

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 : SREYA G

Year/Branch/Section : II /CSE/ D

Register No. : 230701334

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:

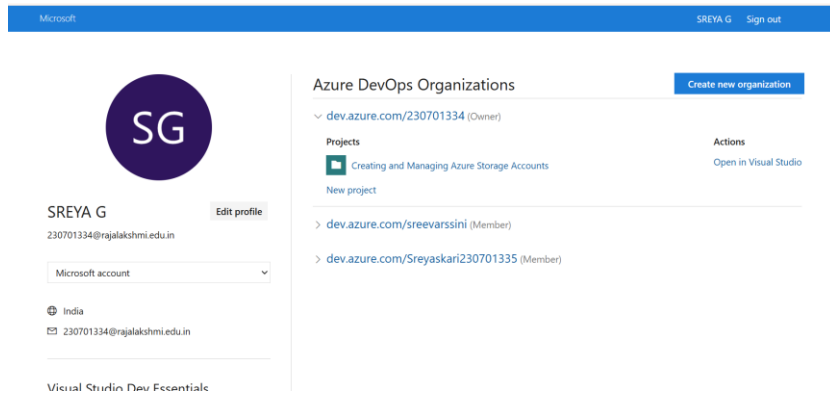
To create User Stories.

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.

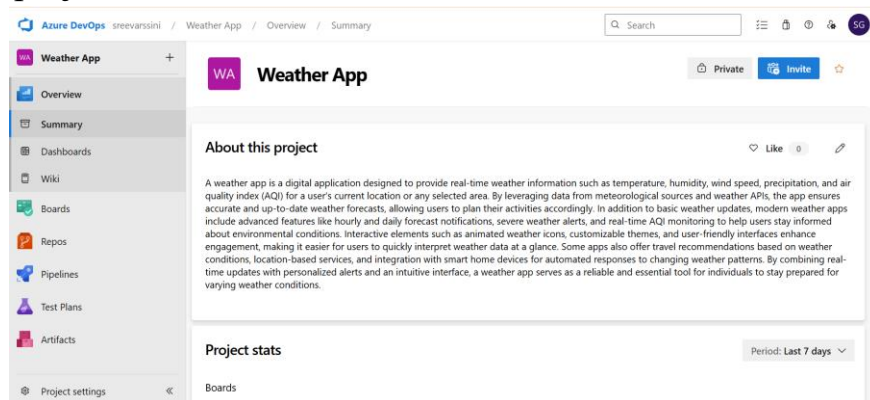
PROCEDURE:

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.

Weather App

Overview

Boards

Work Items

Boards

Backlogs

Sprints

Queries

Delivery Plans

Repos

Pipelines

Test Plans

Artifacts

Project settings

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

Recently updated

Back to Work Items

14

As a user, I want an easy-to-use dashboard that displays weather information in a visually appealing way so that I can understand it quickly.

0 Comments

Add Tag

Save

Follow

Details

Updated by STEPH G. Yesterday

Story

Name

Done

Weather App

Reason

None

None

Weather App

Description

Planning

Deployment

Development

Classification

Acceptance Criteria

Description

The goal of this feature is to create a clean and intuitive weather dashboard that presents key weather information at a glance. The dashboard will prioritize simplicity and visual clarity, ensuring that users can easily access current weather conditions, hourly and weekly forecasts, temperature, humidity, wind speed, and precipitation chances. Visual elements like icons, charts, and color-coded indicators will enhance readability and appeal. The interface will be responsive across devices and adaptable for both light and dark modes. This design will allow users of all ages and technical backgrounds to instantly interpret weather updates without needing to navigate through complex menus or text-heavy screens.

Acceptance Criteria

- Home screen shows current weather prominently.
- Key data is grouped (Temperature, AQI, Forecast).
- Design uses intuitive icons for readability.
- Fonts are legible and well-spaced.
- UI adapts to screen size and orientation.
- UI supports accessibility (colorblind-friendly).
- User can manage dashboard widgets.
- Smooth transitions when switching views.
- Home page loads in less than 2 seconds.
- App uses minimalist design (no clutter).
- Clear visual separation between sections.
- Color theme reflects current weather (e.g., gray for rain).
- Forecast lists are scrollable for next days.
- Scrollable sections for related data.
- All important data visible without scrolling.

Planning

Story Points

Priority

2

Risk

Low

Classification

View area

Business

Deployment

To track releases associated with this work item, go to [Repos](#) 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 20270102-C20440-SC_20250201

Updated Yesterday

Succeeded

Related Work

Add link

Parent

13 Interactive UI

Updated Apr 17 @ New

Weather App

Overview

Boards

Work Items

Boards

Backlogs

Sprints

Queries

Delivery Plans

Repos

Pipelines

Test Plans

Artifacts

Project settings

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

Recently updated

Back to Work Items

16

As a user, I want the app to include animated weather icons (rain, snow, sunny, etc.) so that I can instantly recognize the weather conditions.

0 Comments

Add Tag

Save

Follow

Details

Updated by STEPH G. Yesterday

Story

Name

Done

Weather App

Reason

None

None

Weather App

Description

Planning

Deployment

Development

Classification

Acceptance Criteria

Description

This feature focuses on enhancing the user experience by integrating **animated weather icons** that visually represent current weather conditions in real-time. Instead of relying solely on text, the app will use dynamic animations (e.g., raindrops falling for rain, sun rays for sunny, snowflakes drifting for snow) to make the interface more engaging and immediately informative. These animations will serve as quick visual cues, allowing users to recognize the weather at a glance without reading detailed data. The icons will be lightweight to ensure fast loading and optimized for performance across all devices. This not only improves accessibility but also adds a modern and interactive feel to the overall UI.

Acceptance Criteria

- Animated icons reflect real-time weather data.
- Rain icon includes falling drops animation.
- Snow icon includes snowflakes drifting.
- Thunderstorms include lightning flashes.
- Sunny icon glows with animated rays.
- Cloudy icon shows moving clouds.
- Animation updates every 30 minutes.
- Users can disable animations in settings.
- Animations are GPU-optimized for performance.
- App uses fallback static icon if animation fails.
- All animations are under 1MB for download.
12. Animation speed adjustable via settings.
- Animations load instantly on launch.
14. Weather animation matches forecast when forecast is tapped.
15. Animation is consistent across all devices and platforms.

Planning

Story Points

Priority

2

Risk

Low

Classification

View area

Business

Deployment

To track releases associated with this work item, go to [Repos](#) 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 20270102-C20440-SC_20250201

Updated Yesterday

Succeeded

Related Work

Add link

Parent

15 Interactive UI

Updated Apr 17 @ New

Weather App

Overview

Boards

Work Items

Boards

Backlogs

Sprints

Queries

Delivery Plans

Repos

Pipelines

Test Plans

Artifacts

Project settings

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

Recently updated

Back to Work Items

15

As a user, I want to switch between different themes (light/dark mode) so that I can customize my viewing experience.

0 Comments

Add Tag

Save

Follow

Details

Updated by STEPH G. Yesterday

Story

Name

Done

Weather App

Reason

None

None

Weather App

Description

Planning

Deployment

Development

Classification

Acceptance Criteria

Description

This feature allows users to personalize the app interface by choosing between **light and dark themes** based on their preferences or lighting conditions. The theme toggle will be easily accessible from the settings or dashboard area, enabling seamless switching with a smooth transition effect. The light mode will offer a clear and bright interface ideal for daytime use, while the dark mode will reduce eye strain in low-light environments and help conserve battery life on OLED screens. All elements of the UI—including background, text, icons, and charts—will adapt accordingly to maintain readability and aesthetic consistency across both modes. This customization option enhances user comfort and ensures the app remains usable and visually pleasant in any environment.

Acceptance Criteria

- App supports manual Light/Dark theme toggle.
- Default theme is system-based (follows OS).
- Users can enable automatic switching by time.
- Themes apply to all screens uniformly.
- Theme setting is saved on device.
- Light theme uses white background and dark text.
- Dark theme uses dark gray background and white text.
- Theme affects widgets and notification panels too.
- Theme preview available in settings.
- UI elements adapt visibility per theme (e.g., icons).
- No flicker during theme switch.
12. Theme settings persist after app updates.
13. Theme doesn't affect data layout.
14. Background images adapt per theme.
16. Theme change triggers confirmation toast.

Planning

Story Points

Priority

2

Risk

Low

Classification

View area

Business

Deployment

To track releases associated with this work item, go to [Repos](#) 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 20270102-C20440-SC_20250201

Updated Yesterday

Succeeded

Related Work

Add link

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

EPIC : Interactive UI

This epic “Interactive UI” focuses on creating a modern, visually engaging interface that responds to real-time weather data and user preferences. It enhances user experience through dynamic visuals, smooth interactions, and customizable themes, making the app both intuitive and aesthetically pleasing.

The Epic consists of two major components :

1. **Animated Weather Icons** – Displays dynamic weather animations (rain, snow, sun, etc.) for instant visual recognition. Optimized for smooth performance with an option to disable animations.
2. **Theme Customization** – Users can switch between light, dark, and auto mode (auto adjusts based on time). Includes a transparent widget option and supports high-resolution displays.

FEATURES :

The **Interactive UI** enhances the user experience by making the weather app visually appealing, easy to use, and responsive. It emphasizes clarity, real-time feedback, and personalization through dynamic visuals and adaptable design, ensuring users can quickly understand weather updates at a glance.

1. **Animated Weather Icons** : Visually displays current weather through smooth, animated icons for an engaging and instantly understandable experience.

Key Functionalities:

- Real-time animations for rain, snow, sun, cloud, and storm conditions.
- Smooth GPU-optimized visuals for minimal battery usage.
- Animation updates every 30 minutes with live data sync.
- Option to disable animations via settings.
- Static fallback icons appear if animations fail to load.

2. **Theme Customization** : Lets users choose between light, dark, or auto themes to match preferences or ambient lighting.

Key Functionalities:

- Manual toggle for Light, Dark, and Auto themes.
- Auto mode adjusts theme based on time of day or OS setting.
- Uniform theme application across all screens and widgets.
- Smooth transition with no flicker between themes.
- Settings persist after app restart or update.

- 3. Responsive Layout Design :** Ensures the app's interface adapts seamlessly to different devices and screen sizes.

Key Functionalities:

- Auto-layout adapts to mobile, tablet, and desktop screens.
- Components resize and reposition based on screen orientation.
- Readable fonts and well-spaced elements for all screen sizes.
- Supports both portrait and landscape modes.
- Maintains consistent user experience across resolutions.

- 4. Widget Reordering & Personalization :** Enables users to customize their dashboard by reordering or selecting specific widgets.

Key Functionalities:

- Drag-and-drop support for rearranging dashboard widgets.
- Option to show/hide specific weather cards (e.g., wind, humidity).
- Personalized layout saved locally and restored on app launch.
- Visual feedback during widget rearrangement for clarity.
- Dynamic adjustment of layout based on widget priority.

USER STORY 1:

As a user, I want an easy-to-use dashboard that displays weather information in a visually appealing way so that I can understand it quickly.

Acceptance Criteria :

1. Home screen shows current weather prominently.
2. Key data is grouped (Temperature, AQI, Forecast).
3. Design uses intuitive icons for readability.
4. Fonts are legible and well-spaced.
5. UI adapts to screen size and orientation.
6. UI supports accessibility (colorblind-friendly).
7. User can rearrange dashboard widgets.
8. Smooth transitions when switching views.

9. Home page loads in less than 2 seconds.
- 10.App uses minimalistic design (no clutter).
- 11.Clear visual separation between sections.
- 12.Color theme reflects current weather (e.g., gray for rain).
- 13.Forecast tiles are swipeable for next days.
- 14.Scrollable sections for detailed data.
- 15.All important data visible without scrolling.

USER STORY 2 :

As a user, I want the app to include animated weather icons (rain, snow, sunny, etc.) so that I can instantly recognize the weather conditions.

Acceptance Criteria :

1. Animated icons reflect real-time weather data.
2. Rain icon includes falling drops animation.
3. Snow shows snowflakes drifting.
4. Thunderstorm includes lightning flashes.
5. Sunny icon glows with animated rays.
6. Cloudy icon shows moving clouds.
7. Animation updates every 30 minutes.
8. Users can disable animations in settings.
9. Animations are GPU-optimized for performance.
- 10.App uses fallback static icon if animation fails.
- 11.All animations are under 1MB for download.
- 12.Animation speed adjustable via settings.

13. Animations load instantly on launch.

14. Weather animation matches forecast when forecast is tapped.

15. Animation is consistent across all devices and platforms.

USER STORY 3 :

As a user, I want to switch between different themes (light/dark mode) so that I can customize my viewing experience.

Acceptance Criteria :

1. App supports manual Light/Dark theme toggle.
2. Default theme is system-based (follows OS).
3. Users can enable automatic switching by time.
4. Themes apply to all screens uniformly.
5. Theme setting is saved on device.
6. Light theme uses white background and black text.
7. Dark theme uses dark gray background and white text.
8. Theme affects widgets and notification panels too.
9. Theme preview available in settings.
10. UI elements adjust visibility per theme (e.g., icons).
11. No flicker during theme switch.
12. Theme settings persist after app updates.
13. Theme doesn't affect data layout.
14. Background images adjust per theme.
15. Theme change triggers confirmation toast.

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.

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

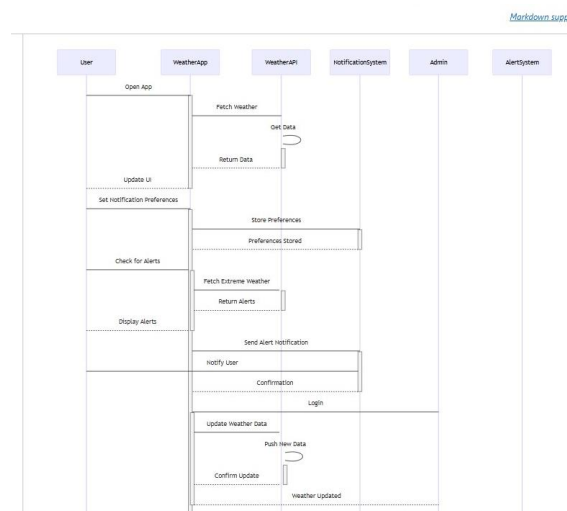
WeatherApp-->-Admin: Weather Updated

Admin->+AlertSystem: Update Alert System

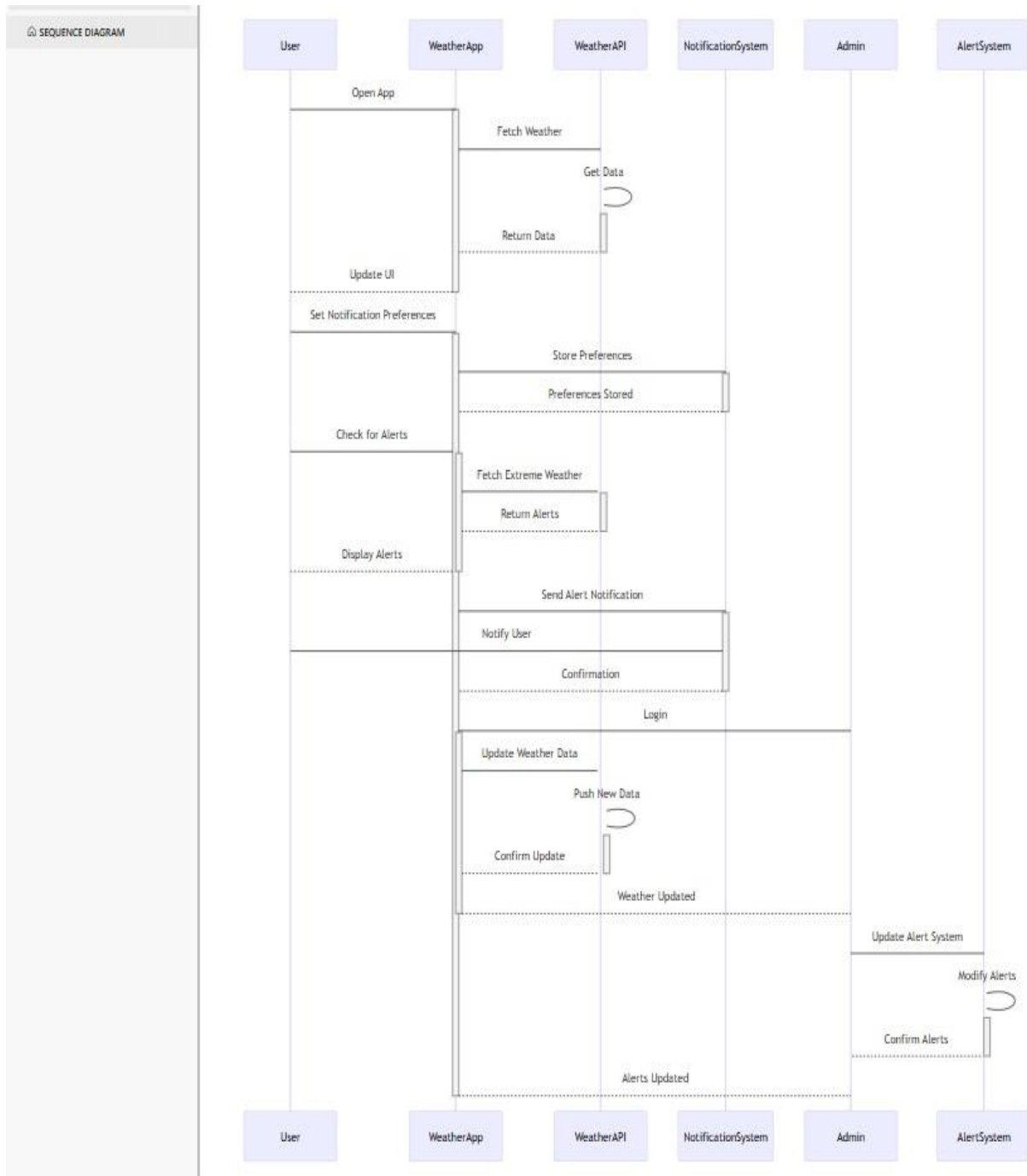
AlertSystem->+AlertSystem: Modify Alerts

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

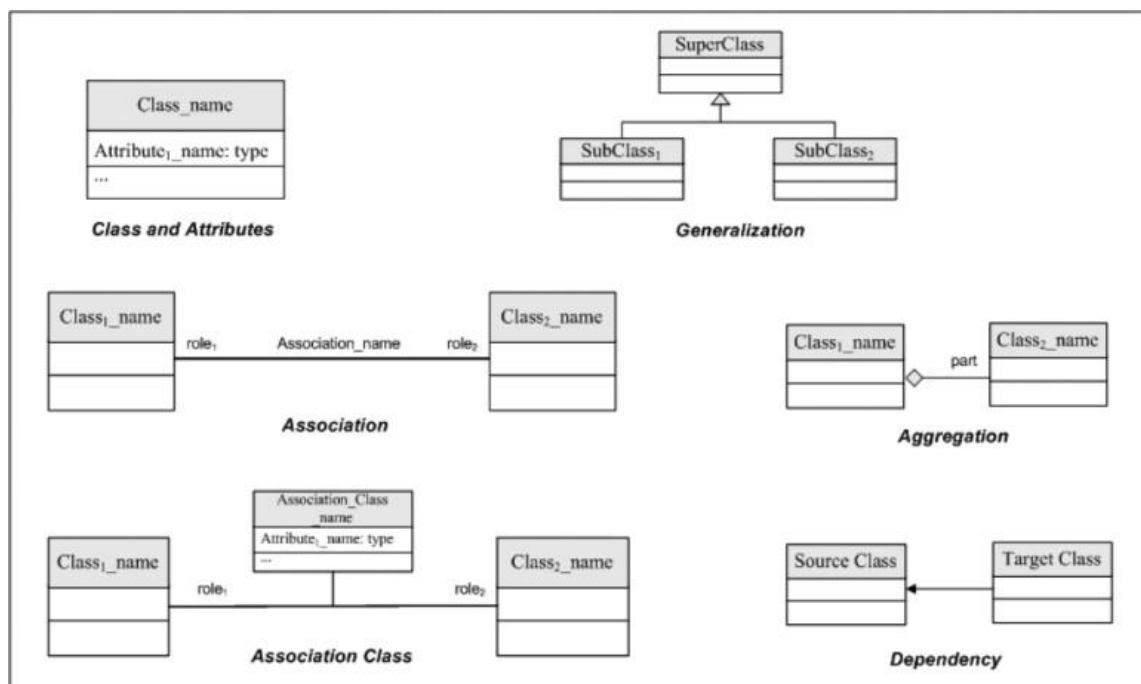
CLASS DIAGRAM

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.

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()
}

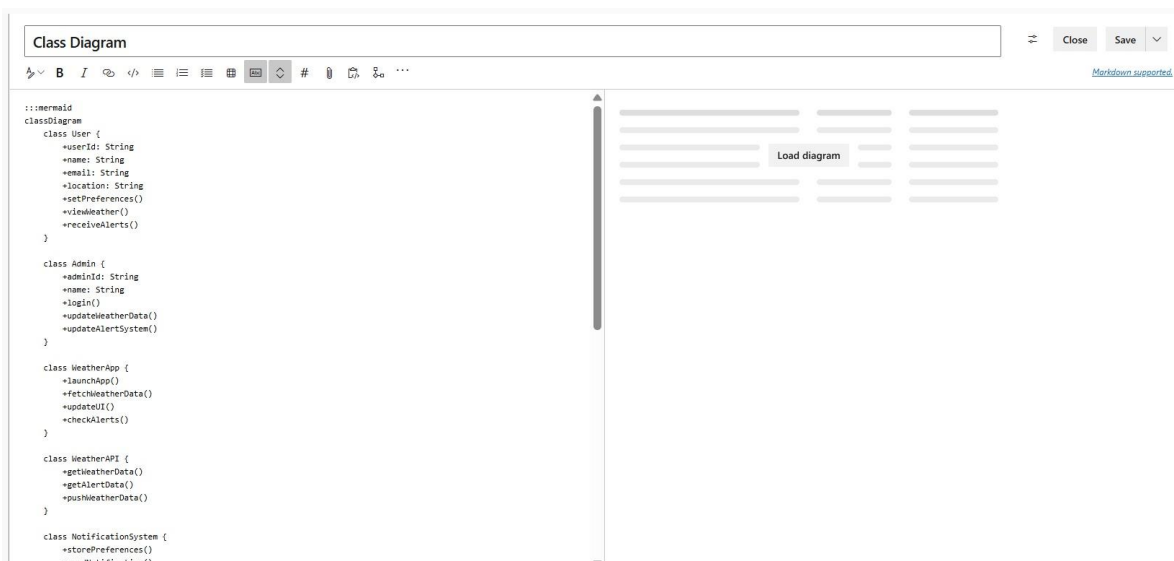
class NotificationSystem {
  +storePreferences()
  +sendNotification()
  +confirmDelivery()
}

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
```

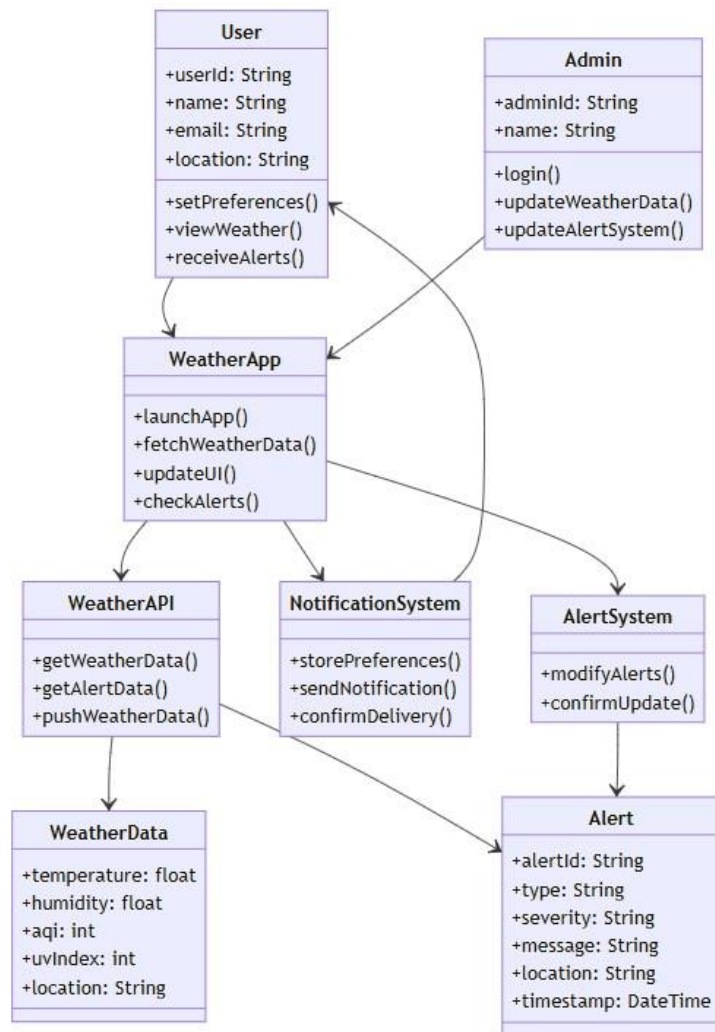


Weather-App.wiki

Enter page title

SEQUENCE DIAGRAM

CLASS DIAGRAM



RESULT:

Thus the class diagram has been designed successfully.

EX NO: 7

USE CASE DIAGRAM

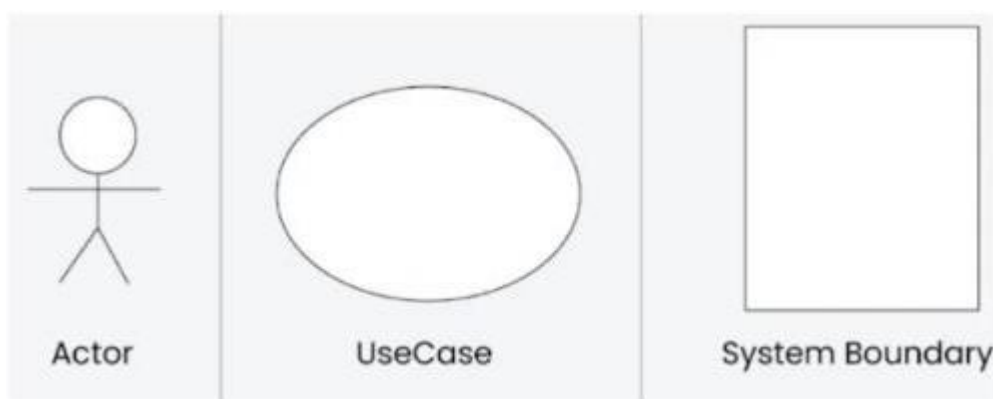
AIM:

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.

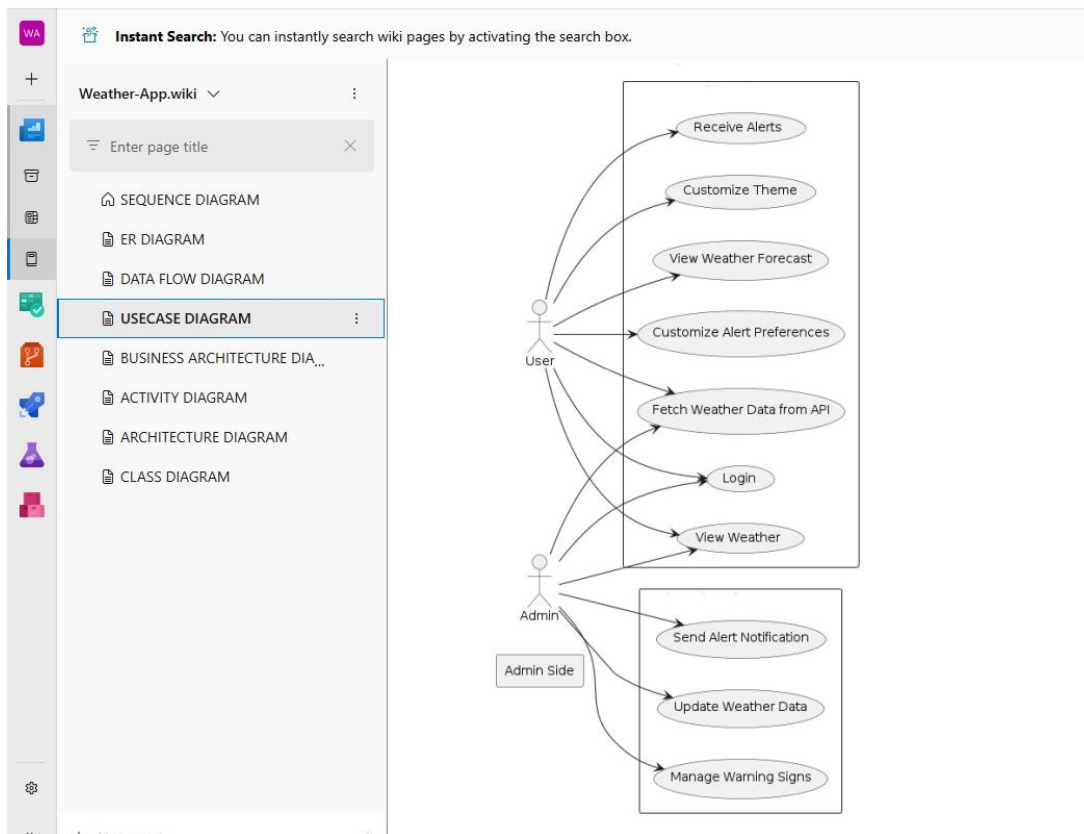
Step 2: Upload the Diagram to Azure DevOps

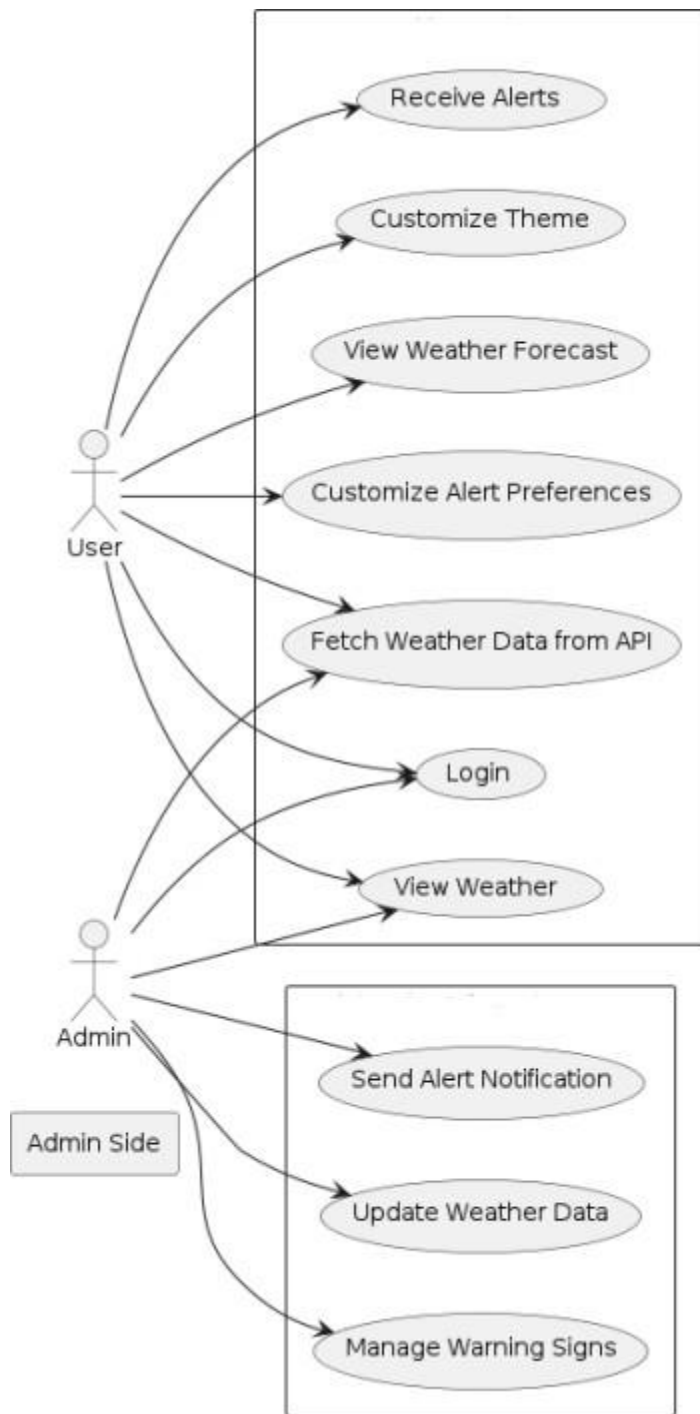
Option 1: Add to Azure DevOps Wiki

- 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



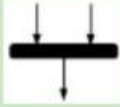








ACTIVITY DIAGRAM

AIM :

To draw a sample activity diagram for the Weather Application.

THEORY:

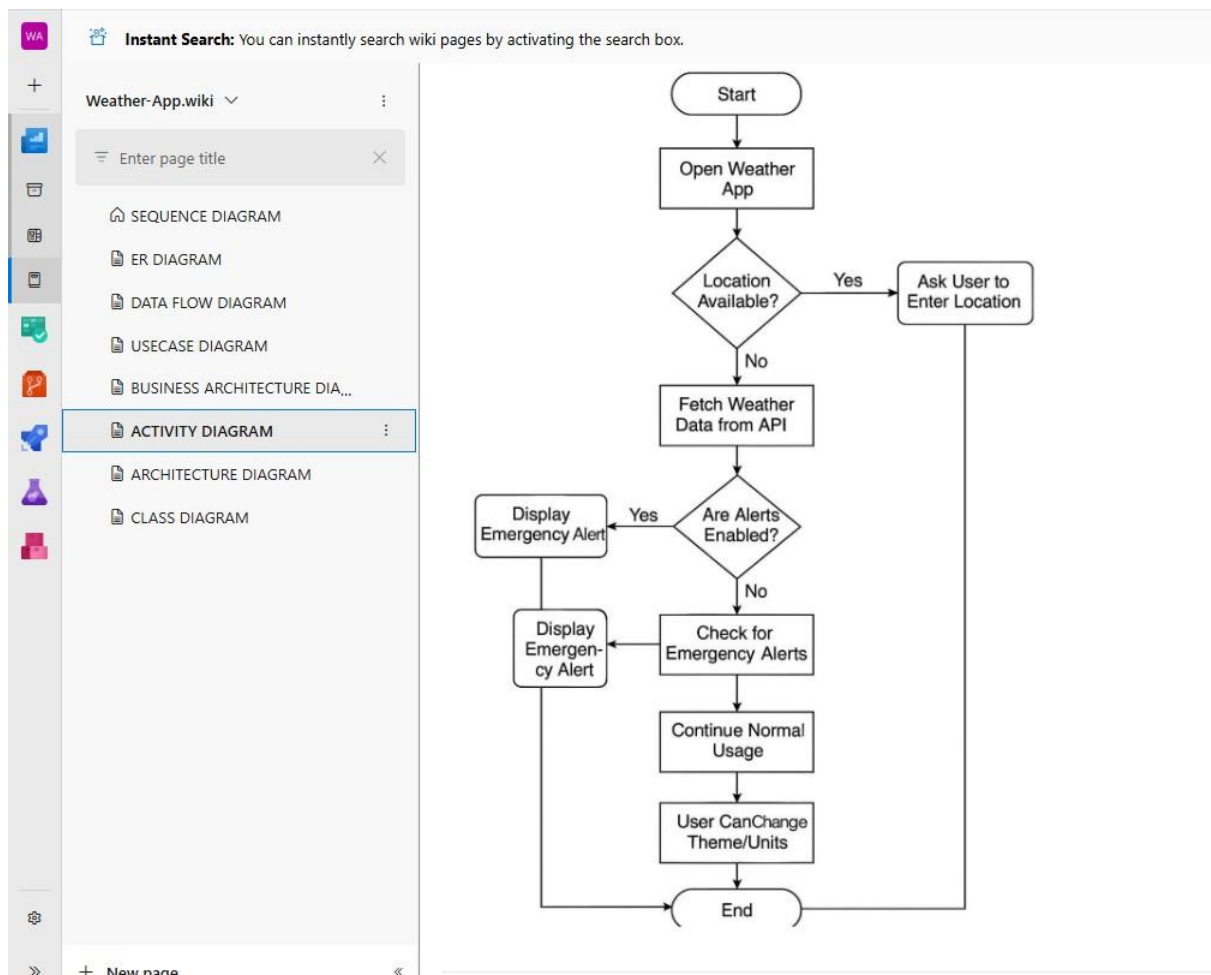
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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 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:

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:

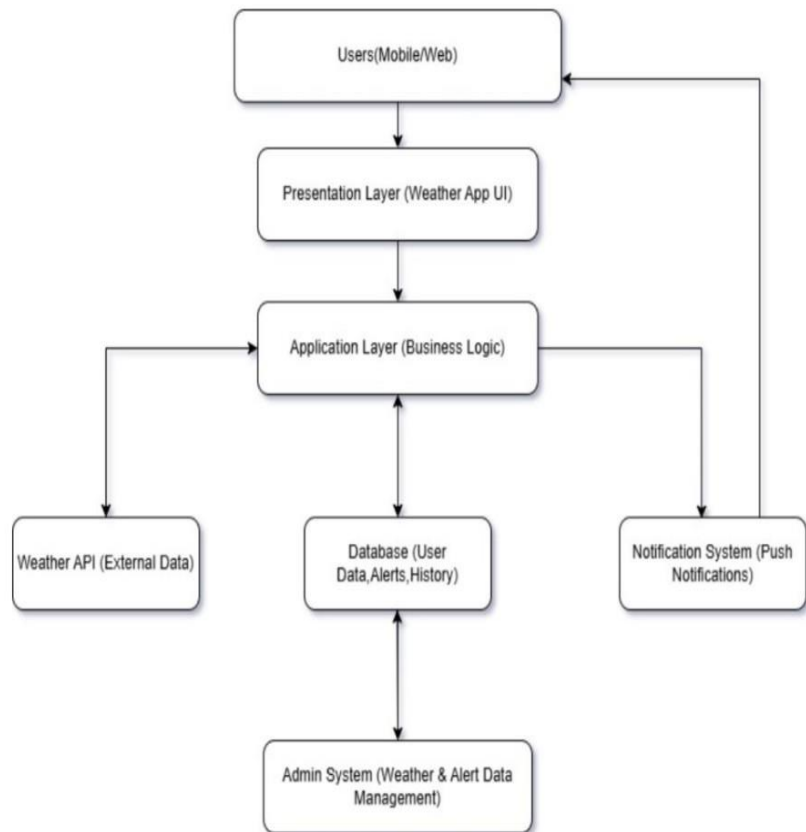
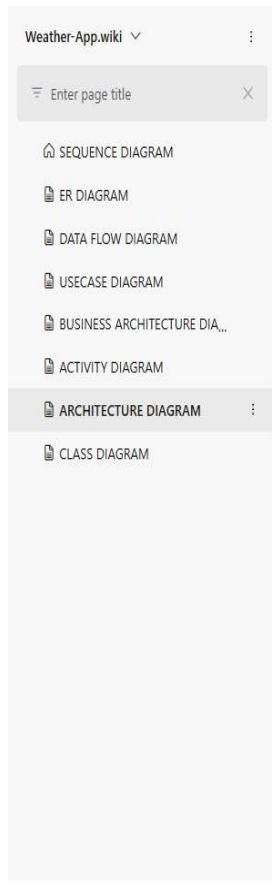
An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.



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

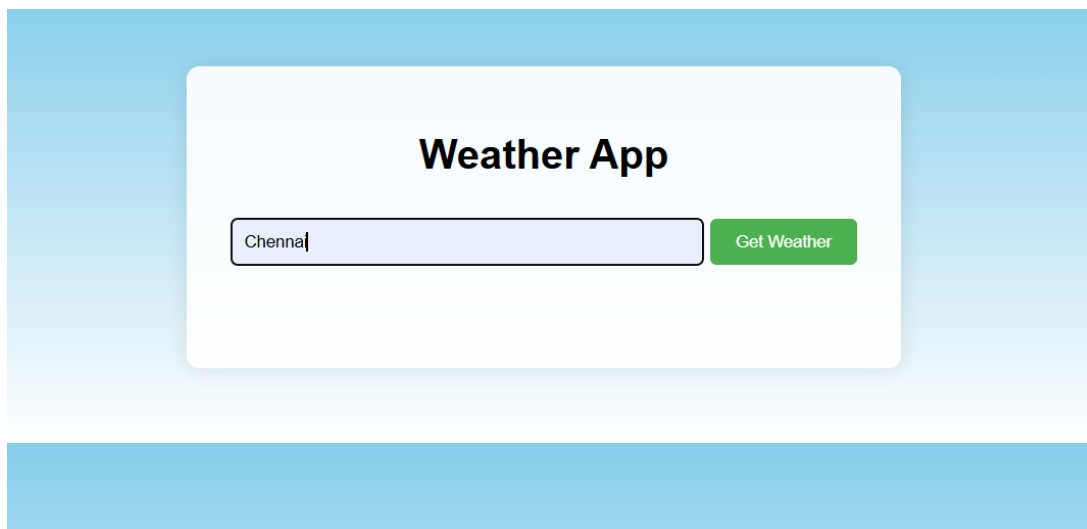
USER INTERFACE

AIM:

Design User Interface for the Weather App.

UI DESIGNS OF WEATHER APP:

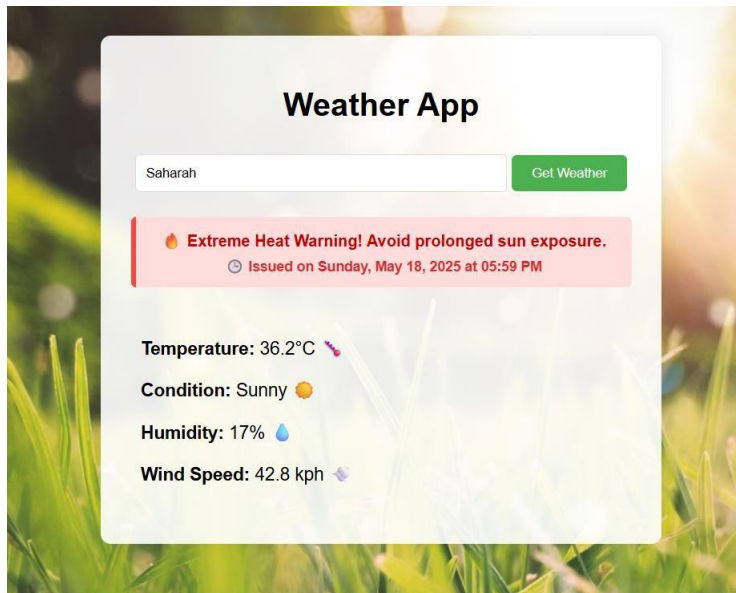
HOME PAGE:



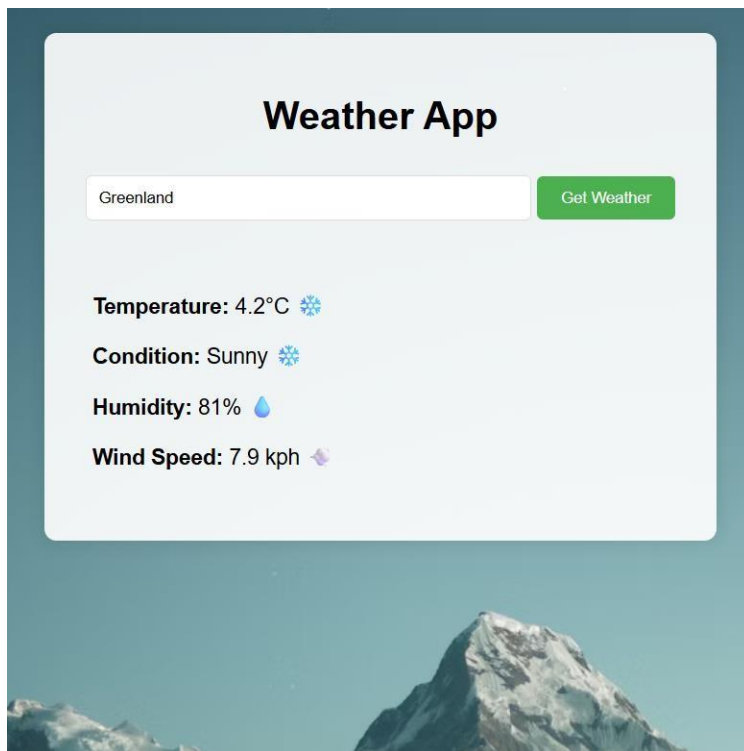
WEATHER DISPLAY:



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)



About this project

Like 0



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 any selected area. By leveraging data from meteorological sources and weather APIs, the app ensures accurate and up-to-date weather forecasts, allowing users to plan their activities accordingly. In addition to basic weather updates, modern weather apps include advanced features like hourly and daily forecast notifications, severe weather alerts, and real-time AQI monitoring to help users stay informed about environmental conditions. Interactive elements such as animated weather icons, customizable themes, and user-friendly interfaces enhance engagement, making it easier for users to quickly interpret weather data at a glance. Some apps also offer travel recommendations based on weather conditions, location-based services, and integration with smart home devices for automated responses to changing weather patterns. By combining real-time updates with personalized alerts and an intuitive interface, a weather app serves as a reliable and essential tool for individuals to stay prepared for varying weather conditions.

Project stats

Period: Last 7 days

Boards

14 Work items created

0 Work items completed

Repos

2 Pull requests opened

16 Commits by 3 authors

Pipelines



Builds succeeded

Members

4



CS23432-SC-230701334

index.html
script.js
style.css

main / Type to find a file or folder...

Files

Contents History

Name ↑	Last change	Commits
index.html	Tuesday	5d7f3373 Add files via upload 230701334
script.js	Tuesday	5d7f3373 Add files via upload 230701334
style.css	Tuesday	5d7f3373 Add files via upload 230701334

Set up build

Clone



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.

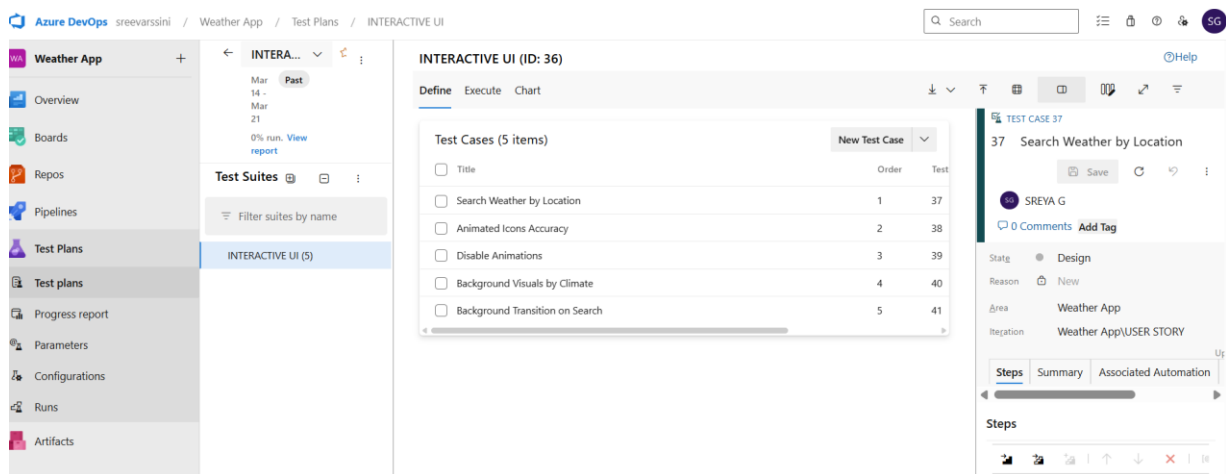
Step 5: Execute Test Cases

- 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



INTERACTIVE UI (ID: 36)

Define

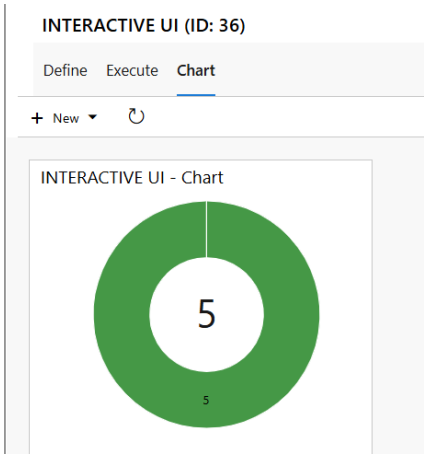
Execute

Chart

Test Points (5 items)

Run for web application

<input type="checkbox"/> Title	Outcome	Order	Test Case Id	Configuration	Tester
<input type="checkbox"/> Search Weather by Location	Passed	1	37	Windows 10	SREYA G
<input type="checkbox"/> Animated Icons Accuracy	Passed	2	38	Windows 10	SREYA G
<input type="checkbox"/> Disable Animations	Passed	3	39	Windows 10	SREYA G
<input type="checkbox"/> Background Visuals by Climate	Passed	4	40	Windows 10	SREYA G
<input type="checkbox"/> Background Transition on Search	Passed	5	41	Windows 10	SREYA G



Azure DevOps

sreevarssini / Weather App / Test Plans

Q Search

SG

WA Weather App

+

Overview

Boards

Repos

Pipelines

Test Plans

Test plans

Progress report

Parameters

Configurations

Runs

Artifacts

Test Plans

+ New Test Plan

Mine All

Title	Test Plan ID	State	Area Path	Iteration	Assigned To	☆
INTERACTIVE UI	35	Active	Weather App	Weather App\USER STORY	<div>SG</div> SREYA G	☆
LIVE WEATHER UPDATES	28	Active	Weather App	Weather App\USER STORY	<div>2</div> 230701335	☆
EMERGENCY ALERTS	21	Active	Weather App	Weather App\USER STORY	<div>SS</div> 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)

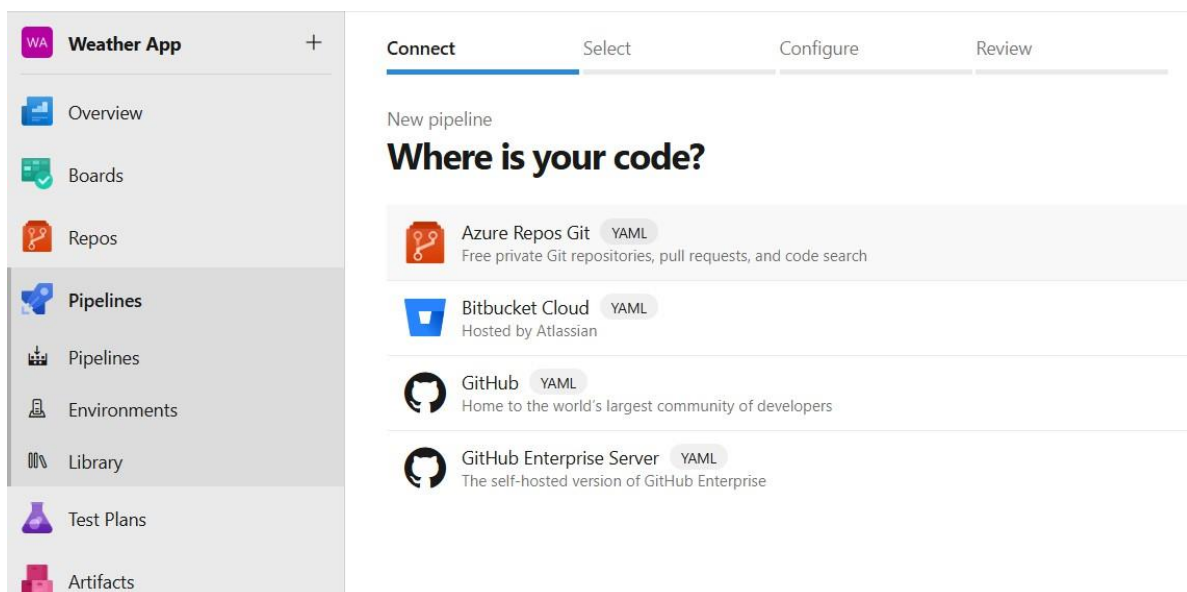
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

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

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

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

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

Summary Code 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

Azure DevOps sreevarssini / Weather App / Boards / Work items

Search

Weather App

Overview

Boards

Work items

Boards

Backlogs

Sprints

Queries

Delivery Plans

Analytics views

Repos

Pipelines

Test Plans

Artifacts

Project settings

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

Recently updated Back to Work Items

3 of 10

USER STORY 16

As a user, I want the app to include animated weather icons (rain, snow, sunny, etc.) so that I can instantly recognize the weather conditions.

SREYA G

Comments Add Tag

Save Follow

Updated by SREYA G Yesterday

State New

Area Weather App

Reason New

Iteration Weather App

Acceptance Criteria

1. Animated icons reflect real-time weather data.
2. Rain icon includes falling drops animation.
3. Snow shows snowflakes drifting.
4. Thunderstorm includes lightning flashes.
5. Sunny icon glows with animated rays.
6. Cloudy icon shows moving clouds.
7. Animation updates every 30 minutes.
8. Users can disable animations in settings.
9. Animations are GPU-optimized for performance.
10. App uses fallback static icon if animation fails.
11. All animations are under 1MB for download.
12. Animation speed adjustable via settings.
13. Animations load instantly on launch.
14. Weather animation matches forecast when forecast is tapped.
15. Animation is consistent across all devices and platforms.

Story Points

Priority 2

Risk

Classification

Value area Business

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 Yesterday Succeeded

Related Work

Add link

Parent

13 Interactive UI

Updated Apr 17 @ New

RESULT:

Thus the CI/CD pipeline has been successfully implemented.