

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING LAB MANUAL**

**CS23432 – Software Construction**

**(REGULATION 2023)**

**RAJALAKSHMI ENGINEERING COLLEGE**

**Thandalam, Chennai-602015**

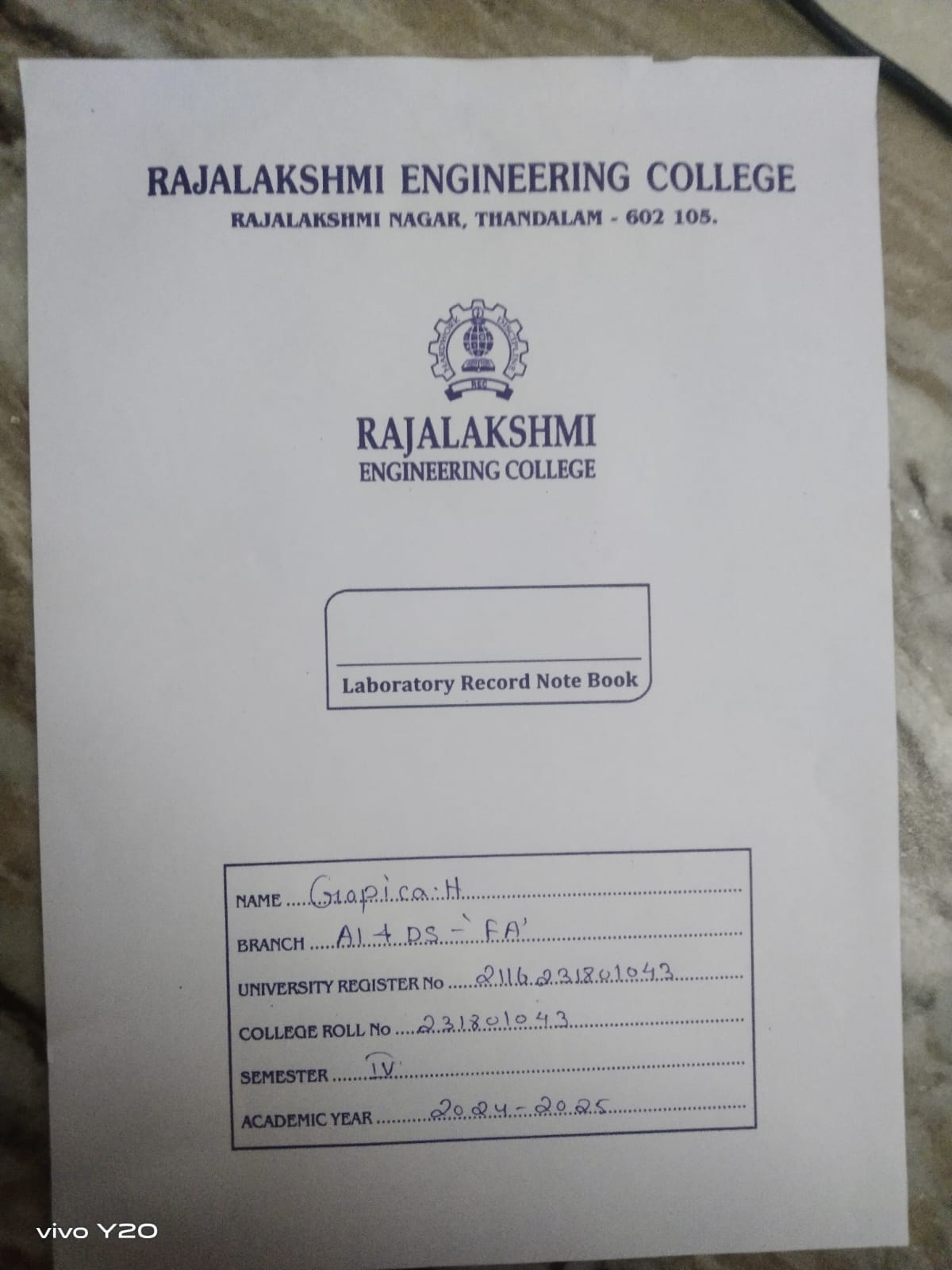
Name: Gopica H

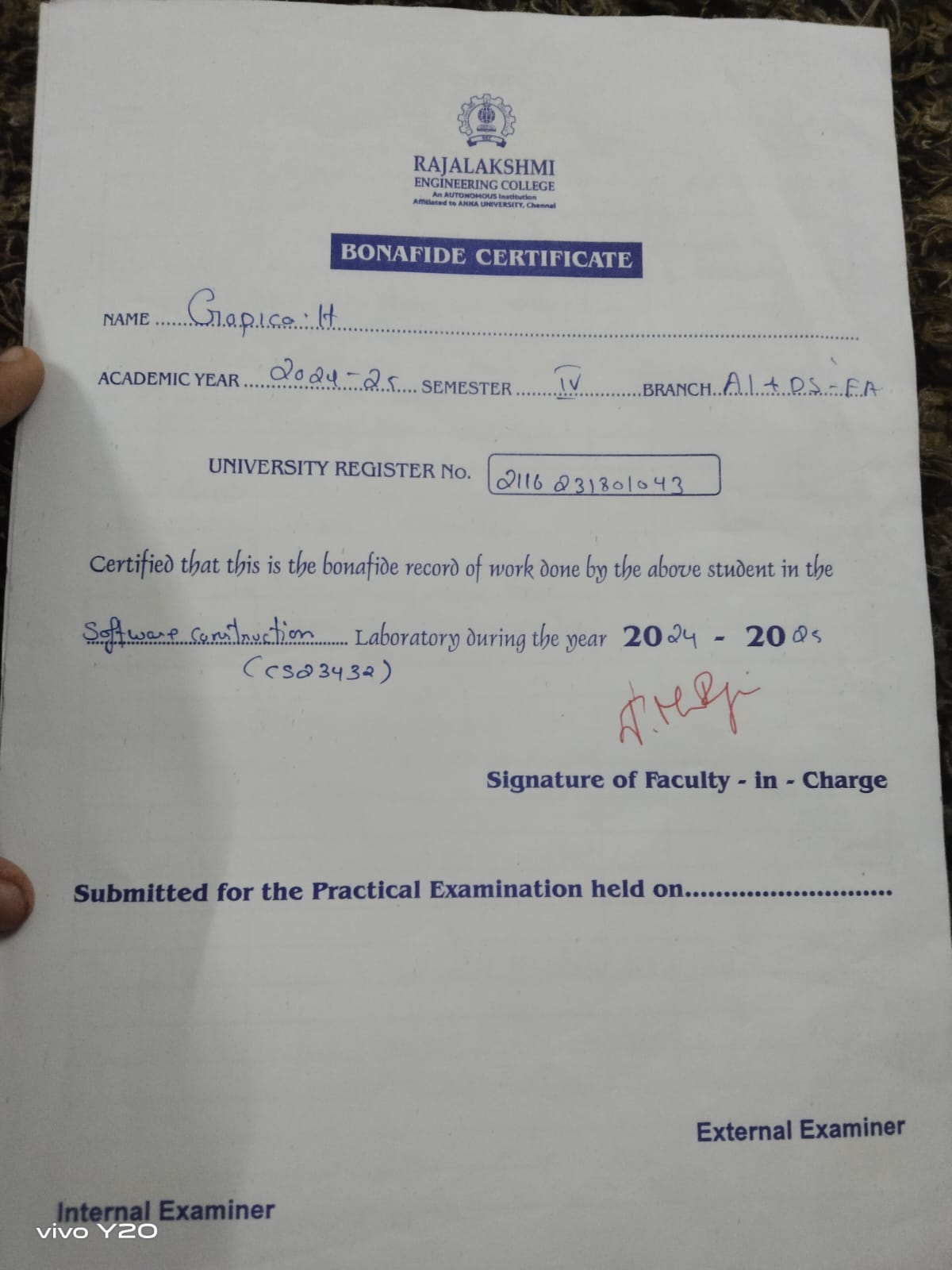
Register No: 231801043

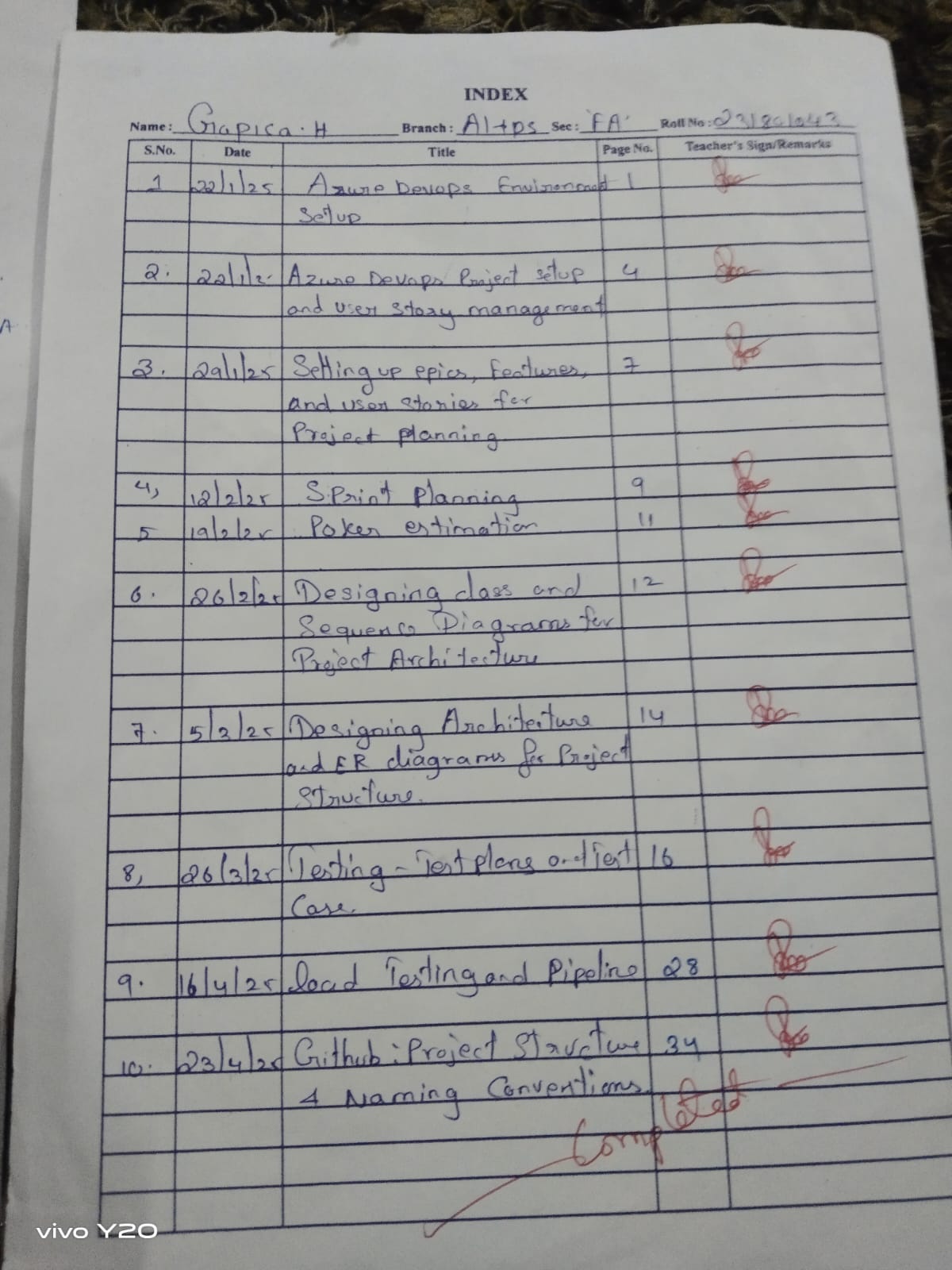
Year / Branch / Section: 2nd / AI&DS / FA

Semester: IV

Academic Year: 2024 - 2025







INDEX

|  |  |  |
| --- | --- | --- |
| S.No. | Date | Title |
| 1. | 21/1/25 | Azure Devops Environment Setup. |
| 2. | 21/1/25 | Azure Devops Project Setup and User Story Management. |
| 3. | 28/1/25 | Setting Up Epics, Features, And User Stories for Project Planning. |
| 4. | 11/2/25 | Sprint Planning. |
| 5. | 18/2/25 | Poker Estimation. |
| 6. | 25/2/25 | Designing Class and Sequence Diagrams for Project Architecture. |
| 7. | 04/3/25 | Designing Architectural and ER Diagrams for Project Structure. |
| 8. | 25/3/25 | Testing – Test Plans and Test Cases. |
| 9. | 15/4/25 | Load Testing and Pipelines. |
| 10. | 22/4/25 | GitHub: Project Structure & Naming Conventions. |

**EXP NO:** **1**

**AZURE DEVOPS ENVIRONMENT SETUP**

Create Epic, Features, User Stories, Task

**Aim:**

To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

**INSTALLATION**

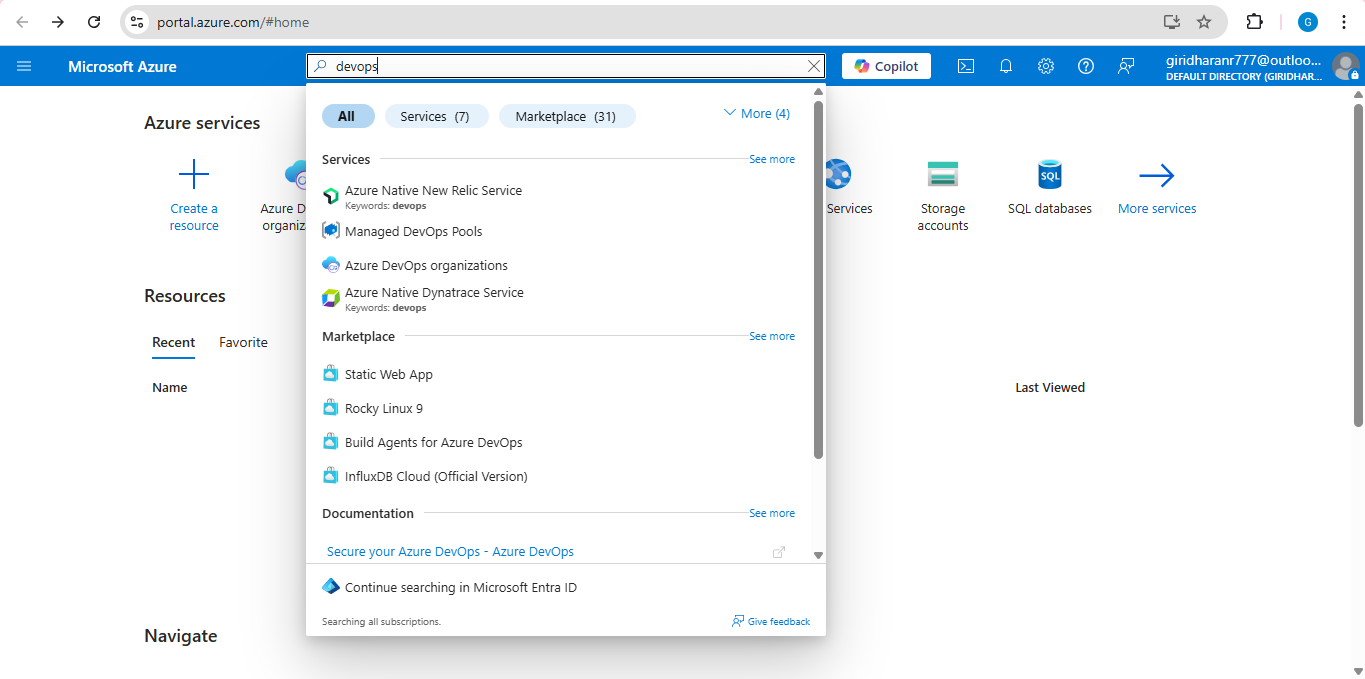
1.Open your web browser and go to the Azure website: <https://azure.microsoft.com/en-us/get-started/azure-portal>.  
Sign in using your Microsoft account credentials.  
If you don't have a Microsoft account, you can create one here: <https://signup.live.com/?lic=1>



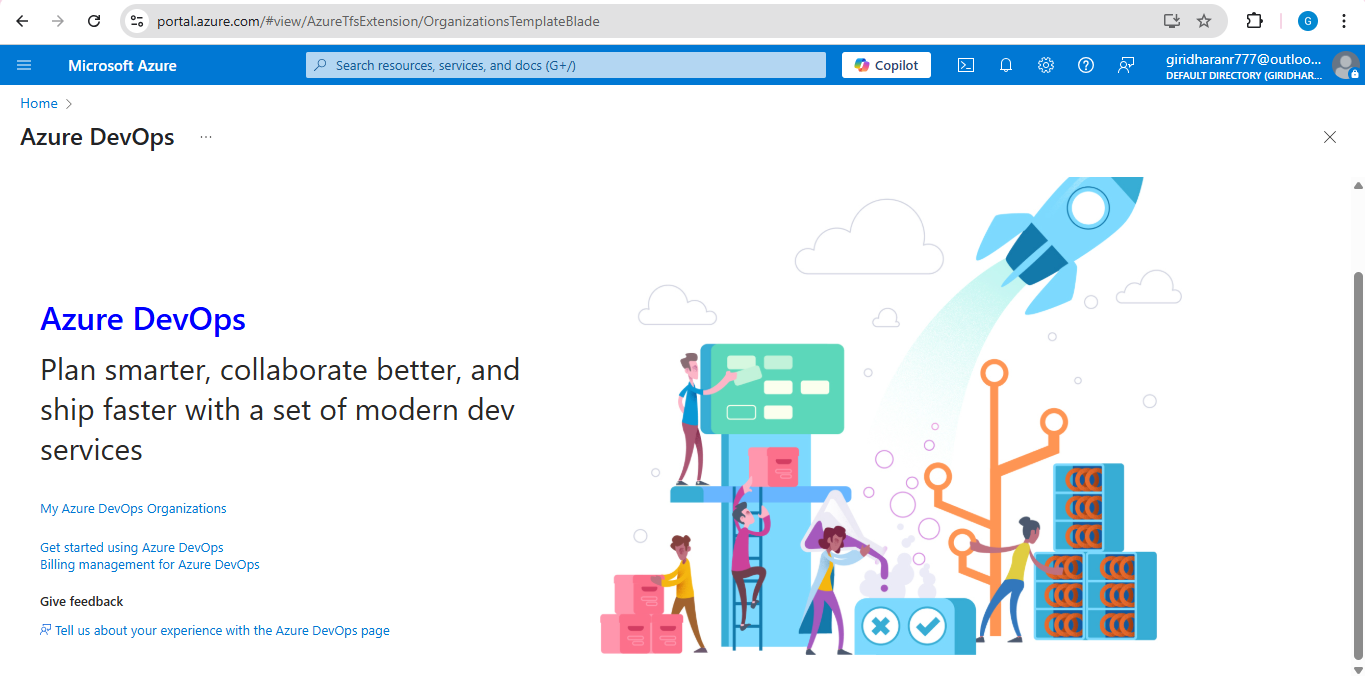
2.Azure home page



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



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



**Result:**

Successfully accessed the Azure DevOps environment and created a new organization through the Azure portal.

**EXP NO:** **2**

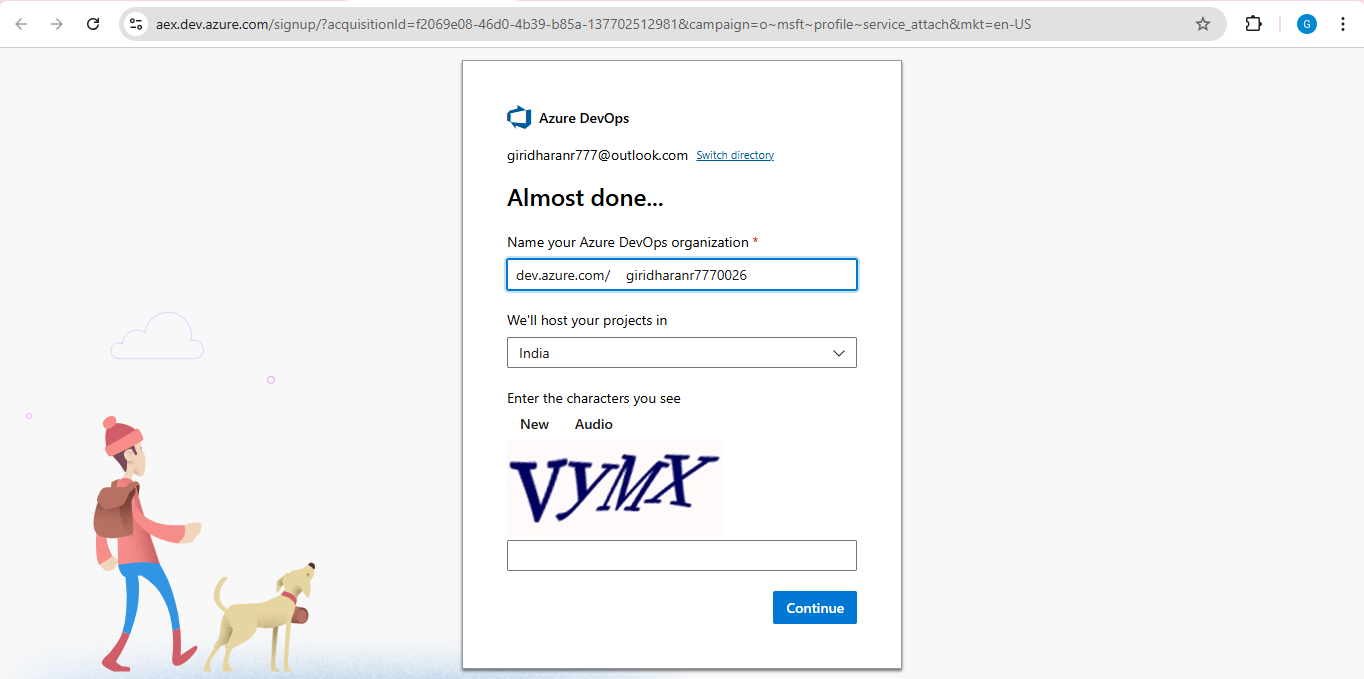
**AZURE DEVOPS PROJECT SETUP AND USER STORY MANAGEMENT**

Create Epic, Features, User Stories, Task

**Aim:**

To set up an Azure DevOps project for efficient collaboration and agile work management.

1.Create An Azure Account



2.Create the First Project in Your Organization

a. 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.

b. On the organization’s **Home page**, click on the **New Project** button.

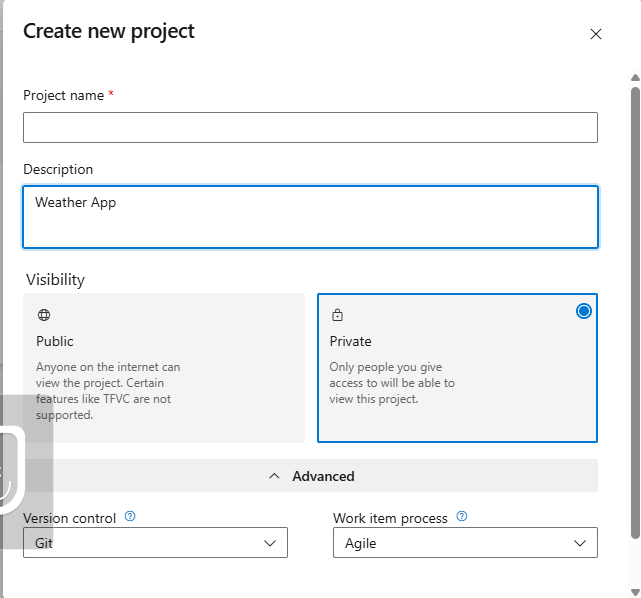
c. Enter the project name, description, and visibility options:

***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).

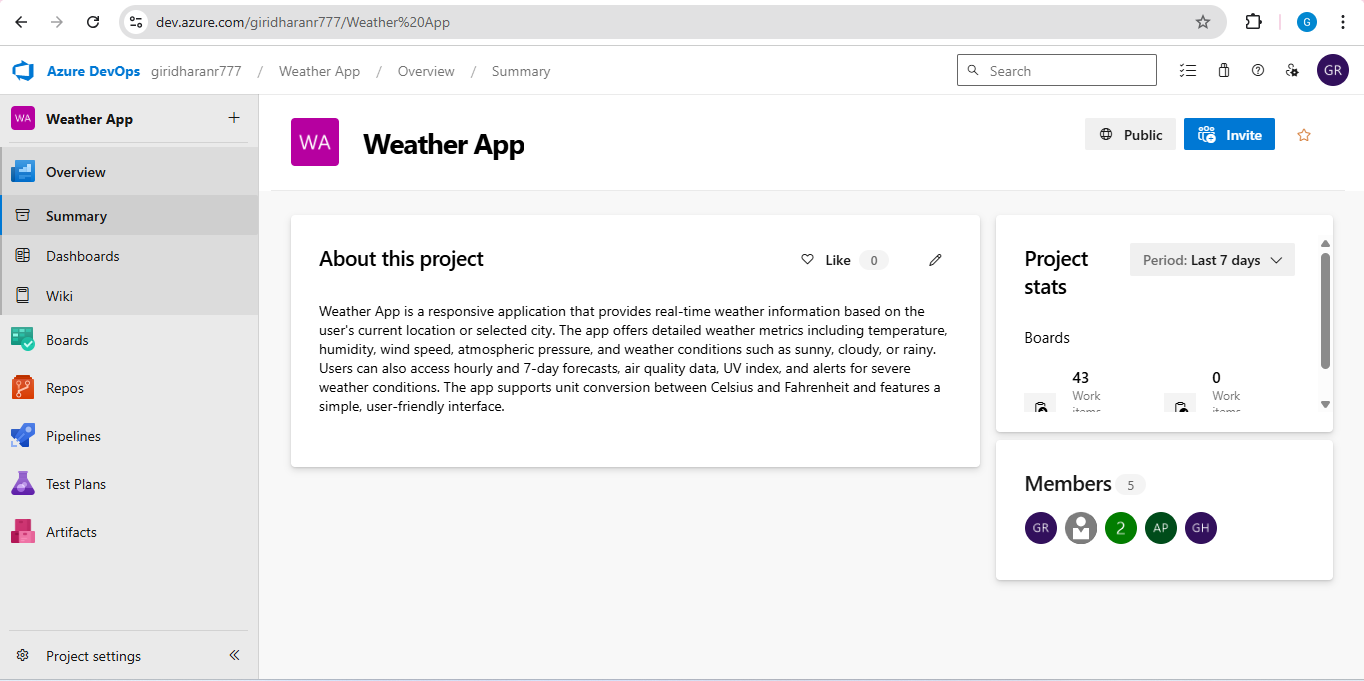
d. Once you’ve filled out the details, click **Create** to set up your first project.



3.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.



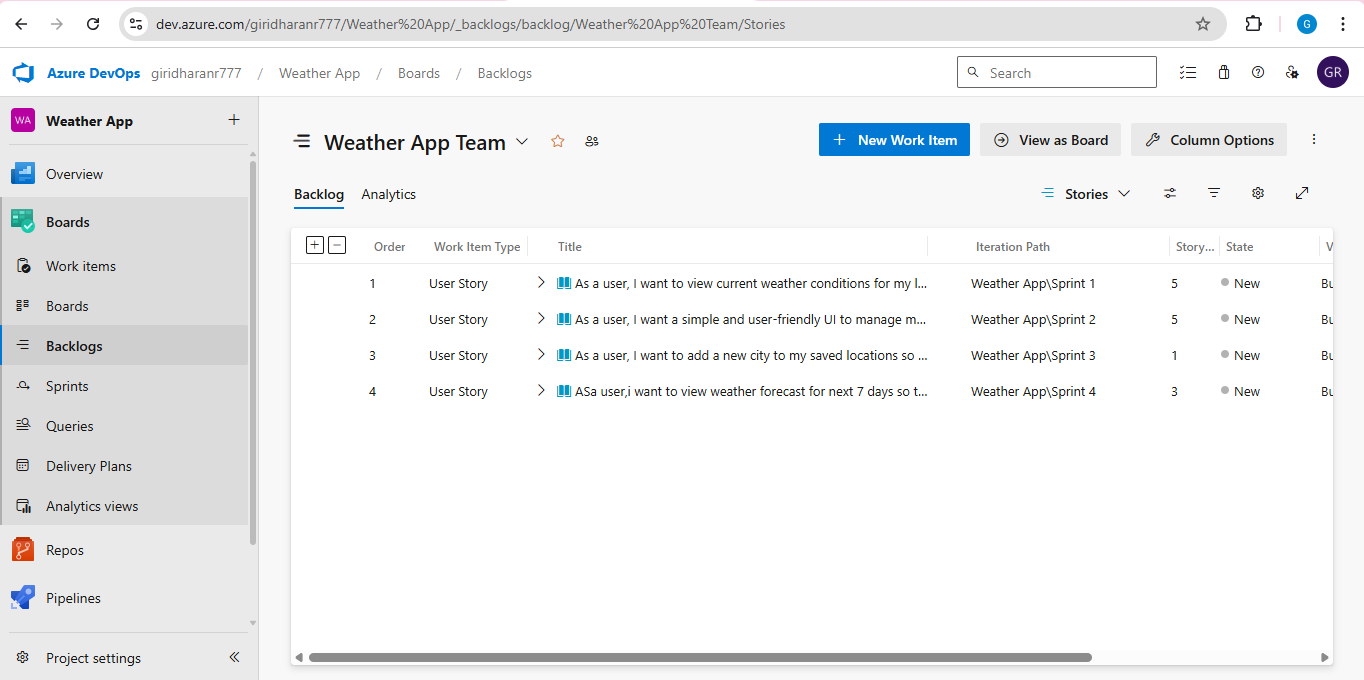
4.Project dashboard

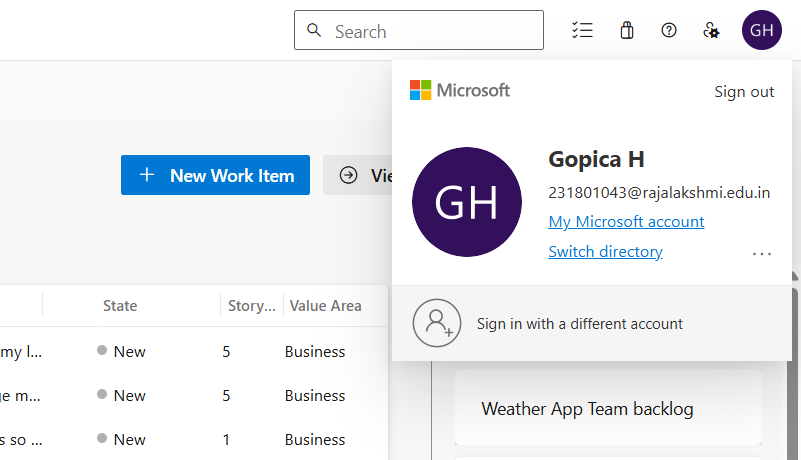


5.To manage user stories:

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.





**Result:**

Successfully created an Azure DevOps project with user story management and agile workflow setup.

**EXP NO:** **3**

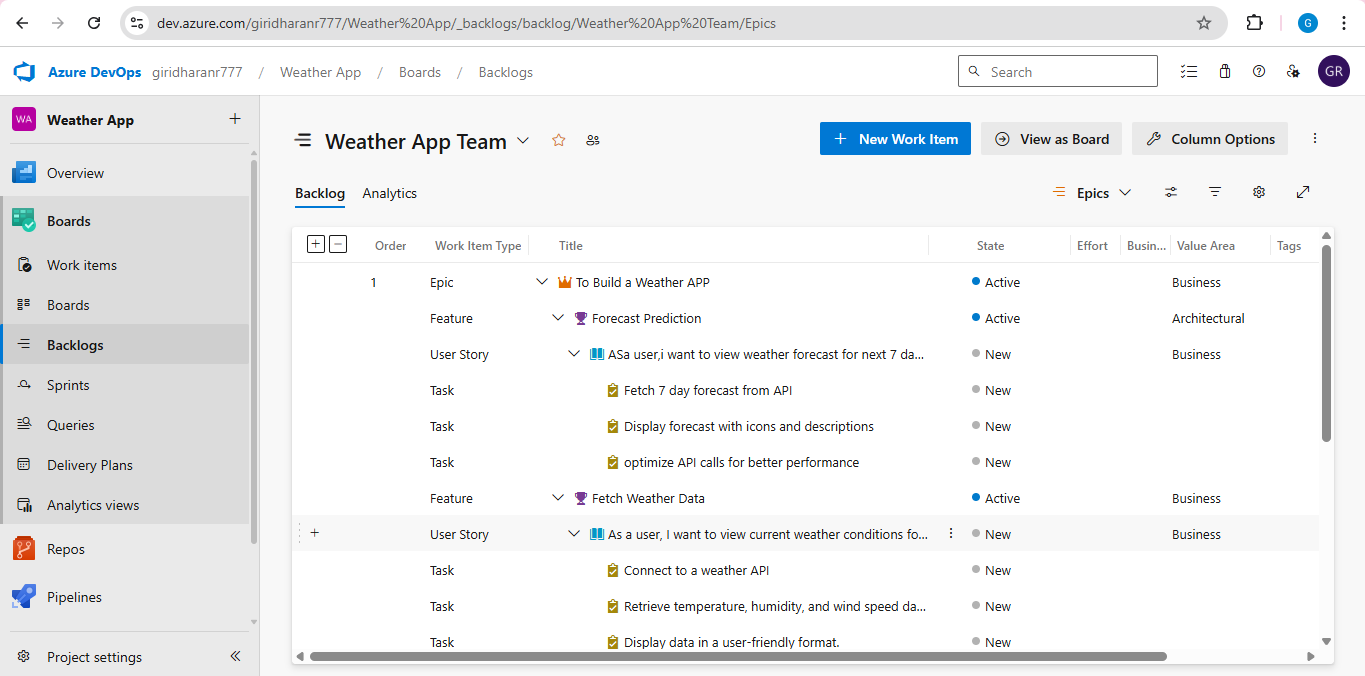
**SETTING UP EPICS, FEATURES, AND USER STORIES FOR PROJECT PLANNING**

Create Epic, Features, User Stories, Task

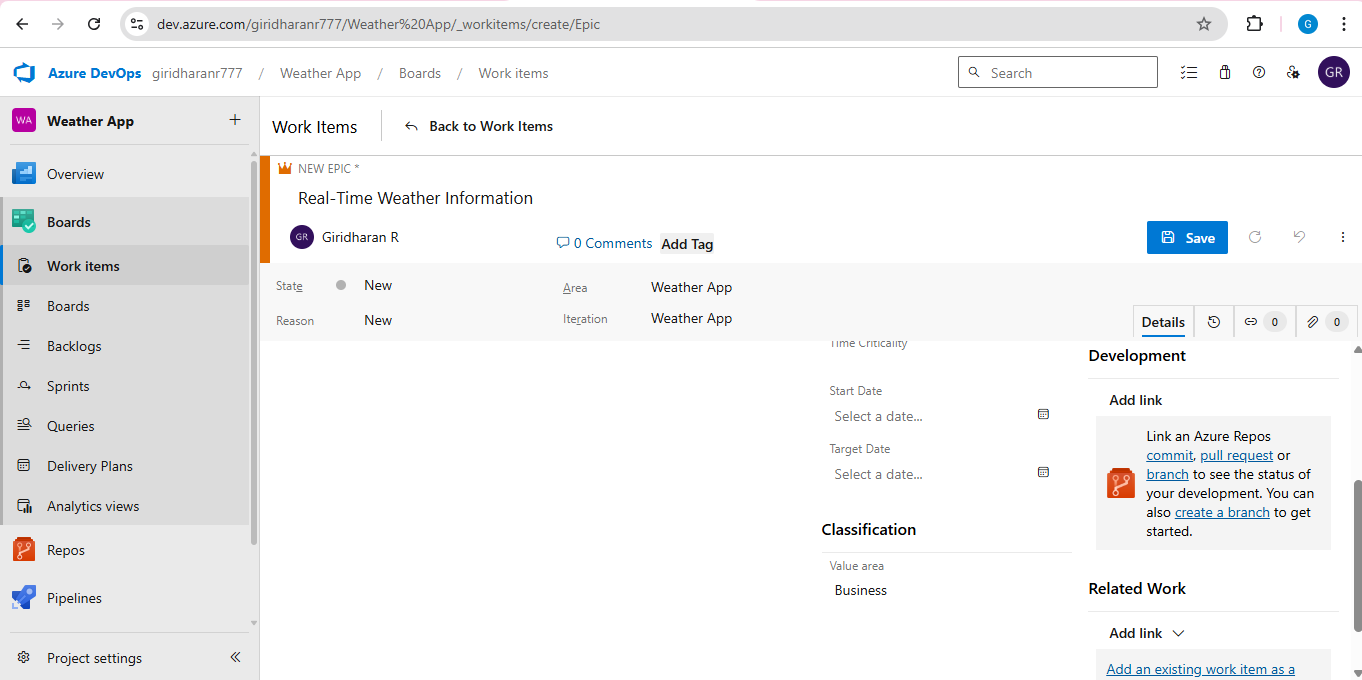
**Aim:**

To learn about how to create epics, user story, features, backlogs for your assigned project.

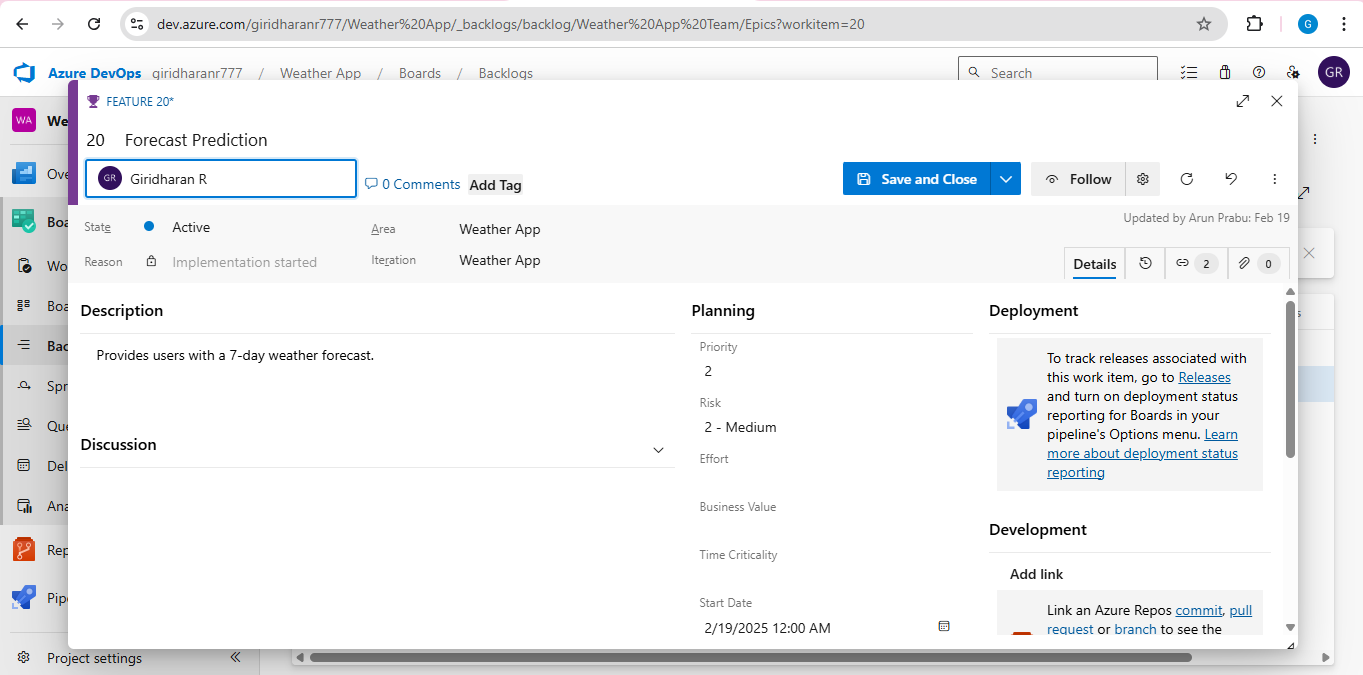
**Create Epic, Features, User Stories, Task**



**1.Fill in Epics**



**2.Fill in Features**



### **3.Fill in User Story Details**



**Result:**

Thus, the creation of epics, features, user story and task has been created successfully.

**EXP NO:** **4**

**SPRINT PLANNING**

Create Epic, Features, User Stories, Task

**Aim:**

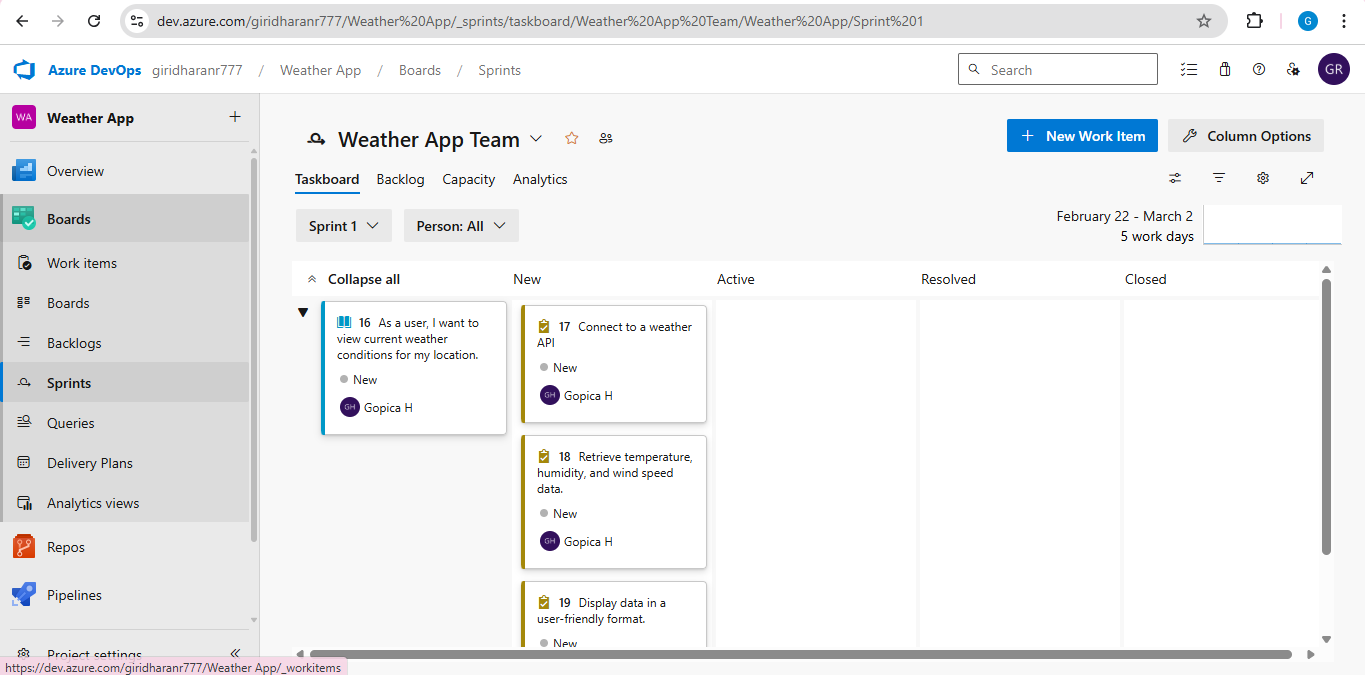
To assign user story to specific sprint for Weather App Project.

**Sprint Planning**

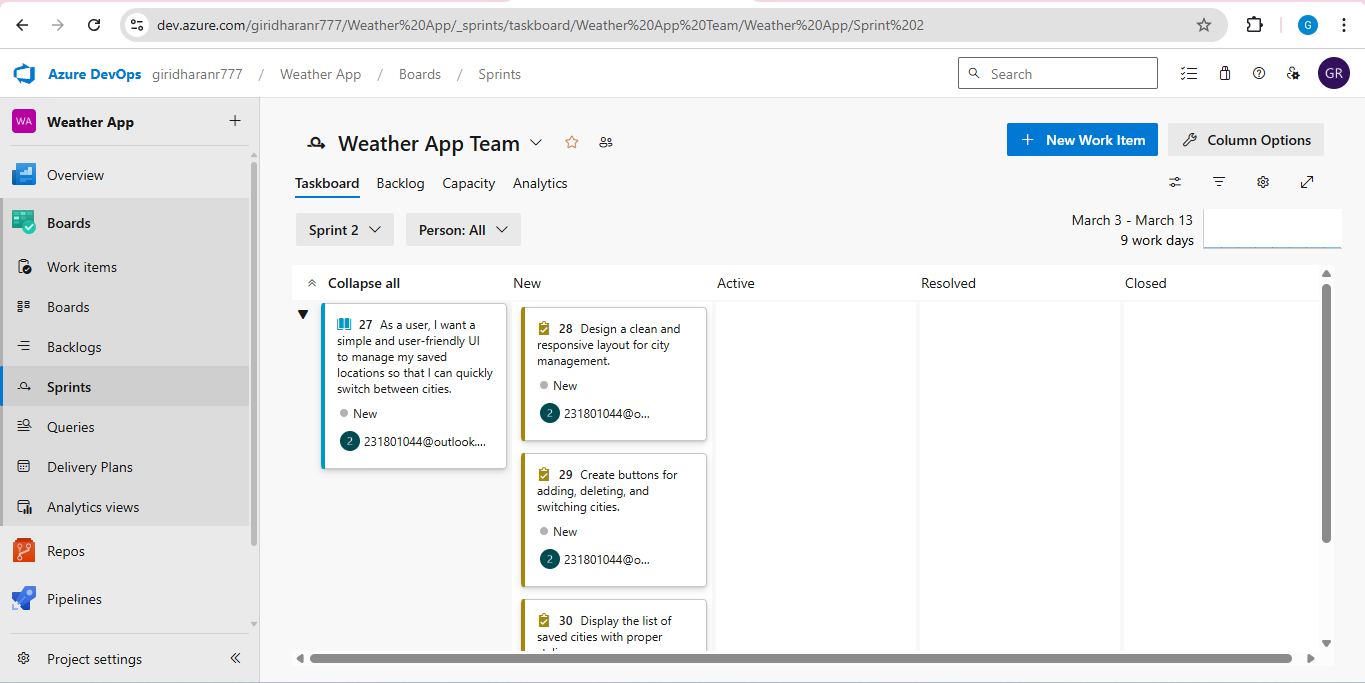
**Sprint 1**



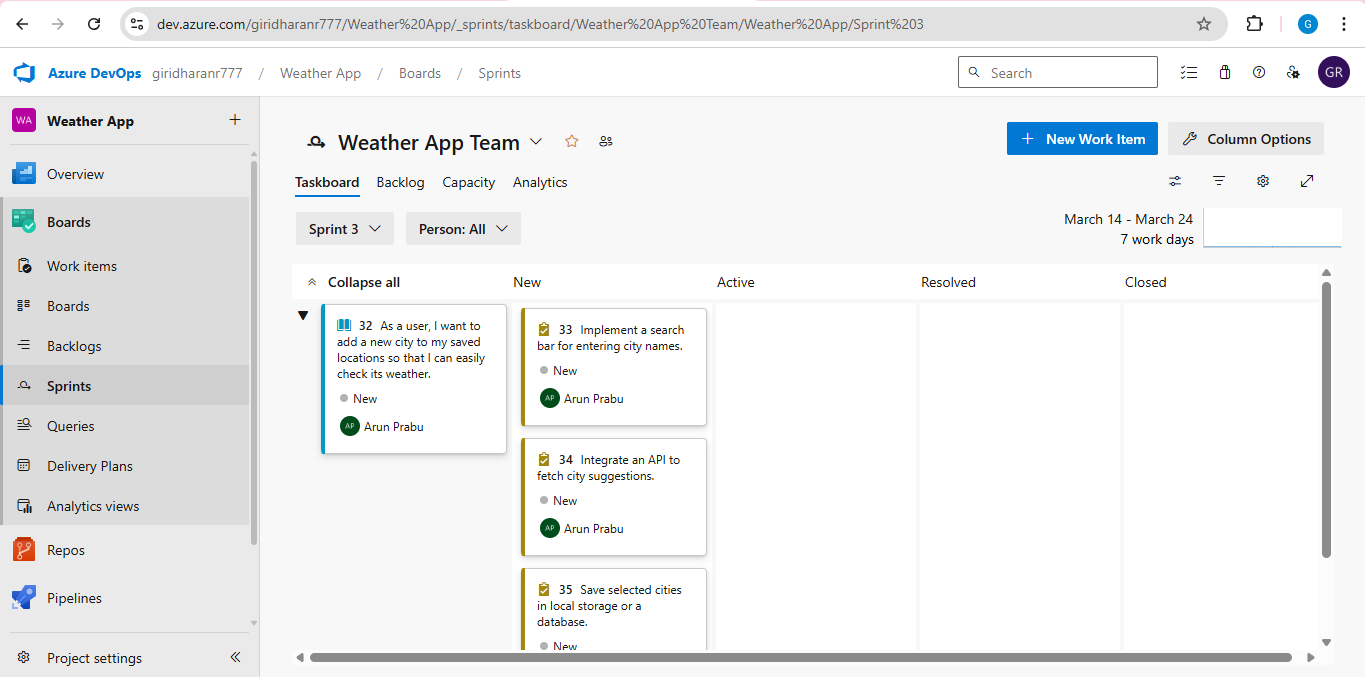
**Sprint 2**



**Sprint 3**



**Sprint 4**



**Result:**

The Sprints are created for the Weather App Project.

**EXP NO:** **5**

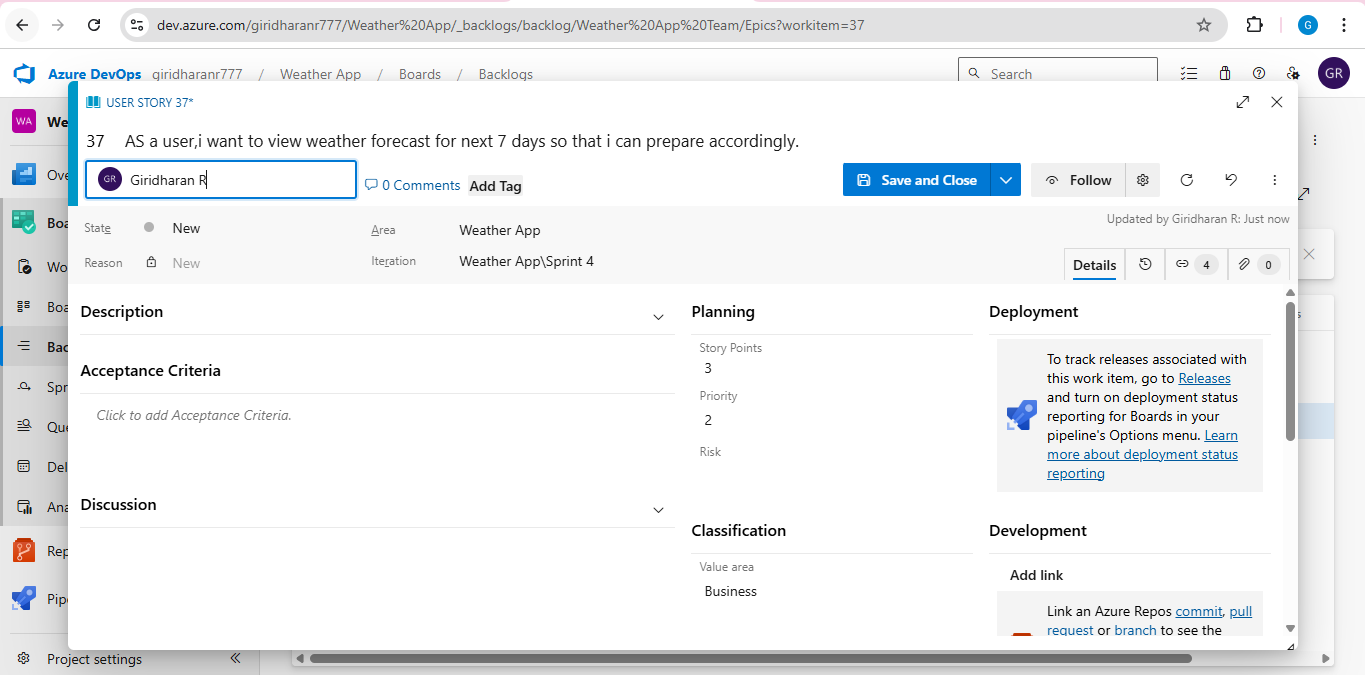
**POKER ESTIMATION**

Create Epic, Features, User Stories, Task

**Aim:**

Create Poker Estimation for the user stories Weather App Project.

**Poker Estimation**



**Result:**

The Estimation/Story Points is created for the project using Poker Estimation.

**EXP NO:** **6**

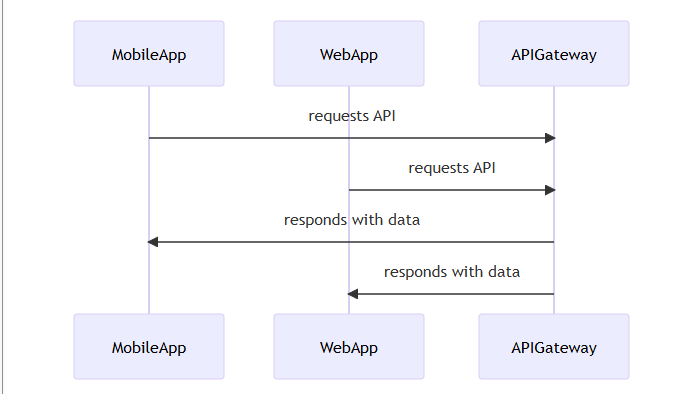
**DESIGNING CLASS AND SEQUENCE DIAGRAMS FOR PROJECT ARCHITECTURE**

Create Epic, Features, User Stories, Task

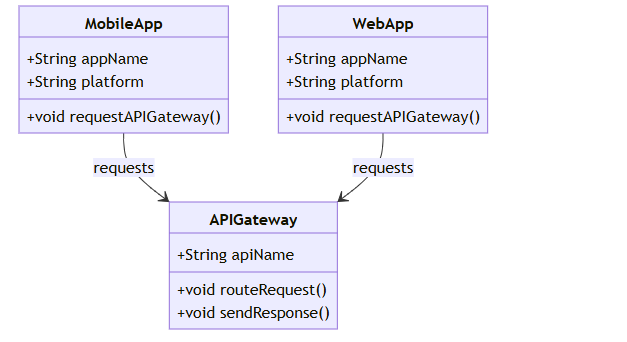
**Aim:**

To Design a Class Diagram and Sequence Diagram for the given Project.

**6A. Class Diagram**



**6B. Sequence Diagram**



**Result:**

The Class Diagram and Sequence Diagram is designed Successfully for the Weather App.

**EXP NO:** **7**

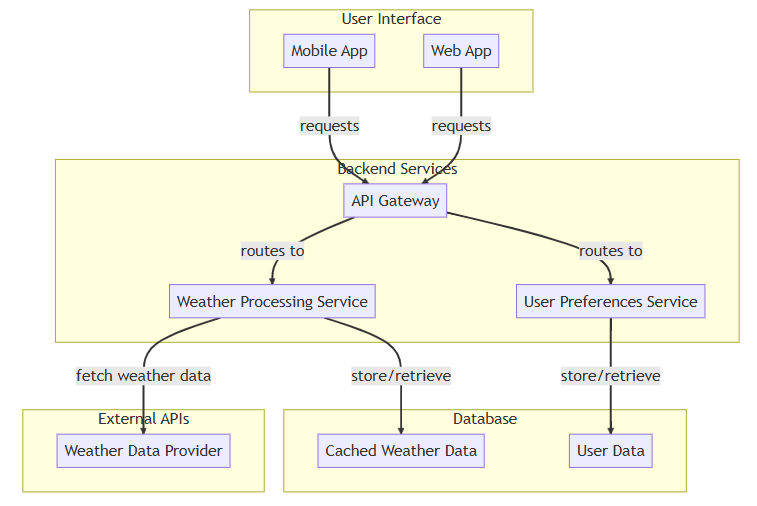
**DESIGNING ARCHITECTURAL AND ER DIAGRAMS FOR PROJECT STRUCTURE**

Create Epic, Features, User Stories, Task

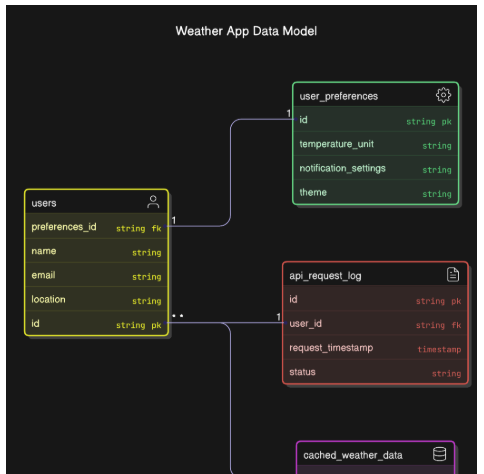
**Aim:**

To Design an Architectural Diagram and ER Diagram for the given Project.

**7A. Architectural Diagram**



**7B.ER Diagram**



**Result:**

The Architecture Diagram and ER Diagram is designed Successfully for the Weather App.

**EXP NO:** **8**

**TESTING – TEST PLANS AND TEST CASES**

Create Epic, Features, User Stories, Task

**Aim:**

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

**Test Planning and Test Case**

**Test Case Design Procedure**

1. **Define User Interactions**

User Signup & Login

Viewing and Managing Favorite Locations

Fetching Real-time Weather Data

Editing Weather Dashboards (rename, reorder, customize)

Creating Smart Weather Alerts based on conditions (temperature, humidity, location, etc.)

1. **Design Happy Path Test Cases**

Focused on validating that all features function as expected under normal conditions.

Example: User logs in successfully, checks the current weather for a city, or saves a favorite city for quick access.

1. **Design Error Path Test Cases**

Simulate negative or unexpected scenarios to test robustness and error handling.

Example: Login fails with invalid credentials, save fails when offline, no recommendations found.

1. **Break Down Steps and Expected Results**

Each test case contains step-by-step actions and a corresponding expected outcome.

Ensures clarity for both testers and automation scripts.

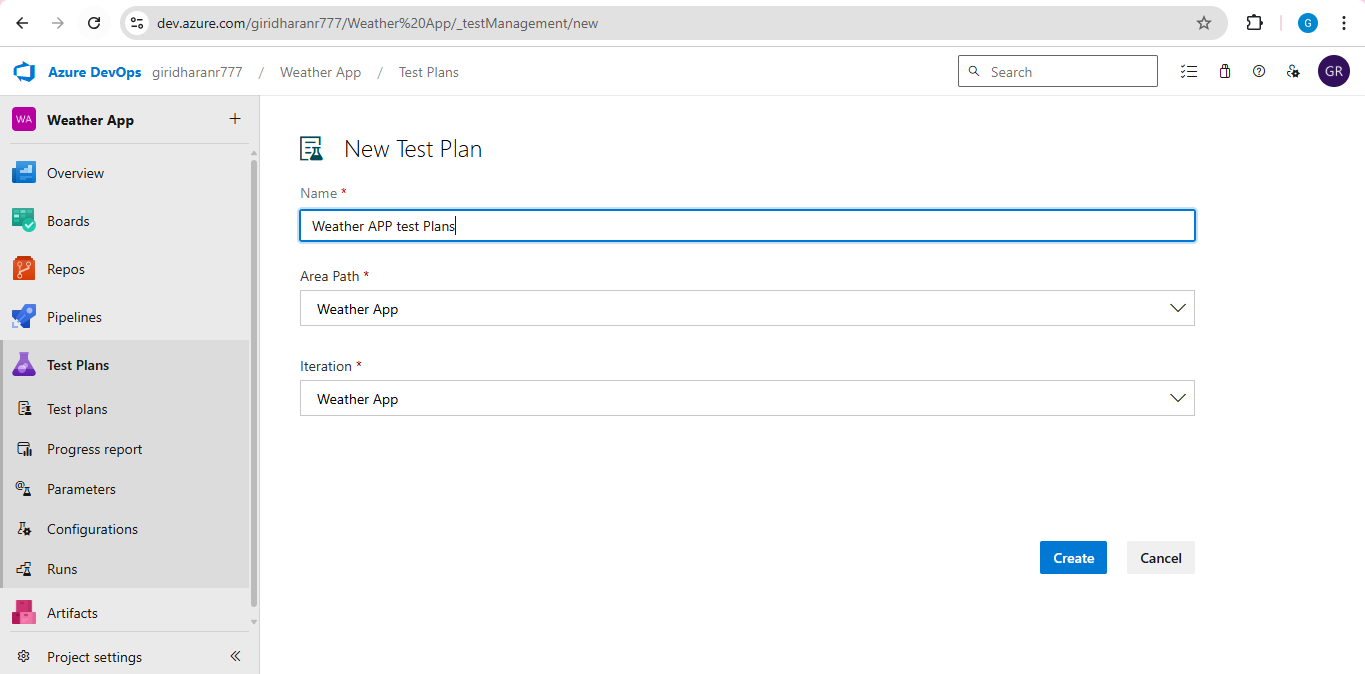
1. **Use Clear Naming and IDs**

Test cases are named clearly (e.g., TC01 – Successful Login, TC10 – Forecast).

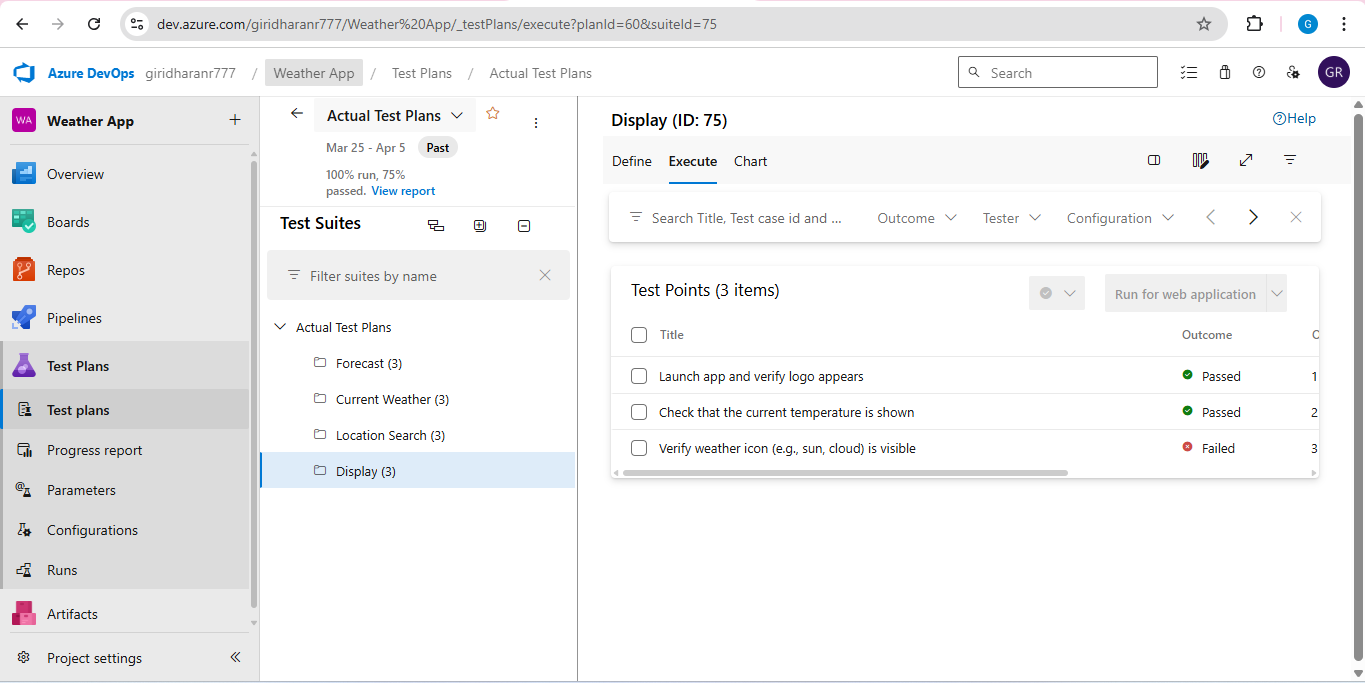
Helps in quick identification and linking to user stories or features.

1. **Separate Test Suites**
   * Grouped test cases based on functionality (e.g., Login, Current Weather , Recommendation System).
   * Improves organization and test execution flow in Azure DevOps.
2. **Prioritize and Review**
   * Critical user actions are marked high-priority.
   * Reviewed for completeness and traceability against feature requirements.

**1.New test plan**



**2.Test suite**

****

**3.Test case**

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

Weather App– Test Plans

**USER STORIES**

• As a user, I want to sign up and log in securely so that I can access my saved cities and settings (ID: 79).  
• As a user, I need to see all my saved cities' weather in one place (ID: 76).  
• As a user, I should be able to add a new city to my favorites list as needed (ID: 73).  
• As a user, I should be able to rename, remove, or reorder my saved cities (ID: 68).  
• As a user, I need to have real-time weather updates and alerts (ID: 65).

**Test Suites**

**Test Suit: TS01 - User Login (ID: 86)**

1. **TC01 – Successful Sign Up**
   * **Action:**▪ Go to the Sign-Up page.  
     ▪ Enter valid name, email, and password.  
     ▪ Click "Sign Up".
   * **Expected Results:**▪ Sign-Up form is displayed.  
     ▪ Fields accept values without error.  
     ▪ Account is created, and the user is redirected to the Weather Dashboard**.**
   * **Type:** Happy Path
2. **TC02 – Secure Login**
   * **Action:**▪ Go to the Login page.  
     ▪ Enter valid email and password.  
     ▪ Click on "Login".
   * **Expected Results:**▪ Login form is displayed.  
     ▪ Fields accept data without error.  
     ▪ User is logged in and redirected to the Weather Dashboard.
   * **Type:** Happy Path
3. **TC03 – Sign Up with Existing Email**
   * **Action:**▪ Go to the Sign-Up page.  
     ▪ Enter a name and an already registered email.  
     ▪ Click on "Sign Up".
   * **Expected Results:**▪ Fields accept data.  
     ▪ Error message "Email already registered" is displayed.
   * **Type:** Error Path
4. **TC04 – Login with Wrong Password**
   * **Action:**▪ Go to the Login page.  
     ▪ Enter valid email and incorrect password.  
     ▪ Click on "Login".
   * **Expected Results:**▪ Input is accepted.  
     ▪ Error message "Invalid username or password" is shown**.**
   * **Type:** Error Path

**Test Suit: TS02 - View Saved Cities (ID: 87)**

1. **TC05 – View Saved Cities Page**
   * **Action:**▪ Log in successfully.  
     ▪ Navigate to "My Cities" section.
   * **Expected Results:**▪ All saved cities and their current weather data are displayed clearly.
   * **Type:** Happy Path
2. **TC06 – Saved Cities Loading Failure**
   * **Action:**▪ Disconnect from the internet.  
     ▪ Navigate to "My Cities".
   * **Expected Results:**▪ Network is offline.  
     ▪ Error message "Unable to load saved cities" is shown**.**
   * **Type:** Error Path

**Test Suit: TS03 - Real-Time Weather Updates (ID: 88)**

1. **TC07 – Real-Time Weather Display**
   * **Action:**▪ Open a city's weather page.  
     ▪ Observe the weather data and updates.
   * **Expected Results:**▪ Weather data (temperature, humidity, wind speed, etc.) is displayed and updates in real time.
   * **Type:** Happy Path
2. **TC08 – Weather Data Not Updating**
   * **Action:**▪ Open a city's weather page.  
     ▪ Simulate server delay or offline mode.
   * **Expected Results:**▪ Weather data remains static or shows fallback/default message.
   * **Type:** Error Path

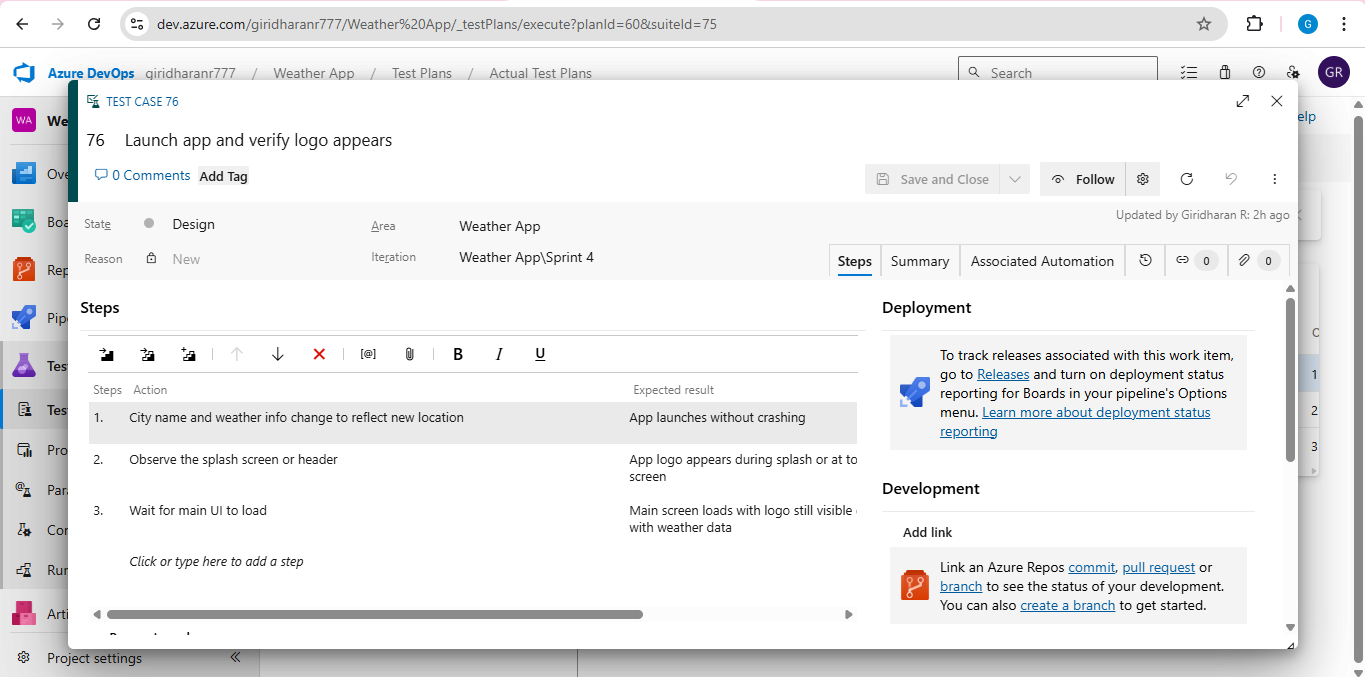
**Test Suit: TS04 - Manage Saved Cities (ID: 89)**

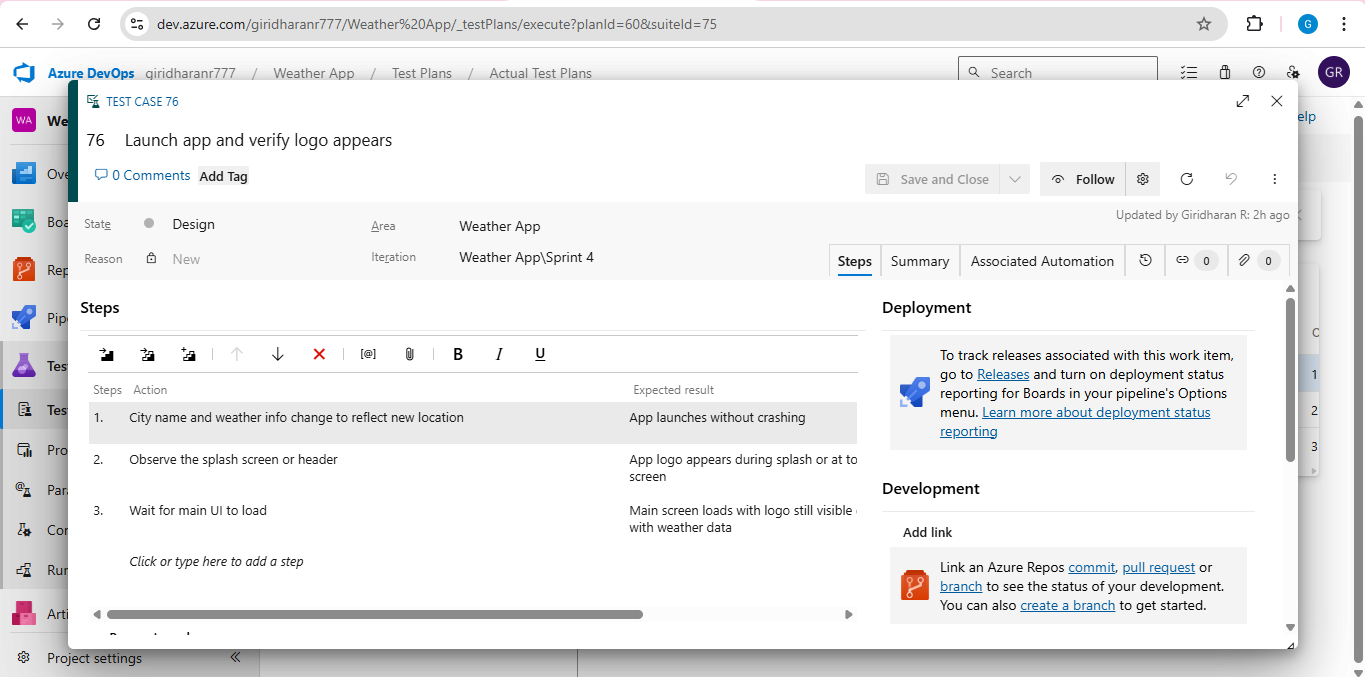
1. **TC09 – Rename Saved City Successfully**
   * **Action:**▪ Navigate to "My Cities".  
     ▪ Click "Rename" next to a saved city.  
     ▪ Enter a new name and click "Save".
   * **Expected Results:**▪ City name updates successfully.
   * **Type:** Happy Path
2. **TC10 – Rename City with Blank Name**
   * **Action:**▪ Click "Rename" on a saved city.  
     ▪ Leave the field blank.  
     ▪ Click "Save".
   * **Expected Results:**▪ Error message "City name cannot be empty" is shown.
   * **Type:** Error Path
3. **TC11 – Change Cities Order**
   * **Action:**▪ Open "My Cities".  
     ▪ Drag and drop cities to reorder.  
     ▪ Click "Save".
   * **Expected Results:**▪ Saved cities' order is updated and saved.
   * **Type:** Happy Path
4. **TC12 – Change Cities Order Fails**
   * **Action:**▪ Login and go to “My Cities”.  
     ▪ Select a city.  
     ▪ Go offline or simulate server error.  
     ▪ Reorder cities and click “Save Order”.
   * **Expected Results:**▪ Error message: "Failed to update order. Please check your connection".
   * **Type:** Error Path

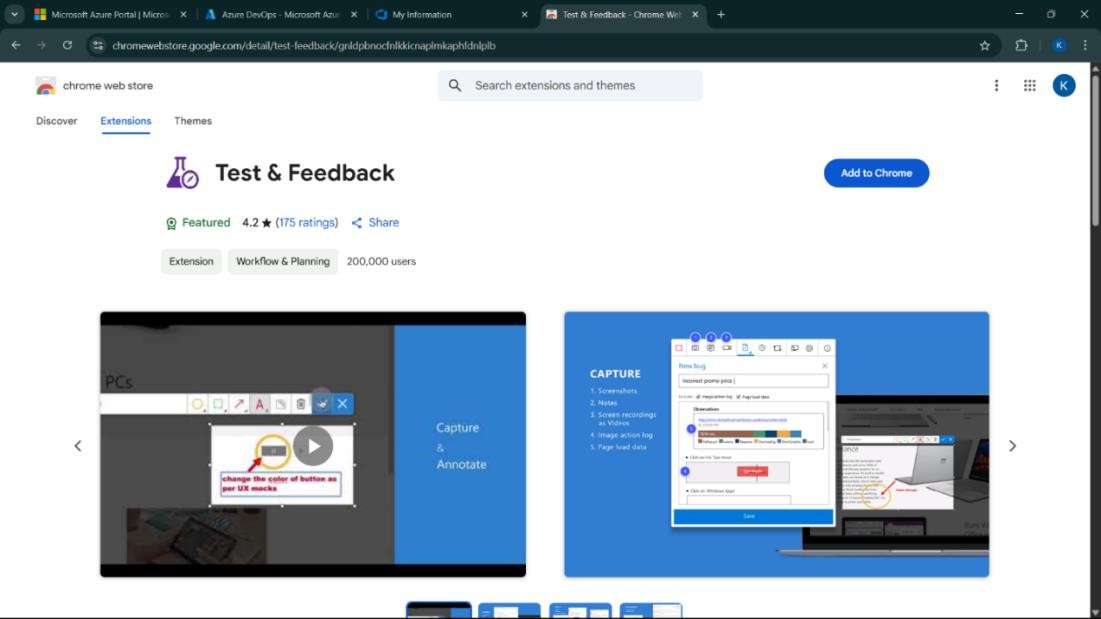
**Test Suit: TS05 - Smart Weather Alerts (ID: 90)**

1. **TC13 – Generate Weather Alerts Based on Conditions**
   * **Action:**▪ Login with valid credentials.  
     ▪ Click on "Set Weather Alerts".  
     ▪ Select conditions like rain, storm, or temperature threshold.  
     ▪ Click "Save Alerts".
   * **Expected Results:**▪ Weather alerts are generated and saved based on selected conditions.
   * **Type:** Happy Path
2. **TC14 – Fail to Set Weather Alerts Due to Missing Selection**
   * **Action:**▪ Login with valid credentials.  
     ▪ Click on "Set Weather Alerts".  
     ▪ Leave alert conditions blank or select invalid options.  
     ▪ Click "Save Alerts".
   * **Expected Results:**▪ Error message: "Please select at least one valid alert condition".
   * **Type:** Error Path

**Test Cases**

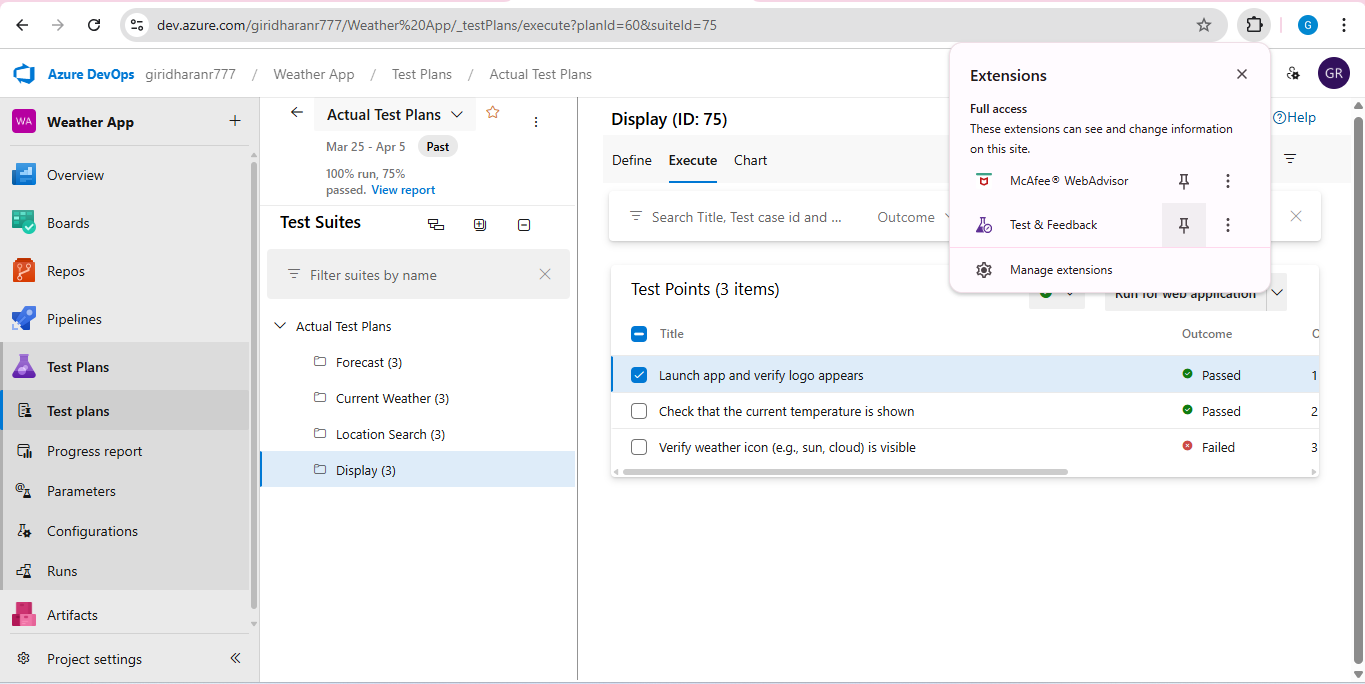
****

  
**4.Installation of test**

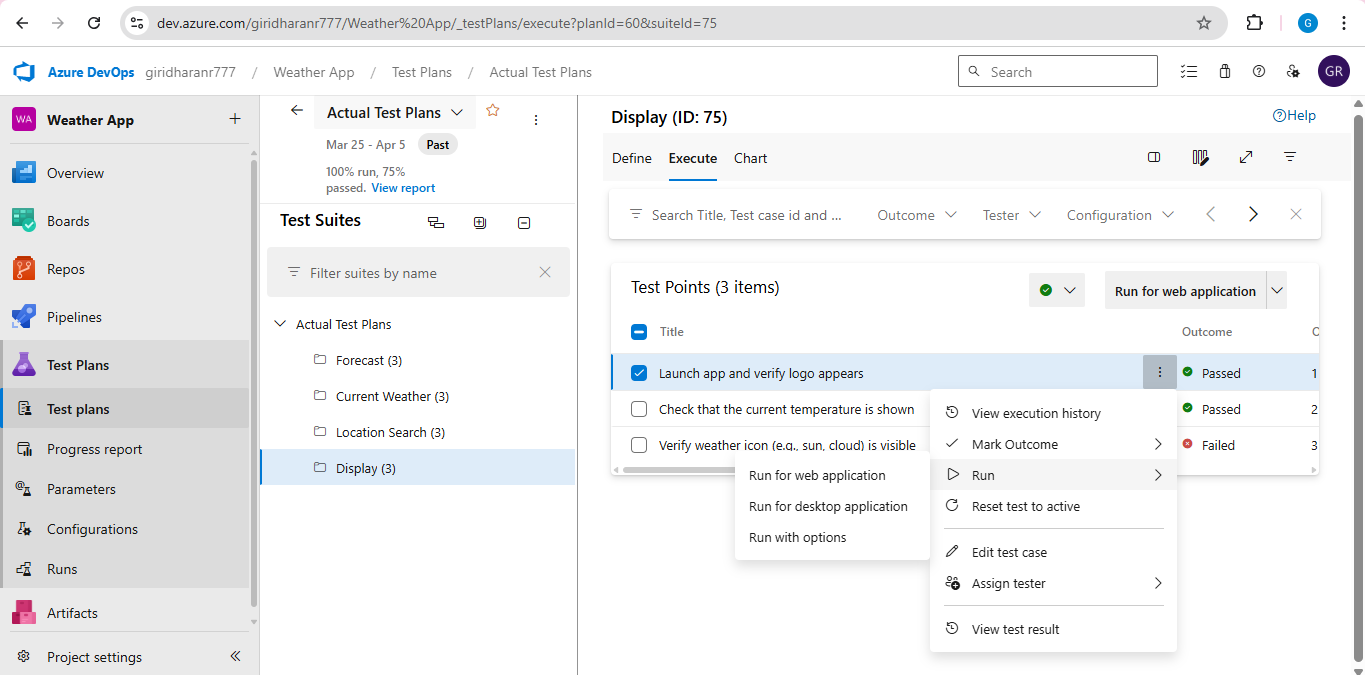
****

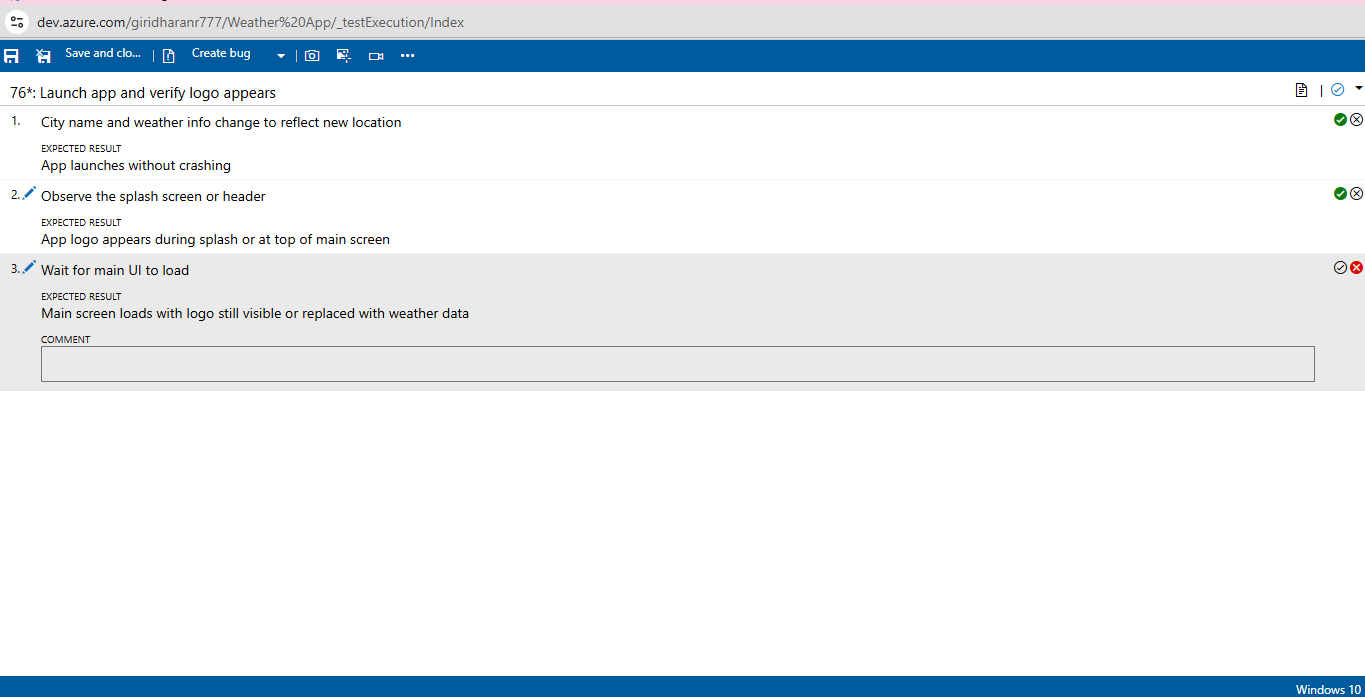
Test and feedback

Showing it as an extension

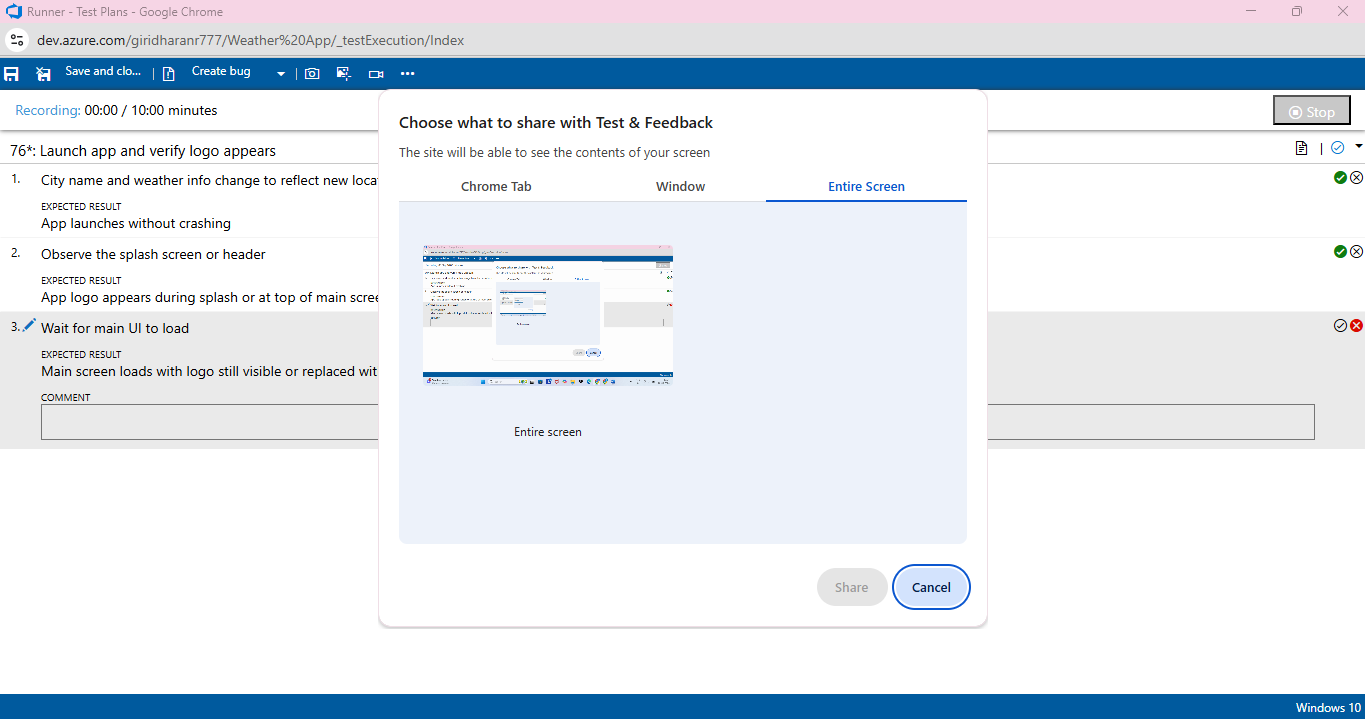


**5.Running the test cases**

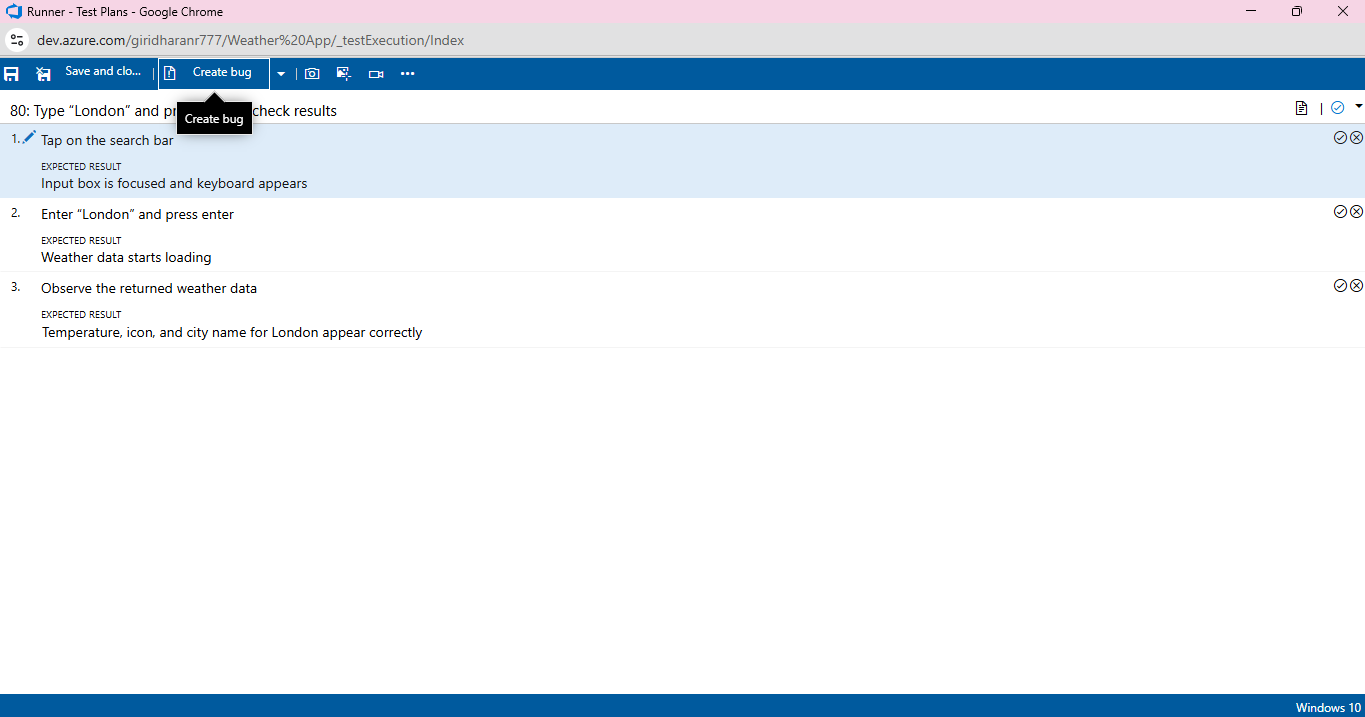
****

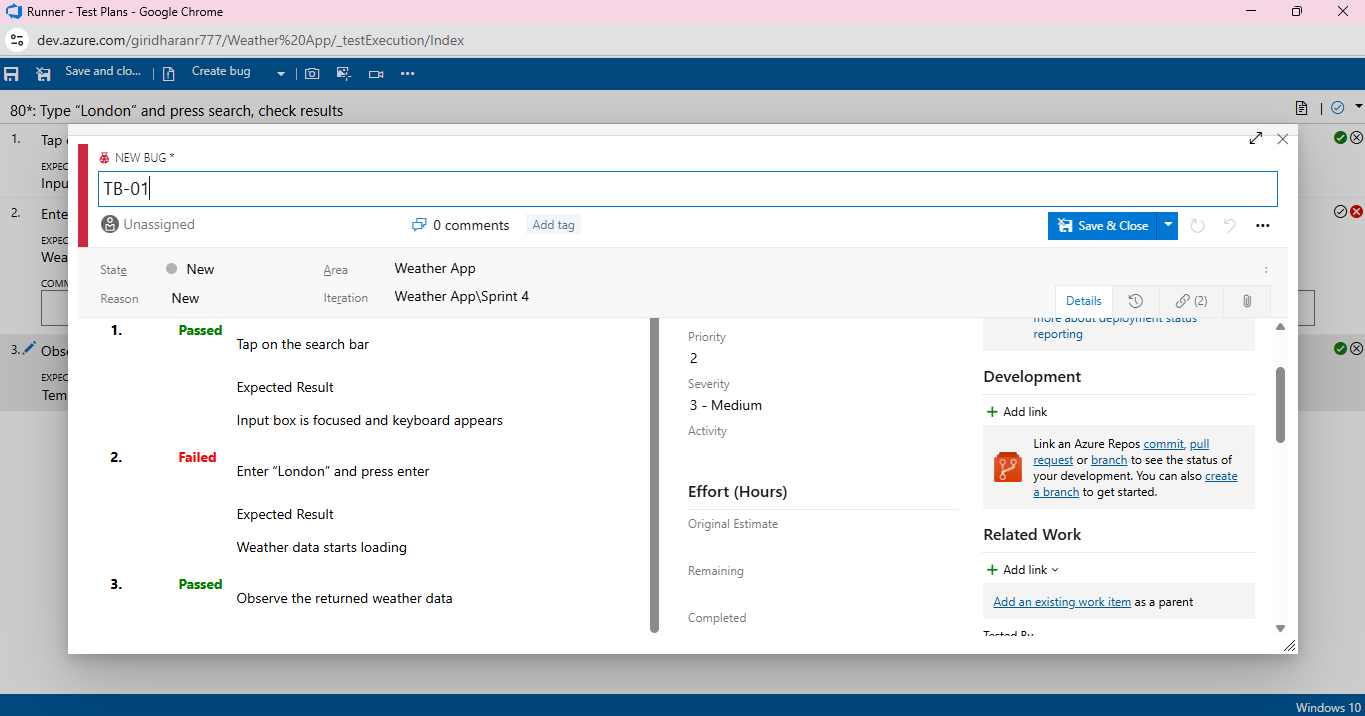
****

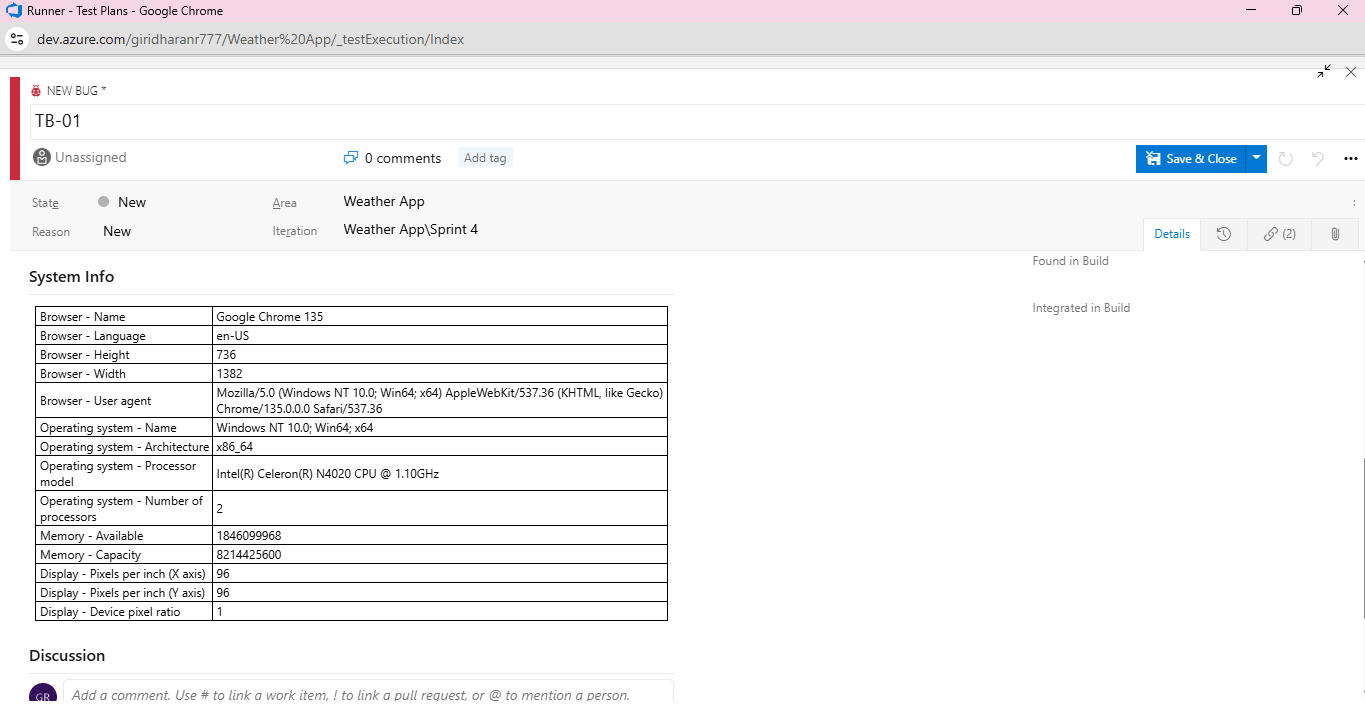
**6.Recording the test case**

****

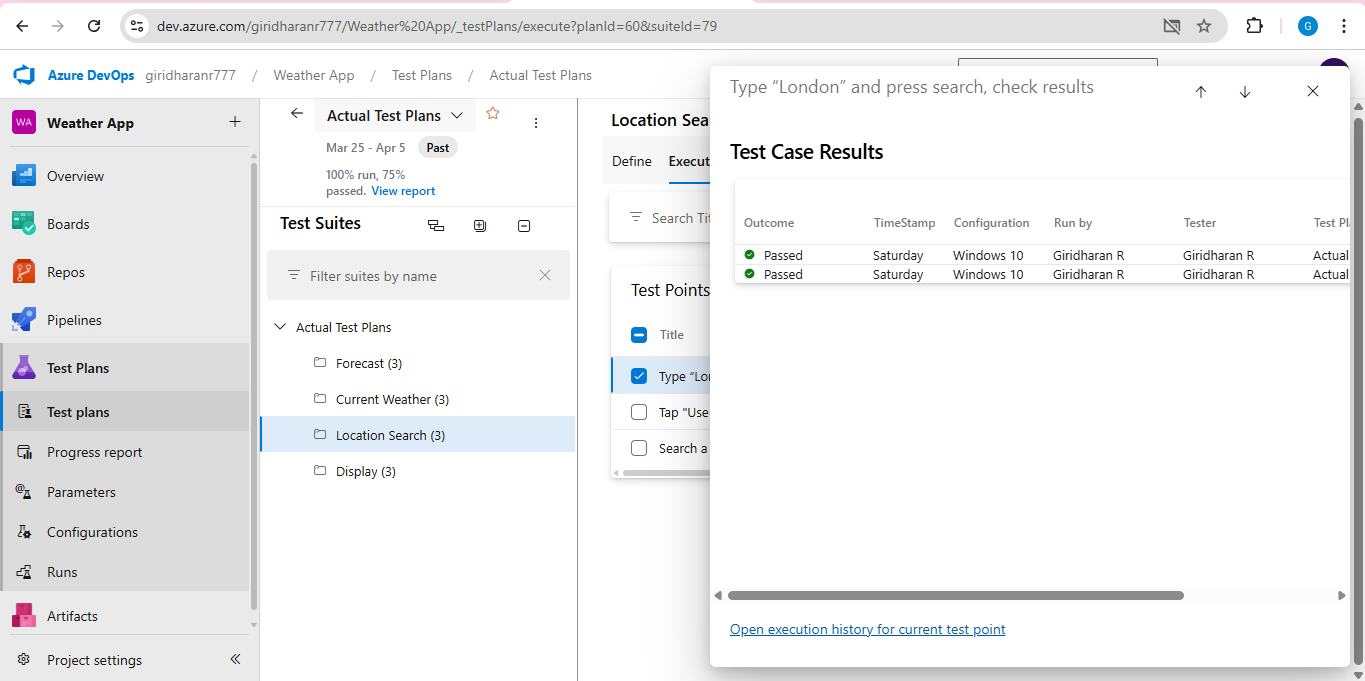
**7.Creating the bug**

****

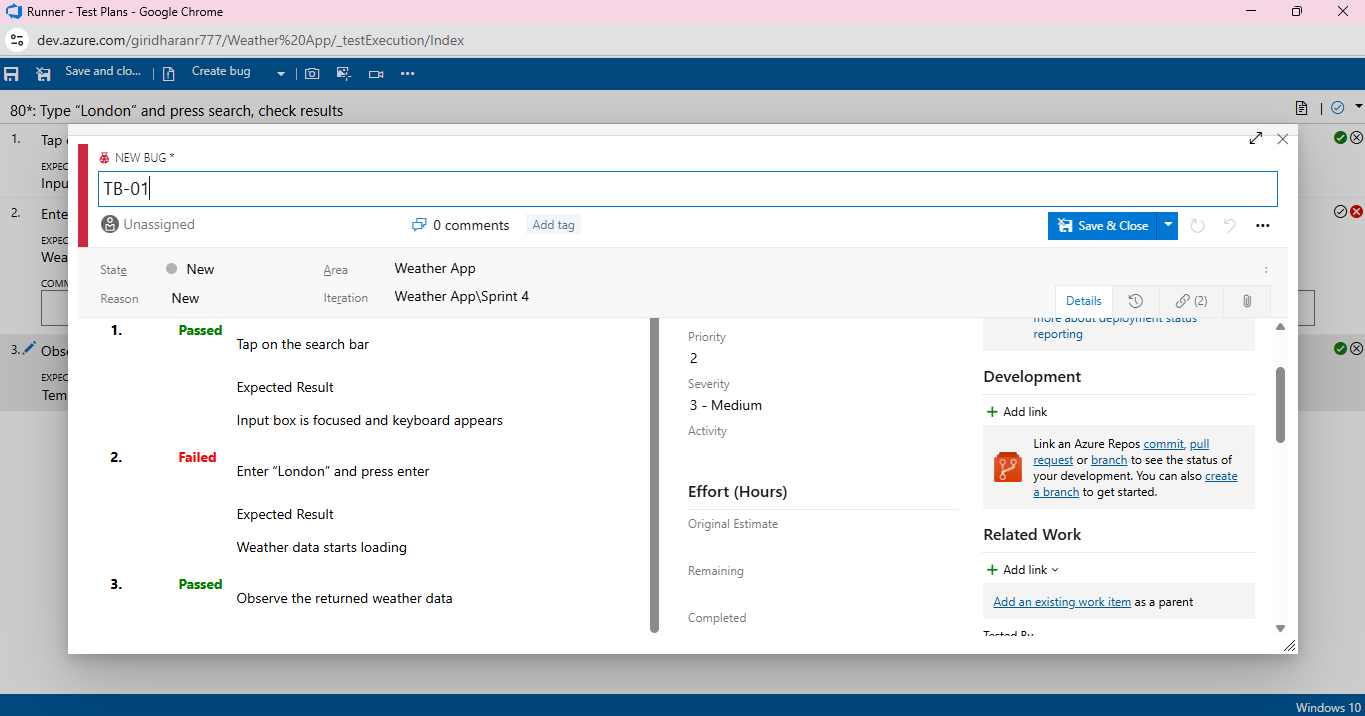


****

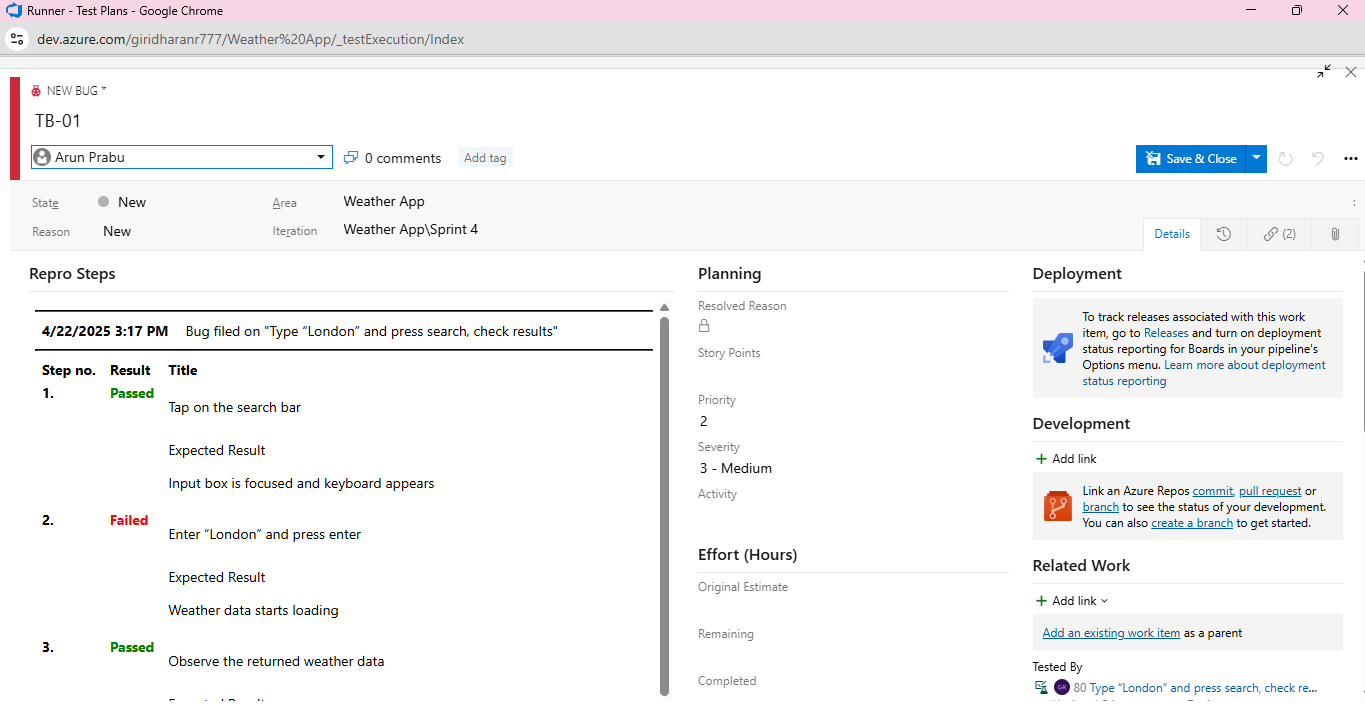
**8.Test case results**

****

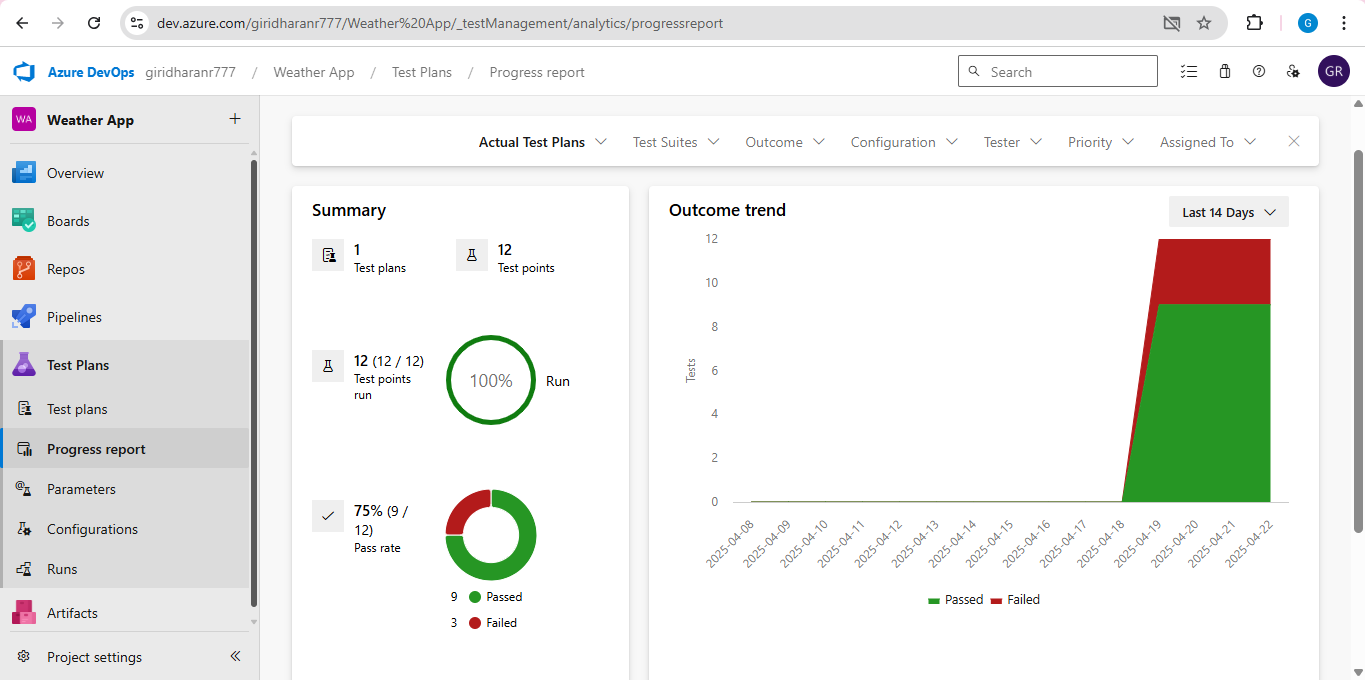
**9.Test report summary**



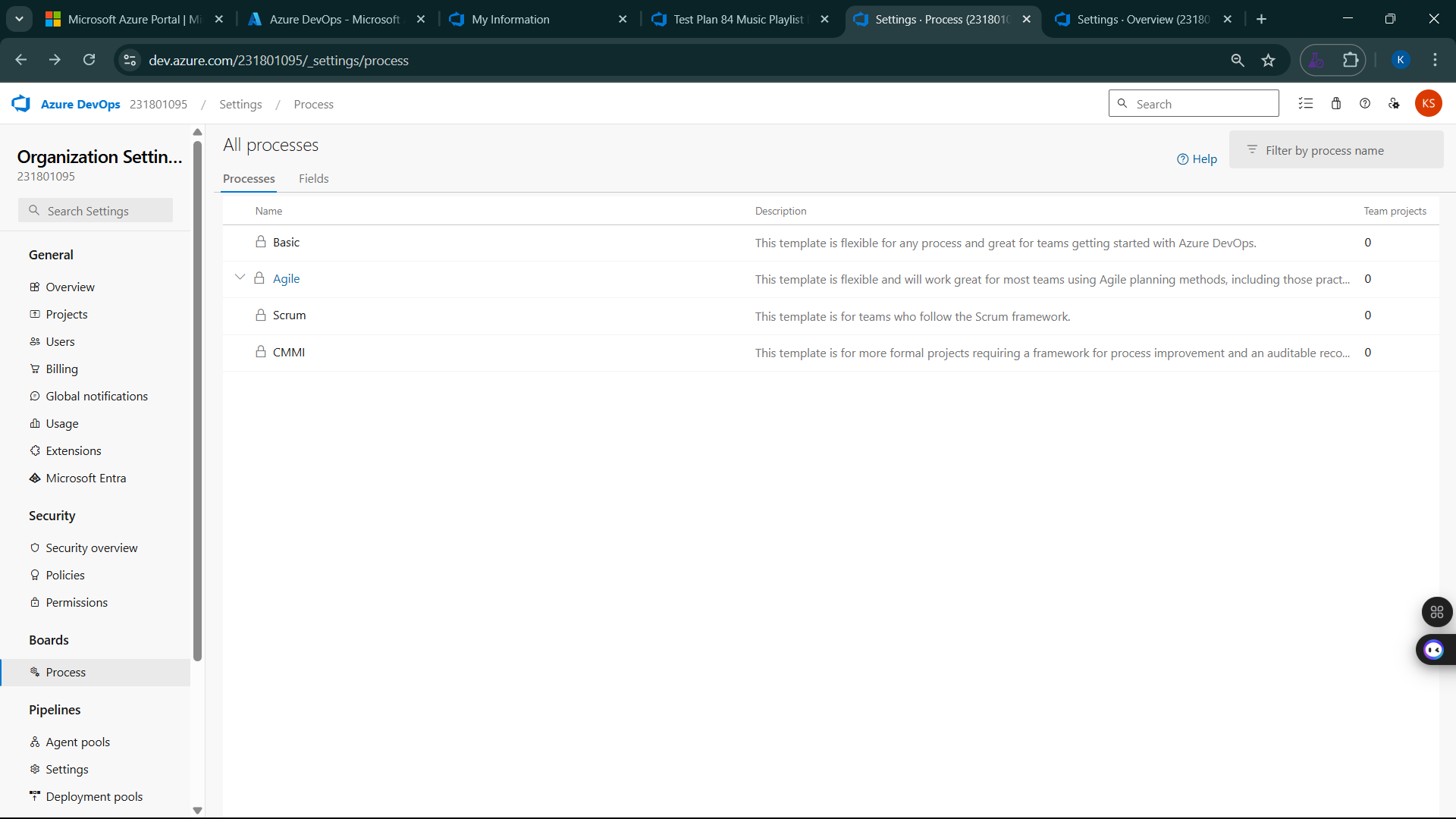
* Assigning bug to the developer and changing state

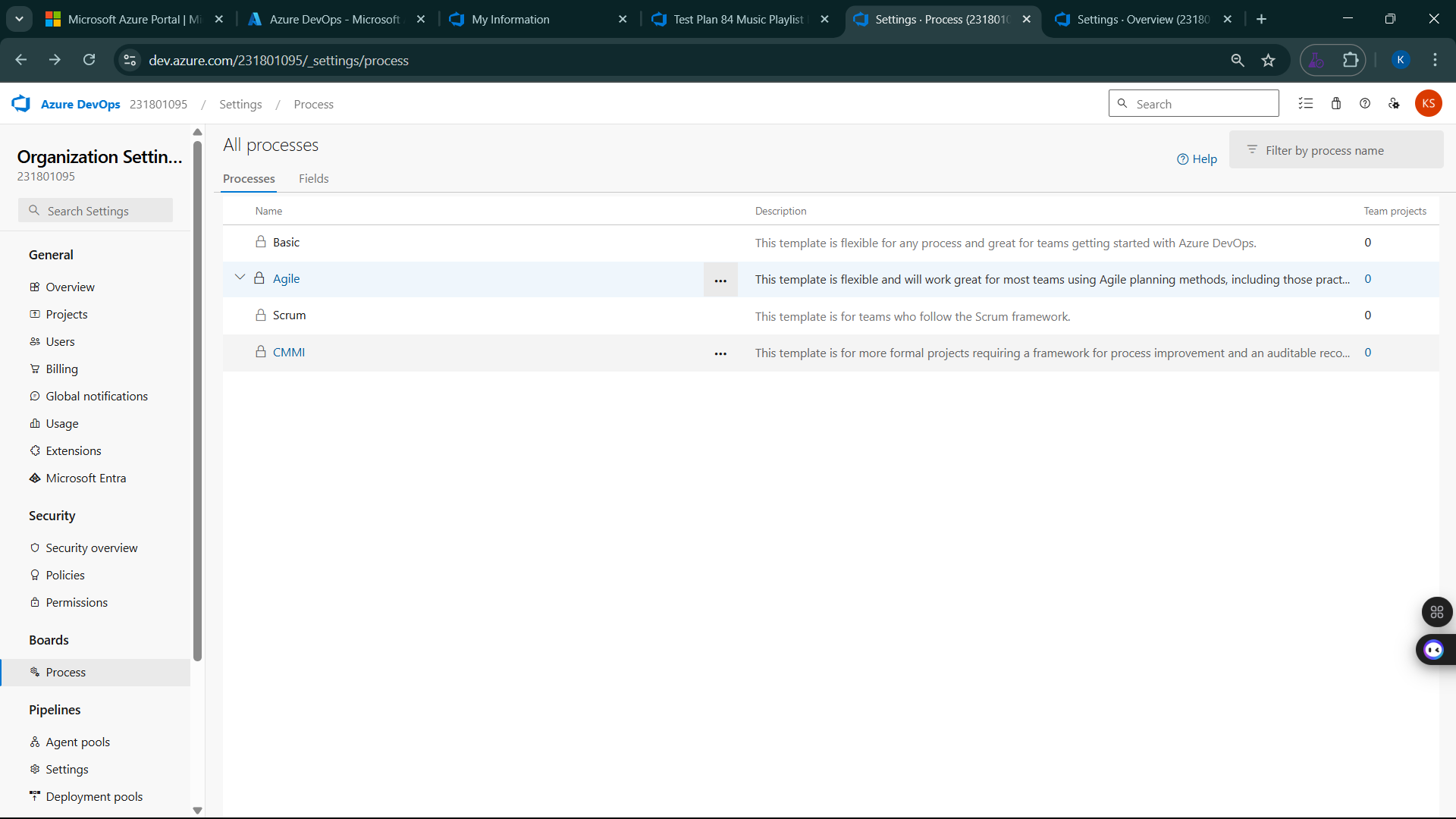


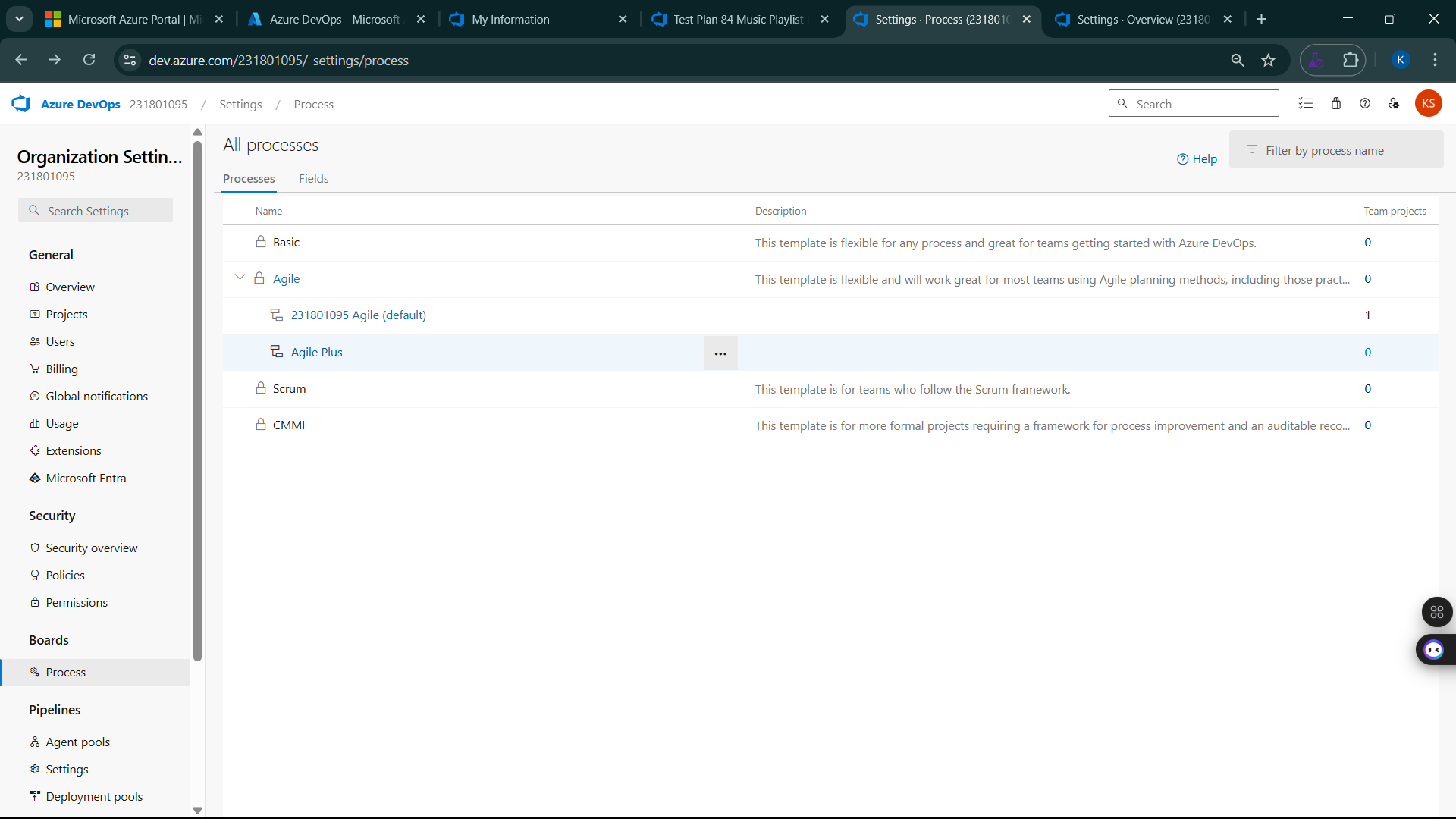
**10.Progress report**

****

**11.Changing the test template**

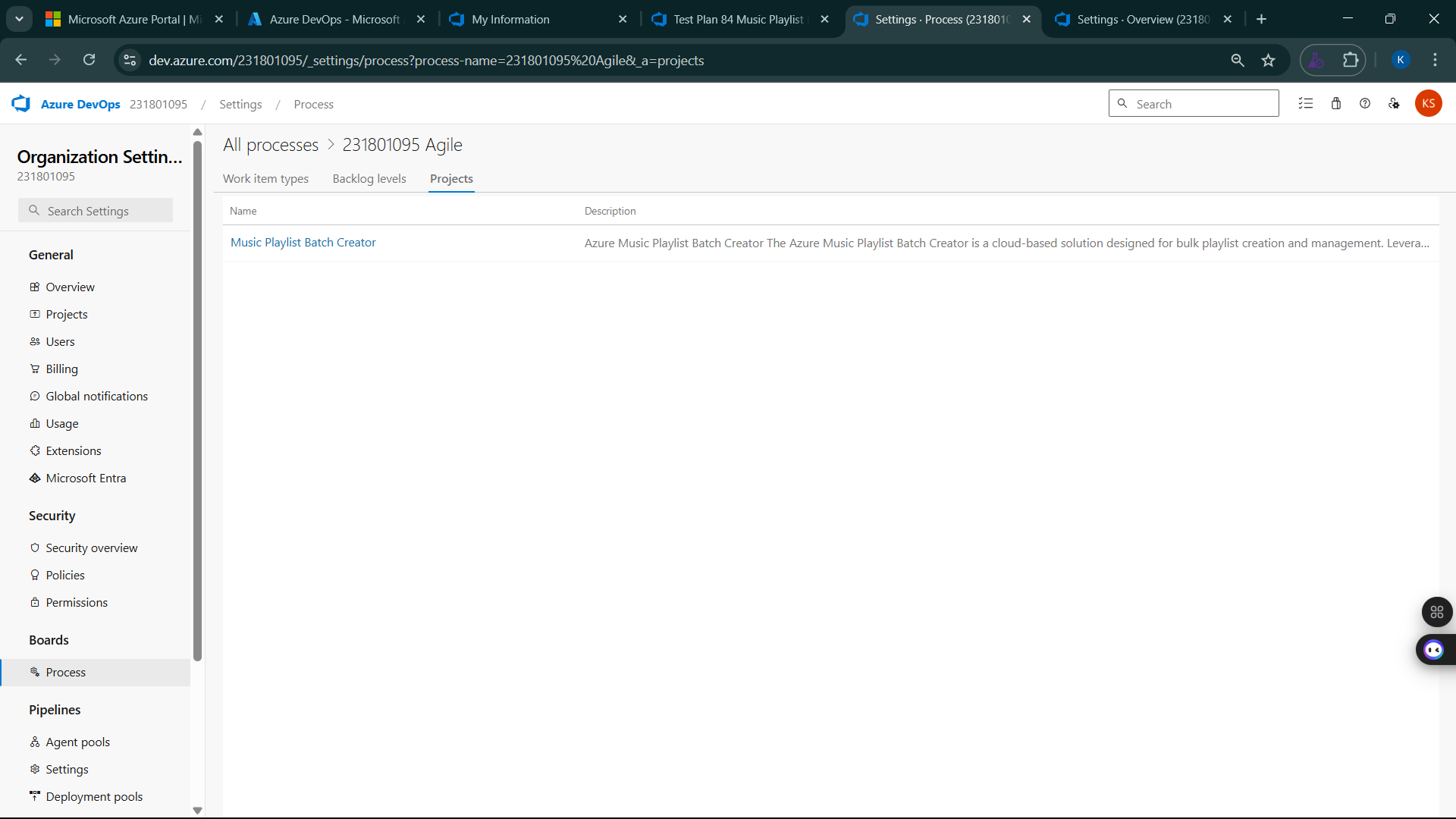


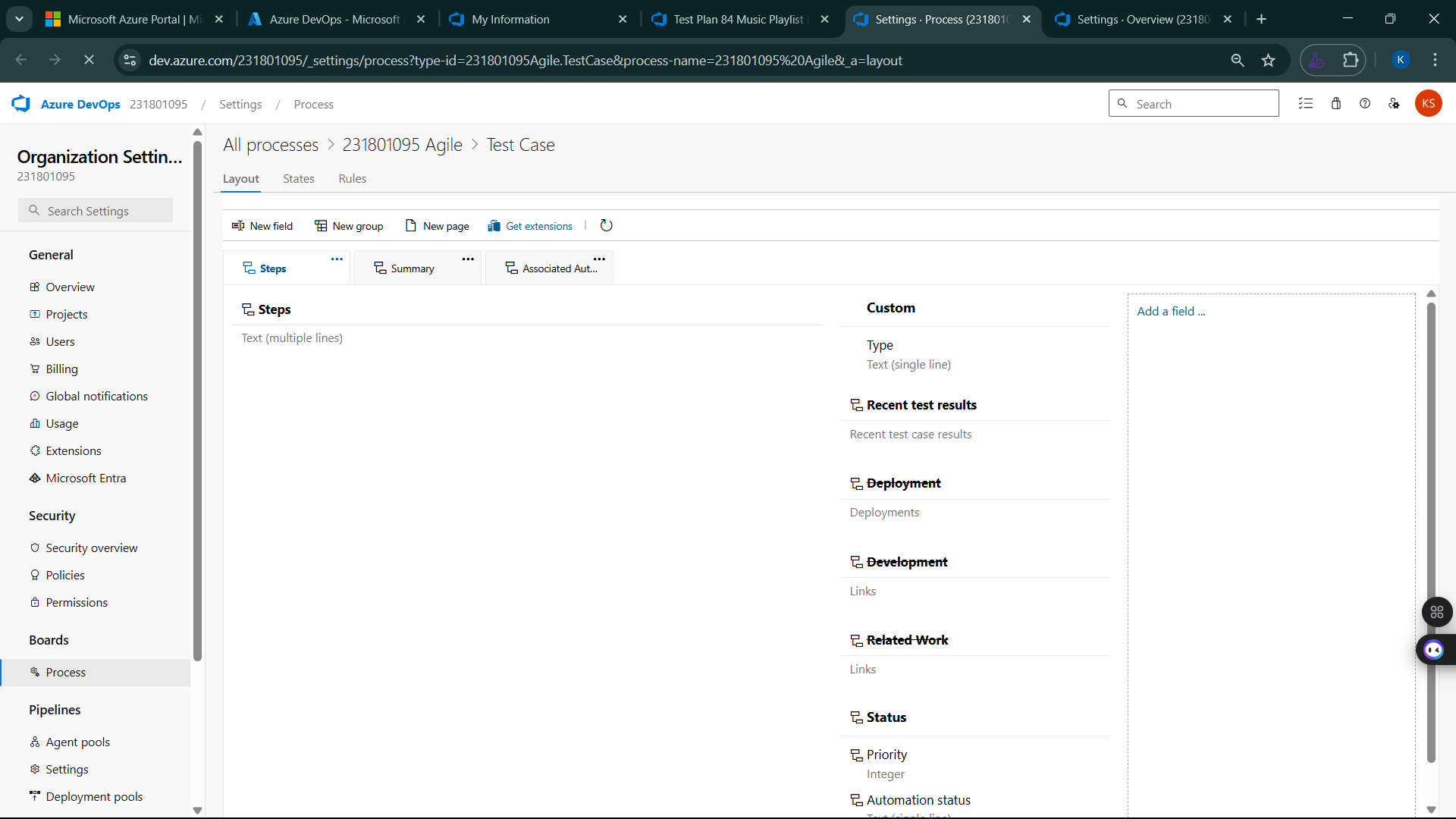




**12.View the new test case template**







**Result:**

The test plans and test cases for the user stories is created in Azure DevOps with Happy Path and Error Path

**EXP NO:** **9**

**LOAD TESTING AND PIPELINES**

Create Epic, Features, User Stories, Task

**Aim:**

To create an Azure Load Testing resource and run a load test to evaluate the performance of a target endpoint and to create and demonstrate an Azure DevOps pipeline for automating application builds, tests, and deployment.

**Load Testing**

Azure Load Testing:

Azure Load Testing allows you to simulate high traffic and stress tests for your web applications and APIs to understand how they perform under load. It helps identify performance bottlenecks, scalability issues, and optimize resource usage before deployment.

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

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

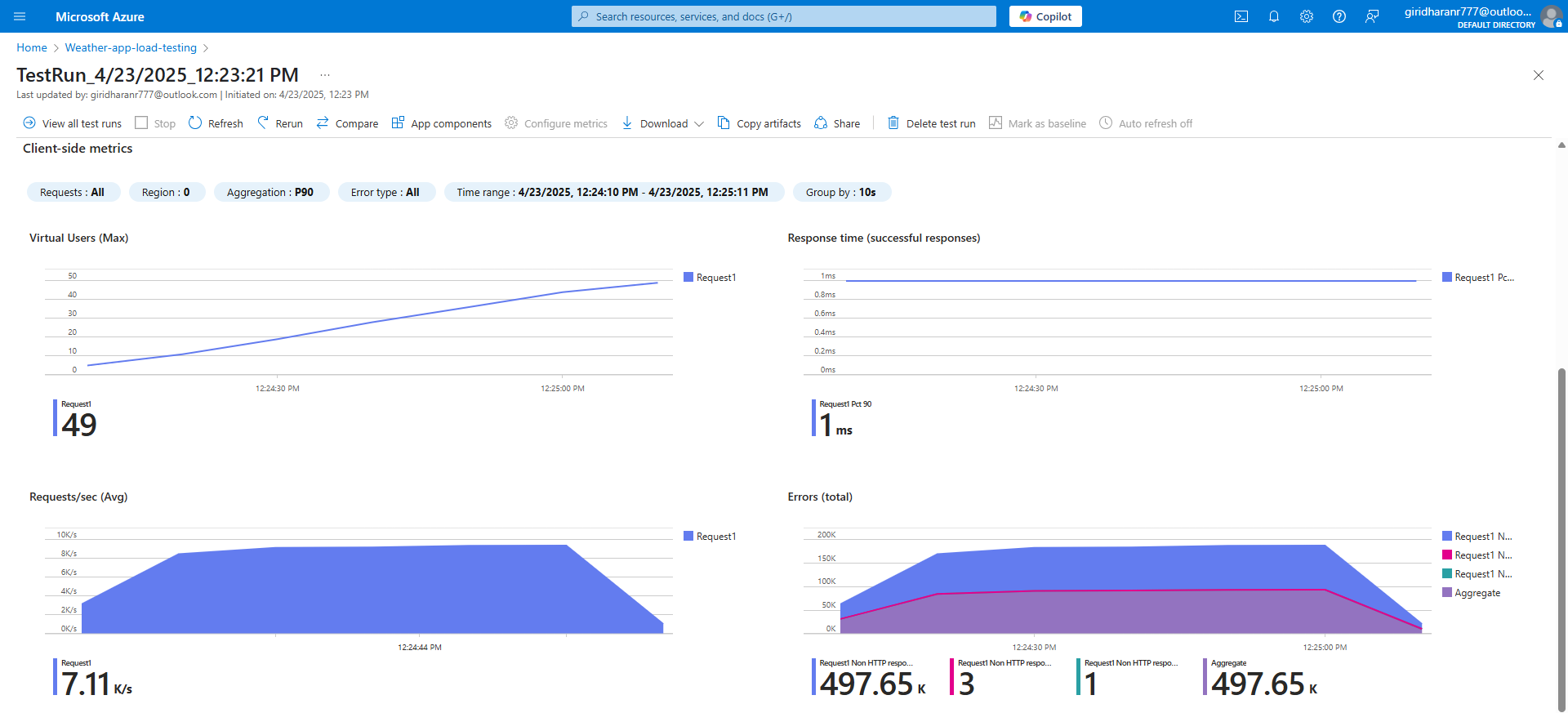
1. Sign in to Azure Portal  
   Go to <https://portal.azure.com> and log in.
2. Create the Resource
   * Go to *Create a resource* → Search for “Azure Load Testing”.
   * Select Azure Load Testing and click Create.
3. Fill in the Configuration Details
   * Subscriptio*n:* Choose your Azure subscription.
   * *Resource* *Group:* Create new or select an existing one.
   * *Name:* Provide a unique name (no special characters).
   * *Location:* Choose the region for hosting the resource.
4. (Optional) Configure tags for categorization and billing.
5. Click Review + Create, then Create.
6. Once deployment is complete, click Go to resource.

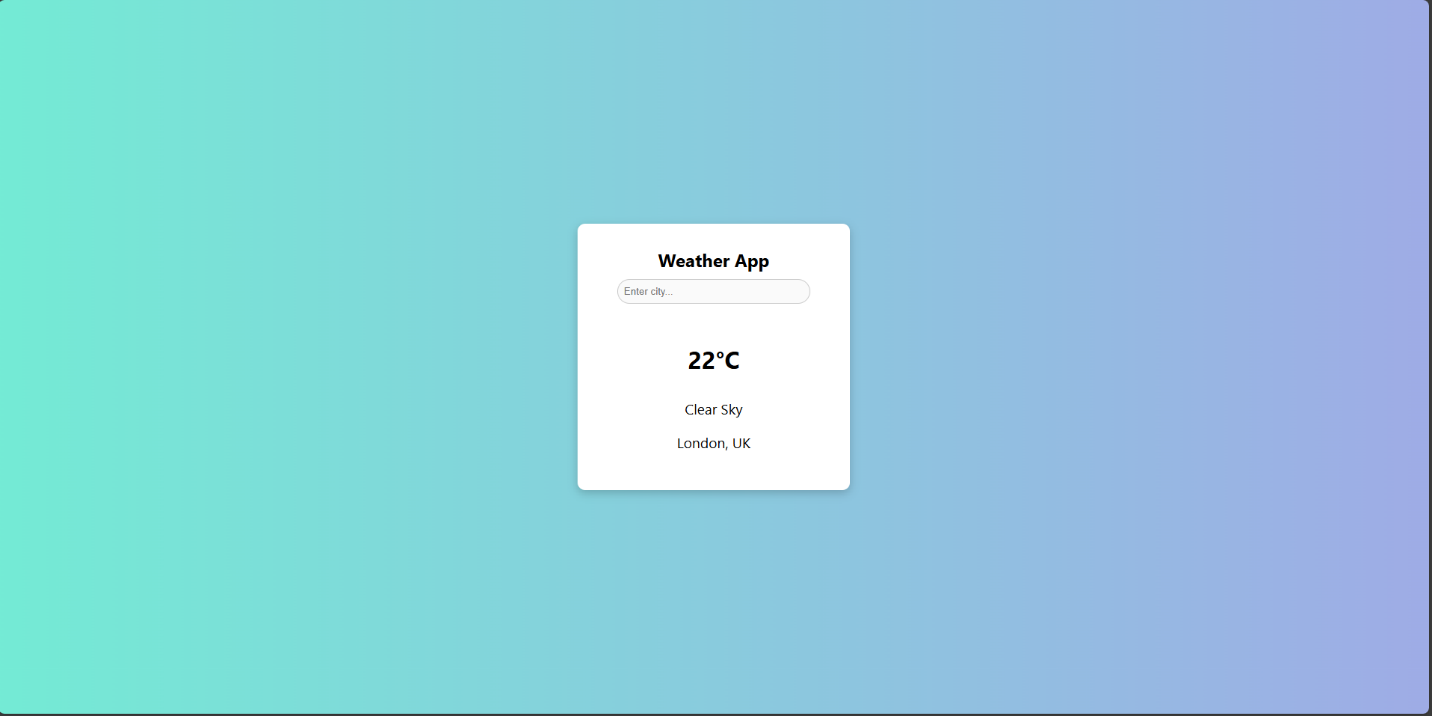
**Steps to Create and Run a Load Test:**

Once your resource is ready:

1. Go to your Azure Load Testing resource and click Add HTTP requests > Create.
2. Basics Tab
   * *Test Name:* Provide a unique name.
   * *Description:* (Optional) Add test purpose.
   * *Run After Creation:* Keep checked.
3. Load Settings
   * *Test URL:* Enter the target endpoint (e.g., https://yourapi.com/products).
4. Click Review + Create → Create to start the test.

**Load Testing**

****



**Pipelines**

**Description:**

This experiment demonstrates how to connect a GitHub-hosted Flask-based Weather App project with Azure DevOps. The pipeline will automatically install dependencies, run basic tests, and publish artifacts. This ensures that every commit triggers checks for reliability and smooth deployment.

Steps:

1. Connect GitHub to Azure DevOps:
   * In Azure DevOps, create a new project.
   * Create a pipeline and select GitHub as the source.
   * Authorize access to your GitHub repository, ensuring that Azure DevOps can pull the repository for your pipeline.
2. Create azure-pipelines.yml in Your Repo Root:
   * In your GitHub repository, create a new file called azure-pipelines.yml in the root directory.
   * Add the following basic pipeline configuration for Python and Flask:

**yml Code**

trigger:

- main # Trigger pipeline when changes are pushed to the main branch

pool:

vmImage: ubuntu-latest # Use a hosted Ubuntu agent

steps:

# Step 1: Checkout the code from GitHub

- checkout: self

# Step 2: Set up Python environment

- task: UsePythonVersion@0

inputs:

versionSpec: '3.x' # Use the latest Python 3.x version

displayName: "Set up Python"

# Step 3: Install dependencies from the correct path

- script: |

python -m pip install --upgrade pip

pip install -r project/requirements.txt # Adjusted path to requirements.txt

displayName: "Install dependencies"

# Step 4: Run a simple Python script to check the environment

- script: |

python -c "print('Weather App')"

displayName: "Run a Python script"

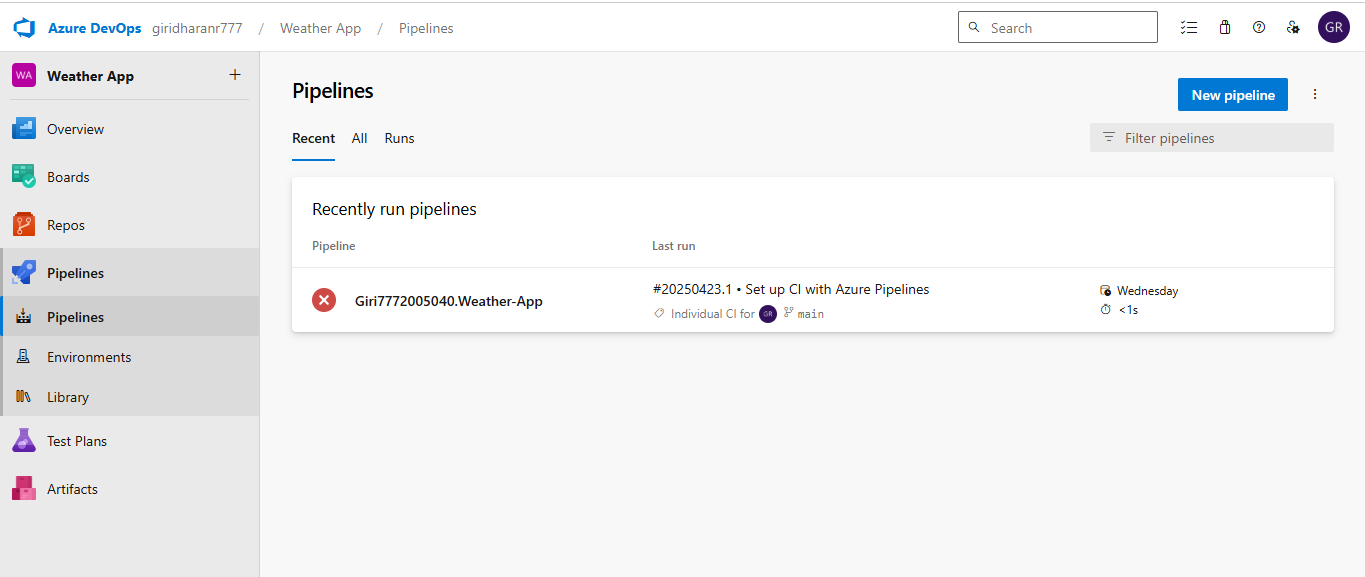
1. Pipeline Tasks Include:

* Setting up the Python environment using the UsePythonVersion task.
* Installing project dependencies from project/requirements.txt. Make sure the path to requirements.txt is correct (it is located under the project folder).
* Running a simple Python script to verify that Python is set up correctly and the pipeline works.

1. Run and Monitor Pipeline:

* Commit changes to the main branch of your repository to trigger the pipeline in Azure DevOps.
* Monitor the logs in the Azure DevOps portal to view logs, errors, or success messages and ensure everything runs smoothly.

**Pipeline**

****

**Result:**

Successfully created the Azure Load Testing resource and executed a load test to assess the performance of the specified endpoint and also demonstrated pipelines in azure devops.

**EXP NO:** **10**

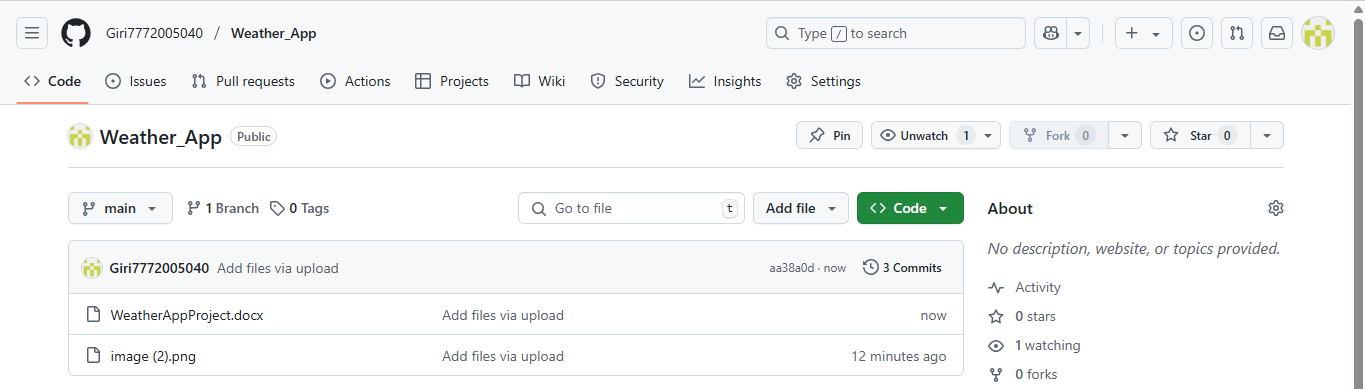
**GITHUB: PROJECT STRUCTURE & NAMING CONVENTIONS**

Create Epic, Features, User Stories, Task

**Aim:**

To provide a clear and organized view of the project's folder structure and file naming conventions, helping contributors and users easily understand, navigate, and extend the Weather App.

**GitHub Project Structure**

****

**Result:**

The GitHub repository clearly displays the organized project structure and consistent naming conventions, making it easy for users and contributors to understand and navigate the codebase.