

RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM – 602 105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025



**RAJALAKSHMI
ENGINEERING COLLEGE**

CS23432

SOFTWARE CONSTRUCTION

Lab Manual

2024-2025

Name : Sree Varssini K S

Year/Branch/Section : II /CSE/ D

Register No. : 230701332

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 Usecase and Class Diagram
7	Designing Interaction Diagrams
8	Design Interface
9	Implementation – Design a Web Page based on Scrum Methodology
10	Testing using Azure.
11	Deployment

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:

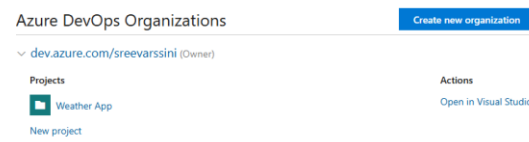
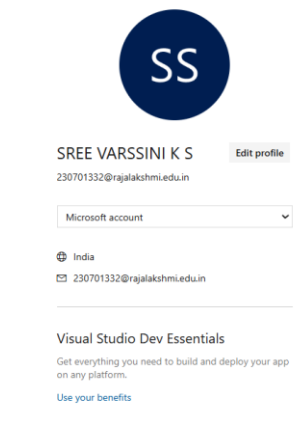
To create User Stories.

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.

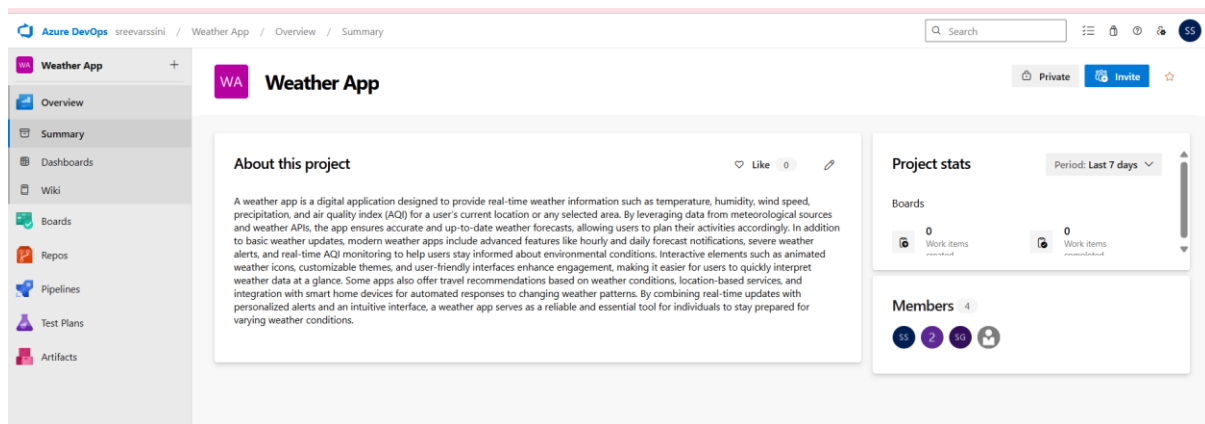
PROCEDURE:

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.

USER STORY 18

18 As a user, I want the app to provide safety tips along with emergency alerts so that I know how to respond to hazardous weather conditions.

SREE VARSSINI K S0 CommentsAdd Tag

SaveFollowSettingsShareLink

Updated by SREE VARSSINI K S: Apr 17

Details🔍🔄🔗📄

StatgNewAreaWeather AppReasonNewIterationWeather AppUSER STORY

Description

This user story aims to give users greater control over the types of weather alerts they receive by allowing them to customize their notification preferences. Users can choose specific weather events—such as storms, heavy rain, heatwaves, snowfall, or high winds—for which they want to be alerted. This personalization ensures that users receive only the most relevant and meaningful warnings based on their interests, location, or sensitivity to certain conditions (e.g., health concerns during heatwaves). By avoiding unnecessary or unwanted alerts, the feature enhances user satisfaction, prevents alert fatigue, and ensures that important warnings are more likely to be noticed and acted upon.

Acceptance Criteria

1. Each alert includes a "View Safety Tips" link.

2. Tips are tailored to the alert type (e.g., earthquake vs. heatwave).

3. Safety tips are stored locally for offline access.

4. Tips include icons/images for better understanding.

5. Tips are written in bullet points for clarity.

6. Tips can be bookmarked for quick access.

7. The app supports safety tip translation to user language.

8. Tips are reviewed and updated regularly by admins.

9. Tips open in a new section/popup within the app.

10. Tips are less than 300 words per alert.

11. Each tip page has a share option.

12. Users can give feedback on tip usefulness.

13. App allows offline viewing of previously viewed tips.

14. Tips follow national emergency guidelines.

15. Tips are not shown if not relevant to the alert.

Planning

Story Points

Priority2

Risk

Classification

Value areaBusiness

Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Development

Add link

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

Related Work

Add link

[Add an existing work item as a parent](#)

USER STORY 17

17 As a user, I want to customize alert preferences for different weather events (storms, heavy rain, heatwaves, etc.) so that I receive only relevant warnings.

SREE VARSSINI K S0 CommentsAdd Tag

SaveFollowSettingsShareLink

Updated by SREE VARSSINI K S: Apr 17

Details🔍🔄🔗📄

StatgNewAreaWeather AppReasonNewIterationWeather AppUSER STORY

Description

This user story aims to give users greater control over the types of weather alerts they receive by allowing them to customize their notification preferences. Users can choose specific weather events—such as storms, heavy rain, heatwaves, snowfall, or high winds—for which they want to be alerted. This personalization ensures that users receive only the most relevant and meaningful warnings based on their interests, location, or sensitivity to certain conditions (e.g., health concerns during heatwaves). By avoiding unnecessary or unwanted alerts, the feature enhances user satisfaction, prevents alert fatigue, and ensures that important warnings are more likely to be noticed and acted upon.

Acceptance Criteria

1. Users can enable/disable individual weather event types.

2. Users can select severity levels (High, Medium, Low).

3. Preferences are saved after app restart.

4. Users can change preferences at any time.

5. Default preference includes all alert types.

6. Preferences affect both push and in-app notifications.

7. Alert filtering works even when offline (based on last sync).

8. Users receive only alerts matching set preferences.

9. Changes in preferences are reflected instantly.

10. Preference options include checkboxes or toggles.

11. Alert history respects current preferences.

12. Preference setting is accessible via settings menu.

13. Custom preferences persist across app updates.

14. Invalid preference formats are not accepted.

15. Users can reset preferences to default.

Planning

Story Points

Priority2

Risk

Classification

Value areaBusiness

Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Development

Add link

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

Related Work

Add link

[Add an existing work item as a parent](#)

USER STORY 6

6 As a user, I want to receive instant notifications for extreme weather conditions so that I can take necessary precautions.

SREE VARSSINI K S0 CommentsAdd Tag

SaveFollowSettingsShareLink

Updated by SREE VARSSINI K S: Apr 17

Details🔍🔄🔗📄

StatgNewAreaWeather AppReasonNewIterationWeather AppUSER STORY

Description

This user story focuses on enhancing user safety by providing real-time notifications for extreme weather conditions such as thunderstorms, hurricanes, heatwaves, heavy rainfall, or snowstorms. The primary goal is to ensure users are immediately informed when potentially hazardous weather is detected in their area. These alerts will be triggered based on data received from trusted weather data sources and pushed to the user through notifications on the app. Timely alerts empower users to take prompt action, such as staying indoors, securing property, or adjusting travel plans, thereby reducing the risk of harm or inconvenience. This feature is critical for improving the app's reliability and value, especially during emergencies.

Acceptance Criteria

1. The system must fetch extreme weather alerts in real-time from an official weather API.

2. Users should receive push notifications for severe weather alerts within 10 seconds of an update.

3. The notification must contain alert details, including type (storm, flood), severity, location, and expected impact.

4. Users must be able to customize the type and severity of alerts they receive.

5. If the user dismisses an alert, it should not reappear unless updated by the weather source.

6. A history of the last 5 emergency alerts should be accessible in the app.

7. Alerts should be color-coded based on severity (e.g., red for high, orange for medium, yellow for low).

8. If the alert is for the user's current location, a warning icon should be displayed on the home screen.

9. The system should automatically update and remove expired alerts.

Planning

Story Points

Priority2

Risk

Classification

Value areaBusiness

Deployment

To track releases associated with this work item, go to [Releases](#) and turn on deployment status reporting for Boards in your pipeline's Options menu. [Learn more about deployment status reporting](#)

Development

Add link

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

Related Work

Add link

Parent

10 Emergency Alerts

Updated Apr 17 @ New

EPIC: Emergency Alert Notification

The Emergency Alerts feature in the weather app is designed to provide users with instant notifications about extreme weather conditions, ensuring they can take necessary precautions. This system is crucial for safety and preparedness, especially in cases of severe storms, hurricanes, floods, heatwaves, or snowfall. By leveraging real-time weather data from reliable sources, the app ensures that users are always informed about potentially hazardous weather conditions.

This epic consists of two major components:

1. **Real-Time Weather Alert System** – Automatically fetches and delivers extreme weather notifications.
2. **Customizable Alert Preferences** – Allows users to personalize which alerts they receive based on alert type and severity.

FEATURES:

The **Emergency Alerts** feature in the weather app ensures that users receive instant notifications regarding extreme weather conditions to take necessary precautions. Below are the detailed features categorized based on functionality.

1. Real-Time Weather Alert System : The app fetches severe weather alerts (storms, hurricanes, floods, heatwaves, etc.) from trusted weather APIs in real-time and instantly notifies users.

Key Functionalities:

- The system should integrate with weather APIs such as OpenWeather, AccuWeather, or NOAA for fetching real-time alerts.
- Alerts should be updated automatically every 5 minutes to ensure up-to-date information.
- Users should receive push notifications for alerts that match their selected preferences (severity, type, location).
- The system should process geolocation-based alerts to notify users based on their current or saved locations.
- Alerts should contain detailed information, including:
 - Alert Type (Storm, Hurricane, Heatwave, Flood, etc.)
 - Severity Level (High, Medium, Low)
 - Affected Region
 - Expected Duration
 - Precautionary Measures

2. Customizable Alert Preferences: Users can personalize their alert settings based on severity level, location, and specific weather conditions.

Key Functionalities:

- Users can enable/disable specific alert categories (e.g., only receive storm warnings, but not heatwaves).
- Users can select severity levels (High, Medium, Low) for filtering alerts.
- The system should allow users to set notifications for multiple locations (Home, Work, Travel Destinations).
- Multi-language support for alerts (English, Spanish, French, etc.).
- Admins can manually override alert settings to push critical emergency notifications.

3. Instant and Persistent Notifications: Users receive instant notifications for extreme weather conditions, with color-coded alerts based on severity.

Key Functionalities:

- Push Notifications: Alerts should be received within 10 seconds of an update.
- Color-coded alert system:
 - ☐ **Red** → High severity (Immediate danger: Tornado, Hurricane)
 - ☐ **Orange** → Medium severity (Potential risk: Heavy Rainfall, Snowstorm)
 - ☐ **Yellow** → Low severity (Mild disturbances: Light Rain, Windy)
- If an alert is dismissed, it will not reappear unless the alert information is updated.
- Urgent Sound Notification: If an alert is critical, the app should have a default emergency alarm enabled.
- Notifications should be sent even when the app is running in the background.

4. Alert History & Auto-Update System: Users can access past alerts and the system will auto-remove expired alerts.

Key Functionalities:

- A history log of the last 5 alerts should be available for reference.
- Expired alerts should be automatically removed from the system.
- The home screen should display an alert icon if an active severe alert exists for the user's location.
- Alerts should include recommended safety actions, such as evacuation instructions or shelter locations.
- Admin functionality: In case of API failure, admins should be able to manually push emergency alerts.

5. Multi-Platform & Accessibility Features: The emergency alerts system should be accessible across devices and user-friendly for all users.

Key Functionalities:

- Alerts should be synced across multiple devices (if the user logs in on different phones/tablets).
- Alerts should be voice-supported for visually impaired users.
- Wearable device support (smartwatches should receive emergency alerts).
- Alerts should be visible as pop-ups if the user is actively using the app.
- Offline Mode: The last fetched weather alerts should remain accessible even without an internet connection.

USER STORY 1:

As a user, I want to receive instant notifications for extreme weather conditions so that I can take necessary precautions.

Acceptance Criteria:

1. The system must fetch extreme weather alerts in real-time from an official weather API.
2. Users should receive push notifications for severe weather alerts within 10 seconds of an update.
3. The notification must contain alert details, including type (storm, flood), severity, location, and expected impact.
4. Users must be able to customize the type and severity of alerts they receive.
5. If the user dismisses an alert, it should not reappear unless updated by the weather source.
6. A history of the last 5 emergency alerts should be accessible in the app.
7. Alerts should be color-coded based on severity (e.g., red for high, orange for medium, yellow for low).
8. If the alert is for the user's current location, a warning icon should be displayed on the home screen.
9. The system should automatically update and remove expired alerts.
10. Emergency alerts should include recommended safety actions.
11. If an alert is critical, the app should have an urgent sound notification enabled by default.
12. Users should receive alerts even when the app is running in the background.
13. The system should support multiple languages for alerts.
14. Admins should be able to manually push emergency alerts in case of API failure.

USER STORY 2:

As a user, I want to customize alert preferences for different weather events (storms, heavy rain, heatwaves, etc.) so that I receive only relevant warnings.

Acceptance criteria:

1. Users can enable/disable individual weather event types.

2. Users can select severity levels (High, Medium, Low).
3. Preferences are saved after app restart.
4. Users can change preferences at any time.
5. Default preference includes all alert types.
6. Preferences affect both push and in-app notifications.
7. Alert filtering works even when offline (based on last sync).
8. Users receive only alerts matching set preferences.
9. Changes in preferences are reflected instantly.
10. Preference options include checkboxes or toggles.
11. Alert history respects current preferences.
12. Preference setting is accessible via settings menu.
13. Custom preferences persist across app updates.
14. Invalid preference formats are not accepted.
15. Users can reset preferences to default.

USER STORY 3:

As a user, I want the app to provide safety tips along with emergency alerts so that I know how to respond to hazardous weather conditions.

Acceptance Criteria:

1. Each alert includes a “View Safety Tips” link.
2. Tips are tailored to the alert type (e.g., earthquake vs. heatwave).
3. Safety tips are stored locally for offline access.
4. Tips include icons/images for better understanding.
5. Tips are written in bullet points for clarity.
6. Tips can be bookmarked for quick access.
7. The app supports safety tip translation to user language.
8. Tips are reviewed and updated regularly by admins.
9. Tips open in a new section/popup within the app.
10. Tips are less than 300 words per alert.
11. Each tip page has a share option.
12. Users can give feedback on tip usefulness.
13. App allows offline viewing of previously viewed tips.
14. Tips follow national emergency guidelines.
15. Tips are not shown if not relevant to the alert.

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

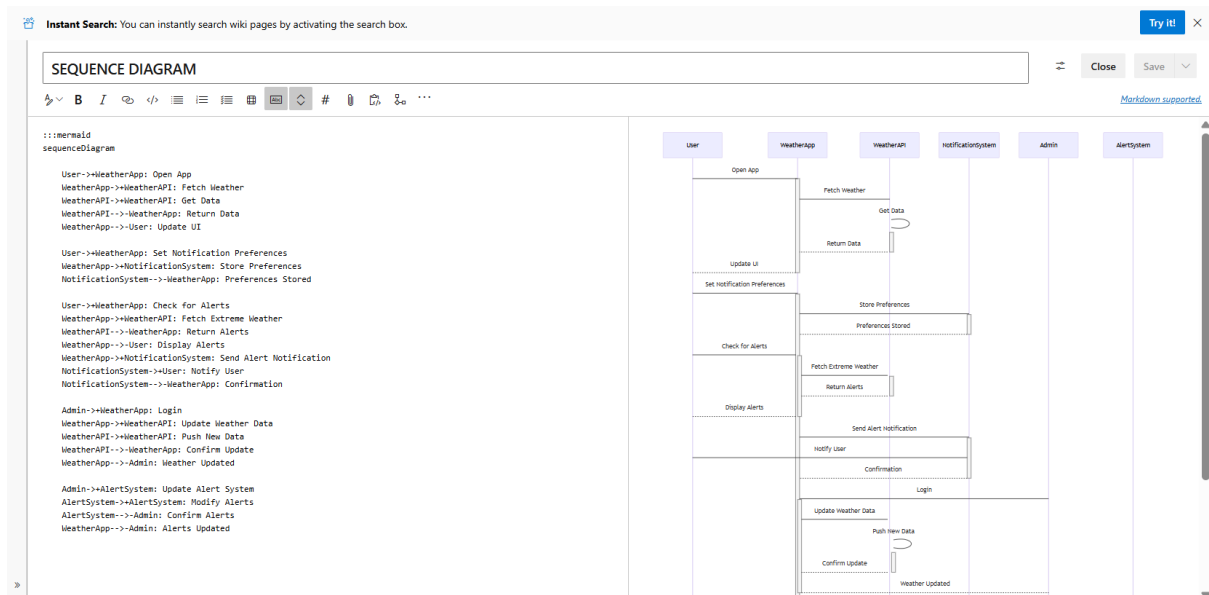
WeatherApp-->-Admin: Weather Updated

Admin->+AlertSystem: Update Alert System

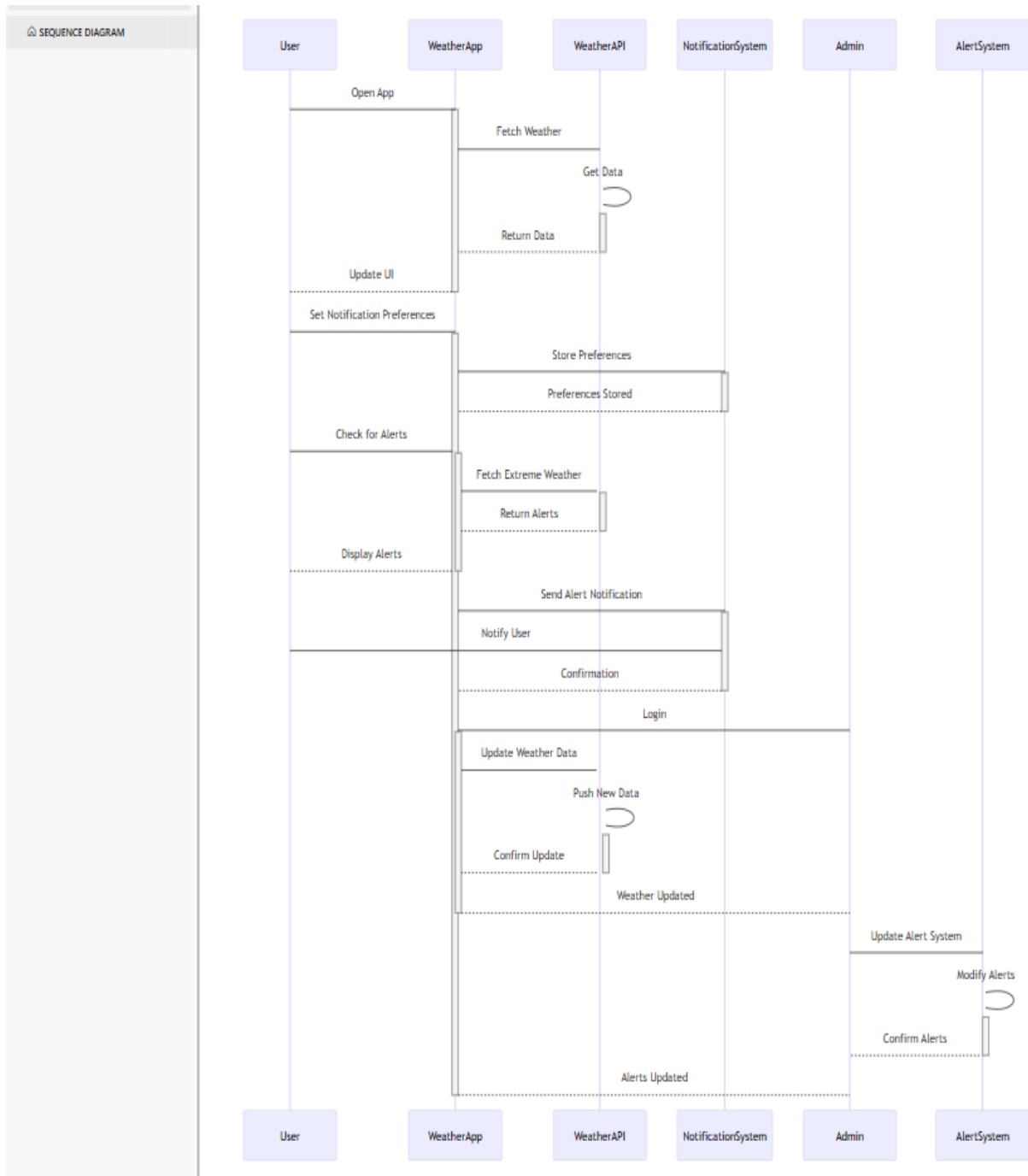
AlertSystem->+AlertSystem: Modify Alerts

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

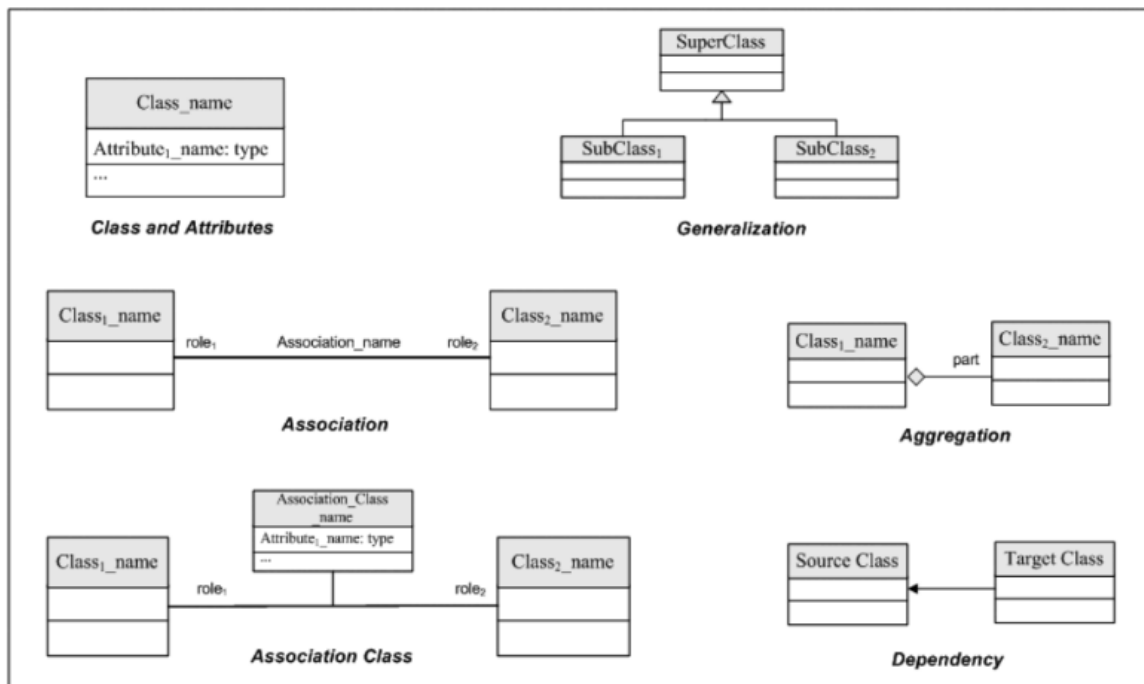
CLASS DIAGRAM

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
```

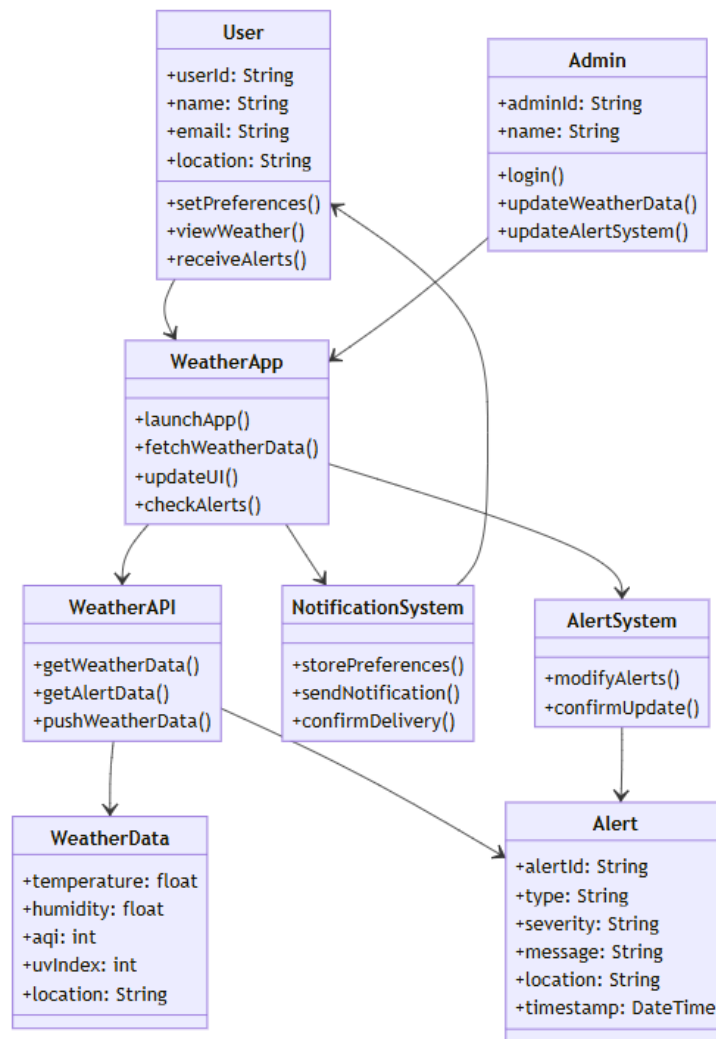


Weather-App.wiki

Enter page title

SEQUENCE DIAGRAM

CLASS DIAGRAM



RESULT:

Thus the class diagram has been designed successfully.

EX NO: 7

USE CASE DIAGRAM

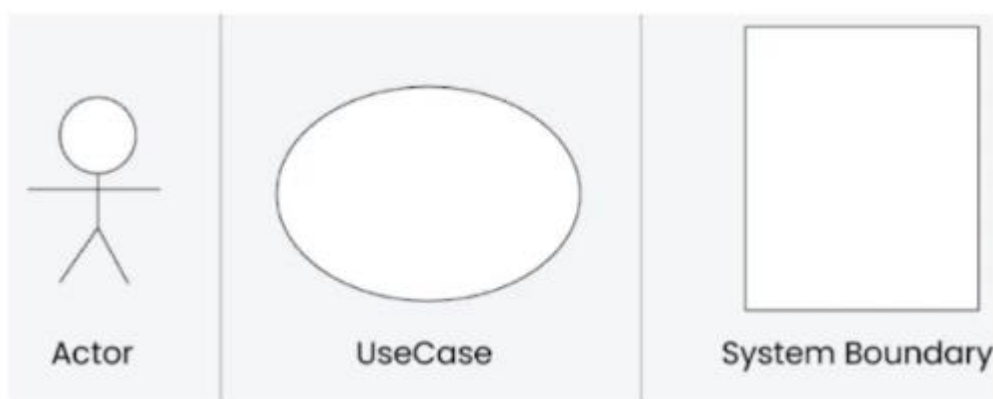
AIM:

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.

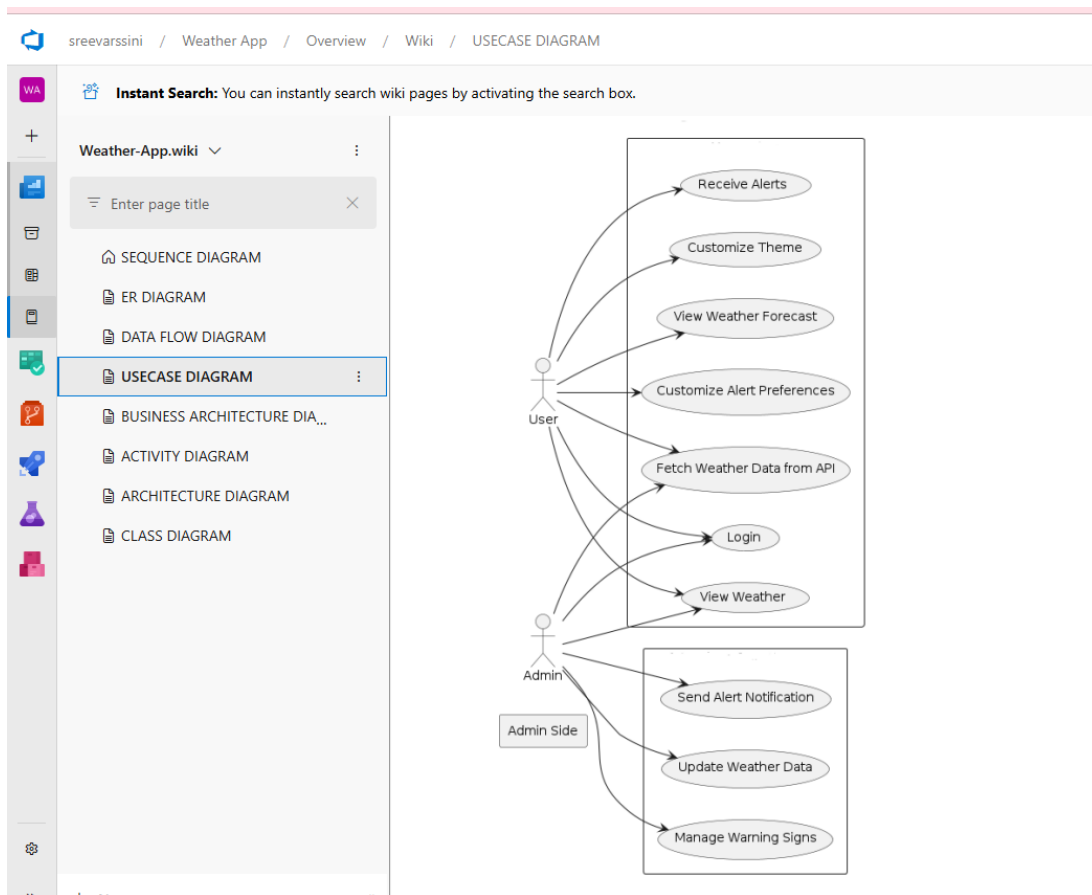
Step 2: Upload the Diagram to Azure DevOps

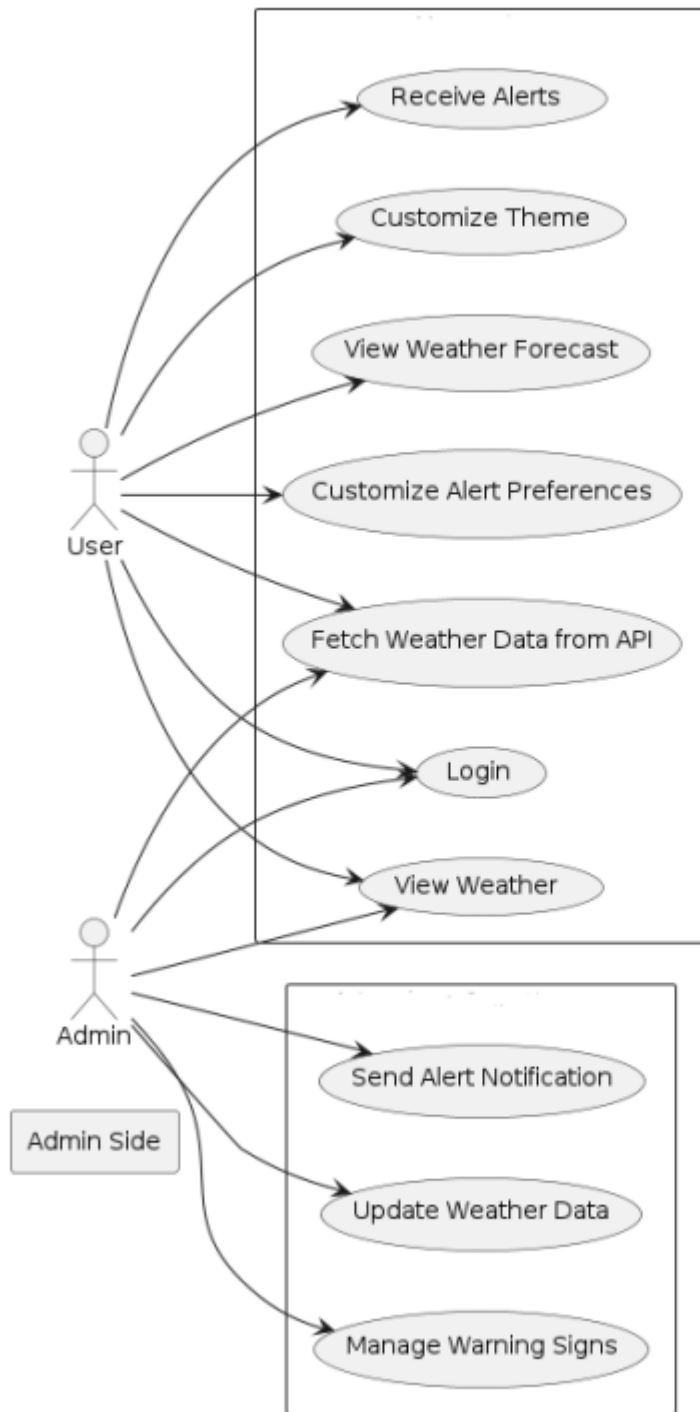
Option 1: Add to Azure DevOps Wiki

- 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



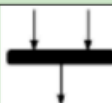








ACTIVITY DIAGRAM

AIM :

To draw a sample activity diagram for the Weather Application.

THEORY:

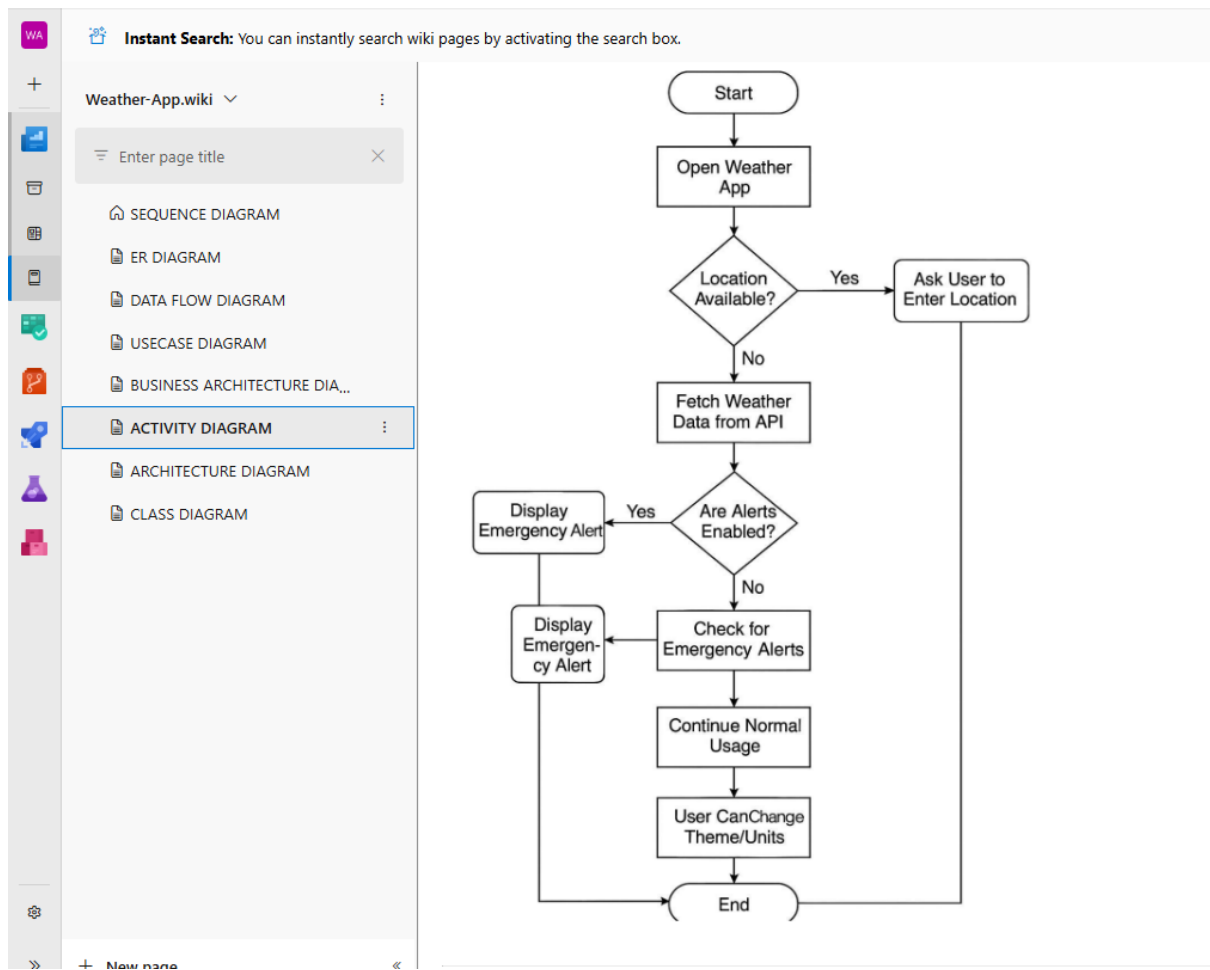
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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.

Weather-App.wiki

Enter page title

SEQUENCE DIAGRAM

ER DIAGRAM

DATA FLOW DIAGRAM

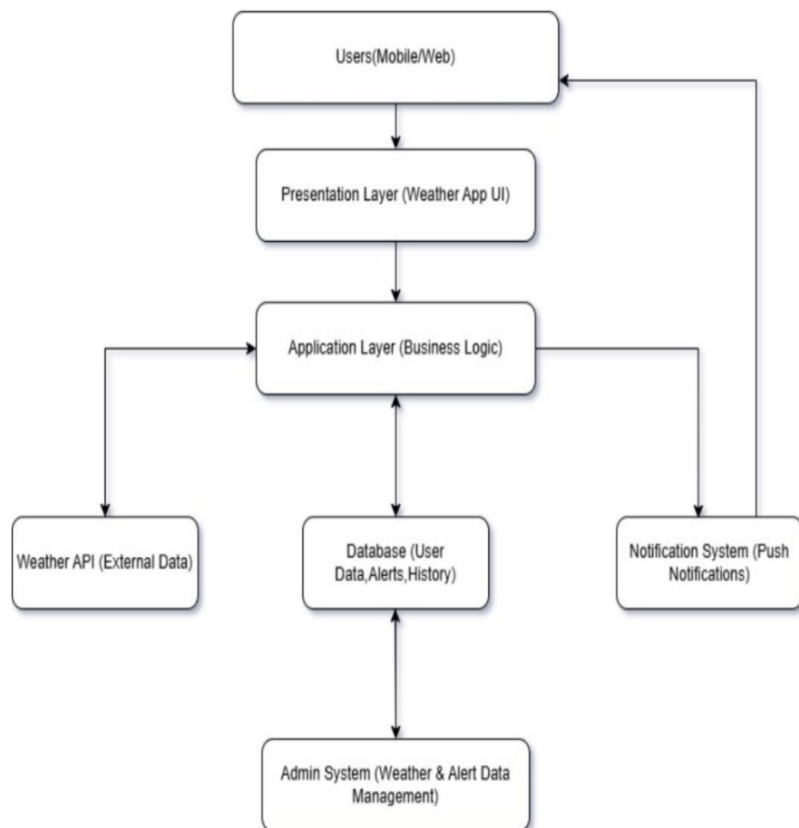
USECASE DIAGRAM

BUSINESS ARCHITECTURE DIA...

ACTIVITY DIAGRAM

ARCHITECTURE DIAGRAM

CLASS DIAGRAM



RESULT:

The architecture diagram was designed successfully

EX NO: 10

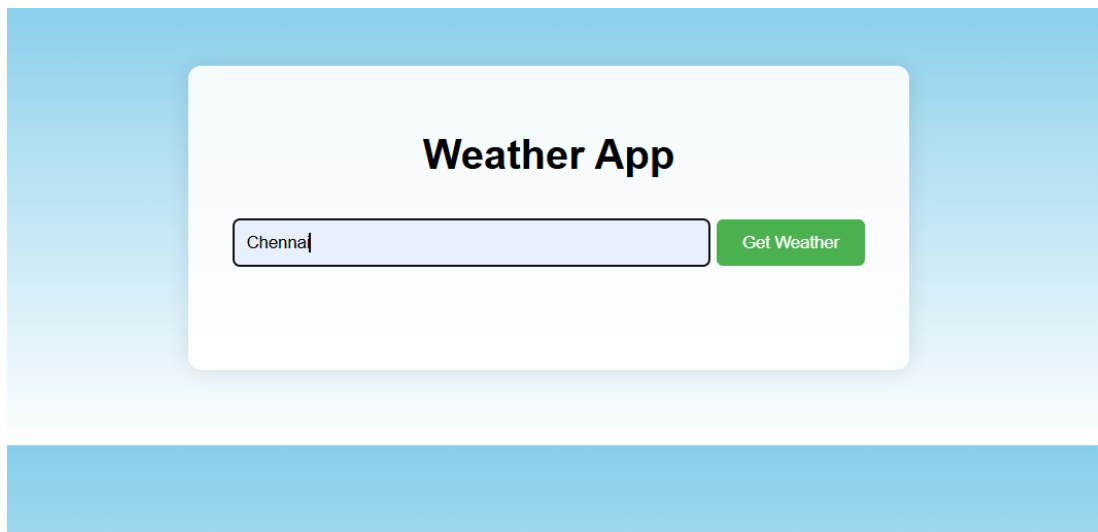
USER INTERFACE

AIM:

Design User Interface for the Weather App.

UI DESIGNS OF WEATHER APP:

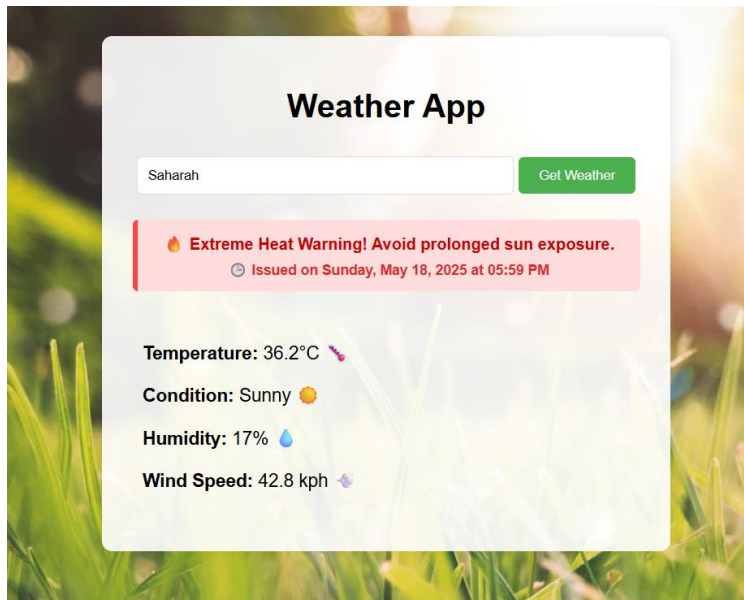
HOME PAGE:



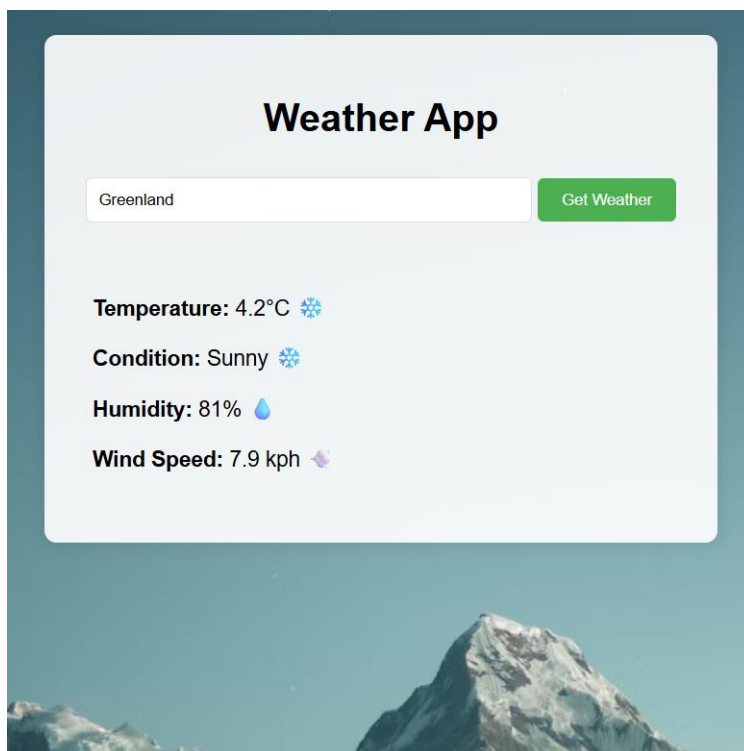
WEATHER DISPLAY:



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)

S sreevarssini

New organization

sreevarssini

Projects

My work items

My pull requests

WA

Weather App

A weather app is a digital application designed to provide real-time weather information such as temperature, humidity, wind speed, precipitation, and air quality index (AQI) for a user's current location or...

WA Weather App

Overview

Boards

Repos

Files

Commits

Pushes

Branches

Tags

Pull requests

Advanced Security

Pipelines

Test Plans

230701332-CS23432-SC

Weather App Diagrams

azure-pipelines.yml

index.html

script.js

style.css

main

Type to find a file or folder...

Files

succeeded

Clone

Contents History

Name ↑	Last change	Commits
Weather App Diagrams	May 8	674ab4d0 Add files via upload 230701332-SREEVAR...
azure-pipelines.yml	5h ago	e09bb1ea Set up CI with Azure Pipelines SREE VARS...
index.html	Yesterday	2410b100 Add files via upload 230701332-SREEVAR...
script.js	Yesterday	2410b100 Add files via upload 230701332-SREEVAR...
style.css	Yesterday	2410b100 Add files via upload 230701332-SREEVAR...

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

WA Weather App

Overview

Boards

Repos

Pipelines

Test Plans

Test plans

Progress report

Parameters

Configurations

EMERGENCY ALER...

Mar 14 - Mar 21

Past

100% run, 100% passed. [View report](#)

Test Suites

Filter suites by name

EMERGENCY ALERTS (5)

EMERGENCY ALERTS (ID: 22)

Define Execute Chart

Test Cases (5 items)

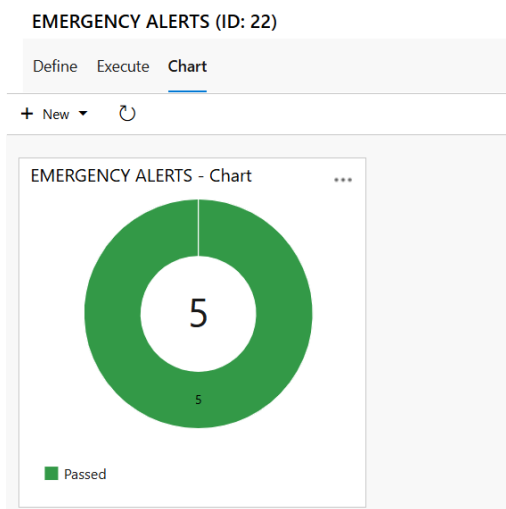
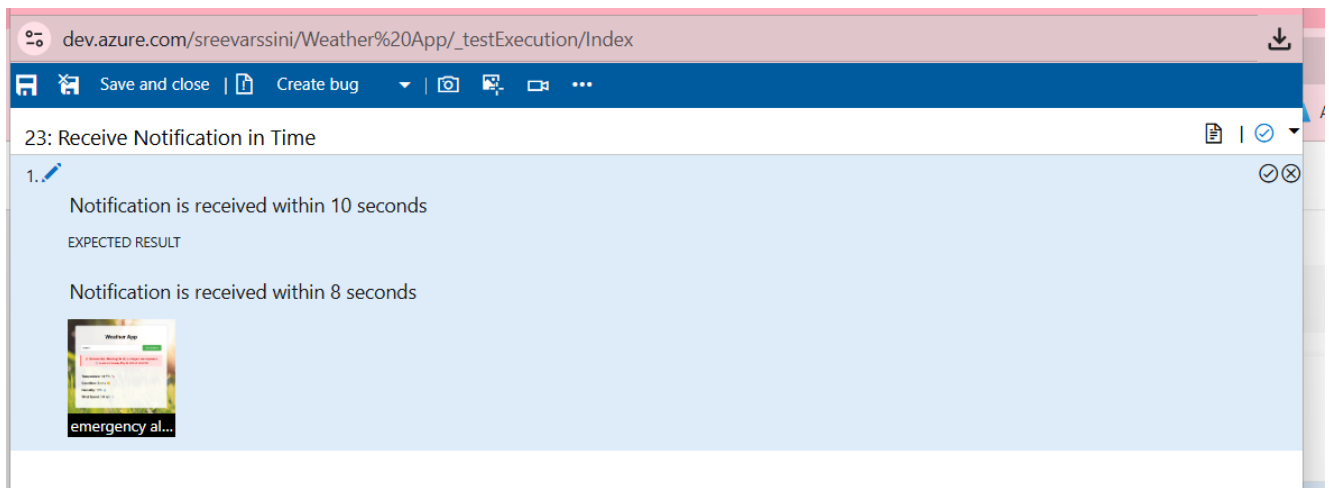
Title	Order	Test Case Id	Assigned To	Sta
Receive Notification in Time	1	23	SREE VARSSINI ... De	
Alert Contains Severity Level	2	24	SREE VARSSINI ... De	
Alert Color Coding	3	25	SREE VARSSINI ... De	
Alert Includes Event Type	4	26	SREE VARSSINI ... De	
Alert Shows Timestamp and Location	5	27	SREE VARSSINI ... De	

EMERGENCY ALERTS (ID: 22)

Define **Execute** Chart

Test Points (5 items)

<input type="checkbox"/> Title	Outcome	Order	Test Case Id	Configuration	Tester
<input type="checkbox"/> Receive Notification in Time	Passed	1	23	Windows 10	SREE VARSSINI ...
<input type="checkbox"/> Alert Contains Severity Level	Passed	2	24	Windows 10	SREE VARSSINI ...
<input type="checkbox"/> Alert Color Coding	Passed	3	25	Windows 10	SREE VARSSINI ...
<input type="checkbox"/> Alert Includes Event Type	Passed	4	26	Windows 10	SREE VARSSINI ...
<input type="checkbox"/> Alert Shows Timestamp and Location	Passed	5	27	Windows 10	SREE VARSSINI ...



Weather App

Overview Boards Repos Pipelines Test Plans Test plans Progress report Parameters Configurations

Test Plans

Mine All

Filter by title State Area Path Iteration Assigned To

Title	Test Plan ID	State	Area Path	Iteration	Assigned To	
INTERACTIVE UI	35	Active	Weather App	Weather App\USER STORY	SREYA G	
LIVE WEATHER UPDATES	28	Active	Weather App	Weather App	230701335	
EMERGENCY ALERTS	21	Active	Weather App	Weather App\USER STORY	SREE VARSSINI K S	

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)

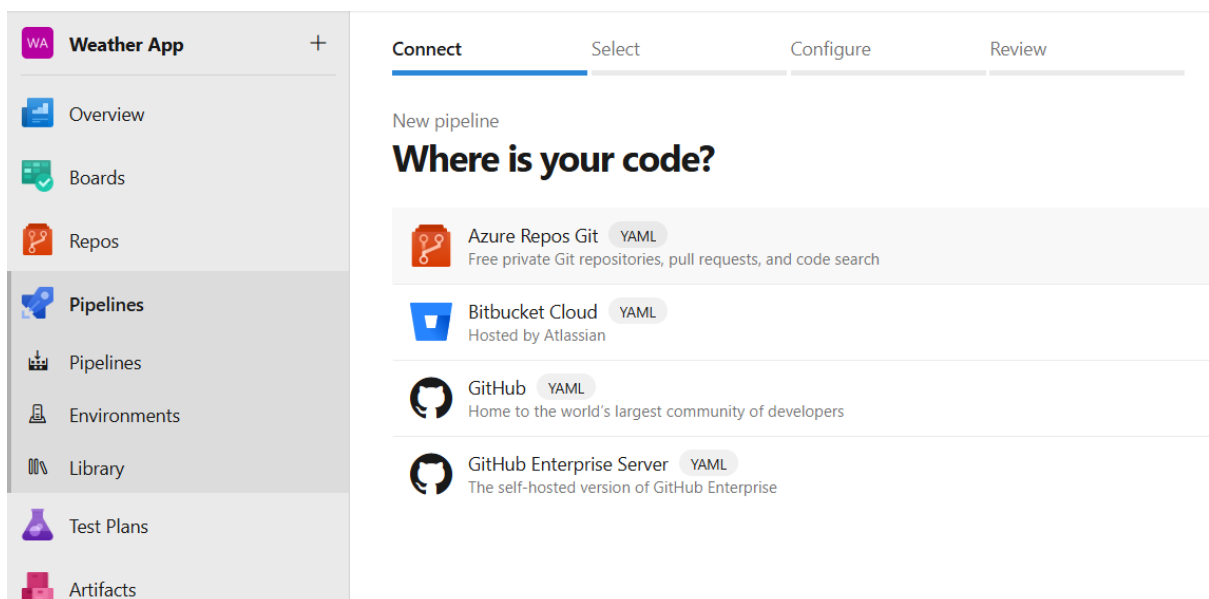
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

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.



WA

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

WA

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 *

```
1  # HTML
2  # Archive your static HTML project and save it with the build record.
3  # Add steps that build, run tests, deploy, and more:
4  # https://aka.ms/yaml
5
6  trigger:
7    - main
8
9  pool:
10   name: Default
11
12  steps:
13    Settings
14    - task: ArchiveFiles@2
15      inputs:
16        rootFolderOrFile: '$(build.sourcesDirectory)'
17        includeRootFolder: false
18    Settings
19    - task: PublishBuildArtifacts@1
```

WA Weather App

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Library

Test Plans

Artifacts

Project settings

#20250520.1 • Set up CI with Azure Pipelines

230701332-CS23432-SC

Run new

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

View retention leases

SummaryCode Coverage

Individual CI by SREE VARSSINI K S

View 10 changes

Repository and version

230701332-CS23432-SC

main

e09bb1ea

Time started and elapsed

Today at 1:09 PM

19s

Related

0 work items

1 published

Tests and coverage

Get started

Jobs

Name	Status	Duration
Job	Success	12s

6 As a user, I want to receive instant notifications for extreme weather conditions so that I can take necessary precautions.

SREE VARSSINI K S

0 CommentsAdd Tag

SaveFollow

Updated by SREE VARSSINI K S: 5h ago

Details

Description

This user story focuses on enhancing user safety by providing real-time notifications for extreme weather conditions such as thunderstorms, hurricanes, heatwaves, heavy rainfall, or snowstorms. The primary goal is to ensure users are immediately informed when potentially hazardous weather is detected in their area. These alerts will be triggered based on data received from trusted weather data sources and pushed to the user through notifications on the app. Timely alerts empower users to take prompt action, such as staying indoors, securing property, or adjusting travel plans, thereby reducing the risk of harm or inconvenience. This feature is critical for improving the app's reliability and value, especially during emergencies.

Acceptance Criteria

1. The system must fetch extreme weather alerts in real-time from an official weather API.

2. Users should receive push notifications for severe weather alerts within 10 seconds of an update.

3. The notification must contain alert details, including type (storm, flood), severity, location, and expected impact.

4. Users must be able to customize the type and severity of alerts they receive.

5. If the user dismisses an alert, it should not reappear unless updated by the weather source.

6. A history of the last 5 emergency alerts should be accessible in the app.

7. Alerts should be color-coded based on severity (e.g., red for high, orange for medium, yellow for low).

8. If the alert is for the user's current location, a warning icon should be displayed on the home screen.

9. The system should automatically update and remove expired alerts.

10. The system should provide a way for users to manage their alert preferences.

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 5h ago

Succeeded

Related Work

Add link

Add an existing work item as a parent

RESULT:

Thus the CI/CD pipeline has been successfully implemented.