

**EX NO: 1**

**Date: 21/01/2025**

## **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

### **Result:**

The study was successfully completed.

EX NO: 2

Date: 28/01/2025

## PROBLEM STATEMENT

### AIM:

To prepare PROBLEM STATEMENT for your given project.

### Problem Statement:

The current process for evaluating candidate features, design changes, and performance improvements for our weather application lacks a centralized, data-driven approach. This results in:

**Subjective decision-making:** Feature prioritization and implementation often rely on opinions and intuition rather than quantifiable impact on user engagement, satisfaction, and application performance.

**Inefficient A/B testing analysis:** Analyzing the results of A/B tests and other experiments is time-consuming and prone to manual errors, hindering our ability to quickly identify winning variations and iterate effectively.

**Limited understanding of user impact:** We lack a comprehensive understanding of how different application aspects affect key user metrics such as daily active users, session duration, feature adoption, and churn rate.

**Difficulty in identifying optimization opportunities:** Pinpointing specific areas within the application that could benefit from performance enhancements or design refinements is challenging without robust analytical tools.

**Lack of personalized recommendations:** The application currently offers a one-size-fits-all experience, and we lack the ability to analyze user behavior patterns to deliver tailored recommendations for features or settings that could enhance individual user value and retention.

### Proposed Solution:

To address these challenges, we propose the development of a "Candidate Performance Analysis & Recommendation Tool." This tool will provide a unified platform to:

**Define and track key performance indicators (KPIs)** relevant to user engagement, satisfaction, and application performance.

**Centralize data from various sources**, including A/B testing platforms, analytics dashboards, user feedback channels, and performance monitoring tools.

**Automate the analysis of experimental data**, providing clear insights into the statistical significance and practical impact of different variations.

**Visualize user behavior and performance metrics** through interactive dashboards and reports, enabling a deeper understanding of user interactions and identifying areas for improvement.

**Generate data-driven recommendations** for feature prioritization, design optimizations, and performance enhancements based on their predicted impact on key metrics.

**Facilitate personalized recommendations** for users based on their past behavior and preferences, aiming to increase feature discovery, user engagement, and overall satisfaction.

**Goal:**

The primary goal of this tool is to empower the product development team with actionable insights to make data-informed decisions, leading to a more engaging, performant, and user-centric weather application, ultimately driving increased user satisfaction and retention.

**Key Features (Anticipated):**

**Experiment Tracking and Analysis:** Ability to log and analyze A/B tests, multivariate tests, and other experiments.

**KPI Definition and Monitoring:** Interface to define, track, and visualize key performance indicators.

**User Segmentation and Cohort Analysis:** Tools to segment users based on various criteria and analyse their behaviour over time.

**Impact Prediction and Prioritization:** Algorithms to predict the potential impact of proposed changes on key metrics and prioritize them accordingly.

**Personalized Recommendation Engine:** System to generate tailored recommendations for users based on their individual usage patterns.

**Performance Monitoring Integration:** Connection to performance monitoring tools to correlate application performance with user behaviour.

**User Feedback Integration:** Ability to incorporate user feedback data into the analysis.

**Reporting and Visualization:** Customizable dashboards and reports to communicate insights effectively.

This problem statement provides a comprehensive overview of the challenges we face and the proposed solution with the "Candidate Performance Analysis & Recommendation Tool" for our weather application. It sets the stage for further discussions and the development of this valuable.

**Result:**

The problem statement was written successfully.

**EX NO: 3**

**DATE: 04/02/2025**

## **AGILE PLAN**

**Aim:**

To prepare an agile Plan for weather application.

### **AGILE PLANNING**

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

### **Agile Plan: Weather App**

#### **I. Product Vision:**

To deliver a reliable, accurate, and personalized weather experience that empowers users in their daily planning.

#### **II. Core Principles:**

User-first approach, open communication, embracing change, constant learning, and teamwork.

#### **III. Team Structure:**

Dedicated, cross-functional team: Product Owner (prioritizing features), Scrum Master (facilitating process), UI/UX Designers, Front-end, Back-end, and Mobile (iOS/Android) Developers, and QA Testers.

#### **IV. Agile Approach:**

Primarily Scrum with two-week Sprints for focused development. Standard Scrum ceremonies: Sprint Planning, daily 15-minute stand-ups, Sprint Reviews (showcase & feedback), and Sprint Retrospectives (improvement).

#### **V. Initial Product Focus (Minimum Viable Product - MVP):**

- Display current temperature and weather conditions (user's current location).
- Show daily high and low temperatures (current location).
- Enable users to search for weather information for different locations.
- Provide a 5-day weather forecast for a selected location.

#### **VI. Iterative Releases:**

Launch an MVP with core features, followed by incremental updates with new functionalities and enhancements based on user feedback and backlog priorities. Aim for frequent deployments, ideally with CI/CD.

#### **VII. Communication & Collaboration:**

Daily team sync-ups, Sprint Reviews with stakeholders for feedback, regular progress updates, and utilizing shared communication tools. Transparent Product Backlog.

## **VIII. Measuring Success:**

Track Sprint Velocity (team output), Burndown Charts (sprint progress), Cycle Time (story completion), User Satisfaction (app store ratings, feedback), Key Usage Metrics (DAU/MAU, session duration, feature adoption), and App Performance (crash rates, load times).

## **IX. Continuous Improvement:**

Regular Sprint Retrospectives to identify process improvements and adapt the plan based on user feedback and data insights.

## **Result:**

Thus the Agile plan was completed successfully.

**EX NO: 4**

**Date: 18/02/2025**

## **CREATE USER STORIES**

### **Aim:**

To create User Stories for

### **THEORY**

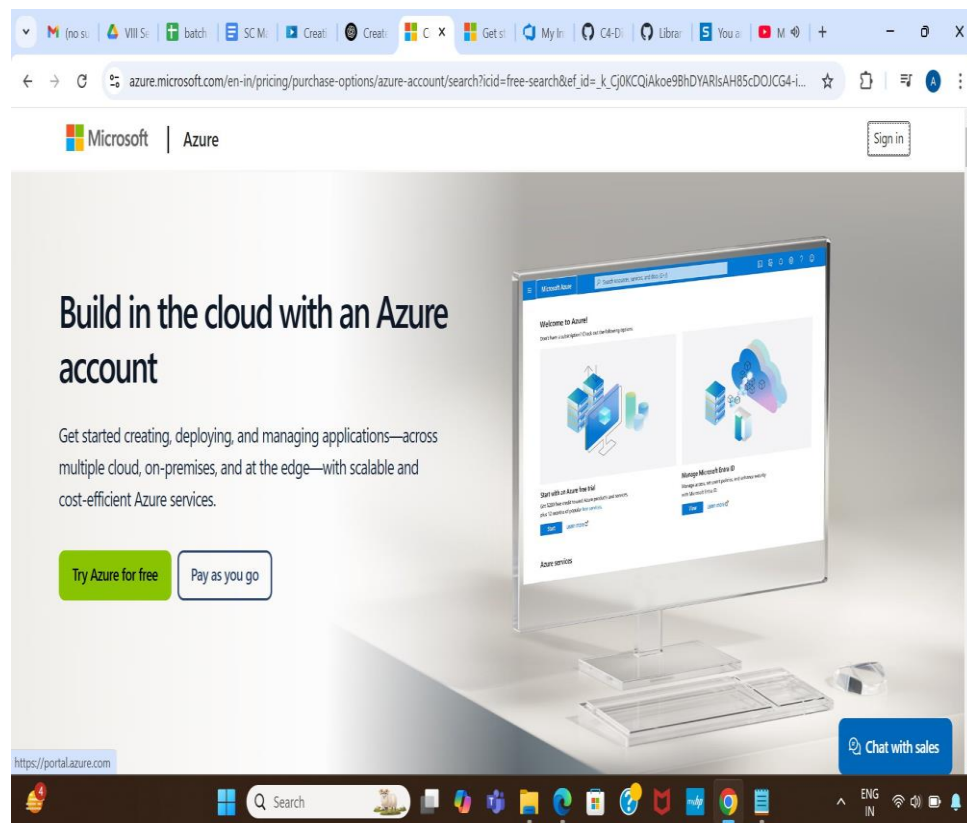
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

### **User story template**

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

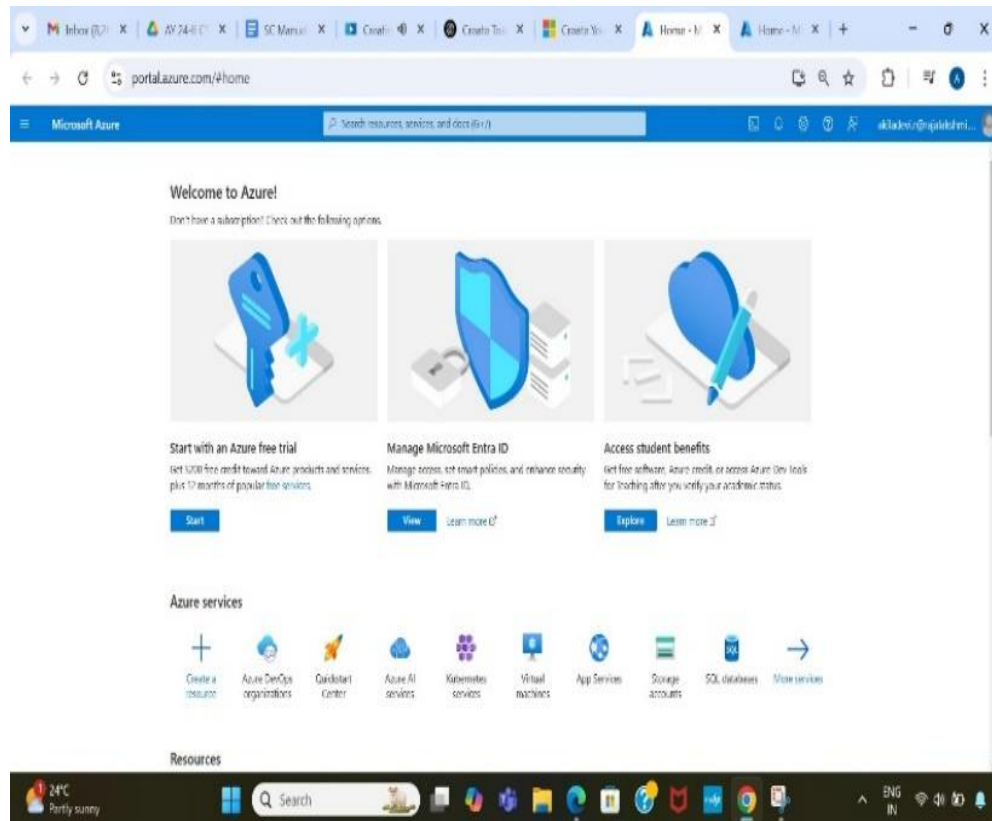
### **Procedure:**

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

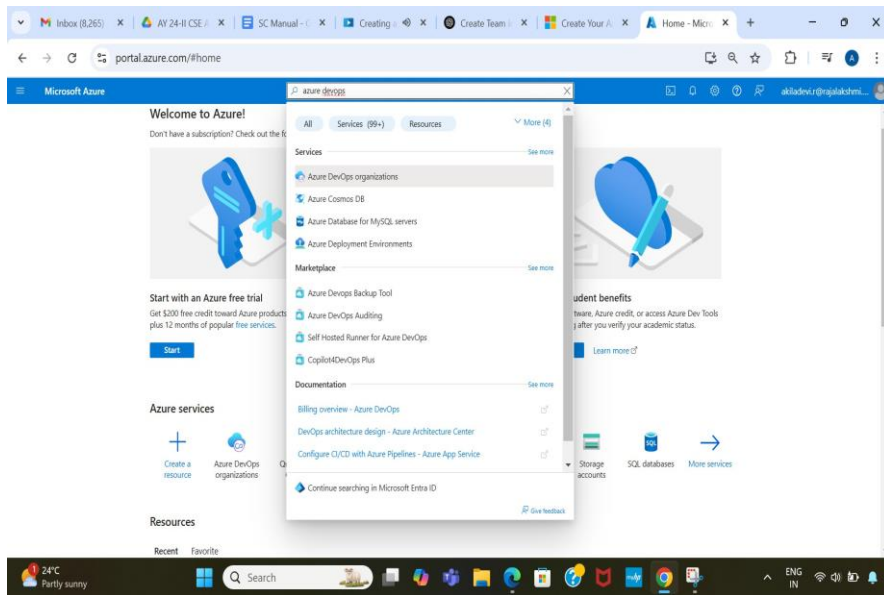




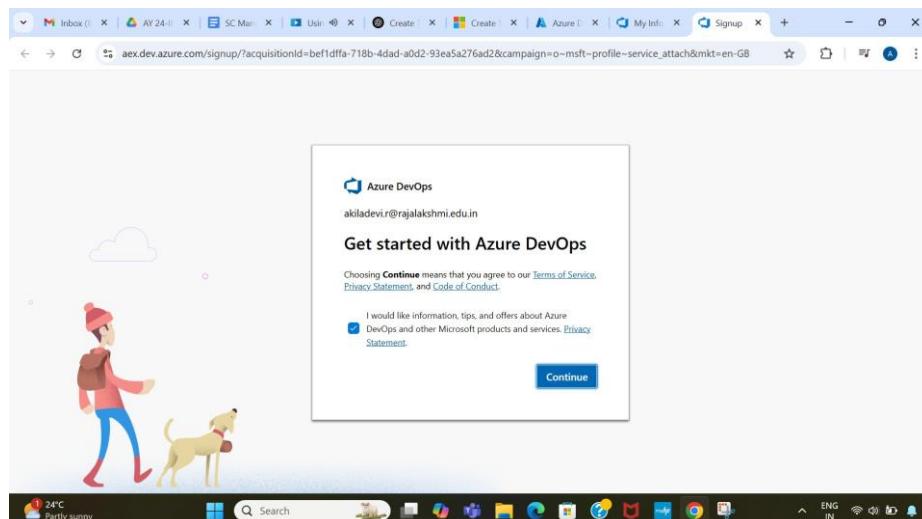
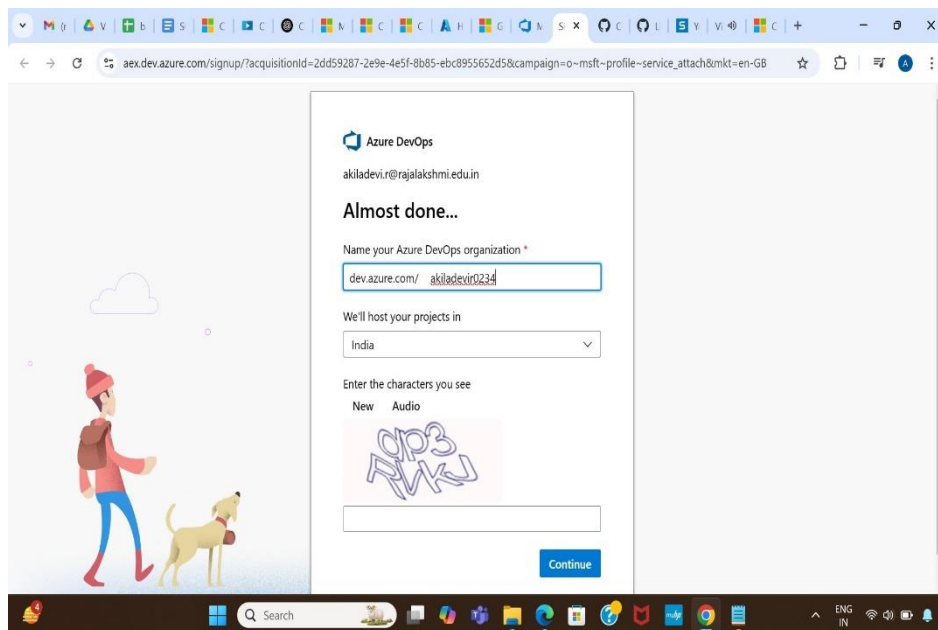
### 3. Azure home page



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



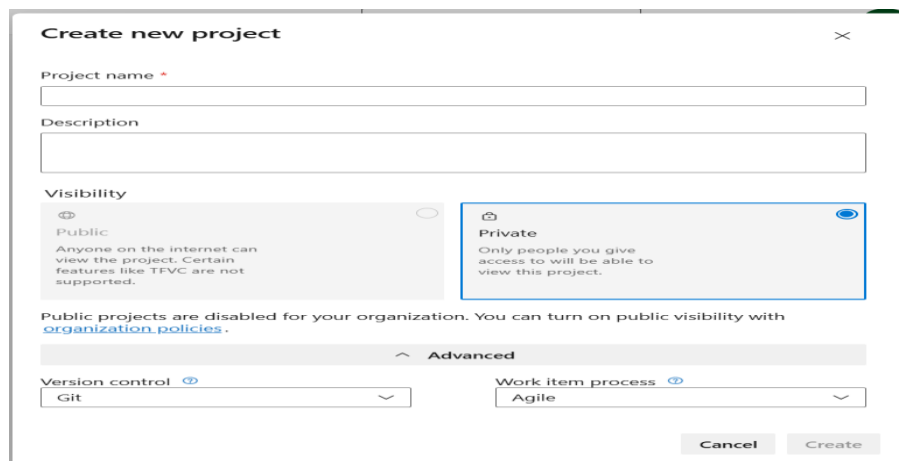
5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.



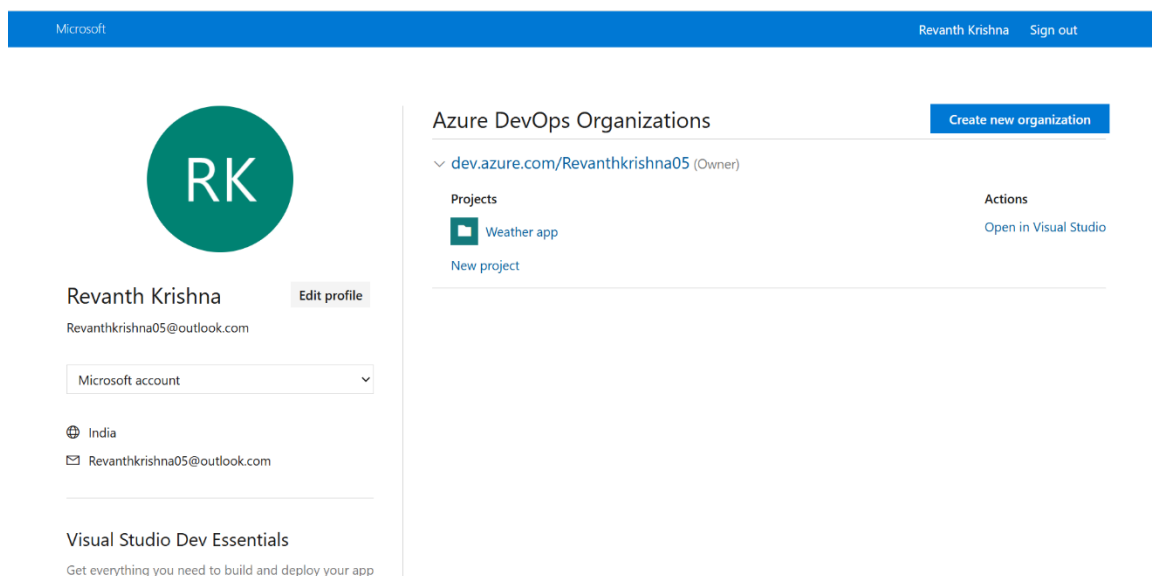
## 6. Create the First Project in Your Organization

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

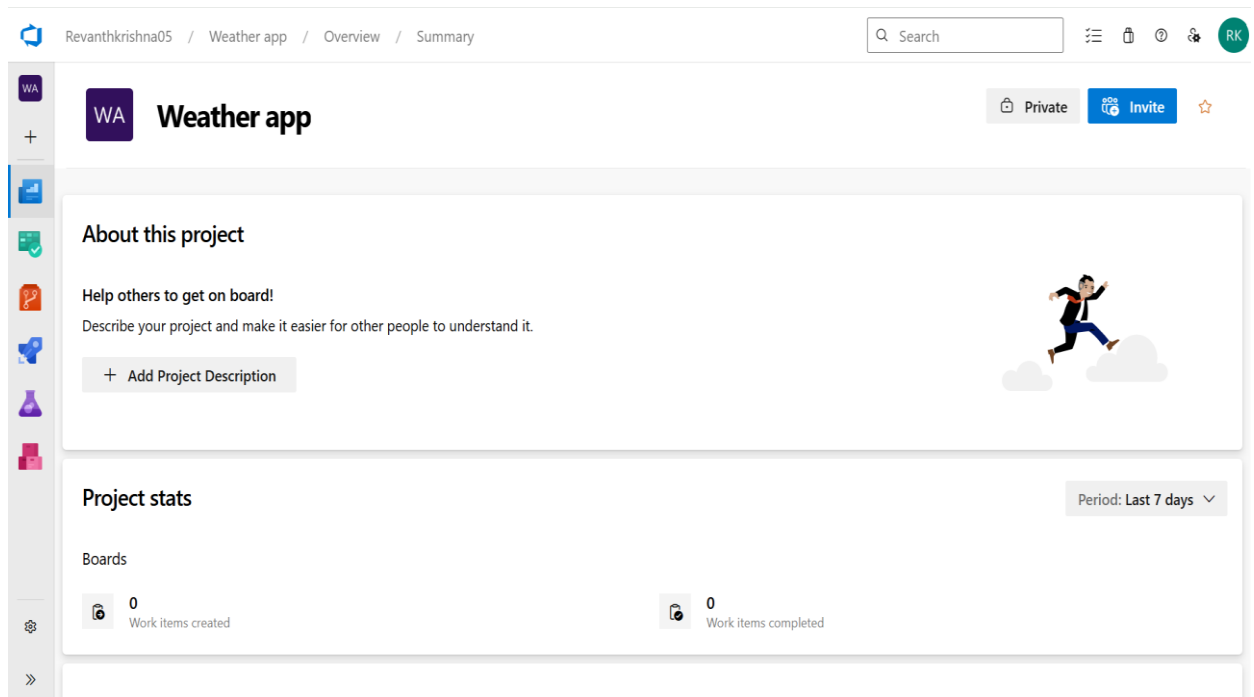
- i. On the organization's **Home page**, click on the **New Project** button.
- ii. Enter the project name, description, and visibility options:
  - o **Name:** Choose a name for the project (e.g., **LMS**).
  - o **Description:** Optionally, add a description to provide more context about the project.
  - 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.

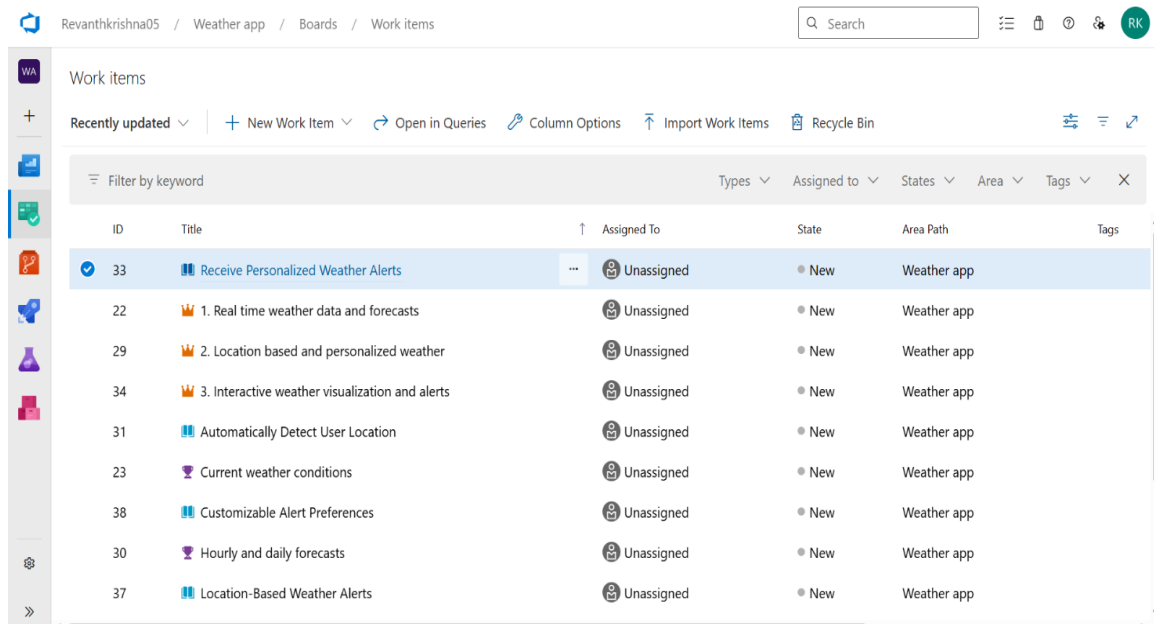


## 8. Project dashboard



## 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 'View Current weather' user story in Azure DevOps. The form is titled '26 View Current weather' and is located under the 'Weather app' area. The status is 'New'. The description is 'As a user I want to see the current temperature and weather conditions so that i can decide what to wear'. The acceptance criteria are 'Weather data (temperature, humidity, wind speed) is fetched and displayed.', 'Data updates every X seconds/minutes without manual refresh.', and 'Displayed values are clear and formatted correctly.' The planning section shows 'Story Points' as 2 and 'Risk' as 'Low'. The classification section shows 'Value area' as 'Business'. The deployment section shows a link to 'Link an Azure Repos commit, pull'. The development section shows a link to 'Link an Azure Repos commit, pull'.

Revanthkrishna05 / Weather app / Boards / Work items

Search

Recently updated | Back to Work Items

14 of 14

USER STORY 26

26 View Current weather

No one selected | 0 Comments | Add Tag

Save | Follow | Details | 1 | 0

Updated by Revanth Krishna: Saturday

State: New | Area: Weather app | Reason: New | Iteration: Weather app

**Description**

As a user I want to see the current temperature and weather conditions so that i can decide what to wear

**Acceptance Criteria**

Weather data (temperature, humidity, wind speed) is fetched and displayed.  
Data updates every X seconds/minutes without manual refresh.  
Displayed values are clear and formatted correctly.

**Planning**

Story Points  
Priority  
2  
Risk

**Classification**

Value area  
Business

**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](#)

### Result:

The user story was written successfully.

**EX NO : 5**

**Date:**

**25/02/2025**

## **SEQUENCE DIAGRAM**

### **Aim:**

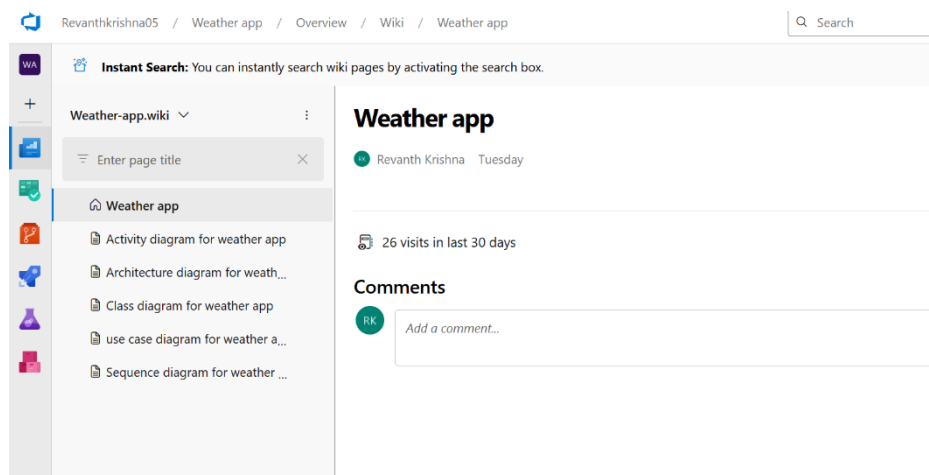
To design a Sequence Diagram by using Mermaid.js

### **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
```

```
sequenceDiagram
```

```
participant U as User
```

```
participant W as Web App
```

```
participant APIM as Azure API Management
```

```
participant F as Azure Function
```

```
participant API as External Weather API
```

```
participant DB as Azure Database
```

```

U->>W: Open app
W->>APIM: Request weather data
APIM->>F: Query weather data
F->>API: Fetch from External Weather API
API-->>F: Return weather data
F->>DB: Save/cache data (optional)
F-->>APIM: Weather data
APIM-->>W: Weather data
W-->>U: Display weather data
W->>DB: Save search history/preferences
...

```

## Explanation:

participant defines the entities involved.

->> represents a direct message.

-->> represents a response message.

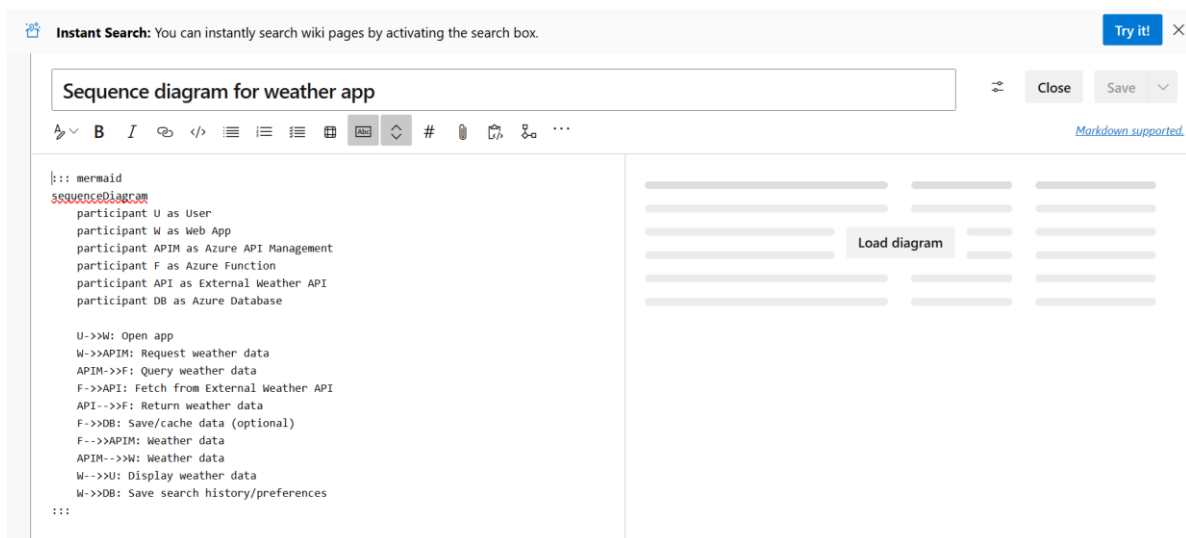
+ after ->> activates a participant.

- after -->> deactivates a participant. alt / else

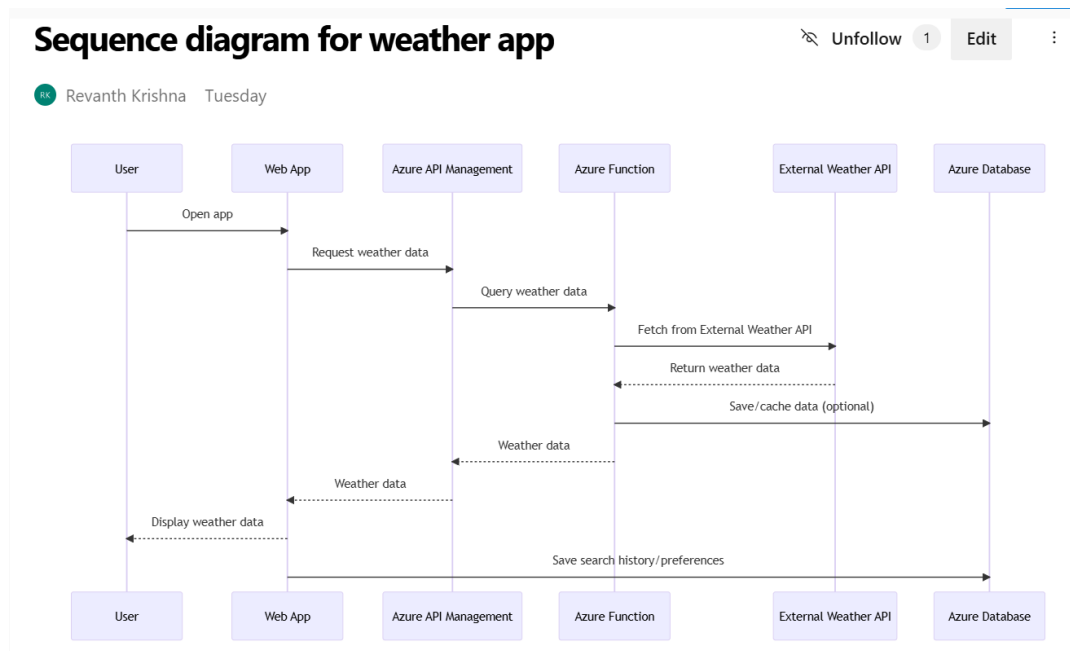
for conditional flows.

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**

**Date: 04/03/2025**

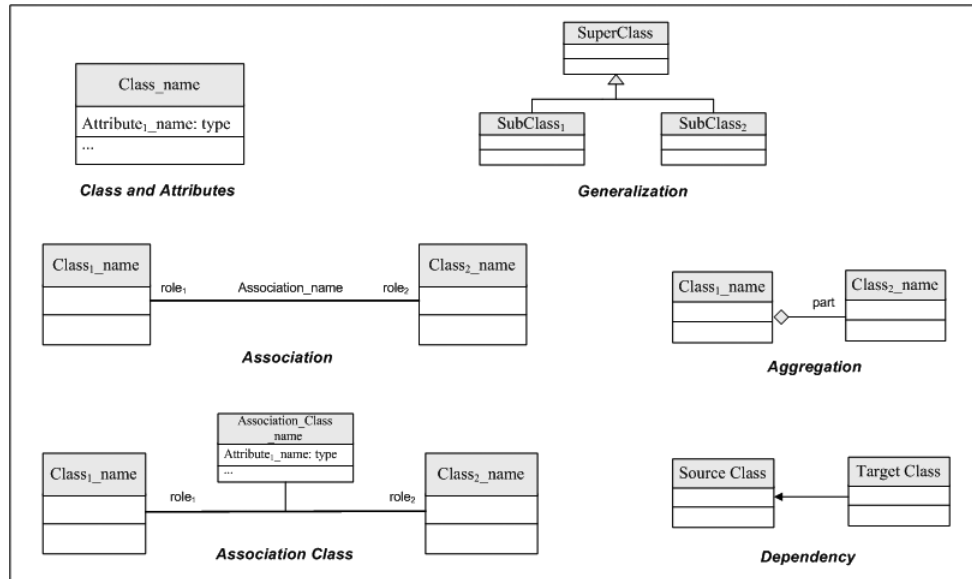
## **CLASS DIAGRAM**

**AIM :-**

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

### **THEORY**

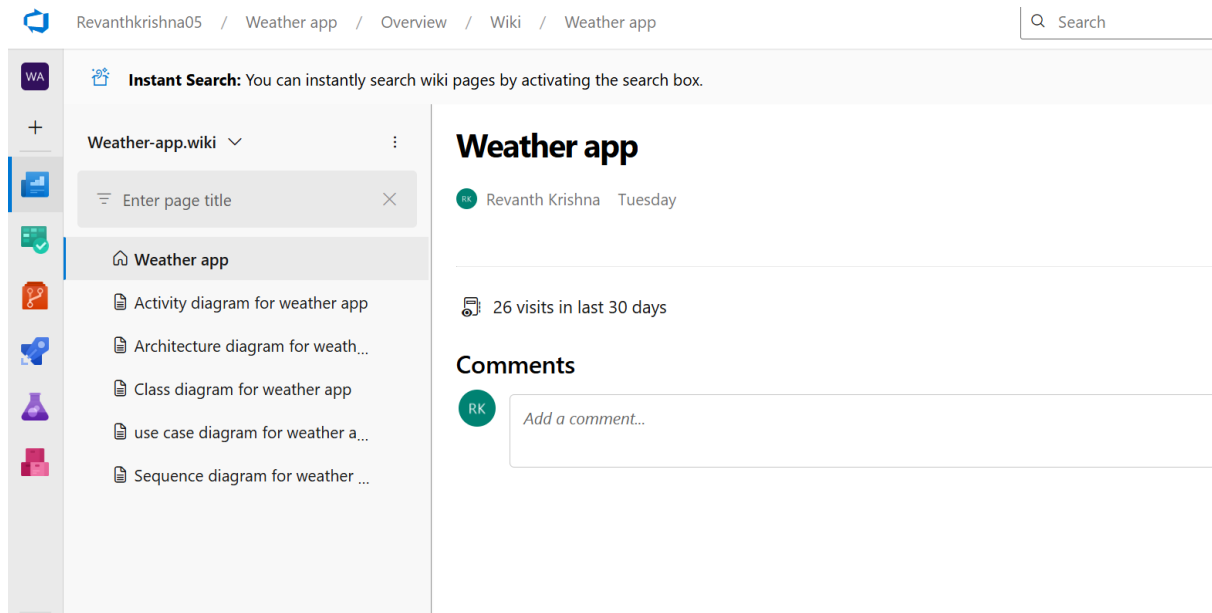
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



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 WeatherController {
    +showWeather(location)
    +showForecast(location)
}

class WeatherService {
    +getCurrentWeather(location)
    +getForecast(location)
}

class WeatherData {
    +temperature
    +humidity
    +condition
    +timestamp
    +getSummary()
}

class LocationService {
    +getUserLocation()
    +validateLocation(input)
}

class User {
    +userId
```

```

+username
+email
+getPreferredLocation()
}

```

```

class Pipeline {
+build()
+test()
+deploy()
}

```

```

class Repo {
+commitChanges()
+clone()
}

```

```

class Boards {
+trackProgress()
+addTask(task)
}

```

```

WeatherController --> WeatherService
WeatherController --> LocationService
WeatherService --> WeatherData
User --> LocationService
Pipeline --> Repo
Repo --> Boards
:::

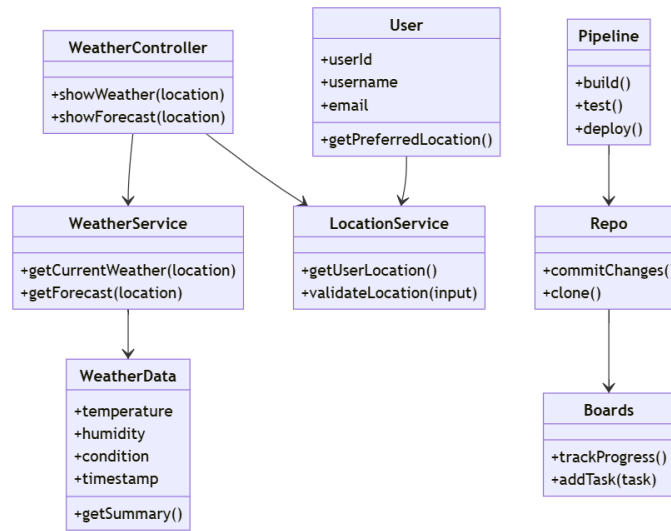
```

## Relationship Types

	Type	Description
<		Inheritance
\*		Composition
o		Aggregation
>		Association
<		Association
▷		Realization

## Class diagram for weather app

Revanth Krishna Tuesday



### Result:

The use case diagram was designed successfully.

**EX NO: 7**  
**Date: 11/03/2025**

## **USECASE DIAGRAM**

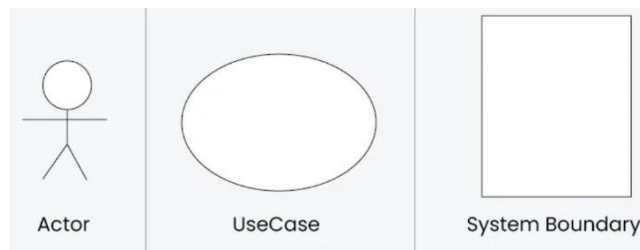
### **Aim:**

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.

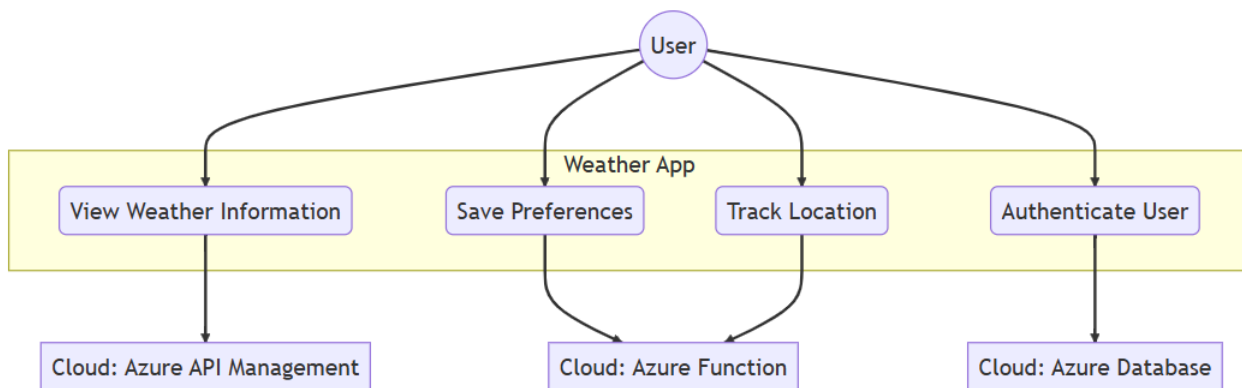
- Use Markdown to embed the diagram:
- ![Use Case Diagram](attachments/use\_case\_diagram.png)

#### 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

## use case diagram for weather app

 Revanth Krishna Tuesday



### Result:

The use case diagram was designed successfully

## EX NO. 8

Date: 18/03/2025



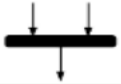


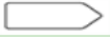





### ACTIVITY DIAGRAM

#### AIM :-

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

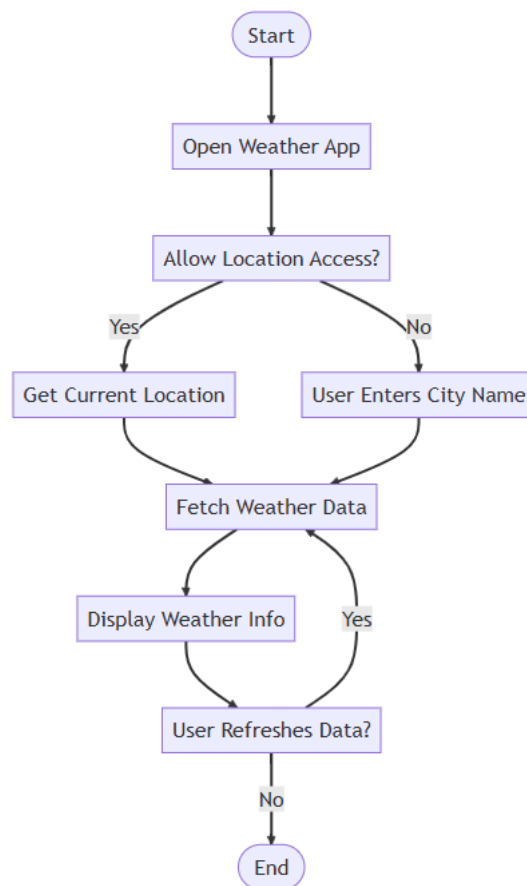
#### THEORY

Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

Notations	Symbol	Meaning
Start		Shows the beginning of a process
Connector		Shows the directional flow, or control flow, of the activity
Joint symbol		Combines two concurrent activities and re-introduces them to a flow where one activity occurs at a time
Decision		Represents a decision
Note		Allows the diagram creators to communicate additional messages
Send signal		Show that a signal is being sent to a receiving activity
Receive signal		Demonstrates the acceptance of an event
Flow final symbol		Represents the end of a specific process flow
Option loop		Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate		Represents a transition that invokes the last active state.
End		Marks the end state of an activity and represents the completion of all flows of a process

#### Procedure

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



**Result:**

The activity diagram was designed successfully



## EX NO. 9

Date: 25/03/2025

### 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

## Architecture diagram for weather app

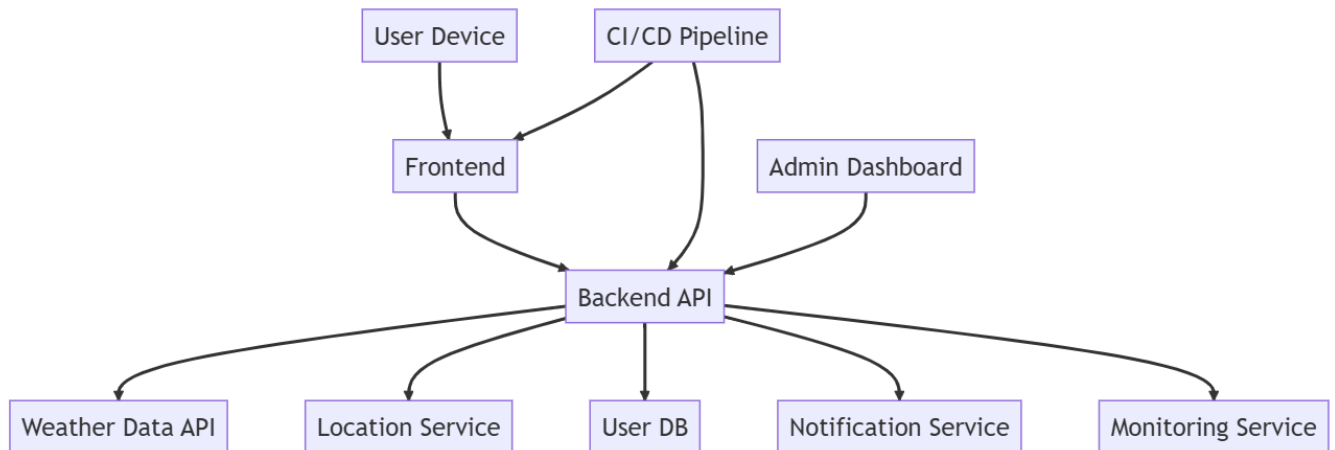
Unfollow

1

Edit

:

RK Revanth Krishna Tuesday



### Result:

The architecture diagram was designed successfully

**EX NO. 10**

**Date:**

**01/04/2025**

## **USER INTERFACE**

**Aim:**

Design User Interface for the given project



**Result:**

The UI was designed successfully.

## **IMPLEMENTATION**

**Aim:**

To implement the given project "Weather App" based on Agile Methodology.

**Procedure:**

Step 1: Set Up an Azure DevOps Project

Log in to Azure DevOps.

Click "New Project" → Enter project name (Weather App) → Click "Create".

Inside the project, navigate to "Repos" to store the source 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:

bash

Copy

Edit

git clone <repo\_url>

cd <repo\_folder>

Add web application code:

(HTML, CSS, JavaScript files for Weather App)

Commit and push:

bash

Copy

Edit

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 a Starter Pipeline or a pre-configured template.

Modify the azure-pipelines.yml file (Example for Node.js/JavaScript app):

yaml

Copy

Edit

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 the application.

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 Weather App application was successfully implemented and deployed on Azure using Agile Methodology.

## **TESTING – TEST PLANS AND TEST CASES**

### **Aim:**

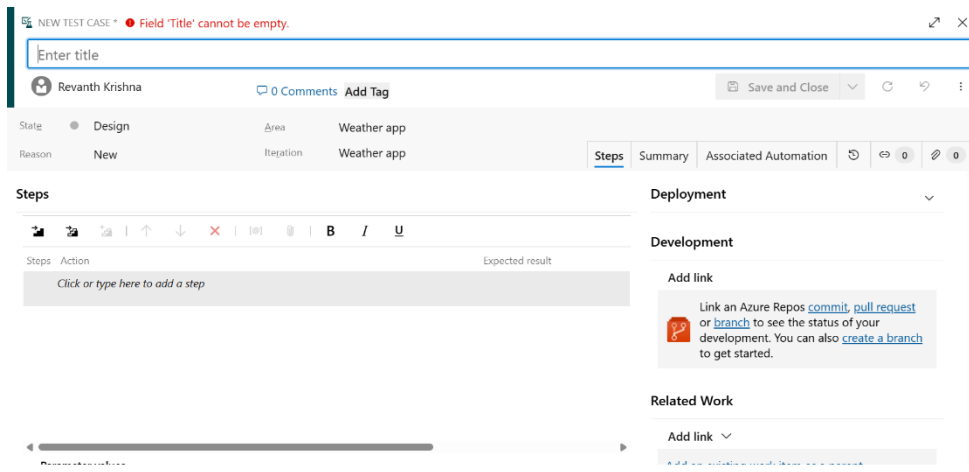
Test Plans and Test Case and write two test cases for at least five user stories showcasing the happy path and error scenarios in azure DevOps platform.

Test Planning and Test Case

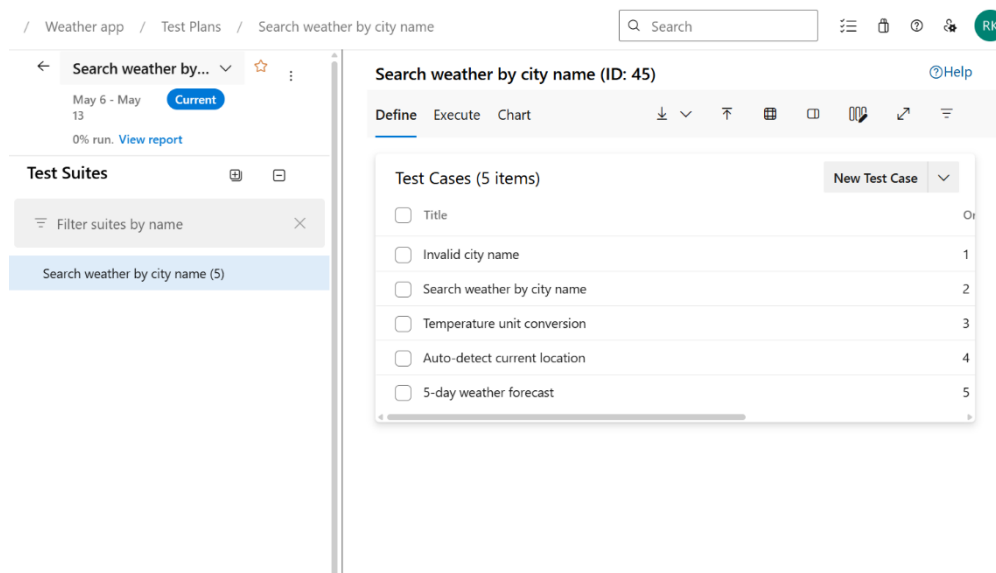
### **Test Case Design Procedure**

1. Understand Core Features of the Application
  - User Signup & Login
  - Viewing and Managing Playlists
  - Fetching Real-time Metadata
  - Editing playlists (rename, recorder, record)
  - Creating smart audio playlists based on categories (mood, genre, artist, etc.)
2. Define User Interactions
  - Each test case simulates a real user behaviour (e.g., logging in, renaming a playlist, adding a song)
3. Design Happy Path Test Cases
  - Focused on validating that all features function as expected under normal conditions.
  - Example: User logs in successfully, adds item to playlist, or creates a category-based playlist.
4. Design Error Path Test Cases
  - Simulate negative or unexpected scenarios to test robustness and error handling.
  - Example: Login fails with invalid credentials, save fails when offline, no recommendations found.
5. Break Down Steps and Expected Results
  - Each test case contains step-by-step actions and a corresponding expected outcome.
  - Ensures clarity for both testers and automation scripts.
6. Use Clear Naming and IDs
  - Test cases are named clearly (e.g., TC01 – Successful Login, TC10 – Save Playlist Fails).
  - Helps in quick identification and linking to user stories or features.
7. Separate Test Suites
  - Grouped test cases based on functionality (e.g., Login, Playlist Editing, Recommendation System).
  - Improves organization and test execution flow in Azure DevOps.
8. Prioritize and Review
  - Critical user actions are marked high-priority.
  - Reviewed for completeness and traceability against feature requirements.

## 1.New test plan



## 2.Test suite



## 3.Test case

Give two test cases for at least five user stories showcasing the happy path and error scenarios in azure DevOps platform.

Weather app Creator Test Plans

Okay, let's connect Weather App User Stories to Test Cases. This will demonstrate how user needs are translated into specific tests to ensure the app functions correctly.

### Understanding the Connection

- **User Stories:** These are short, simple descriptions of a feature told from the perspective of the user. They help define the "what" of the system.
- **Test Cases:** These are detailed steps that validate whether the software meets the requirements outlined in the user stories. They define the "how" to test the "what."

**Example: Weather App**

Let's create some user stories and corresponding test cases for a basic weather application.

### **User Stories**

1. **As a user, I want to be able to search for the weather in a specific city so that I can plan my day.**
2. **As a user, I want to see the current temperature in my searched city so that I know what to wear.**
3. **As a user, I want to see a 5-day forecast so that I can plan for the week.**
4. **As a user, I want to see an icon representing the weather condition (e.g., sunny, cloudy, rainy) so that I can quickly understand the weather.**
5. **As a user, I want to be notified if there is a severe weather alert for my searched city so that I can take necessary precautions.**

### **Test Cases**

Here's how these user stories can be translated into test cases. Note that each user story can have multiple test cases to cover different scenarios (happy path and error paths).

#### **User Story 1: Search for Weather in a Specific City**

- **TC1.1 - Valid City Search (Happy Path)**
  - Action:
    - 1. Open the Weather App.
      2. Enter a valid city name (e.g., "London").
      3. Click the "Search" button.
  - Expected Result:
    1. The weather information for London is displayed.
- **TC1.2 - Invalid City Search (Error Path)**
  - Action:
    1. Open the Weather App.
    2. Enter an invalid city name (e.g., "asdfghjkl").
    3. Click the "Search" button.
  - Expected Result:
    1. An error message is displayed (e.g., "City not found").
- **TC1.3 - Empty City Search (Error Path)**
  - Action:
    1. Open the Weather App.
    2. Leave the city search field empty.
    3. Click the "Search" button.
  - Expected Result:
    1. An error message is displayed (e.g., "Please enter a city").



### **User Story 2: See Current Temperature**

- **TC2.1 - Temperature Display (Happy Path)**
  - Action:
    1. Search for a valid city (e.g., "New York").
  - Expected Result:
    1. The current temperature for New York is displayed in the correct unit (e.g., Celsius or Fahrenheit).
- **TC2.2 - Temperature After City Change**
  - Action:
    1. Search for city "A".
    2. Search for city "B".
  - Expected Result:
    1. The temperature displayed updates correctly for both city "A" and then city "B".

### **User Story 3: See 5-Day Forecast**

- **TC3.1 - 5-Day Forecast Display (Happy Path)**
  - Action:
    1. Search for a valid city.
    2. Navigate to the "5-Day Forecast" section.
  - Expected Result:
    1. The forecast for the next 5 days is displayed, including date, temperature (high/low), and weather condition.
- **TC3.2 - 5-Day Forecast Data Availability**
  - Action:
    1. Search for a valid city.
    2. Navigate to the "5-Day Forecast" section.
  - Expected Result:
    1. Data is available for all 5 days.

### **User Story 4: Weather Condition Icons**

- **TC4.1 - Correct Icon Display (Happy Path)**
  - Action:
    1. Search for a city with sunny weather.
  - Expected Result:
    1. A sun icon is displayed.
- **TC4.2 - Icon Change with Condition**
  - Action:
    1. Search for a city.
    2. Observe the icon.
  - Expected Result:

1. The icon accurately reflects the current weather condition (e.g., changes from sun to clouds).

### User Story 5: Severe Weather Alerts

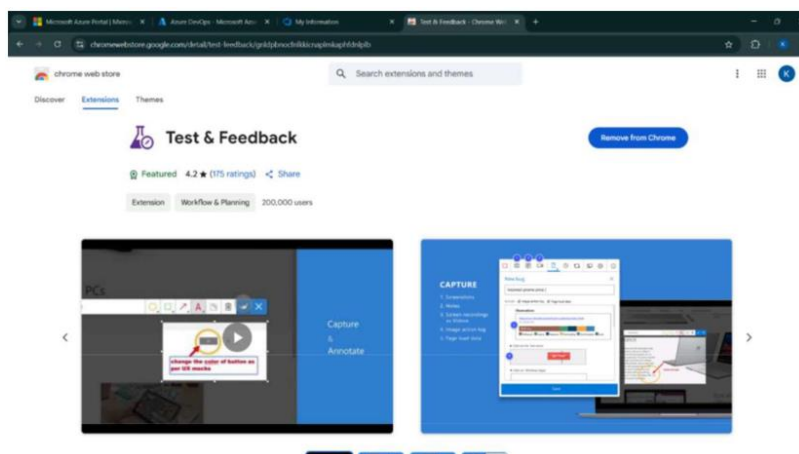
- **TC5.1 - Alert Display (Happy Path - Alert Present)**
  - Action:
    1. Search for a city with a severe weather alert (e.g., "Tornado Warning").
  - Expected Result:
    1. A prominent alert message is displayed to the user.
- **TC5.2 - No Alert Display (Happy Path - No Alert)**
  - Action:
    1. Search for a city with no severe weather alert.
  - Expected Result:
    1. No alert message is displayed.
- **TC5.3 - Alert Dismissal**
  - Action:
    1. Search for a city with a severe weather alert.
    2. Dismiss the alert.
  - Expected Result:
    1. The alert is dismissed, and the user can continue using the app.

### Key Considerations

- **Test Data:** Use a variety of test data (valid, invalid, boundary values) to ensure thorough testing.
- **Test Environment:** Test in different environments (e.g., different devices, operating systems, network conditions).
- **Automation:** Automate test cases where possible to improve efficiency.
- **Prioritization:** Prioritize test cases based on risk and impact.

By connecting user stories to detailed test cases, you can ensure that your weather application meets user needs and functions reliably

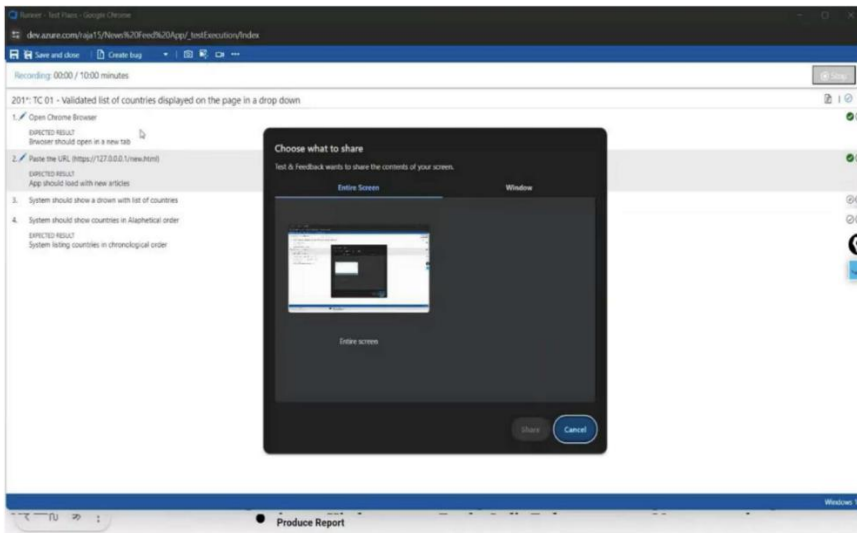
### 4.Installation of test



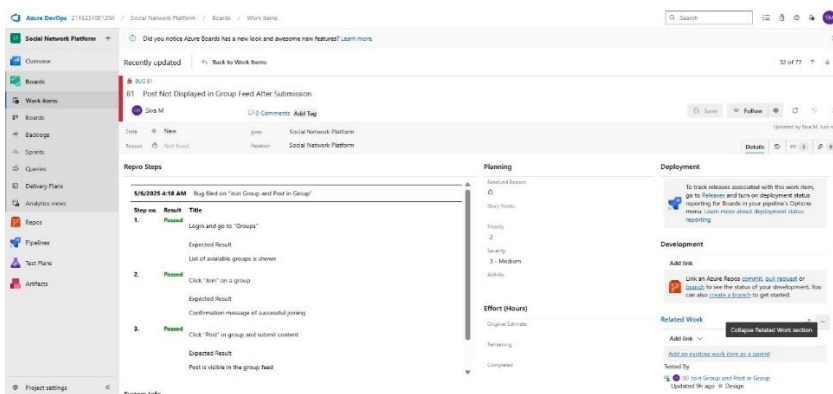
Test and feedback

Showing it as an extension

# 1. Recording the Test Cases



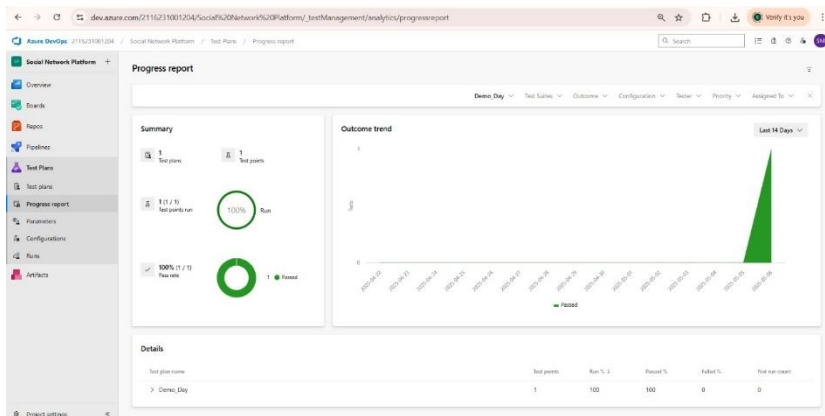
# 6. Creating Bugs



# 7. Test Case Results



## 8. Progress Report



### LOAD TESTING PROCEDURE :

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

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

##### 1. Sign in to Azure Portal

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

##### 2. Create the Resource

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

##### 3. Fill in the Configuration Details

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

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

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

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

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

Once your resource is ready:

##### 1. Go to your Azure Load Testing resource and click Add HTTP requests > Create.

##### 2. Basics Tab

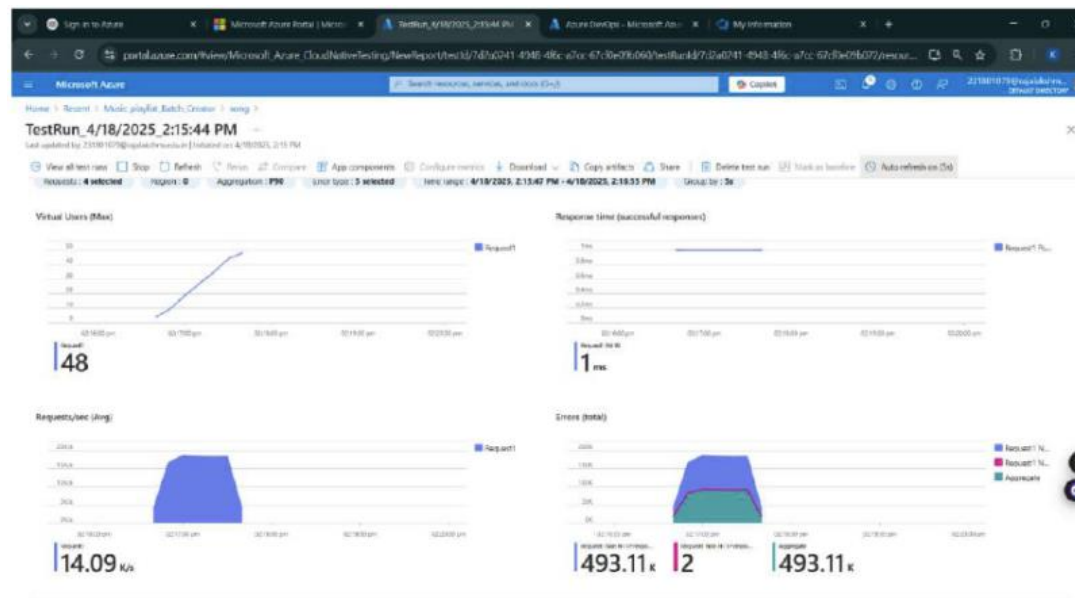
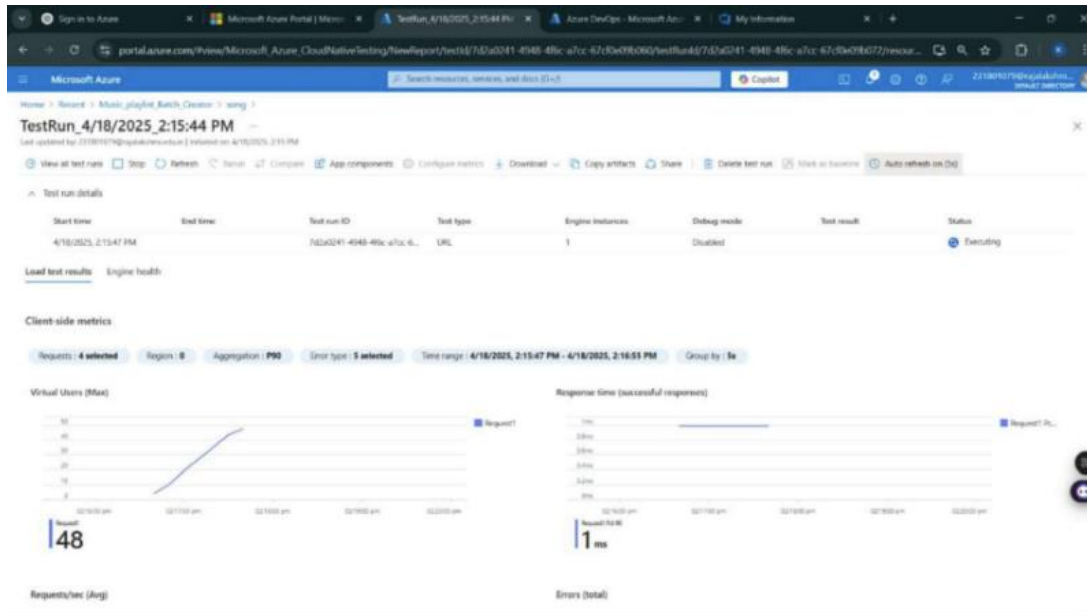
- Test Name: Provide a unique name.
- Description: (Optional) Add test purpose.
- Run After Creation: Keep checked.

##### 3. Load Settings

- Test URL: Enter the target endpoint (e.g., <https://yourapi.com/products>).

##### 4. Click Review + Create — Create to start the test.

## Load Testing



## RESULT:

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

