



**RAJALAKSHMI
ENGINEERING COLLEGE**

An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

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

EVEN SEMESTER



CS23432 SOFTWARE ENGINEERING LAB

LAB MANUAL

SECOND YEAR

FOURTH SEMESTER

2024- 2025

EVEN SEMESTER

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	3	
2		Writing Problem Statement	5	
3		Designing Project using AGILE-SCRUM Methodology by using Azure.	8	
4		Agile Planning	12	
5		User stories – Creation	15	
6		Architecture Diagram Using AZURE	18	
7		Designing Usecase Diagram using StarUML	20	
8		Designing Activity Diagrams using StarUML	22	
9		Designing Sequence Diagrams using StarUML	24	
10		Design Class Diagram	28	
10		Design User Interface	30	
11		Implementation – Design a Web Page based on Scrum Methodology	36	
12		Testing	22	

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: “-“

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.

EXP: NO: 02

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

Problem Statement:

Problem Statement: Development of a Social Networking Platform

In the modern digital era, people increasingly rely on online platforms to build and maintain personal and professional relationships. However, many existing social networking platforms are either too generic, overly commercialized, or fail to cater to specific user needs like privacy, ease of use, or customization. Additionally, smaller communities and interest-based groups often lack a dedicated space that allows for secure, seamless, and engaging interaction.

The problem is the lack of a versatile, user-friendly, and privacy-conscious social networking platform that enables individuals to:

- Connect and interact with friends, family, or professional contacts.
- Share thoughts, media, and updates securely.
- Form or join communities of shared interests.
- Manage privacy and content preferences effectively.

Objective:

To design and develop a web-based social networking platform that:

- Allows users to register, create profiles, and manage their network of connections.
- Supports multimedia sharing (text, images, videos).
- Enables real-time messaging and notifications.
- Provides options to join or create public/private groups or communities.
- Includes features for content moderation and user privacy controls.
- Is built using web technologies like HTML, CSS, JavaScript, and potentially backend technologies for data management.

Result:

The problem statement was written successfully.

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>

Microsoft Azure

Search resources, services, and docs (G+)

Copilot

231701056@rajalakshm...

DEFAULT DIRECTORY

Home >

Azure DevOps ...

We've made it easier to manage Azure DevOps billing and subscriptions. You can [set up billing](#), [change your subscription](#) or pay for more users and resources within Azure DevOps. [Learn more](#)

Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services

[My Azure DevOps Organizations](#)

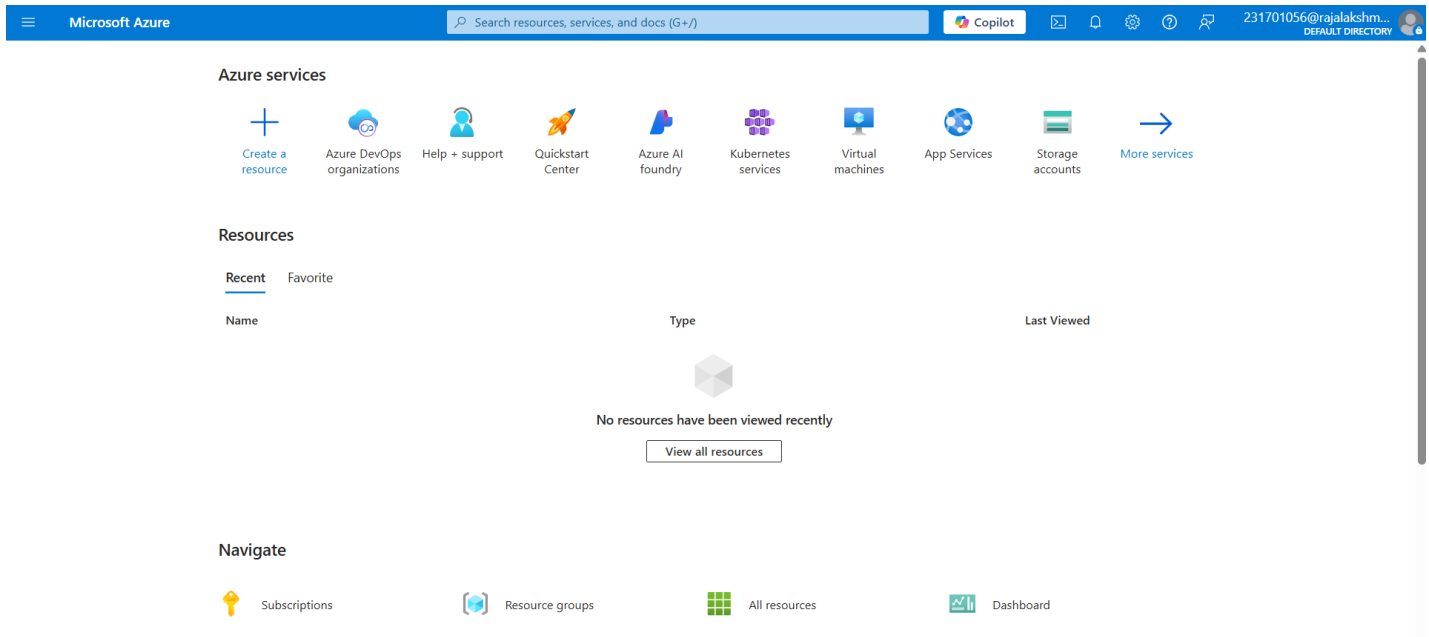
[Get started using Azure DevOps](#)

[Billing management for Azure DevOps](#)

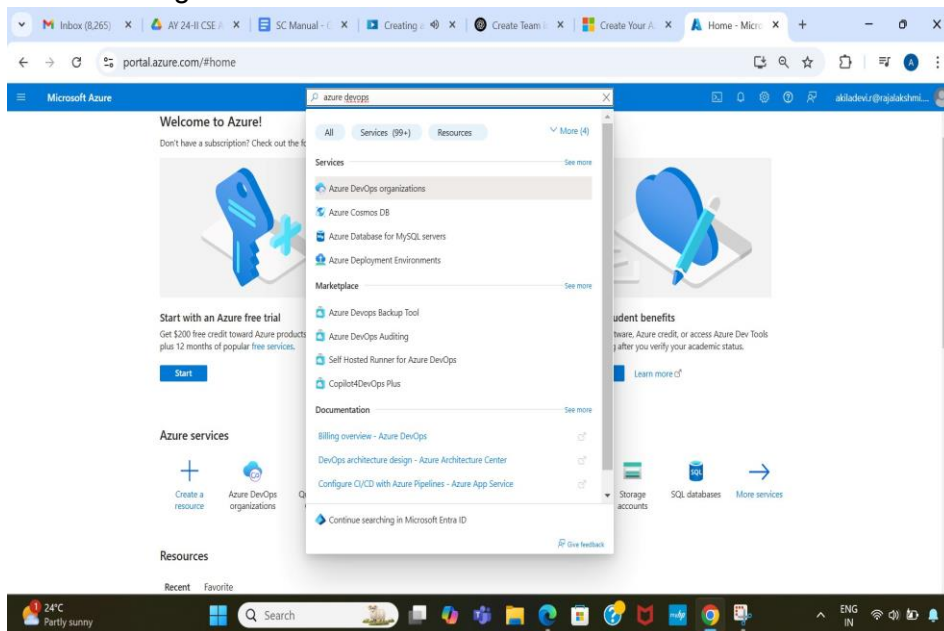
[Give feedback](#)

[Tell us about your experience with the Azure DevOps page](#)

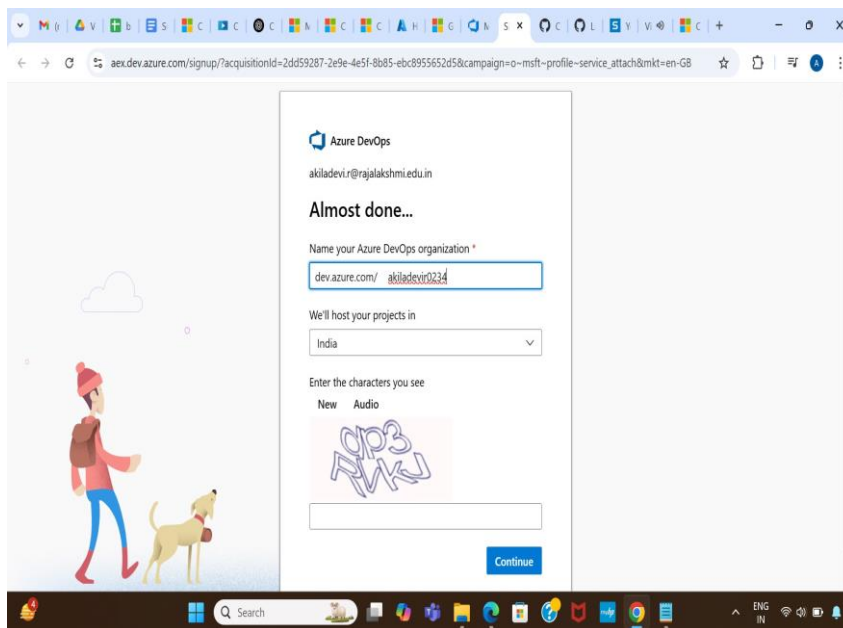
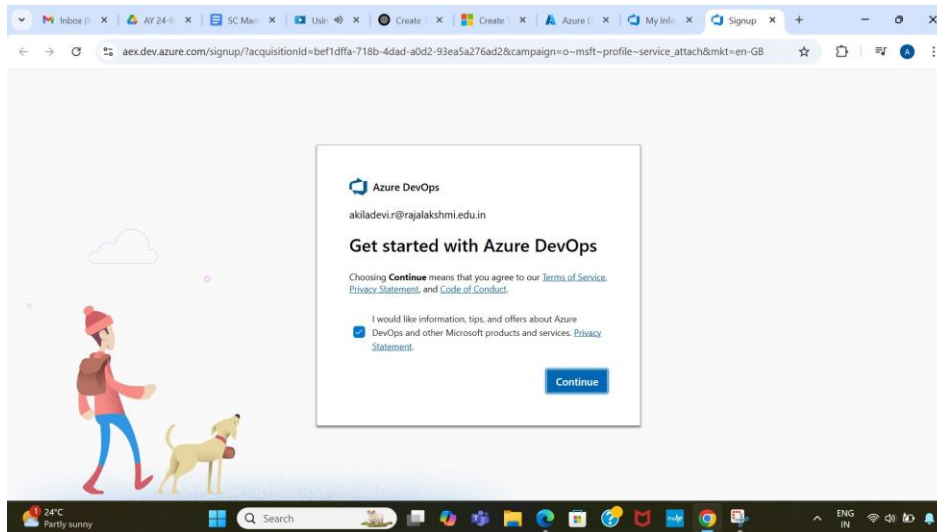
3. Azure home page



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



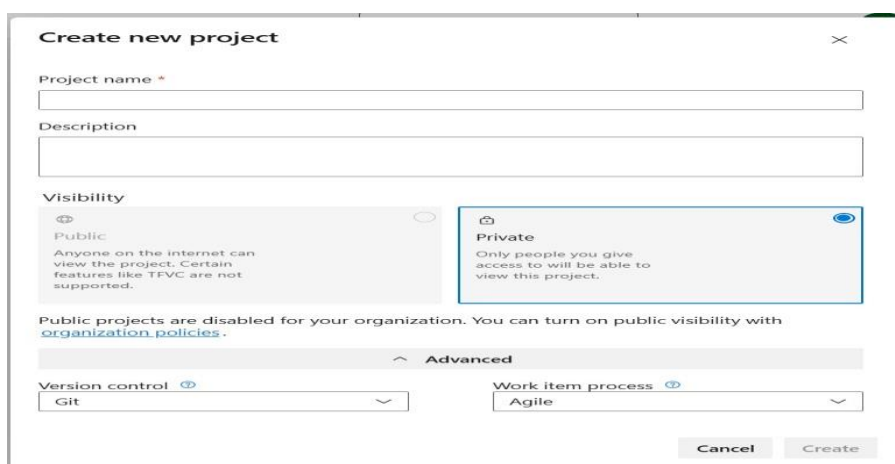
5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.



6. Create the First Project in Your Organization

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

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



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



Sujitha K

Edit profile

231701056@rajalakshmi.edu.in

Microsoft account

India

231701056@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/SOCIALNETWORKPLATFORM (Owner)

Projects

SOCIAL NETWORK PLATFORM

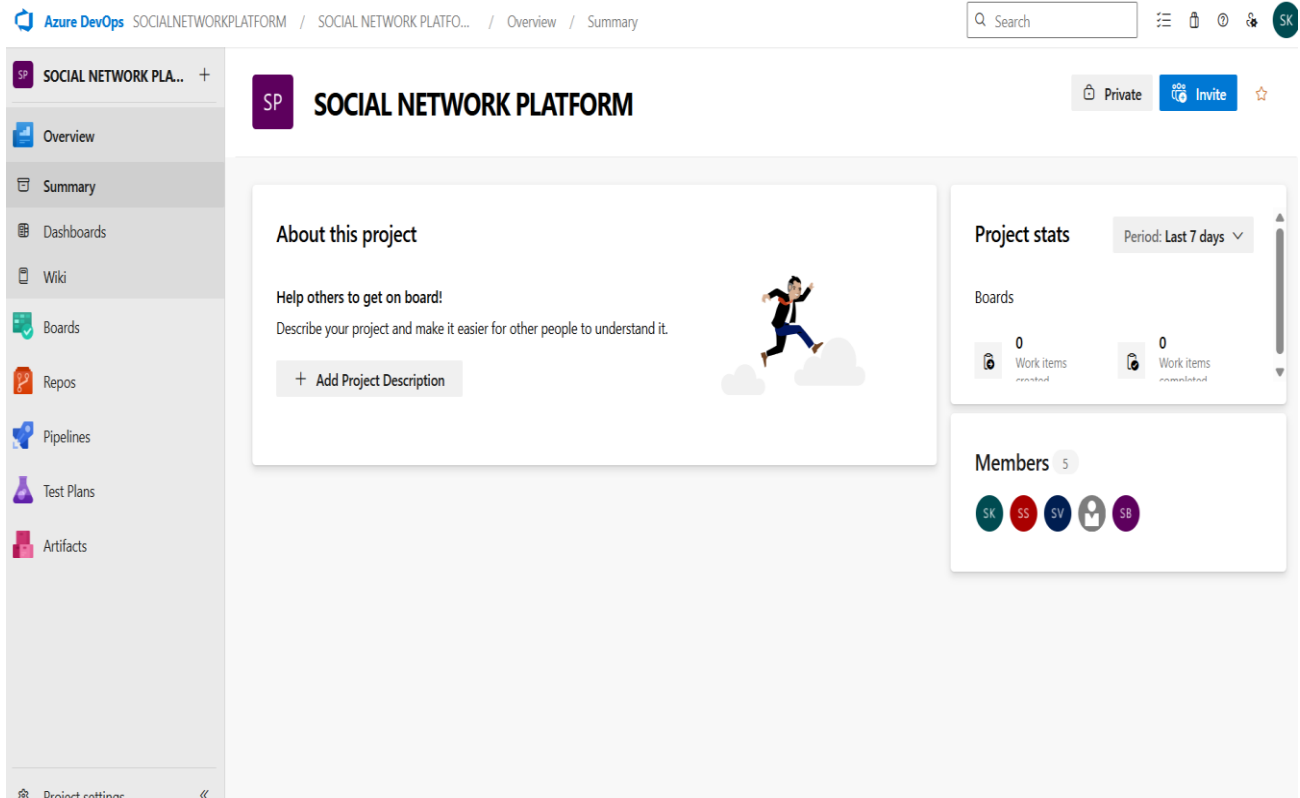
New project

Actions

[Open in Visual Studio](#)

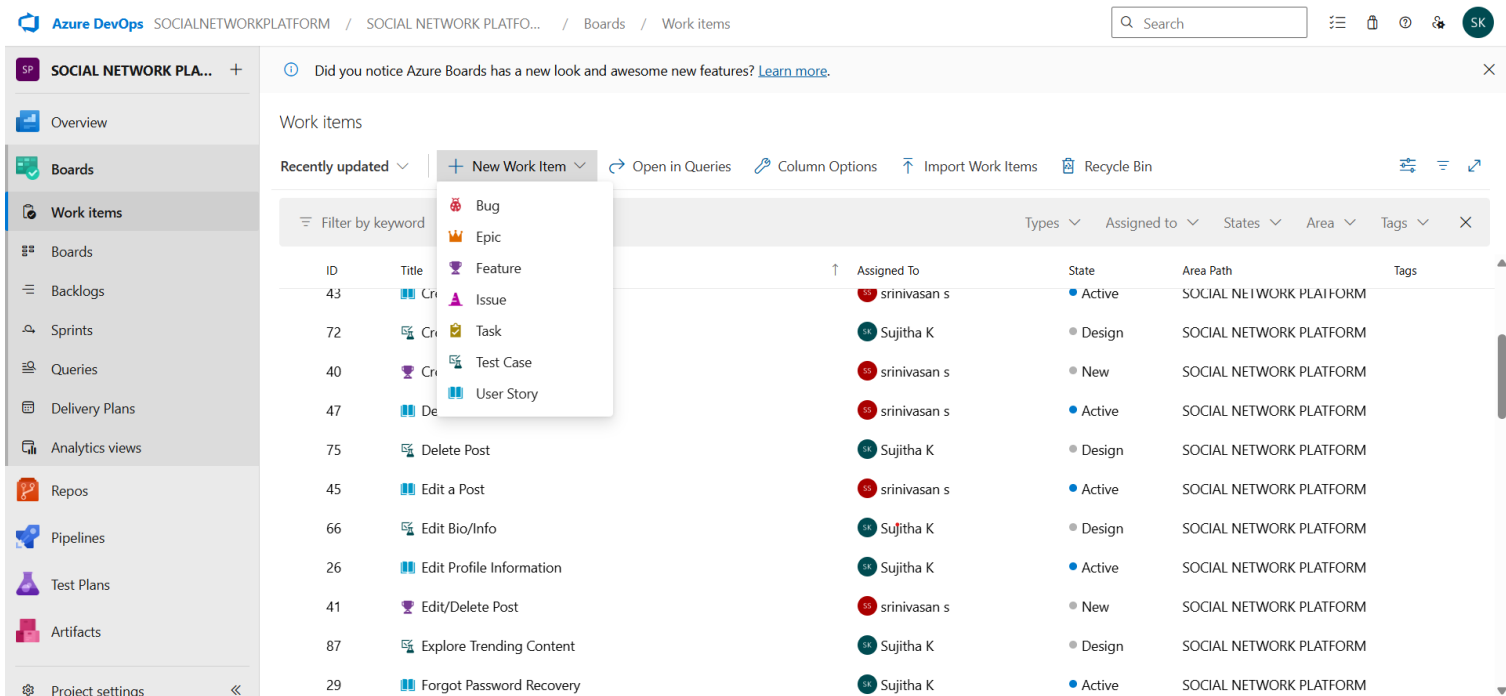
dev.azure.com/TEAM-14 (Owner)

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 Boards interface for editing a work item. On the left is a sidebar with navigation options: Overview, Boards, Work items, Boards, Backlogs, Sprints, Queries, Delivery Plans, Analytics views, Repos, Pipelines, Test Plans, and Artifacts. The main area displays the work item 'TEST CASE 66' with the title 'Edit Bio/Info' and the author 'Sujitha K'. It includes a '0 Comments' section and an 'Add Tag' button. The work item details show 'State: Design', 'Area: SOCIAL NETWORK PLATFORM', and 'Reason: New'. The 'Steps' section contains a single step: '1. Navigate to Profile → Edit → Save' with the 'Expected result' of 'Updated info is shown'. To the right, there are tabs for 'Steps', 'Summary', and 'Associated Automation'. Below the steps, there are sections for 'Deployment' and 'Development', each with an 'Add link' button and instructions on how to link Azure Repos items.

SOCIAL NETWORK PLA... +

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

Recently updated | [Back to Work Items](#) 15 of 53 ↑ ↓

TEST CASE 66 Edit Bio/Info

Sujitha K 0 Comments Add Tag Save Follow Settings Refresh Undo More

Updated by Sujitha K: May 11

State: Design Area: SOCIAL NETWORK PLATFORM Reason: New Iteration: SOCIAL NETWORK PLATFORM

Steps Summary Associated Automation 0 0

Steps

Steps Action Expected result

1. Navigate to Profile → Edit → Save Updated info is shown

Click or type here to add a step

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

Link an Azure Repos [commit](#), [pull request](#) or [branch](#) to see the status of your development. You can also [create a branch](#) to get started.

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.

4.

```
sequenceDiagram
    participant User1 as "User 1"
    participant User2 as "User 2"
    participant PostSystem as "Post System"
    participant MessageSystem as "Message System"
    participant FriendSystem as "Friend System"
```

```
User1->>PostSystem: Create new post
PostSystem->>PostSystem: Save post
PostSystem-->>User1: Post ID
```

```
User1->>User2: Send friend request
FriendSystem->>User1: Store friend request
FriendSystem->>User2: Notify user about request
User2->>User1: Accept friend request
FriendSystem->>User1: Update friendship status
FriendSystem->>User2: Update friendship status
```

```
User1->>MessageSystem: Send message to User2
MessageSystem->>MessageSystem: Save message
MessageSystem-->>User1: Message Sent Confirmation
MessageSystem-->>User2: New message notification
```

```
User1->>PostSystem: Like User2's post
PostSystem->>PostSystem: Update like count
PostSystem-->>User1: Like confirmation:::
```

Explanation:

participant defines the entities involved.

->> represents a direct message.

-->> represents a response message. + after ->> activates a participant.

- after -->> deactivates a participant.

alt / else for conditional flows. loop can be used for repeated actions.

-> Solid line without arrow

--> Dotted line without arrow

->> Solid line with arrowhead

-->> Dotted line with arrowhead

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

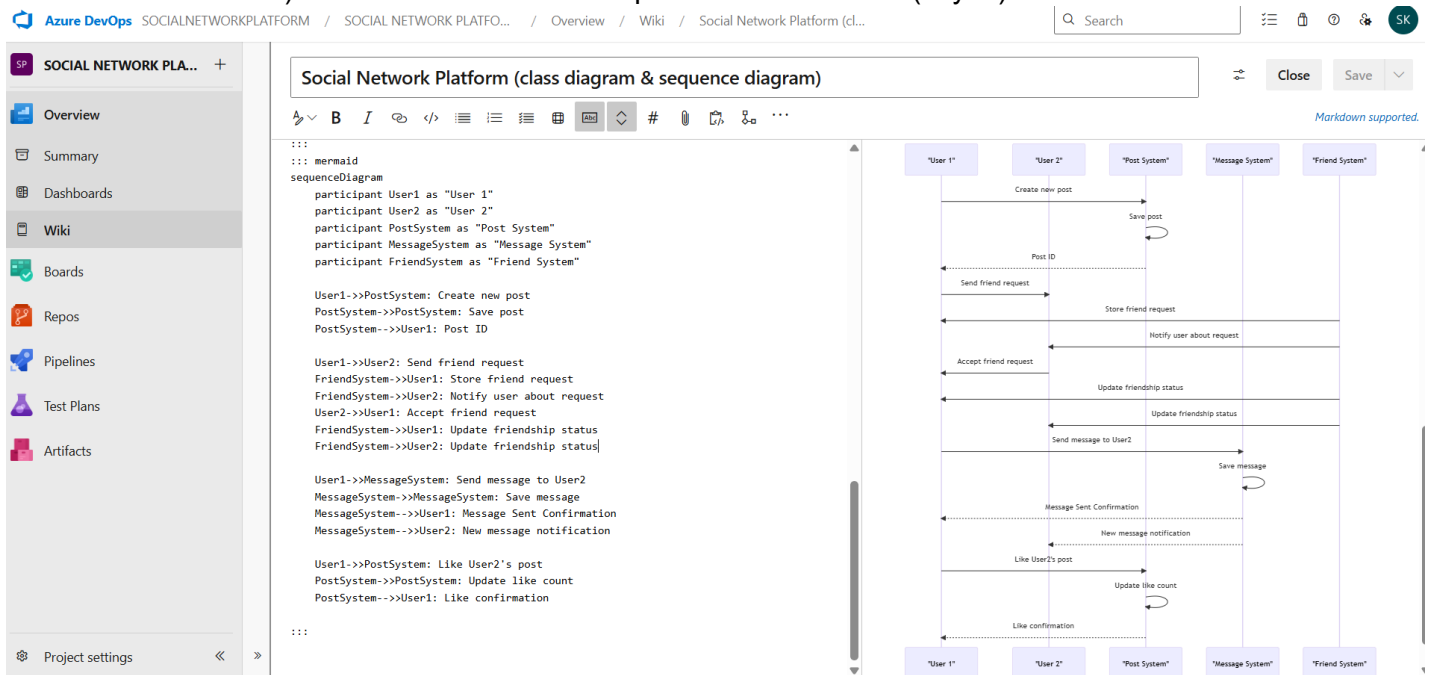
<<-->> Dotted line with bidirectional arrowheads (v11.0.0+)

-x Solid line with a cross at the end

--x Dotted line with a cross at the end

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

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



4. click wiki menu and select the page

Result:

The sequence diagram was drawn successfully.

EX NO. 6

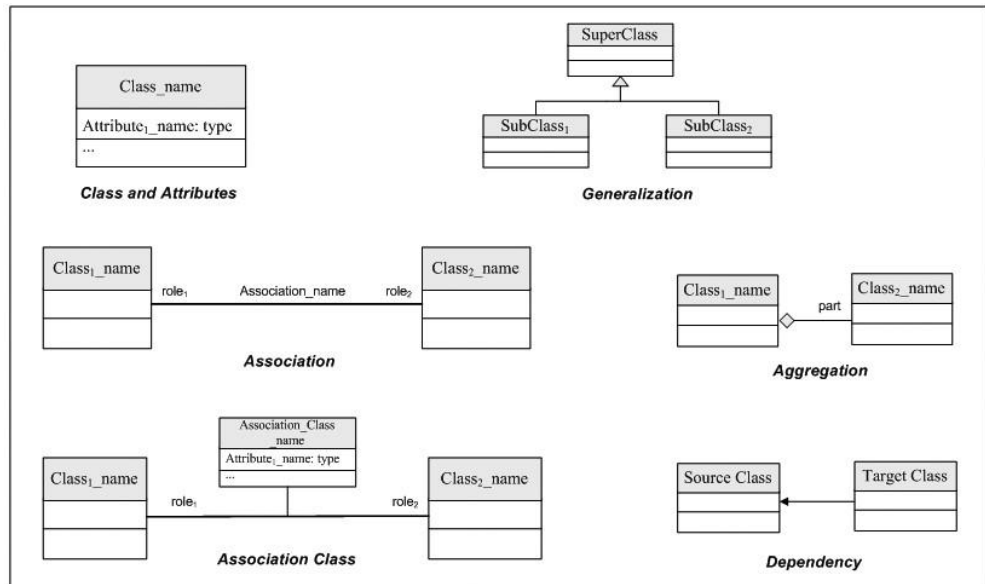
CLASS DIAGRAM

AIM :-

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

THEORY

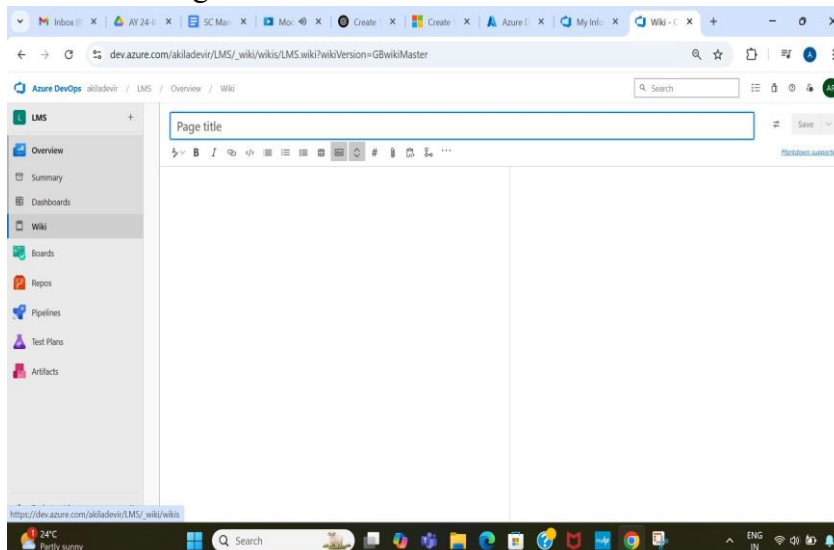
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



Notations in class diagram

Procedure:

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



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

```

::: mermaid
classDiagram
    class User {
        +int userId
        +string userName
        +string email
        +string password
        +string profilePicture
        +list posts
        +list friends
        +list messages
        +followUser(user: User)
        +sendMessage(to: User, message: string)
        +addFriend(friend: User)
    }

    class Post {
        +int postId
        +string content
        +date timestamp
        +User author
        +list comments
        +likePost(user: User)
        +addComment(comment: string)
    }

    class Comment {
        +int commentId
        +string content
        +date timestamp
        +User commenter
    }

    class Message {
        +int messageId
        +string content
        +date timestamp
        +User sender
        +User receiver
        +readMessage()
    }

    class Friendship {
        +User user1
        +User user2
        +date friendshipDate
        +acceptRequest()
        +removeFriendship()
    }

```

```

User "1" -- "0..*" Post : creates
Post "1" -- "0..*" Comment : has
User "1" -- "0..*" Message : sends
User "1" -- "0..*" Friendship : has

```

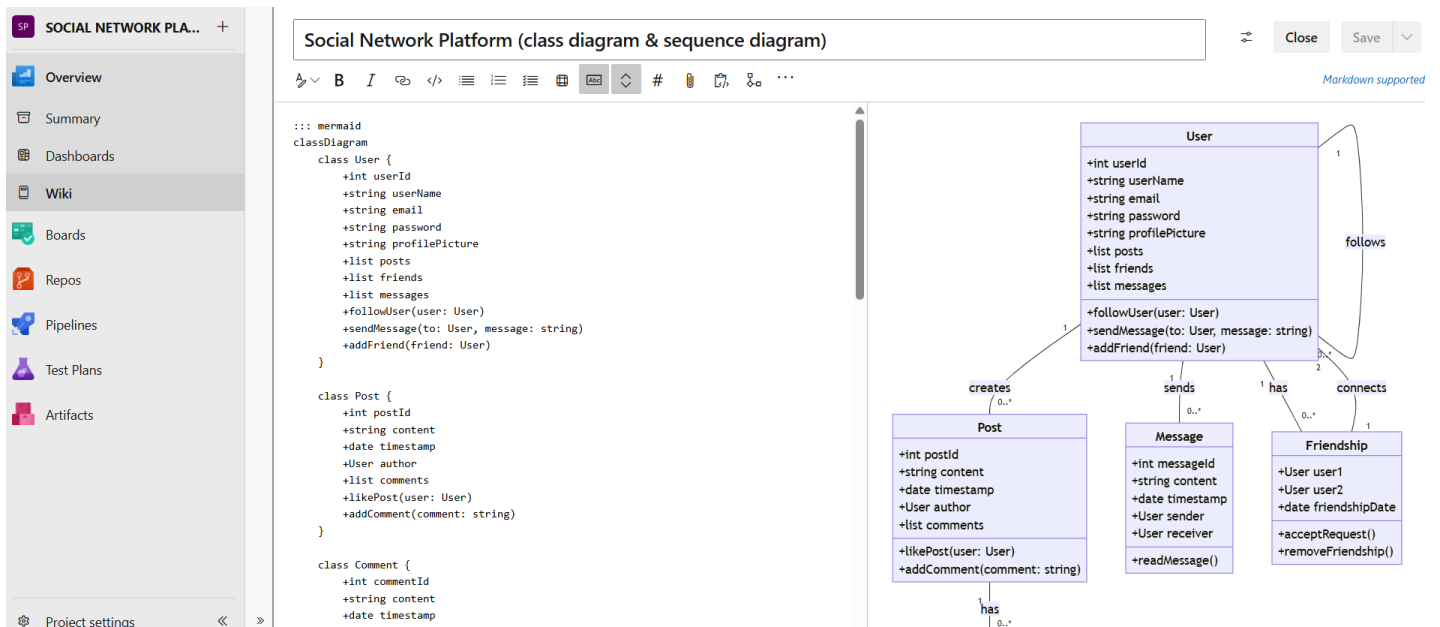
Friendship "1" -- "2" User : connects
User "1" -- "0..*" User : follows

...

Relationship Types

Type Description

- <| Inheritance
- * Composition
- o Aggregation
- > Association
- < Association
- |> Realization



Result:

The use case diagram was designed successfully.

EX NO: 7

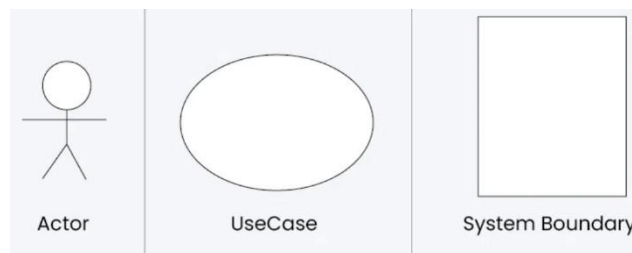
USECASE DIAGRAM

Aim:

Steps to draw the Use Case Diagram using draw.io

Theory:

- UCD shows the relationships among actors and use cases within a system which Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model and communicate the scope of a development project
- **Use Cases**
- **Actors**
- **Relationships**
- **System Boundary Boxes**



Procedure

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

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

Step 2: Upload the Diagram to Azure DevOps

Option 1: Add to Azure DevOps Wiki ● Open Azure DevOps and go to your project.

- Navigate to Wiki (Project > Wiki).
- Click "Edit Page" or create a new page.
- Drag & Drop the exported PNG/JPG image.
- Use Markdown to embed the diagram:
- ![Use Case Diagram](attachments/use_case_diagram.png)

Option 2: Attach to Work Items in Azure Boards

- Open Azure DevOps → Navigate to Boards (Project > Boards).
- Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram.
- Add comments or descriptions to explain the use case.

Actors:

- **User:** Regular person who uses the platform.
- **Admin:** Manages the platform and oversees user content.

Use Cases (Main Functions):

1. **Register / Login**
2. **Create/Update Profile**
3. **Send Friend Request**
4. **Accept/Reject Friend Request**
5. **Post Content** (text/image/video)
6. **Comment / Like Post**
7. **Create/Join Group**
8. **Chat with Friends**
9. **View Notifications**
10. **Report User/Post**
11. **Manage Reported Conte**
12. **nt** (*Admin only*)
13. **Ban/Unban User** (*Admin only*)

EX NO. 8



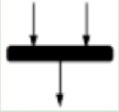








ACTIVITY DIAGRAM

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 Logs In]
    B --> C{Authenticated?}
    C -- No --> X[Show Error]
    C -- Yes --> D[Show Feed]
    D --> E[Click Create Post]
    E --> F[Enter Content]
    F --> G[Click Post Button]
    G --> H[Validate Input]
    H --> I{Valid?}
    I -- No --> Y[Show Validation Error]
    I -- Yes --> J[Save Post]
    J --> K[Update Feed]
    K --> L([End])

```

...

Azure DevOps SOCIALNETWORKPLATFORM / SOCIAL NETWORK PLATFO... / Overview / Wiki / ACTIVITY DIAGRAM

Search

SOCIAL NETWORK PLA... +

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

Project settings

ACTIVITY DIAGRAM

```

::: mermaid
flowchart TD
    A([Start]) --> B[User Logs In]
    B --> C{Authenticated?}
    C -- No --> X[Show Error]
    C -- Yes --> D[Show Feed]
    D --> E[Click Create Post]
    E --> F[Enter Content]
    F --> G[Click Post Button]
    G --> H[Validate Input]
    H --> I{Valid?}
    I -- No --> Y[Show Validation Error]
    I -- Yes --> J[Save Post]
    J --> K[Update Feed]
    K --> L([End])

```

Result:

The activity diagram was designed successfully

EX NO. 9

ARCHITECTURE DIAGRAM

Aim:

Steps to draw the Architecture Diagram using draw.io.

Theory:

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



Procedure

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

graph LR

```
A[Client (Web/Mobile App)] --> B[Frontend (HTML/CSS/JavaScript)]
B --> C[Backend API Server (Node.js/PHP/Python)]
C --> D[Authentication Service (JWT/OAuth)]
C --> E[Database (MySQL/MongoDB)]
C --> F[Media Storage (e.g., AWS S3)]
G[Admin Panel] --> C
```

Result:

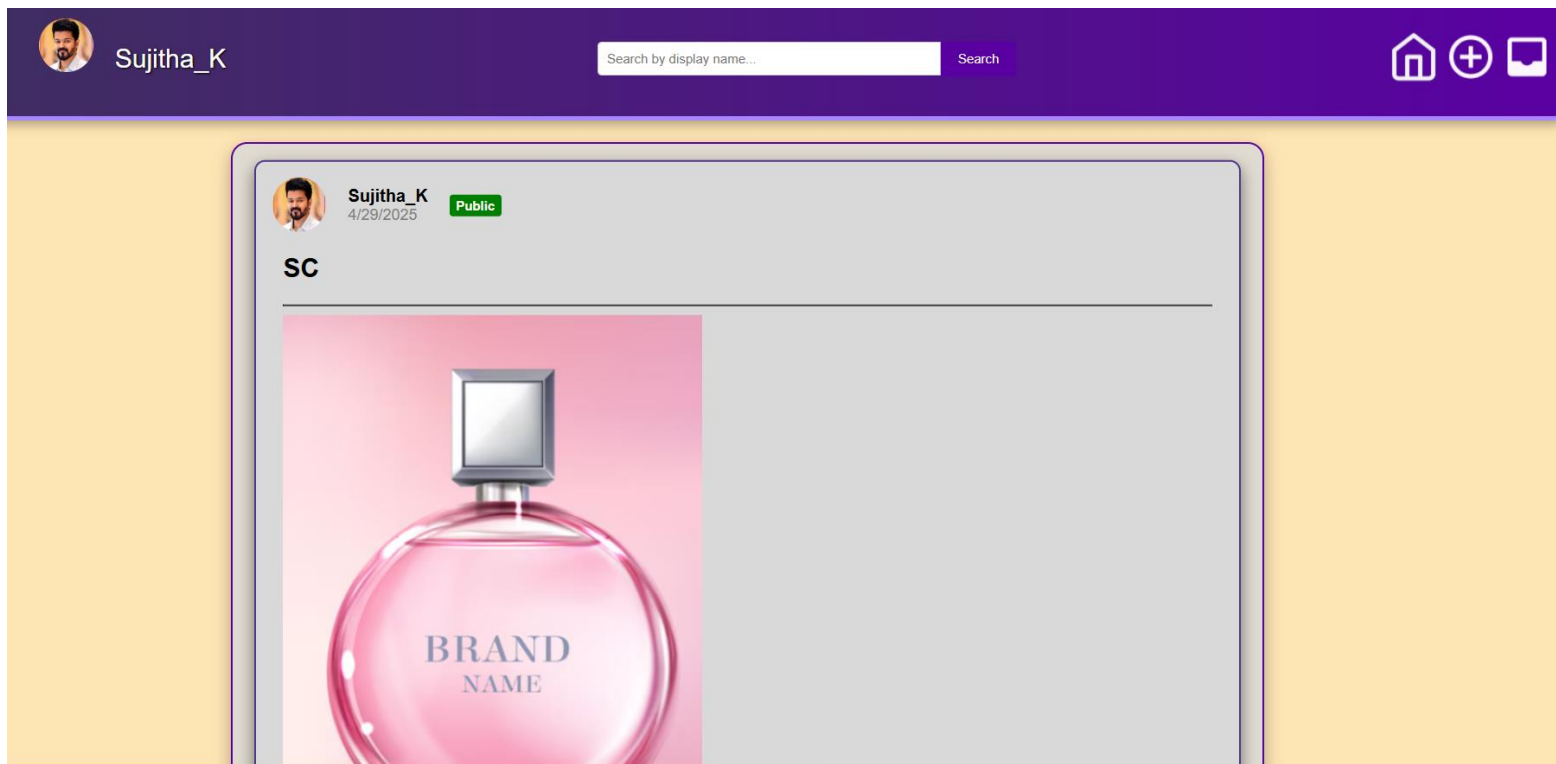
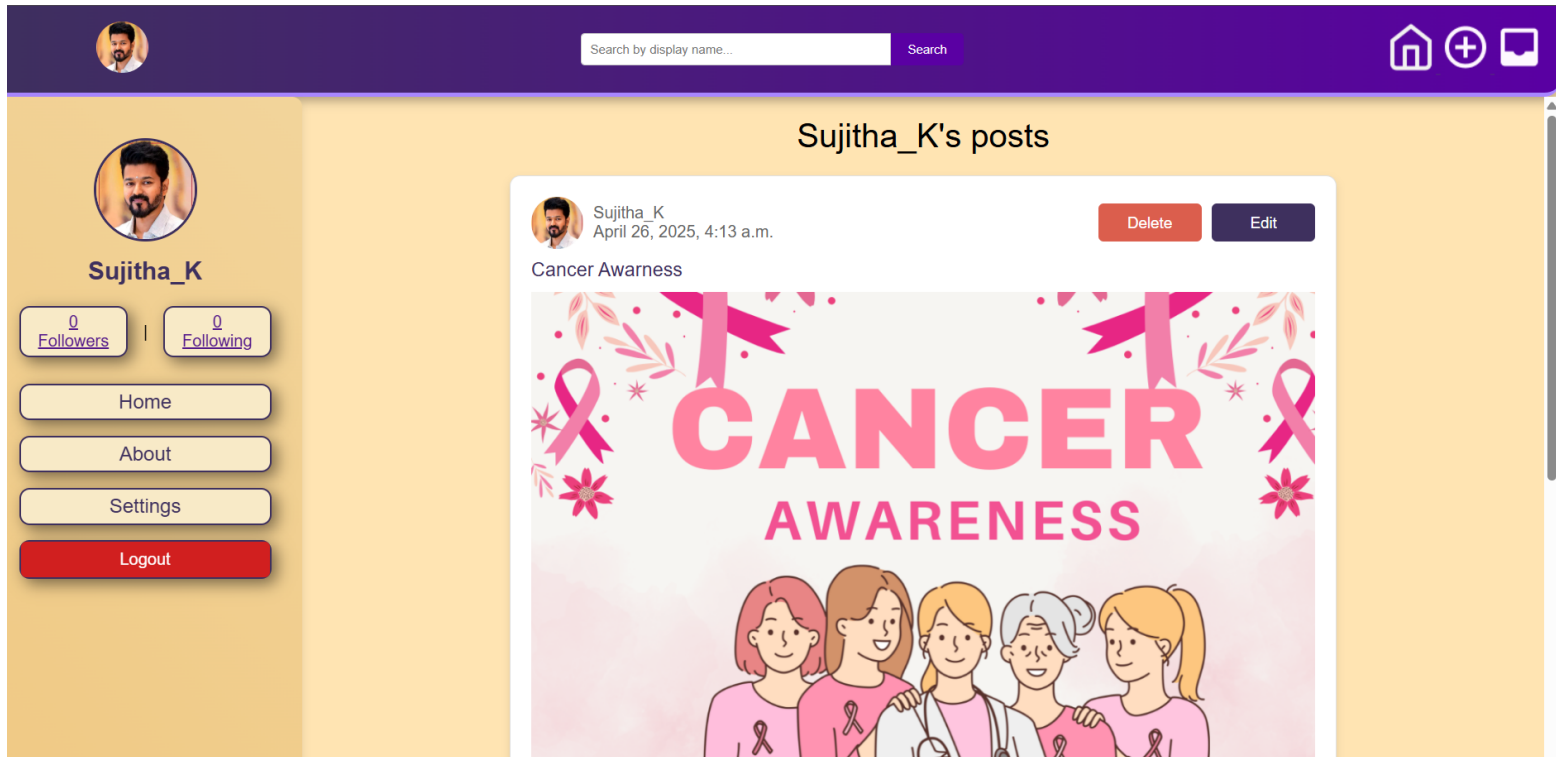
The architecture diagram was designed successfully





EX NO. 10

USER INTERFACE

Aim:

Design User Interface for the given project






Inbox


Follow Requests

Comments




Sujitha_K commented on your post

Likes







Sujitha_K liked the post "SC"

2025-04-29 07:15:10



Sujitha_K liked the post "Cancer Awarness"

2025-04-26 04:13:32



Create New Post

Title

Enter the title of the post

Description

Enter the description of the post

Visibility

Public

▼

Content Type

Plain Text


▼


Content

Enter the content of the post



Sujitha_K

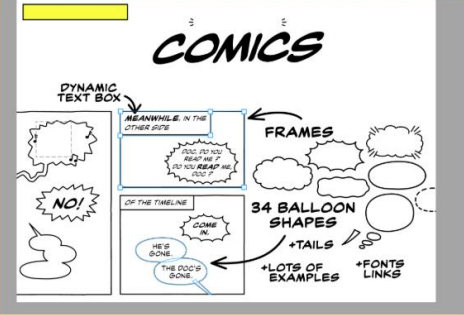




sriniv

April 24, 2025, 6:40 p.m.

sc project





1 like(s)



1 comment(s)



share

Comments



Sujitha_K 4/26/2025

WOW

Add a comment ...

Add Comment

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.

[illegible]

Result

Thus the application was successfully implemented.