RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23432

SOFTWARE CONSTRUCTION

Laboratory Record Note Book

Name : **Nishanth P**

Year / Branch / Section : **II – B.Tech Information Technology - AE**

Register No. : **231001138**

Semester : **IV**

Academic Year : **2024-2025**

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM − 602 105

BONAFIDE CERTIFICATE

NAME ___NISHANTH P___ REGISTER NO. ___231001138___

ACADEMIC YEAR 2024-25 SEMESTER- IV BRANCH: B. Tech Information Technology [AD/AE]. This Certification is the Bonafide record of work done by the above student in the CS23432- Software Construction Laboratory during the year 2024-2025.

Signature of Faculty-in

Charge

Submitted for the Practical Examination held on _____

Internal Examiner   External Examiner

LAB PLAN

CS23432-SOFTWARE CONSTRUCTION LAB

EX NO: 1
Date: 21/01/2025

Study of Azure DevOps

AIM:

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

STUDY:

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

1.   Understanding Azure DevOps

Azure DevOps consists of five key services:

1.1   Azure Repos (Version Control)

Supports Git repositories and Team Foundation Version Control (TFVC). Provides

features like branching, pull requests, and code reviews.

1.2   Azure Pipelines (CI/CD)

Automates build, test, and deployment processes.

Supports multi-platform builds (Windows, Linux, macOS).

Works with Docker, Kubernetes, Terraform, and cloud providers (Azure, AWS, GCP).

1.3   Azure Boards (Agile Project Management)

Manages work using Kanban boards, Scrum boards, and dashboards. Tracks

user stories, tasks, bugs, sprints, and releases.

1.4   Azure Test Plans (Testing)

Provides manual, exploratory, and automated testing.

Supports test case management and tracking.

1.5   Azure Artifacts (Package Management)

Stores and manages NuGet, npm, Maven, and Python packages. Enables

versioning and secure access to dependencies.

Getting Started with Azure DevOps

Step 1: Create an Azure DevOps Account Visit

Azure DevOps.

Sign in with a Microsoft Account.
Create an Organization and a Project.
Step 2: Set Up a Repository

(Azure Repos)

Navigate to Repos.

Choose Git or TFVC for version control. Clone the

repository and push your code.

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

Go to Pipelines → New Pipeline.

Select a source code repository (Azure Repos, GitHub, etc.).
Define the pipeline using YAML or the Classic Editor. Run the pipeline

to build and deploy the application.

Step 4: Manage Work with Azure Boards Navigate to

Boards.

Create work items, user stories, and tasks. Organize sprints and

track progress.

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

Test Plans.

Create and run test cases

View test results and track

Result:

The study was successfully completed.

EX NO: 2
Date: 28/01/2025

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

Problem Statement:

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

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

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

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

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

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

Proposed Solution:

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

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

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

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

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

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

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

Goal:

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

Key Features (Anticipated):

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

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

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

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

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

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

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

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

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

Result:
The problem statement was written successfully.

EX NO: 3
DATE: 04/02/2025

<p align="center">**AGILE PLAN**</p>

Aim:

> To prepare an agile Plan for weather application.

AGILE PLANNING

Agile planning is a part of the Agile methodology, which is a project management style with an incremental, iterative approach. Instead of using an in-depth plan from the start of the project—which is typically product-related—Agile leaves room for requirement changes throughout and relies on constant feedback from end users.

With Agile planning, a project is broken down into smaller, more manageable tasks with the ultimate goal of having a defined image of a project's vision. Agile planning involves looking at different aspects of a project's tasks and how they'll be achieved, for example:

· Roadmaps to guide a product's release ad schedule

· Sprints to work on one specific group of tasks at a time

· A feedback plan to allow teams to stay flexible and easily adapt to change

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

· Steps in Agile planning process

1. Define vision
2. Set clear expectations on goals
3. Define and break down the product roadmap
4. Create tasks based on user stories
5. Populate product backlog
6. Plan iterations and estimate effort

7. Conduct daily stand-ups

8. Monitor and adapt

Agile Plan: Weather App

I. Product Vision:

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

II. Core Principles:

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

III. Team Structure:

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

IV. Agile Approach:

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

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

- Display current temperature and weather conditions (user's current location).

- Show daily high and low temperatures (current location).

- Enable users to search for weather information for different locations.

- Provide a 5-day weather forecast for a selected location.

VI. Iterative Releases:

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

VII. Communication & Collaboration:

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

VIII. Measuring Success:

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

IX. Continuous Improvement:

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

Result:
Thus the Agile plan was completed successfully.

EX NO: 4
Date: 18/02/2025

CREATE USER STORIES

Aim:

To create User Stories for

THEORY

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

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

Procedure:
1. Open your web browser and go to the Azure website: _https://azure.microsoft.com/en-in_ Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for _https://signup.live.com/?lic=1_
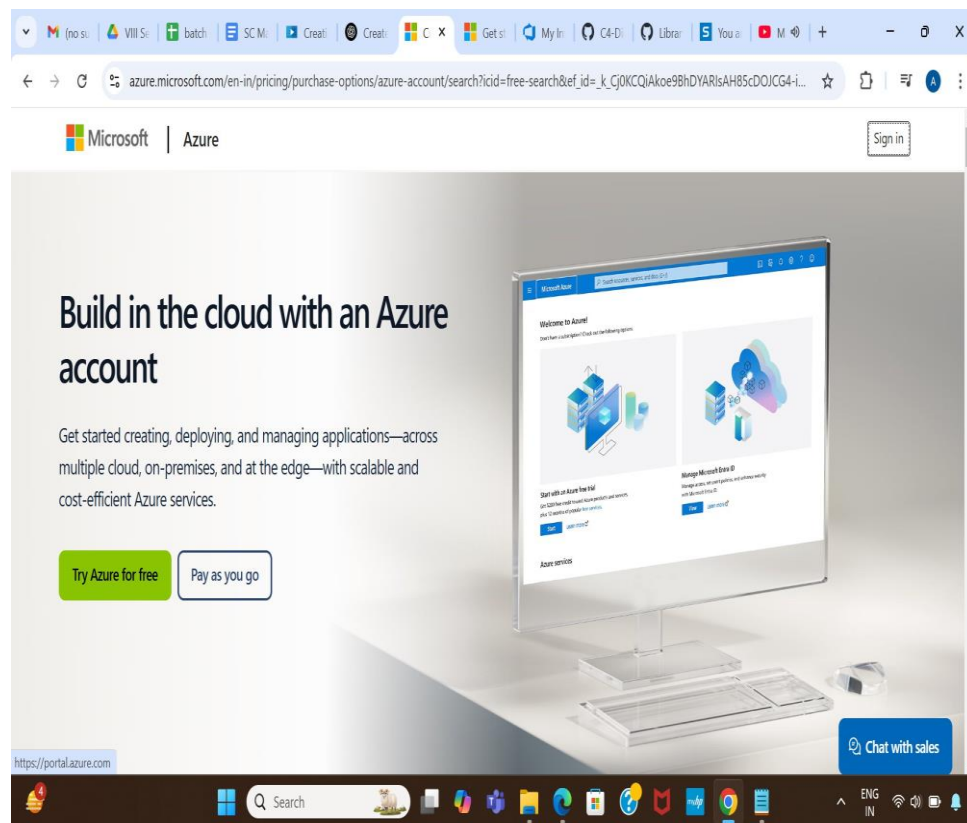
3. Azure home page
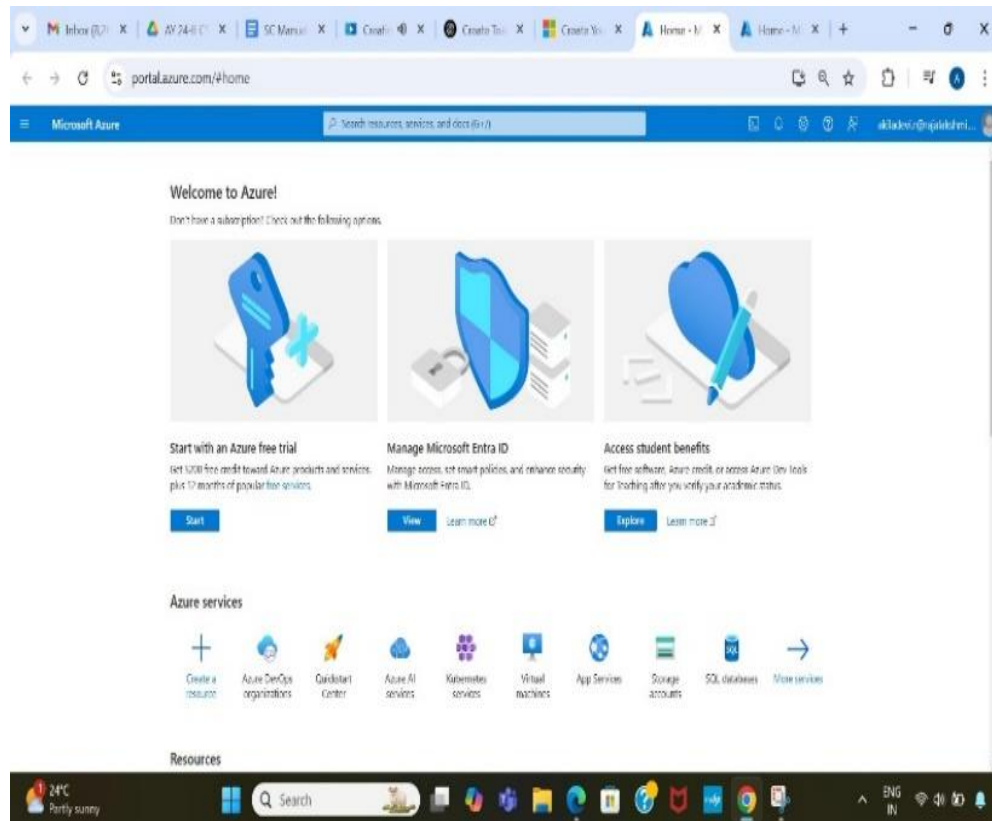
4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.
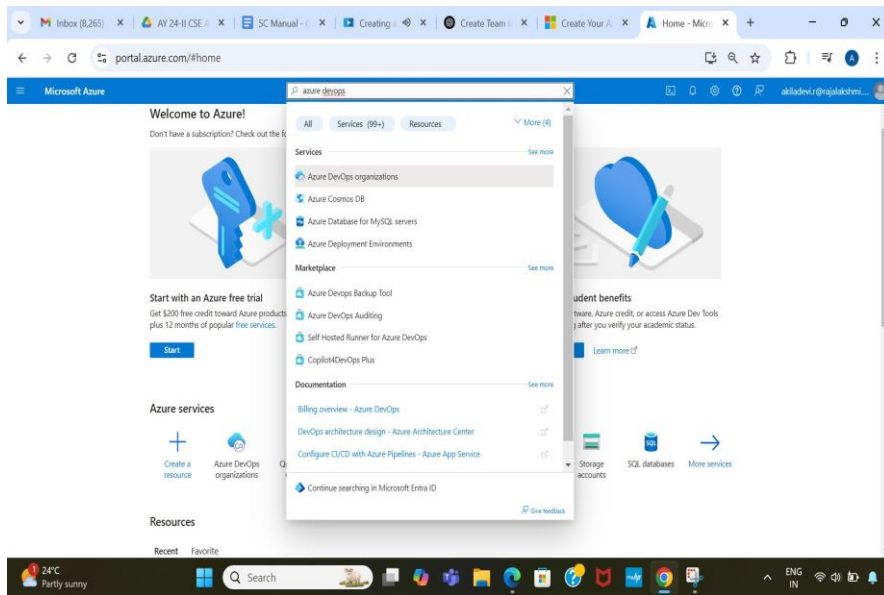
5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.
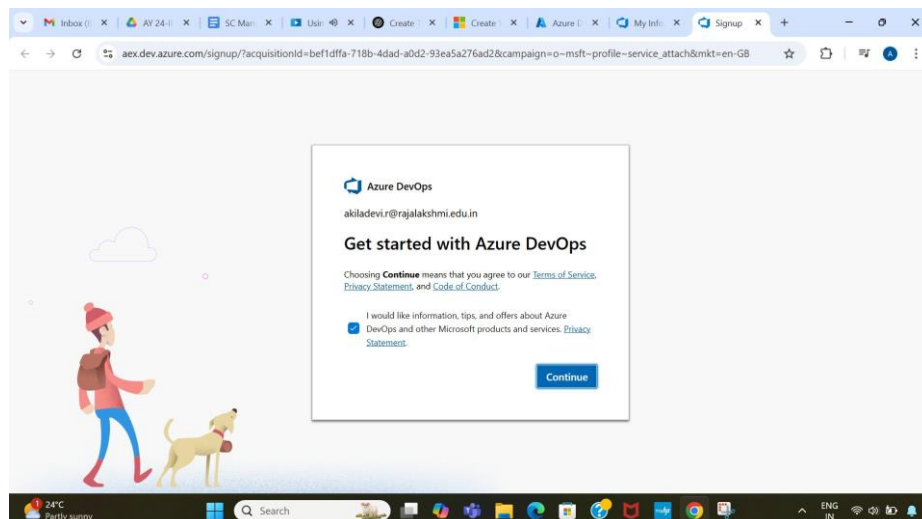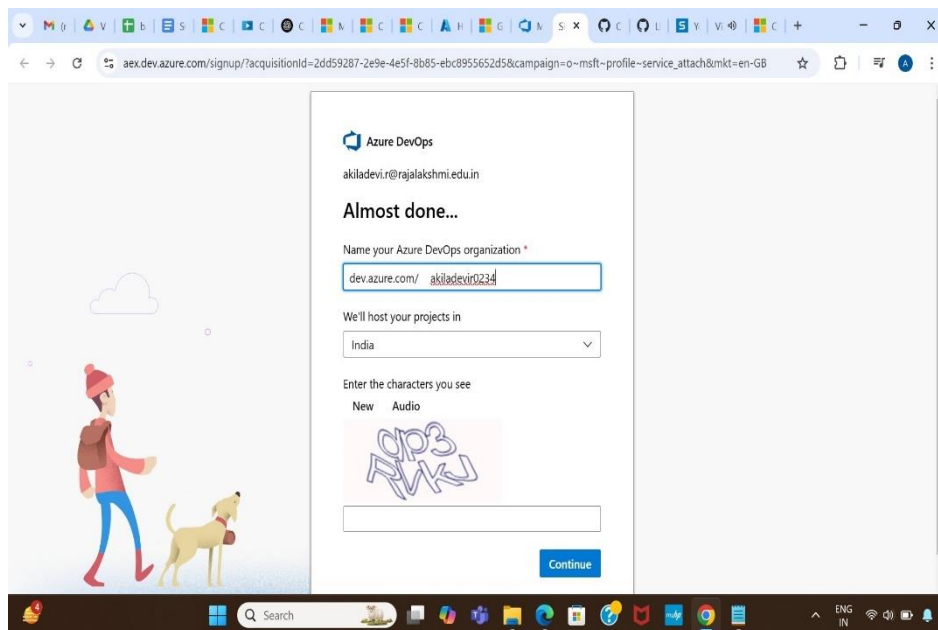
6. Create the First Project in Your Organization

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

   i.   On the organization's **Home page**, click on the **New Project** button.
   ii.   Enter the project name, description, and visibility options:
   - **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.
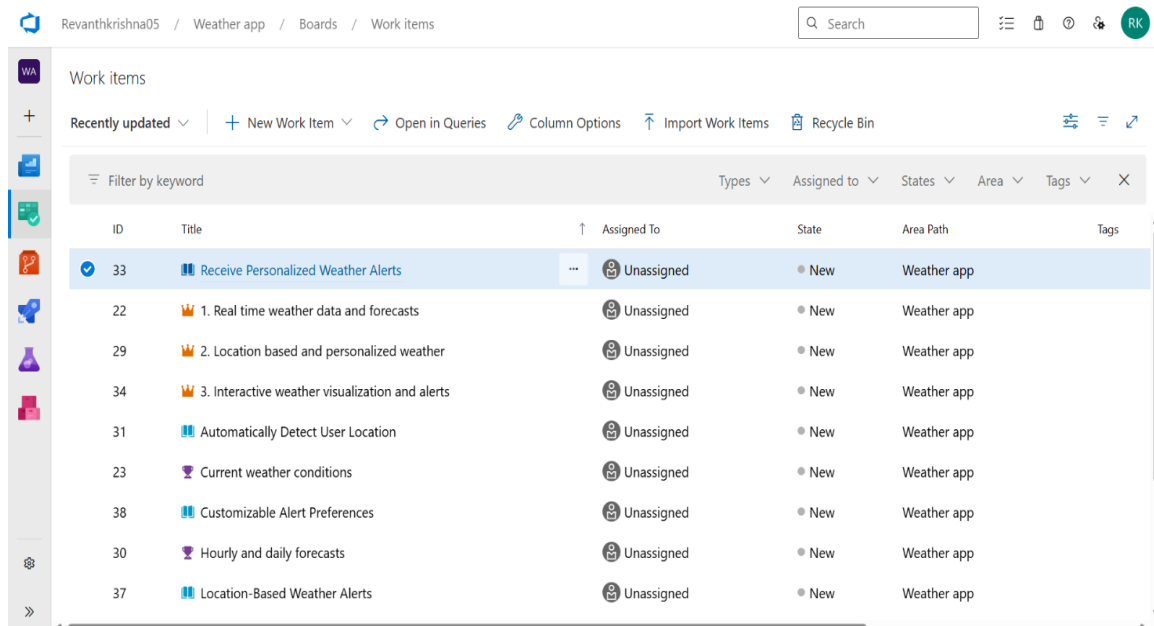
8. Project dashboard
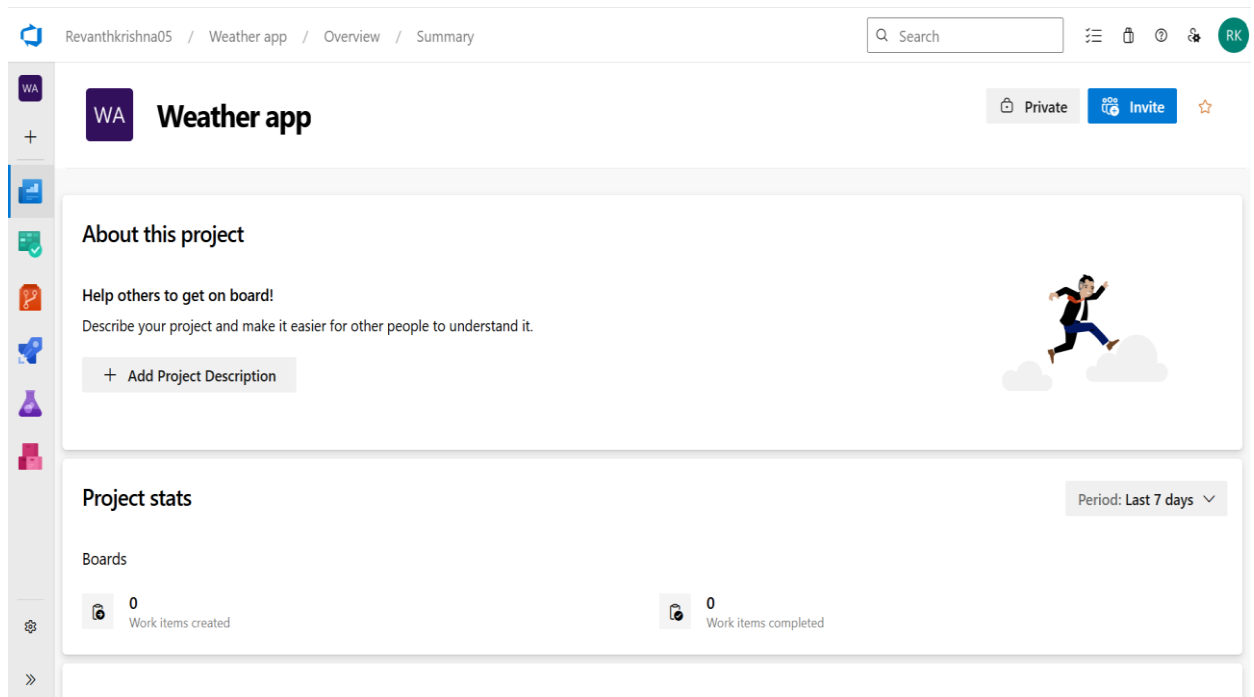
9. To manage user stories
a. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards** page, where you can manage work items, backlogs, and sprints.

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

10. Fill in User Story Details

Result:

The user story was written successfully.

EX NO : 5
Date:
25/02/2025

<div align="center">

SEQUENCE DIAGRAM

</div>

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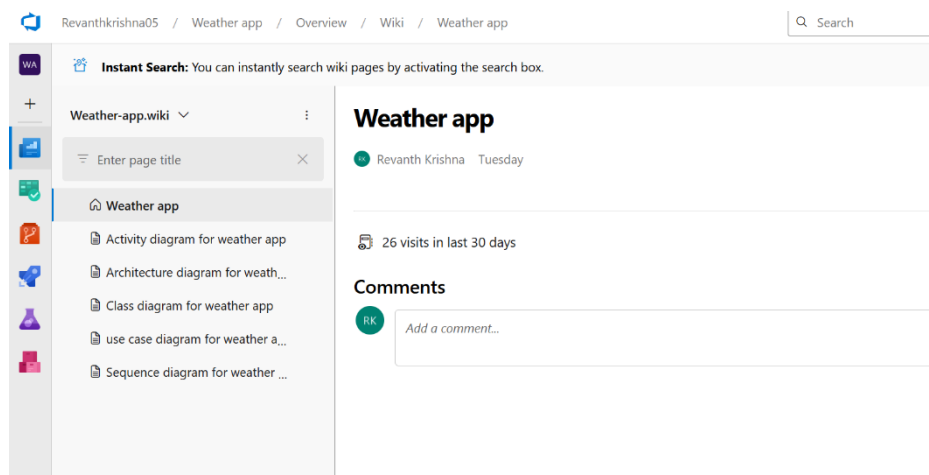
3. Write code for drawing sequence diagram and save the code.

::: mermaid
sequenceDiagram
participant U as User
participant W as Web App
participant APIM as Azure API Management
participant F as Azure Function
participant API as External Weather API
participant DB as Azure Database

U->>W: Open app

W->>APIM: Request weather data

APIM->>F: Query weather data

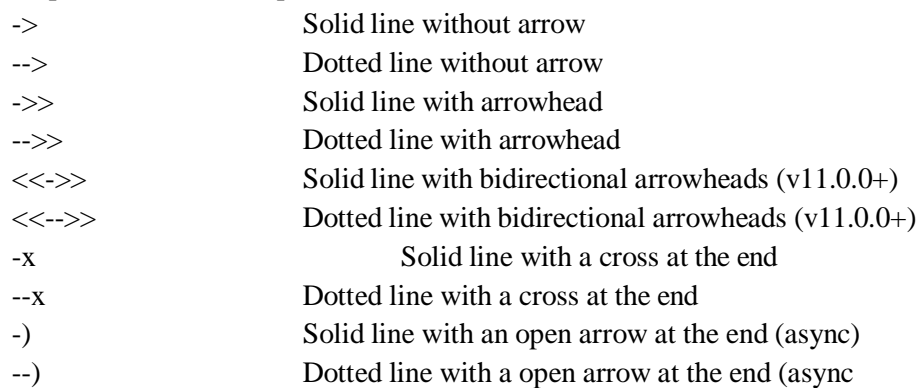F->>API: Fetch from External Weather API

API-->>F: Return weather data

F->>DB: Save/cache data (optional)

F-->>APIM: Weather data

APIM-->>W: Weather data

W-->>U: Display weather data

W->>DB: Save search history/preferences

:::

Explanation:

participant defines the entities involved.

->> represents a direct message.

-->> represents a response message.

+ after ->> activates a participant.

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

for conditional flows.

loop can be used for repeated actions.

->	Solid line without arrow
-->	Dotted line without arrow
->>	Solid line with arrowhead
-->>	Dotted line with arrowhead
<<->>	Solid line with bidirectional arrowheads (v11.0.0+)
<<-->>	Dotted line with bidirectional arrowheads (v11.0.0+)
-x	Solid line with a cross at the end
--x	Dotted line with a cross at the end
-)	Solid line with an open arrow at the end (async)
--)	Dotted line with a open arrow at the end (async

4. click wiki menu and select the page
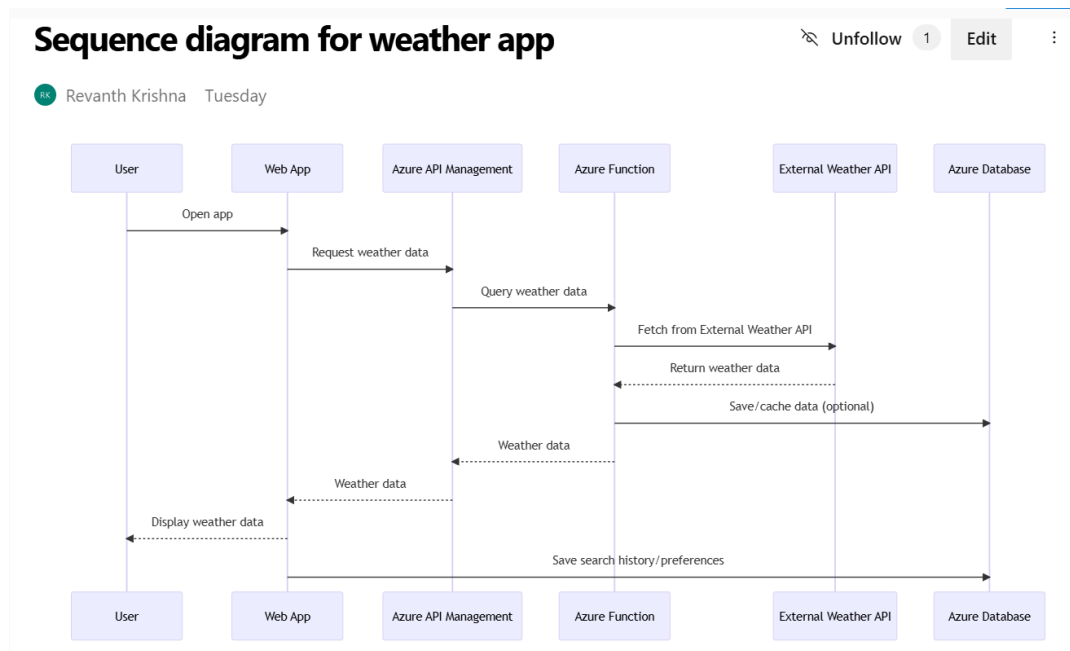
Result:

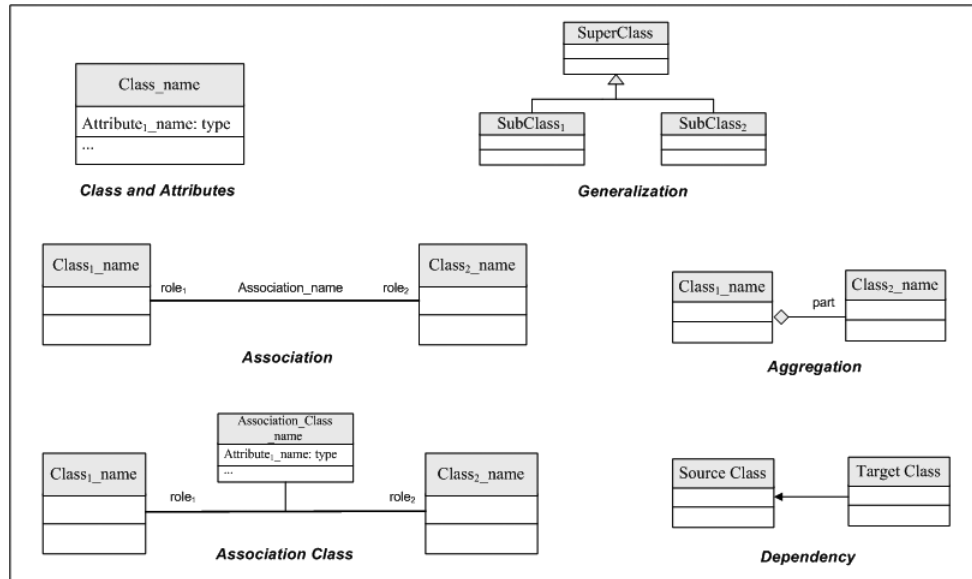The sequence diagram was drawn successfully.

EX NO. 6
Date: 04/03/2025

CLASS DIAGRAM

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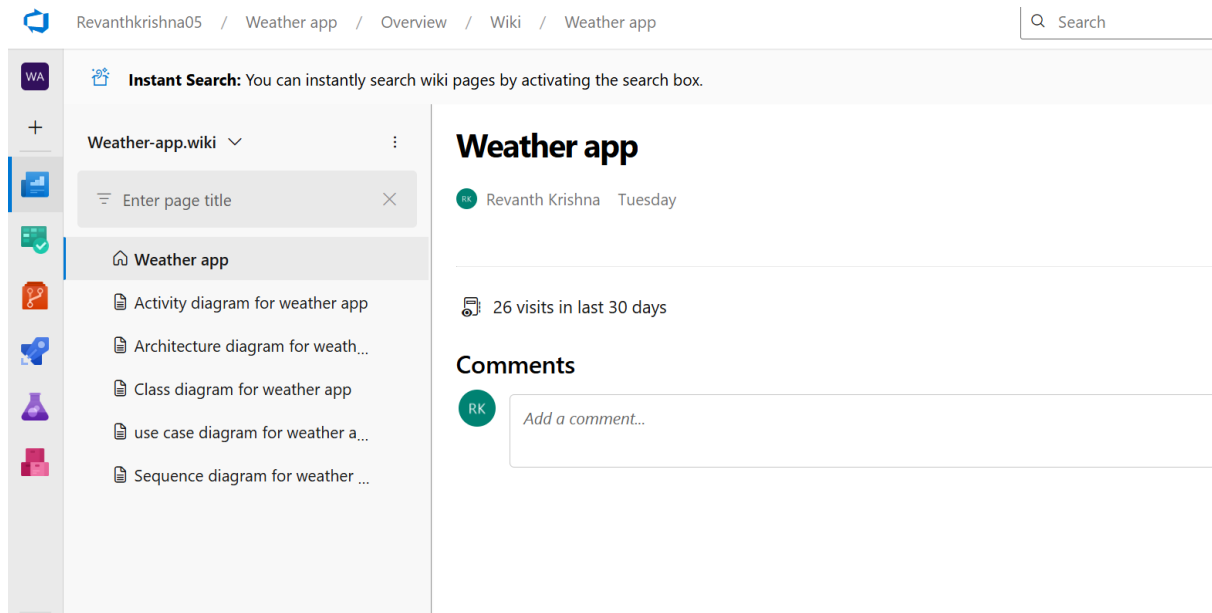To draw a sample class diagram for your project or system.

THEORY

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

Notations in class diagram

Procedure:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu

WA

+

Instant Search: You can instantly search wiki pages by activating the search box.

Weather-app.wiki ⌄

⋮

≡ Enter page title ✕

⌂ Weather app

📄 Activity diagram for weather app

📄 Architecture diagram for weath...

📄 Class diagram for weather app

📄 use case diagram for weather a...

📄 Sequence diagram for weather ...

Weather app

RK Revanth Krishna   Tuesday

📊 26 visits in last 30 days

Comments

RK   *Add a comment...*

---

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

::: mermaid
classDiagram

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

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

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

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

class User {
    +userId

```
    +username
    +email
    +getPreferredLocation()
}

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

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

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

WeatherController --> WeatherService
WeatherController --> LocationService
WeatherService --> WeatherData
User --> LocationService
Pipeline --> Repo
Repo --> Boards
::::
```
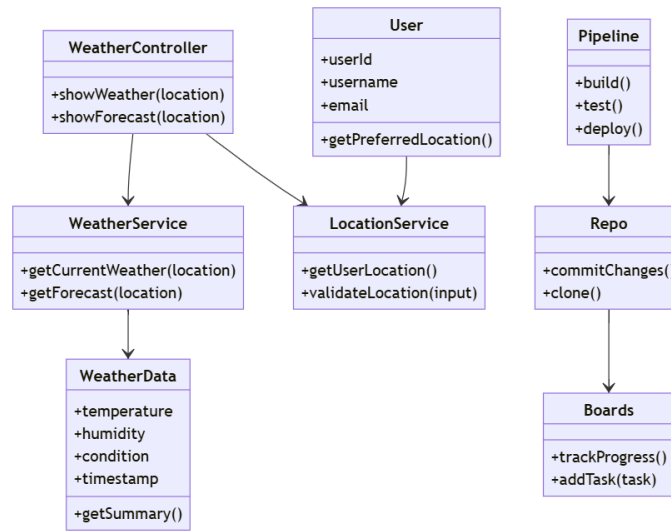
Relationship Types

Type	Description
<|	Inheritance
*	Composition
o	Aggregation
>	Association
<	Association
|>	Realization

Class diagram for weather app

Revanth Krishna   Tuesday

WeatherController

+showWeather(location)
+showForecast(location)

User

+userId
+username
+email

+getPreferredLocation()

Pipeline

+build()
+test()
+deploy()

WeatherService

+getCurrentWeather(location)
+getForecast(location)

LocationService

+getUserLocation()
+validateLocation(input)

Repo

+commitChanges()
+clone()

WeatherData

+temperature
+humidity
+condition
+timestamp

+getSummary()

Boards

+trackProgress()
+addTask(task)

Result:

The use case diagram was designed successfully.

<div align="center"><u>**USECASE DIAGRAM**</u></div>

Aim:

Steps to draw the Use Case Diagram using draw.io

Theory:

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

• **Use Cases**

• **Actors**

• **Relationships**

• **System Boundary Boxes**
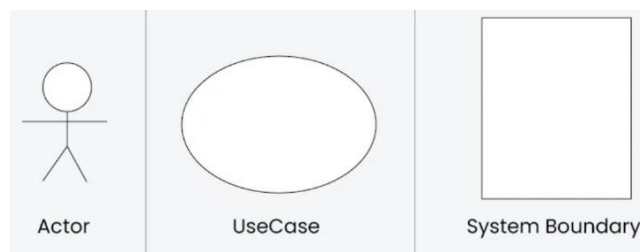
Procedure

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

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

PNG/JPG/SVG. Step 2: Upload the Diagram to Azure DevOps

Option 1: Add to Azure DevOps Wiki
• Open Azure DevOps and go to your project.
• Navigate to Wiki (Project > Wiki).
• Click "Edit Page" or create a new page.
• Drag & Drop the exported PNG/JPG image.
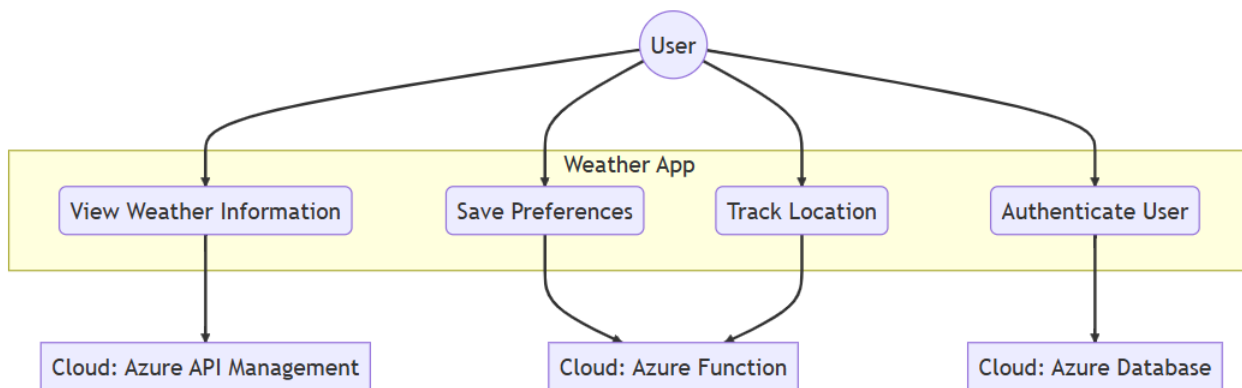
- Use Markdown to embed the diagram:
- ![Use Case Diagram](attachments/use_case_diagram.png)

Option 2: Attach to Work Items in Azure Boards
- Open Azure DevOps → Navigate to Boards (Project > Boards).
- Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram.
- Add comments or descriptions to explain the use case diagram

use case diagram for weather app

RK Revanth Krishna   Tuesday

Result:

The use case diagram was designed successfully

EX NO. 8

Date: 18/03/2025

ACTIVITY DIAGRAM

AIM :-

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

THEORY

Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different

actions are connected and how a system moves from one state to another.

Notations	Symbol	Meaning
Start	⬬	Shows the beginning of a process
Connector	⟶	Shows the directional flow, or control flow, of the activity
Joint symbol		Combines two concurrent activities and re-introduces them to a flow where one activity occurs at a time
Decision	◇	Represents a decision
Note		Allows the diagram creators 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

1. Draw diagram in draw.io
2. Upload the diagram in Azure DevOps wiki
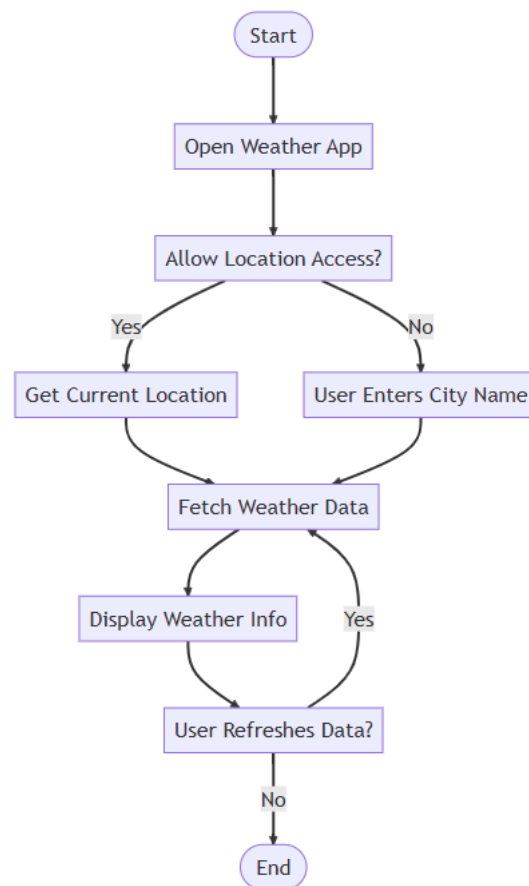
Result:
The activity diagram was designed successfully

EX NO. 9

Date: 25/03/2025

ARCHITECTURE DIAGRAM

Aim:

Steps to draw the Architecture Diagram using draw.io.

Theory:

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

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

Architecture diagram for weather app

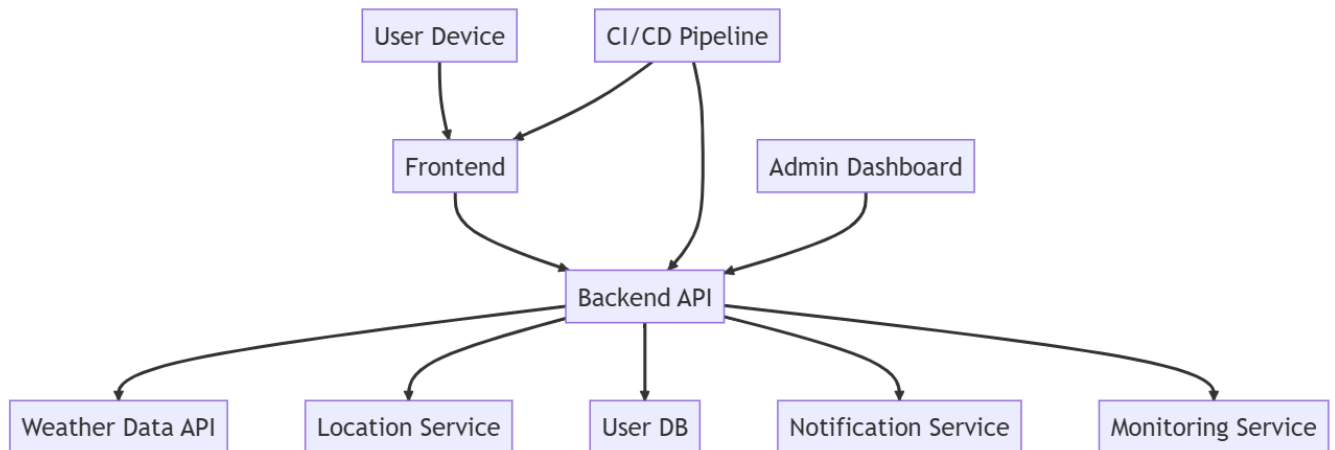RK Revanth Krishna   Tuesday

Result:
The architecture diagram was designed successfully

EX NO. 10
Date:
01/04/2025

USER INTERFACE

Aim:

Design User Interface for the given project

Result:
  The UI was designed successfully.

EX NO. 11
Date: 08/04/2025

IMPLEMENTATION

Aim:

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

Procedure:
Step 1: Set Up an Azure DevOps Project
Log in to Azure DevOps.

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

Inside the project, navigate to "Repos" to store the source code.

Step 2: Add Your Web Application Code
Navigate to Repos → Click "Clone" to get the Git URL.

Open Visual Studio Code / Terminal and run:

bash
Copy
Edit
git clone <repo_url>
cd <repo_folder>
Add web application code:

(HTML, CSS, JavaScript files for Weather App)

Commit and push:

bash
Copy
Edit
git add .
git commit -m "Initial commit"
git push origin main
Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)
Navigate to Pipelines → Click "New Pipeline".

Select Git Repository (Azure Repos, GitHub, or Bitbucket).

Choose a Starter Pipeline or a pre-configured template.

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

yaml
Copy
Edit

```yaml
trigger:
  - main

pool:
  vmImage: 'ubuntu-latest'

steps:
  - task: UseNode@1
    inputs:
      version: '16.x'

  - script: npm install
    displayName: 'Install dependencies'

  - script: npm run build
    displayName: 'Build application'

  - task: PublishBuildArtifacts@1
    inputs:
      pathToPublish: 'dist'
      artifactName: 'drop'
```

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

Step 4: Set Up Release Pipeline (CD - Continuous Deployment)
Go to Releases → Click "New Release Pipeline".

Select Azure App Service or Virtual Machines (VMs) for deployment.

Add an artifact (from the build pipeline).

Configure deployment stages (Dev, QA, Production).

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

Result:
Thus, the Weather App application was successfully implemented and deployed on Azure using Agile Methodology.