration and login

Student->>WebApp: Register/Login

WebApp->>AuthService: Validate credentials

AuthService-->>WebApp: Success/Failure

WebApp-->>Student: Authenticated

%% Student starts quiz

Student->>WebApp: View Available Quizzes

WebApp->>QuizService: Fetch quizzes

QuizService-->>WebApp: Quiz list

WebApp-->>Student: Display quizzes

%% Student takes quiz

Student->>WebApp: Start Quiz

WebApp->>QuestionBank: Fetch questions

QuestionBank-->>WebApp: Questions

WebApp-->>Student: Display questions

Student->>WebApp: Submit Answers

WebApp->>ResultService: Evaluate and store result

ResultService-->>WebApp: Score

WebApp-->>Student: Show Score

%% Instructor manages quiz

Instructor->>WebApp: Login

WebApp->>AuthService: Validate Instructor

AuthService-->>WebApp: Authenticated

WebApp-->>Instructor: Dashboard

Instructor->>WebApp: Create/Edit Quiz

WebApp->>QuizService: CRUD Quiz

QuizService-->>WebApp: Success

Instructor->>WebApp: Upload Question Images

WebApp->>QuestionBank: Store Images

QuestionBank-->>WebApp: Upload Success

Instructor->>WebApp: View Student Reports

WebApp->>ResultService: Fetch Reports

ResultService-->>WebApp: Report Data

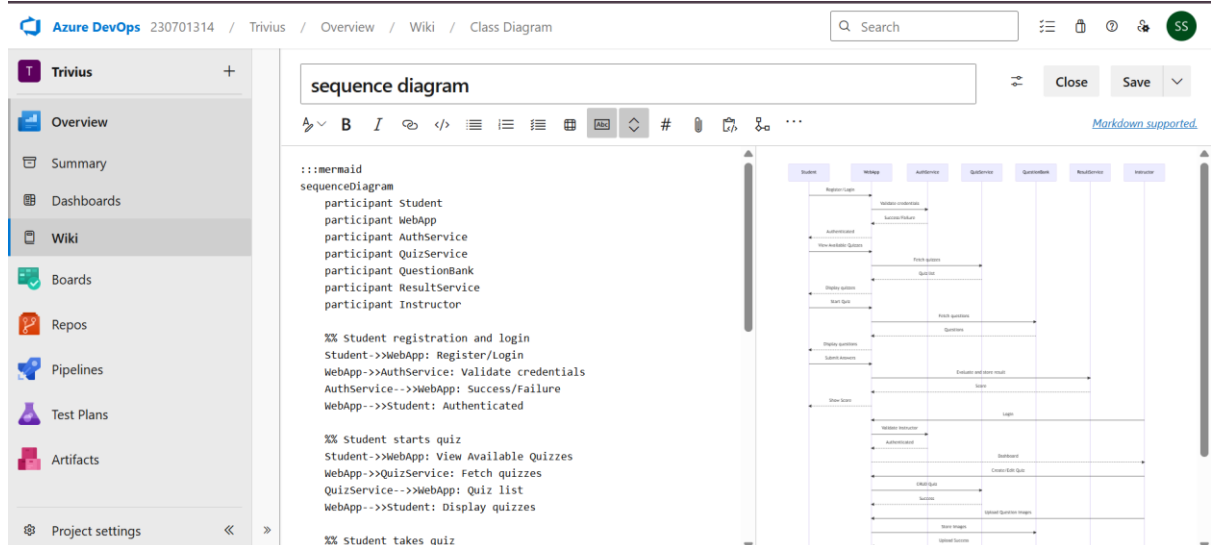
WebApp-->>Instructor: Display Report

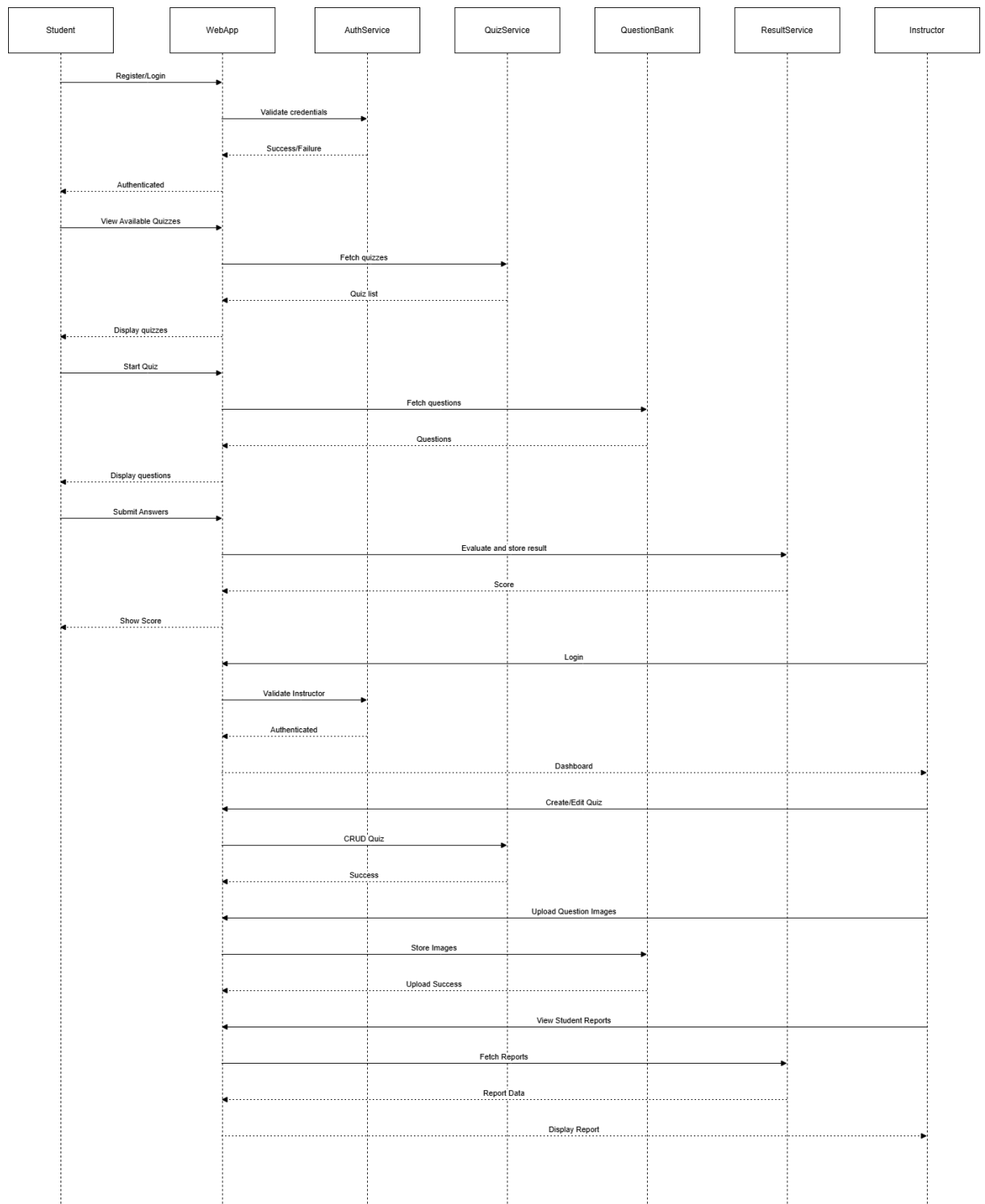
EXPLANATION:

- 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

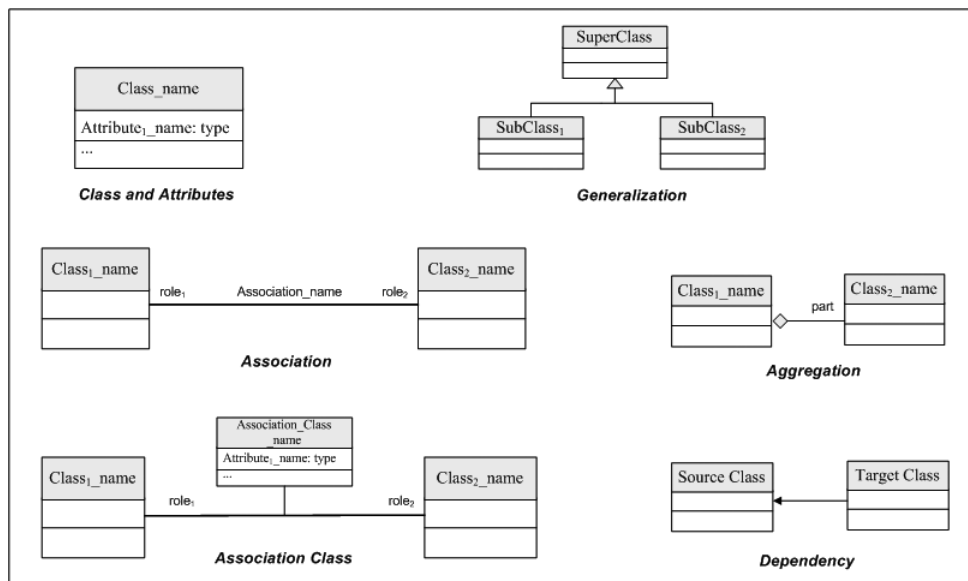
CLASS DIAGRAM

AIM :-

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

THEORY

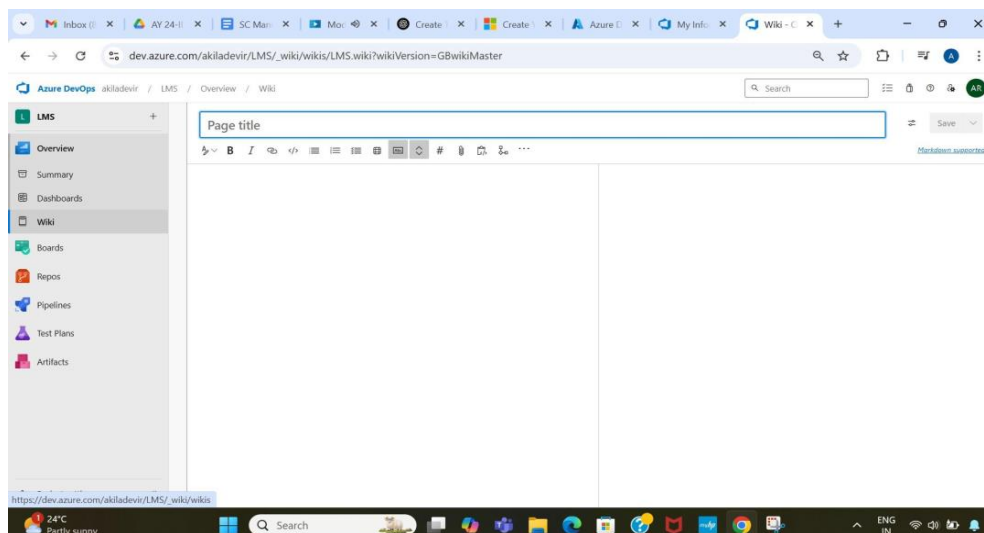
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



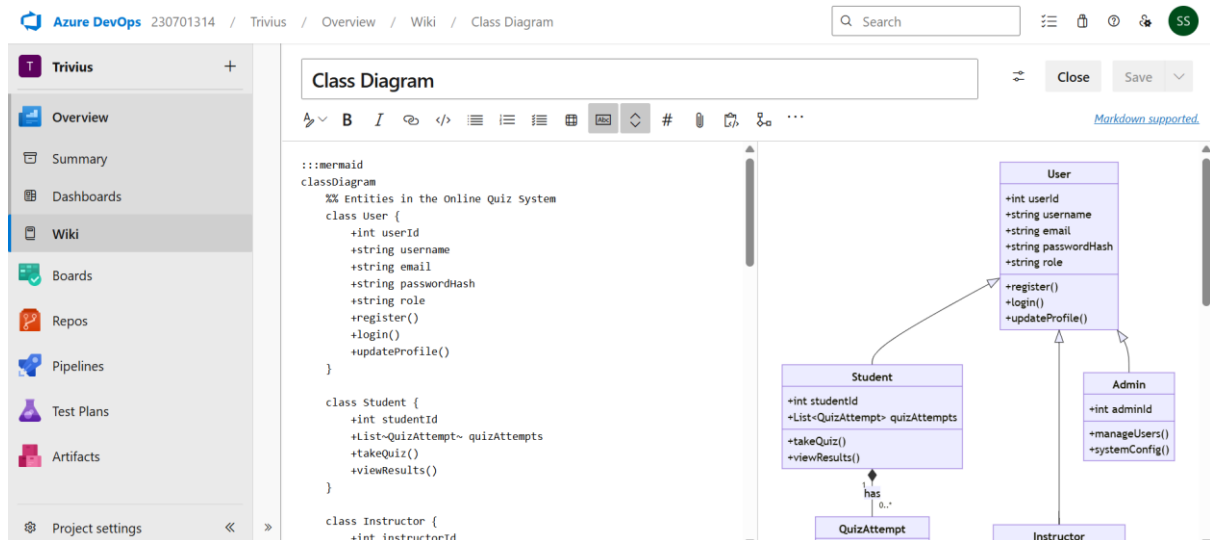
NOTIONS IN CLASS DIAGRAM

Procedure:

1. Open a project in Azure DevOps Organisations.
2. To design select wiki from menu.



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



classDiagram

%% Entities in the Online Quiz System

class User {

+int userId

+string username

+string email

+string passwordHash

+string role

+register()

+login()

+updateProfile()

}

class Student {

+int studentId

+List<QuizAttempt> quizAttempts

+takeQuiz()

+viewResults()

}

```
class Instructor {  
    +int instructorId  
    +createQuiz()  
    +manageQuestionBank()  
    +generateReports()  
}
```

```
class Admin {  
    +int adminId  
    +manageUsers()  
    +systemConfig()  
}
```

```
class Quiz {  
    +int quizId  
    +string title  
    +string description  
    +datetime startTime  
    +datetime endTime  
    +int timeLimit  
    +List~Question~ questions  
    +addQuestion()  
}
```

```
class Question {  
    +int questionId  
    +string text  
    +string type  
    +int points  
    +string mediaUrl
```

```
+List~Option~ options
+uploadImage()
}
```

```
class QuizAttempt {
  +int attemptId
  +datetime startTime
  +datetime endTime
  +int score
  +submitAnswers()
}
```

```
class Report {
  +int reportId
  +generateStudentReport()
  +generateQuestionStats()
  +exportPDF()
}
```

%% Relationships

User <|-- Student

User <|-- Instructor

User <|-- Admin

Instructor "1" *-- "0..*" Quiz : creates

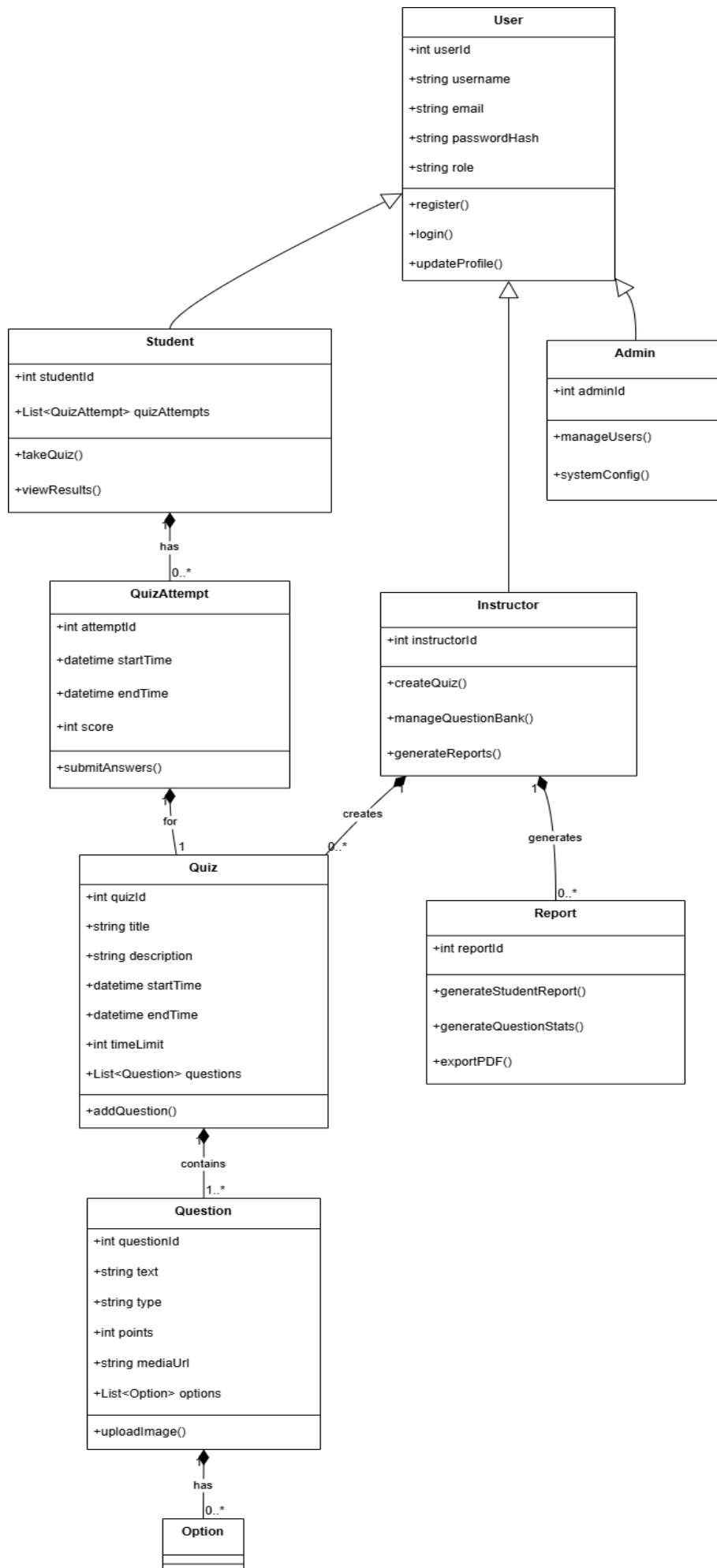
Quiz "1" *-- "1..*" Question : contains

Question "1" *-- "0..*" Option : has

Student "1" *-- "0..*" QuizAttempt : has

QuizAttempt "1" *-- "1" Quiz : for

Instructor "1" *-- "0..*" Report : generates



RESULT:

The class diagram was designed successfully.

EX NO: 7

USECASE DIAGRAM

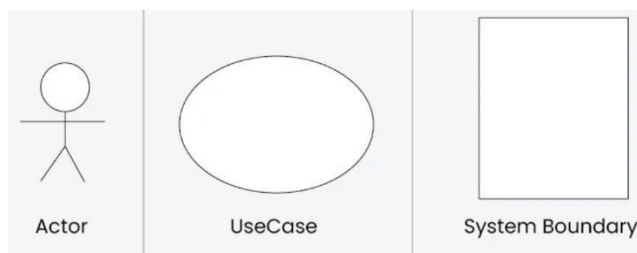
AIM:

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

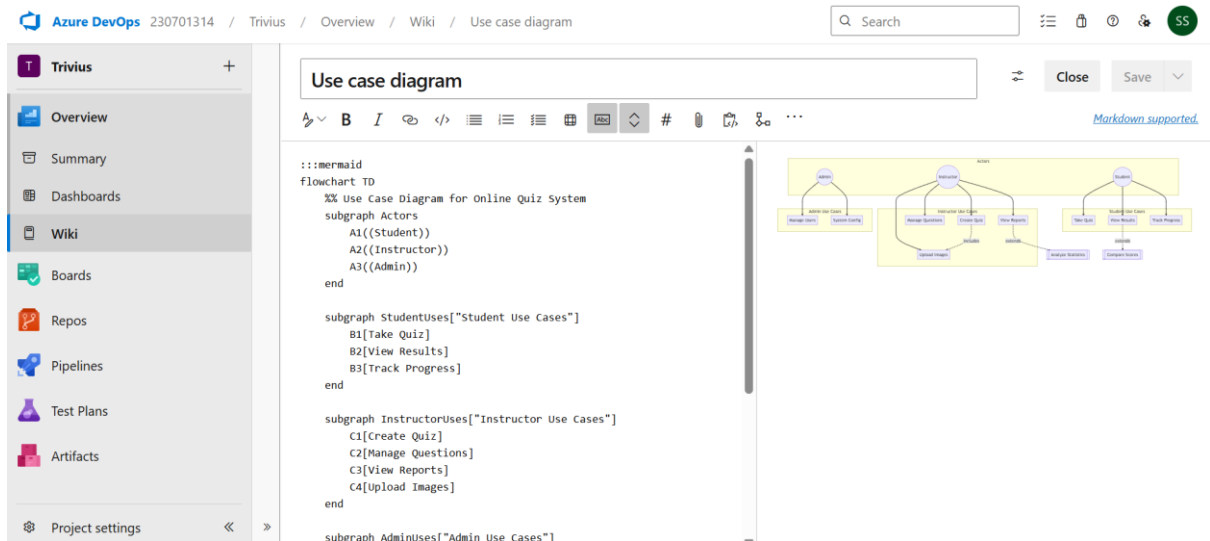
Option 1: Add to Azure DevOps Wiki

- 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.



```

:::mermaid

```

```

flowchart TD

```

```

%% Use Case Diagram for Online Quiz System

```

```

subgraph Actors

```

```

    A1((Student))

```

```

    A2((Instructor))

```

```

    A3((Admin))

```

```

end

```

```

subgraph StudentUses["Student Use Cases"]

```

```

    B1[Take Quiz]

```

```

    B2[View Results]

```

```

    B3[Track Progress]

```

```

end

```

```

subgraph InstructorUses["Instructor Use Cases"]

```

```

    C1[Create Quiz]

```

```

    C2[Manage Questions]

```

```

    C3[View Reports]

```

```

    C4[Upload Images]

```

```

end

```

```

subgraph AdminUses["Admin Use Cases"]

```

```

    D1[Manage Users]

```

```
D2[System Config]
end
```

```
%% Relationships
```

```
A1 --> B1 & B2 & B3
```

```
A2 --> C1 & C2 & C3 & C4
```

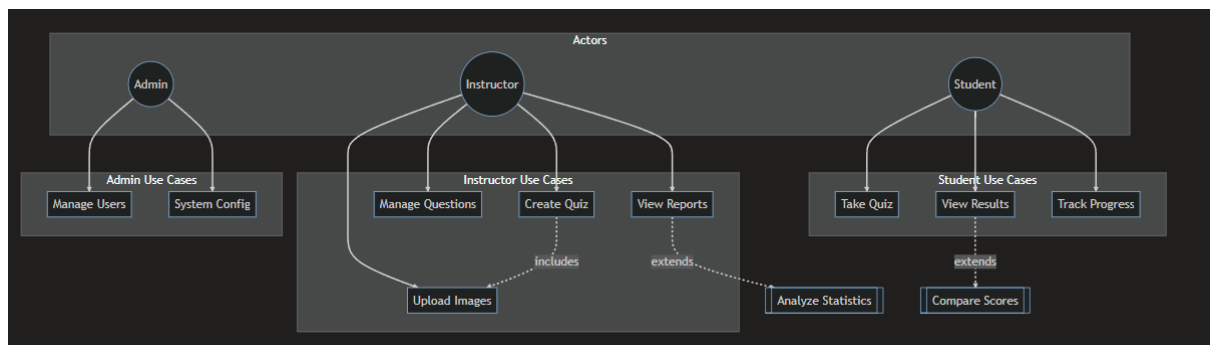
```
A3 --> D1 & D2
```

```
%% Extensions
```

```
C1 -.->|includes| C4
```

```
C3 -.->|extends| E1[[Analyze Statistics]]
```

```
B2 -.->|extends| E2[[Compare Scores]]
```



RESULT:

The use case diagram was designed successfully.

EX NO:8


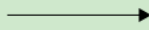









ACTIVITY DIAGRAM

AIM:

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

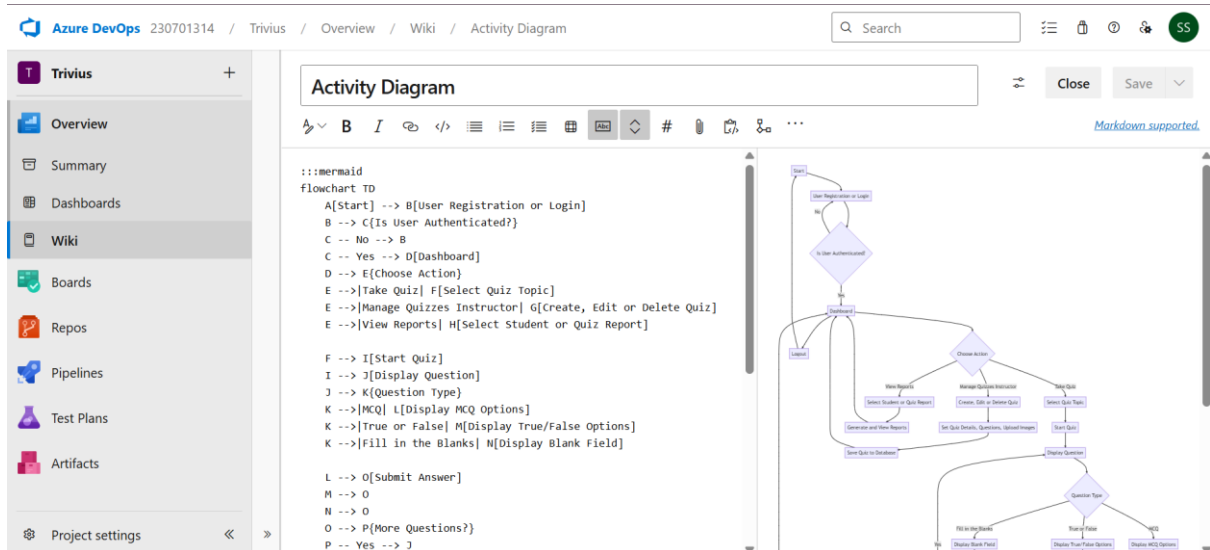
THEORY:

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



```

:::mermaid

```

```

flowchart TD

```

```

  A[Start] --> B[User Registration or Login]

```

```

  B --> C{Is User Authenticated?}

```

```

  C -- No --> B

```

```

  C -- Yes --> D[Dashboard]

```

```

  D --> E{Choose Action}

```

```

  E -->|Take Quiz| F[Select Quiz Topic]

```

```

  E -->|Manage Quizzes Instructor| G[Create, Edit or Delete Quiz]

```

```

  E -->|View Reports| H[Select Student or Quiz Report]

```

```

  F --> I[Start Quiz]

```

```

  I --> J[Display Question]

```

```

  J --> K{Question Type}

```

```

  K -->|MCQ| L[Display MCQ Options]

```

```

  K -->|True or False| M[Display True/False Options]

```

```

  K -->|Fill in the Blanks| N[Display Blank Field]

```

```

  L --> O[Submit Answer]

```

```

  M --> O

```

```

  N --> O

```

```

  O --> P{More Questions?}

```

```

  P -- Yes --> J

```

```

  P -- No --> Q[Submit Quiz]

```

```

  Q --> R[Calculate Score]

```

```

  R --> S[Save Score to Database]

```

```

  S --> T[Show Result to User]

```

```

  G --> U[Set Quiz Details, Questions, Upload Images]

```

```

  U --> V[Save Quiz to Database]

```

```

  H --> W[Generate and View Reports]

```

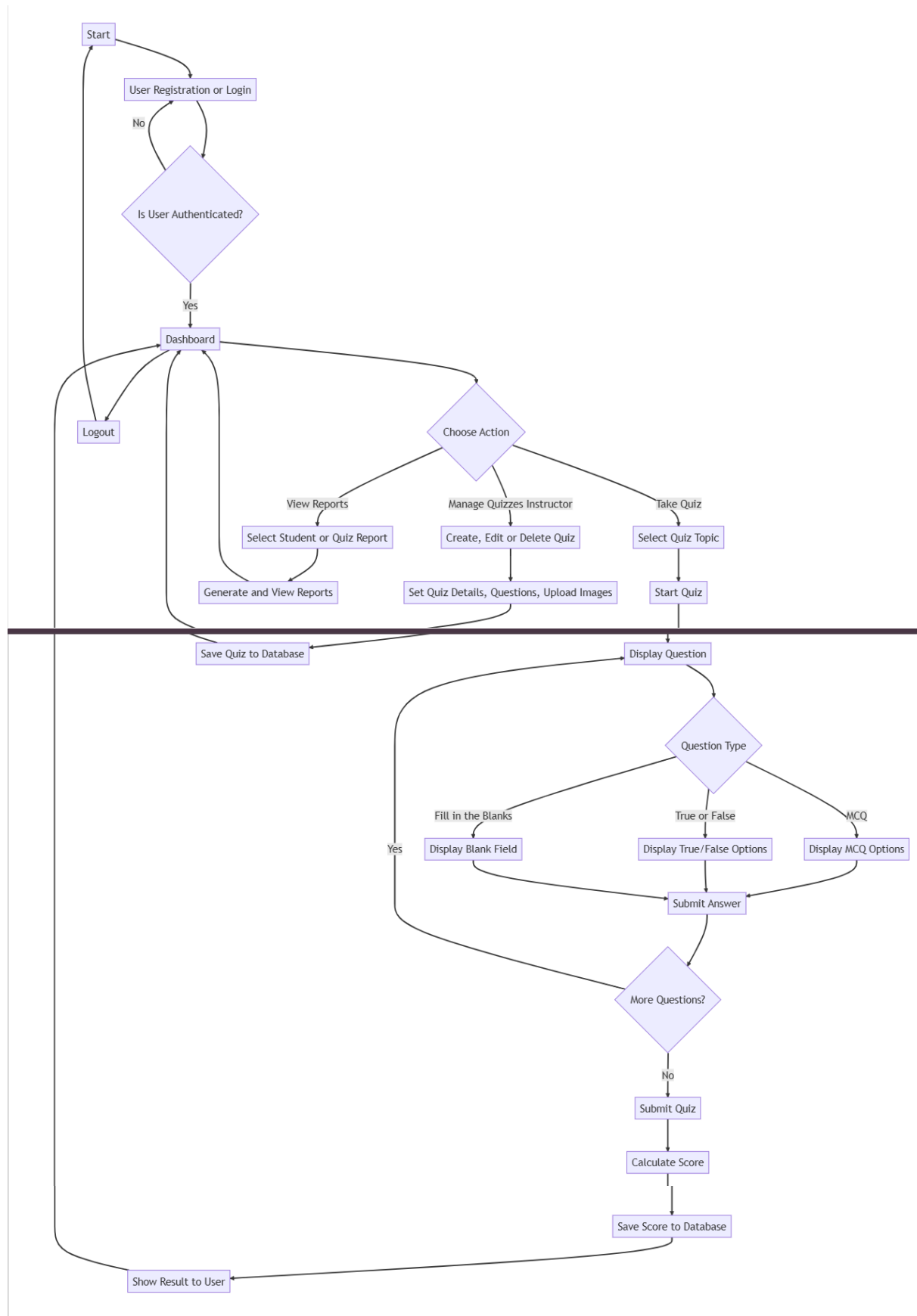
$T \rightarrow D$

$V \rightarrow D$

$W \rightarrow D$

$D \rightarrow Z[\text{Logout}]$

$Z \rightarrow A$



RESULT:

The activity diagram was designed successfully.

EX NO: 9

ARCHITECTURE DIAGRAM

AIM:

Steps to draw the Architecture Diagram using draw.io.

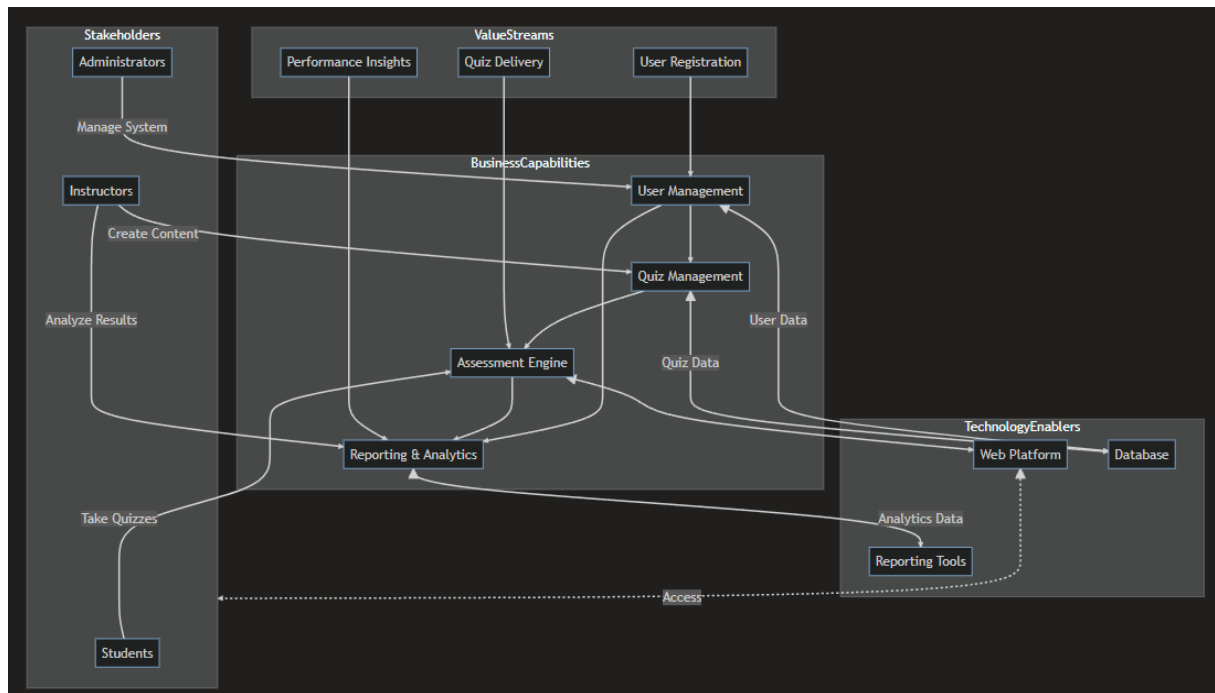
Theory:

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 given project.

INTERFACE:

Login page:

Trivius

HomeAbout

LoginSign Up

Login

Enter your email and password to access your account

Email

Password

Forgot password?

Sign In

Don't have an account? Sign up

Demo Accounts:

Use Student DemoUse Teacher Demo

Teacher Dashboard:

Trivius

HomeAdmin DashboardAbout

Hi, shradhaLogout

Teacher Dashboard

View and manage student results

Total Students

6

Average Score

5/30

Highest Score

25/30

Lowest Score

0/30

Student Results

Search by name or email

Name

Export CSV

STUDENT NAME	EMAIL	LEVEL 1	LEVEL 2	LEVEL 3	OVERALL	LAST ATTEMPT
John	john@gmail.com	0/10	0/10	0/10	0/30	No attempts

Student Dashboard:

Trivius

[Home](#) [My Dashboard](#) [About](#)

Hi, Shruthi [Logout](#)

Student Dashboard

Welcome, Shruthi

[Quiz Levels](#) [My Results](#)

Level 1: Easy

Easy

Basic general knowledge questions suitable for beginners.

10 questions

Your score: 8/10

Restart

Level 2: Medium

Medium

Intermediate questions that require broader knowledge.

10 questions

Your score: 9/10

Restart

Level 3: Hard

Hard

Advanced questions that challenge even knowledgeable participants.

10 questions

Your score: 8/10

Restart

Edit with Lovable

Quiz Attempt:

Trivius

[Home](#) [My Dashboard](#) [About](#)

Hi, sid [Logout](#)

Level 2 Quiz

Question 1 of 10

Who invented the lightbulb?

A

 Alexander Graham Bell

B

 Thomas Edison

C

 Nikola Tesla

D

 Isaac Newton

Time Remaining

00:48

Next Question

QuizMaster

Test your knowledge with our interactive quizzes. Progress through difficulty levels and track your growth.

Quick Links

[Home](#)
[Dashboard](#)
[About](#)

Contact Us

If you have any questions or feedback, please reach out to us.
Email: support@quizmaster.com

© 2025 QuizMaster. All rights reserved.

Quiz Results:

Trivius

HomeMy DashboardAbout

Hi, sidLogout

Quiz Completed

Quiz Results

Level 2 - Keep practicing!

Score

3/10

30%

Time Spent

4m 26s

~27s per question

Accuracy

✓ 3 : ✗ 7

Question Summary

✗ Question 1

Not answered

✗ Question 2

Your answer: D

✗ Question 3

Your answer: C

✗ Question 4

Your answer: B

✗ Question 5

Your answer: A

✓ Question 6

Your answer: B

✗ Question 7

Your answer: D

✗ Question 8

Your answer: A

✓ Question 9

Your answer: C

✓ Question 10

Your answer: C

Back to Dashboard

Student Report:

Trivius

HomeMy DashboardAbout

Hi, ShruthiLogout

Student Dashboard

Welcome, Shruthi

Quiz Levels

My Results

Your Quiz Results

Performance Summary

Test Scores

Level 1:

8/10

Level 2:

9/10

Level 3:

8/10

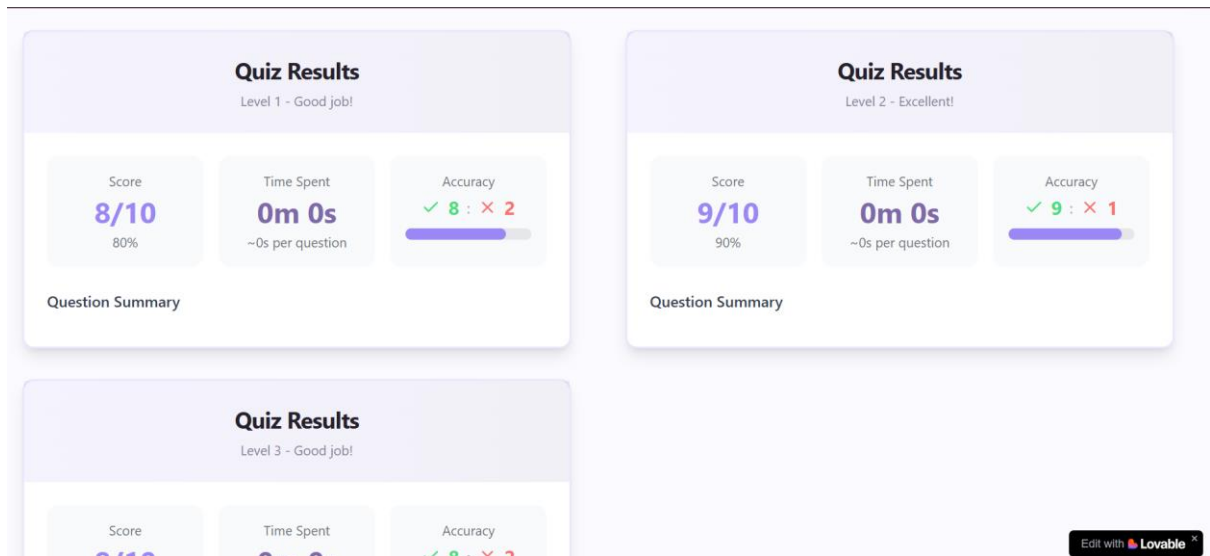
Total:

25/30

Last Attempt

5/20/2025

Edit with Lovable



RESULT:
The UI was designed successfully.

EX NO:11

IMPLEMENTATION

AIM:

To implement and deploy a web-based project titled “Quiz Master” using the Agile software development methodology, with version control, integration, and continuous deployment handled through Azure DevOps.

OVERVIEW:

This experiment focuses on the end-to-end development of a web application using the Agile approach and deploying it through Azure DevOps tools. The selected project, **Quiz Master**, is an interactive web-based quiz application designed to allow users to take quizzes, view scores, and analyze performance. Agile methodology is particularly well-suited for such iterative development, allowing continuous improvement and delivery. The practical implementation demonstrates not only software construction but also the collaborative integration and automation features of modern DevOps practices

Methodology Used:

Agile Software Development Life Cycle (Agile SDLC)

Tools and Technologies:

- **Frontend:** HTML, CSS, JavaScript (React.js)
- **Backend:** Node.js (or Python / .NET if applicable)
- **CI/CD Platform:** Azure DevOps
- **Source Control:** Git (via Azure Repos)
- **Build and Release Pipelines:** Azure Pipelines
- **Deployment Environment:** Azure App Service (or VMs)

1. Project Setup & Initialization

- i. Create a Git Repository
- ii. Initialize a new repository in Azure Repos for version control.
- iii. Clone the repository locally using git clone.
- iv. Set Up Project Structure

2. Frontend Development (React.js)

- i. Initialize React App
- ii. Key Frontend Components
- iii. Quiz List Page (Displays available quizzes)
- iv. Quiz Page (Displays questions & options)
- v. Results Page (Shows score & feedback)
- vi. Integrate API Calls
- vii. Use axios or fetch to connect with the backend.

- viii. Styling
- ix. Use CSS Modules or TailwindCSS for responsive design.

3. Backend Development (Node.js)

- i. Initialize Node.js Project
- ii. API Endpoints
- iii. GET /quizzes → Fetch all quizzes
- iv. GET /quiz/:id → Get a specific quiz
- v. POST /submit → Submit quiz answers & calculate score
- vi. Database Integration (MongoDB)
- vii. Define a Quiz Schema (questions, options, correct answers).
- viii. Use Mongoose for database operations.
- ix. Test API
- x. Use Postman or Thunder Client to verify endpoints.

4. Database Setup

- i. Choose a Database
- ii. MongoDB Atlas (Cloud-based) or Azure Cosmos DB.
- iii. Configure Connection
- iv. Store DB credentials in environment variables.

5. CI/CD Pipeline (Azure DevOps)

A. Continuous Integration (CI) Pipeline

- i. Create azure-pipelines.yml
- ii. Run Tests
- iii. Add unit tests (Jest for React, Mocha/Chai for Node.js).

B. Continuous Deployment (CD) Pipeline

- i. Deploy to Azure App Service
- ii. Configure Azure Pipelines to deploy:
- iii. Frontend → Static Web App
- iv. Backend → Azure App Service
- v. Set Up Environment Variables
- vi. Store secrets (DB connection strings) in Azure Key Vault.

6. Deployment & Monitoring

- i. Manual Testing
- ii. Verify UI, API, and database interactions.
- iii. Automated Monitoring (Optional)
- iv. Use Azure Application Insights for error tracking.
- v. Post-Deployment Checks

- vi. Ensure the app is accessible via the deployed URL.

RESULT:

Thus the application was successfully implemented.

EX NO:12**TESTING****AIM:**

To implement testing in azure.

Testing Methodology:

Following Agile Testing, we conducted functional and exploratory tests at the end of each sprint. Tests were written collaboratively and executed as part of Continuous Integration. Each feature was verified with clear test scenarios and expected outcomes.

Test ID	Test Title	Preconditions	Test Steps	Expected Result	Actual Result	Status
TC01	User Registration	App deployed; Registration page accessible	1. Open the app 2. Navigate to registration 3. Enter valid user details 4. Click "Sign Up"	User account created, redirected to login page	As expected	Pass
TC02	User Login	Registered user exists	1. Go to login page 2. Enter valid username and password 3. Click "Login"	User is redirected to dashboard	As expected	Pass
TC03	Invalid Login	Registered user exists	1. Go to login page 2. Enter incorrect credentials 3. Click "Login"	Error message displayed: "Invalid username or password"	As expected	Pass
TC04	Create New Quiz (Admin)	Logged in as Admin	1. Login as Admin 2. Go to "Create Quiz" 3. Enter quiz details and questions 4. Click "Save"	New quiz is saved and listed under quiz dashboard	As expected	Pass

TC05	Attempt Quiz (User)	Quiz exists; User logged in	1. Login as User 2. Click on a quiz 3. Answer all questions 4. Submit quiz	Quiz is submitted, score is calculated and displayed	As expected	Pass
TC06	View Quiz Results	Quiz attempt completed	1. Login 2. Navigate to "My Results" 3. Click on a past quiz	User can see past results with question breakdown	As expected	Pass
TC07	Timer Functionality	Quiz with timer enabled	1. Start a quiz 2. Wait till timer runs out without submission	Quiz auto-submits and result is shown	As expected	Pass
TC08	Access Admin Panel (Unauthorized User)	Logged in as normal user	1. Login as regular user 2. Try accessing /admin or Admin menu	Access denied or redirected with "Unauthorized" message	As expected	Pass
TC09	Dashboard Performance Load	Multiple users logged in	1. Simulate 20+ users logging in 2. Monitor dashboard responsiveness	Dashboard remains responsive, no crashes	As expected	Pass
TC10	Logout Functionality	User is logged in	1. Click "Logout" on any page	Session ends, redirected to login screen	As expected	Pass

PROCEDURE:

1. Setting Up Test Plans

Step 1: Access Test Plans

Go to Azure DevOps → Test Plans (under "Test Management").

Ensure you have the Basic + Test Plans license (required for manual testing).

Step 2: Create a Test Plan

Click + New Test Plan.

Enter:

Name: Quiz Master - Regression Test Suite

Area Path: Select your project area (e.g., QuizMaster\Testing).

Iteration: Assign to a sprint (e.g., Sprint 2).

2. Define Test Suites

Test Suites organize test cases into logical groups (e.g., by feature, priority).

Step 3: Add Test Suites

Static Suite (Fixed set of test cases)

Example: Login Functionality, Quiz Submission.

Requirement-based Suite (Linked to User Stories)

Example: As a user, I should be able to submit a quiz.

Query-based Suite (Dynamic, based on filters).

3. Create Test Cases

Test cases define step-by-step validation for features.

Step 4: Add Test Cases

Click + New Test Case under a suite.

Fill in:

Title: Verify quiz submission with correct answers.

Steps:

Action Expected Result

Select correct answers Score increases

Click "Submit" Results page displays

Assign Priority (1–4) and State (Design, Ready, Closed).

Step 5: Link to User Stories/Bugs

Use the Links tab to associate test cases with:

User Stories (e.g., US-101: Quiz Submission).

Bugs (if reproducing an issue).

4. Execute Test Cases

Step 6: Run Manual Tests

Open the Test Plan → Execute.

Select a test case → Run.

Use the Test Runner to:

Mark steps as Pass/Fail.

Add comments or attachments (screenshots).

File a Bug directly if a test fails.

Step 7: Automated Test Integration

Link automated tests (e.g., Selenium) to test cases:

In the Associated Automation tab, add the test method name.

5. Track Progress & Report

Step 8: Monitor Test Results

Dashboard Widgets:

Test Failure Analysis (Pipelines → Tests).

Test Plan Progress (Burndown charts).

Queries: Filter by Failed Tests or Active Bugs.

Step 9: Export/Share Reports

Export to Excel or generate Power BI dashboards.

TEST PLANS IN AZURE:

The screenshot shows the Azure DevOps Test Plans dashboard for the Trivius project. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines, Test Plans (selected), Test plans, Progress report, Parameters, Configurations, Runs, and Project settings. The main area displays a table of test plans with columns: Title, Test Plan ID, State, Area Path, Iteration, and Assigned To. A filter bar at the top allows filtering by title, state, area path, iteration, and assigned to. The table lists four test plans: Quiz Lifecycle (ID 120), Instructor Functionality (ID 113), Student Functionality (ID 107), and Authentication (ID 101). All are in an 'Active' state and assigned to Trivius.

Title	Test Plan ID	State	Area Path	Iteration	Assigned To
Quiz Lifecycle	120	Active	Trivius	Trivius	230701314
Instructor Functionality	113	Active	Trivius	Trivius	KK Shyaam
Student Functionality	107	Active	Trivius	Trivius	Shruthi S
Authentication	101	Active	Trivius	Trivius	230701314

The screenshot shows the details of the 'Student Functionality' test plan (ID: 108). The left sidebar is the same as the previous screenshot. The main area is divided into two sections: 'Test Suites' and 'Test Cases'. The 'Test Suites' section shows a filter by name and a list of test suites, with 'Student Functionality (4)' selected. The 'Test Cases' section shows a list of test cases with columns: Title, Order, and a checkbox. The test cases are: View available quizzes (Order 1), Start and attempt a quiz (Order 2), Submit quiz before timeout (Order 3), and View performance report (Order 4).

Title	Order
View available quizzes	1
Start and attempt a quiz	2
Submit quiz before timeout	3
View performance report	4

EX NO: 13

DEPLOYMENT

AIM:

To implement the CI/CD pipeline in azure.

PROCEDURE:

1. Classic Release Pipeline (GUI-based)

Step 1: Create a New Release Pipeline

Navigate to Pipelines → Releases.

Click New Pipeline (or New → New Release Pipeline).

Step 2: Add an Artifact Source

Source Type: Build (Linked to your CI pipeline).

Project: Select your project.

Source (Build Pipeline): Choose the CI pipeline that produces the build artifact.

Default Version: Latest (or specify a branch).

Step 3: Define Stages (Environments)

Example Stages:

Dev → Test → Staging → Production

Click Add Stage and select a template (e.g., Azure App Service Deployment).

Step 4: Configure Deployment Tasks

Example Task: Deploy to Azure App Service

yaml

- task: AzureWebApp@1

inputs:

```
azureSubscription: '<your-azure-connection>'

appType: 'webApp'

appName: '<your-app-name>'

package: '$(System.DefaultWorkingDirectory)/**/*.zip'
```

Step 5: Connect Azure Subscription

Go to Project Settings → Service Connections.

Click New Service Connection → Azure Resource Manager (ARM).

Enter Azure credentials (Service Principal or Managed Identity).

Step 6: Enable Continuous Deployment Trigger

In the Artifacts section, click the Lightning Bolt (⚡) icon.

Toggle Enable Continuous Deployment Trigger.

Optionally, add branch filters (e.g., main).

Step 7: Save & Run the Release Pipeline

Click Save and manually trigger a release or wait for a new CI build.

2. YAML-based Continuous Delivery

Step 1: Define CD in azure-pipelines.yml

```
yaml
```

```
trigger:
```

```
- main
```

```
stages:
```

```
- stage: DeployToDev

  displayName: 'Deploy to Dev'

  jobs:

    - deployment: DeployWebApp

      environment: 'dev'

      strategy:

        runOnce:

          deploy:

            steps:

              - task: AzureWebApp@1

                inputs:

                  azureSubscription: '<your-azure-connection>'

                  appType: 'webApp'

                  appName: '<your-dev-app-name>'

                  package: '$(Pipeline.Workspace)/**/*.*.zip'
```

Step 2: Add Multi-Stage Deployment (Optional)

yaml

```
- stage: DeployToProd

  displayName: 'Deploy to Production'

  dependsOn: DeployToDev

  condition: succeeded()

  jobs:

    - deployment: DeployProd

      environment: 'prod'

      strategy:
```


runOnce:

deploy:

steps:

- task: AzureWebApp@1

inputs:

azureSubscription: '<your-azure-connection>'

appType: 'webApp'

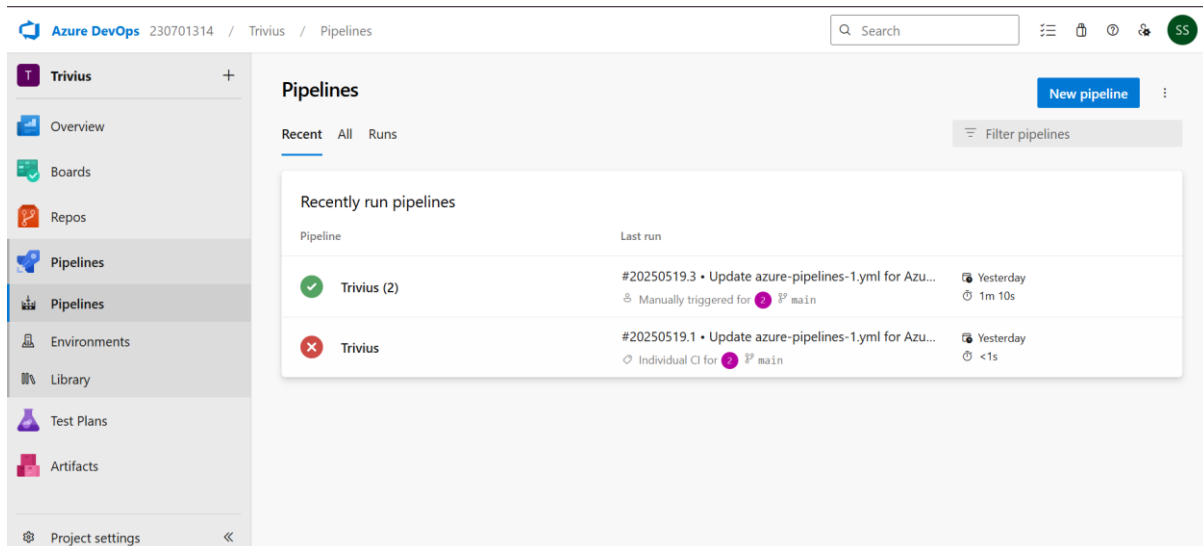
appName: '<your-prod-app-name>'

package: '\$(Pipeline.Workspace)/**/*.*.zip'

Step 3: Commit & Monitor Pipeline

Push changes to trigger the CD pipeline.

Track deployments in Pipelines → Environments.



Trivius

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Library

Test Plans

Artifacts

Project settings

Trivius (2)

RunsBranchesAnalytics

Description	Stages	
<div>#20250519.3 • Update azure-pipelines-1.yml for Azure Pipelines</div> <div>Manually triggered for main e2d97def</div>	<div></div>	<div>Yesterday</div> <div>1m 10s</div>
<div>#20250519.2 • Update azure-pipelines-1.yml for Azure Pipelines</div> <div>Individual CI for main e2d97def</div>	<div></div>	<div>Yesterday</div> <div>2m 29s</div>
<div>#20250519.1 • Set up CI with Azure Pipelines</div> <div>Individual CI for main 2913e489</div>	<div></div>	<div>Yesterday</div> <div><1s</div>

Trivius

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Library

Test Plans

Artifacts

Project settings

Trivius (2)

20250519.2

#20250519.2 • Update azure-pipelines-1.yml for Azure Pipelines

Trivius (2)

Run new

This run is being retained as one of 3 recent runs by main (Branch).View retention leases

SummaryCode Coverage

Individual CI by 230701314

Repository and version

Trivius

main e2d97def

Time started and elapsed

Yesterday at 8:26 PM

2m 29s

Related

0 work items

0 artifacts

Tests and coverage

Get started

View 9 changes

Jobs

Name	Status	Duration
Job	Success	2m 18s

Trivius

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Library

Test Plans

Artifacts

Project settings

Trivius (2)

20250519.2

Jobs in run #20250...

Trivius (2)

Jobs

Job2m 18s

Initialize job17s

Checkout Trivius...7s

Install Node.js51s

npm install and ...59s

Post-job: Check...<1s

Finalize Job<1s

Report build sta...<1s

Job

View raw log

```
1 Pool: Default
2 Agent: SHRUTHI
3 Started: Yesterday at 8:26 PM
4 Duration: 2m 18s
5
6 Job preparation parameters
```

Azure DevOps 230701314 / Trivius / Pipelines / Trivius (2) / 20250519.2

Search

Trivius

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Library

Test Plans

Artifacts

Project settings

Jobs in run #20250...
Trivius (2)

Jobs

Job	Duration
Initialize job	17s
Checkout Trivius...	7s
Install Node.js	51s
npm install and ...	59s
Post-job: Check...	<1s
Finalize Job	<1s
Report build sta...	<1s

Install Node.js

View raw log

```
1 Starting: Install Node.js
2 =====
3 Task : Node.js tool installer
4 Description : Finds or downloads and caches the specified version spec of Node.js and adds it to the
5 Version : 0.247.1
6 Author : Microsoft Corporation
7 Help : https://docs.microsoft.com/azure/devops/pipelines/tasks/tool/node-js
8 =====
9 Acquiring Node called
10 Retry count on download fails: 5 Retry delay: 1000ms
11 Downloading: https://nodejs.org/dist/v20.19.2/node-v20.19.2-win-x64.7z
12 Extracting archive
13 C:\Users\HOME\Downloads\vsts-agent-win-x64-4.255.0\work\_tasks\NodeTool_31c75bbb-bcdf-4706-8d7c-4dae
14
15 7-Zip (r) 18.05 (x86) : Igor Pavlov : Public domain : 2018-04-30
16
17 Scanning the drive for archives:
18 1 file, 19094543 bytes (19 MiB)
19
20 Extracting archive: C:\Users\HOME\Downloads\vsts-agent-win-x64-4.255.0\work\_temp\00d4d082-5c2b-4e17
21 --
```

RESULT:

Thus CI/CD pipeline is implemented successfully.

