RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM – 602 105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025

CS23432 **SOFTWARE CONSTRUCTION**
Lab Manual **2024-2025**

Name : Sreyaskari M

Year/Branch/Section : II /CSE/ D

Register No. : 230701335

Semester : IV

Academic Year: 2024-25

INDEX

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

EX NO : 1 STUDY OF AZURE DEVOPS

AIM:

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

STUDY:

1. Understanding Azure DevOps

Azure DevOps consists of five key services:

1.1 Azure Repos (Version Control)

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

1.2 Azure Pipelines (CI/CD)

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

1.3 Azure Boards (Agile Project Management)

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

1.4 Azure Test Plans (Testing)

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

1.5 Azure Artifacts (Package Management)

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

Getting Started with Azure DevOps

Step 1: Create an Azure DevOps Account

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

Step 2: Set Up a Repository (Azure Repos)

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

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

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

Step 4: Manage Work with Azure Boards

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

Step 5: Implement Testing (Azure Test Plans)

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

RESULT:

The study was successfully completed.

EX NO : 2 PROBLEM STATEMENT

AIM :

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

In today's fast-paced world, timely and accurate weather updates are crucial for planning daily activities and ensuring safety. However, many individuals still rely on general forecasts that lack real-time precision and user interactivity. This creates a gap between available weather data and its effective delivery to end users in a user-friendly format.

The goal of this project is to develop a web-based **Weather Application** that provides real-time weather updates such as temperature, humidity, wind speed, pressure, and air quality index (AQI) for any specified location. The application should feature a clean, interactive user interface built using **HTML, CSS, and JavaScript**, and integrate with a reliable weather API to fetch and display accurate information.

To enhance usability, the application will include dynamic visuals (like emojis and background changes), alert notifications, and the ability to handle both expected and unexpected errors gracefully, ensuring reliability and continuous availability. Users should be able to search by city and receive weather parameters updated in real time with clear, visually engaging output.

RESULT:

The Problem statement is written successfully.

EX NO : 3 AGILE PLANNING

AIM:

To prepare an Agile Plan.

THEORY:

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

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

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

Steps in Agile planning process:

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

RESULT:

Thus the Agile plan was completed successfully.

EX NO: 4 CREATE USER STORY

AIM:

To create User Stories.

THEORY:

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

PROCEDURE:

1. Open your web browser and go to the Azure website: https://azure.microsoft.com/en-in Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.

2. If you don't have a Microsoft account, you can sign up for https://signup.live.com/?lic=1

3. Go to Azure Home Page.

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

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

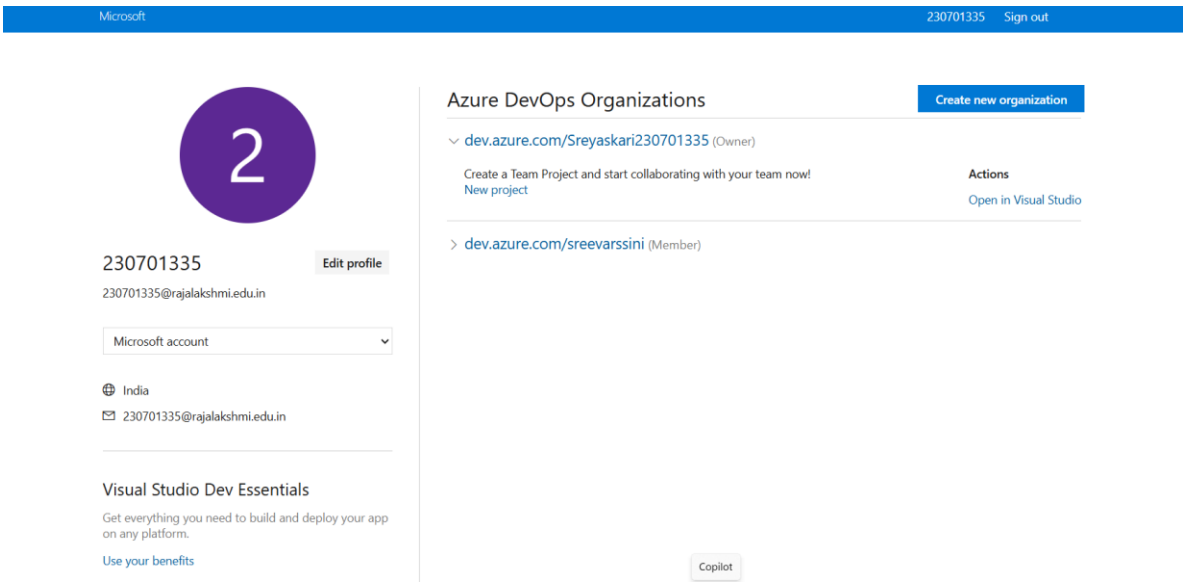
6. Create the First Project in Your Organization. After the organization is set up, you'll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.

(i)On the organization's Home page, click on the New Project button.

(ii)Enter the project name, description, and visibility options:

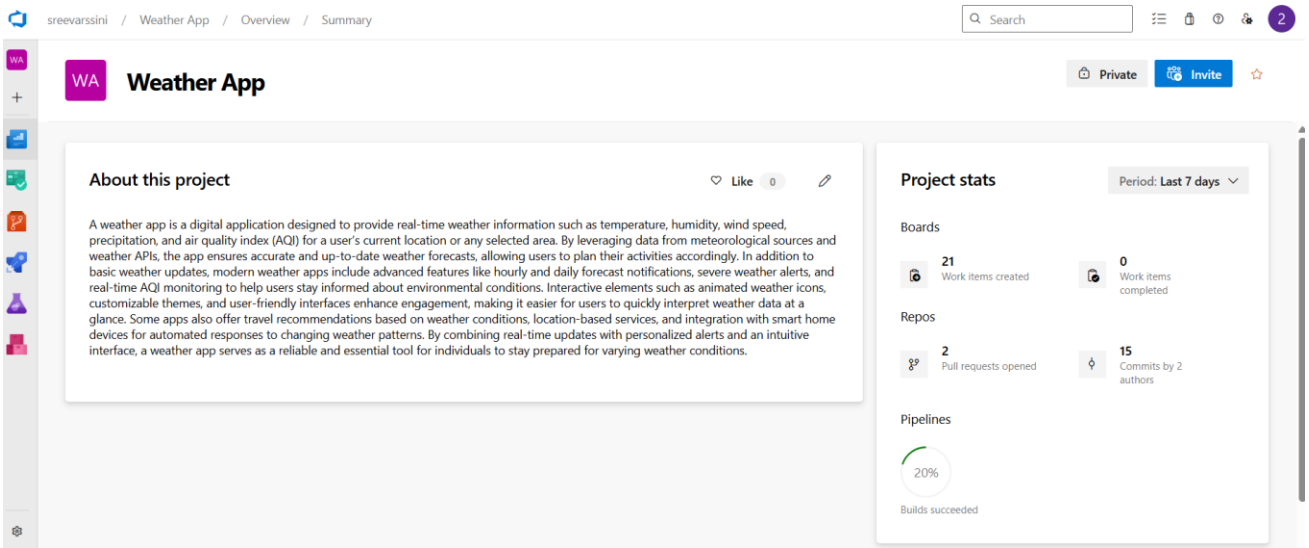
- Name: Choose a name for the project (e.g., LMS).
- Description: Optionally, add a description to provide more context about the project.
- Visibility: Choose whether you want the project to be Private
- (accessible only to those invited) or Public (accessible to anyone).

(iii) Once you've filled out the details, click Create to set up your first project.



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

8. Open project's dashboard.



9. To manage user stories:

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

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

10. Fill in the User Story.

sreevarssini / Weather App / Boards / Work items

Search

Did you notice Azure Boards has a new look and awesome new features? Learn more.

Recently updated | Back to Work Items | 1 of 3

USER STORY 7

7 As a user, I want to check weather conditions for different locations so that I can prepare for travel in advance.

230701335 | 0 Comments | Add Tag | Save | Follow

Updated by 230701335: 57m ago

State: New | Area: Weather App

Reason: New | Iteration: Weather App\USER STORY

Details

Description

The system should allow users to input or select multiple locations and retrieve current weather conditions as well as short-term forecasts. The displayed information should include temperature, precipitation, wind speed, and any weather alerts. This feature will help users make informed decisions and plan accordingly for upcoming travel.

Acceptance Criteria

1. *The system must fetch real-time weather data from a weather API every 15 minutes.*
2. *Users should be able to search for weather conditions by entering a city name, ZIP code, or using GPS location.*
3. *The app should display a 5-day weather forecast for selected locations.*
4. *The temperature should be available in Celsius and Fahrenheit, depending on user preference.*

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

Build

Build 230701332-CS23432-SC_20250520.1

Updated 9h ago Succeeded

Did you notice Azure Boards has a new look and awesome new features? Learn more.

Recently updated | Back to Work Items | 3 of 3

USER STORY 20

20 As a user, I want to get live weather updates for multiple saved locations so that I can monitor weather conditions in places I frequently travel to or have interest in.

230701335 | 0 Comments | Add Tag | Save | Follow

Updated by 230701335: 1h ago

State: New | Area: Weather App

Reason: New | Iteration: Weather App

Details

Description

This feature enables users to track weather in several locations of their choosing in real-time. Users can add, edit, or remove locations and get live updates without having to switch between screens. It's especially useful for users who have family in different areas or travel frequently.

Acceptance Criteria

1. Users can save multiple locations in the app.
2. Weather data for each saved location is updated in real time.
3. Users can view a list/grid view of live weather for all saved locations at once.
4. The user can set a preferred default location among the saved ones.
5. Notifications can be enabled for any or all saved locations

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

Build

Build 230701332-CS23432-SC_20250520.1

Updated 9h ago Succeeded

---

sreevarssini / Weather App / Boards / Work items

Search

Did you notice Azure Boards has a new look and awesome new features? Learn more.

Recently updated | Back to Work Items | 2 of 3

USER STORY 19

19 As a user, I want to receive live weather updates for my current location so that I can stay informed about sudden weather changes.

230701335 | 0 Comments | Add Tag | Save | Follow

Updated by 230701335: 58m ago

State: New | Area: Weather App

Reason: New | Iteration: Weather App

Details

Description

This feature allows the weather app to provide real-time updates based on the user's current GPS location. It will display the most up-to-date weather data such as temperature, wind, humidity, and alerts without requiring the user to refresh the app. This ensures that users are always aware of changing weather conditions and can take action if needed.

Acceptance Criteria

1. The app automatically detects the user's current location using GPS.
2. Live weather data is refreshed at a regular interval (e.g., every 5 minutes).
3. Users can opt-in or opt-out of automatic live updates.
4. The user interface clearly indicates when data was last updated.

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

Build

Build 230701332-CS23432-SC_20250520.1

Updated 9h ago Succeeded

EPIC: LIVE WEATHER UPDATES

Epic Live Weather Updates delivers real-time, high-impact weather coverage with pinpoint accuracy. Stay ahead of storms, heatwaves, and sudden changes with fast, reliable alerts and dynamic reports. Whether it's a calm day or a major weather event, we keep you informed—live and loud.

This epic consists of two major components:

1. Multi-Location Weather Search: Users should be able to check real-time weather conditions for any city or GPS-based location.

2. Detailed Weather Information: The app should display temperature, humidity, wind speed, precipitation, UV index, and air quality index (AQI) for the selected location.

FEATURES:

The **Live Weather Updates** feature ensures users can check real-time weather conditions for multiple locations, helping them plan their travel and daily activities efficiently. Below are the **detailed features** categorized based on functionality.

1. Multi-Location Weather Search

Description: Users can search for and access real-time weather conditions for any city, ZIP code, or their current GPS location.

Key Functionalities:

- Users can **search weather by**:
  - **City Name** (e.g., New York, London)
  - **ZIP Code** (e.g., 10001, 90210)
  - **GPS-based location** (automatic detection of user's location).
- **Auto-suggestion feature** when entering a location to improve usability.
- Users can save **up to 5 favorite locations** for quick access.
- If a location is **not found**, an appropriate **error message** should be displayed.
- Users should be able to **switch between locations** easily using a dropdown or tabs.
- If **network connectivity is low**, the app should **retry fetching data** before showing an error.

2. Detailed Weather Information

Description: The app should display comprehensive weather details, including temperature, humidity, wind speed, precipitation, UV index, and air quality index (AQI).

Key Functionalities:

- **Real-Time Weather Data:** The app fetches updates from a weather API every **15 minutes**.
- **Weather Metrics Displayed:**
  - **Temperature** (Celsius & Fahrenheit, switchable by user).
  - **Humidity** (Percentage).
  - **Wind Speed** (km/h or mph).
  - **Precipitation** (rain/snowfall in mm).
  - **UV Index** (low, moderate, high, very high, extreme).
  - **Air Quality Index (AQI)** categorized as:
    - **Good** (0-50)
    - **Moderate** (51-100)
    - **Unhealthy for Sensitive Groups** (101-150)
    - **Unhealthy** (151-200)
    - **Very Unhealthy** (201-300)
    - **Hazardous** (301+)
- **"Feels Like" Temperature:** Adjusted based on wind speed and humidity.
- Users can view **Sunrise & Sunset Times** for selected locations.
- Weather should **update automatically every 15 minutes** while the app is open.
- If an update fails, the app **retries fetching data 3 times** before displaying a cached version.

3. Hourly and Daily Forecasts

Description: Users should be able to view both **hourly** and **5-day weather forecasts** for better planning.

Key Functionalities:

- **Hourly Forecast:** Displays **temperature, wind speed, precipitation** for the next **24 hours**.
- **5-Day Forecast:** Users can check **daily temperature highs/lows, weather conditions, and precipitation chances**.
- Weather conditions should be visually represented with **icons and animations** (e.g., rain, sunny, snow).
- Each forecast should have a **detailed breakdown** when clicked, showing:
  - Wind speed changes
  - Humidity levels
  - UV index fluctuations
  - Precipitation probability

4. Personalization & UI Enhancements

Description: Users should be able to customize the app's appearance and units of measurement for better readability.

Key Functionalities:

- **Light & Dark Mode** toggle for better visibility.
- **Unit Switching**: Users can toggle between:
  - **Temperature**: Celsius (°C) / Fahrenheit (°F).
  - **Wind Speed**: km/h / mph.
- **Offline Mode:** The last **successful weather update** should be cached for offline access.
- **Network Handling:**
  - If the internet is unstable, the app **notifies users** and retries fetching data later.
  - If an **API error** occurs, the system displays the **last cached data** instead of an empty screen.

USER STORY 1:

As a user, I want to check weather conditions for different locations so that I can prepare for travel in advance.

Acceptance Criteria:

1. The system must fetch real-time weather data from a weather API every 15 minutes.

2. Users should be able to search for weather conditions by entering a city name, ZIP code, or using GPS location.

3. The app should display a 5-day weather forecast for selected locations.

4. The temperature should be available in Celsius and Fahrenheit, depending on user preference.

5. The app should support a “Feels Like” temperature based on humidity and wind speed.

6. Users should see hourly weather predictions for the next 24 hours.

7. The app should display sunrise and sunset times for the selected location.

8. Users must be able to add up to 5 favorite locations for quick access.

9. If a location cannot be found, the app should display an appropriate error message.

10. The air quality index (AQI) should be categorized as Good, Moderate, Unhealthy, etc

USER STORY 2:

As a user, I want to receive live weather updates for my current location so that I can stay informed about sudden weather changes.

Acceptance criteria:

1. The app automatically detects the user's current location using GPS.
2. Live weather data is refreshed at a regular interval (e.g., every 5 minutes).
3. Users can opt-in or opt-out of automatic live updates.
4. The user interface clearly indicates when data was last updated.
5. Severe weather alerts (e.g., storms, floods) are pushed to users in real time.
6. A visual cue (e.g., loading spinner or icon) shows when the app is updating weather data.
7. The app handles location permission requests and errors gracefully.
8. The live weather feed includes temperature, humidity, wind speed, and cloud coverage.
9. If location access is denied, the app prompts the user to manually enter a location.
10. The app can run in the background and still provide live alerts (with notification permissions).

USER STORY 3:

As a user, I want to get live weather updates for multiple saved locations so that I can monitor weather conditions in places I frequently travel to or have interest in.

Acceptance Criteria:

1. Users can save multiple locations in the app.
2. Weather data for each saved location is updated in real time.
3. Users can view a list/grid view of live weather for all saved locations at once.
4. The user can set a preferred default location among the saved ones.
5. Notifications can be enabled for any or all saved locations.
6. The app allows easy removal or reordering of saved locations.
7. Each location card displays temperature, condition icon, and update timestamp.
8. Tapping a location opens a detailed view with extended forecast and live data.
9. Location updates do not slow down app performance or crash the UI.
10. The app uses efficient data fetching to minimize battery and data usage.

RESULT:

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

EX NO: 5 SEQUENCE DIAGRAM

AIM:

To design a Sequence Diagram by using Mermaid.js

THEORY:

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

PROCEDURE:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu.

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

```
::: mermaid
sequenceDiagram
User->+WeatherApp: Open App
   WeatherApp->+WeatherAPI: Fetch Weather
   WeatherAPI->+WeatherAPI: Get Data
   WeatherAPI-->-WeatherApp: Return Data
   WeatherApp-->-User: Update UI

   User->+WeatherApp: Set Notification Preferences
   WeatherApp->+NotificationSystem: Store Preferences
   NotificationSystem-->-WeatherApp: Preferences Stored

   User->+WeatherApp: Check for Alerts
   WeatherApp->+WeatherAPI: Fetch Extreme Weather
   WeatherAPI-->-WeatherApp: Return Alerts
   WeatherApp-->-User: Display Alerts
   WeatherApp->+NotificationSystem: Send Alert Notification
   NotificationSystem->+User: Notify User
   NotificationSystem-->-WeatherApp: Confirmation
```

Admin->+WeatherApp: Login

WeatherApp->+WeatherAPI: Update Weather Data

WeatherAPI->+WeatherAPI: Push New Data

WeatherAPI-->-WeatherApp: Confirm Update

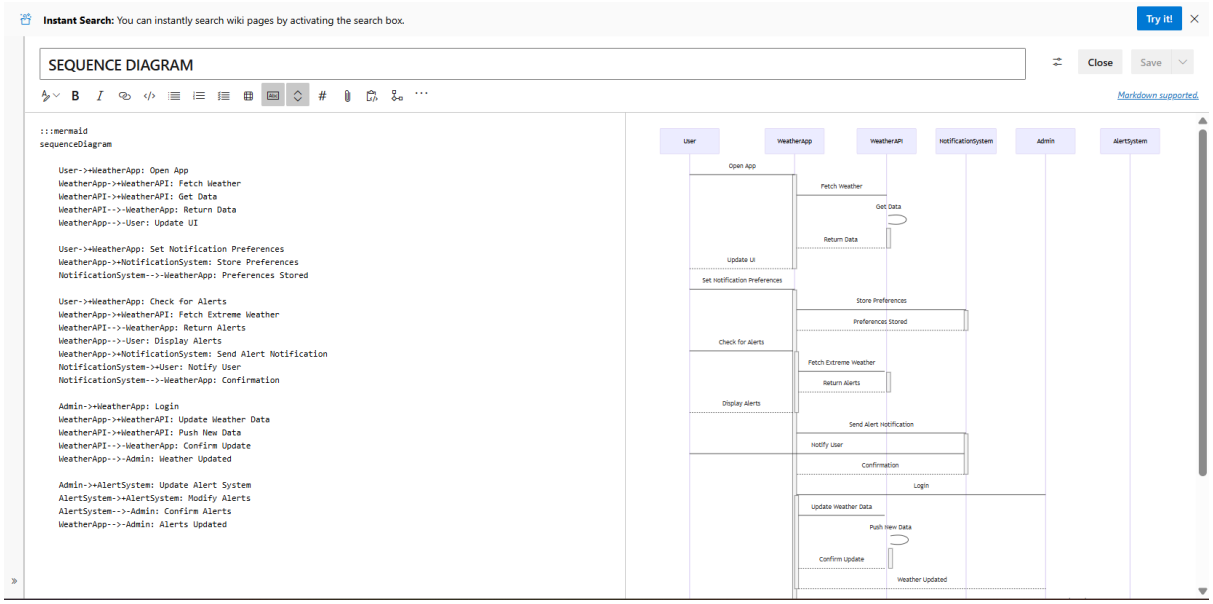
WeatherApp-->-Admin: Weather Updated

Admin->+AlertSystem: Update Alert System

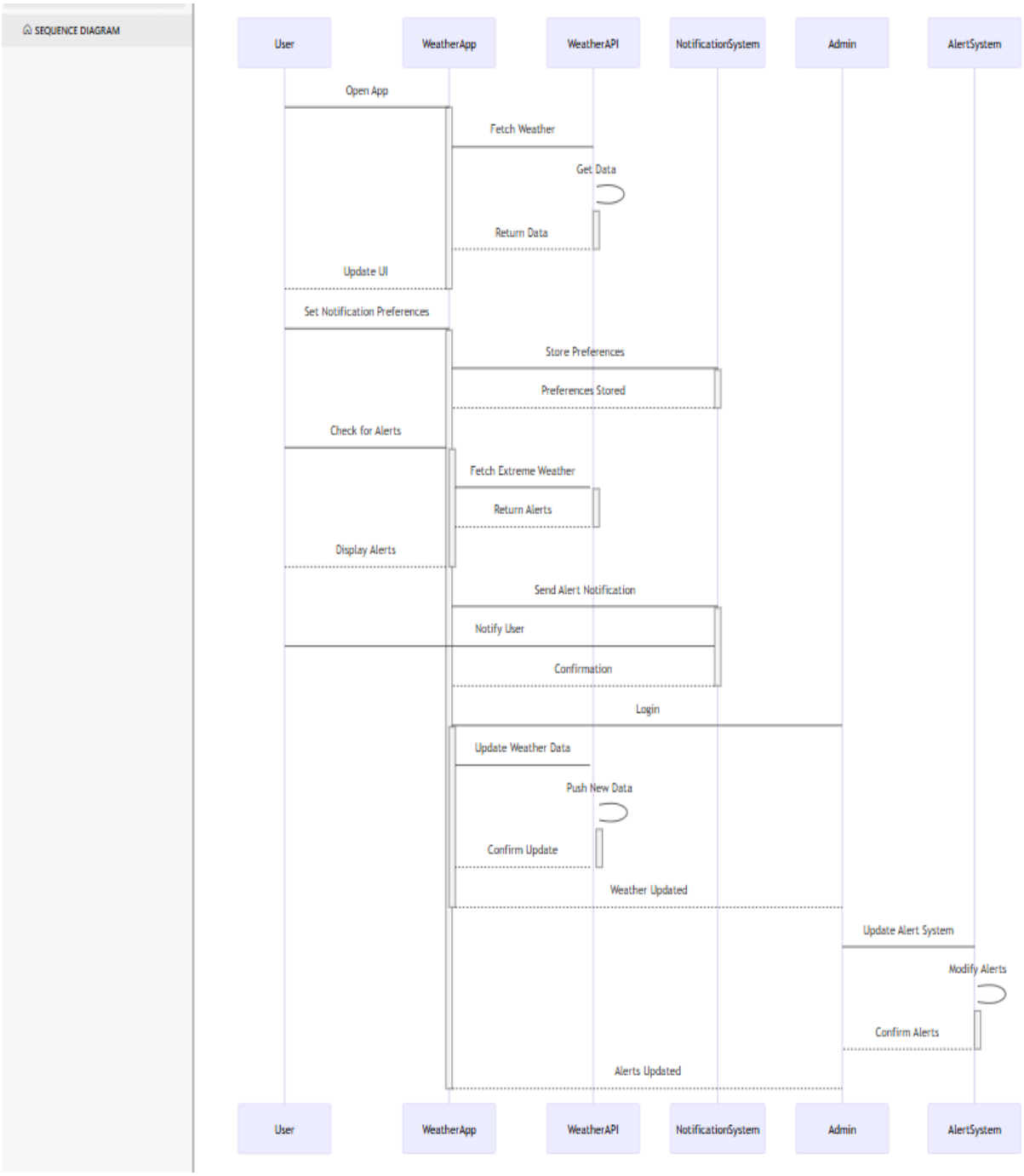
AlertSystem->+AlertSystem: Modify Alerts

AlertSystem-->-Admin: Confirm Alerts

WeatherApp-->-Admin: Alerts Updated



4. Click wiki menu and select the page.



RESULT:

The sequence diagram is drawn successfully.

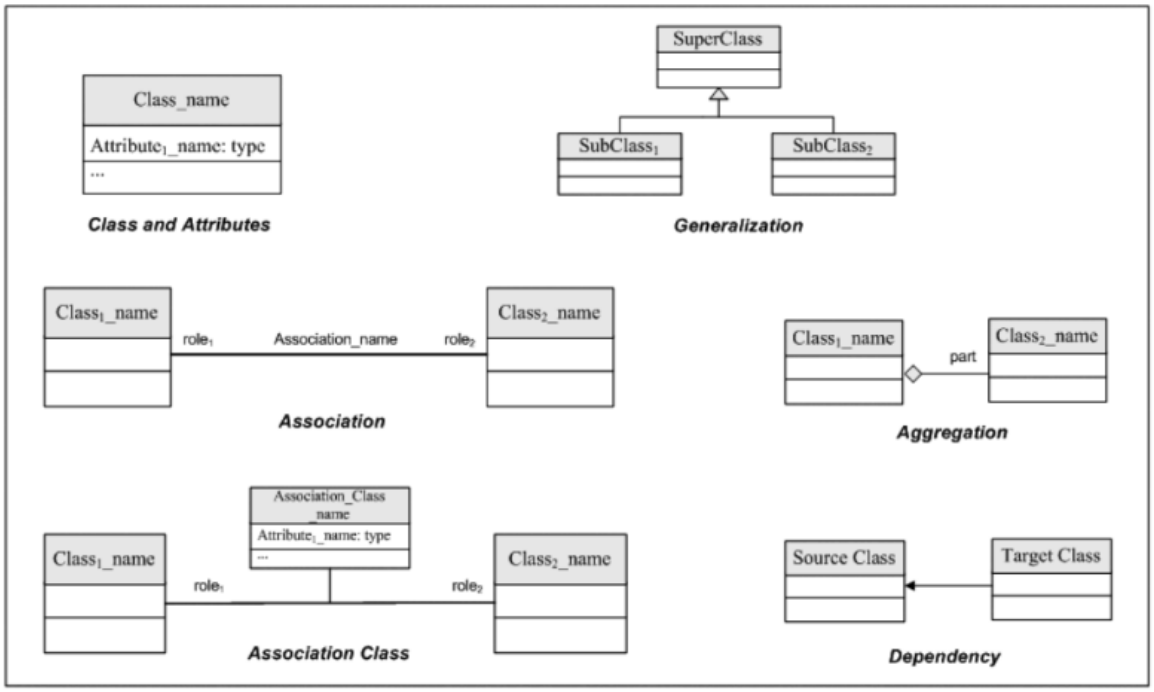
EX NO: 6 CLASS DIAGRAM

AIM:

To draw a simple class diagram.

THEORY:

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



Notations in class diagram

PROCEDURE:

1. Open a project in Azure DevOps Organisations.

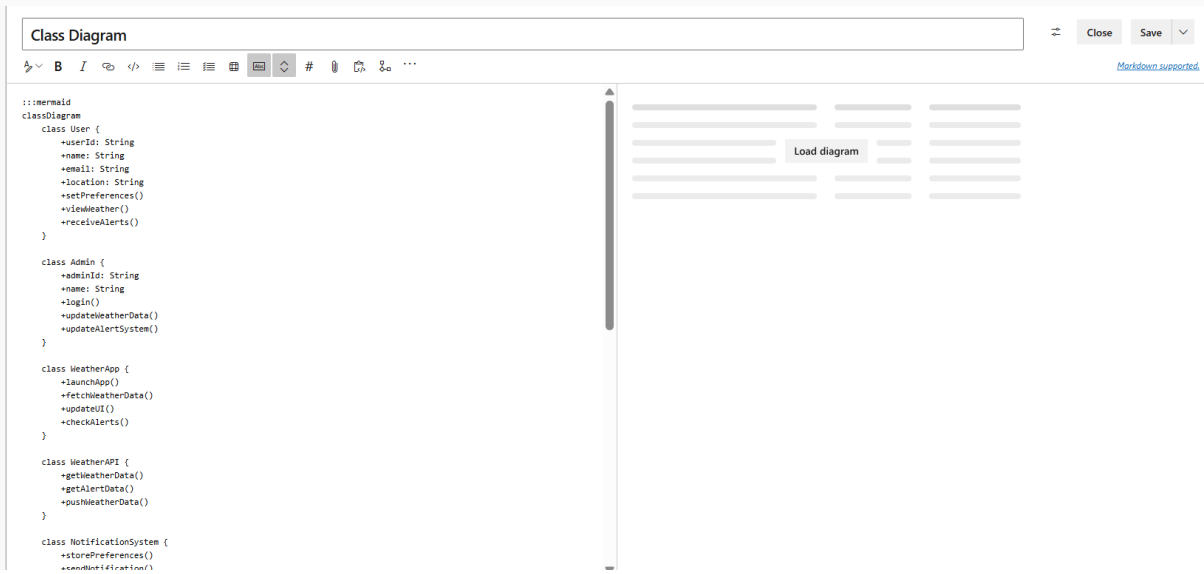
2. To design select wiki from menu.

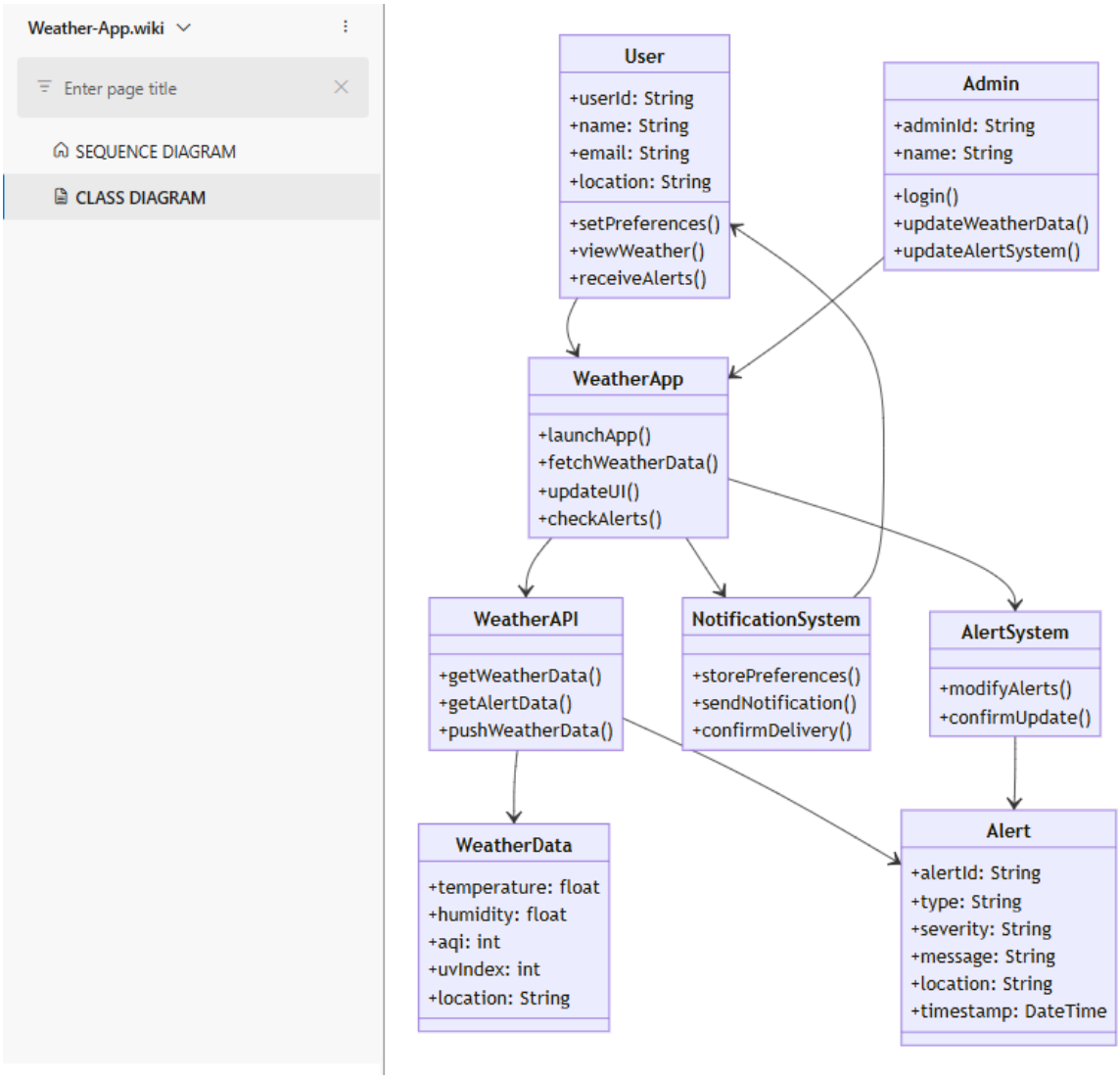
3. Write the code for drawing Class Diagram and save the code.

```
:::mermaid
classDiagram
   class User {
      +userId: String
      +name: String
      +email: String
      +location: String
```

```
    +setPreferences()
    +viewWeather()
    +receiveAlerts()
}

class Admin {
    +adminId: String
    +name: String
    +login()
    +updateWeatherData()
    +updateAlertSystem()
}

class WeatherApp {
    +launchApp()
    +fetchWeatherData()
    +updateUI()
    +checkAlerts()
}

class WeatherAPI {
    +getWeatherData()
    +getAlertData()
    +pushWeatherData()
}

class NotificationSystem {
    +storePreferences()
    +sendNotification()
    +confirmDelivery()
}

class AlertSystem {
    +modifyAlerts()
    +confirmUpdate()
}

class WeatherData {
    +temperature: float
    +humidity: float
    +aqi: int
    +uvIndex: int
    +location: String
}
```

```
class Alert {
    +alertId: String
    +type: String
    +severity: String
    +message: String
    +location: String
    +timestamp: DateTime
}

User --> WeatherApp
Admin --> WeatherApp
WeatherApp --> WeatherAPI
WeatherApp --> NotificationSystem
WeatherApp --> AlertSystem
WeatherAPI --> WeatherData
WeatherAPI --> Alert
NotificationSystem --> User
AlertSystem --> Alert
```





RESULT:

Thus the class diagram has been designed successfully.

EX NO: 7 USE CASE DIAGRAM

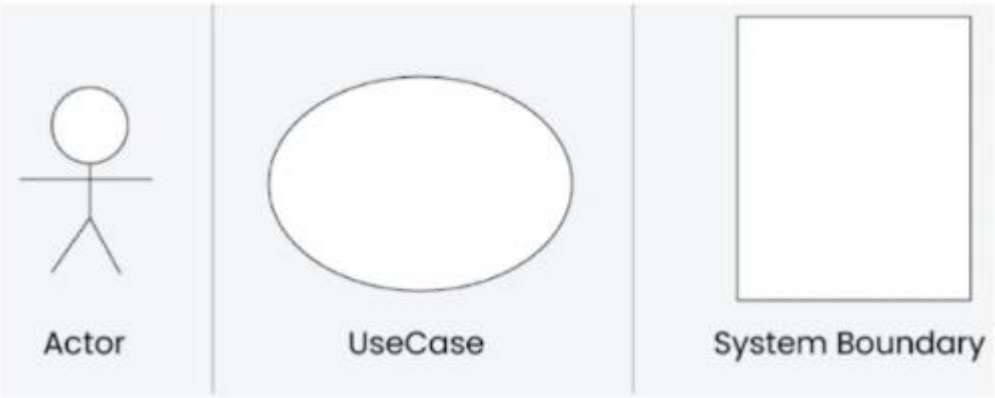
AIM:

Steps to draw the Use Case Diagram using draw.io

THEORY:

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

- Use Cases
- Actors
- Relationships
- System Boundary



PROCEDURE :

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

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

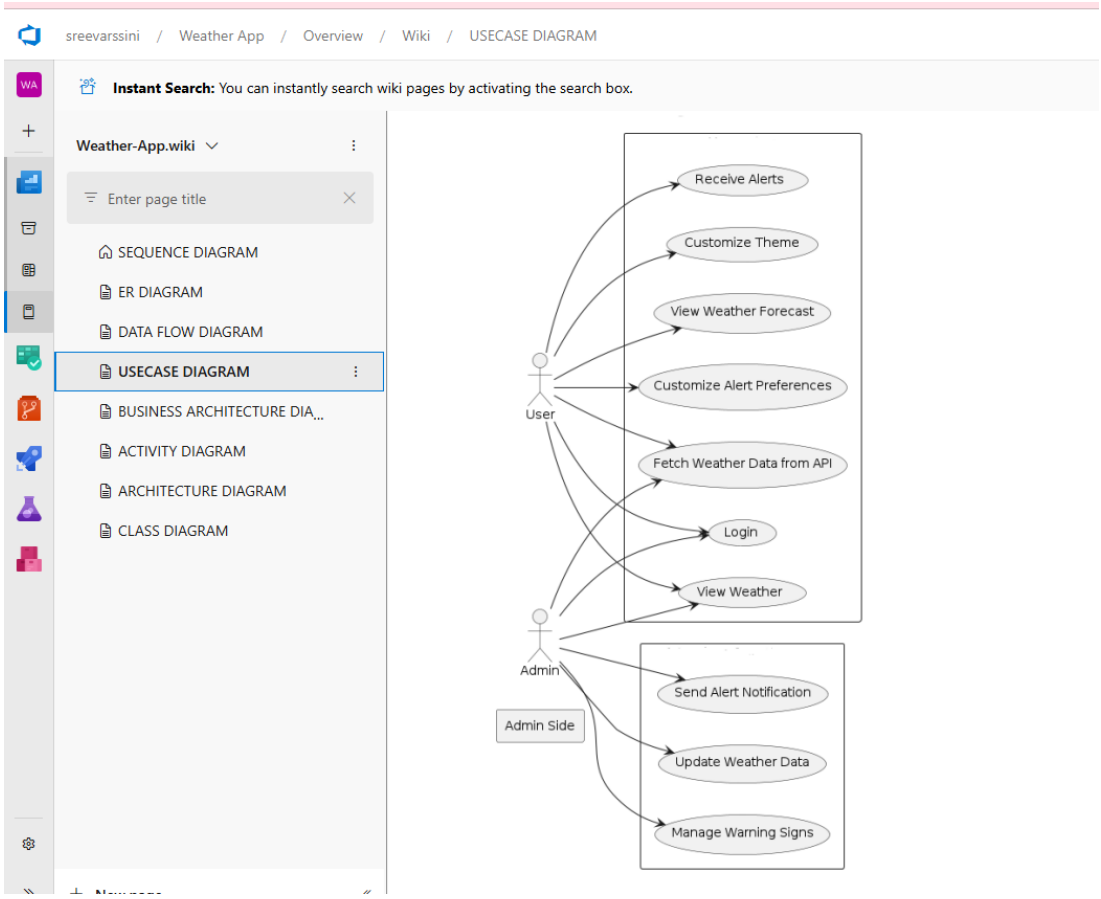
Step 2: Upload the Diagram to Azure DevOps

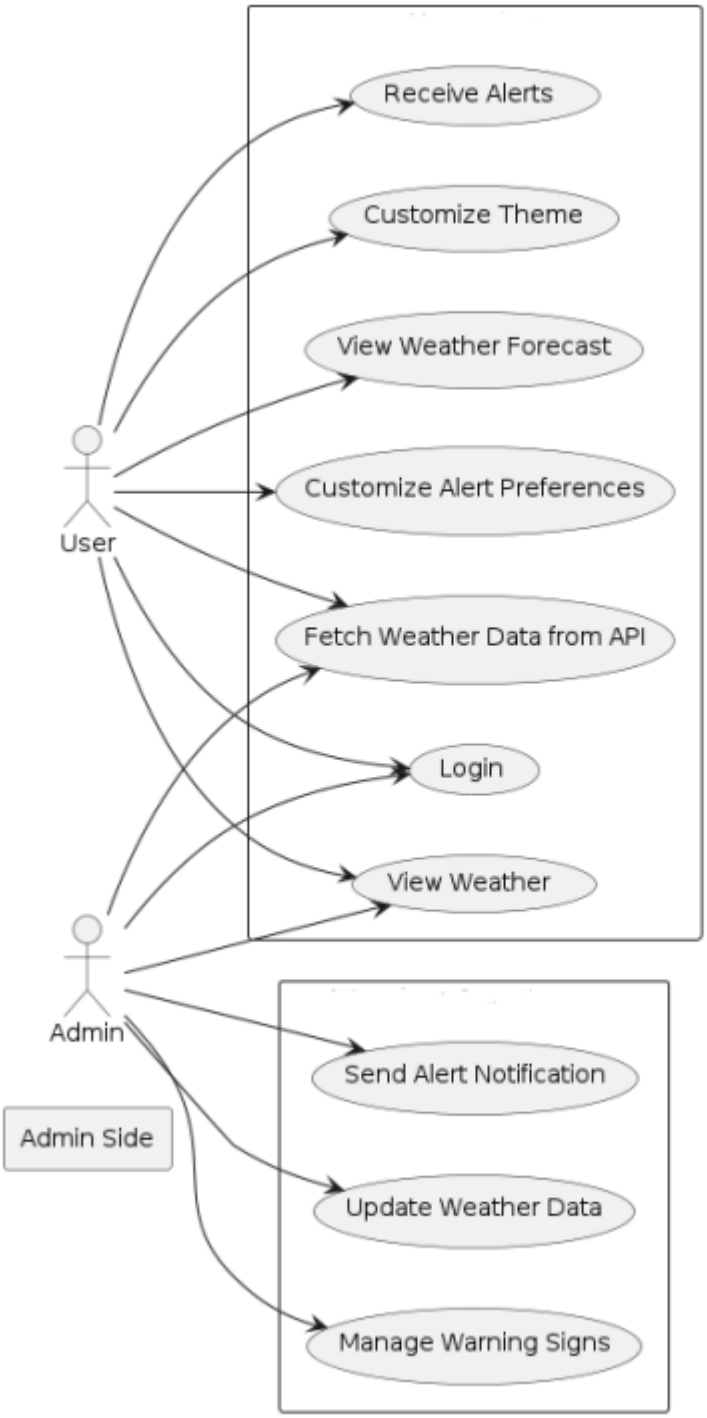
Option 1: Add to Azure DevOps Wiki

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

Option 2: Attach to Work Items in Azure Boards

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





RESULT:

The use case diagram was designed successfully.

EX NO: 8 ACTIVITY DIAGRAM

AIM :

To draw a sample activity diagram for the Weather Application.

THEORY:

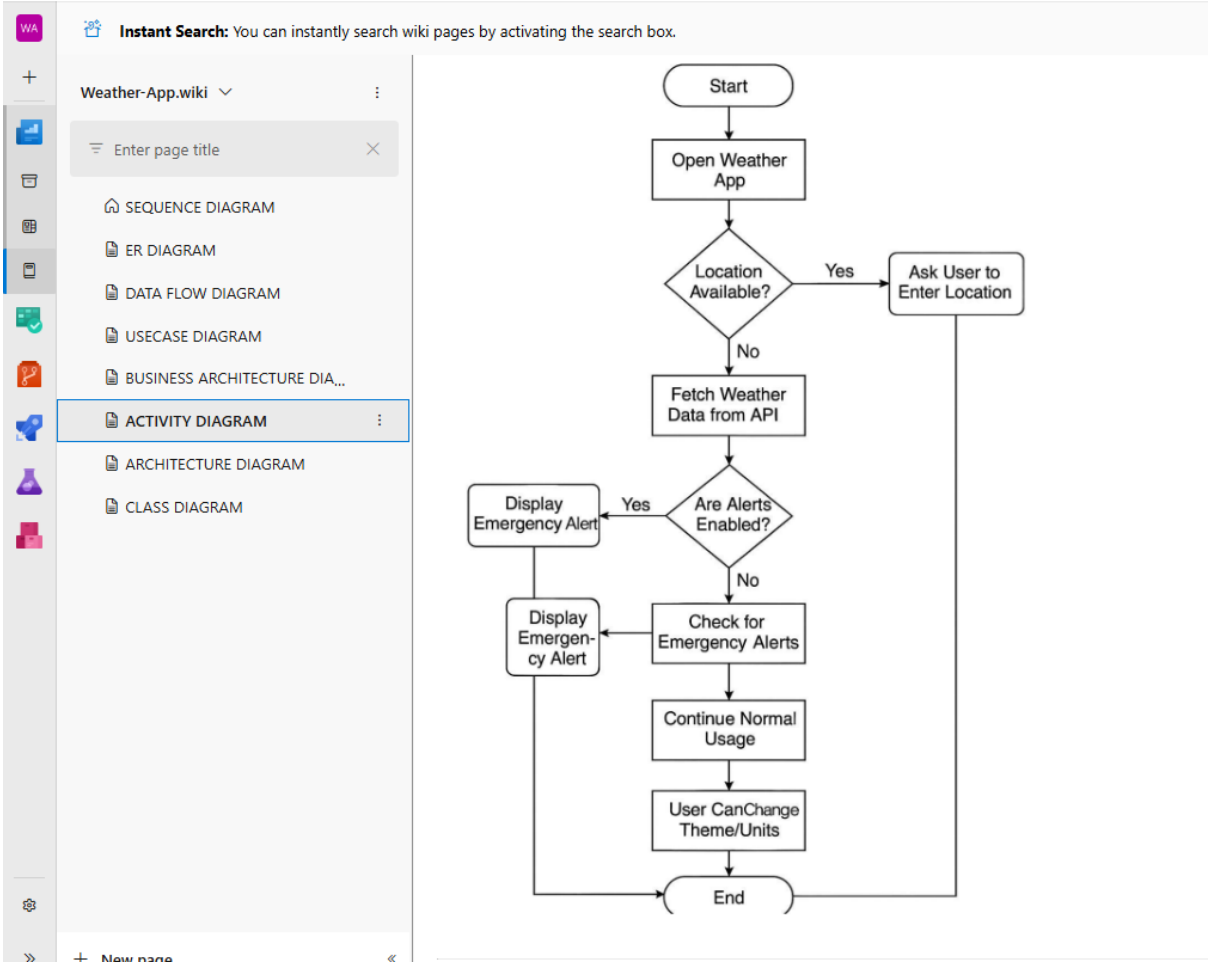
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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 o communicate additional messages
Send signal		Show that a signal is being sent to a receiving activity
Receive signal		Demonstrates the acceptance of an event
Flow final symbol		Represents the end of a specific process flow
Option loop		Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate		Represents a transition that invokes the last active state.
End		Marks the end state of an activity and represents the completion of all flows of a process

PROCEDURE:

Step 1. Draw diagram in draw.io.

Step 2. Upload the diagram in Azure DevOps wiki.



RESULT:

The activity diagram was designed successfully.

EX NO: 9 ARCHITECTURE DIAGRAM

AIM:

To draw the Architecture Diagram using draw.io.

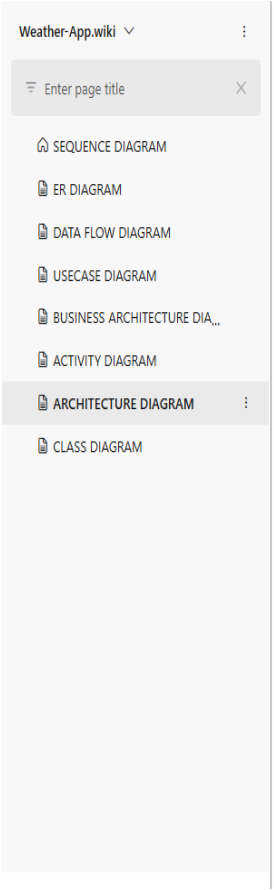
THEORY:

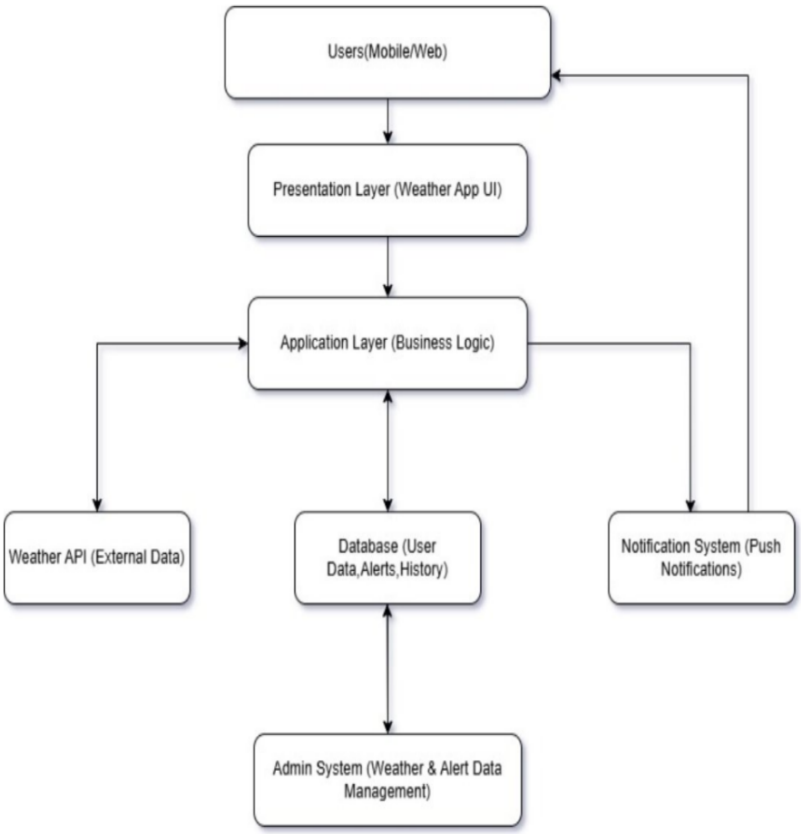
An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

PROCEDURE:

Step 1. Draw diagram in draw.io

Step 2. Upload the diagram in Azure DevOps wiki.





RESULT:

The architecture diagram was designed successfully

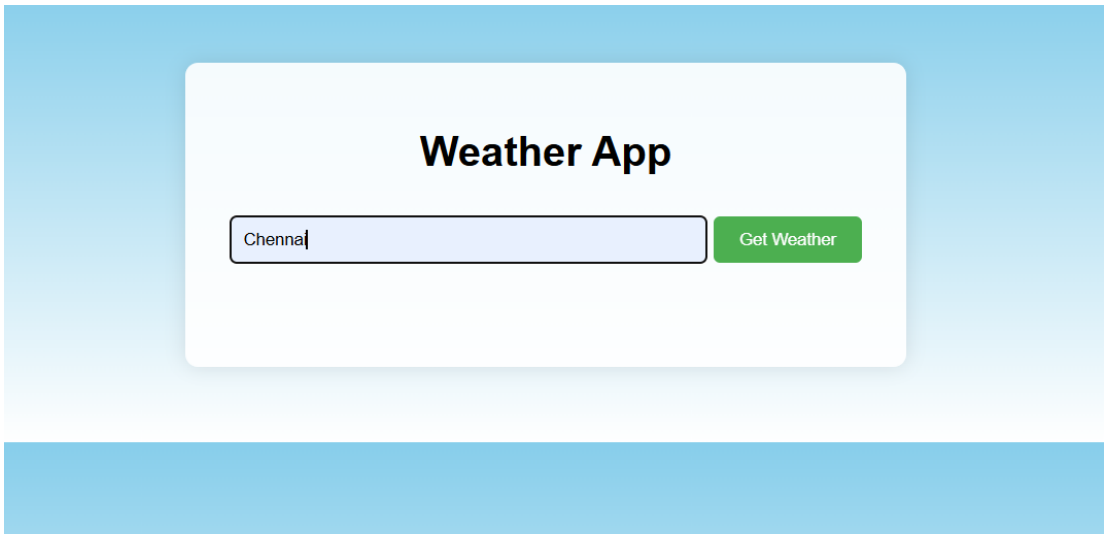
EX NO: 10 USER INTERFACE

AIM:

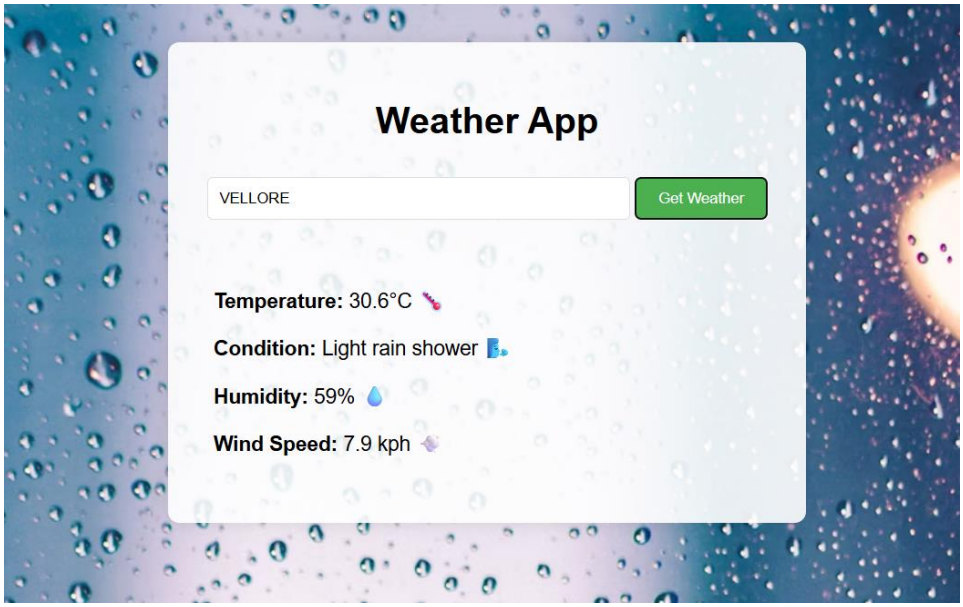
Design User Interface for the Weather App.

UI DESIGNS OF WEATHER APP:

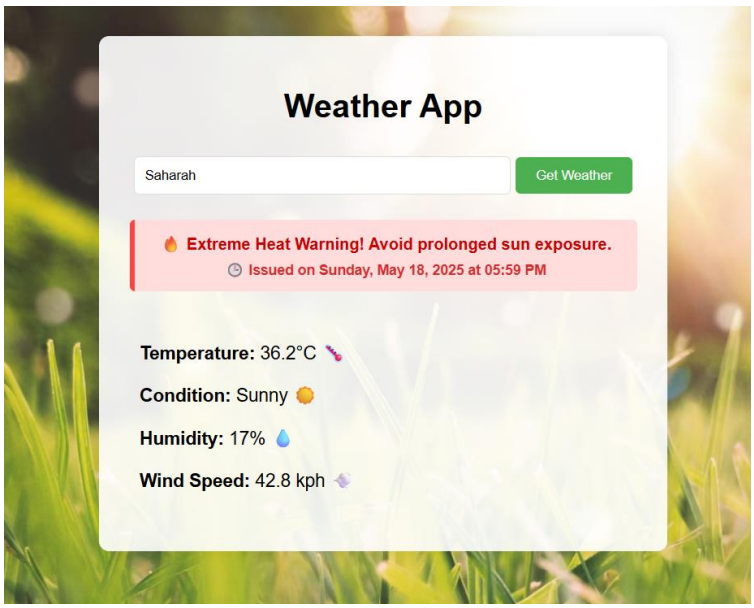
HOME PAGE:



WEATHER DISPLAY:



WEATHER DISPLAY WITH ALERT MESSAGE:



DIFFERENT BACKGROUND FOR APPROPRIATE WEATHER:

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

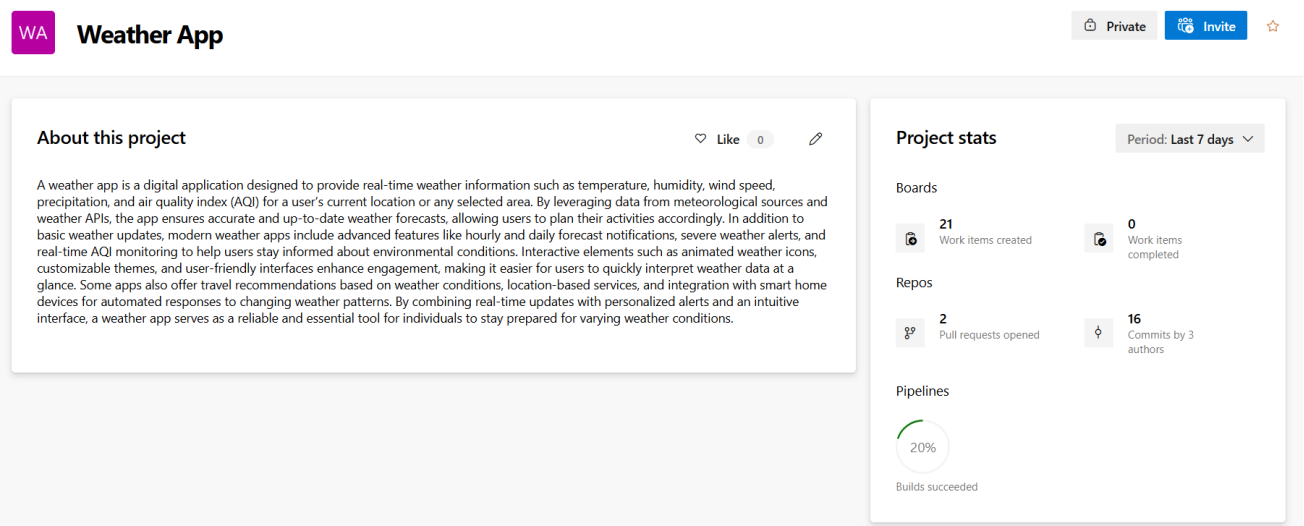
- Navigate to Repos → Click "Clone" to get the Git URL.
- Open Visual Studio Code / Terminal and run:

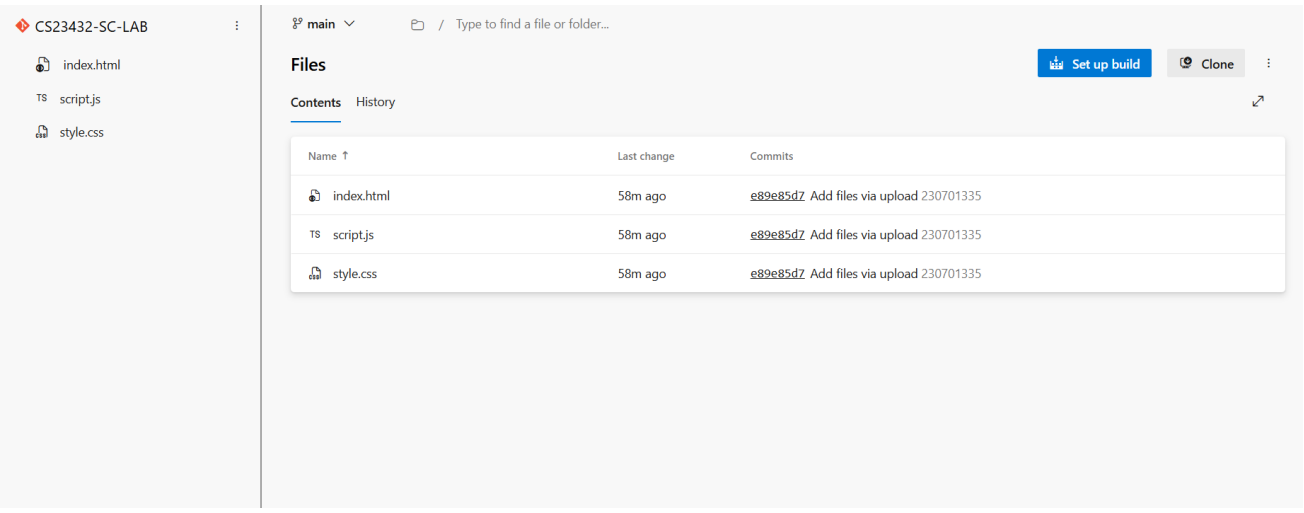
```
git clone <repo_url>
cd <repo_folder>
```

- 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)





RESULT :

Thus the application was successfully implemented.

EX NO: 12 TESTING USING AZURE

AIM:

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

PROCEDURE:

Step 1: Open Azure DevOps Project

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

Step 2: Navigate to Test Plans

- In the left menu, click on Test Plans.
- Click on "New Test Plan" and give it a name (e.g., *Emergency Alerts*).

Step 3: Create Test Suites

- Under the test plan, click "+ New Suite" to add test suites for each Epic or Feature.
    - Suite 1: *Emergency Alerts*
    - Suite 2: *Custom Alert Preferences*
    - Suite 3: *Safety Tips Display*

Step 4: Add Test Cases

- Click on a suite → + New Test Case.
- Enter the following details for each test case:
    - Test Case ID: (e.g., TC001)
    - Title: (e.g., *Receive Notification in Time*)
    - Scenario: Describe the situation being tested.
    - Steps:
        - Launch the app
        - Trigger extreme weather API event
        - Observe time taken for notification
    - Expected Result: Notification should appear within 10 seconds.

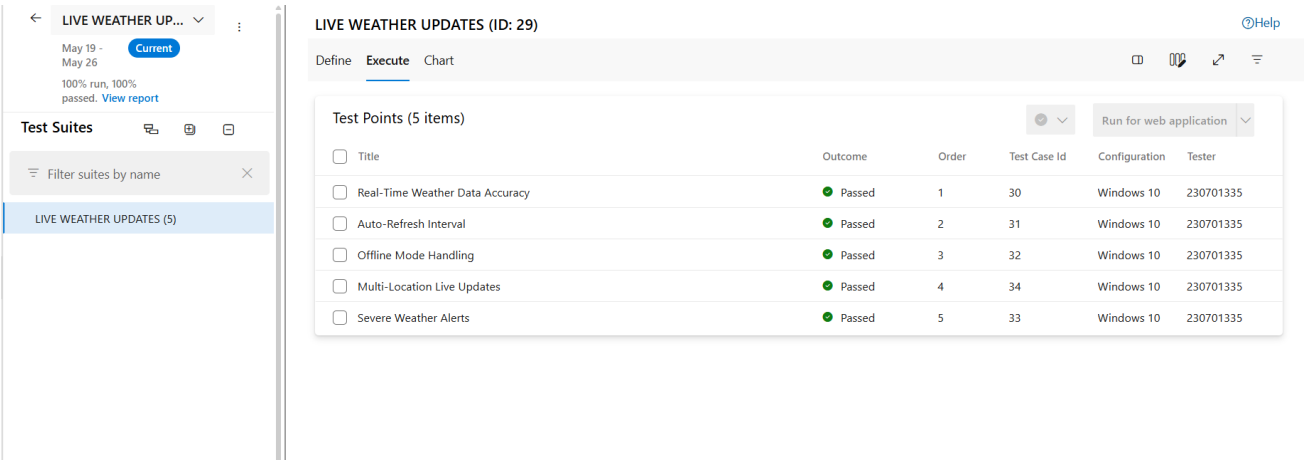
Step 5: Execute Test Cases

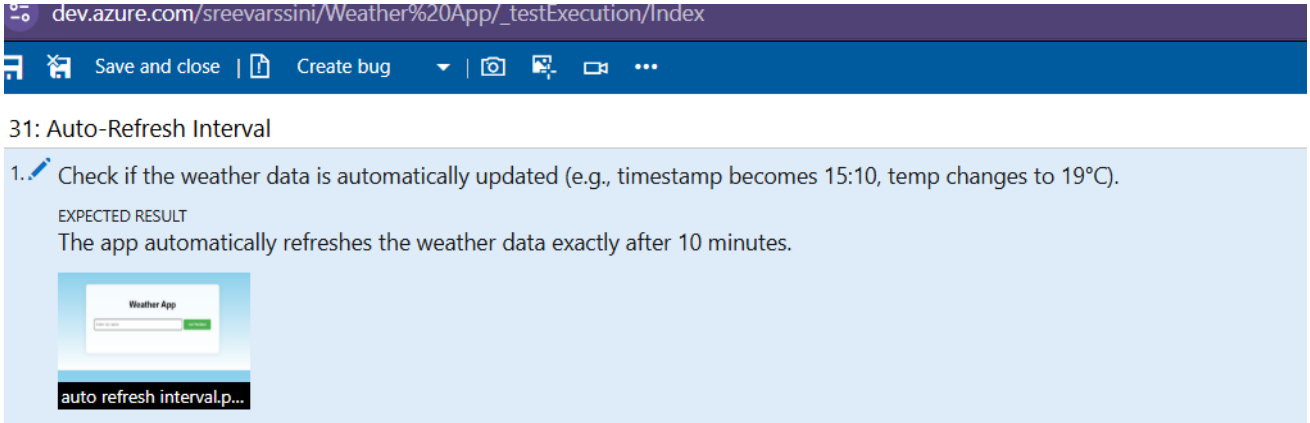
- Click on each test case and select Run for web application.

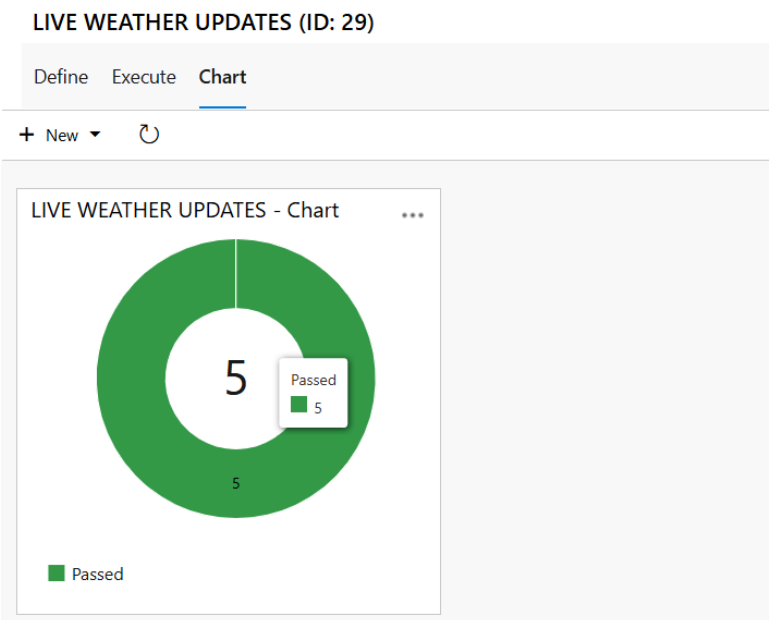
- Follow the steps, mark results as Pass/Fail, and provide Actual Result and Remarks.

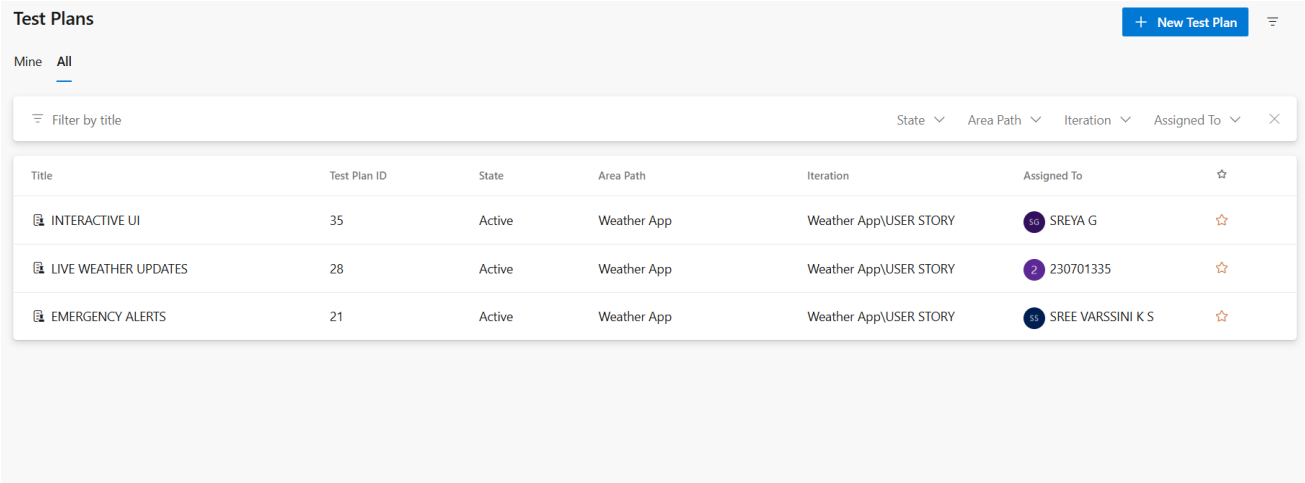
Step 6: Track and Report

- Go to Test Plans → Charts to view test progress.
- Use filters to track:
  - Passed/Failed test cases
  - Test coverage per user story
  - Bugs linked to failed test cases









RESULT:

Thus the application was successfully tested in Azure.

EX NO: 13 CI/CD PIPELINE

AIM:

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

PROCEDURE:

Step 1: Create a Build Pipeline (CI)

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

```
trigger:
  - main
pool:
  name: Default
steps:
  - task: UseNode@2
    inputs:
      version: '18.x'
  - script: npm install
    displayName: 'Install Dependencies'
  - script: npm run build
    displayName: 'Build Application'

  - script: npm run test
    displayName: 'Run Tests'
```

- Save and Run the pipeline to verify.

Step 4: Set Up Release Pipeline (CD)

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

Step 5: Add Approvals and Gates (Optional)

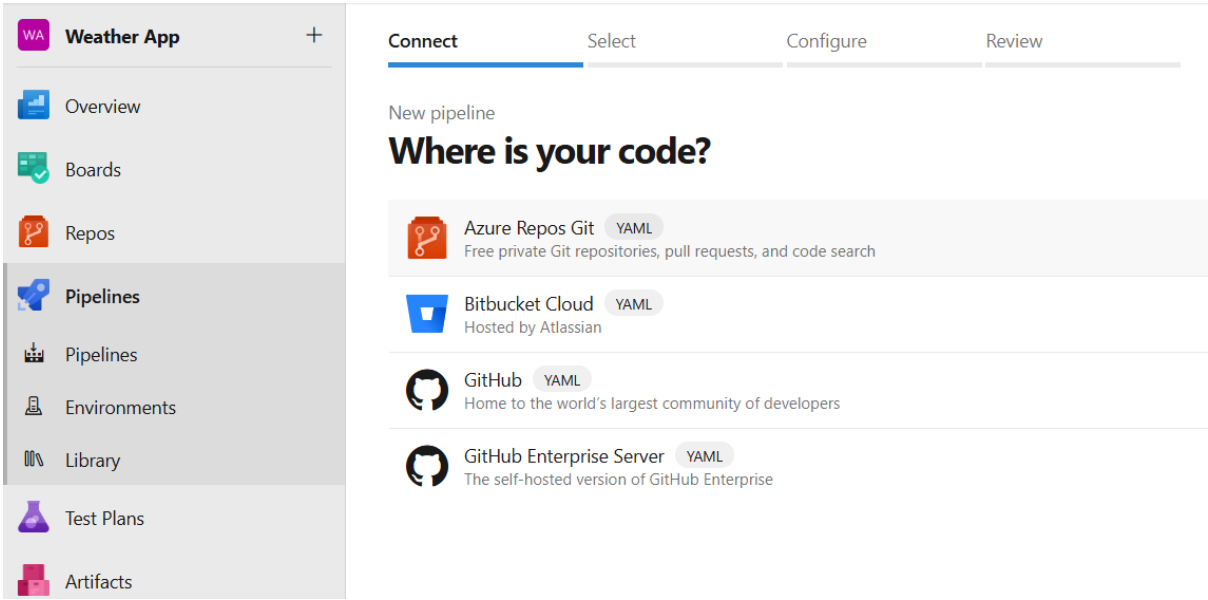
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

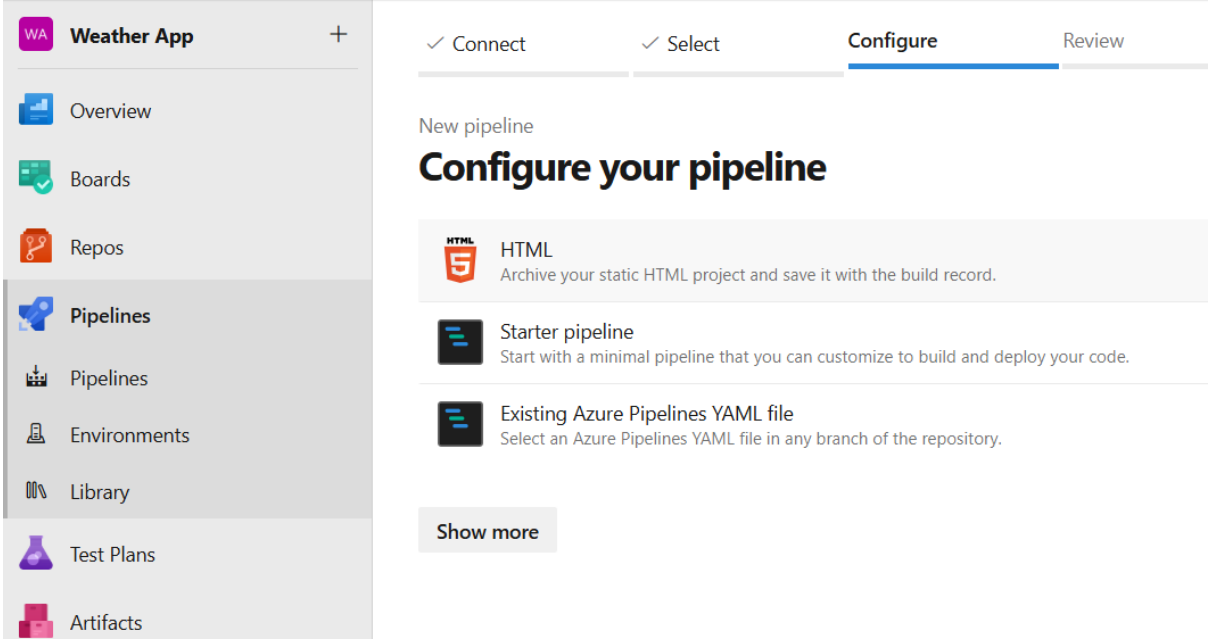
Step 6: Automate Triggering

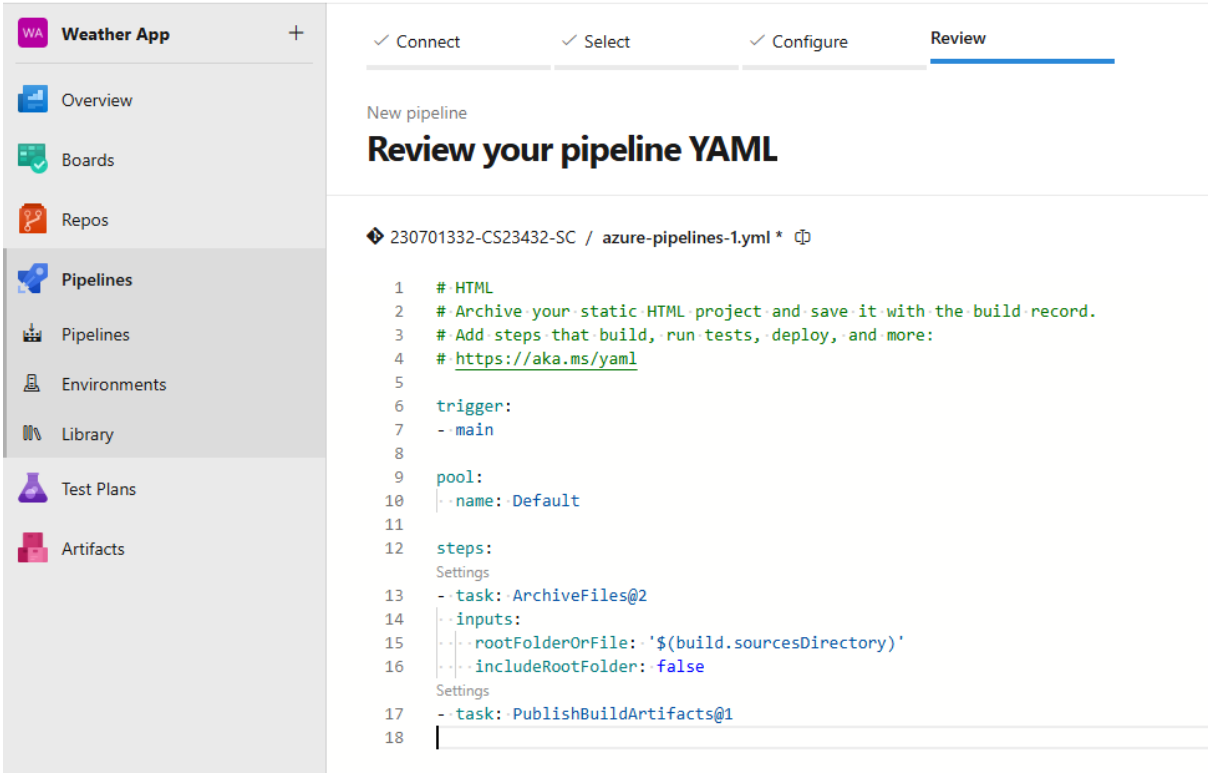
- Ensure the pipeline triggers:
  - On code push to main branch (CI)
  - After successful build for deployment (CD)

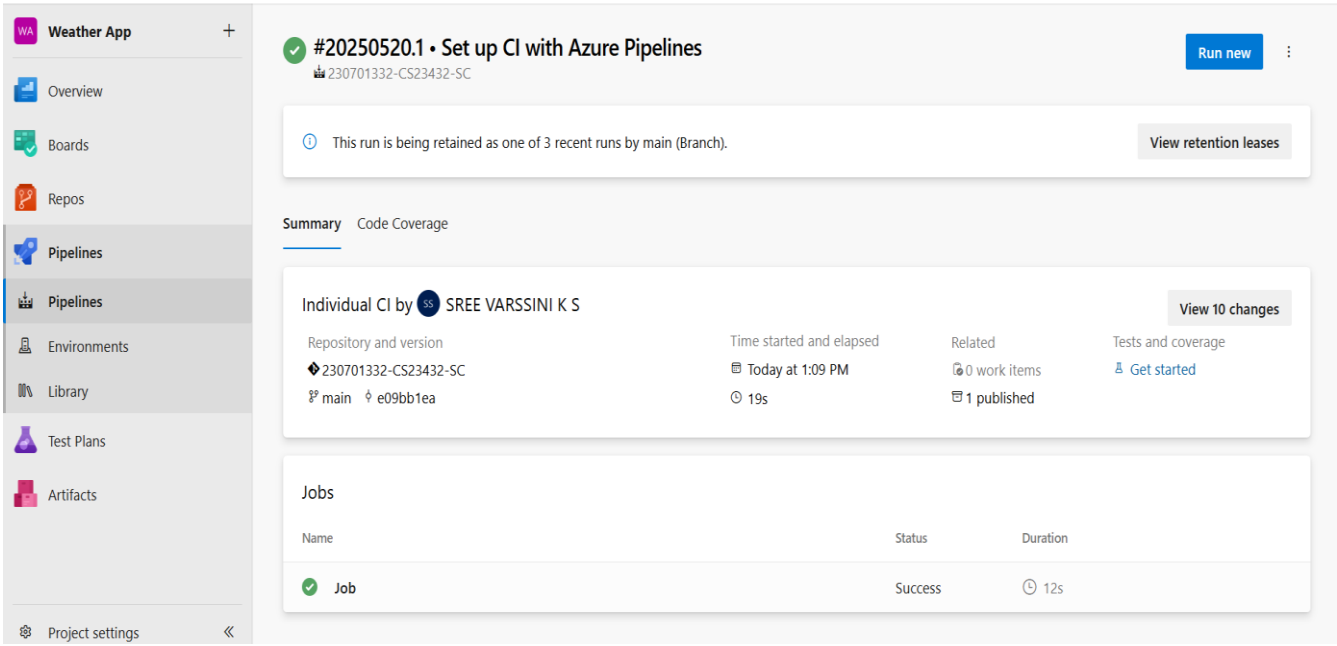
Step 7: Monitor Pipeline

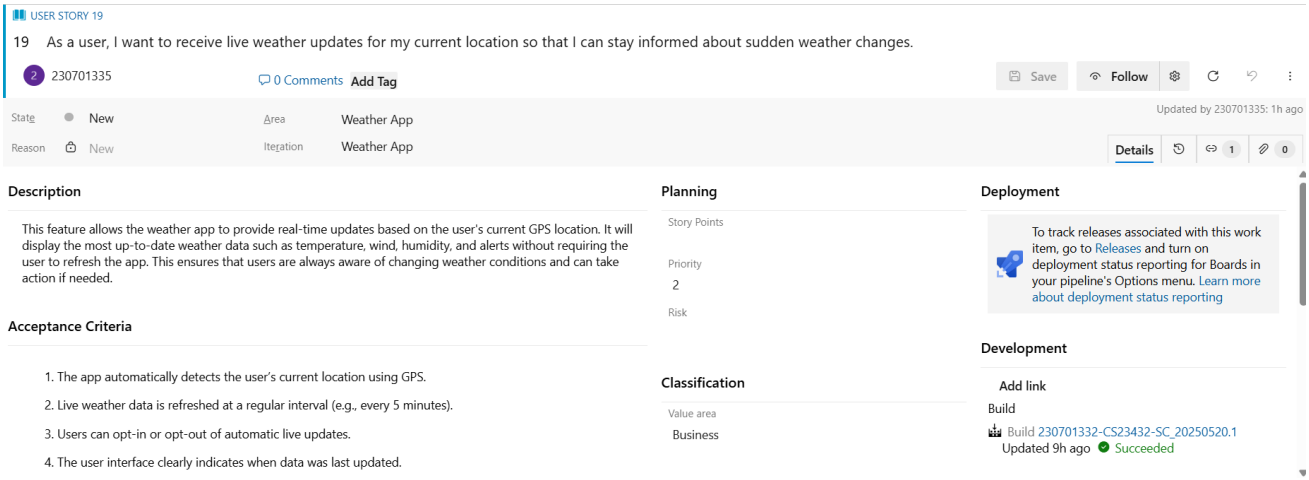
- Track pipeline status under Pipelines → Runs.
- Debug failures and download logs if necessary.
- Use Azure Boards to link builds with user stories and bugs.




Weather App
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Connect
Select
Configure
Review
New pipeline
Configure your pipeline
HTML
Archive your static HTML project and save it with the build record.
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.
Show more


Weather App
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Connect
Select
Configure
Review
New pipeline
Review your pipeline YAML
230701332-CS23432-SC / azure-pipelines-1.yml *
HTML
Archive your static HTML project and save it with the build record.
Add steps that build, run tests, deploy, and more:
https://aka.ms/yaml
trigger:
- main
pool:
  name: Default
steps:
Settings
- task: ArchiveFiles@2
  inputs:
    rootFolderOrFile: '$(build.sourcesDirectory)'
    includeRootFolder: false
Settings
- task: PublishBuildArtifacts@1





RESULT:

Thus the CI/CD pipeline has been successfully implemented.