

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



**RAJALAKSHMI**  
**ENGINEERING COLLEGE**

**CS23432 SOFTWARE CONSTRUCTION**  
**LABORATORY**

**Laboratory Note Book**

Name: Sailesh Rangaraj

Year / Branch / Section: 2nd YEAR / AIML/AC

University Register No. :2116231501141

College Roll No: 231501141

Semester: 4rd SEMESTER

Academic Year: 2024-2025

**RAJALAKSHMI ENGINEERING COLLEGE**  
**[AUTONOMOUS]**

**RAJALAKSHMI NAGAR, THANDALAM – 602 105**

**BONAFIDE CERTIFICATE**

Name: ...SAILESH RANGARAJ .....

Academic Year : .... 2024-2025. ....

Semester : 04

Branch : AIML

**Register No.**

**2116231501141**

Certified that this is the bonafide record of work done by the above student in the

**CS23432 – SOFTWARE CONSTRUCTION** during the year 2024 - 2025.

**Signature of Faculty in-charge**

Submitted for the Practical Examination held on .....

**Internal Examiner**

**External Examiner**

<b>Ex. No.</b>	<b>Title</b>	<b>Page No.</b>
<b>1</b>	<b>Study of Azure DevOps</b>	<b>4</b>
<b>2</b>	<b>Writing Problem Statement – <i>Social Media Platform</i></b>	<b>6</b>
<b>3</b>	<b>Designing Project Using Agile-Scrum Methodology with Azure</b>	<b>7</b>
<b>4</b>	<b>Agile Planning – Epics, User Stories, and Sprint Planning</b>	<b>9</b>
<b>5</b>	<b>User Story Creation in Azure DevOps</b>	<b>13</b>
<b>6</b>	<b>Sequence Diagram – Social Media Platform Transaction Flow</b>	<b>19</b>
<b>7</b>	<b>Class Diagram – Structural Design Using Mermaid.js</b>	<b>23</b>
<b>8</b>	<b>Use Case Diagram – Functional Overview</b>	<b>30</b>
<b>9</b>	<b>Activity Diagram – System Workflow</b>	<b>32</b>
<b>10</b>	<b>Architecture Diagram – Azure Deployment Design</b>	<b>33</b>
<b>11</b>	<b>User Interface Design</b>	<b>34</b>
<b>12</b>	<b>Implementation of Social Media Platform in Azure</b>	<b>37</b>

# **EX NO: 1 STUDY OF AZURE DEVOPS**

## **AIM:**

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

## **STUDY:**

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

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

Azure DevOps consists of five key services:

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

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

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

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

#### **AWS, GCP). 1.3 Azure Boards (Agile Project Management)**

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

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

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

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

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

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

**Step 1:** Create an Azure DevOps Account Visit Azure DevOps.

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

**Step 2:** Set Up a Repository (Azure Repos) Navigate to Repos.

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

**Step 3:** Configure a CI/CD Pipeline (Azure Pipelines) Go to Pipelines→  
New Pipeline.

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

**Step 4:** Manage Work with Azure Boards Navigate to Boards.

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

**Step 5:** Implement Testing (Azure Test Plans) Go to

- Test Plans.
- Create and run test cases
  - View test results and track bug.

**RESULT:**

Thus, the study for the given problem statement was successfully completed.

## **EX NO: 2 WRITING PROBLEM STATEMENT**

### **AIM:**

To prepare the PROBLEM STATEMENT for the given project.

### **PROBLEM STATEMENT:**

Social Media Platform

#### **1. User Authentication and Profile Management**

- Users should be able to sign up, log in, and reset their passwords securely.
- Users must be able to create and manage their profiles, including profile picture, bio, and personal details.
- The system should support editing user information and logging out safely.

#### **2. Content Sharing and Engagement**

- Users should be able to upload photos and videos with optional captions, tags, and location details.
- The system must allow users to like, comment on, and share posts.
- Users should be able to view a real-time feed of content posted by people they follow.

#### **3. Stories and Reels**

- The platform should support uploading short-term stories (24-hour visibility) and short-form videos (Reels).
- Users must be able to view, like, and comment on Reels and Stories.
- Stories should appear in a circular carousel at the top of the feed.

#### **4. Search and Explore**

- The application must include a search functionality for users to discover other profiles, hashtags, and trending content.
- An explore page should showcase popular and trending posts based on user interests and activity.

#### **5. Messaging and Notifications**

- The system must support direct messaging between users, including text, emojis, and media sharing.
- Users should receive notifications for likes, comments, follows, and messages in real-time.
- The system must notify users about account activities and content interactions.

## 6. Privacy and Security

- Users must be able to control who can view their profile and posts (public/private).
- The system should ensure data encryption and secure handling of user information.
- Users should have the ability to report or block other users if necessary.

## RESULT:

Thus, the problem statement for the given problem is successfully written.



## **EX NO: 3 DESIGNING PROJECT USING AGILE-SCRUM METHODOLOGY BY USING AZURE.**

### **AIM:**

To plan a agile model for the given problem statement.

### **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 designing project using agile-scrum methodology by using azure was completed successfully.

## **EX NO: 4 – AGILE PLANNING**

### **AIM:**

To plan the development of a social media application using the Agile methodology.

### **SCOPE:**

To build a mobile/web-based social media application where users can register, share posts, interact with other users through likes, comments, and messages, and manage their personal profiles. The app will also support features like stories, search functionality, and secure authentication.

## **AGILE EPICS & USER STORIES**

### **Epics:**

Epics represent large bodies of work that can be divided into smaller user stories. They generally span across multiple sprints.

### **Epic 1: User Authentication and Profile Management**

**Objective:** Allow users to sign up, log in, and manage their profile.

#### **User Stories:**

- As a user, I want to sign up with my email or phone number.
- As a user, I want to log in securely.
- As a user, I want to edit my profile (bio, photo, etc.).
- As a user, I want to log out safely.

### **Epic 2: Content Posting and Interaction**

**Objective:** Enable users to post photos/videos and interact with others' content.

#### **User Stories:**

- As a user, I want to upload photos and videos with captions.
- As a user, I want to like and comment on posts.
- As a user, I want to view a feed of posts from users I follow.

### **Epic 3: Stories and Reels**

**Objective:** Allow users to post and view short videos and temporary stories.

#### **User Stories:**

- As a user, I want to upload a story that disappears in 24 hours.
- As a user, I want to watch Reels and like or comment on them.
- As a user, I want to swipe through stories posted by others.

## **Epic 4: Search and Explore**

**Objective:** Enable discovery of new content and users through search.

### **User Stories:**

- As a user, I want to search for users and hashtags.
- As a user, I want to explore trending posts.
- As a user, I want to follow new users from the explore page.

## **Epic 5: Messaging and Notifications**

**Objective:** Allow users to chat and receive updates.

### **User Stories:**

- As a user, I want to send and receive direct messages.
- As a user, I want to get notifications when someone likes or comments on my post.
- As a user, I want to receive real-time alerts for new followers and messages.

## **Epic 6: Privacy and Security**

**Objective:** Ensure secure authentication and user control over privacy.

### **User Stories:**

- As a user, I want to enable two-factor authentication.
- As a user, I want to block or report other users.
- As a user, I want to control who can see my posts (public/private).

## **SPRINTS:**

### **Sprint 1: Authentication and Profile Setup**

**Duration:** 2 weeks

**Focus:** Signup, login, and profile management

**Epics Covered:** User Authentication and Profile Management

### **User Stories:**

- Sign up with email/phone
- Log in securely
- Edit profile
- Logout function

## **Sprint 2: Content Posting and Feed**

**Duration:** 2 weeks

**Focus:** Uploading posts and displaying feeds

**Epics Covered:** Content Posting and Interaction

**User Stories:**

- Upload photo/video with caption
- View posts from followed users
- Like and comment on posts

## **Sprint 3: Stories and Reels**

**Duration:** 2 weeks

**Focus:** Short videos and story features

**Epics Covered:** Stories and Reels

**User Stories:**

- Upload a story (24-hour visibility)
- Watch reels and stories
- Interact with story content

## **Sprint 4: Search and Explore Features**

**Duration:** 2 weeks

**Focus:** Discovery tools for users and content

**Epics Covered:** Search and Explore

**User Stories:**

- Search users and hashtags
- Explore trending content
- Follow new users from explore

## **Sprint 5: Messaging and Notifications**

**Duration:** 2 weeks

**Focus:** Real-time interactions and updates

**Epics Covered:** Messaging and Notifications

**User Stories:**

- Direct messaging system
- Notifications for likes/comments
- Alerts for new followers/messages

## **Sprint 6: Privacy and Security**

**Duration:** 2 weeks

**Focus:** User control and secure access

**Epics Covered:** Privacy and Security

**User Stories:**

- Two-factor authentication
- Blocking/reporting users
- Privacy settings for account

## **Sprint 7: Final Testing and Deployment**

**Duration:** 1 week

**Focus:** Testing and launch preparation

**Epics Covered:** All Epics

**User Stories:**

- End-to-end testing
- Performance/load testing
- User acceptance testing and deployment

**RESULT:**

Thus, the agile plan for the problem statement is completed successfully.

## EX NO: 5 USER STORIES – CREATION

### AIM:

To create User Stories for the given problem statement.

### THEORY:

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

User story template

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

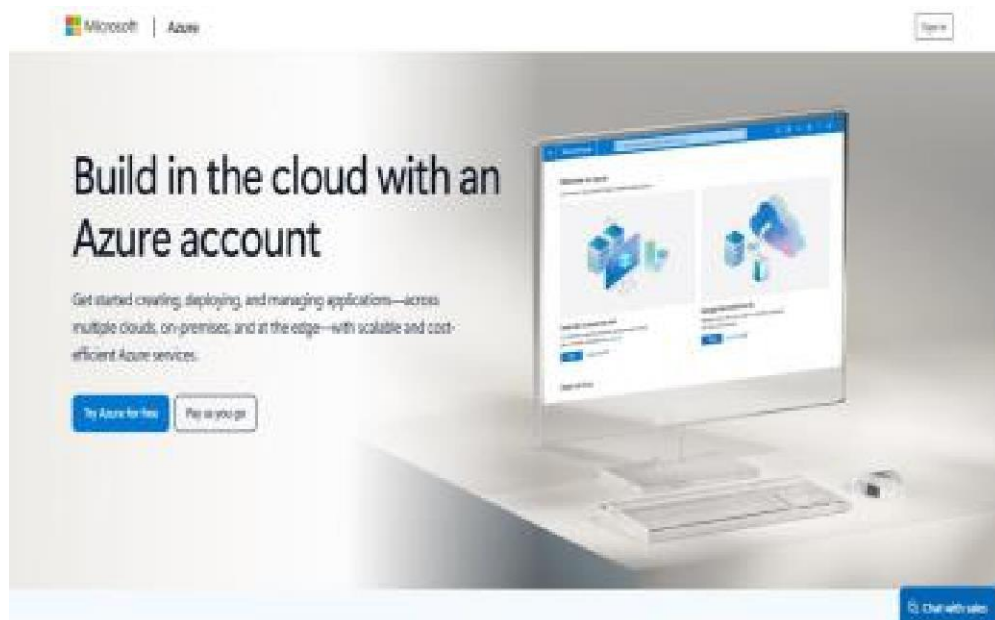
### PROCEDURE:

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

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

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

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

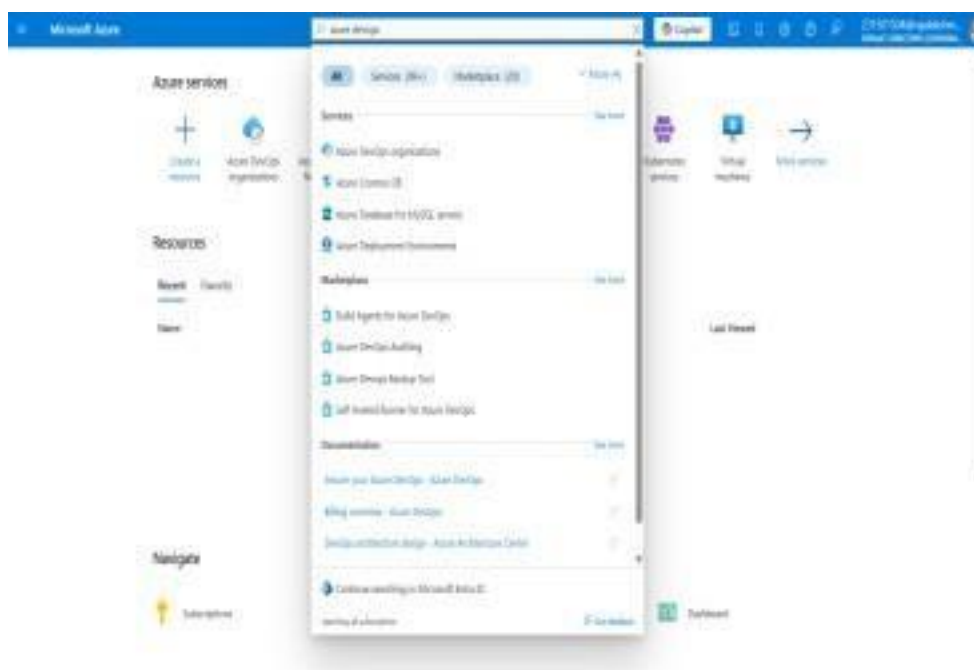




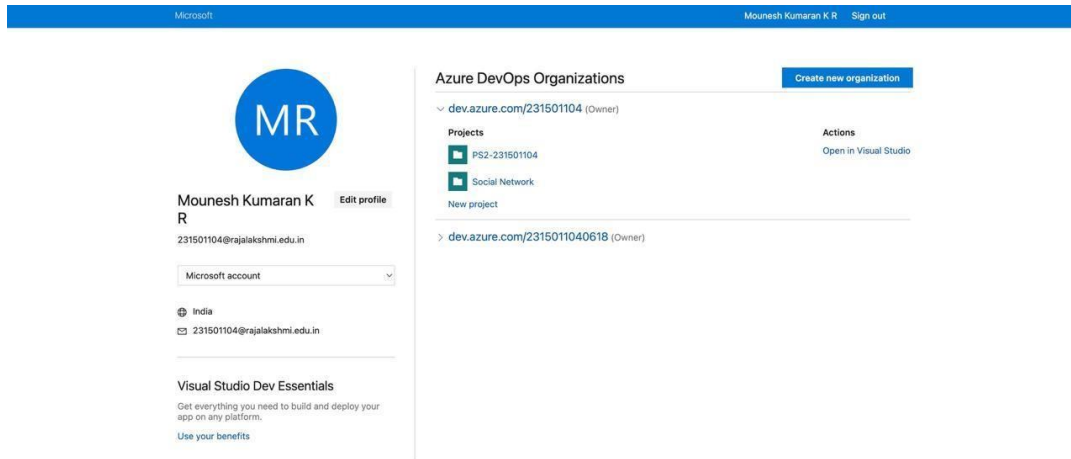
### 3. Azure home page



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



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



## 5. Create the First Project in Your Organization

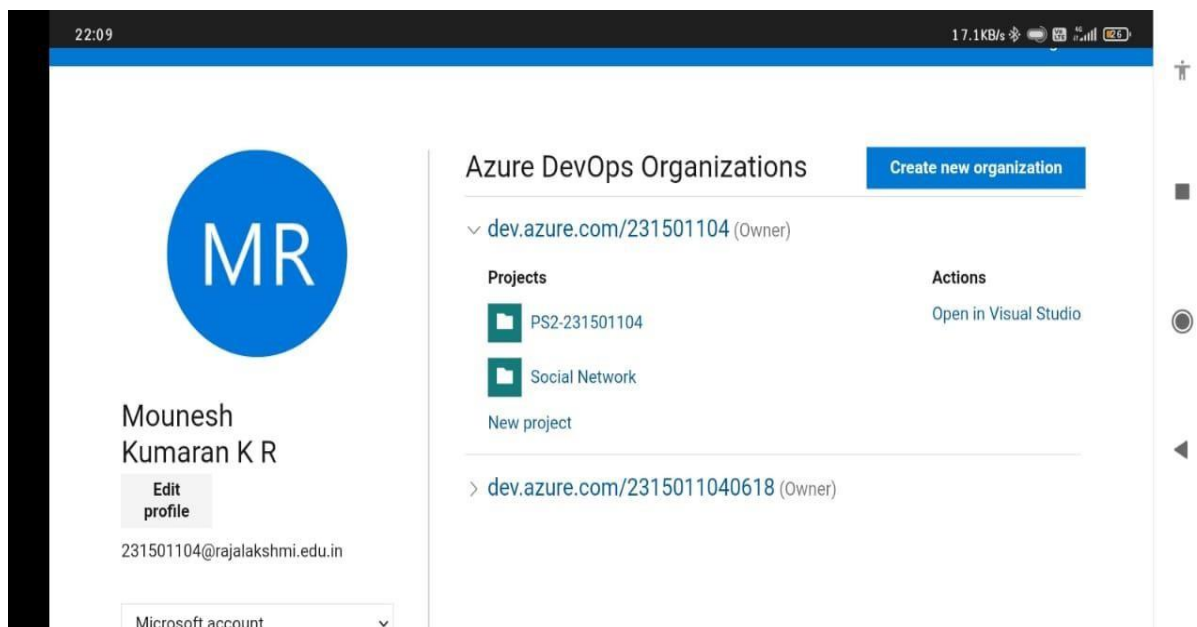
After the organization is set up, you'll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.

- 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**

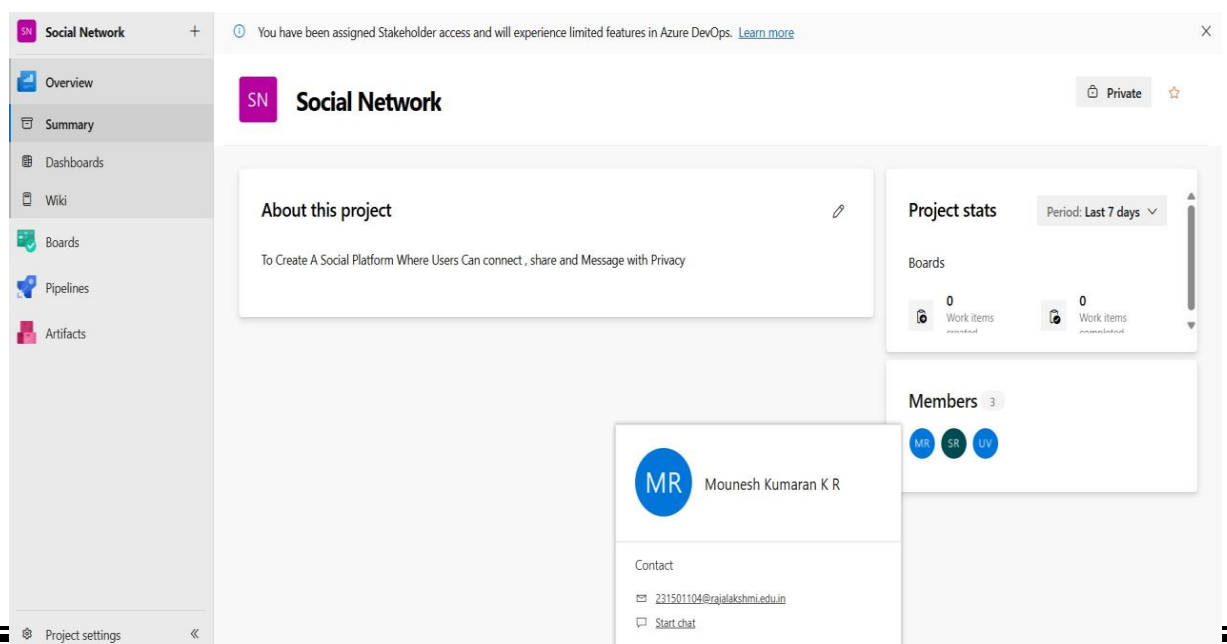
(accessible only to those invited) or **Public** (accessible to anyone).

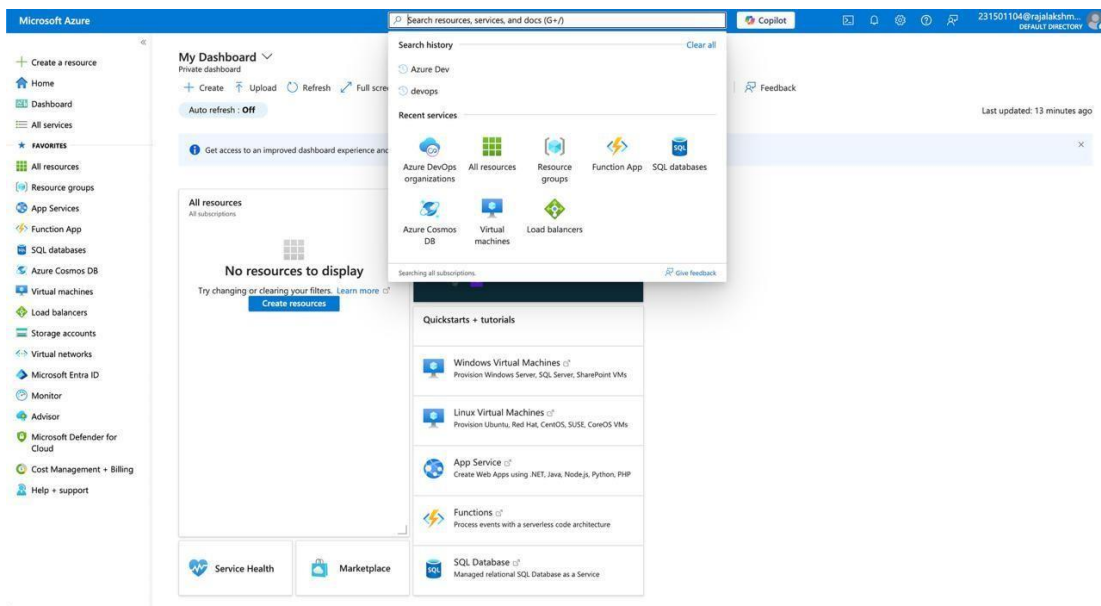
Once you've filled out the details, click **Create** to set up your first project

6. 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.



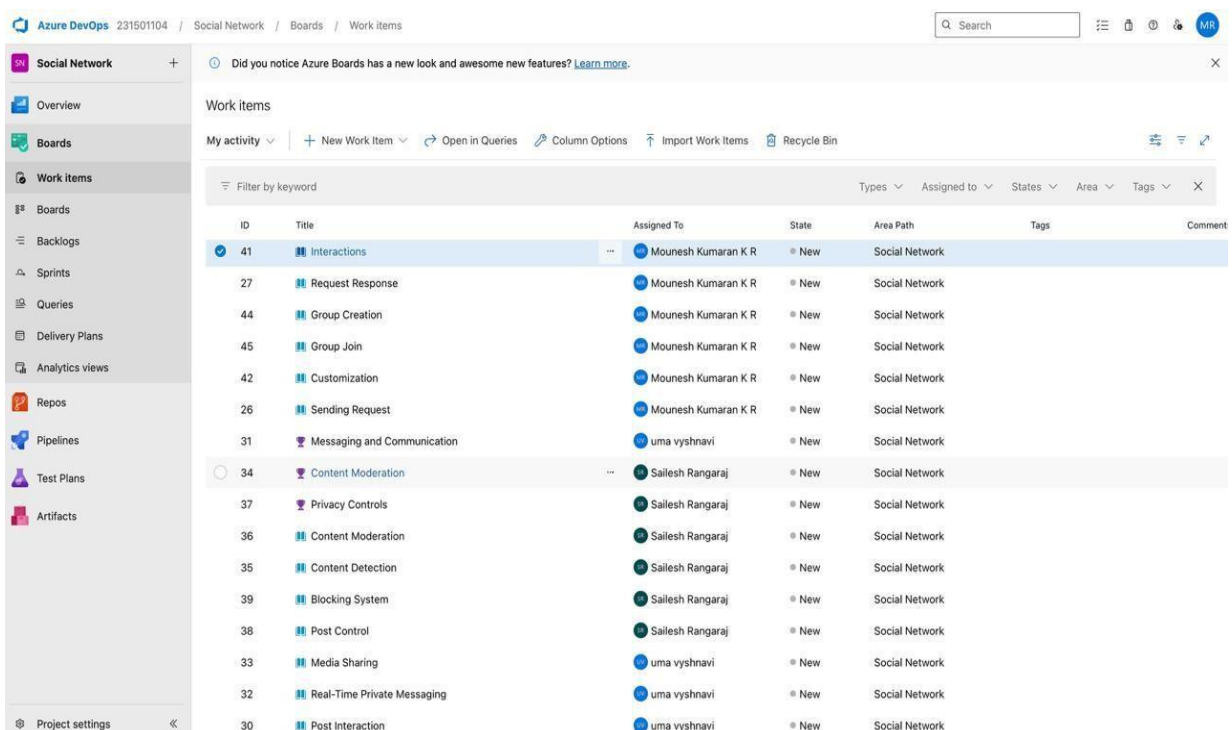
## 7. Project dashboard





## 9. To manage user stories

a. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards** page, where you can manage work items, backlogs, and sprints. b. On the **work items** page, you'll see the option to **Add a work item** at the top. Alternatively, you can find a + button or **Add New Work Item** depending on the view you're in. From the **Add a work item** dropdown, select **User Story**. This will open a form to enter details for the new User Story.



## 10. Fill in User Story Details

The screenshot displays the Azure DevOps interface for a project named 'Social Network'. The left sidebar shows the navigation menu with options like Overview, Boards, Work items, Backlogs, Sprints, Queries, Delivery Plans, Analytics views, Repos, Pipelines, Test Plans, and Artifacts. The main area shows a user story titled '44 Group Creation' by Mounesh Kumaran K R. The story is in the 'New' state and is associated with the 'Social Network' area and 'Social Network\Iteration 1' iteration. The description of the user story is: 'As a user, I want to create groups around interests or events, so that I can build communities with shared goals.' The acceptance criteria are: 'Group creation form includes name, description, privacy (public/private). Private groups require approval to join. Group admin dashboard available.' The planning section shows 'Story Points' as 2 and 'Risk' as 'Business'. The classification section shows 'Value area' as 'Business'. The deployment section has a note: '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.' The development section has a note: '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.' The related work section shows a parent work item: '43 Group and Community Creation' updated on Apr 14.

### Result:

The user story for the given problem statement was written successfully.

## EX NO: 6 SEQUENCE DIAGRAM

### AIM:

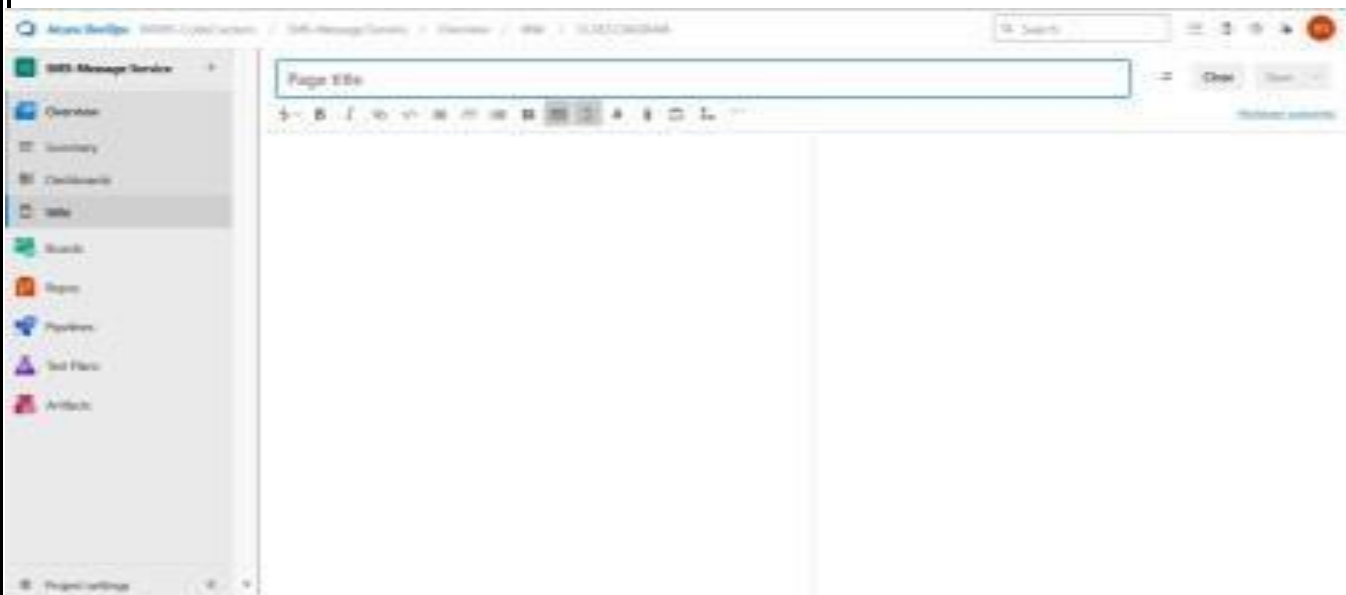
To design a Sequence Diagram by using Mermaid.js for the given problem statement.

### THEORY:

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

### PROCEDURE:

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



3. Write code for drawing sequence diagram and save the code.::: mermaid sequence

:::mermaid

sequenceDiagram

participant User

participant App

participant Login

participant Signup

participant UserProfile

participant UI

participant Post

participant Message

participant ContentModerator

participant Notification

participant Recipient

User->>App: Open App

App->>Login: Enter Email & Password

Login->>Signup: Request OTP Verification

Signup->>Login: generateOTP()

Login->>Signup: Submit OTP

Signup->>UserProfile: registerUser()

UserProfile->>Signup: Account Created

User->>Login: Enter Username & Password

Login->>Login: authenticate()

Login->>User: Success

User->>UI: Load Dashboard

User->>Post: Create Post

Post->>ContentModerator: add()

ContentModerator->>Post: scanPost(post)

Post->>ContentModerator: Post Approved

ContentModerator->>Notification: notify friends (new post)

Notification->>User: Post Published

User->>Message: Open Chat

Message->>ContentModerator: send()

ContentModerator->>Message: scanMessage()

Message->>ContentModerator: Message Clean

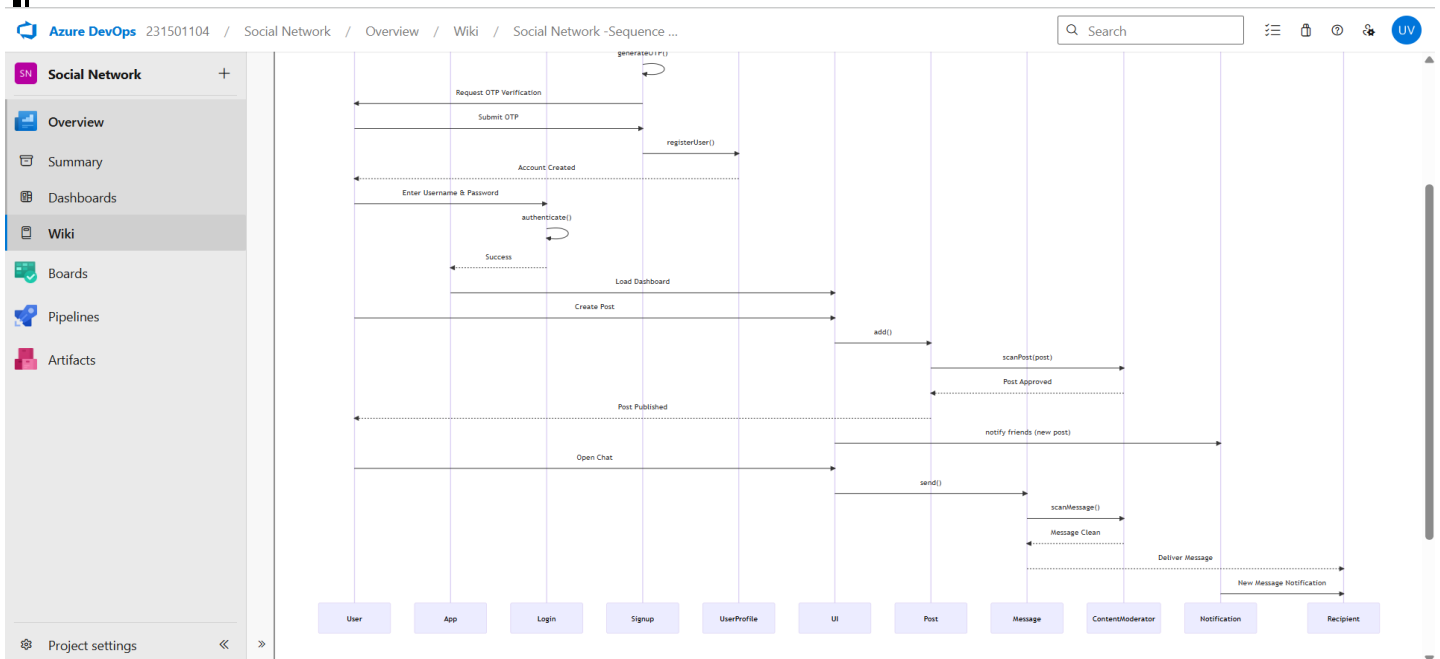
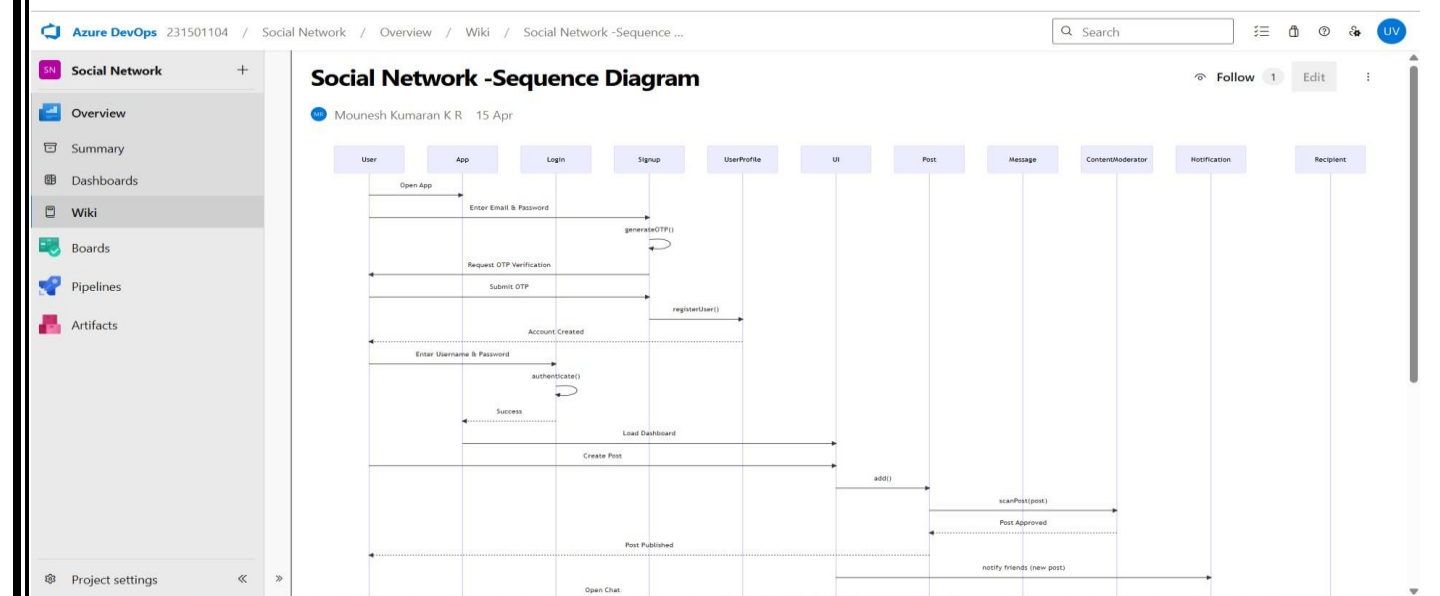
ContentModerator->>Notification: Deliver Message

Notification->>Recipient: New Message Notification

## **EXPLANATION:**

1. User opens the app and enters login details.
2. Signup service handles OTP generation and user registration.
3. After account creation, the user logs in and is authenticated.
4. Once logged in, the user can create a post which is sent to the content moderator.
5. After content is approved, friends are notified.
6. The user can also send messages which are scanned and then delivered to the recipient with a notification.

## 4. click wiki menu and select the page



## RESULT:

Thus, the sequence diagram for the given problem statement was drawn successfully.



## EX NO: 7 CLASS DIAGRAM

### AIM:

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

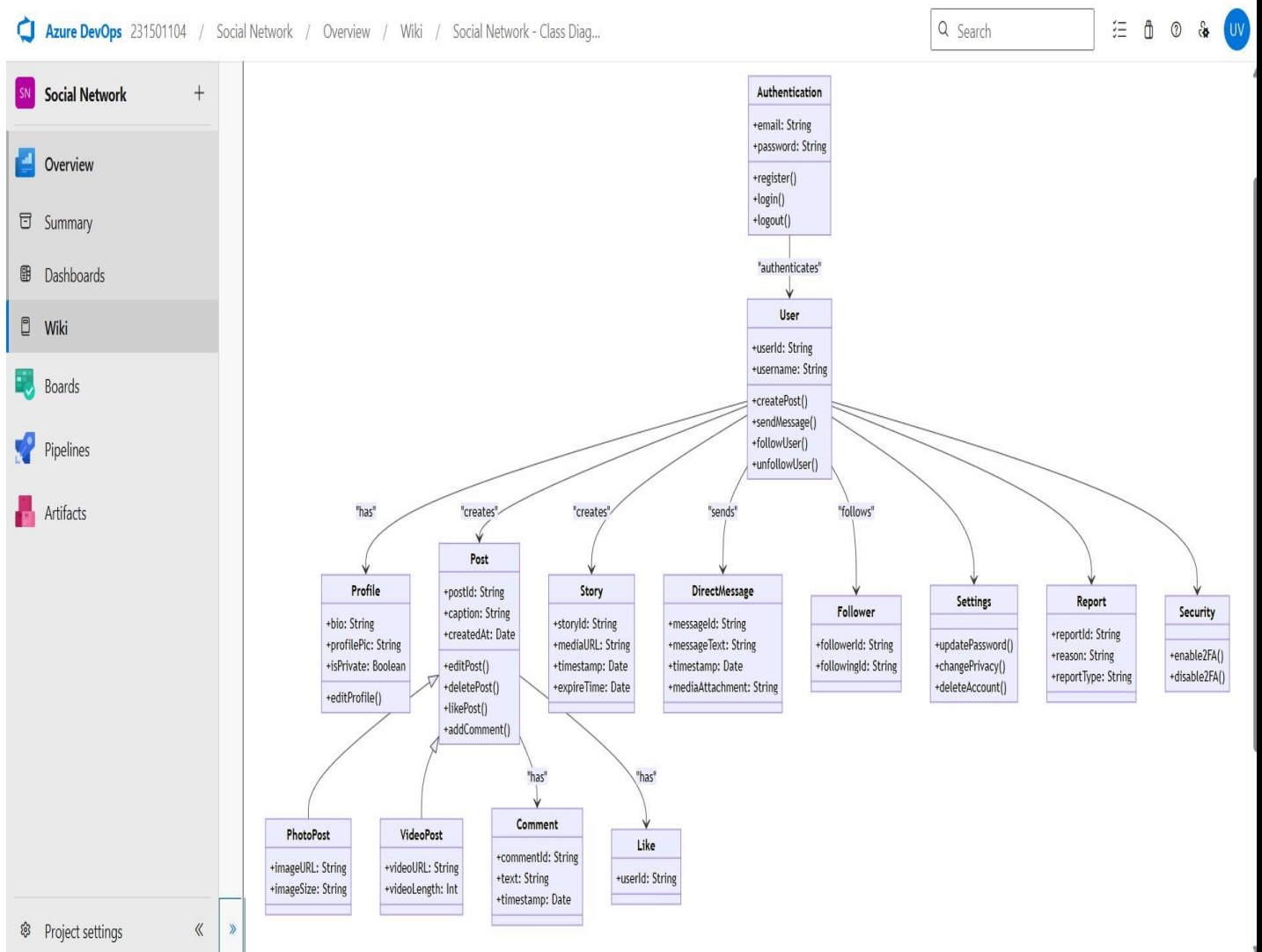
### THEORY:

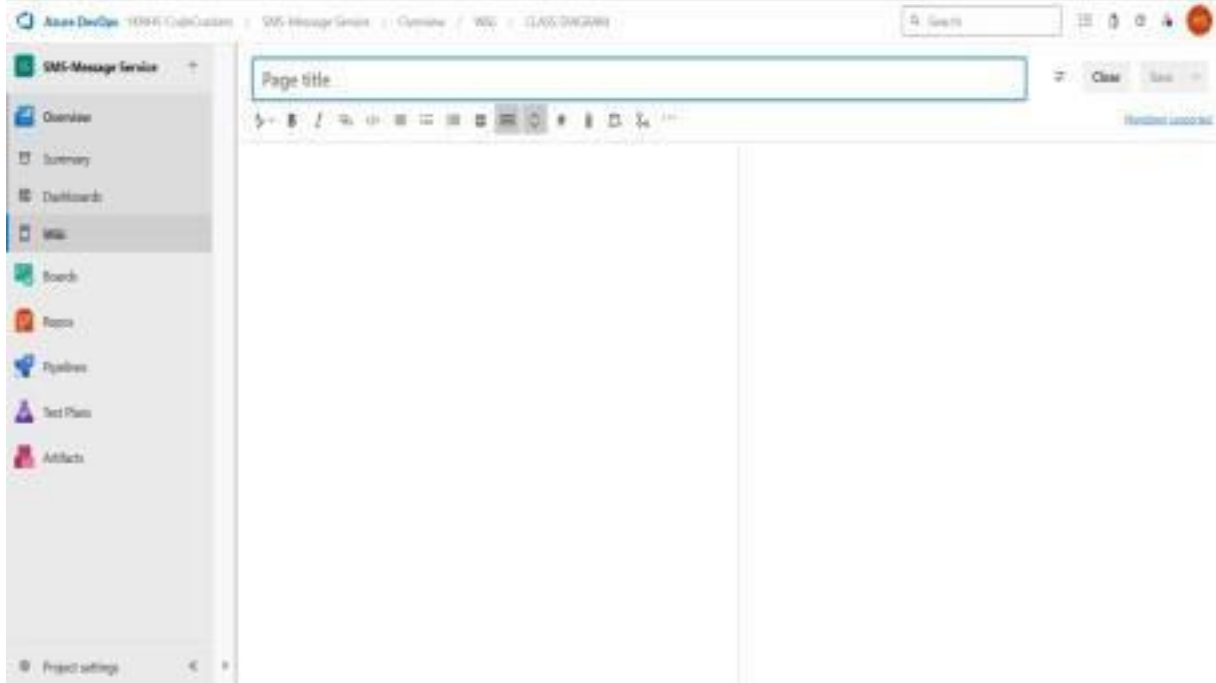
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.

### 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
```

```
%% =====
```

```
%%      CORE CLASSES
```

```
%% =====
```

```
classDiagram
```

```
class Authentication {  
    +email: String  
    +password: String  
    +register()  
    +login()  
    +logout()  
}
```

```
class User {  
    +userId: String  
    +username: String  
    +createPost()  
    +sendMessage()  
    +followUser()
```

```
+unfollowUser()
}

class Profile {
  +bio: String
  +profilePic: String
  +isPrivate: Boolean
  +editProfile()
}

class Post {
  +postId: String
  +caption: String
  +createdAt: Date
  +editPost()
  +deletePost()
  +likePost()
  +addComment()
}

class PhotoPost {
  +imageUrl: String
  +imageSize: String
}

class VideoPost {
  +videoURL: String
  +videoLength: Int
}

class Comment {
  +commentId: String
  +text: String
  +timestamp: Date
}

class Like {
  +userId: String
}

class Story {
  +storyId: String
  +mediaURL: String
  +timestamp: Date
  +expireTime: Date
}

class DirectMessage {
  +messageId: String
```

```
+messageText: String
+timestamp: Date
+mediaAttachment: String
}
```

```
class Follower {
  +followerId: String
  +followingId: String
}
```

```
class Settings {
  +updatePassword()
  +changePrivacy()
  +deleteAccount()
}
```

```
class Report {
  +reportId: String
  +reason: String
  +reportType: String
}
```

```
class Security {
  +enable2FA()
  +disable2FA()
}
```

Authentication --> User : authenticates

User --> Profile : has

User --> Post : creates

User --> Story : creates

User --> DirectMessage : sends

User --> Follower : follows

User --> Settings

User --> Report

User --> Security

Post --> Comment : has

Post --> Like : has

Post --> PhotoPost

Post --> VideoPost

Profile --> PhotoPost

**RESULT:**

Thus, the class diagram for the given problem statement was designed successfully.

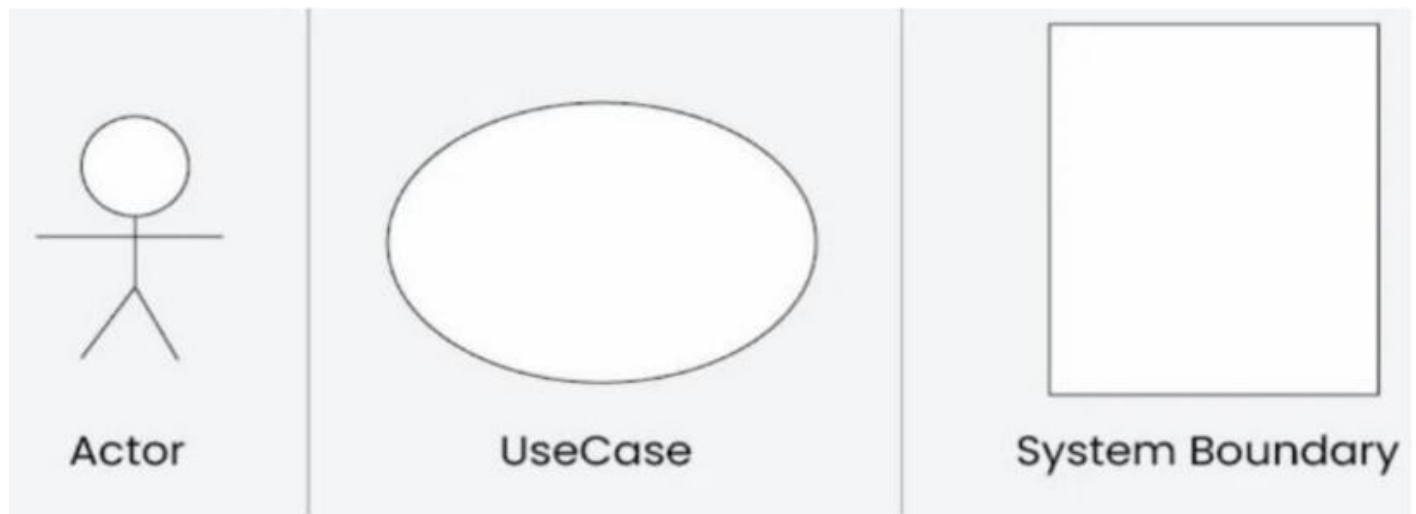
## EX NO: 8 USECASE DIAGRAM

### 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 Diagram.

**RESULT:**

The use case diagram for the given problem statement was designed successfully.



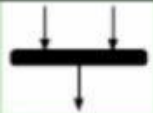


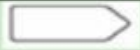


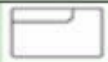


## EX NO: 9 ACTIVITY DIAGRAM

### 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 the Azure Wiki

### RESULT:

Thus, the Activity diagram for the above problem statement done successfully.



## EX NO: 10 ARCHITECTURE DIAGRAM

### AIM:

Steps to draw the Architecture Diagram using draw.io.

### THEORY:

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

#### ARCHITECTURE SYMBOL OVERVIEW



### PROCEDURE:

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

### RESULT:

Thus, the architecture diagram for the given problem statement was designed successfully.

## EX NO: 11 USER INTERFACE

### AIM:

Design User Interface for the given project.

**SocialHub**

**Sign Up**

[Already have an account? Login](#)

**SocialHub**

**Login**

[Don't have an account? Sign up](#)



Had a great day



Like



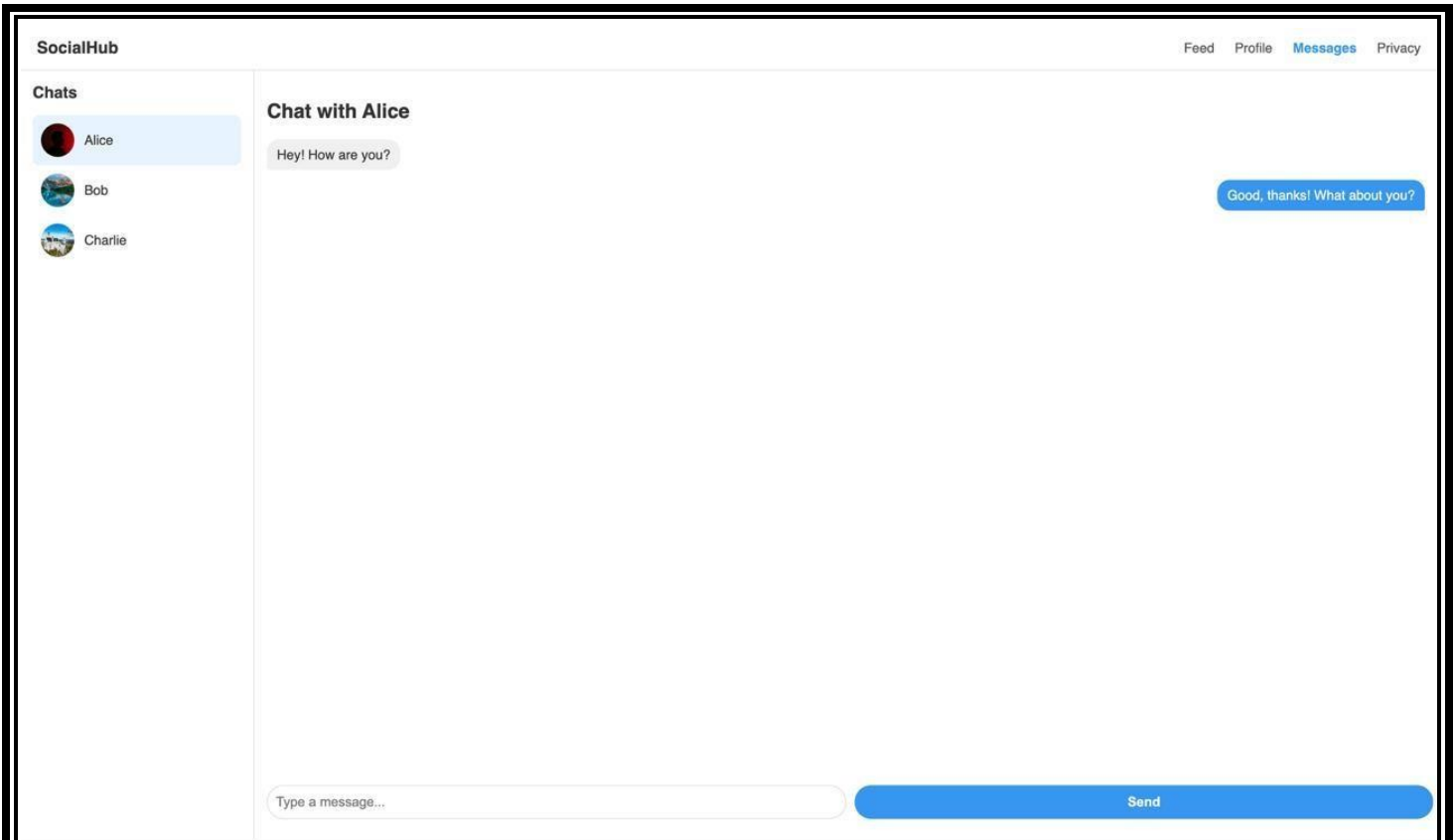
Comment



@demouser

Life is beautiful. Share the joy 🥰





## RESULT:

Thus, the UI for the given problem statement is completed successfully.

## EX NO: 12 IMPLEMENTATIONS

### 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 cd`
- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.).
- Commit & push: `git add . git commit -m "Initial commit" git push`

#### origin main Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)

- Navigate to Pipelines → Click "New Pipeline".
- Select Git Repository (Azure Repos, GitHub, or Bitbucket).
- Choose Starter Pipeline or a pre-configured template for your framework.
- Modify the `azure-pipelines.yml` file (Example for a Node.js app):

trigger:

- main

pool:

vmImage: 'ubuntu-latest'

steps:

task: UseNode@1

inputs:

version: '16.x'

-script: npm install

displayName: 'Install dependencies'

-script: npm run build

displayName: 'Build application'

-task: PublishBuildArtifacts@1

inputs:

pathToPublish: 'dist'

artifactName: 'drop'

Click "Save and Run" → The pipeline will start building app.

Step 4: Set Up Release Pipeline (CD - Continuous

Deployment) • Go to Releases → Click "New Release

Pipeline".

- Select Azure App Service or Virtual Machines (VMs) for

deployment. • Add an artifact (from the build pipeline).

- Configure deployment stages (Dev, QA, Production).

- Click "Deploy" to push your web app to Azure.

**RESULT:**

Thus, the implementation of the given problem statement is done successfully.