

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

**CS23432 – Software Construction**

**(REGULATION 2023)**

**RAJALAKSHMI ENGINEERING COLLEGE**

**Thandalam, Chennai-602015**

**Name:** Akshayaa S

**Register No:** 231801007

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

**Semester:** IV

**Academic Year:** 2024 - 2025

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.No. | Date | Title |
| 1. | 22/1/25 | Azure Devops Environment Setup. |
| 2. | 22/1/25 | Azure Devops Project Setup and User Story Management. |
| 3. | 29/1/25 | Setting Up Epics, Features, And User Stories for Project Planning. |
| 4. | 12/2/25 | Sprint Planning. |
| 5. | 19/2/25 | Poker Estimation. |
| 6. | 26/2/25 | Designing Class and Sequence Diagrams for Project Architecture. |
| 7. | 05/3/25 | Designing Architectural and ER Diagrams for Project Structure. |
| 8. | 26/3/25 | Testing – Test Plans and Test Cases. |
| 9. | 16/4/25 | Load Testing and Pipelines. |
| 10. | 23/4/25 | GitHub: Project Structure & Naming Conventions. |

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

**AZURE DEVOPS ENVIRONMENT SETUP**

Create Epic, Features, User Stories, Task

**Aim:**

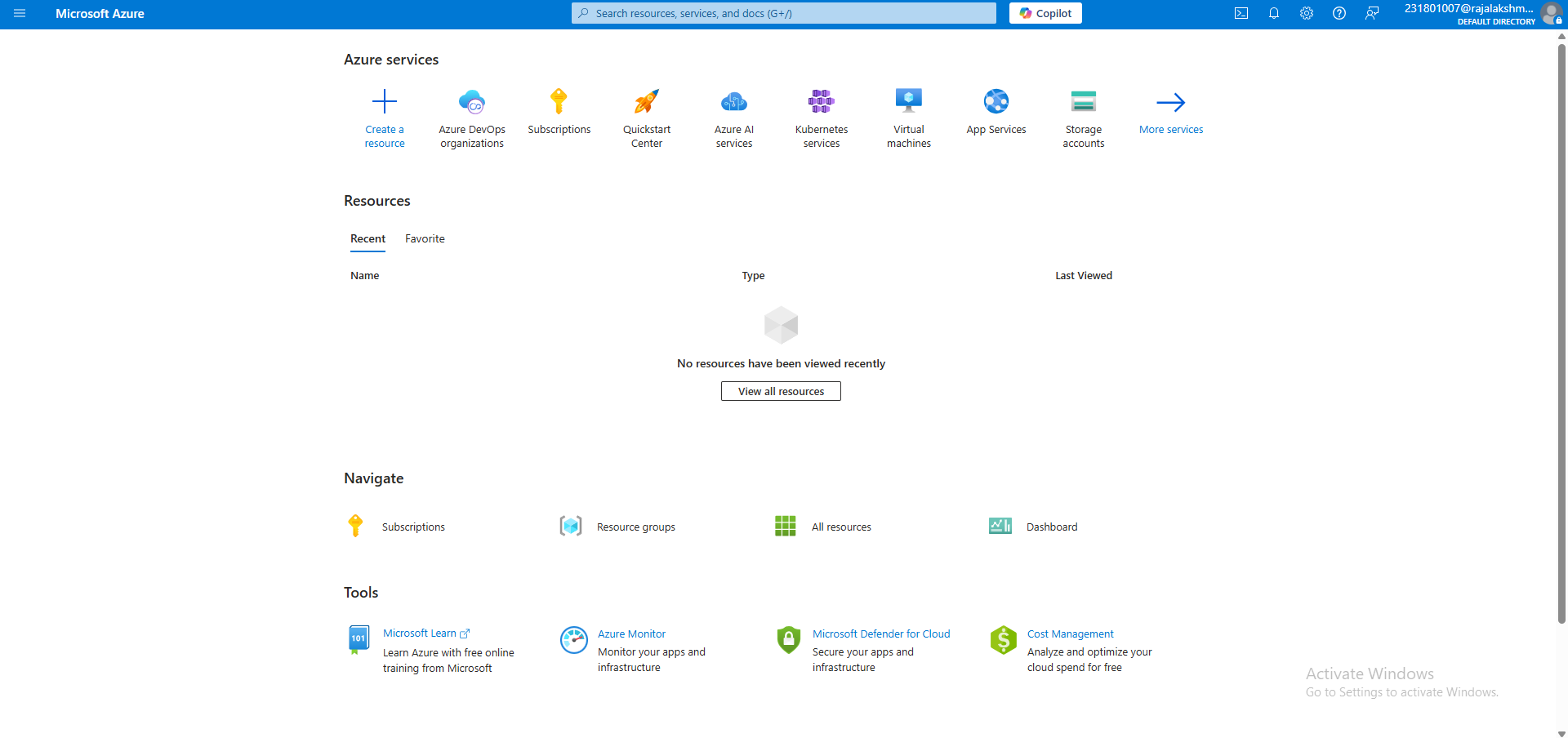
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

**INSTALLATION**

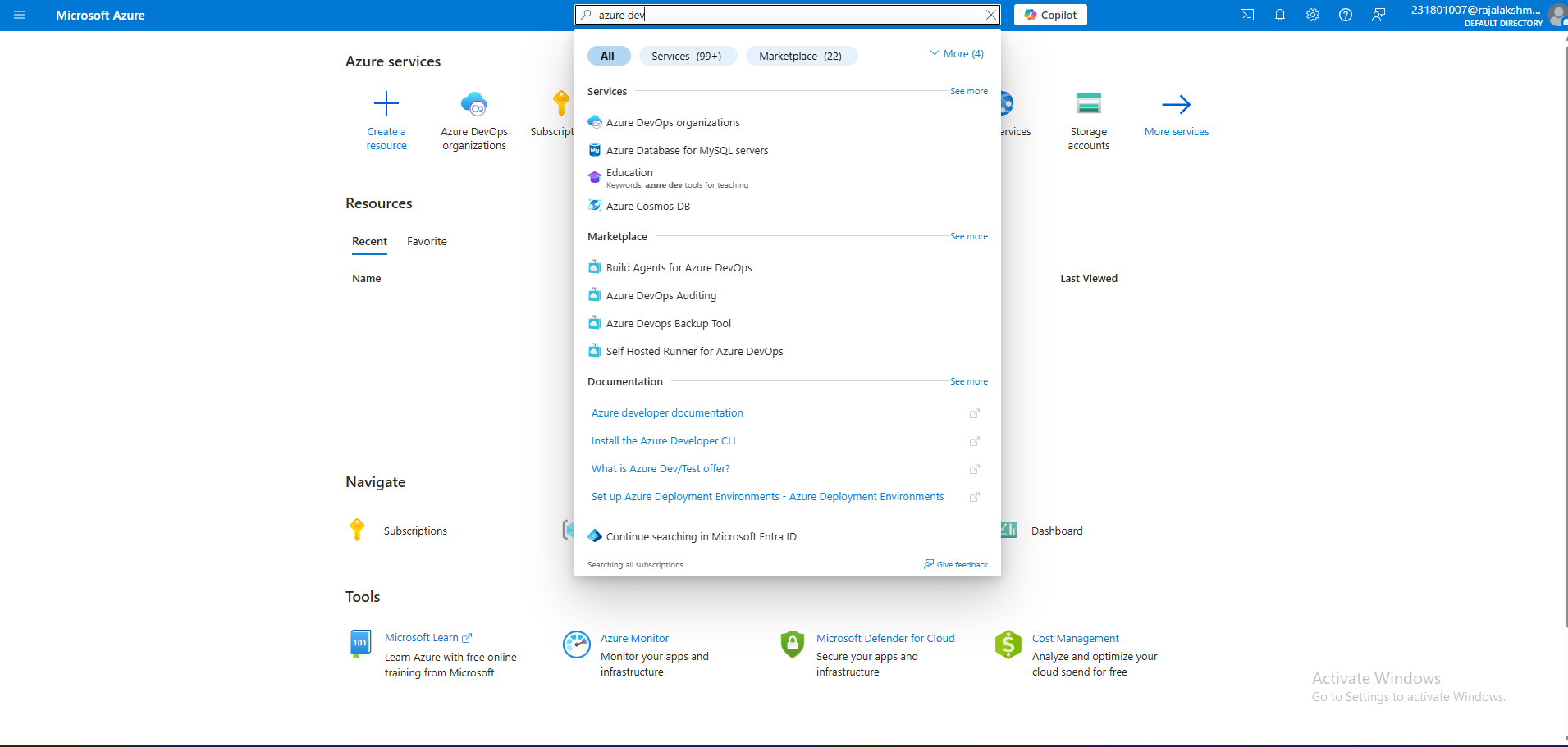
1.Open your web browser and go to the Azure website: <https://azure.microsoft.com/en-us/get-started/azure-portal>.  
Sign in using your Microsoft account credentials.  
If you don't have a Microsoft account, you can create one here: <https://signup.live.com/?lic=1>



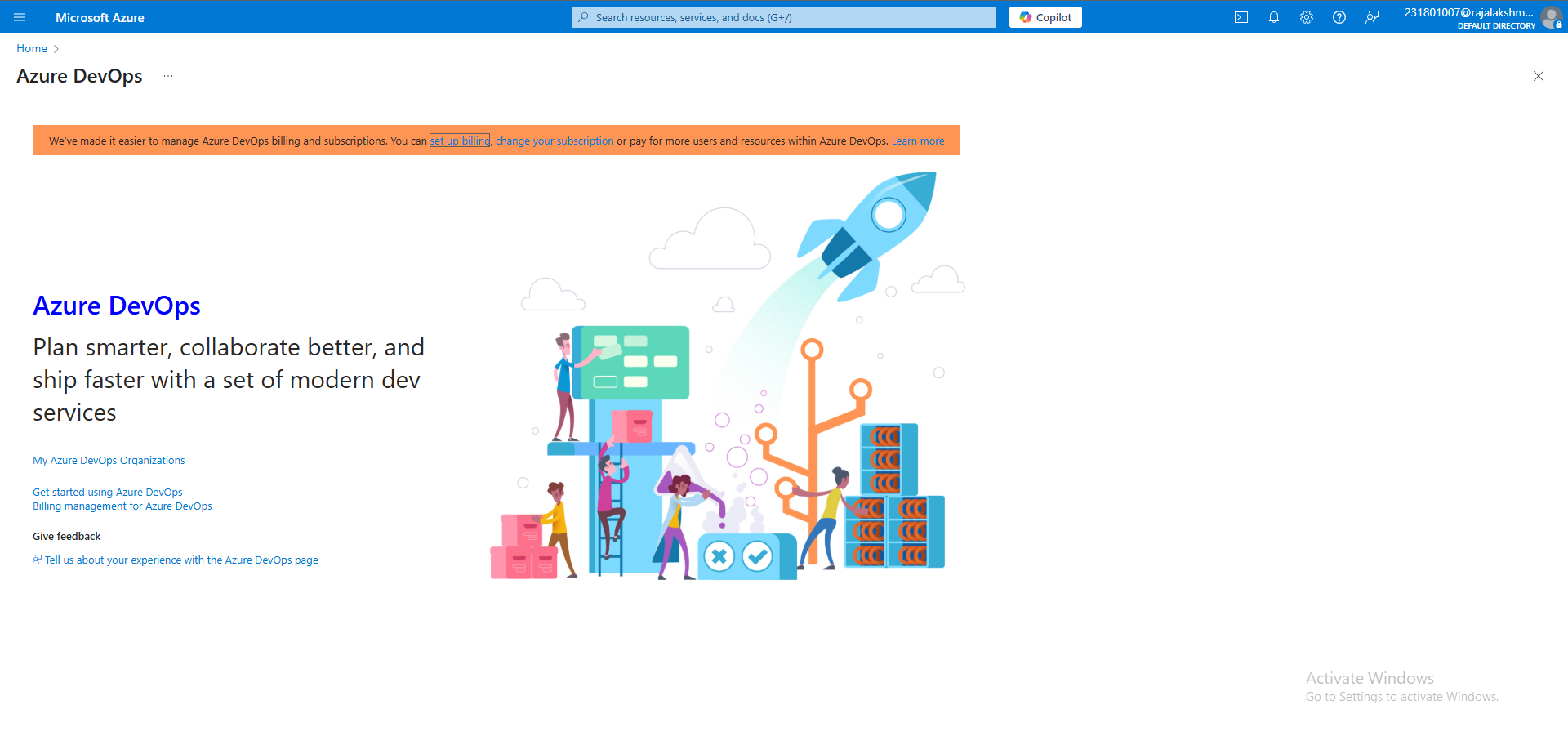
2.Azure home page



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.

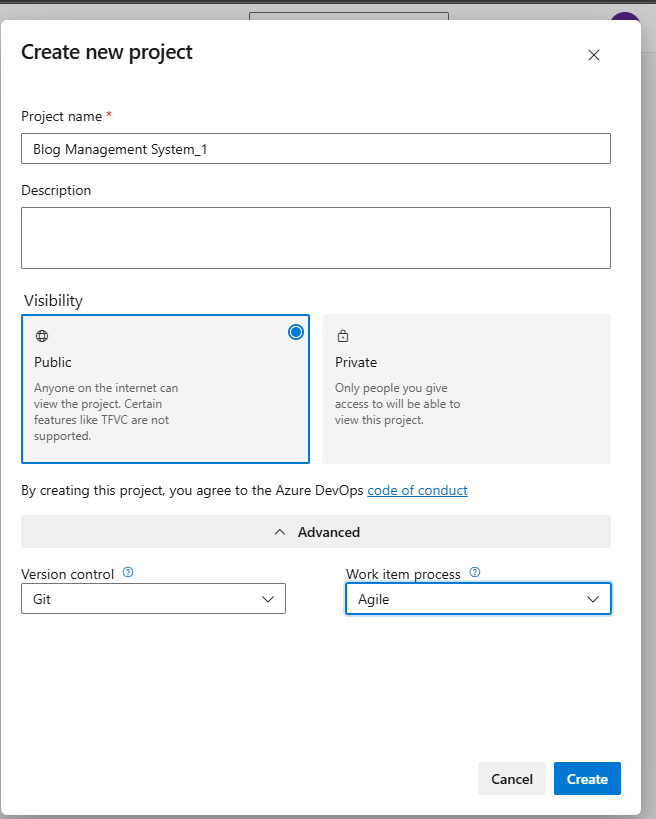
c. Enter the project name, description, and visibility options:

***Name****:* Choose a name for the project (e.g., LMS).

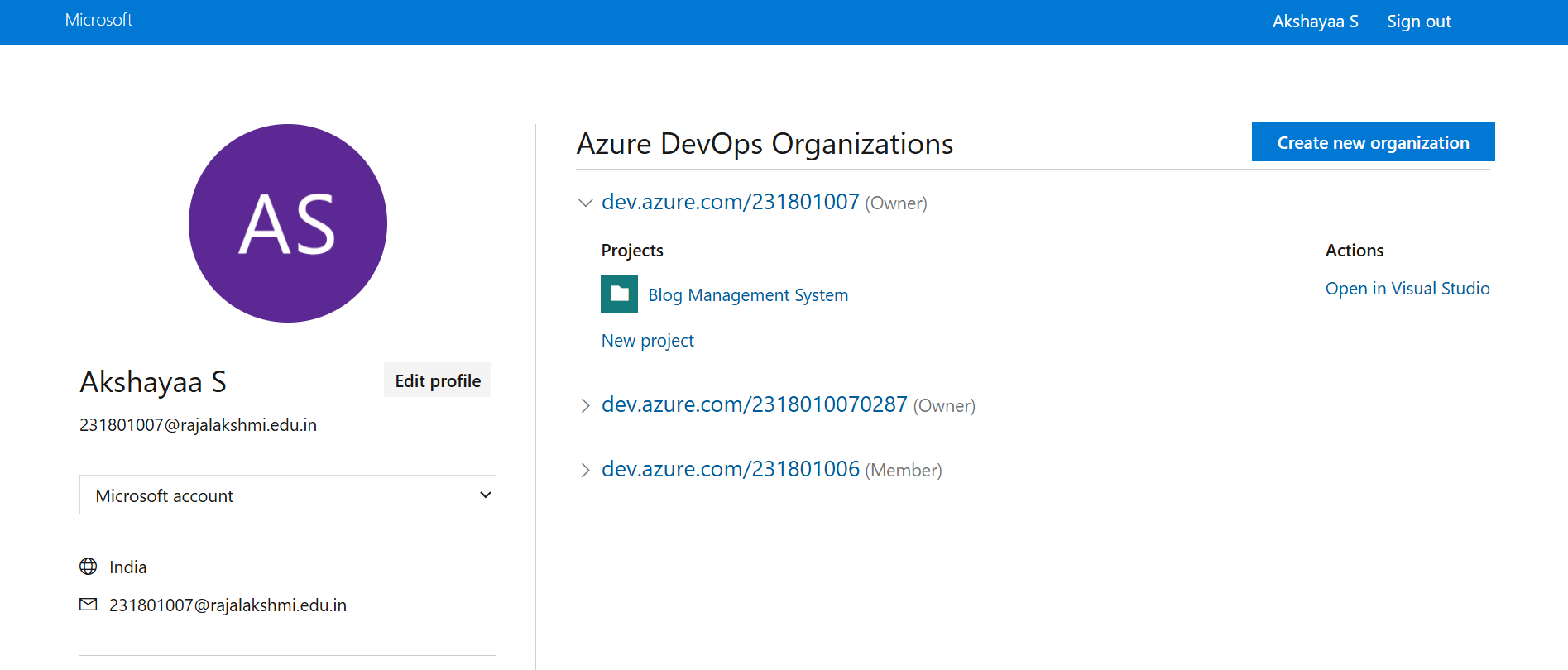
***Description****:* Optionally, add a description to provide more context about the project.

***Visibility****:* Choose whether you want the project to be **Private** (accessible only to those invited) or **Public** (accessible to anyone).

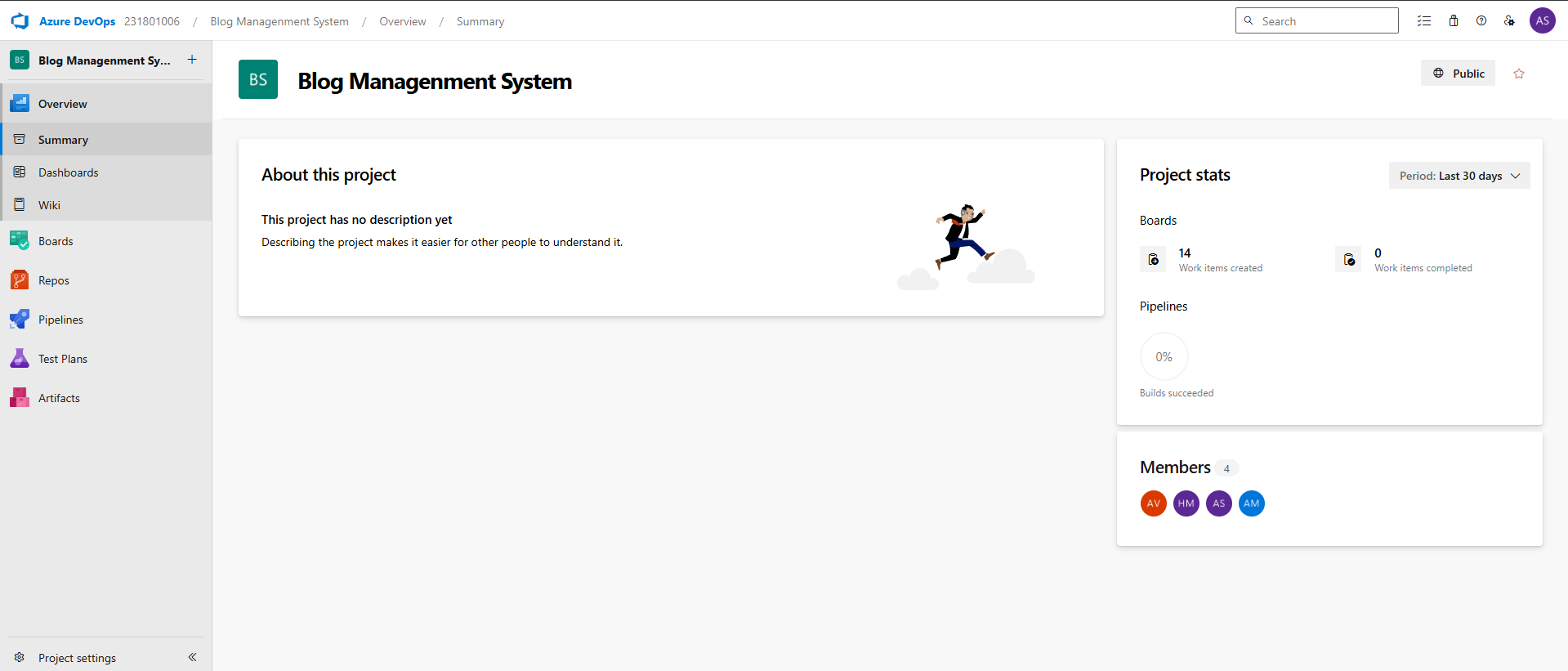
d. Once you’ve filled out the details, click **Create** to set up your first project.



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**

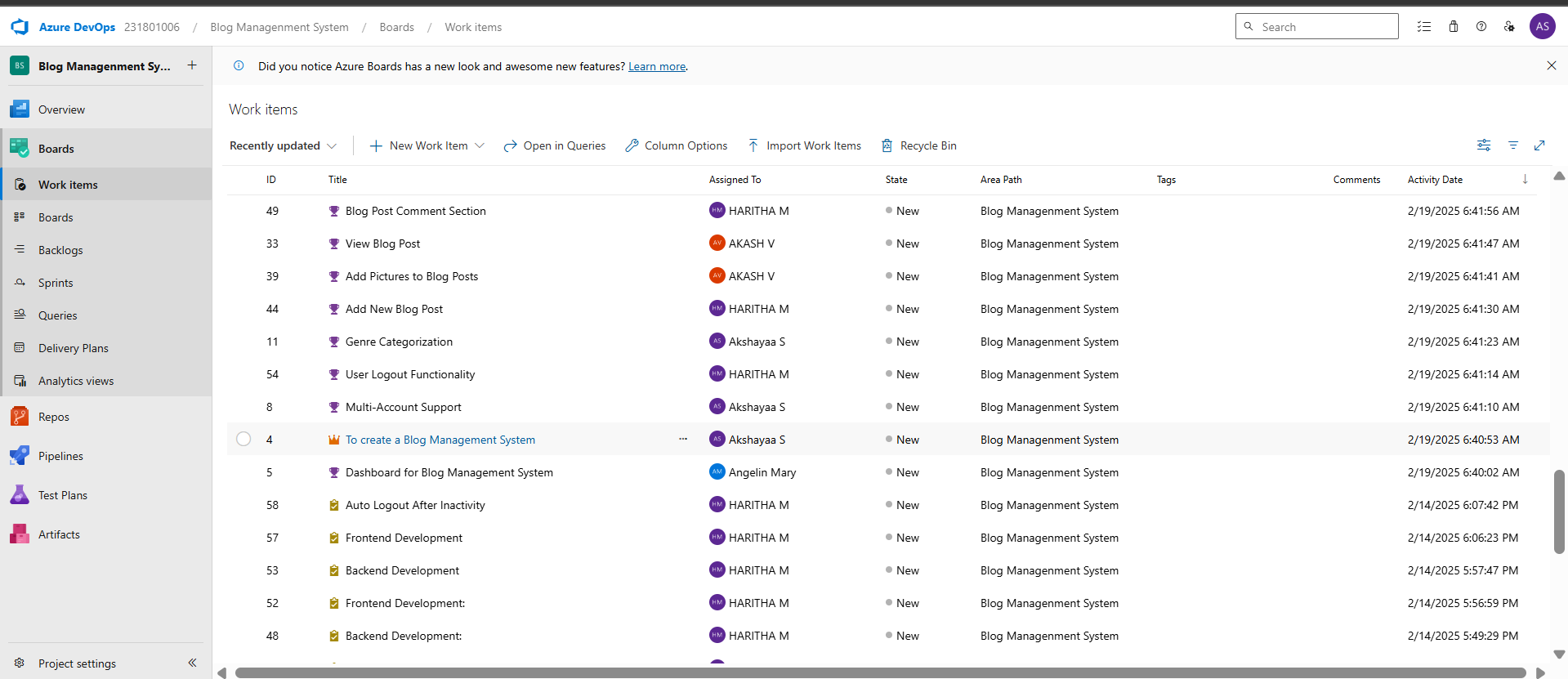
**SETTING UP EPICS, FEATURES, AND USER STORIES FOR PROJECT PLANNING**

Create Epic, Features, User Stories, Task

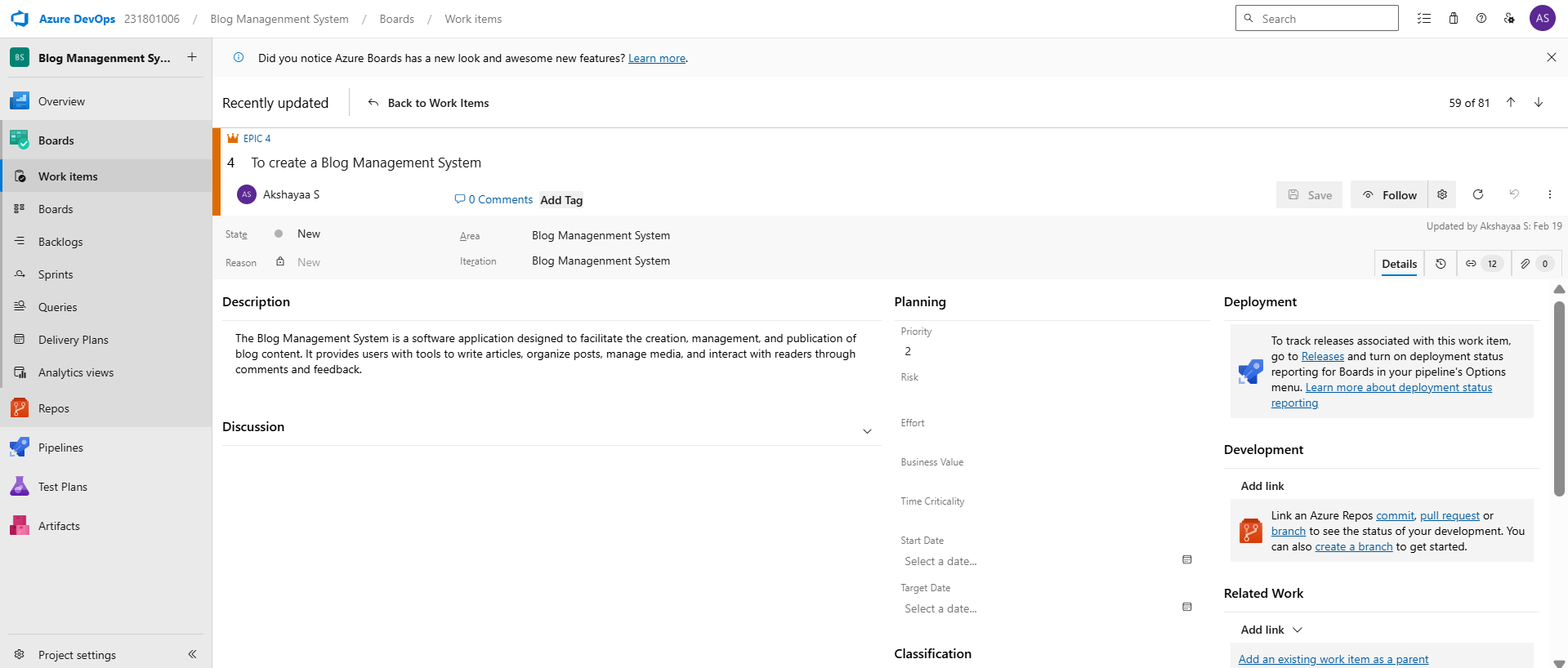
**Aim:**

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

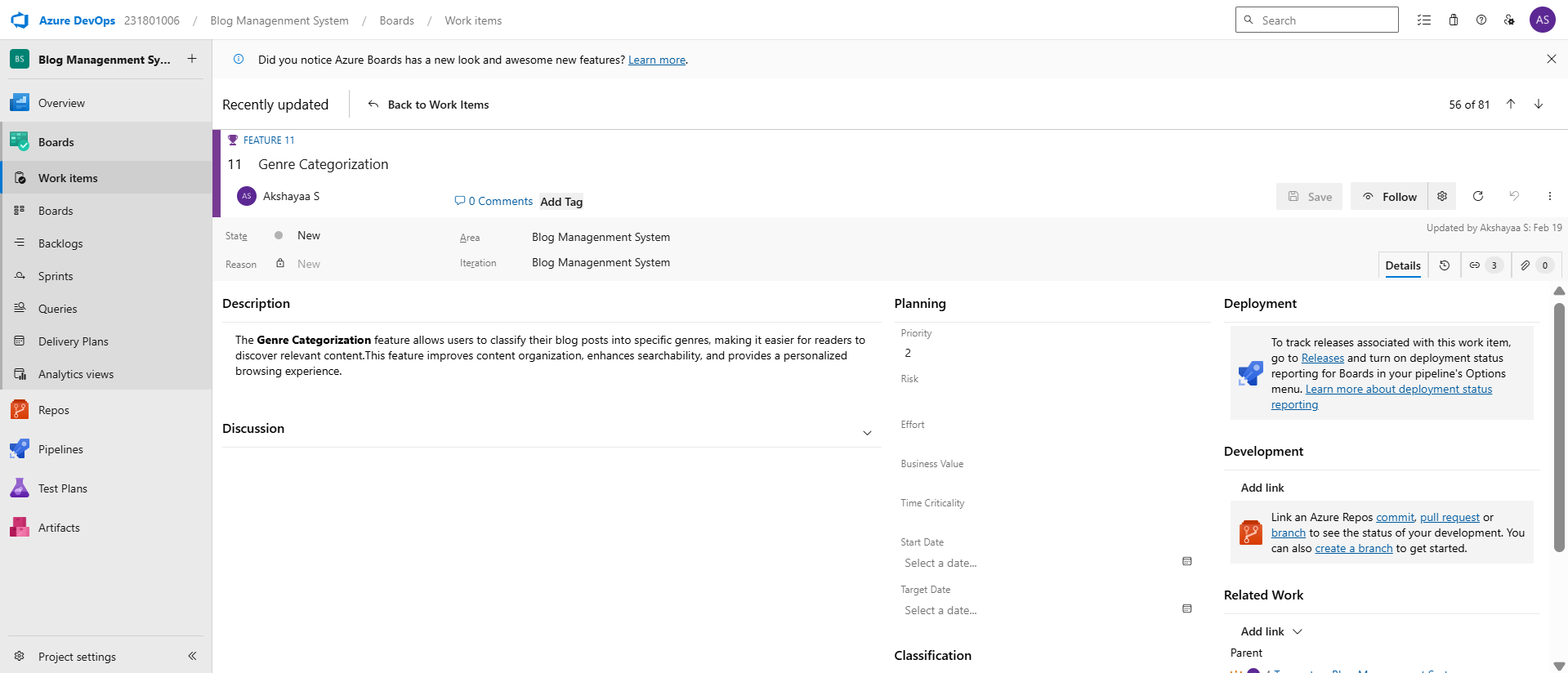
**Create Epic, Features, User Stories, Task**



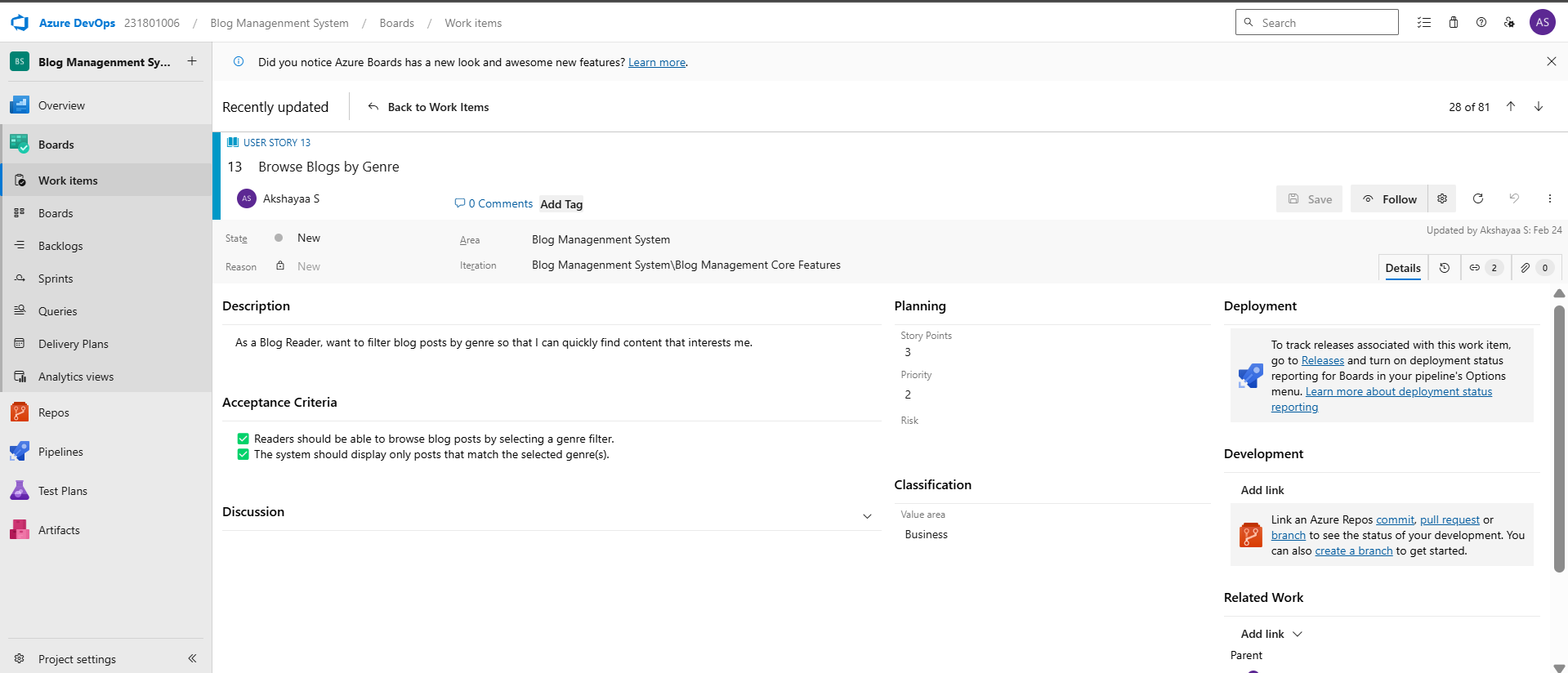
**1.Fill in Epics**



**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**

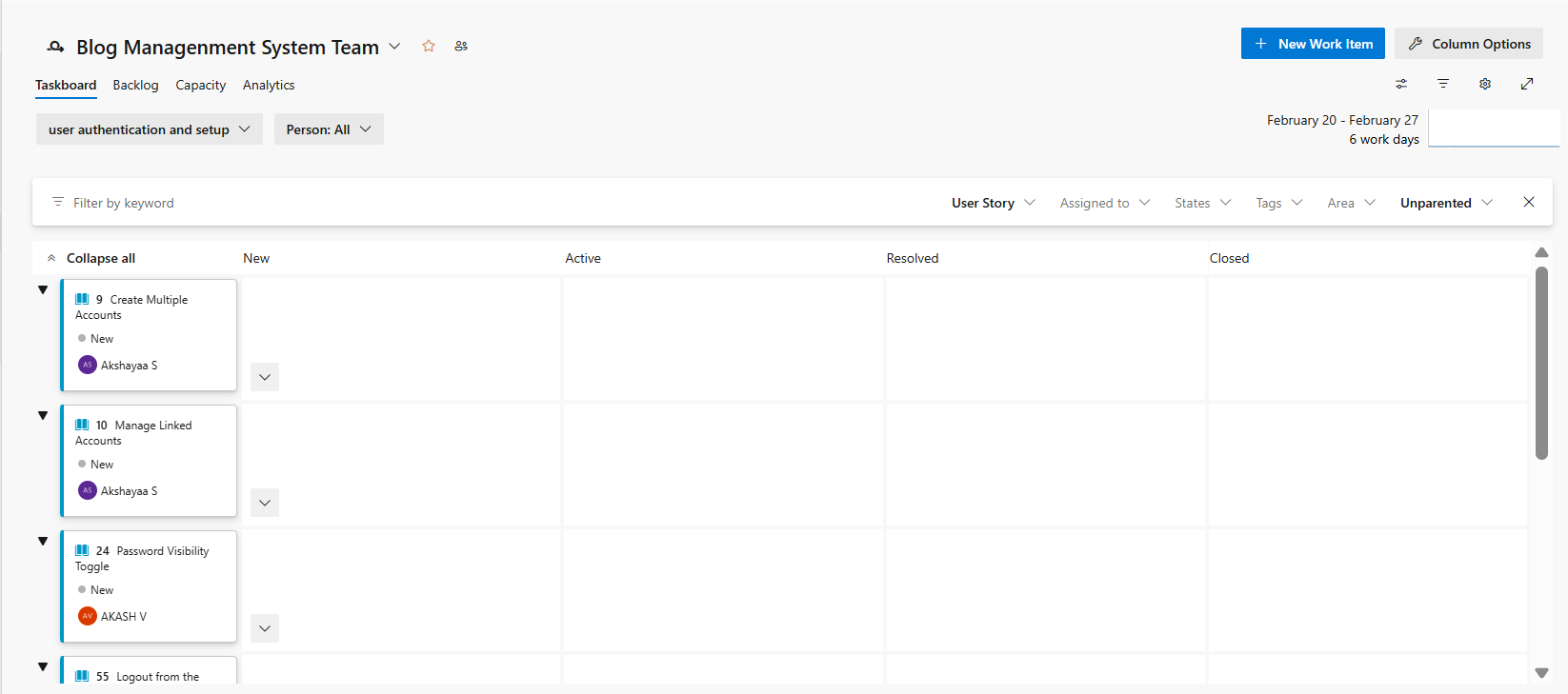
Create Epic, Features, User Stories, Task

**Aim:**

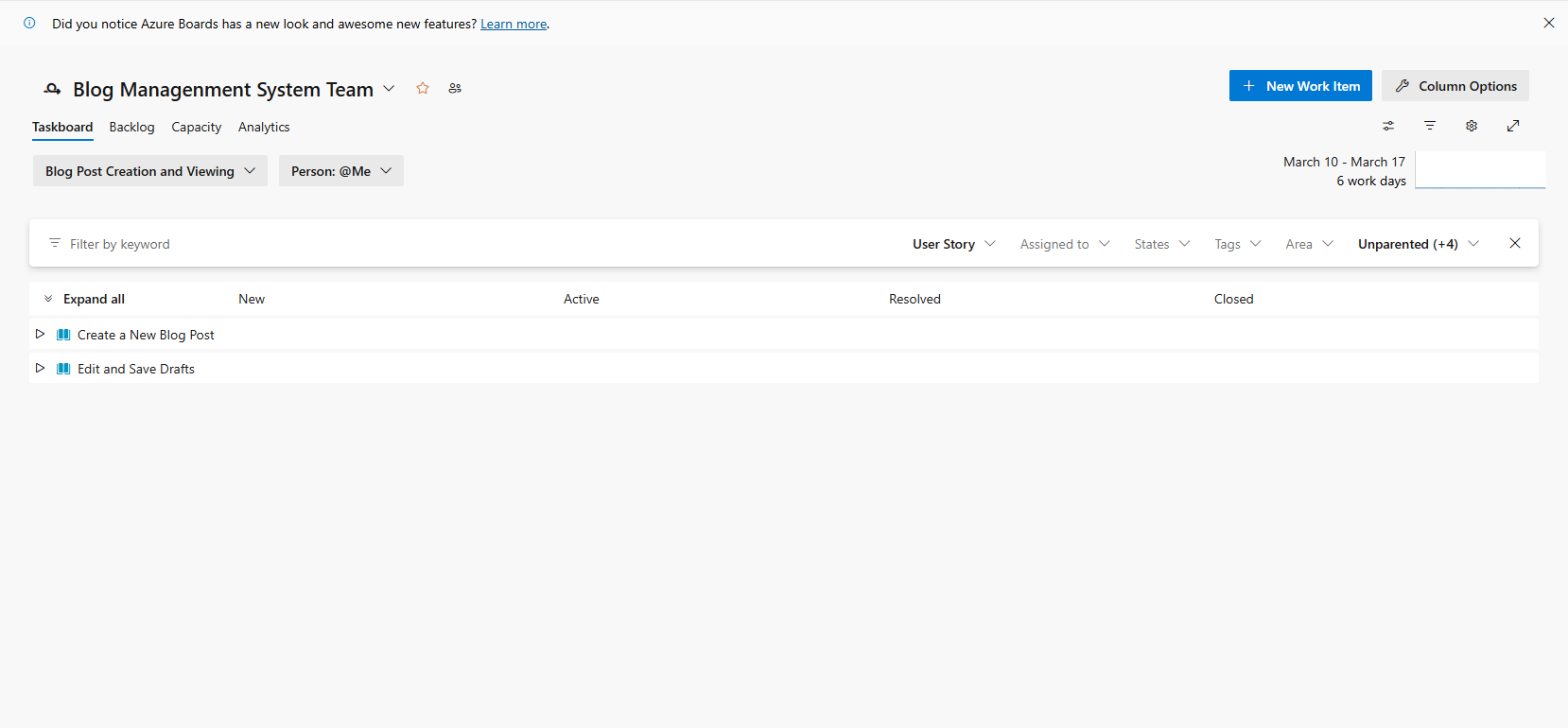
To assign user story to specific sprint for the Music Playlist Batch Creator Project.

**Sprint Planning**

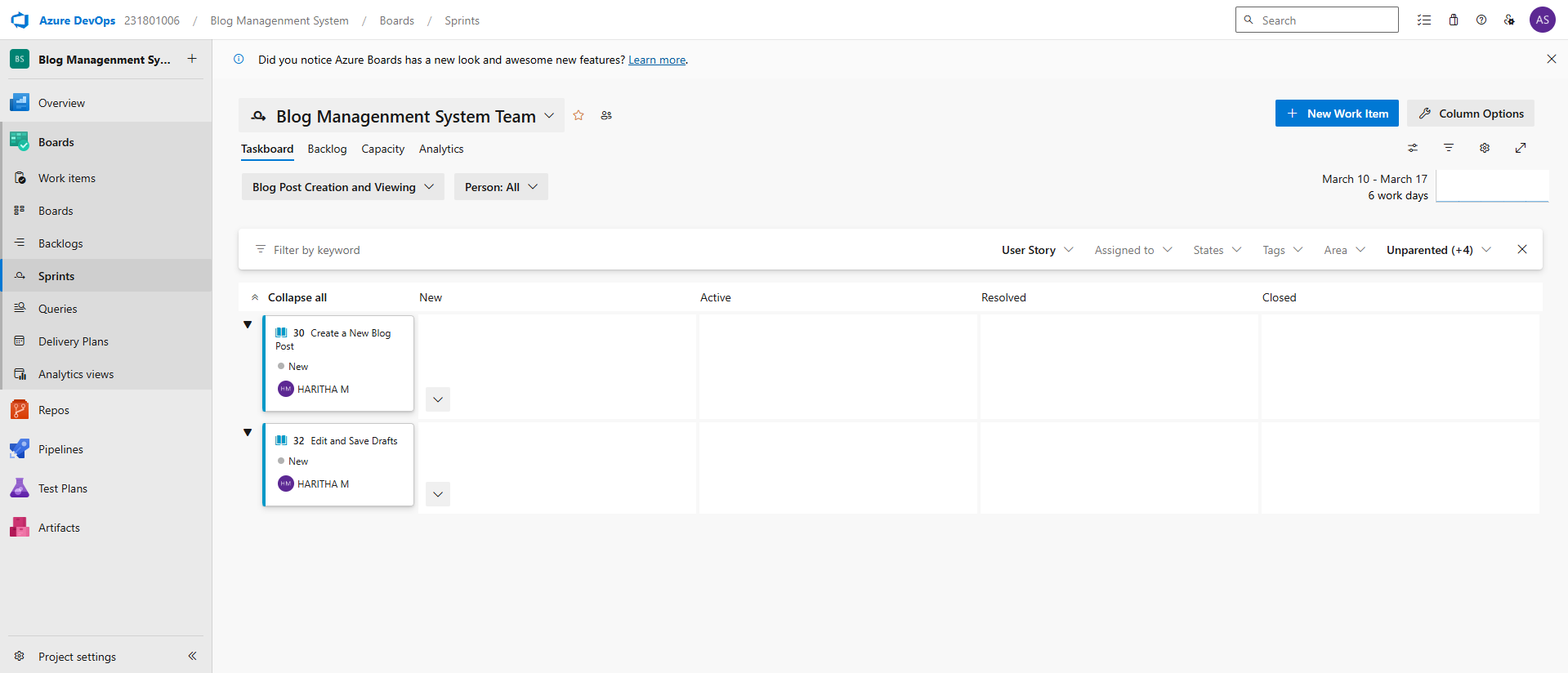
**Sprint 1**



**Sprint 2**



**Sprint 3**



**Result:**

The Sprints are created for the Blog management System

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

**POKER ESTIMATION**

Create Epic, Features, User Stories, Task

**Aim:**

Create Poker Estimation for the user stories – Blog management System

**Poker Estimation**



**Result:**

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

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

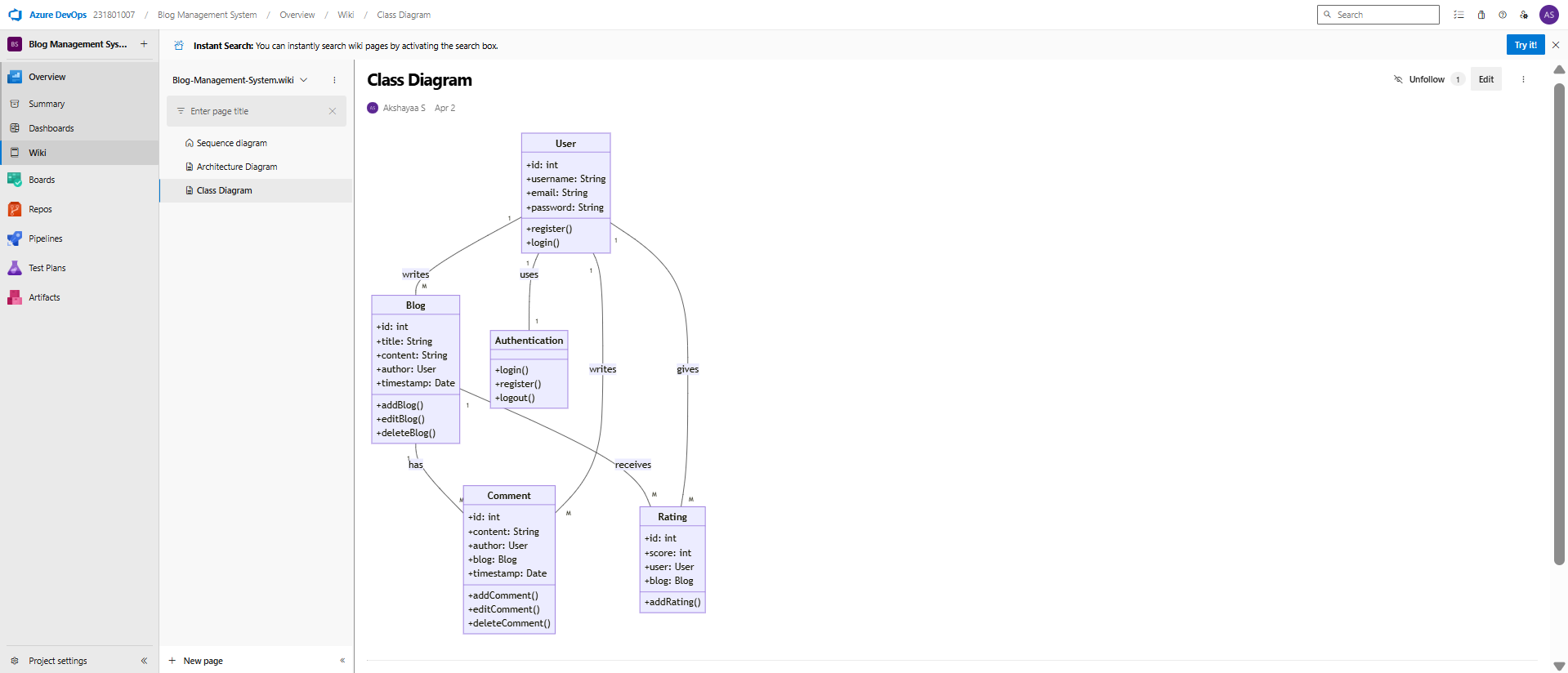
**DESIGNING CLASS AND SEQUENCE DIAGRAMS FOR PROJECT ARCHITECTURE**

Create Epic, Features, User Stories, Task

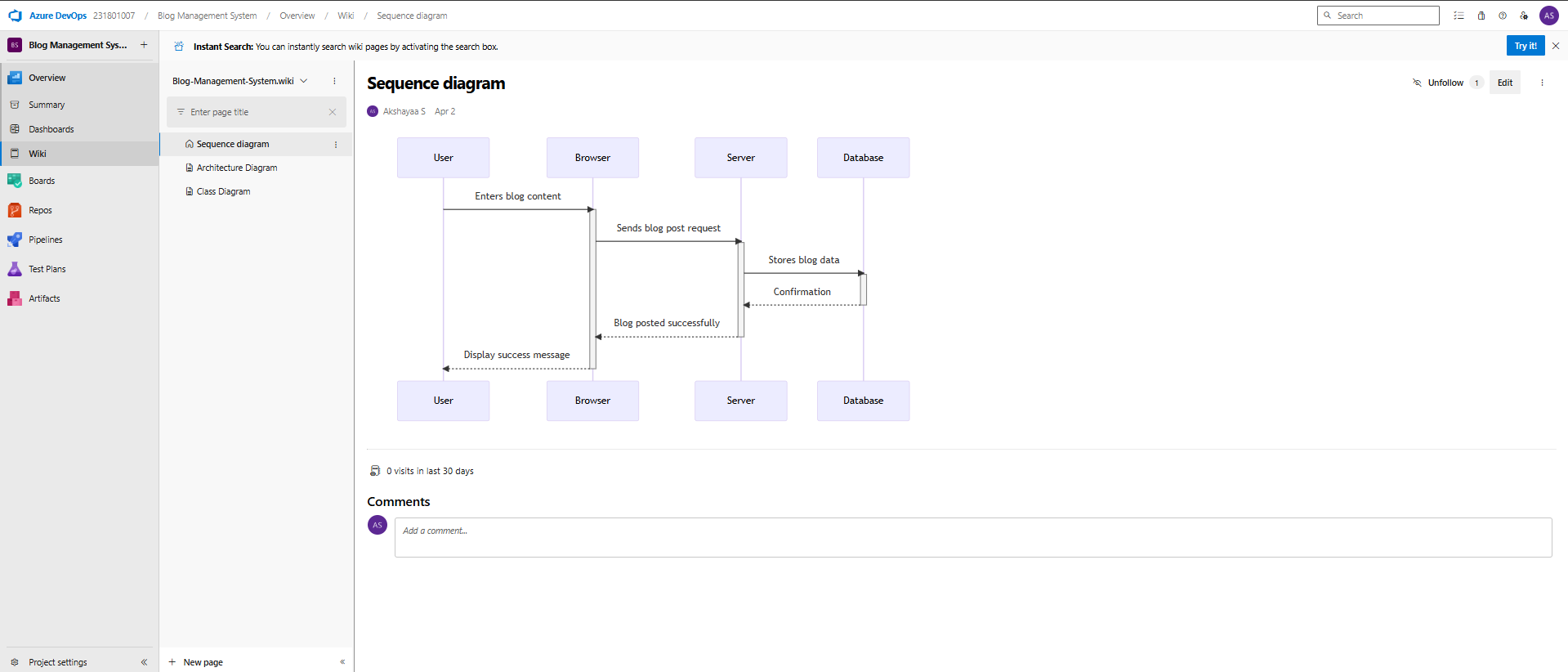
**Aim:**

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 Blog management System.

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

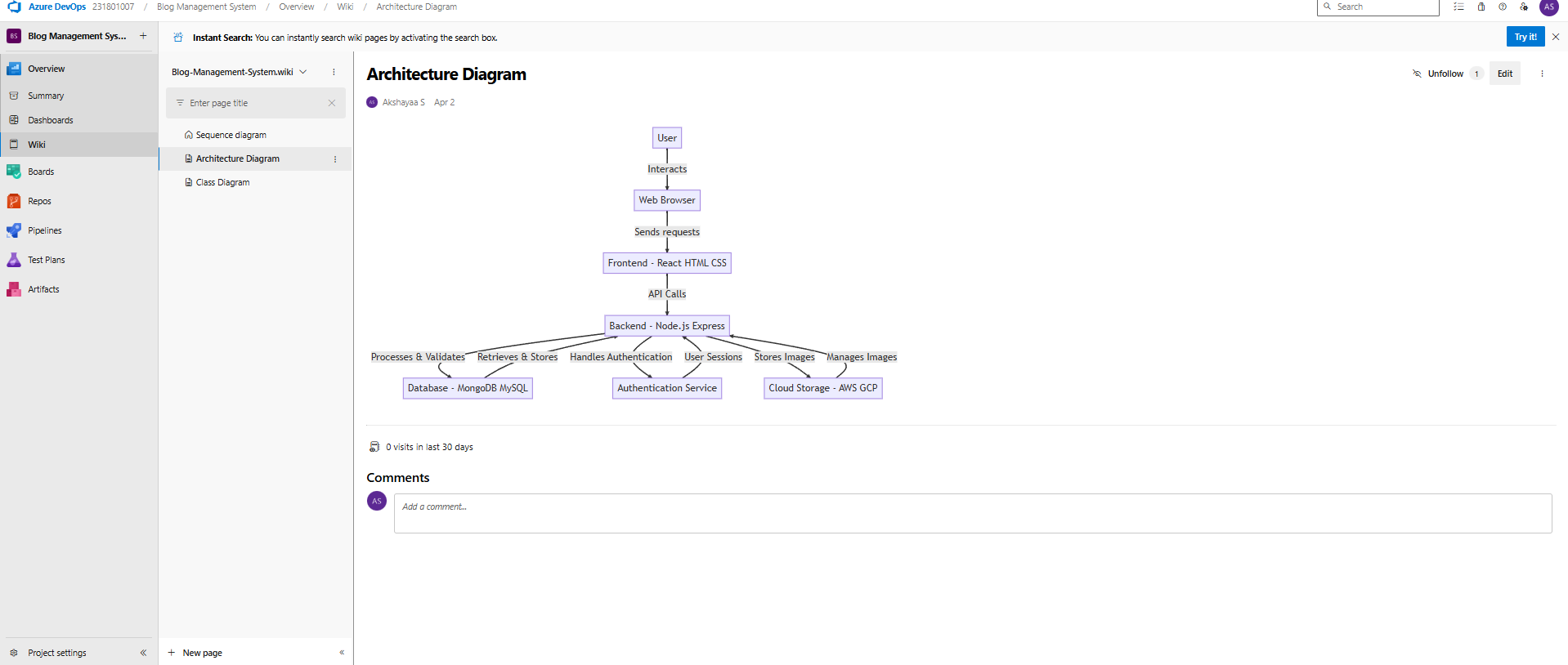
**DESIGNING ARCHITECTURAL AND ER DIAGRAMS FOR PROJECT STRUCTURE**

Create Epic, Features, User Stories, Task

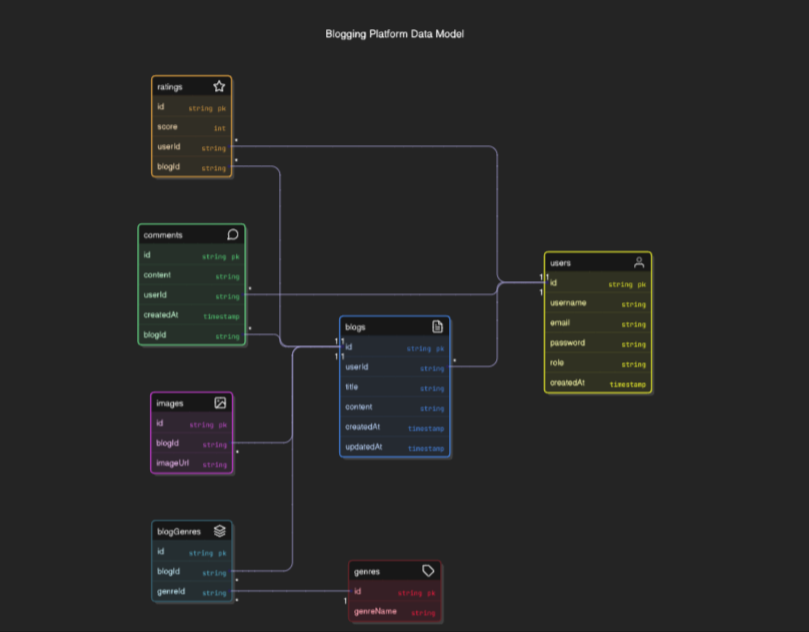
**Aim:**

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 Blog management system

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

**TESTING – TEST PLANS AND TEST CASES**

Create Epic, Features, User Stories, Task

**Aim:**

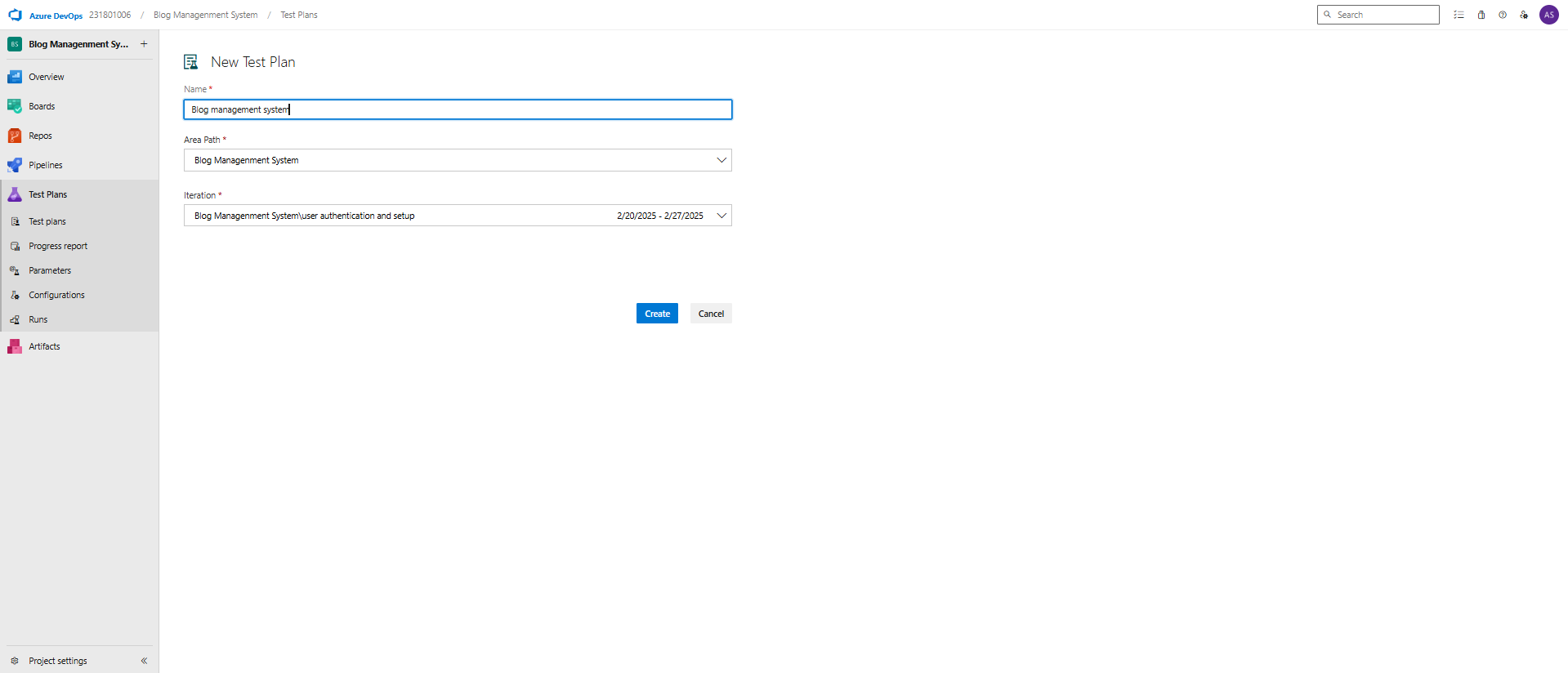
Test Plans and Test Case and write two test cases for at least five user stories showcasing the happy path and error scenarios in azure DevOps platform.

**Test Planning and Test Case**

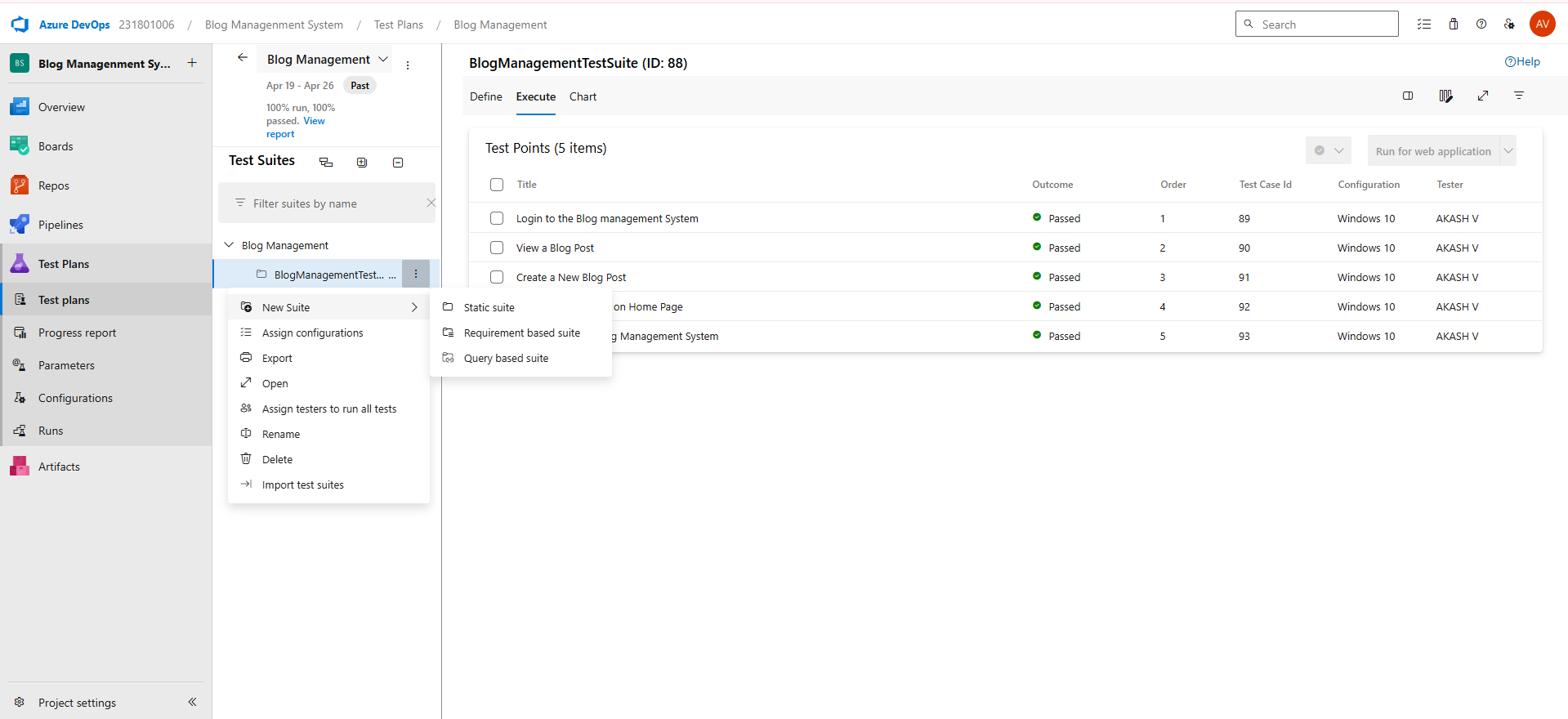
**Test Case Design Procedure**

1. **Understand Core Features of the Application**
   * User Signup & Login
   * Viewing and Managing Blogs
   * Fetching Real-time Metadata
   * Editing the blogs
   * Creating smart blog system based on categories (mood, genre, author, etc.)
2. **Define User Interactions**
   * Each test case simulates a real user behaviour
3. **Design Happy Path Test Cases**
   * Focused on validating that all features function as expected under normal conditions.
4. **Design Error Path Test Cases**
   * Simulate negative or unexpected scenarios to test robustness and error handling.
5. **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.
6. **Use Clear Naming and IDs**
   * Test cases are named clearly.
   * Helps in quick identification and linking to user stories or features.
7. **Separate Test Suites**
   * Grouped test cases based on functionality
   * Improves organization and test execution flow in Azure DevOps.
8. **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.

Music Playlist Batch Creator – Test Plans

**USER STORIES**

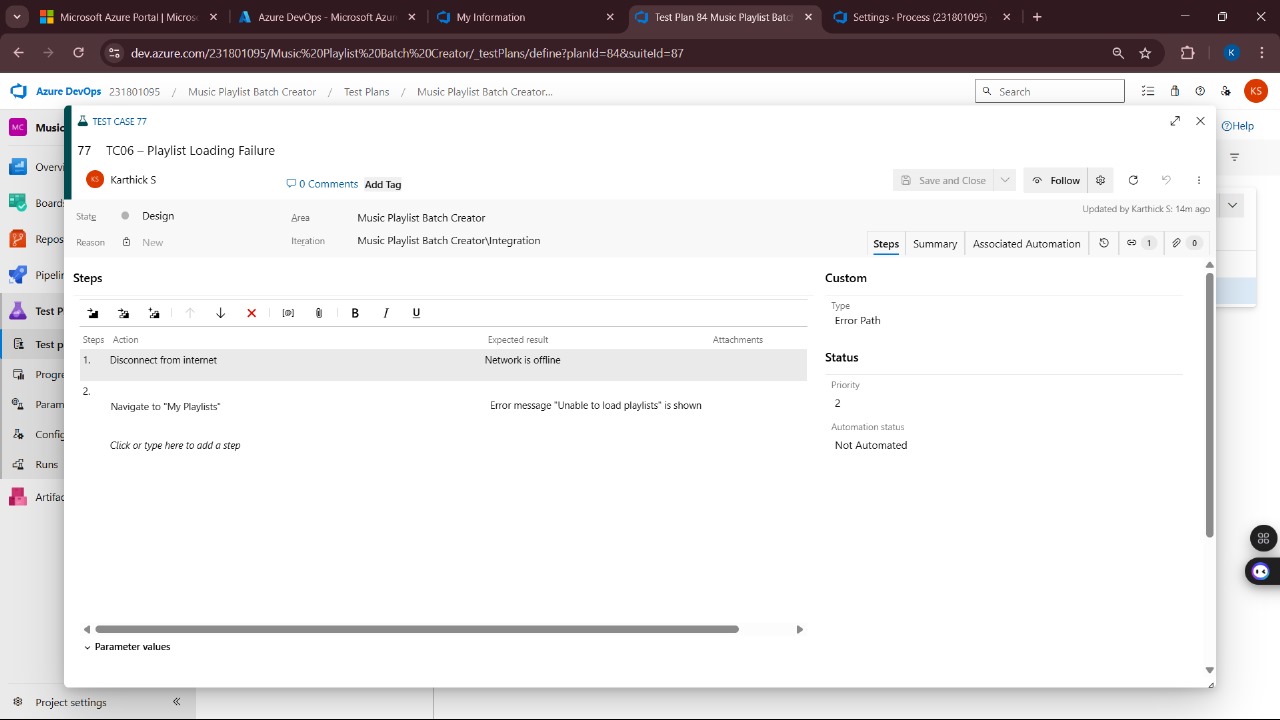
* As a user, I want to sign up and log in securely so that I can access my blogs (ID: 79).
* As a user, I need to see my blog post (ID: 76).
* As a user, I should be able to create my own new blog post (ID: 73).
* As a user, I should be able to see the recent activity on the home page (ID: 68).
* As a user, I need to have a safe logout system to keep my account secure (ID: 65).

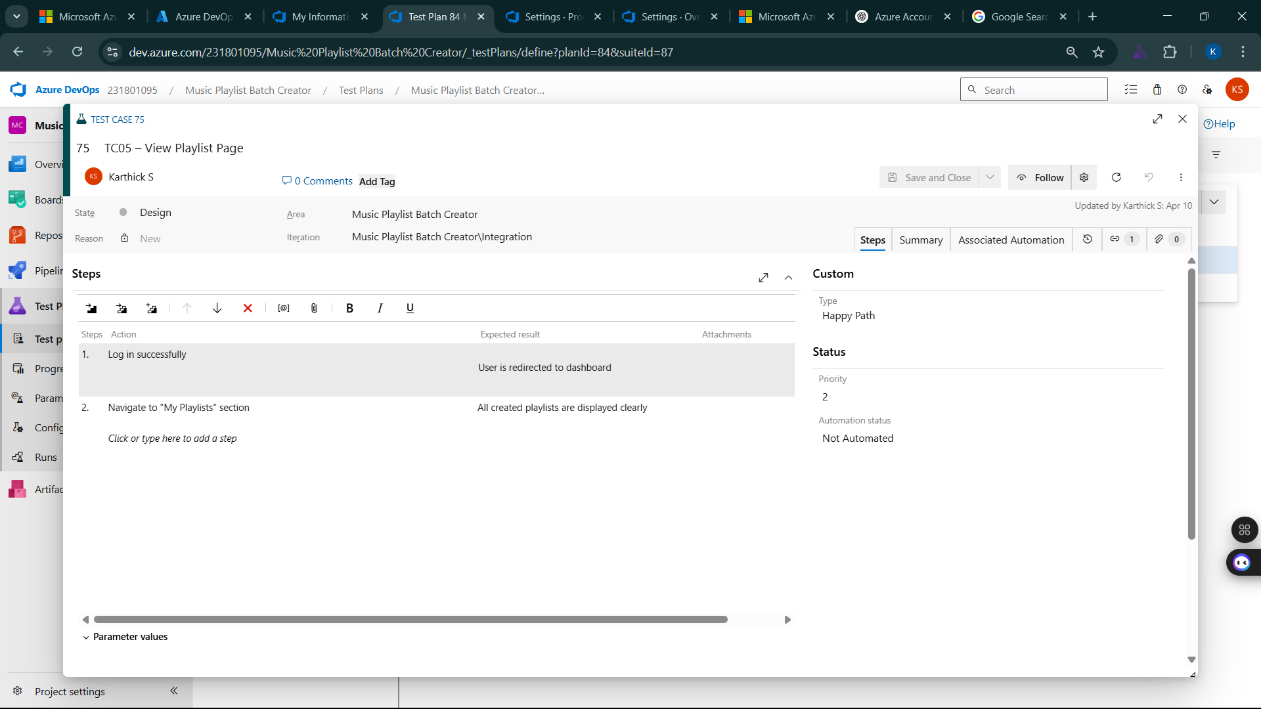
**Test Suites**

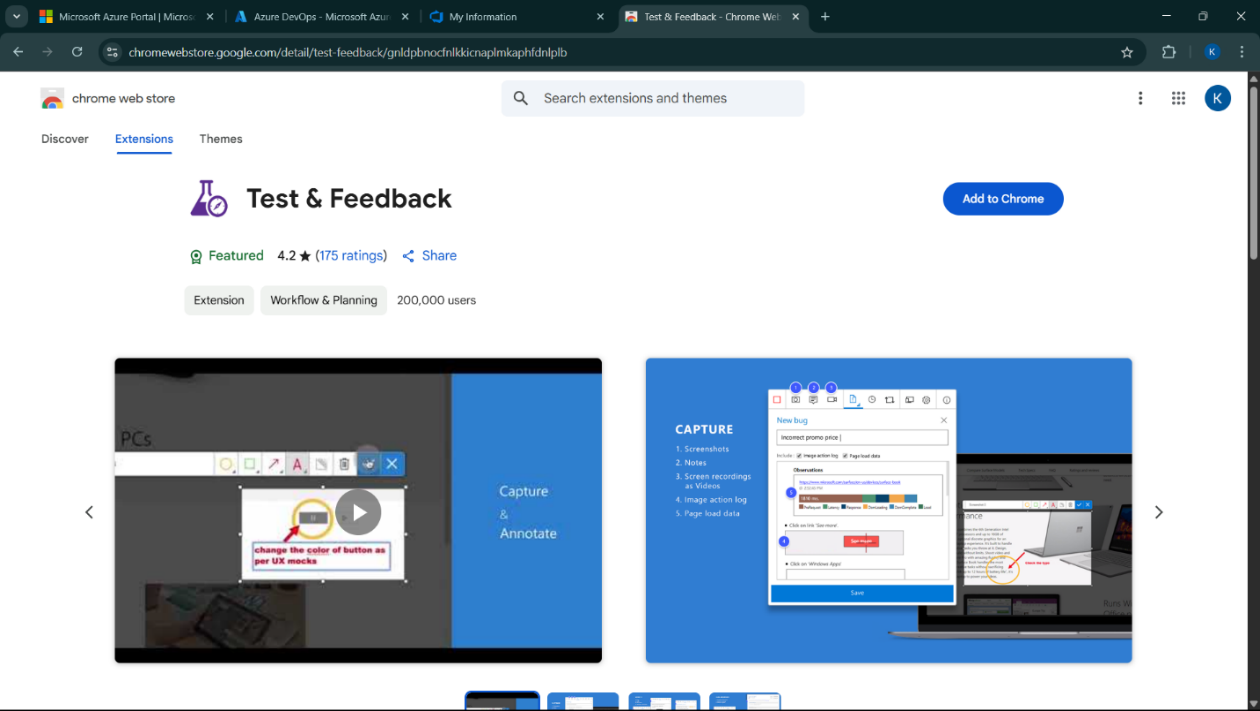
**Test Suit: TS01 - User Login (ID: 86)**

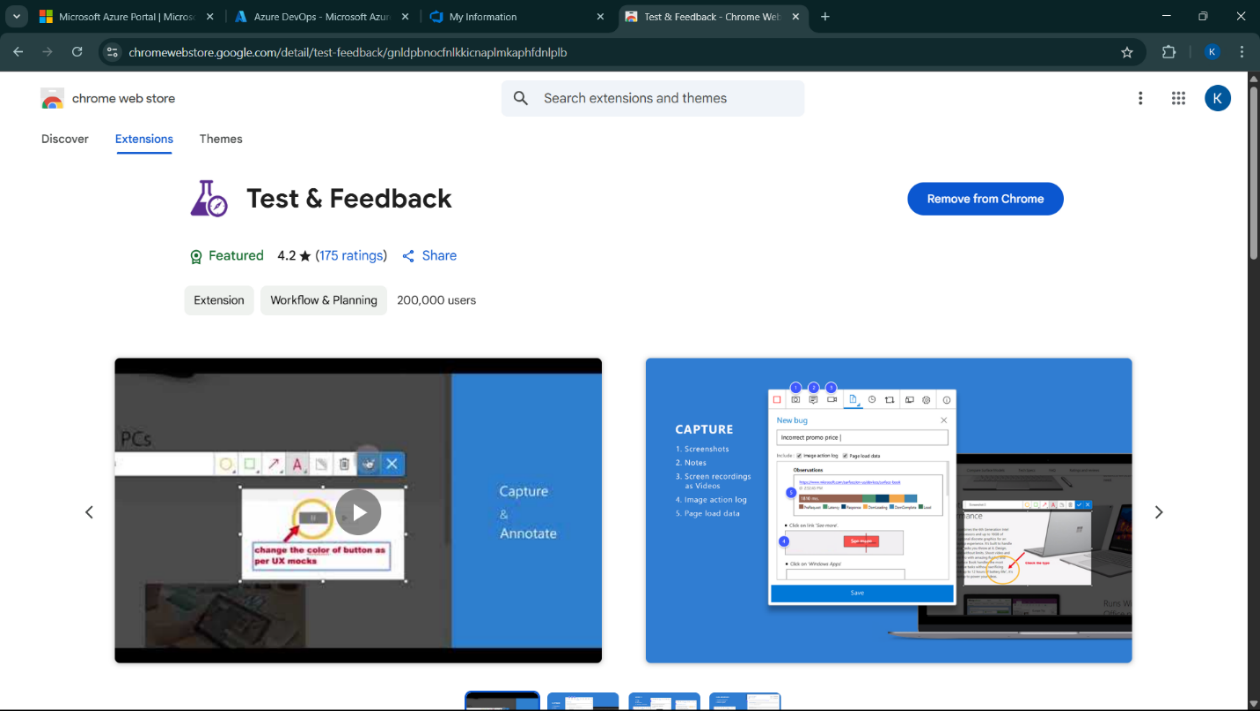
1. **TC01 – Successful login page**
   * **Action:**
     + Navigate to the Blog Management System URL
     + Enter valid name, email, and password.
     + Click "Login".
     + Click on the forgot password .if you have forgotten the password
     + Set the correct new password , and then login again
   * **Expected Results:**
     + Login form is displayed.
     + Fields accept values without error and redirected to the home page
     + Error message prompt user to fill in the correct credentials .
     + Account is created, and the user is redirected to the dashboard.
   * **Type**: Happy Path
2. **TC02 – View a blog post** 
   * **Action:**
     + Go to the blog application URL.
     + Click on the title of the existing blog post
     + Verify the post content is visible
     + Check for related posts and comments
   * **Expected Results:**
     + Home page id displayed after the successful login
     + The selected blog post pages opens
     + Blog post details are correctly displayed
     + Related posts and comments are displayed
   * **Type:** Happy Path

1. **TC03 – Create a New blog posts**
   * **Action:**
     + Navigate to the dashboard or blog post list page
     + Click on the “Create New post” or “+ new Post “ button
     + Enter the title for the post
     + Enter the content/body for the post
     + Click on “Publish” and save
   * **Expected Results:**
     + Blog post list is displayed
     + New blog post form is displayed
     + Title is accepted and shown in the input field
     + Content is accepted
     + Blog post is saved and is redirected to the post view or blog list
   * **Type:** Happy path
2. **TC04 – View Recent Activity on Home Page**
   * **Action:** 
     + Locate the “Recent Activity” section on the homepage
     + Verify that the list displays the recent activities such as posts
     + Click on an item in the recent activity list
   * **Expected Results:**
     + Home page is displayed
     + Recent activity section is visible
     + The recent actions are listed with timestamps and user details
     + Then it redirects to the respective posts
   * **Type:** Error Path
3. **TC04 – Logout from the blog management System** 
   * **Action:** 
     + Locate the profile menu section on the homepage
     + Click on the logout button
     + And then it navigates to thee homepage
   * **Expected Results:**
     + Logout option is visible
     + User is logged out and redirected to the login page
     + The user is redirected to the login page
   * **Type:** Happy path



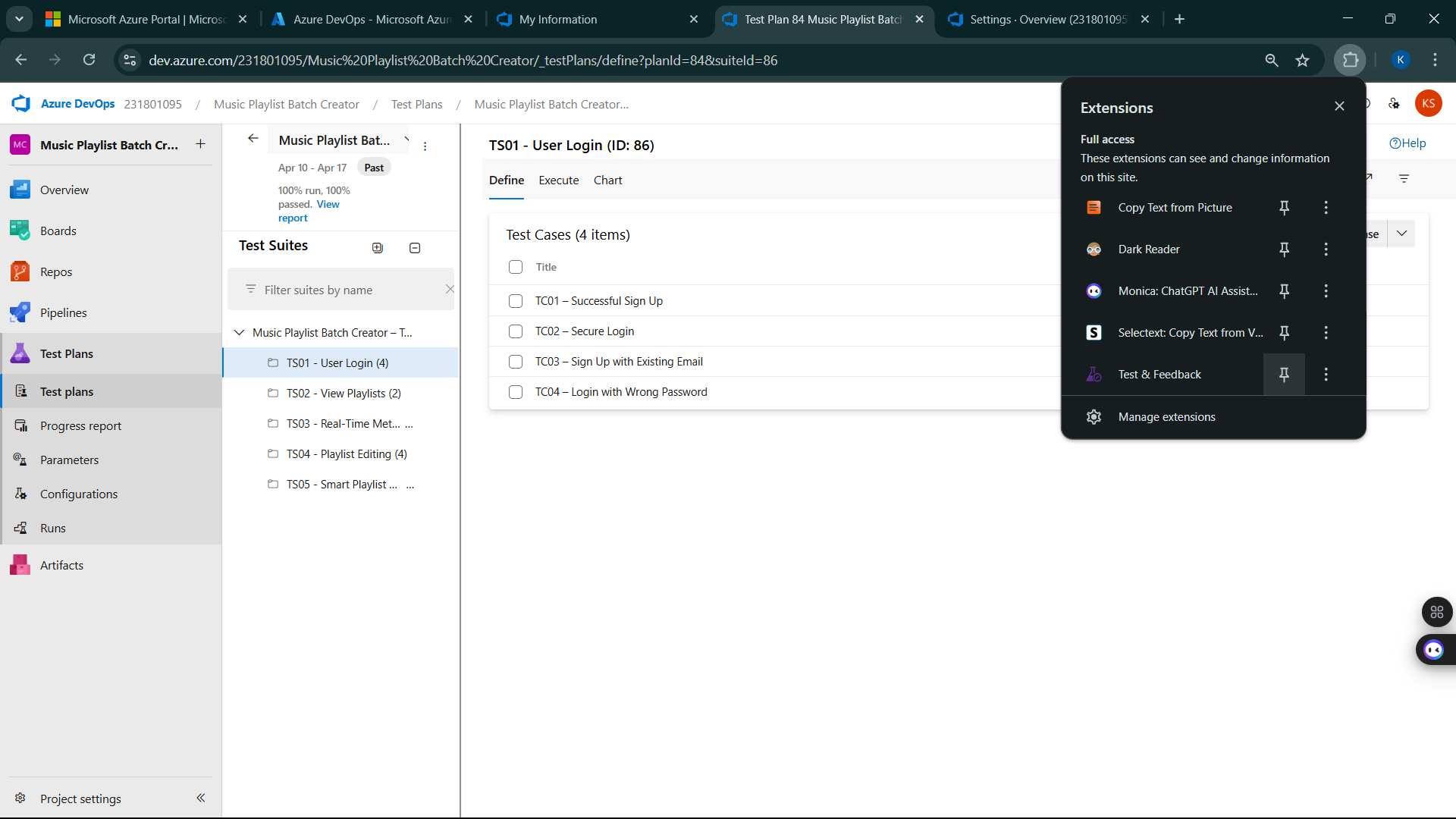
  
**4.Installation of test**



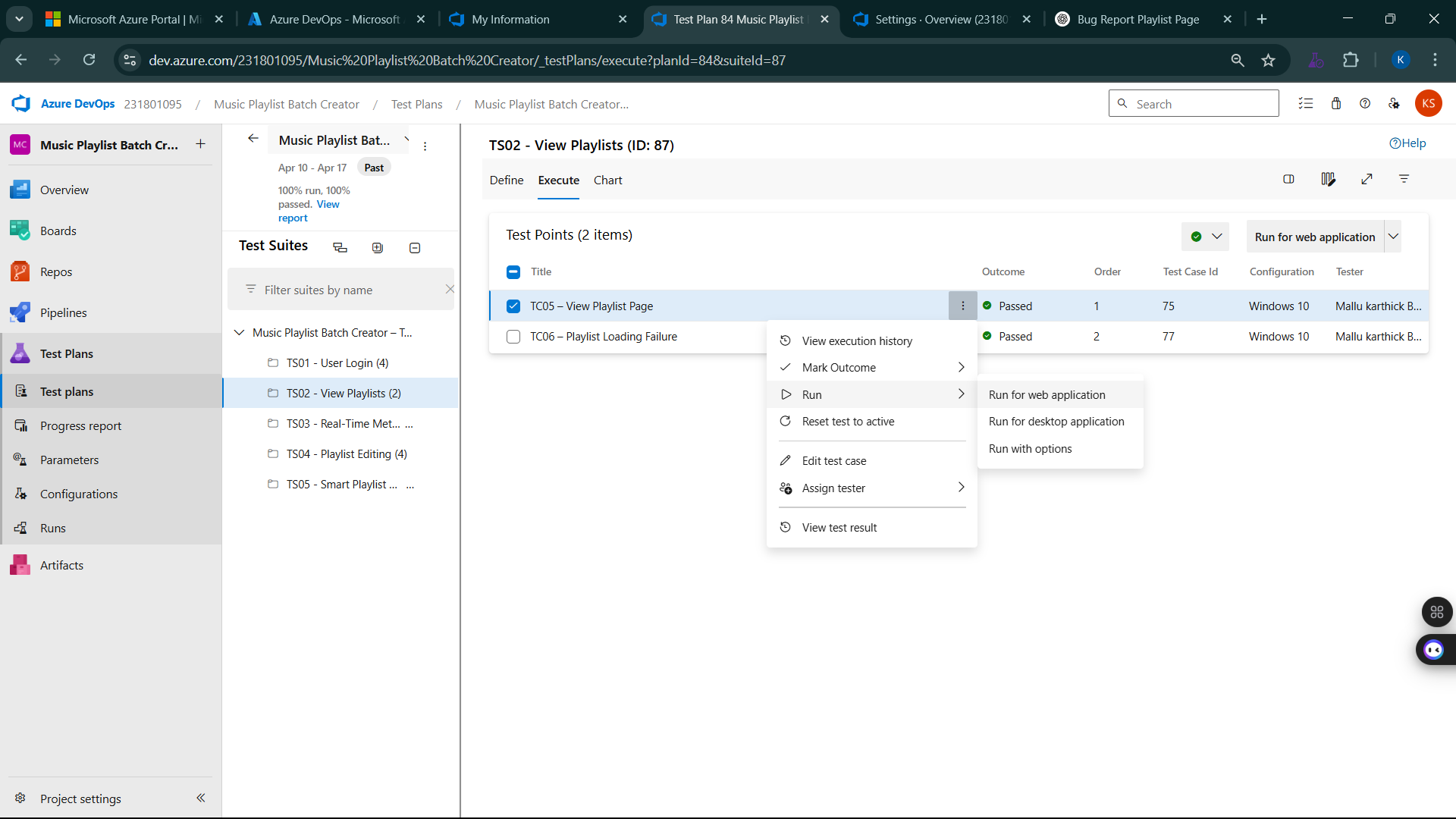


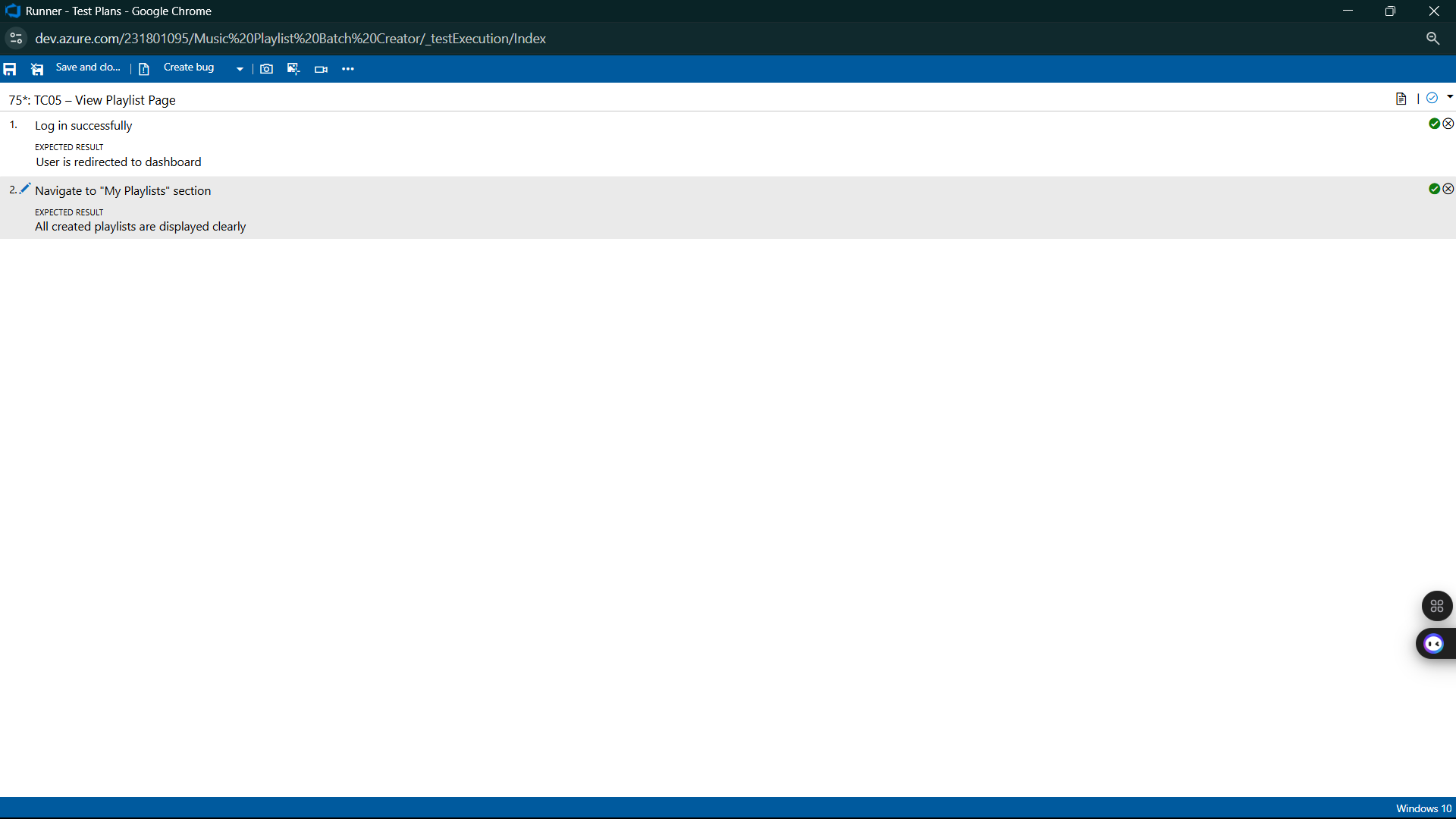
Test and feedback

Showing it as an extension

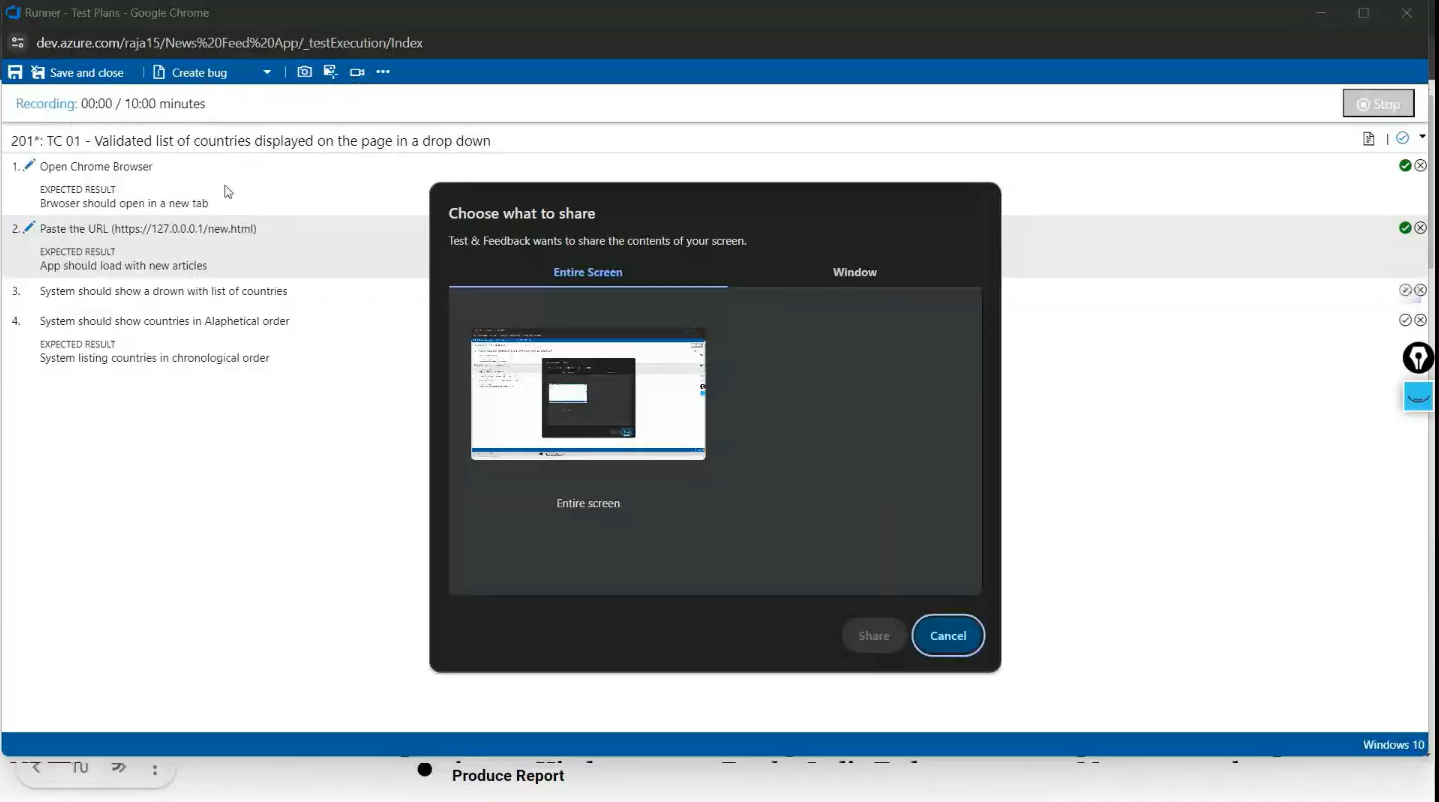


**5.Running the test cases**

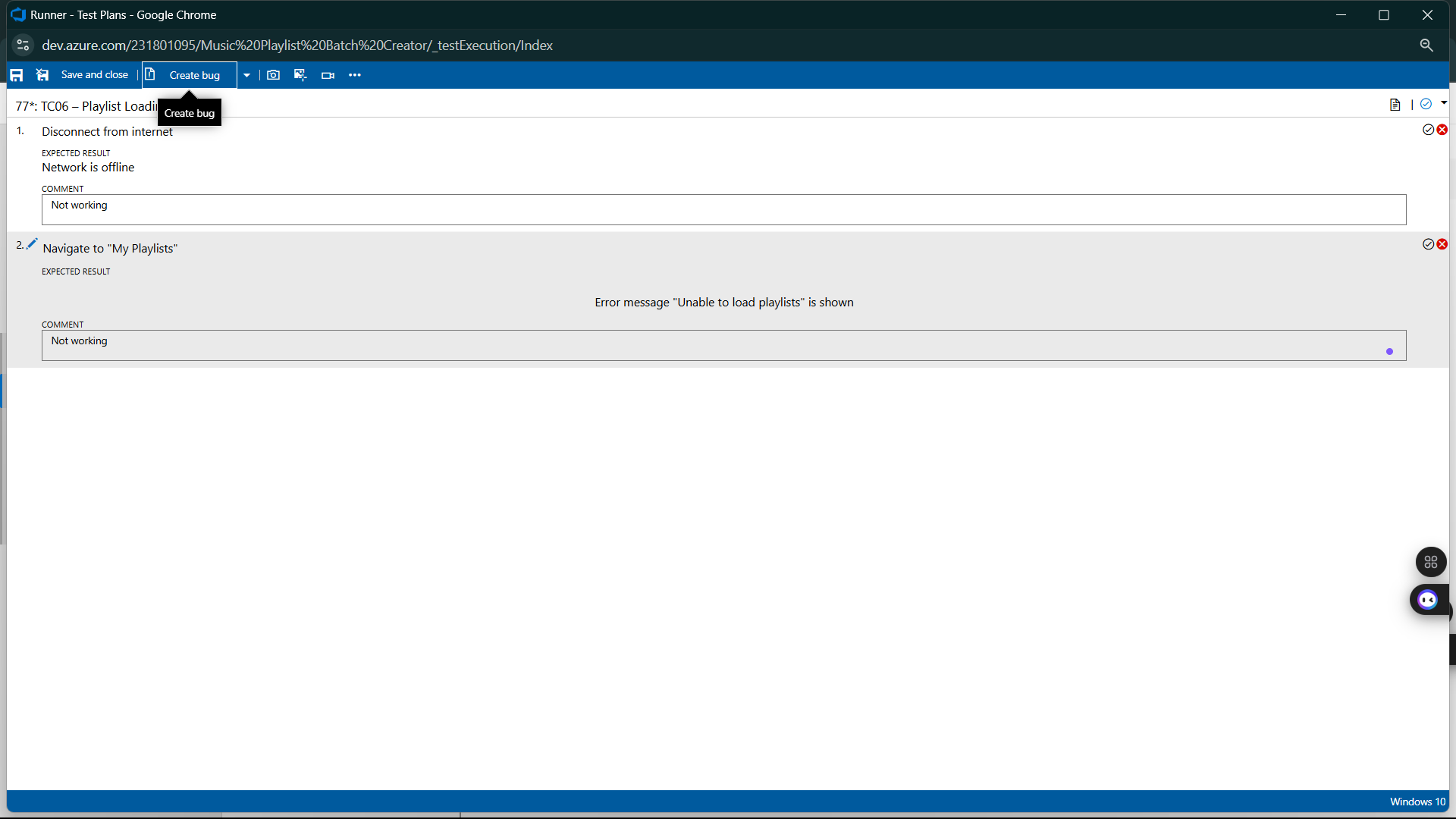


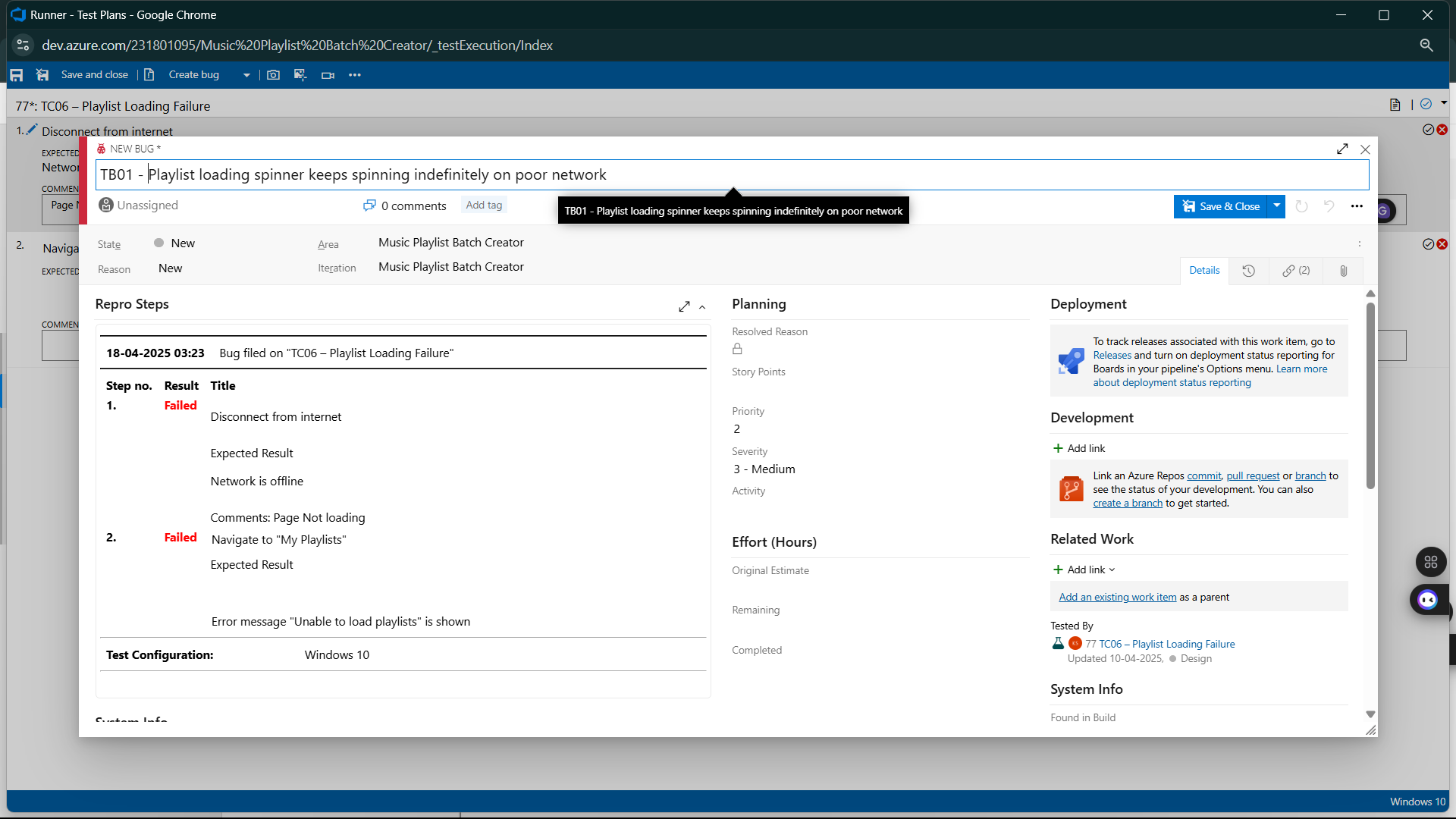


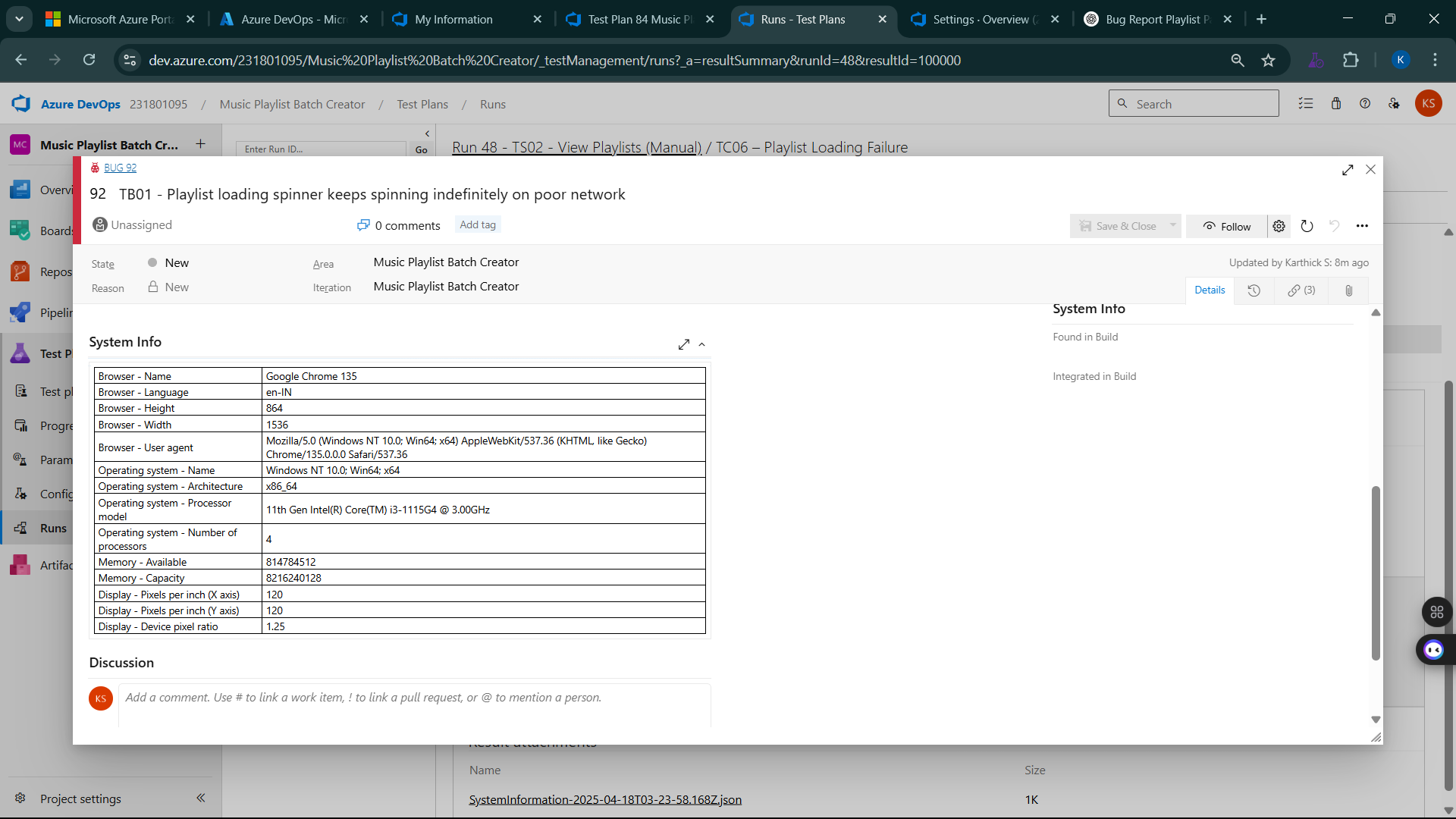
**6.Recording the test case**



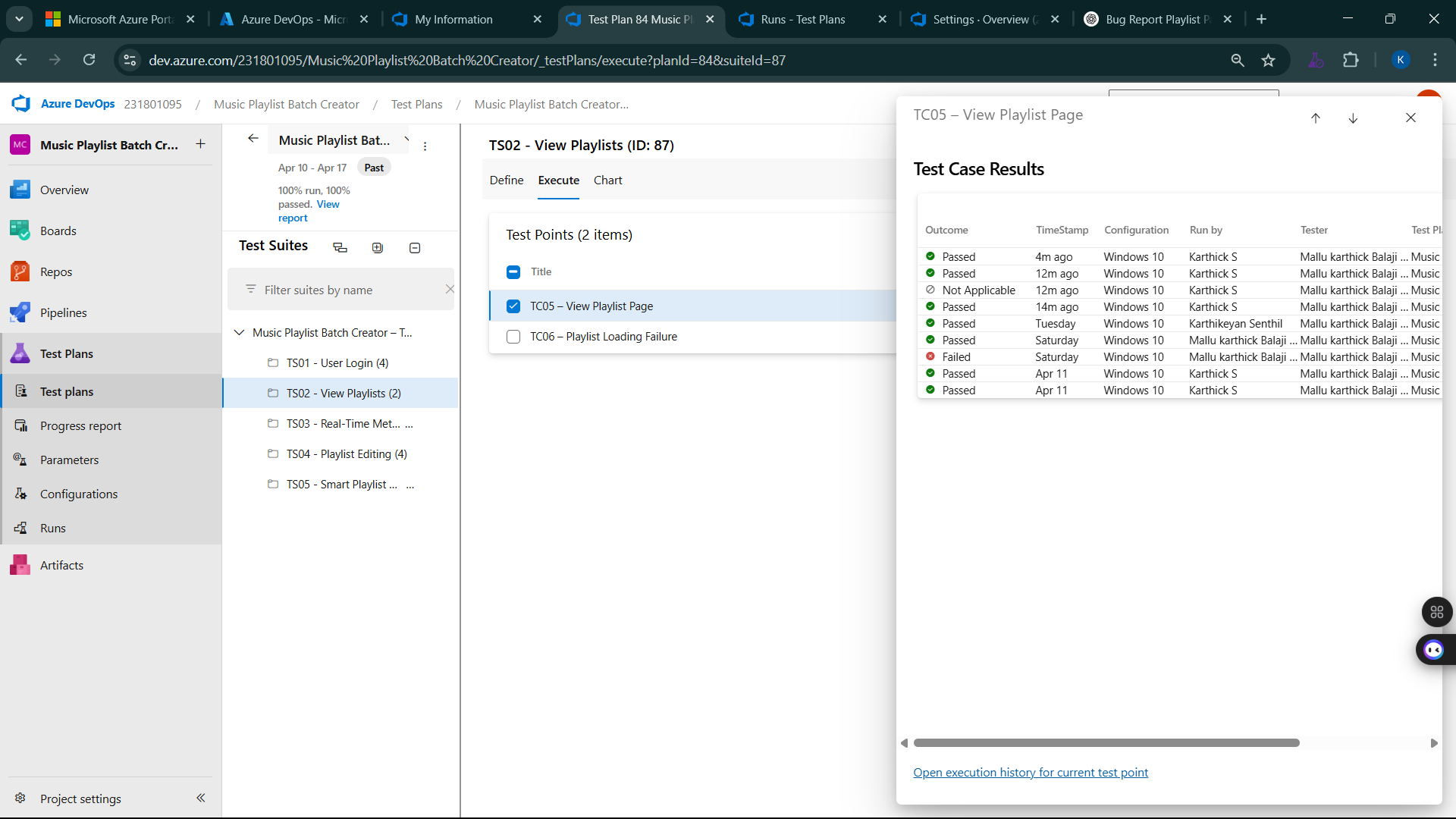
**7.Creating the bug**



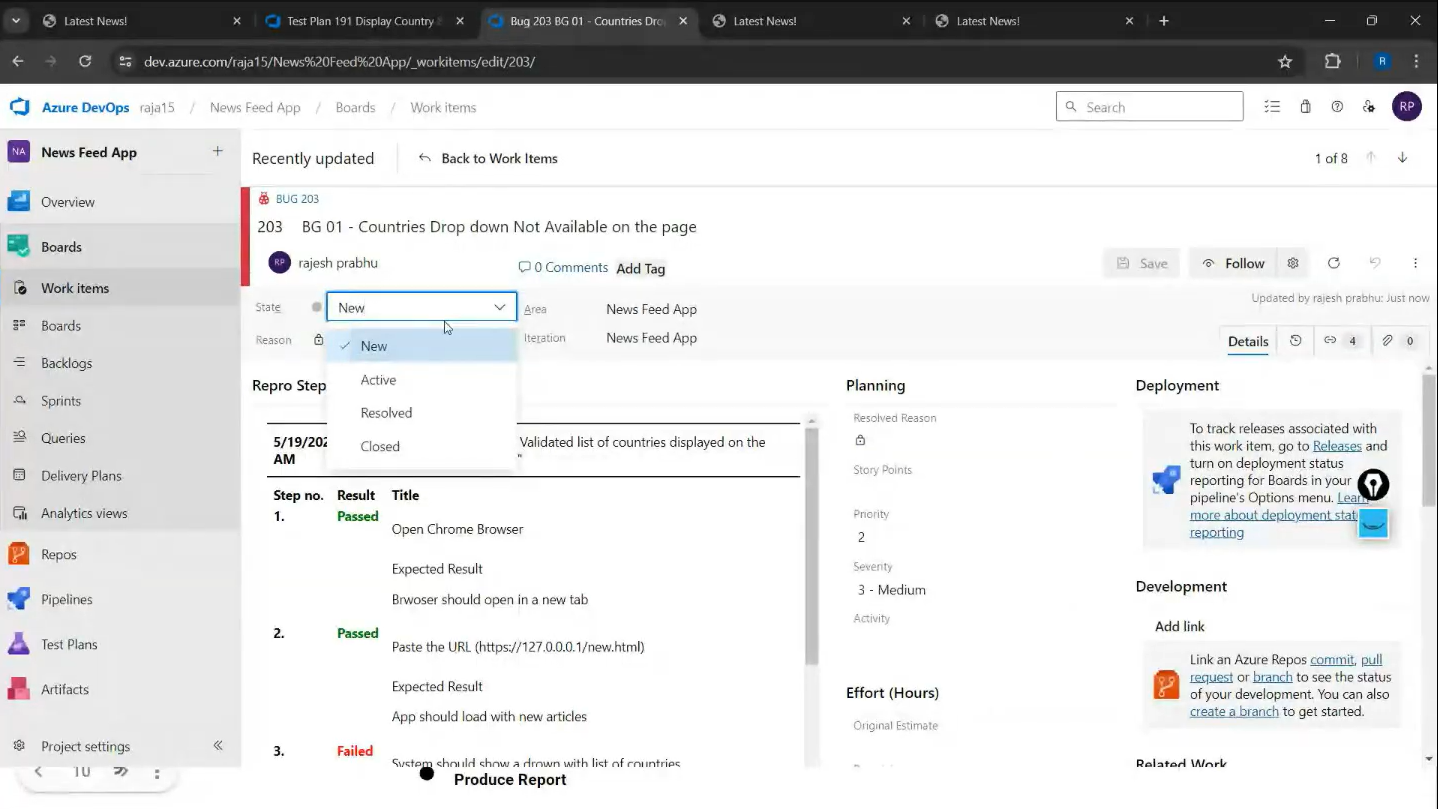




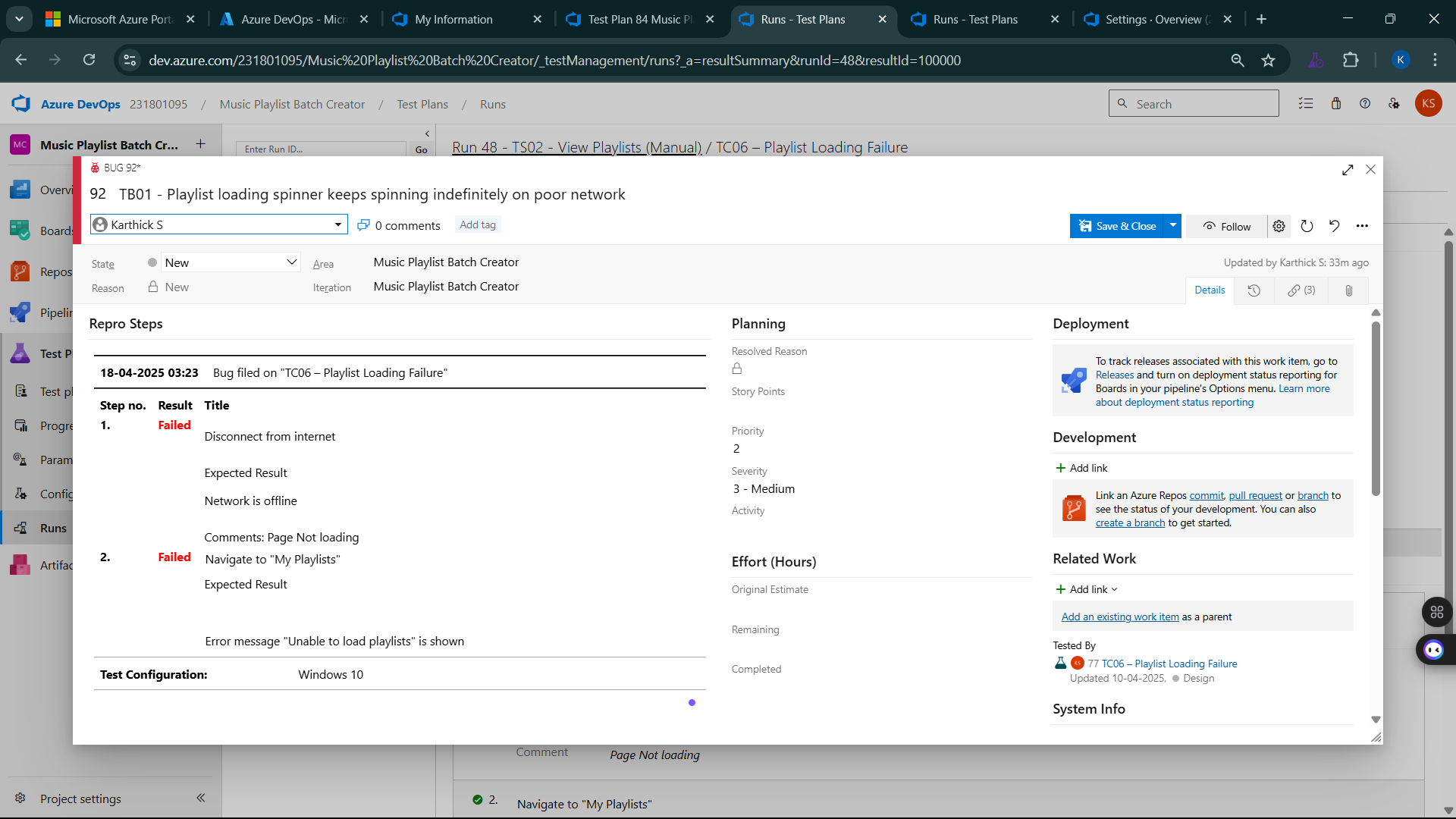
**8.Test case results**



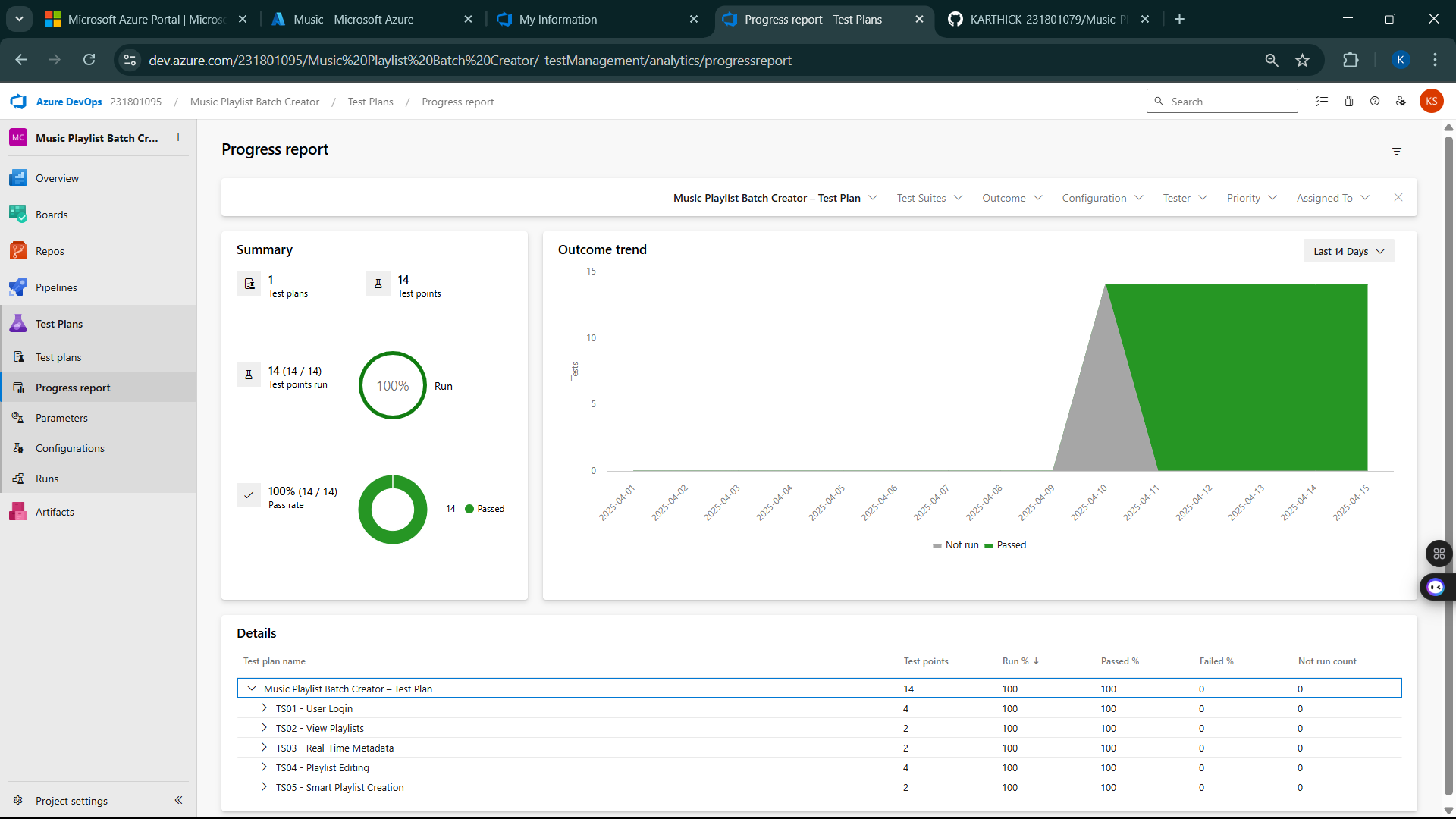
**9.Test report summary**

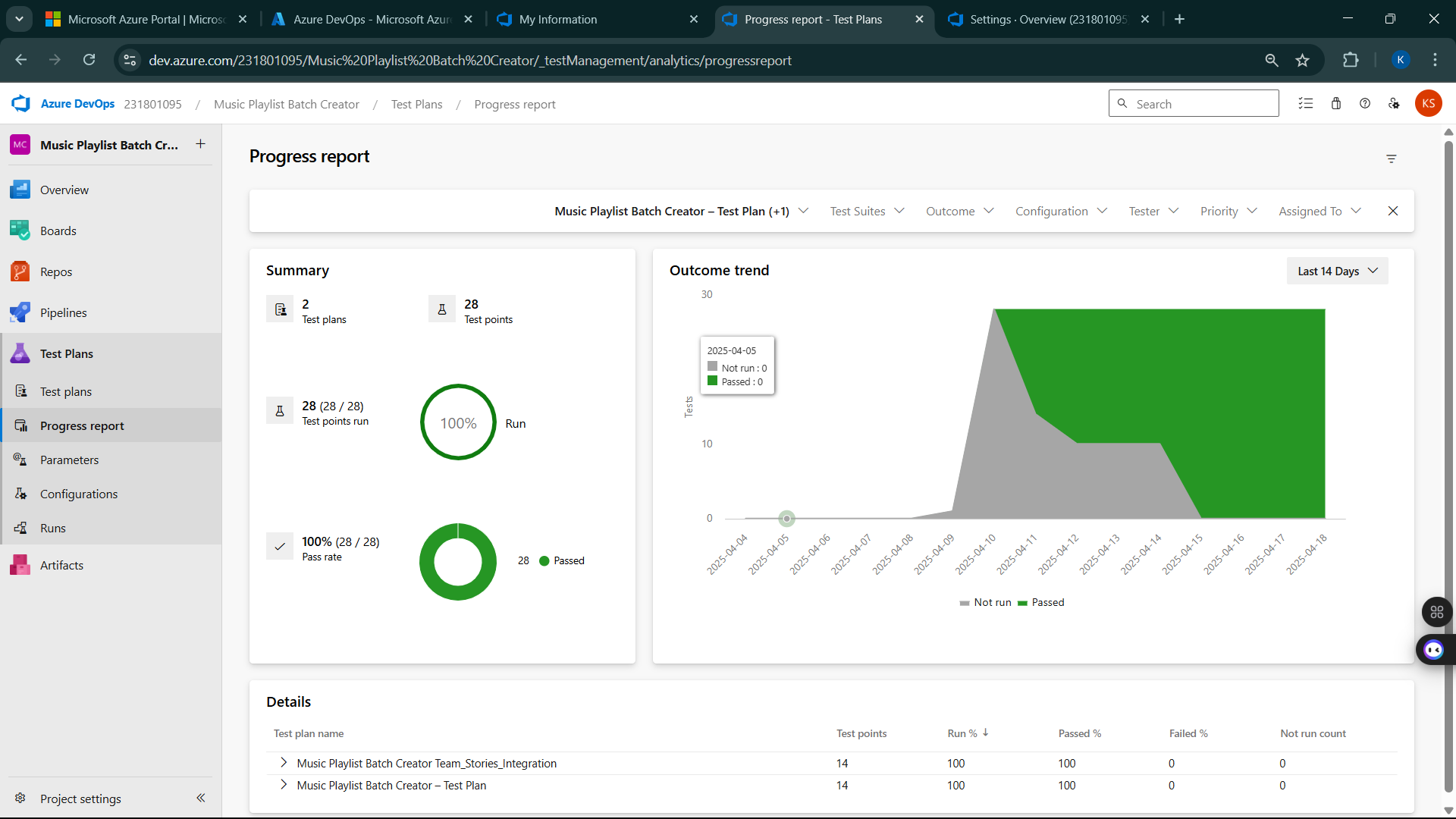


* Assigning bug to the developer and changing state

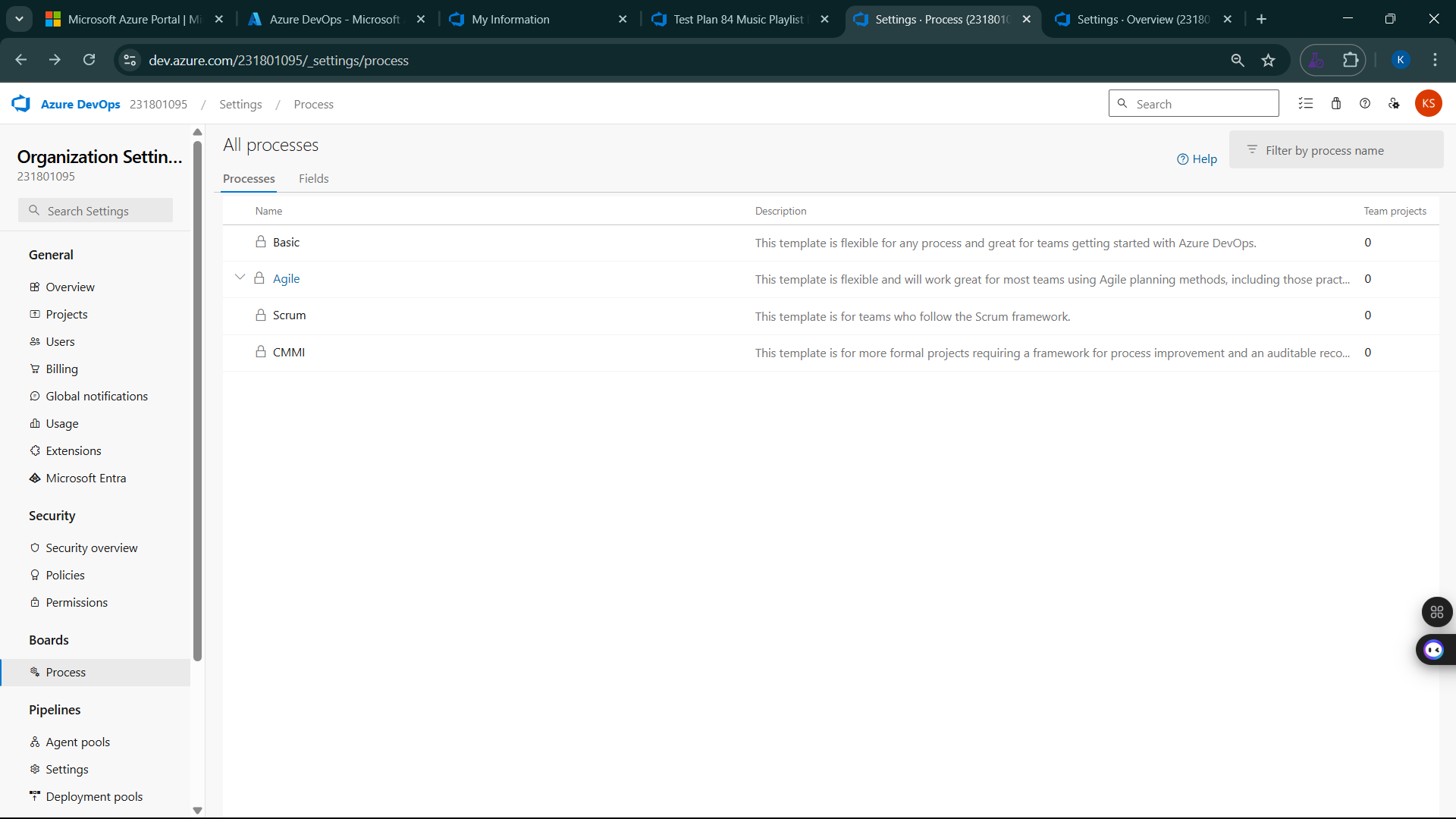


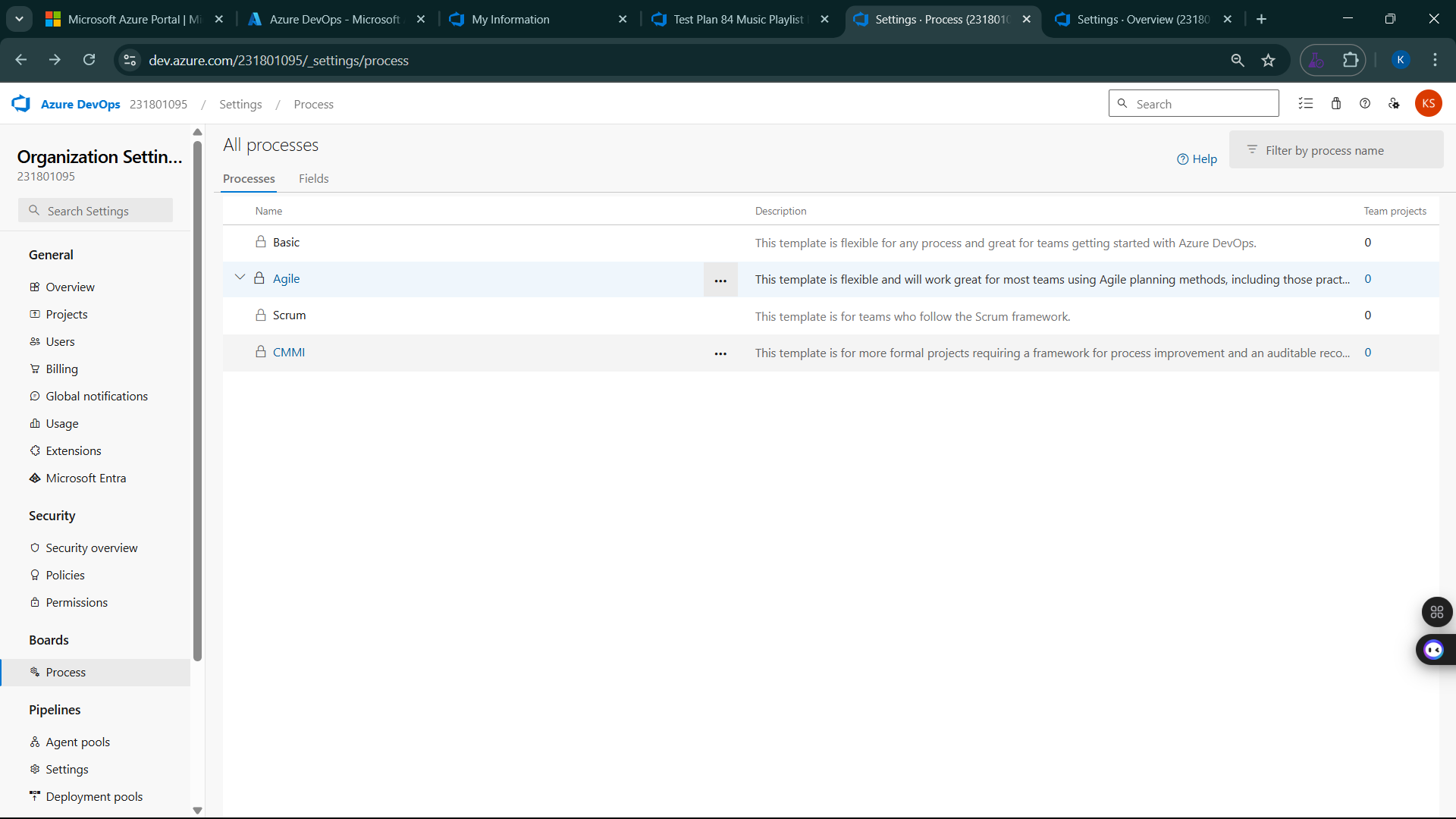
**10.Progress report**

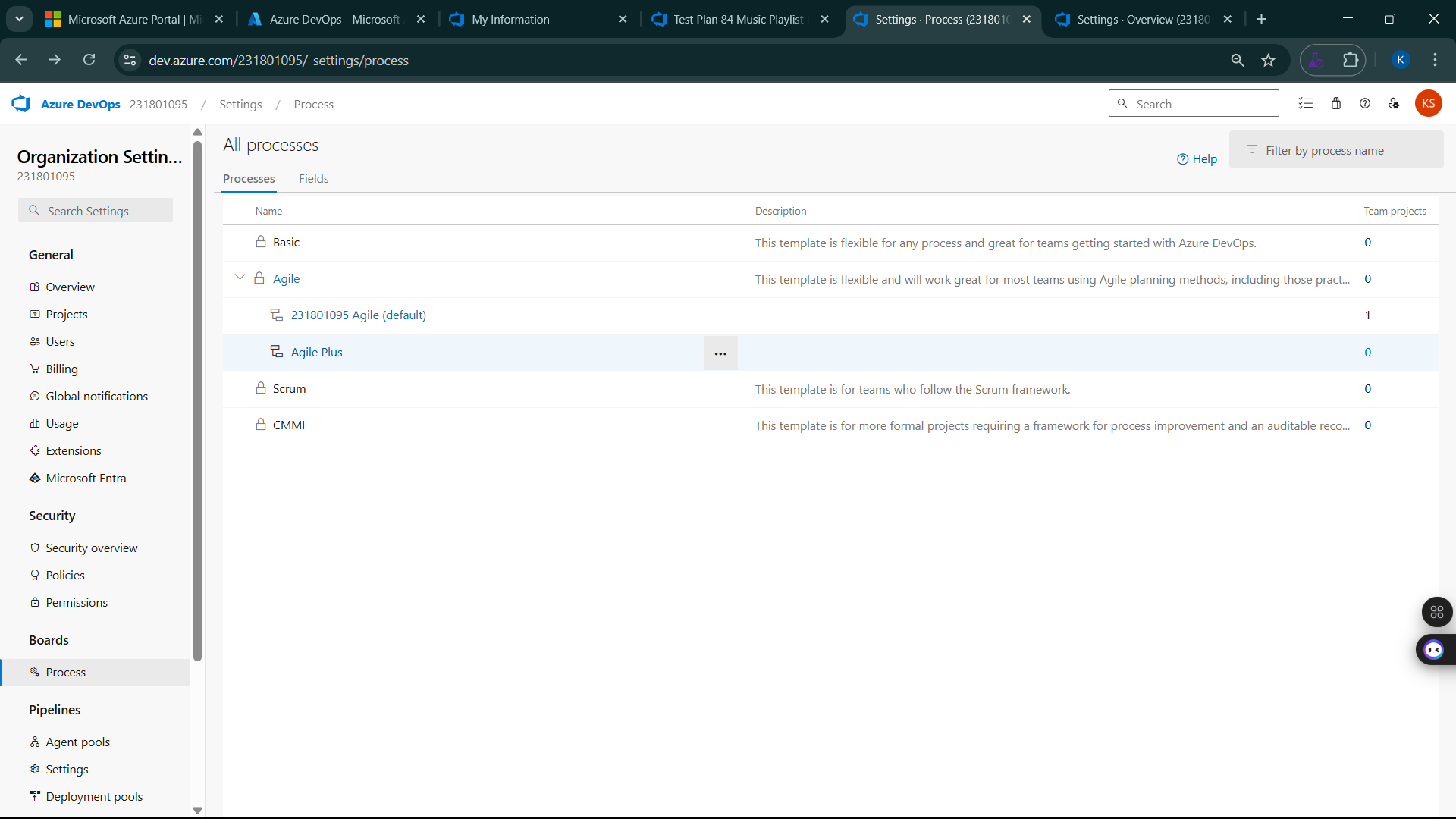




**11.Changing the test template**

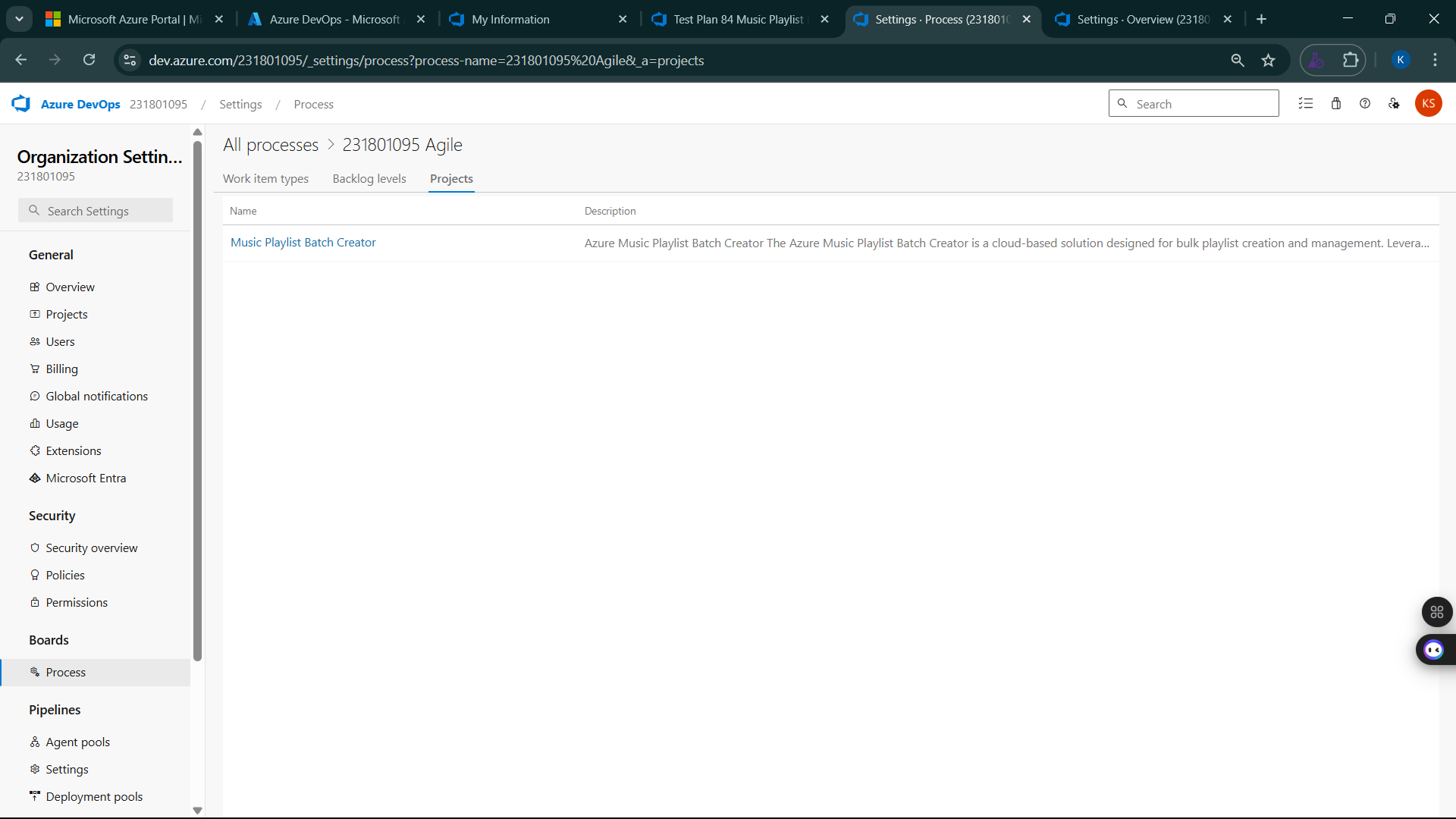


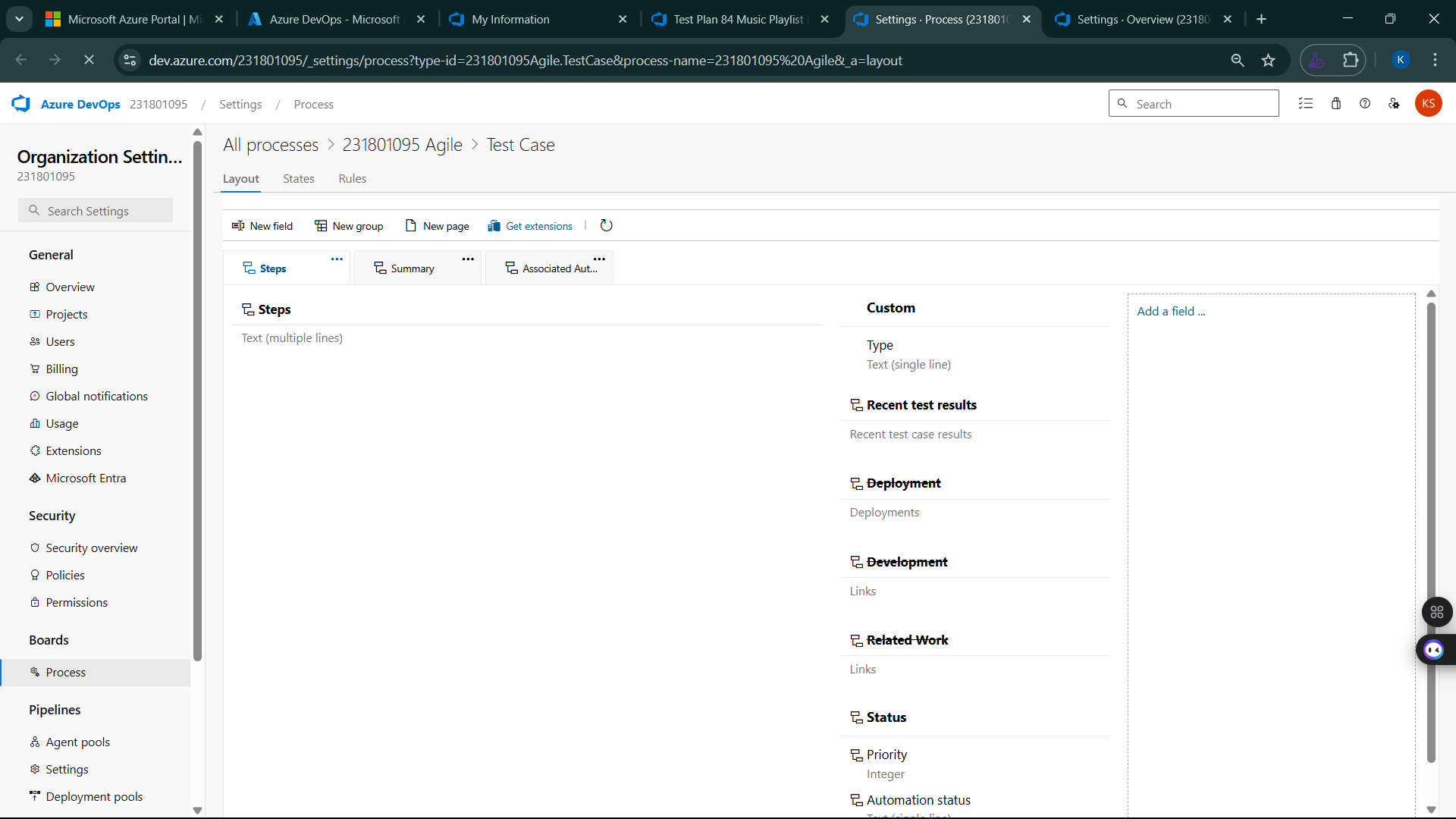




**12.View the new test case template**







**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:

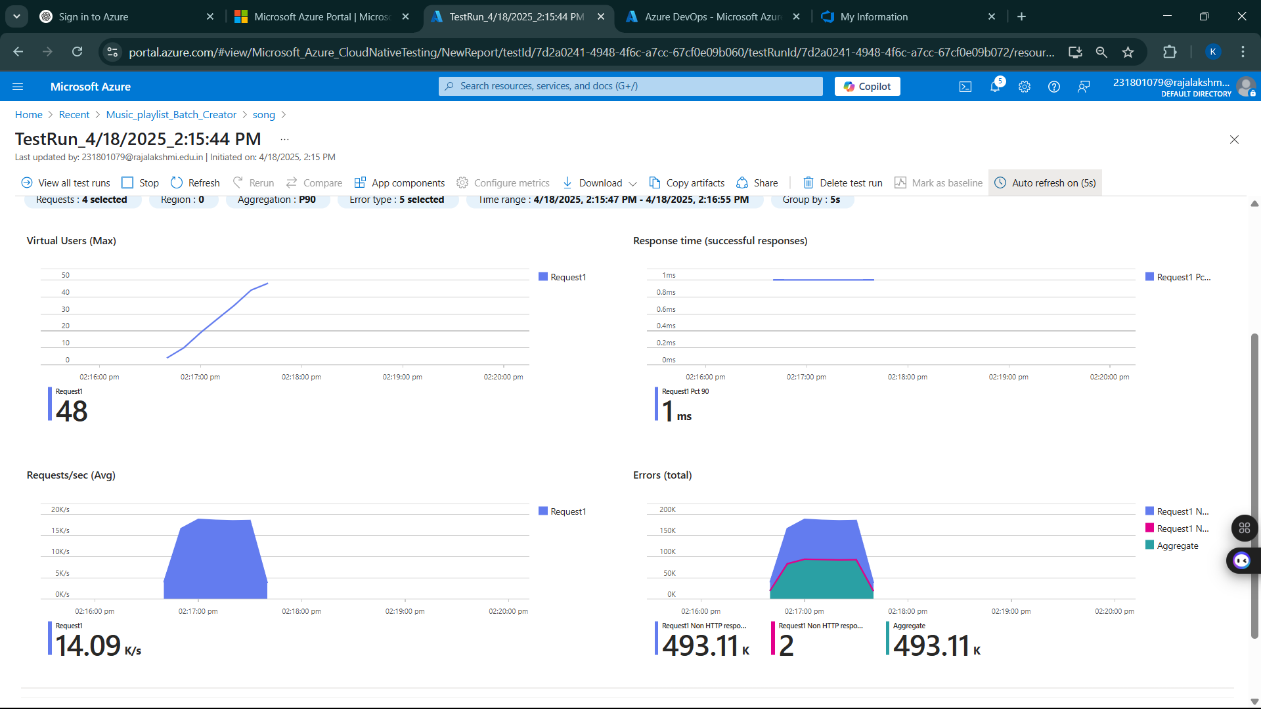
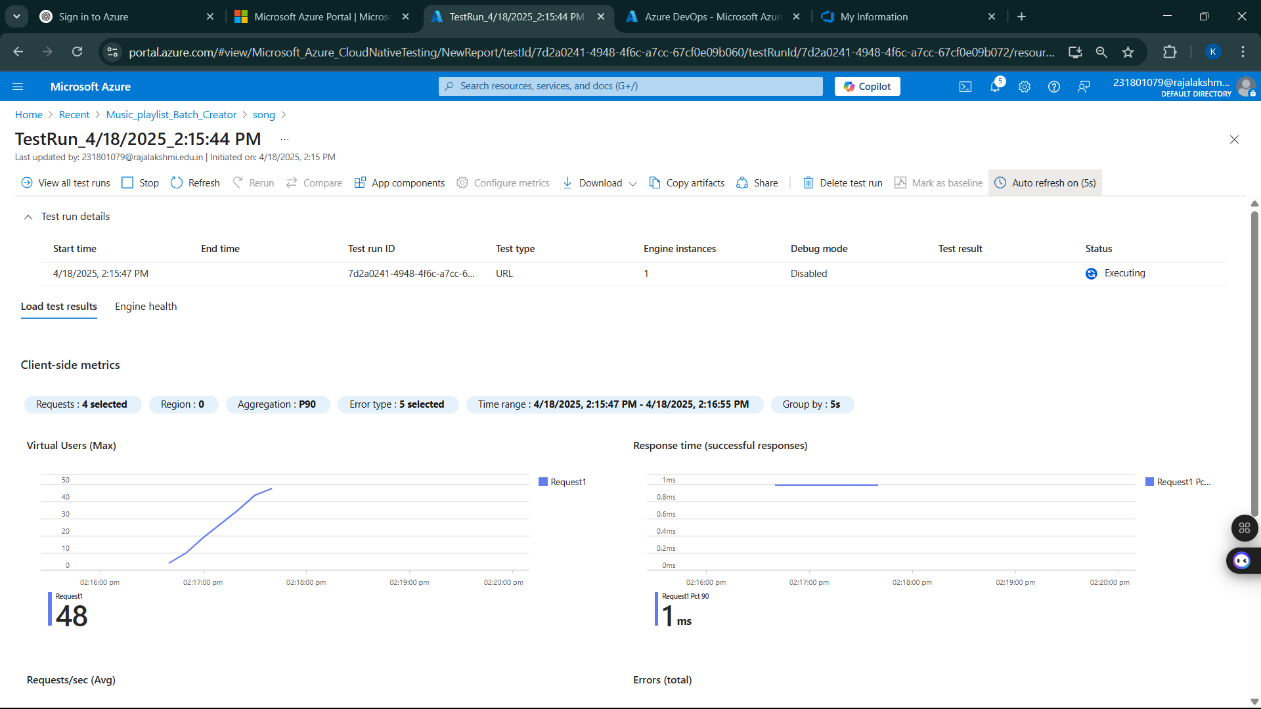
1. Sign in to Azure Portal  
   Go to <https://portal.azure.com> and log in.
2. Create the Resource
   * Go to *Create a resource* → Search for “Azure Load Testing”.
   * Select Azure Load Testing and click Create.
3. Fill in the Configuration Details
   * Subscriptio*n:* Choose your Azure subscription.
   * *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.

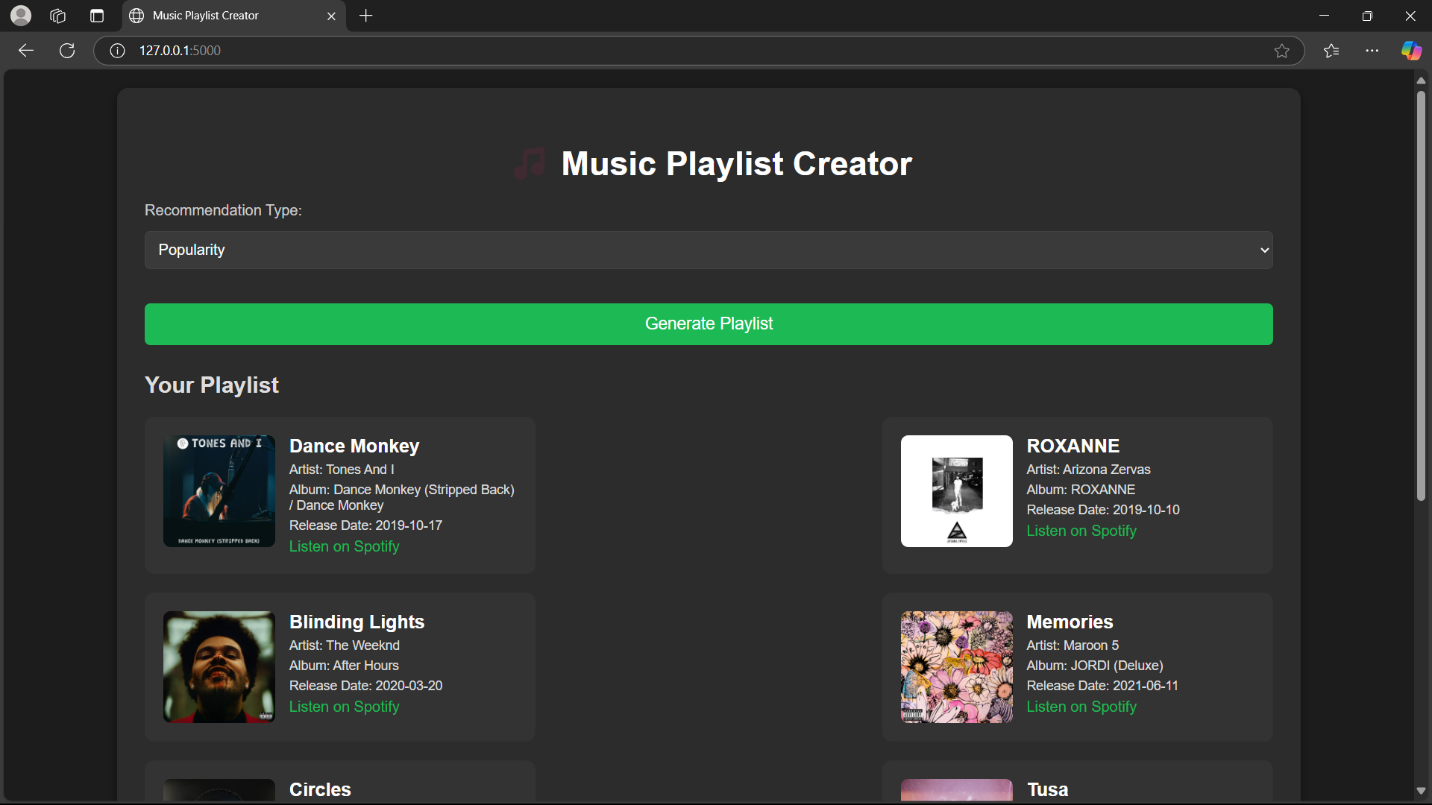
**Steps to Create and Run a Load Test:**

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 music recommendation 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('🎶 Hello from Music Playlist Batch Creator!')"

displayName: "Run a Python script"

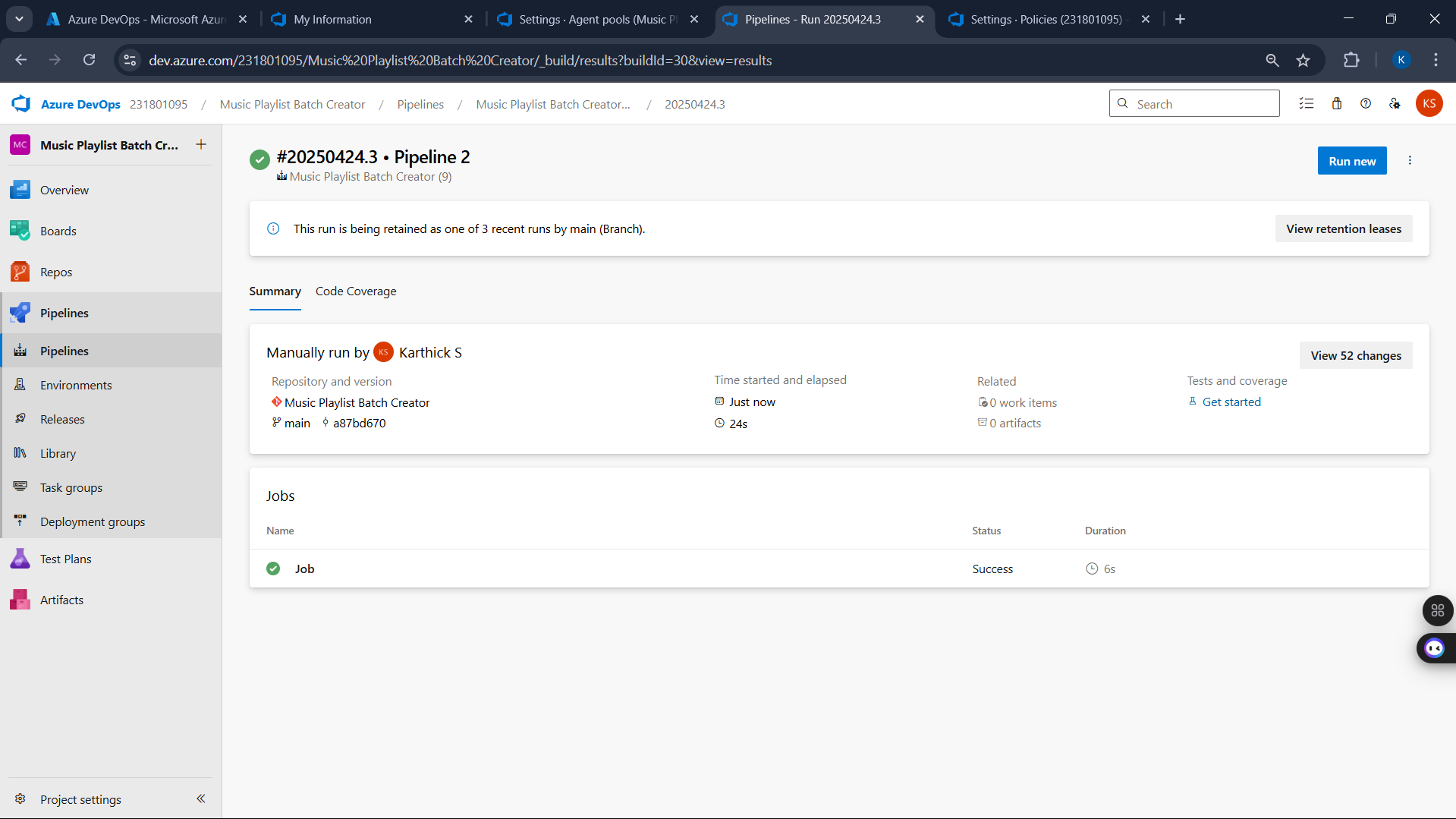
1. Pipeline Tasks Include:

* 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 Music Playlist Batch Creator project.

**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.