**AZURE DEVOPS ENVIRONMENT SETUP**

**EXP NO: 1**

**Aim:**

Create Epic, Features, User Stories, Task

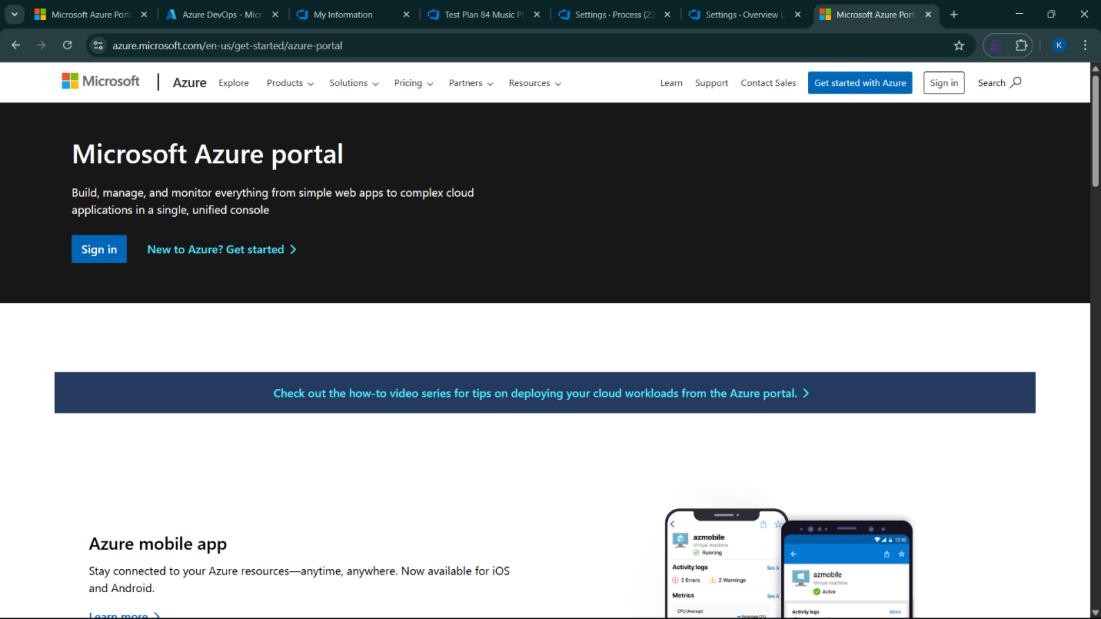
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

**INSTALLATION**

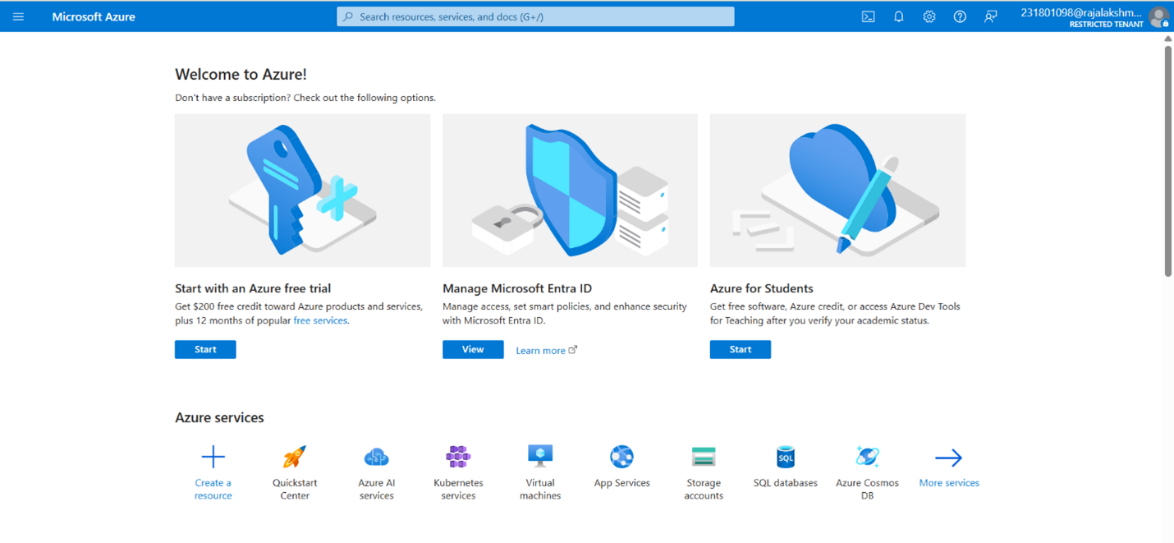
1. Open your web browser and go to the Azure website: [https://azure.microsoft.com/en-us/get-](https://azure.microsoft.com/en-us/get-started/azure-portal) [started/azure-portal](https://azure.microsoft.com/en-us/get-started/azure-portal).

Sign in using your Microsoft account credentials.

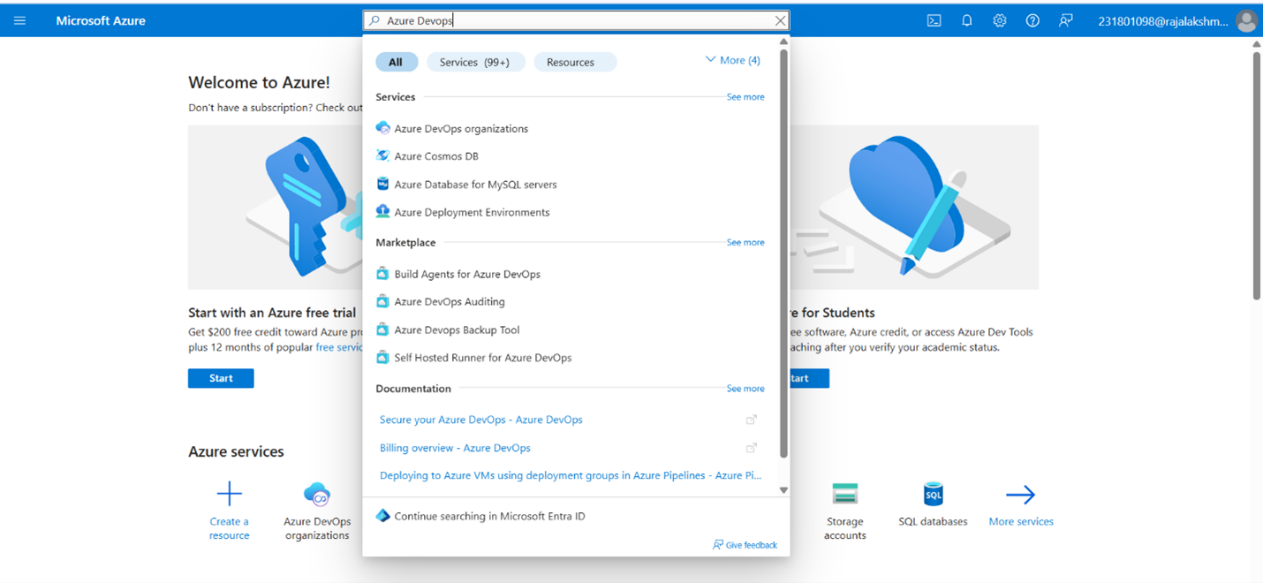
If you don't have a Microsoft account, you can create one here: <https://signup.live.com/?lic=1>



1. Azure home page



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



1. 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.

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

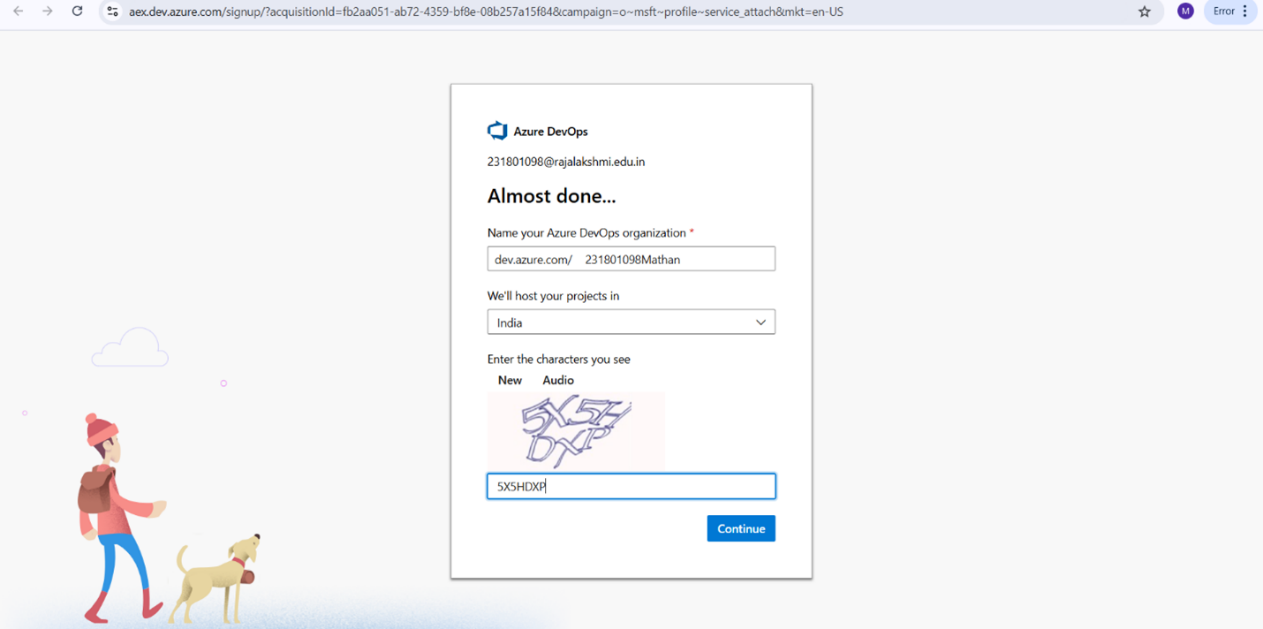
**EXP NO: 2**

**Aim:**

Create Epic, Features, User Stories, Task

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

1. Create An Azure Account

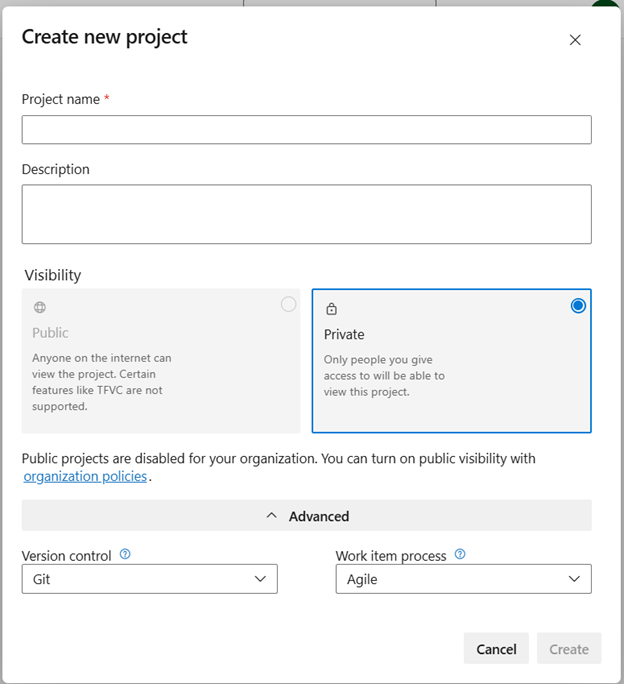


1. Create the First Project in Your Organization
   1. After the organization is set up, you’ll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.
   2. On the organization’s **Home page**, click on the **New Project** button.
   3. Enter the project name, description, and visibility options:

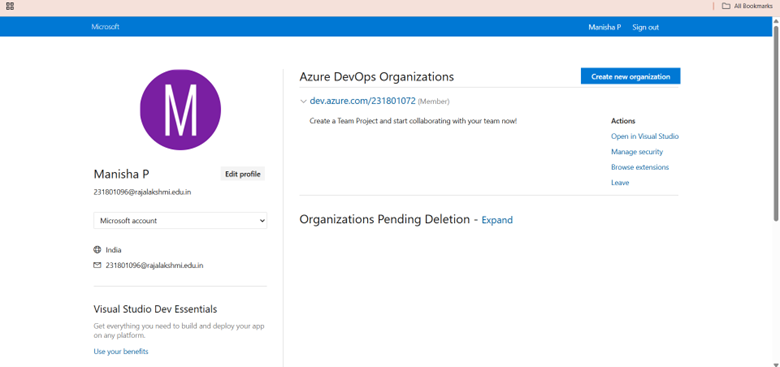
***Name****:* Choose a name for the project (e.g., LMS).

***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).

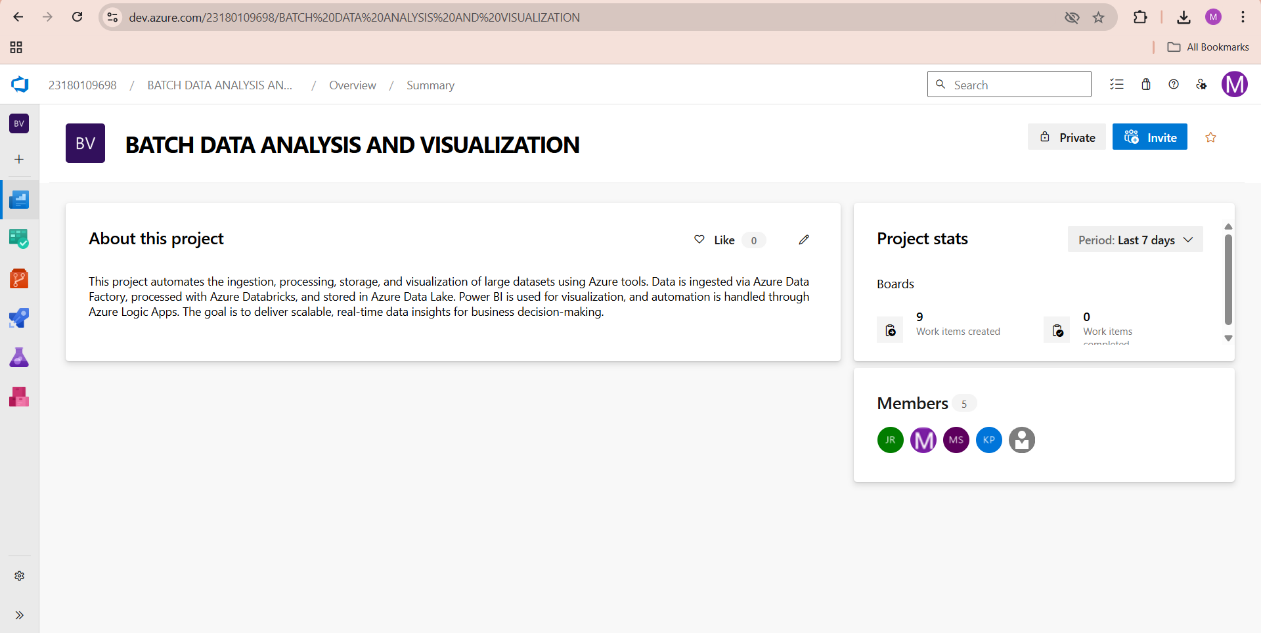
* 1. Once you’ve filled out the details, click **Create** to set up your first project.



1. 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.



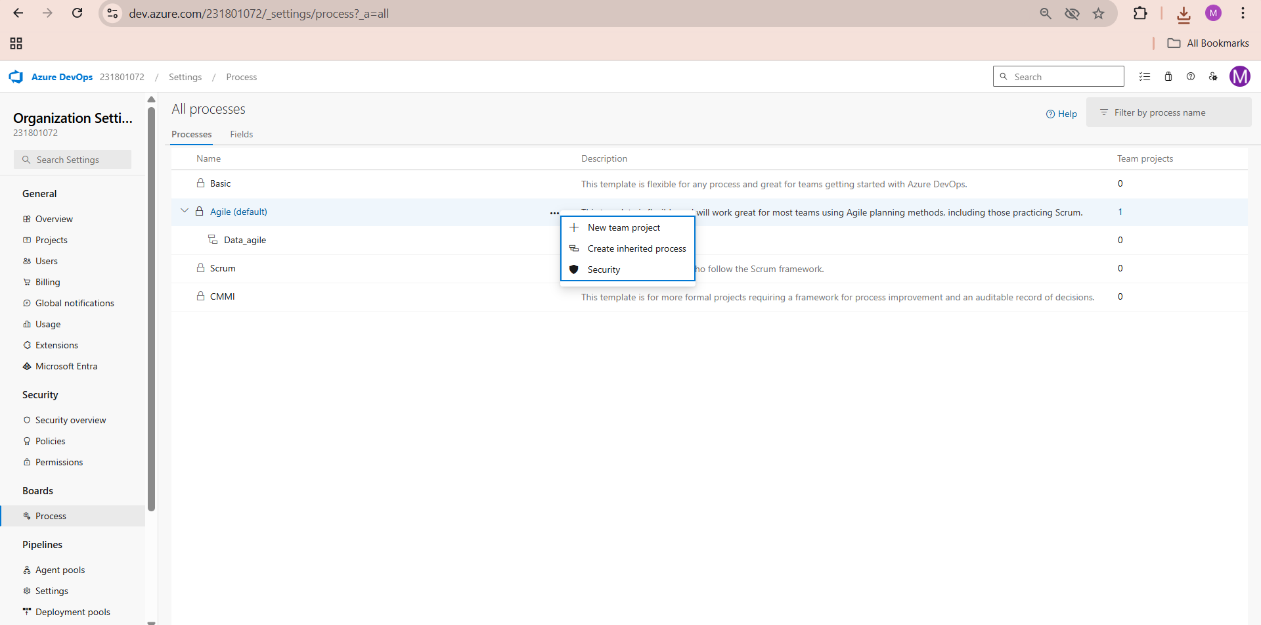
1. Project dashboard

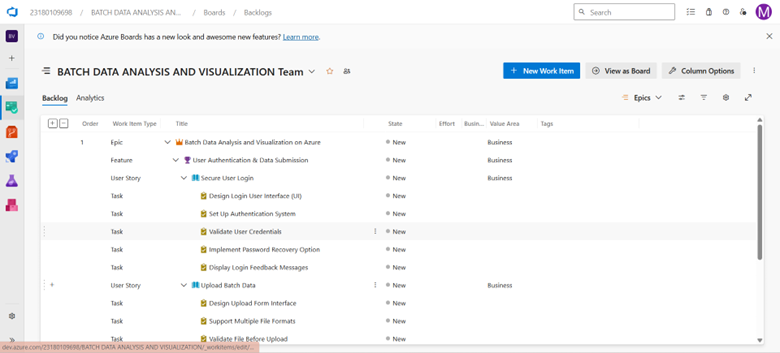


1. To manage user stories:
   1. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards**

page, where you can manage work items, backlogs, and sprints.

* 1. 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:**

setup.

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

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

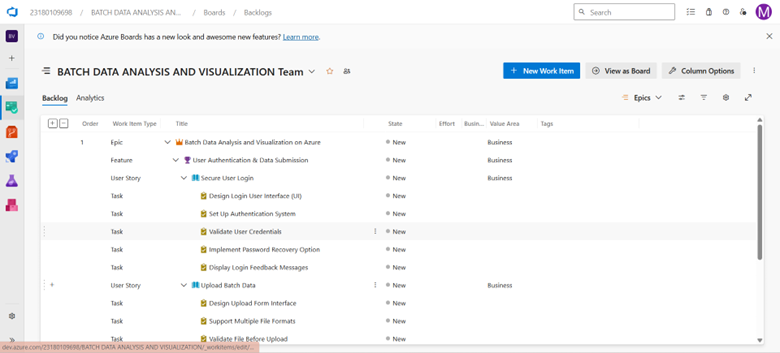
**EXP NO: 3**

**Aim:**

Create Epic, Features, User Stories, Task

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

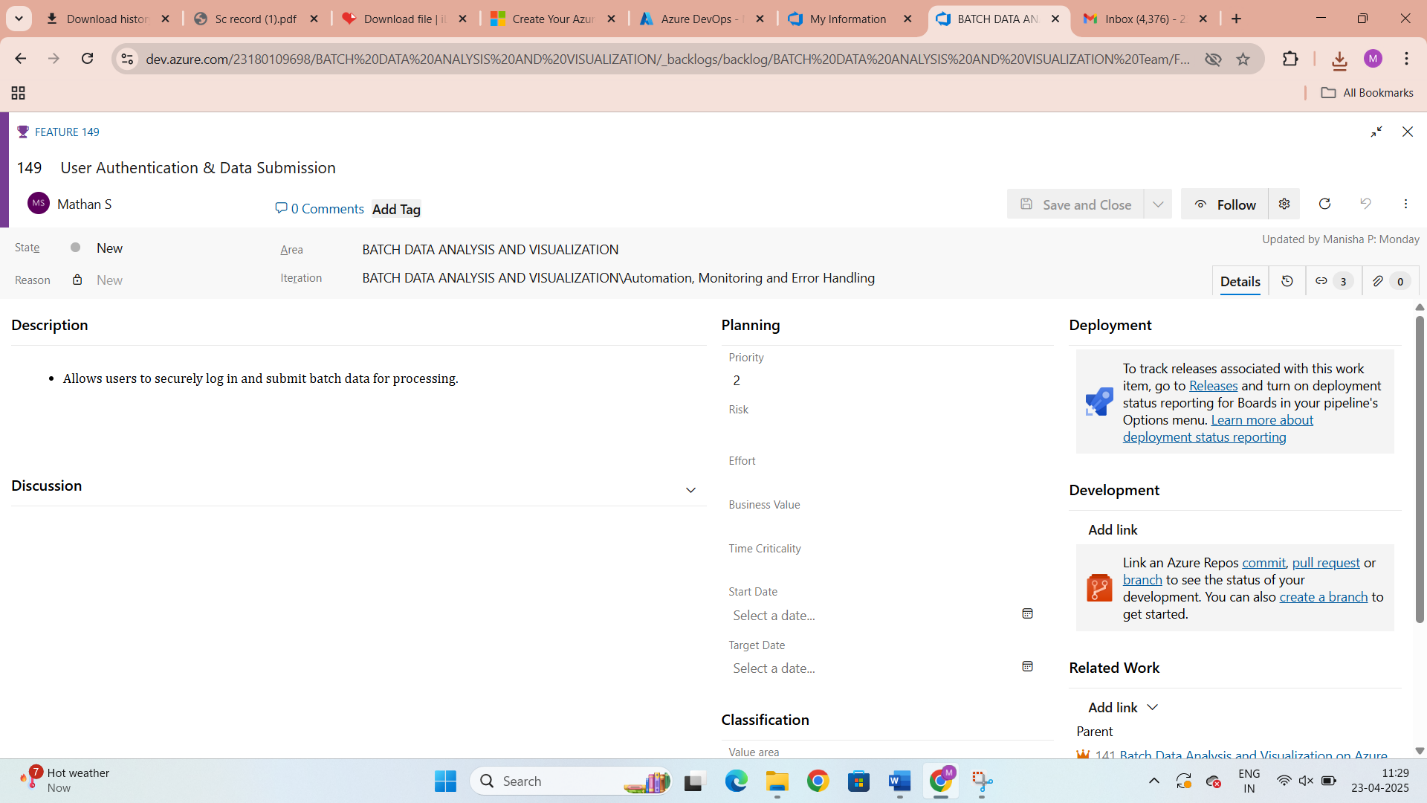
**Create Epic, Features, User Stories, Task**

****

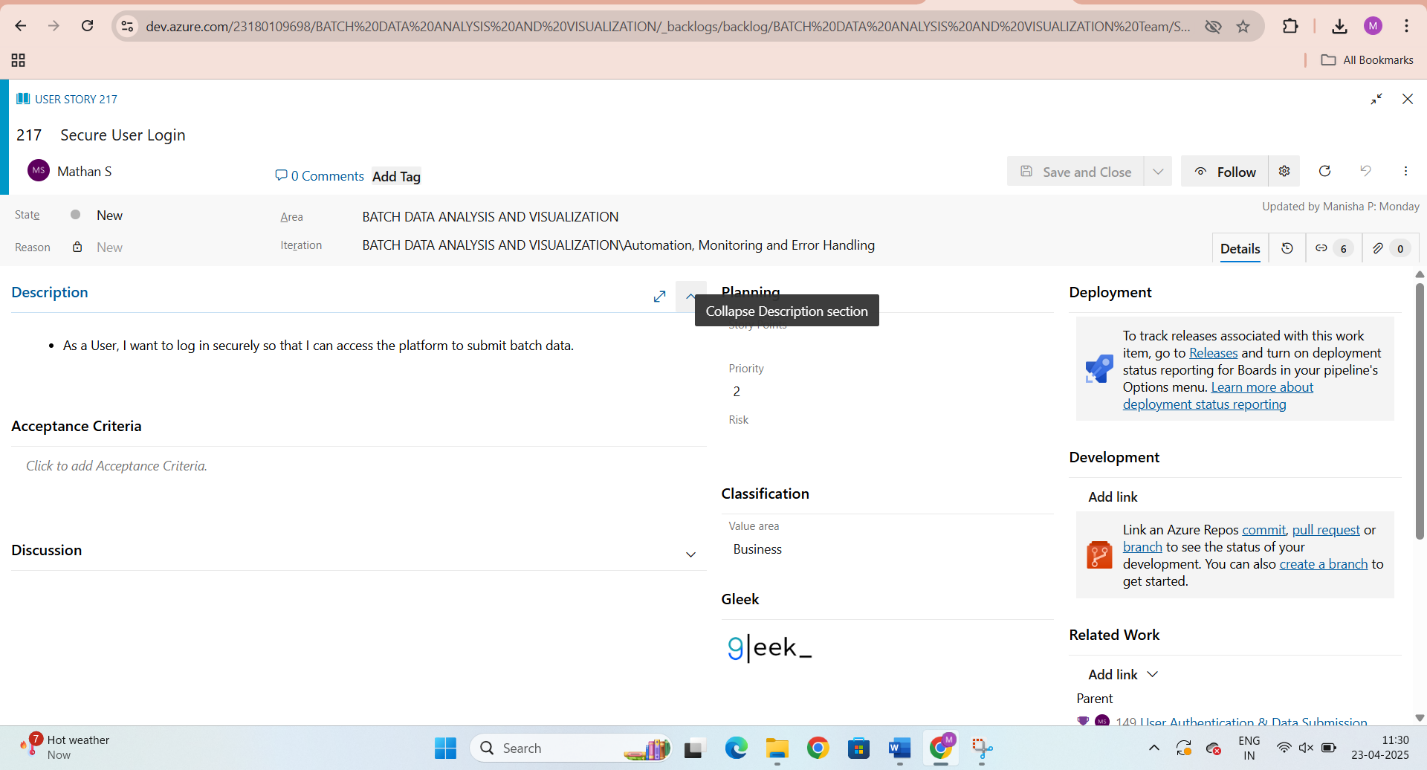
1. **Fill in Epics**

****

1. **Fill in Features**

****

1. **Fill in User Story Details**

****

**Result:**

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

**SPRINT PLANNING**

**EXP NO: 4**

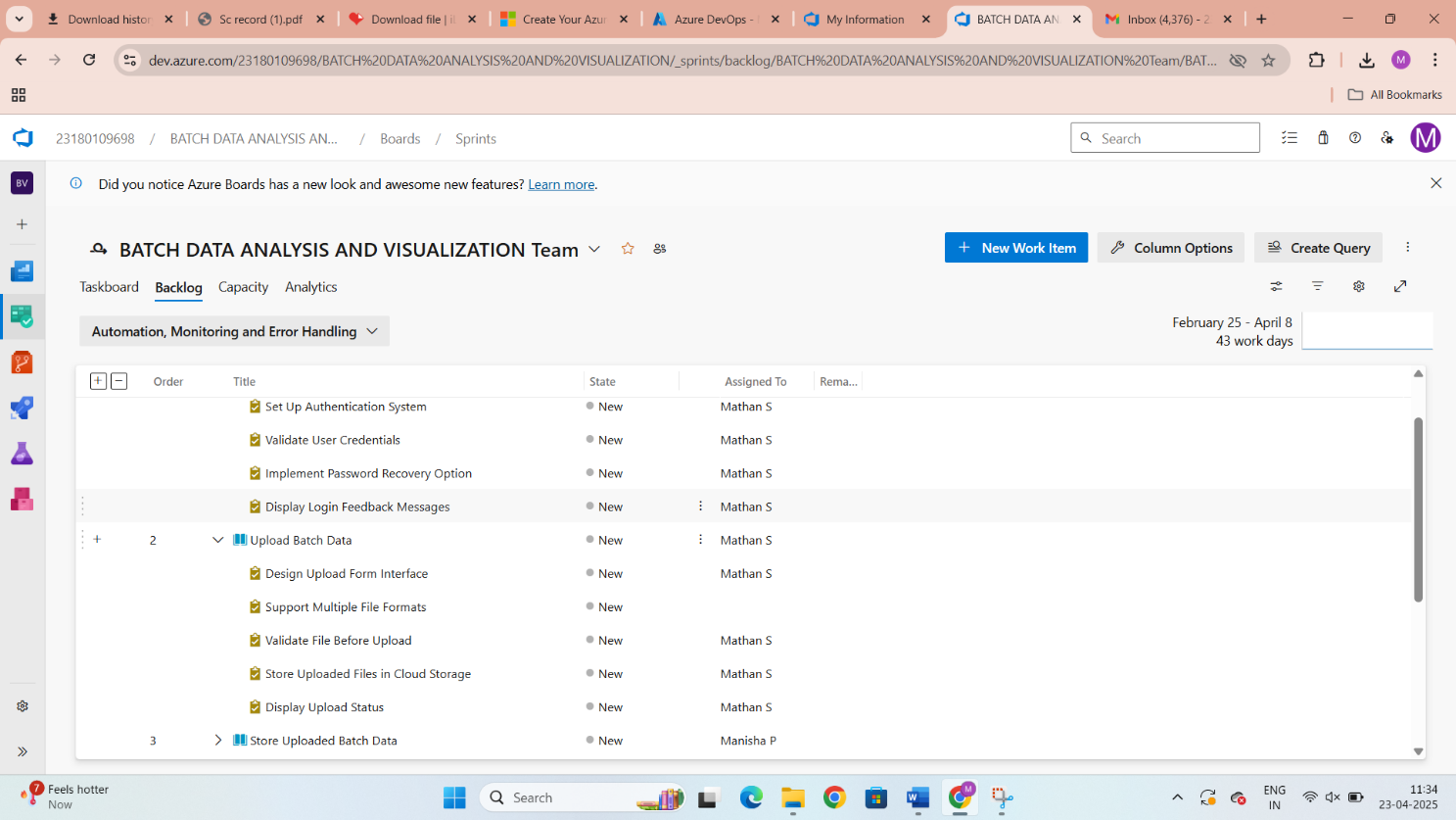
**Aim:**

Create Epic, Features, User Stories, Task

To assign user story to specific sprint for the Batch Data Analysis and Visualization

**Sprint Planning**

**Sprints :**

****

**Result:**

The Sprints are created for the Batch Data Analysis And Visualization.

**POKER ESTIMATION**

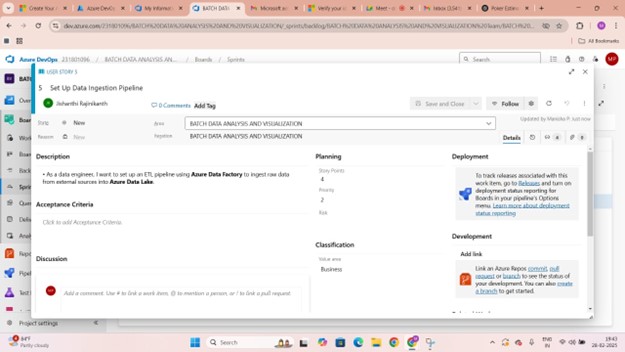
**EXP NO: 5**

**Aim:**

Create Epic, Features, User Stories, Task

Create Poker Estimation for the user stories - Batch Data Analysis And Visualization.

**Poker Estimation**

****

**Result:**

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

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

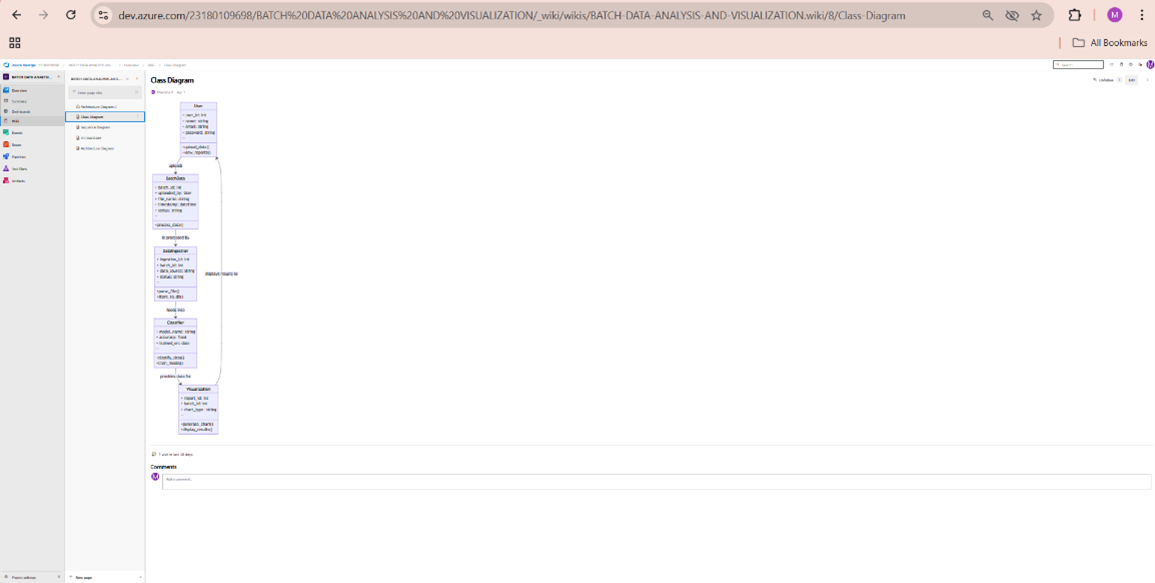
**EXP NO: 6**

**Aim:**

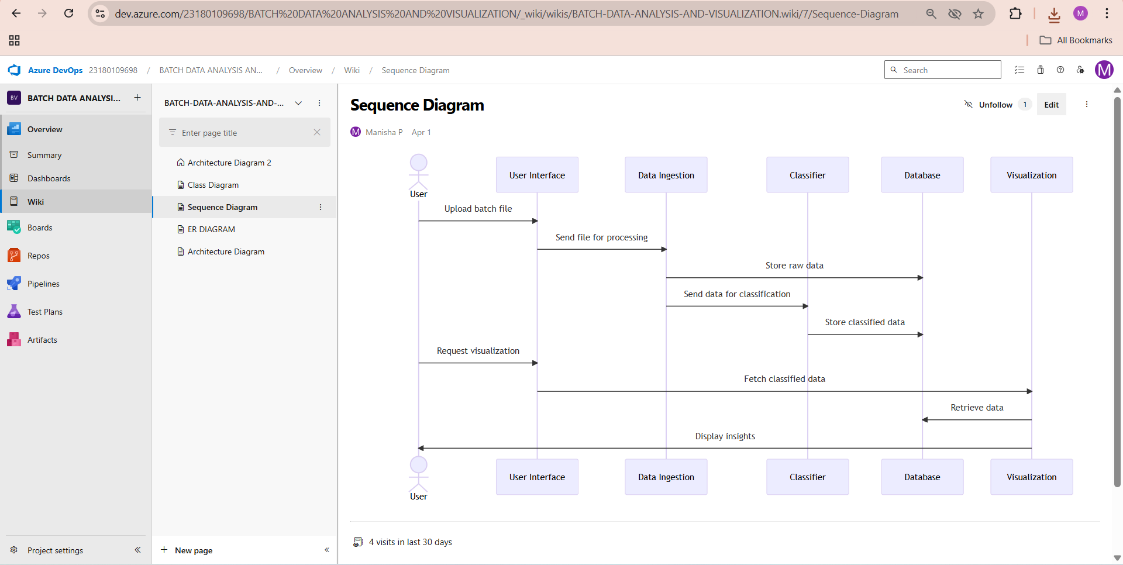
Create Epic, Features, User Stories, Task

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 Music Playlist Batch

Creator.

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

**EXP NO: 7**

**Aim:**

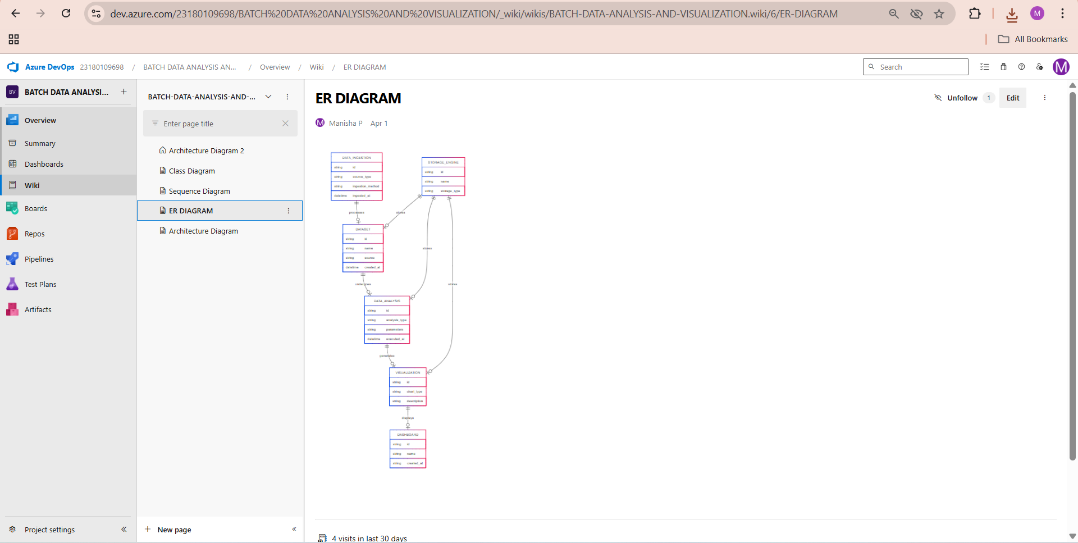
Create Epic, Features, User Stories, Task

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

**7A. Architectural Diagram**

****

**7B.ER Diagram**

****

**Result:**

Creator

The Architecture Diagram and ER Diagram is designed Successfully for the Music Playlist Batch

**TESTING – TEST PLANS AND TEST CASES**

**EXP NO: 8**

**Aim:**

Create Epic, Features, User Stories, Task

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
* Allow users to securely create an account and log into the platform.
* Navigation to Data Upload Feature
* Enable users to navigate from the dashboard to the data upload section.
* Data Upload and Validation
* Allow users to upload CSV/Excel files and validate uploaded data for structure and format.
* Data Preprocessing and Cleaning
* Perform data cleaning activities like handling missing values, removing duplicates, and standardizing data.
* Data Visualization
* Generate visual insights (charts, graphs) based on processed data.

**2. Define User Interactions**

* Each test case simulates a real user behavior:
* Logging in to the platform.
* Uploading raw data files.
* Preprocessing and cleaning uploaded data.
* Generating charts and graphs from the processed data.
* Handling errors when invalid data or incorrect formats are uploaded.

**3. Design Happy Path Test Cases**

* Focused on validating that all features work properly under normal conditions:
* User logs in successfully with valid credentials.
* Data file (CSV/Excel) is uploaded successfully when correct format and structure are used.
* Missing values are handled correctly during preprocessing.
* Duplicate records are identified and removed.
* Graphs are generated correctly based on selected fields.

**4. Design Error Path Test Cases**

* Simulate negative or unexpected scenarios to test system robustness:
* Login fails when credentials are incorrect.
* Upload fails when file format is unsupported (e.g., .txt file).
* Upload fails when mandatory columns are missing in the file.
* Preprocessing fails when the dataset is empty.
* Visualization fails gracefully if selected fields are missing.

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

* Each test case includes:
* Step-by-step actions (e.g., click "Upload," select file, submit)
* Expected outcomes (e.g., "File uploaded successfully," "Error message displayed")
* This helps ensure clarity for manual testers and automation teams.

**6. Use Clear Naming and IDs**

* Test cases are clearly named and numbered for easy identification:
* TC01 – Successful Login
* TC02 – Successful Data Upload
* TC03 – Invalid Login Credentials
* TC04 – Upload File with Invalid Format
* TC05 – Generate Graph from Cleaned Data

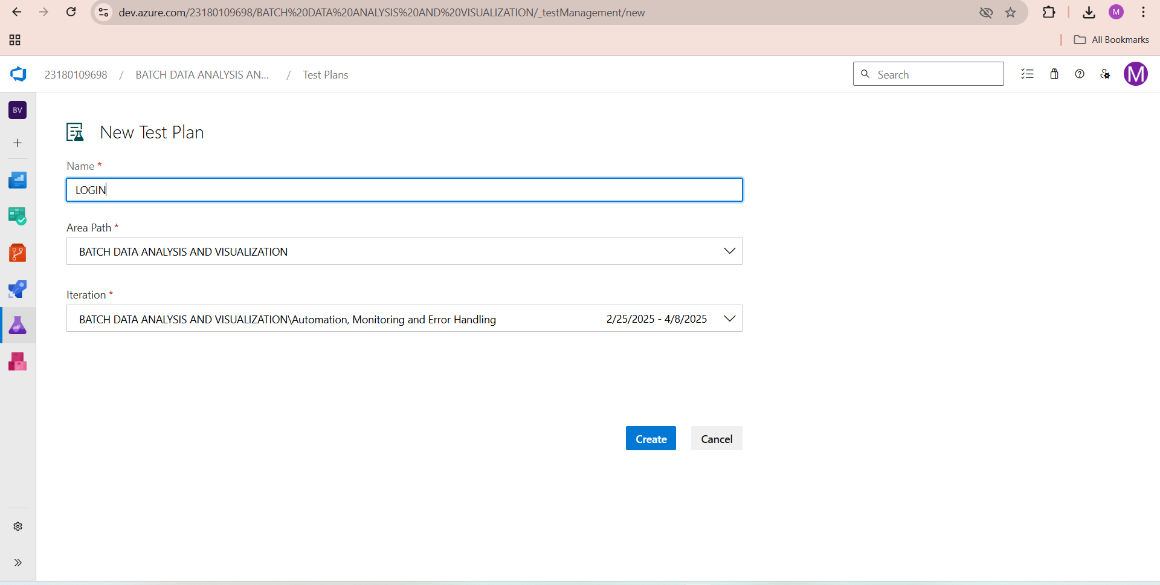
**7. Separate Test Suites**

* Test cases are organized into logical groups for better execution:
* Functional Tests:
* Login
* Upload
* Data Cleaning
* Visualization
* UI Tests:
* Navigation to Upload Section
* Upload Button Visibility
* Visualization Chart Display
* Edge Case Tests:
* Invalid file formats
* Missing fields in data
* Uploading empty datasets
* Login with empty fields

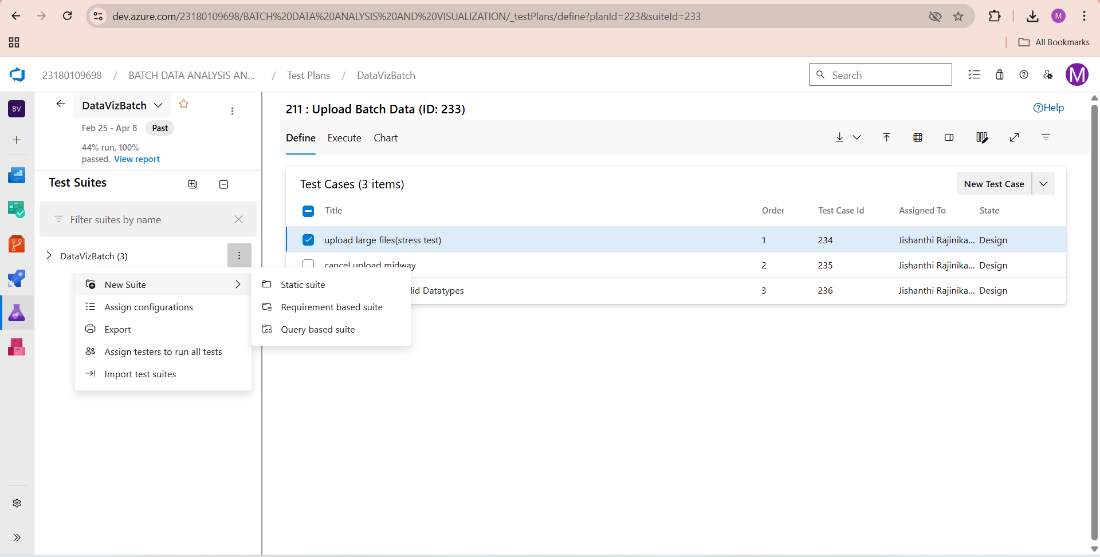
**8. Prioritize and Review**

* Critical user actions like Login, Upload, and Visualization are marked as High Priority.
* Test cases are reviewed for completeness and traceability to feature requirements.
* Priority factors considered:
* Impact on system functionality
* Business importance
* Technical complexity

**New test plan**

****

* **Test suite**



**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 upload and view my data files. (ID: 101)
* As a user, I need to upload CSV/Excel files easily for analysis. (ID: 102)
* As a user, I want the system to clean and preprocess my uploaded data automatically. (ID: 103)
* As a user, I should be able to generate charts and graphs from my processed data. (ID: 104)
* As a user, I need clear error messages if upload or analysis fails. (ID: 105)

**Test Suites**

**Test Suite: TS01 – User Login (ID: 110)**

**1. TC01 – Successful Login**

* **Action:**
  + Go to the Login page.
  + Enter valid username and password.
  + Click "Login."
* **Expected Results:**
  + Login form accepts data.
  + User is redirected to the dashboard.
* **Type:** Happy Path

**2. TC02 – Login with Invalid Credentials**

* **Action:**
  + Go to the Login page.
  + Enter invalid username or password.
  + Click "Login."
* **Expected Results:**
  + Error message: "Invalid Username or Password" is displayed.
* **Type:** Error Path

**Test Suite: TS02 – Data Upload (ID: 111)**

**1. TC03 – Successful Data Upload**

* **Action:**
  + Login successfully.
  + Navigate to the "Upload" section.
  + Select a valid CSV/Excel file.
  + Click "Submit."
* **Expected Results:**
  + File is uploaded successfully.
  + Confirmation message: "Upload Successful."
* **Type:** Happy Path

**2. TC04 – Upload Unsupported File Format**

* **Action:**
  + Login successfully.
  + Navigate to "Upload."
  + Select a .txt or .docx file.
  + Click "Submit."
* **Expected Results:**
  + Error message: "Unsupported file format."
* **Type:** Error Path

**Test Suite: TS03 – Data Preprocessing (ID: 112)**

**1. TC05 – Successful Data Cleaning**

* **Action:**
  + After successful upload, click "Start Preprocessing."
  + Allow the system to clean data (remove duplicates, handle missing values).
* **Expected Results:**
  + Data is cleaned and summary report is shown.
* **Type:** Happy Path

**2. TC06 – Preprocessing Fails on Empty File**

* **Action:**
  + Upload an empty CSV file.
  + Click "Start Preprocessing."
* **Expected Results:**
  + Error message: "Uploaded file is empty."
* **Type:** Error Path

**Test Suite: TS04 – Data Visualization (ID: 113)**

**1. TC07 – Generate Graph Successfully**

* **Action:**
  + After preprocessing, select fields for visualization.
  + Choose graph type (bar, pie, line).
  + Click "Generate Graph."
* **Expected Results:**
  + Graph is displayed correctly based on selected data.
* **Type:** Happy Path

**2. TC08 – Graph Generation Fails without Selecting Fields**

* **Action:**
  + Skip selecting fields.
  + Click "Generate Graph."
* **Expected Results:**
  + Error message: "Please select fields to generate graph."
* **Type:** Error Path

**Test Suite: TS05 – Error Handling & Alerts (ID: 114)**

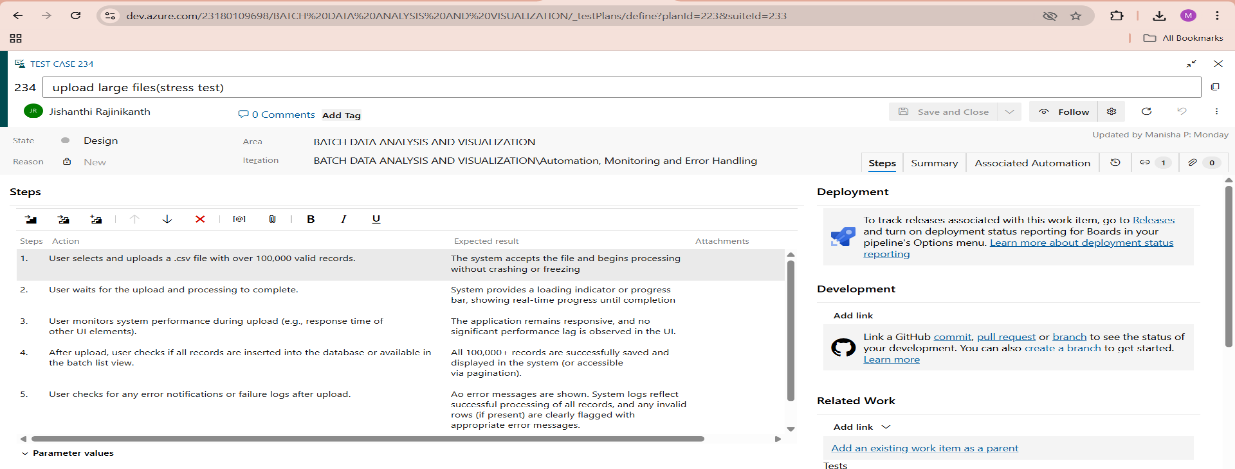
**1. TC09 – Display Upload Error on Server Failure**

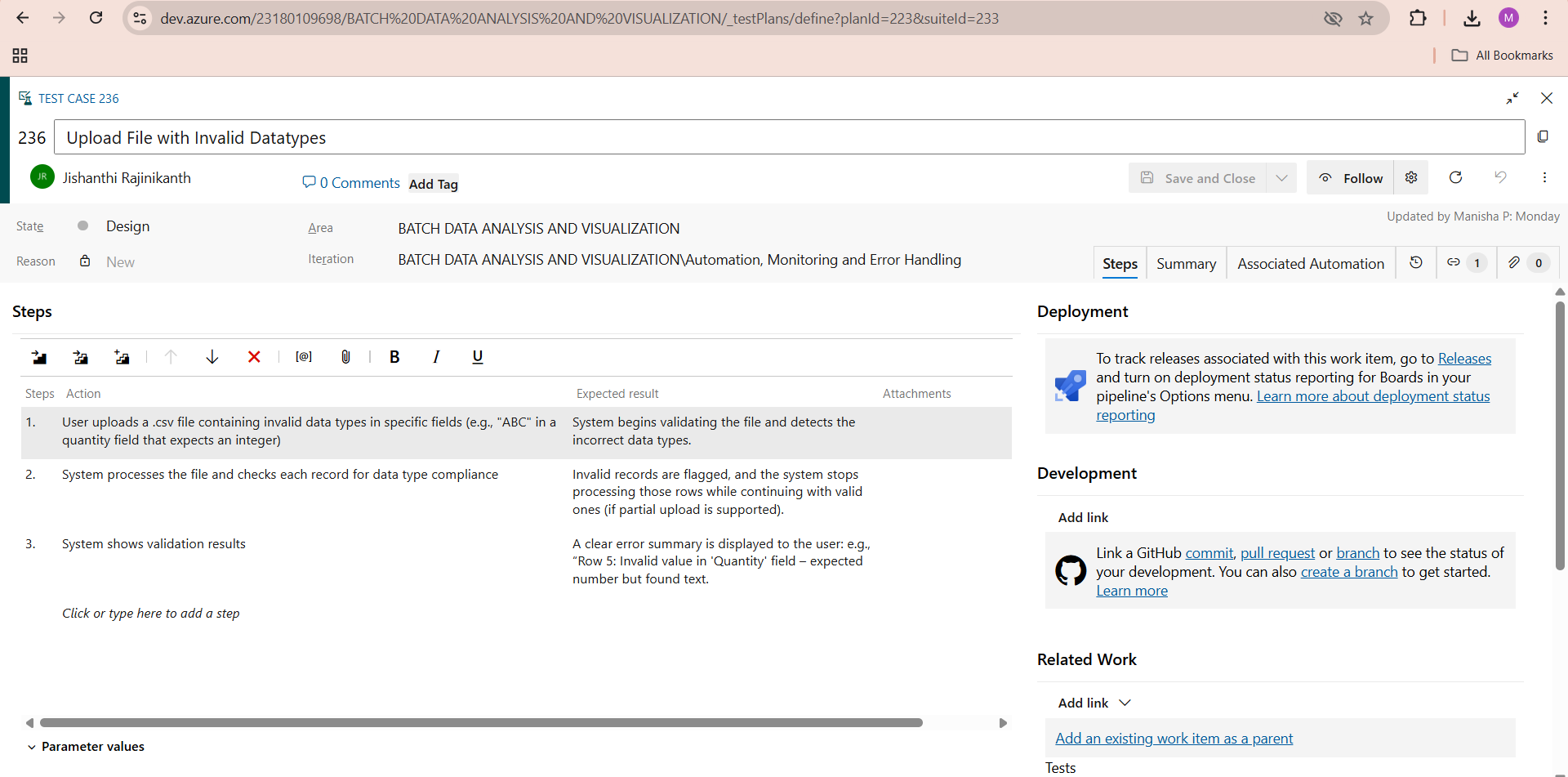
* **Action:**
  + Try uploading when server is down (simulate).
* **Expected Results:**
  + Error message: "Server unavailable. Try again later."
* **Type:** Error Path

**2. TC10 – Missing Mandatory Fields**

* **Action:**
  + Upload a file missing important columns (e.g., 'Name', 'Age').
* **Expected Results:**
  + Error message: "Missing mandatory fields."
* **Type:** Error Path

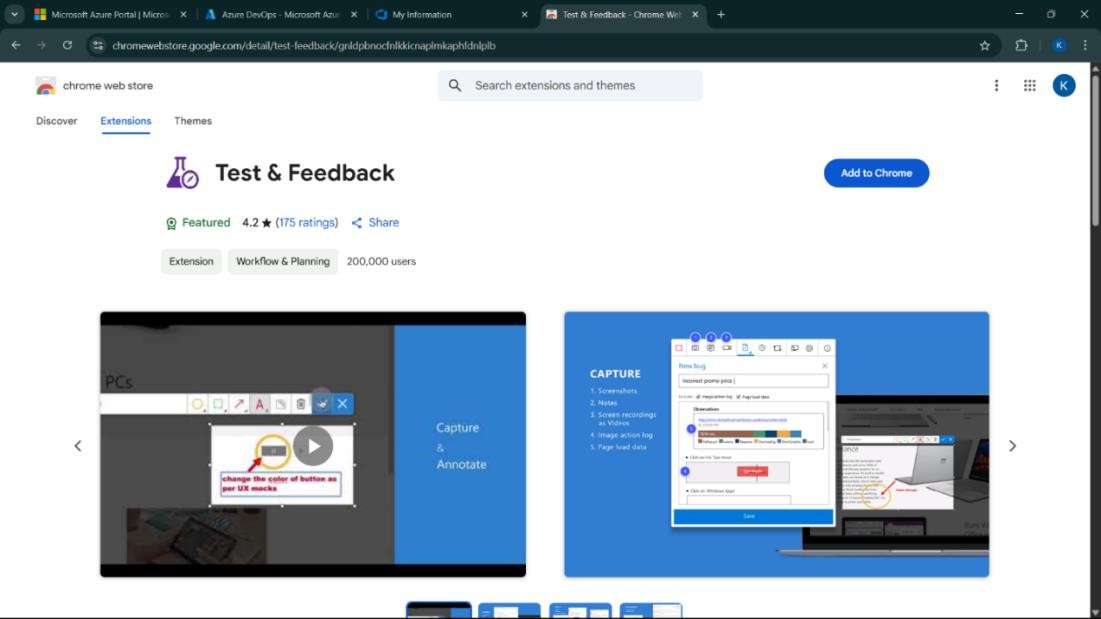
**Test Cases**

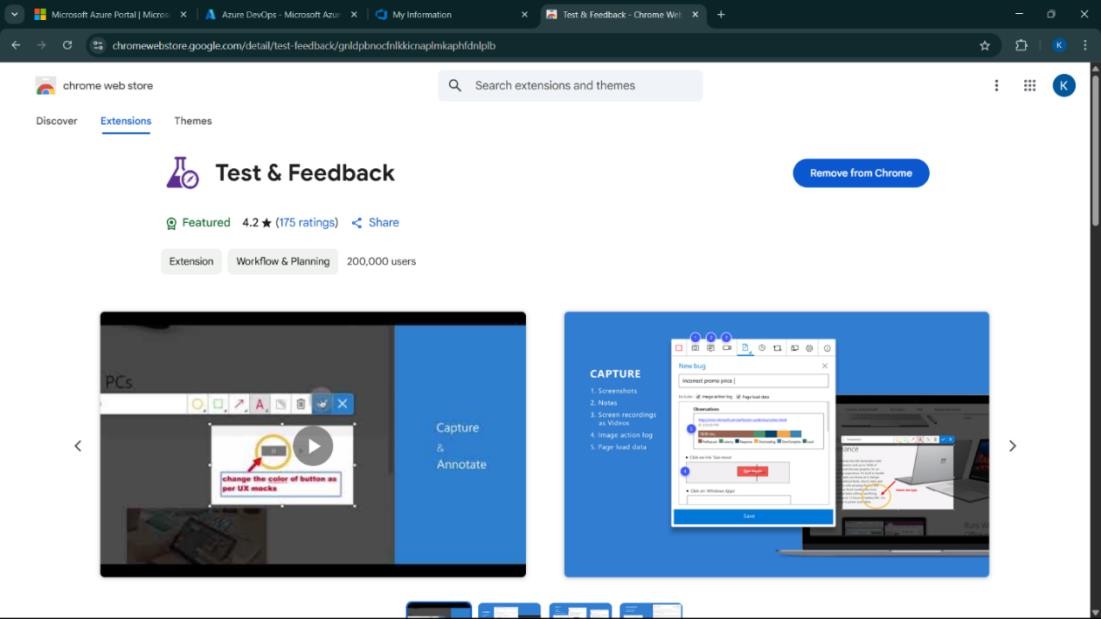
****



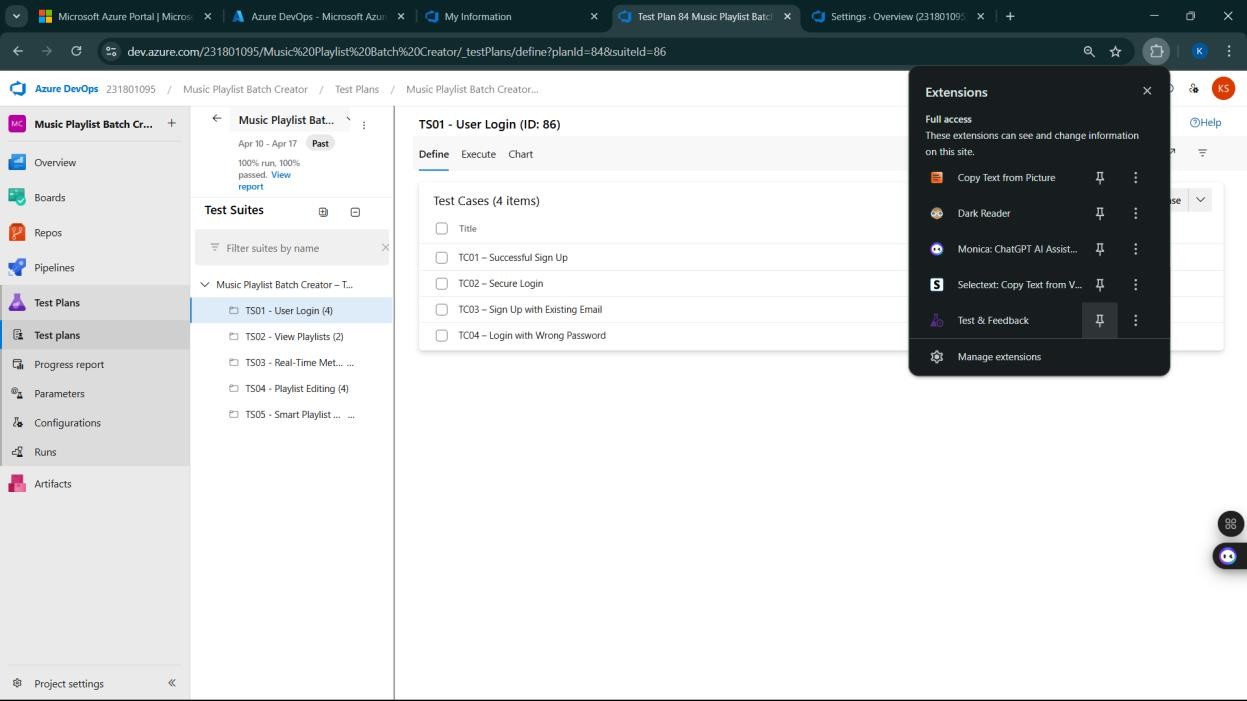
****

**Installation of test**

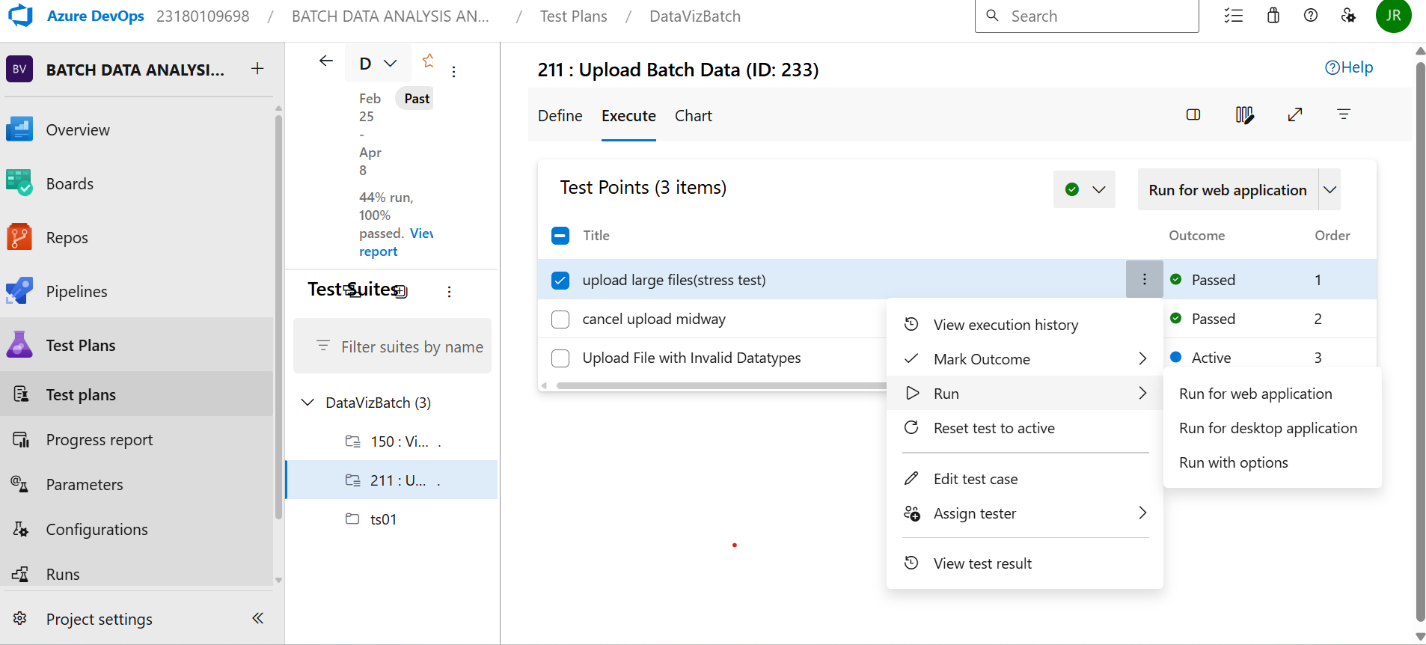
****

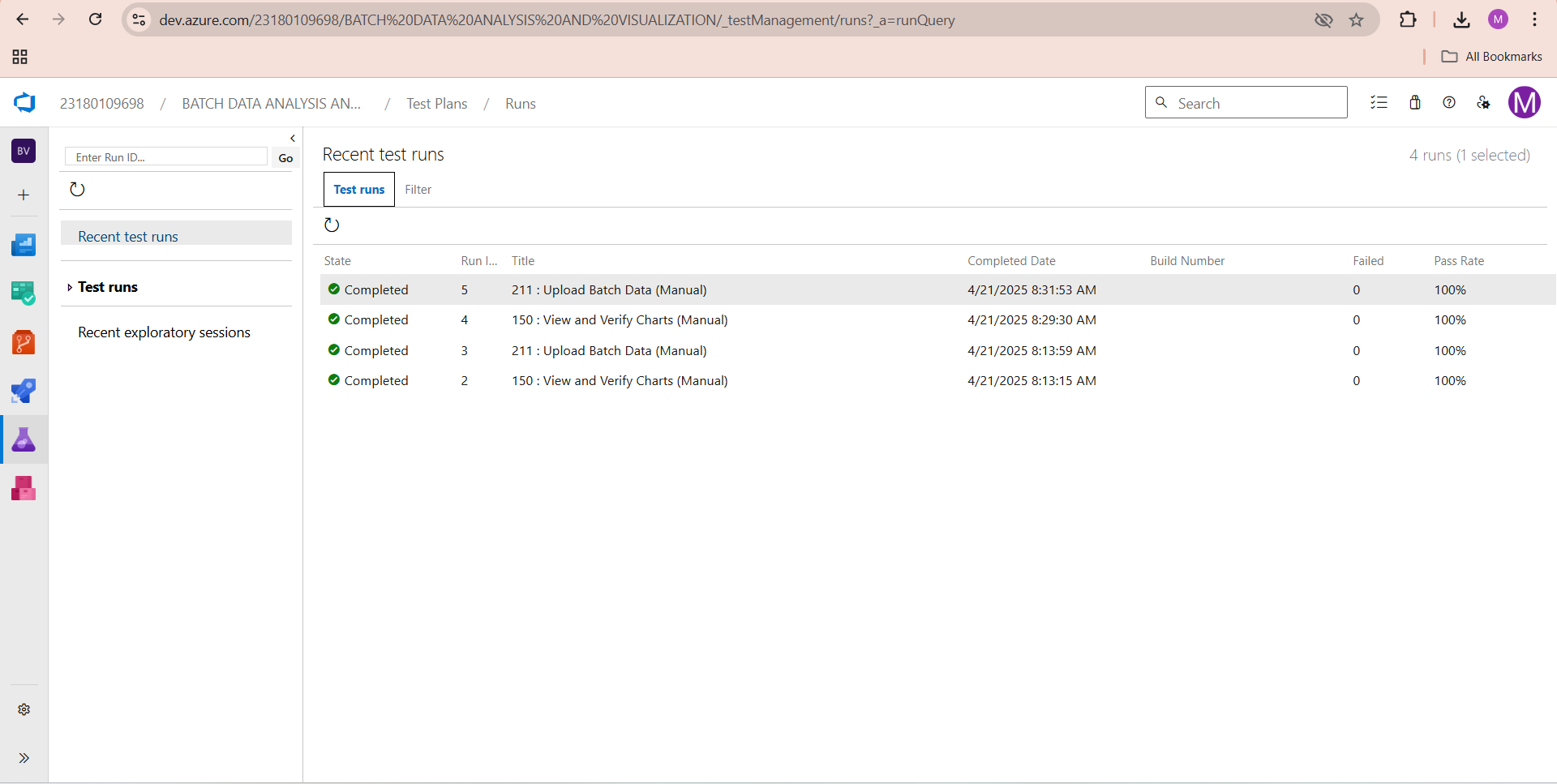


Test and feedback Showing it as an extension

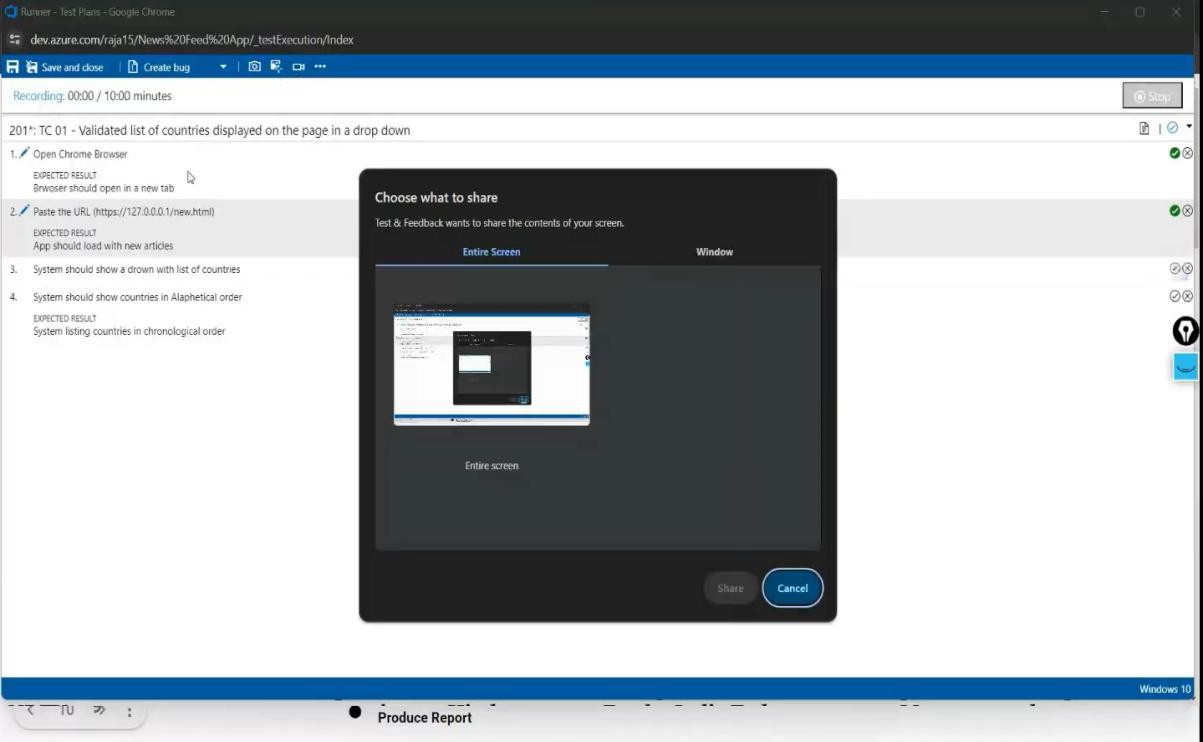


* **Running the test cases**

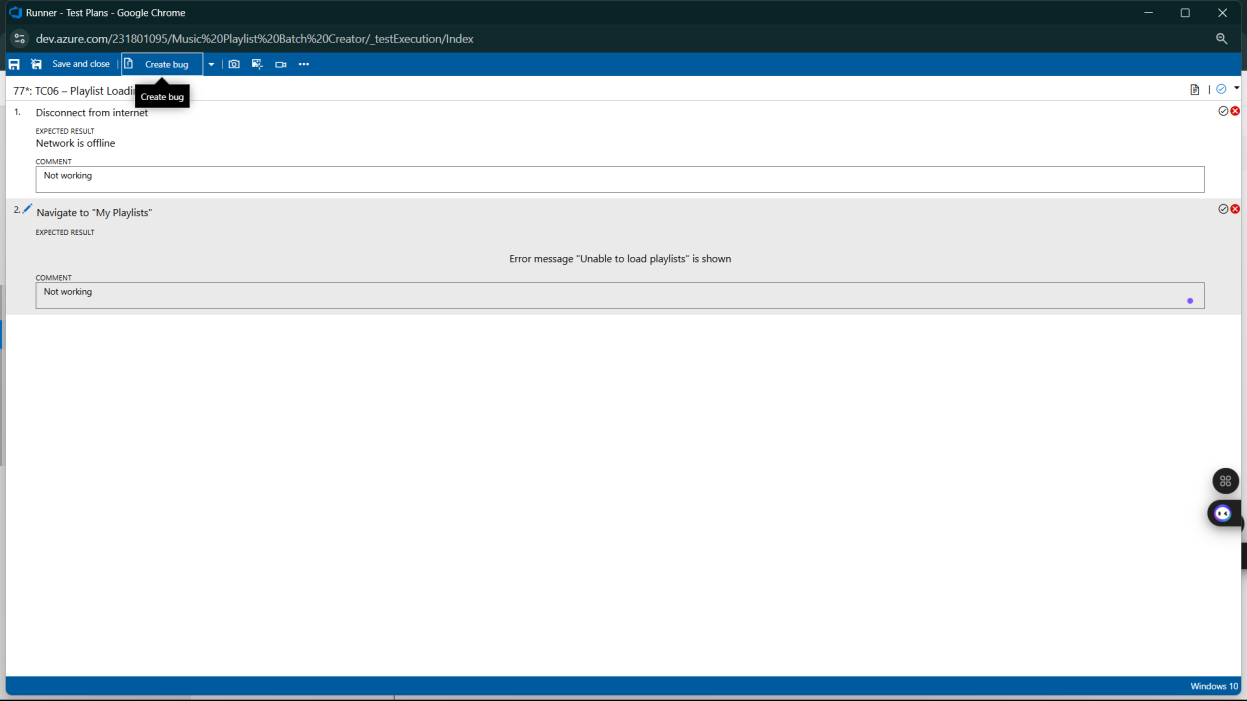
****

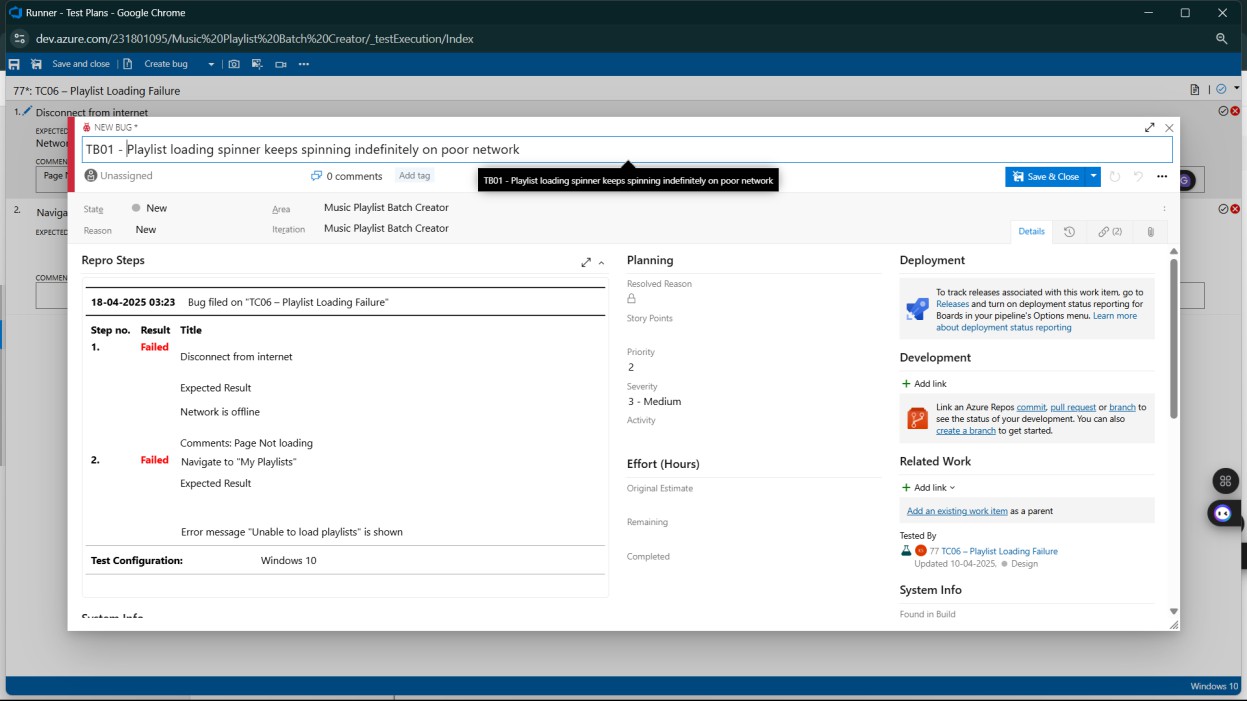
****

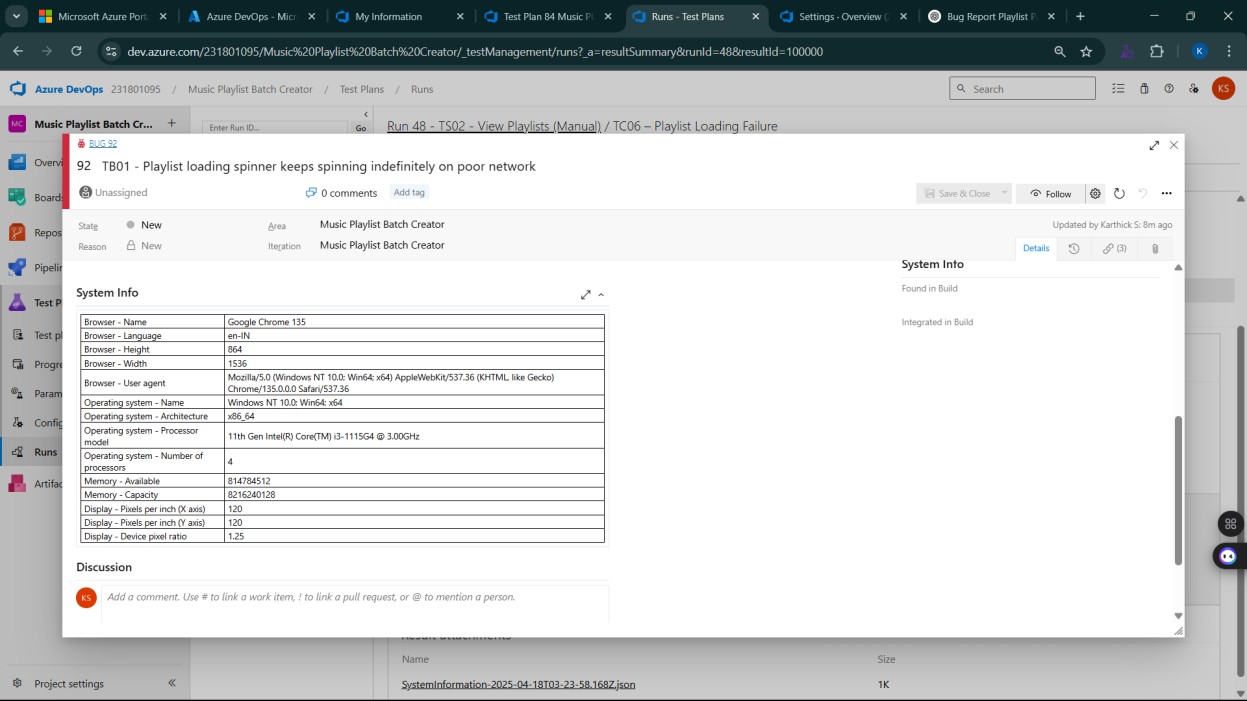
* **Recording the test case**

****

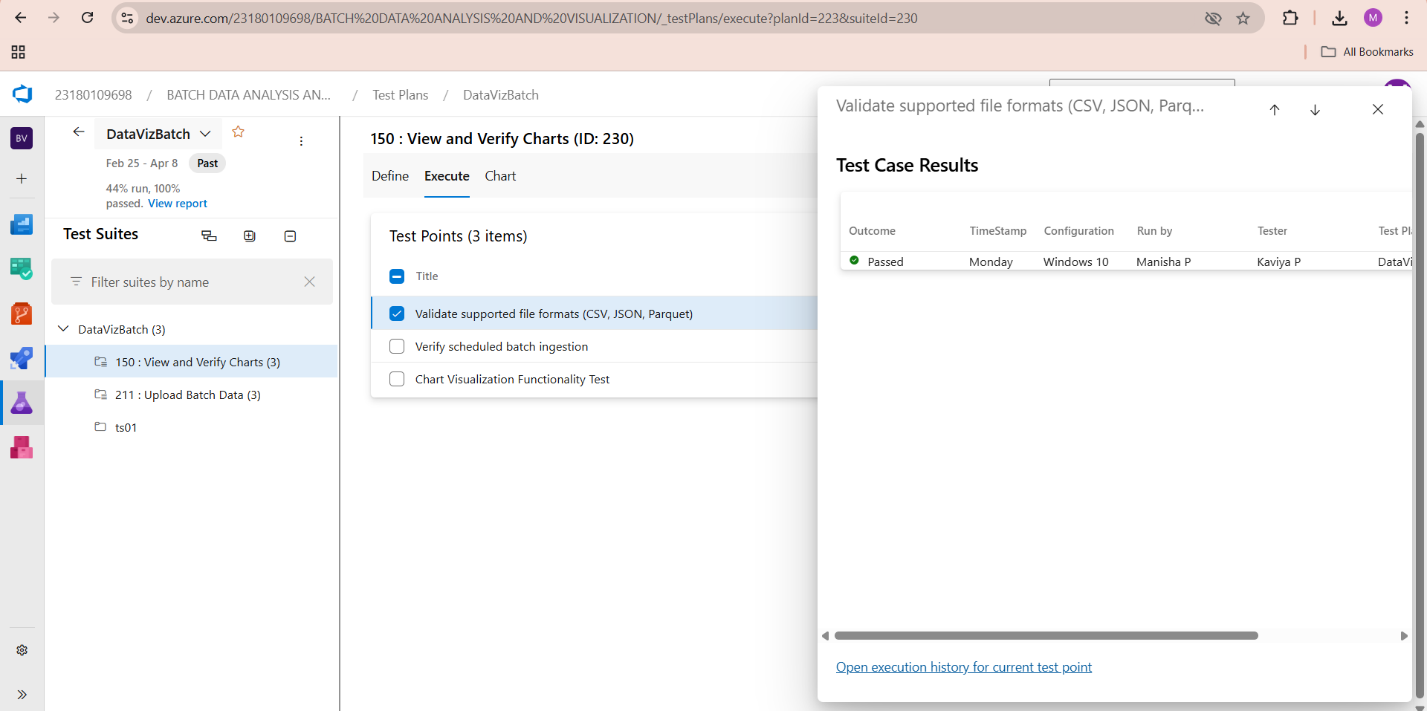
* **Creating the bug**

****

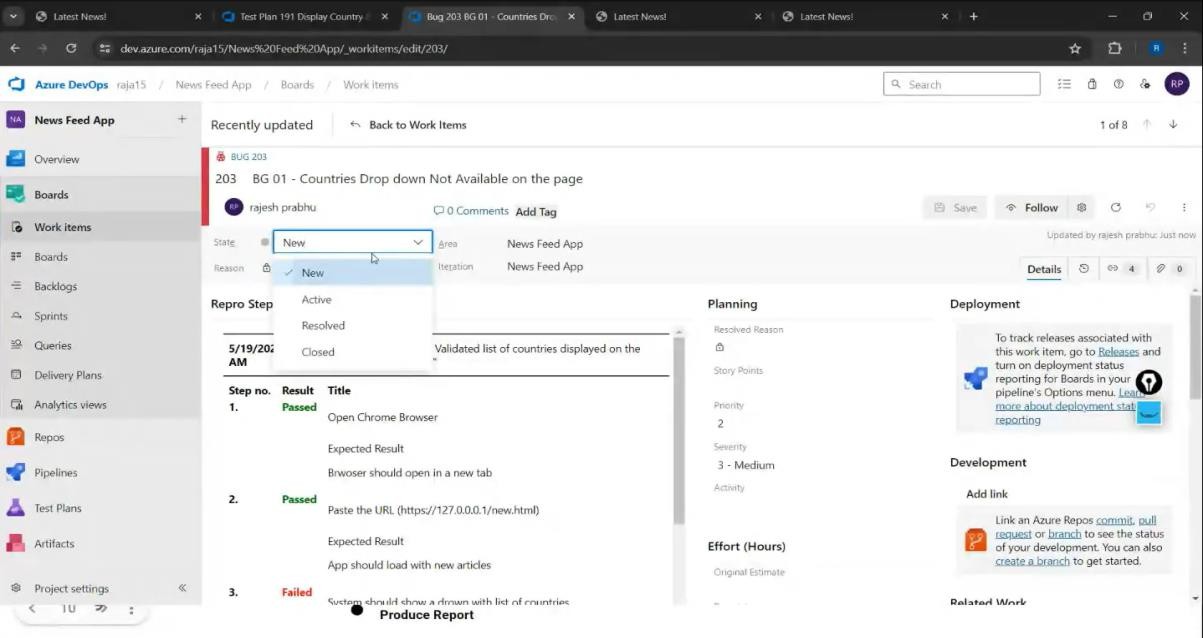


****

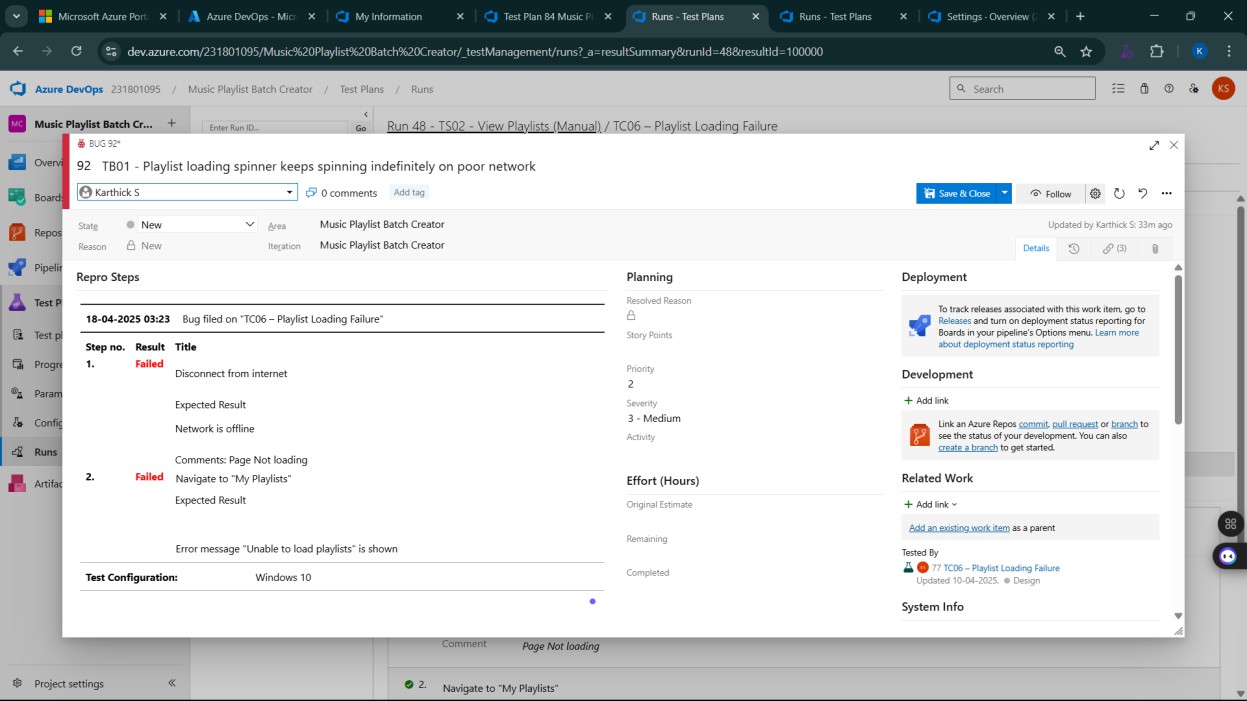
* **Test case results**

****

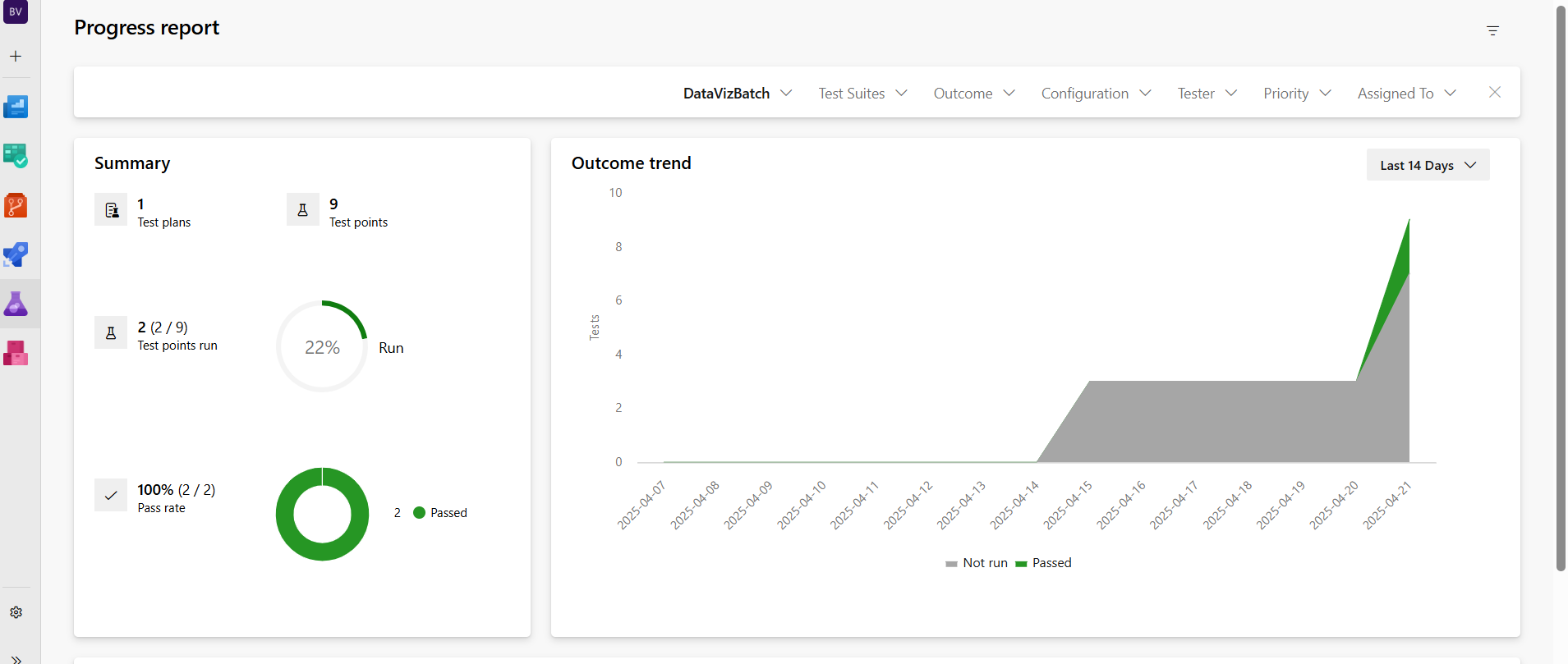
* **Test report summary**

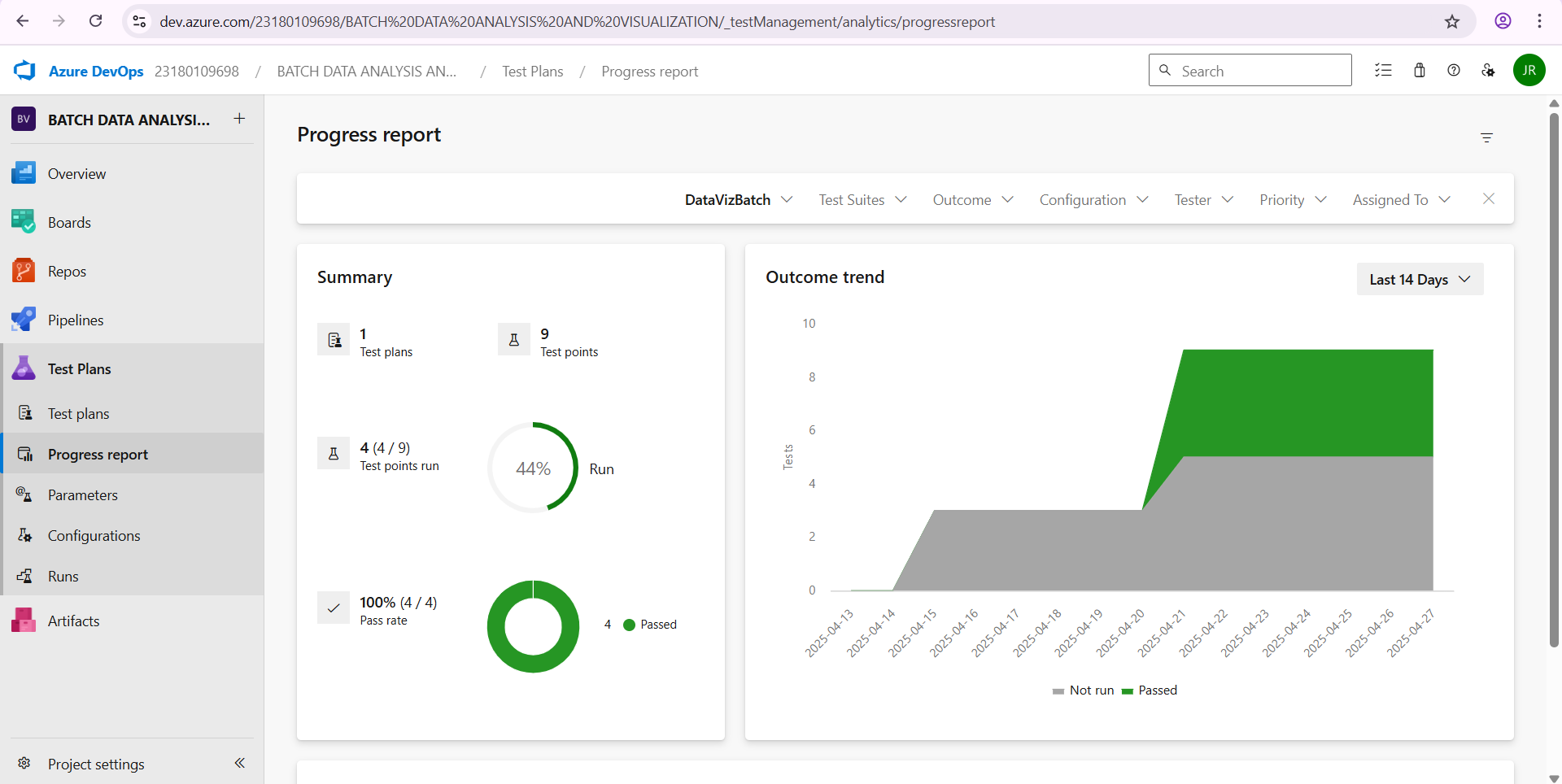
****

* + Assigning bug to the developer and changing state

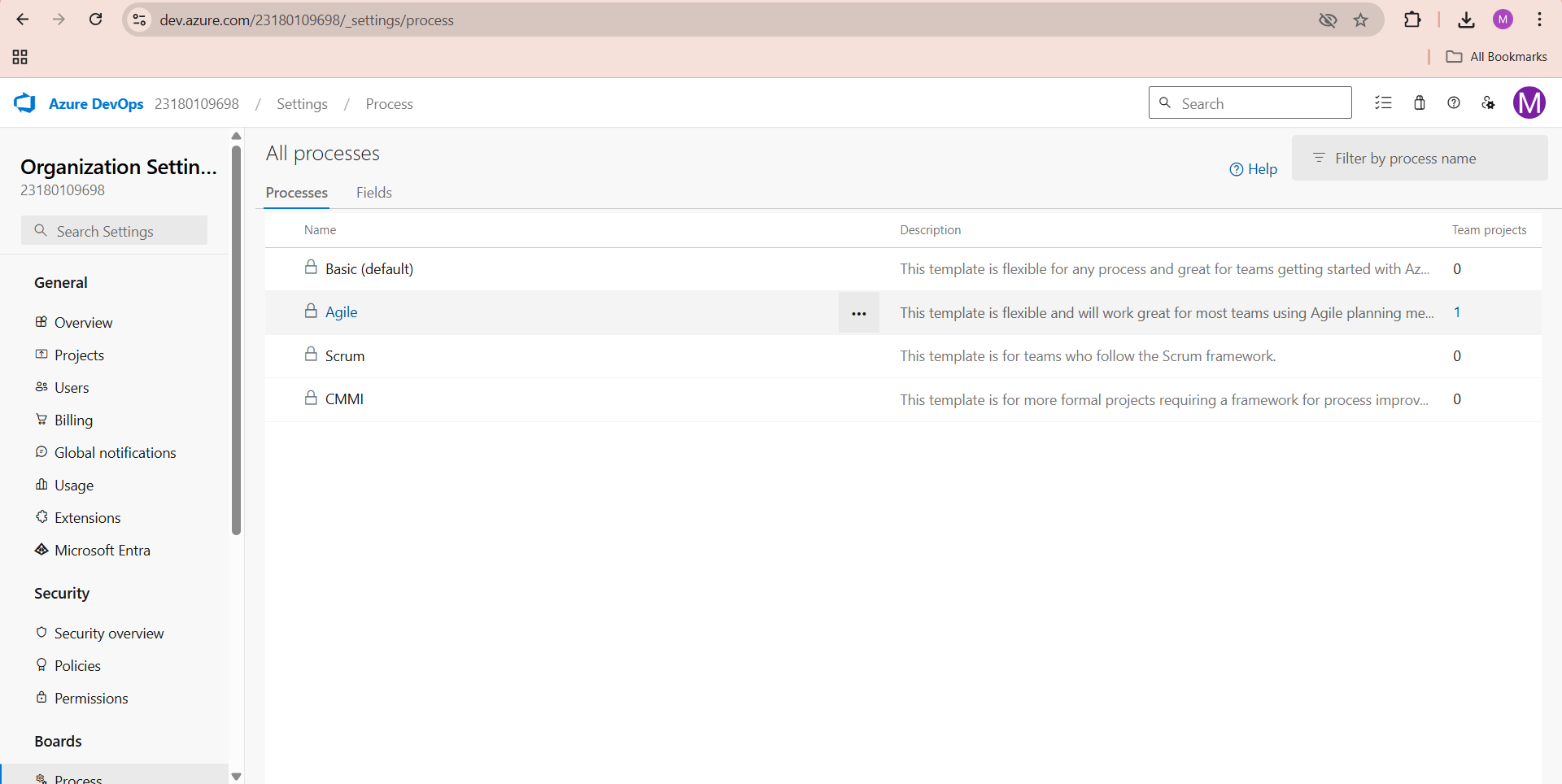


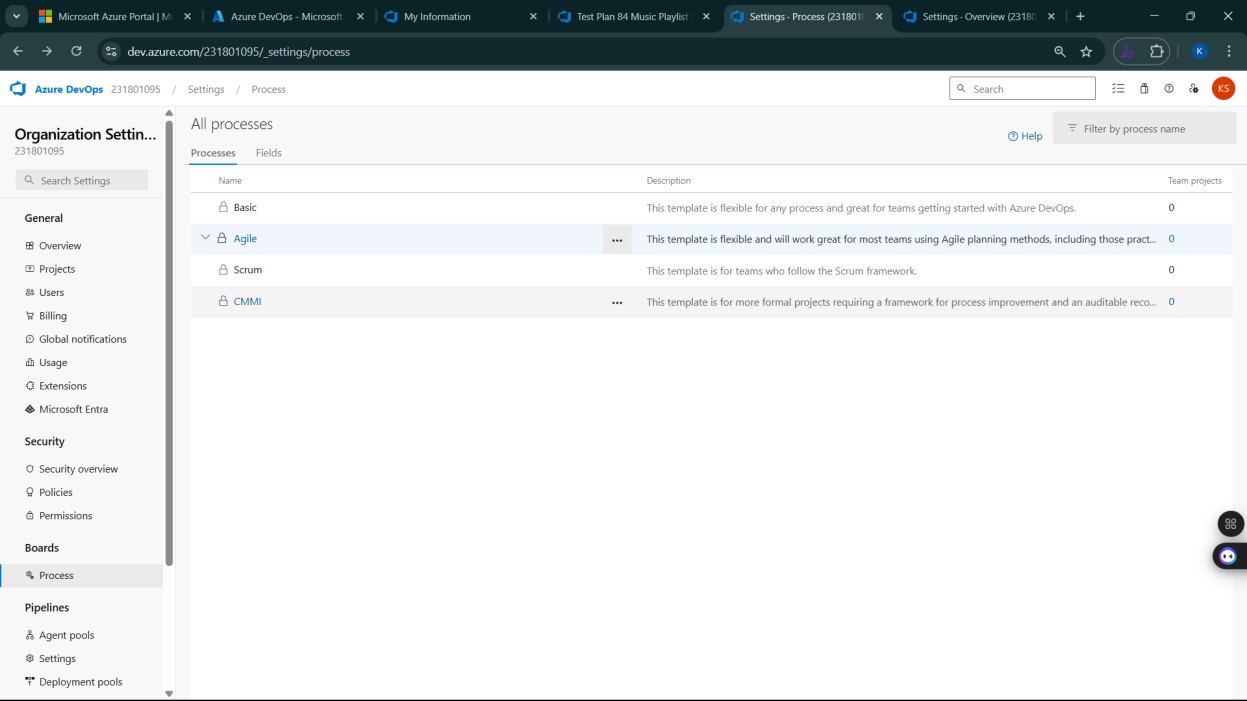
* **Progress report**

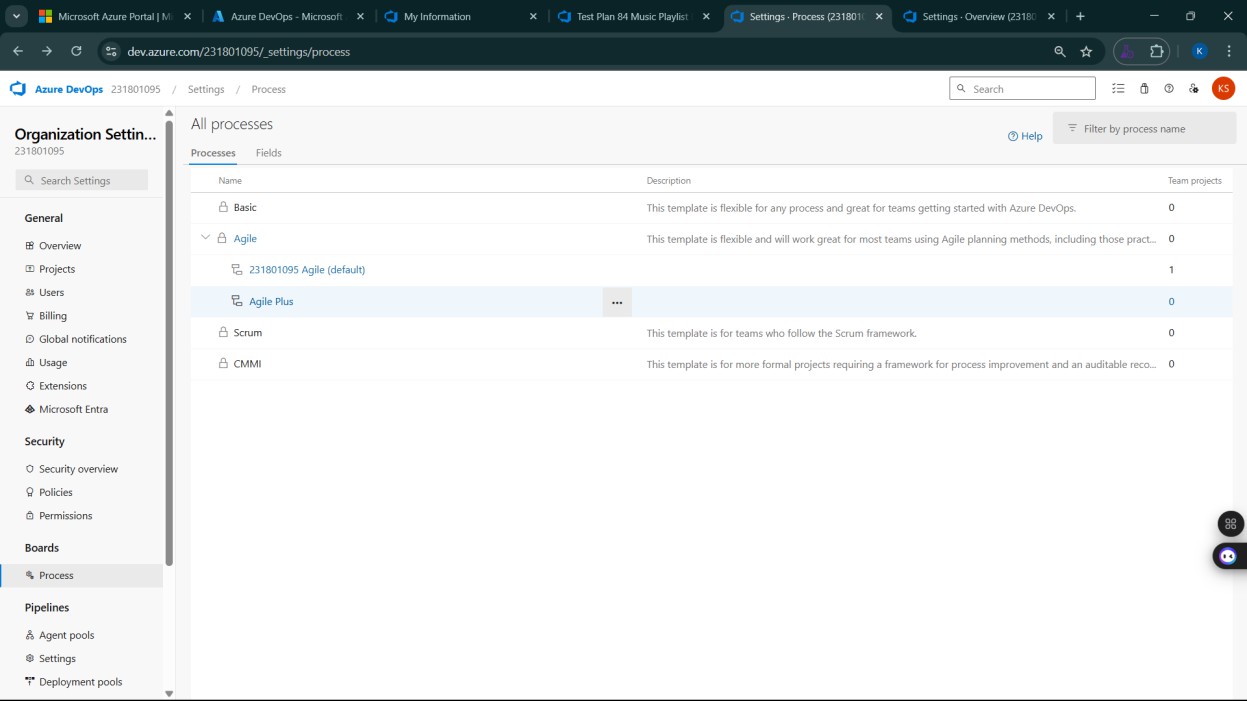
****



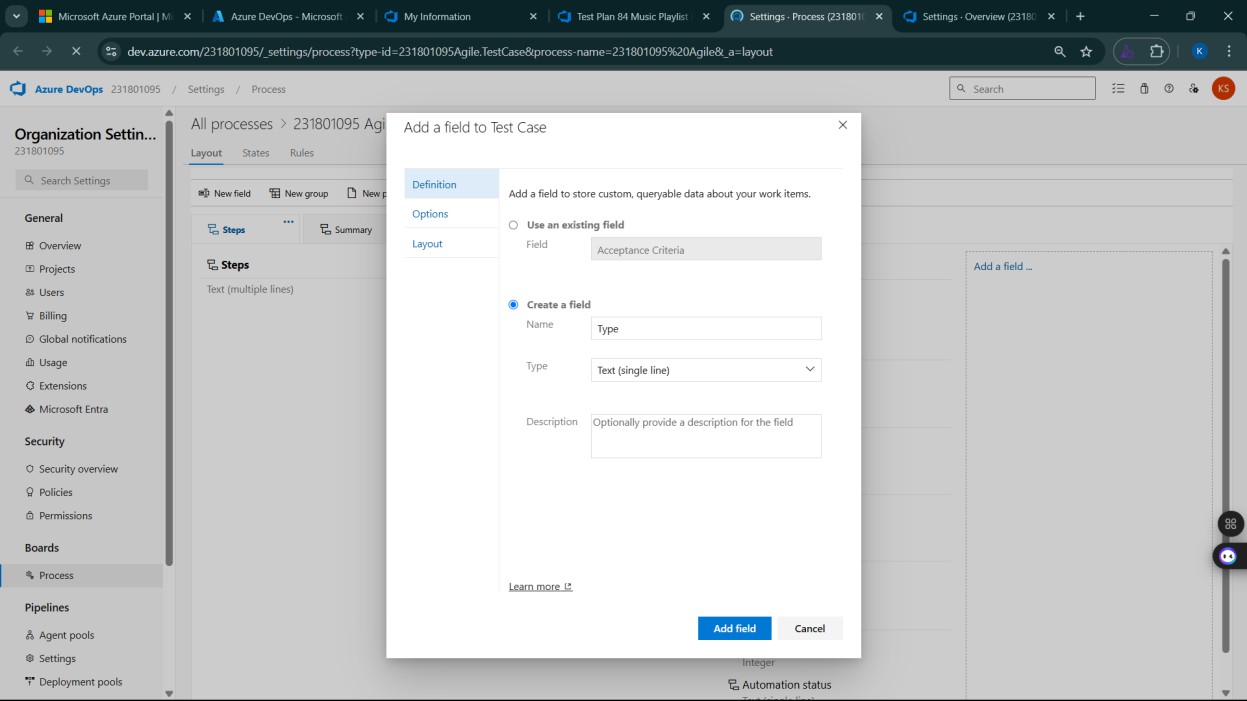
* **Changing the test template**

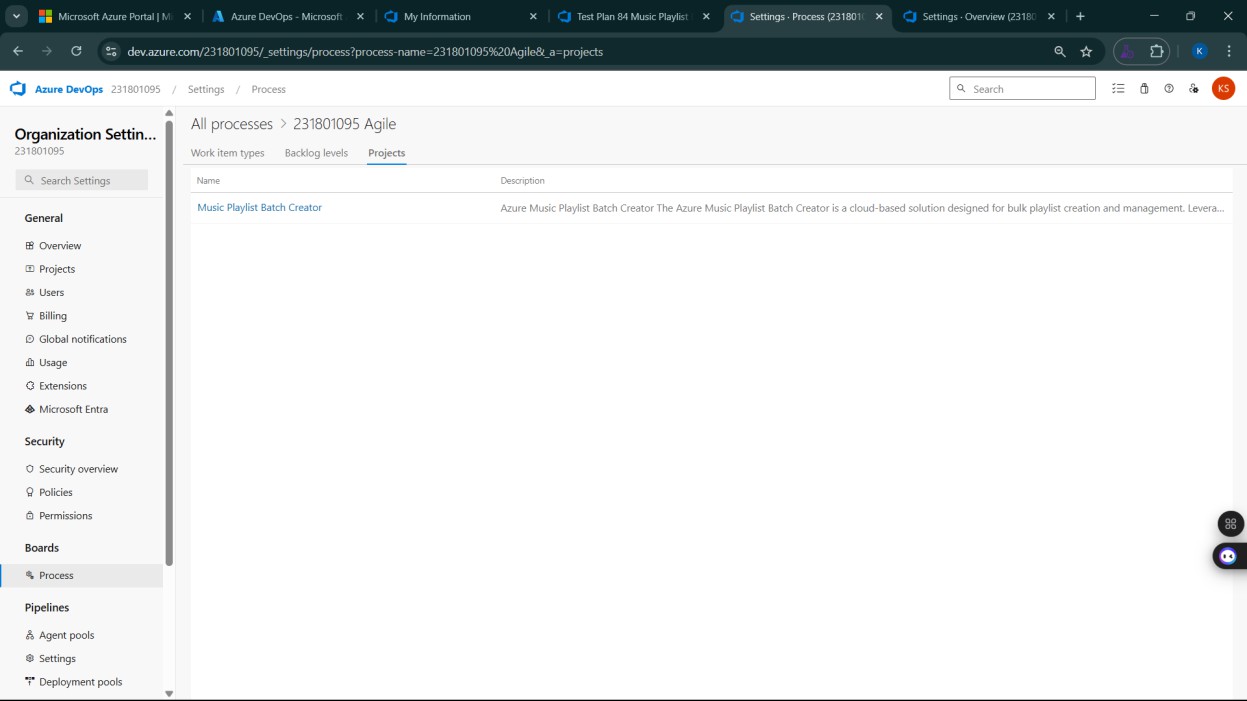
****

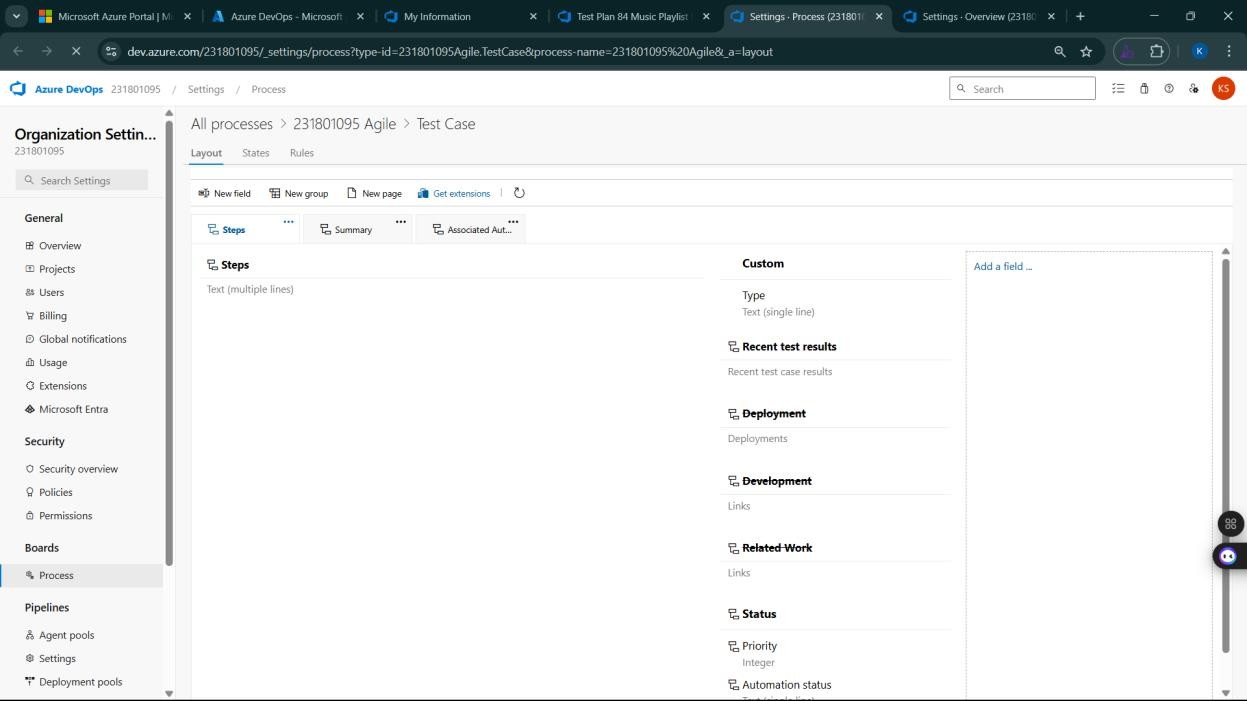


****

* **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

**LOAD TESTING AND PIPELINES**

**EXP NO: 9**

**Aim:**

Create Epic, Features, User Stories, Task

To create an Azure Load Testing resource and run a load test to evaluate the performance of a target endpoint.

**Load Testing**

**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](https://portal.azure.com/) and log in.

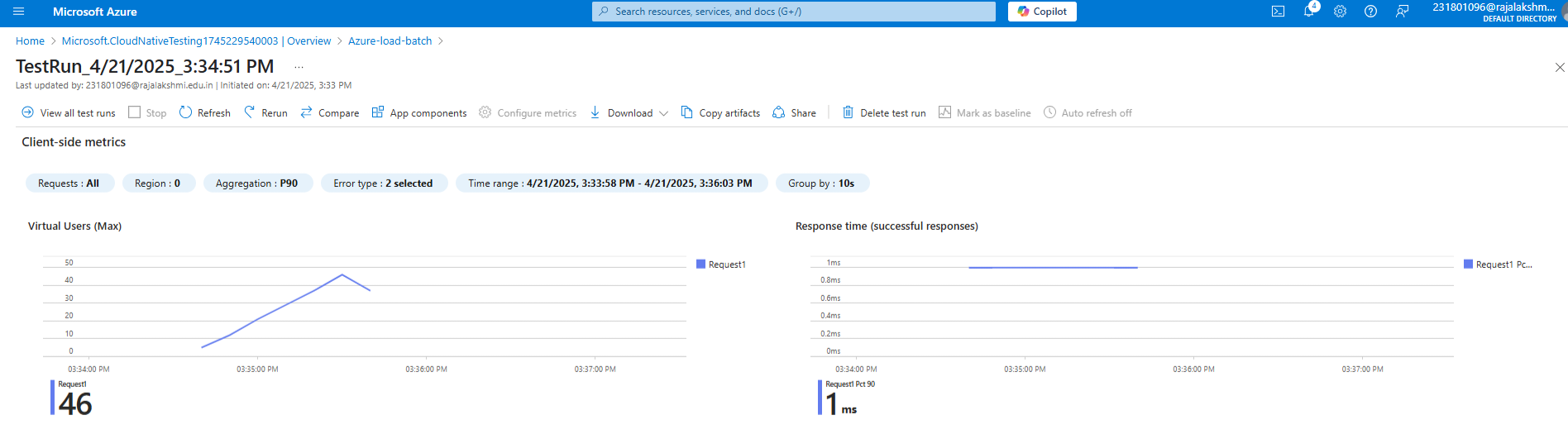
1. Create the Resource
   * Go to *Create a resource* → Search for “Azure Load Testing”.
   * Select Azure Load Testing and click Create.
2. Fill in the Configuration Details
   * Subscriptio*n:* Choose your Azure subscription.
   * *Resource Group:* Create new or select an existing one.
   * *Name:*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

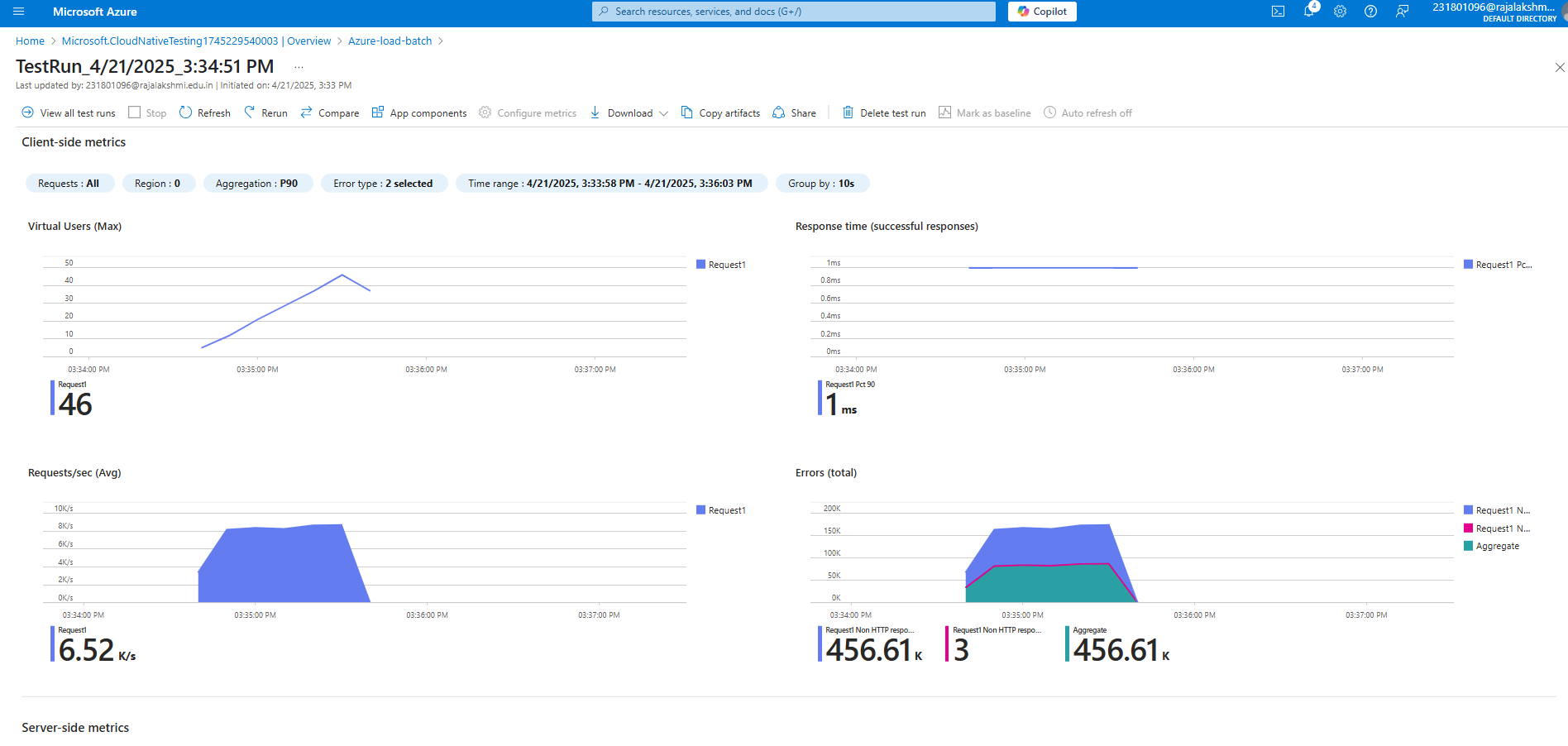
**Steps to Create and Run a Load Test:**

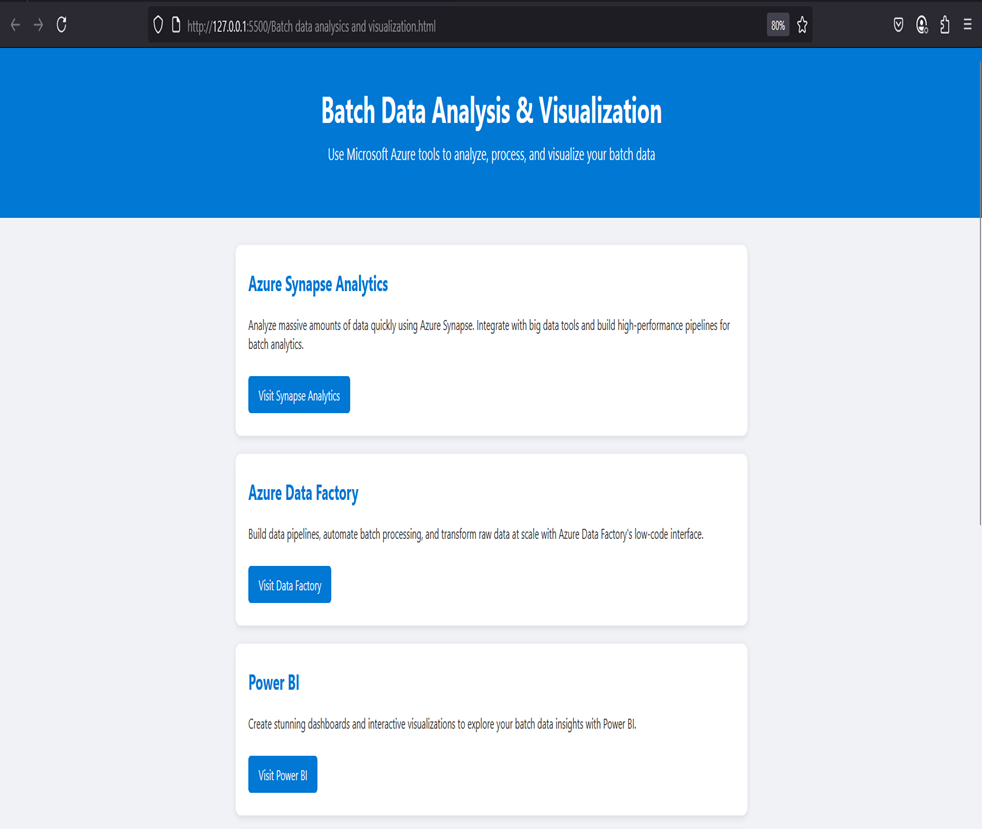
Once your resource is ready:

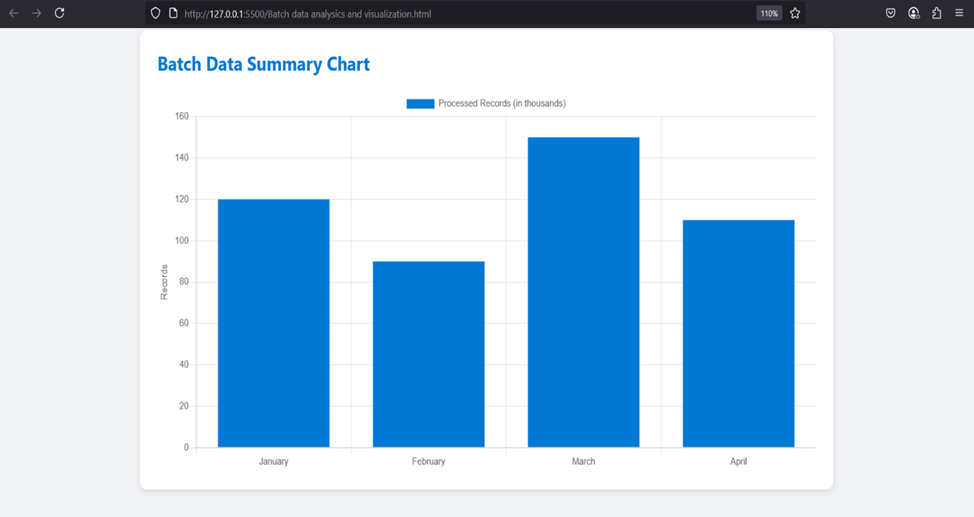
1. Go to your Azure Load Testing resource and click Add HTTP requests > Create.
2. Basics Tab
   * *Test Name:* Provide a unique name.
   * *Description:* (Optional) Add test purpose.
   * *Run After Creation:* Keep checked.
3. Load Settings
   * *Test URL:* Enter the target endpoint (e.g., https://yourapi.com/products).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

o In Azure DevOps, create a new project.

o Create a pipeline and select GitHub as the source.

o 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:

o In your GitHub repository, create a new file called azure-pipelines.yml in the root

directory.

o 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"

3. Pipeline Tasks Include:

o Setting up the Python environment using the UsePythonVersion task.

o Installing project dependencies from project/requirements.txt. Make sure the path to

requirements.txt is correct (it is located under the project folder).

o Running a simple Python script to verify that Python is set up correctly and the pipeline

works.

4. Run and Monitor Pipeline:

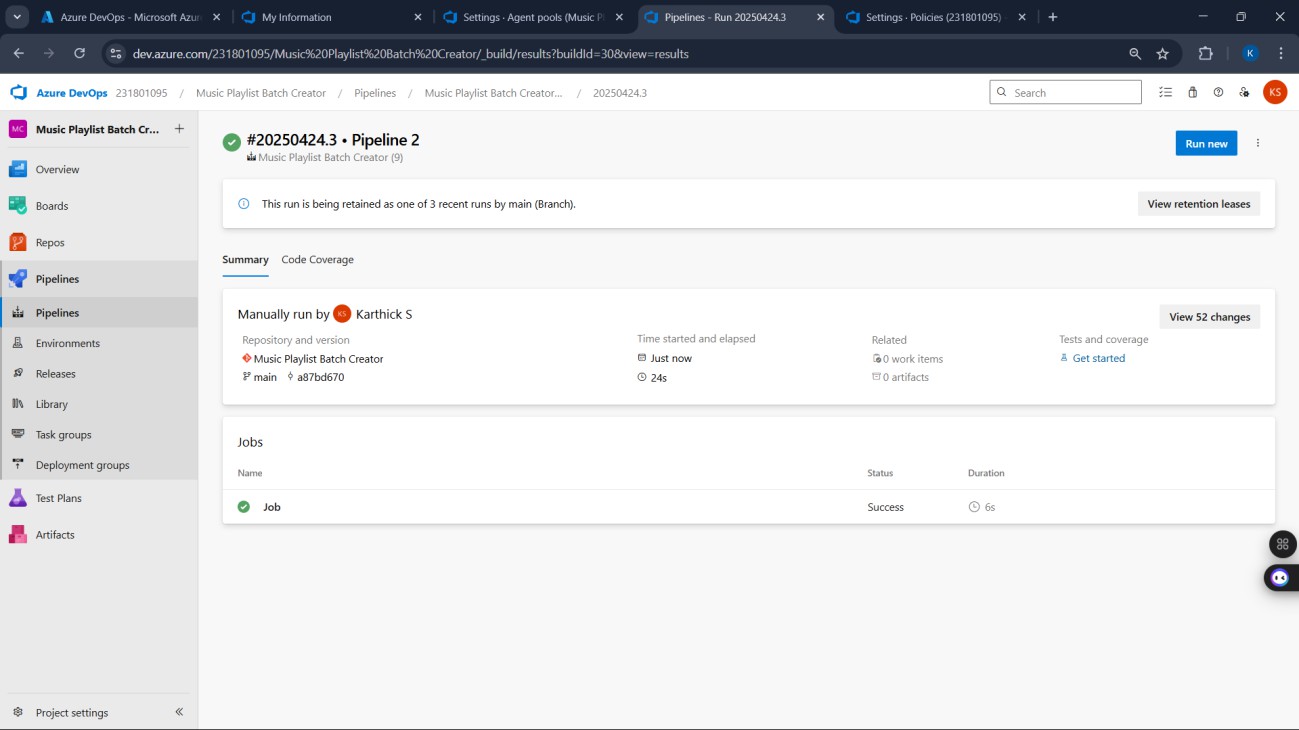
o Commit changes to the main branch of your repository to trigger the pipeline in Azure

DevOps.

o 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.

**GITHUB: PROJECT STRUCTURE & NAMING CONVENTIONS**

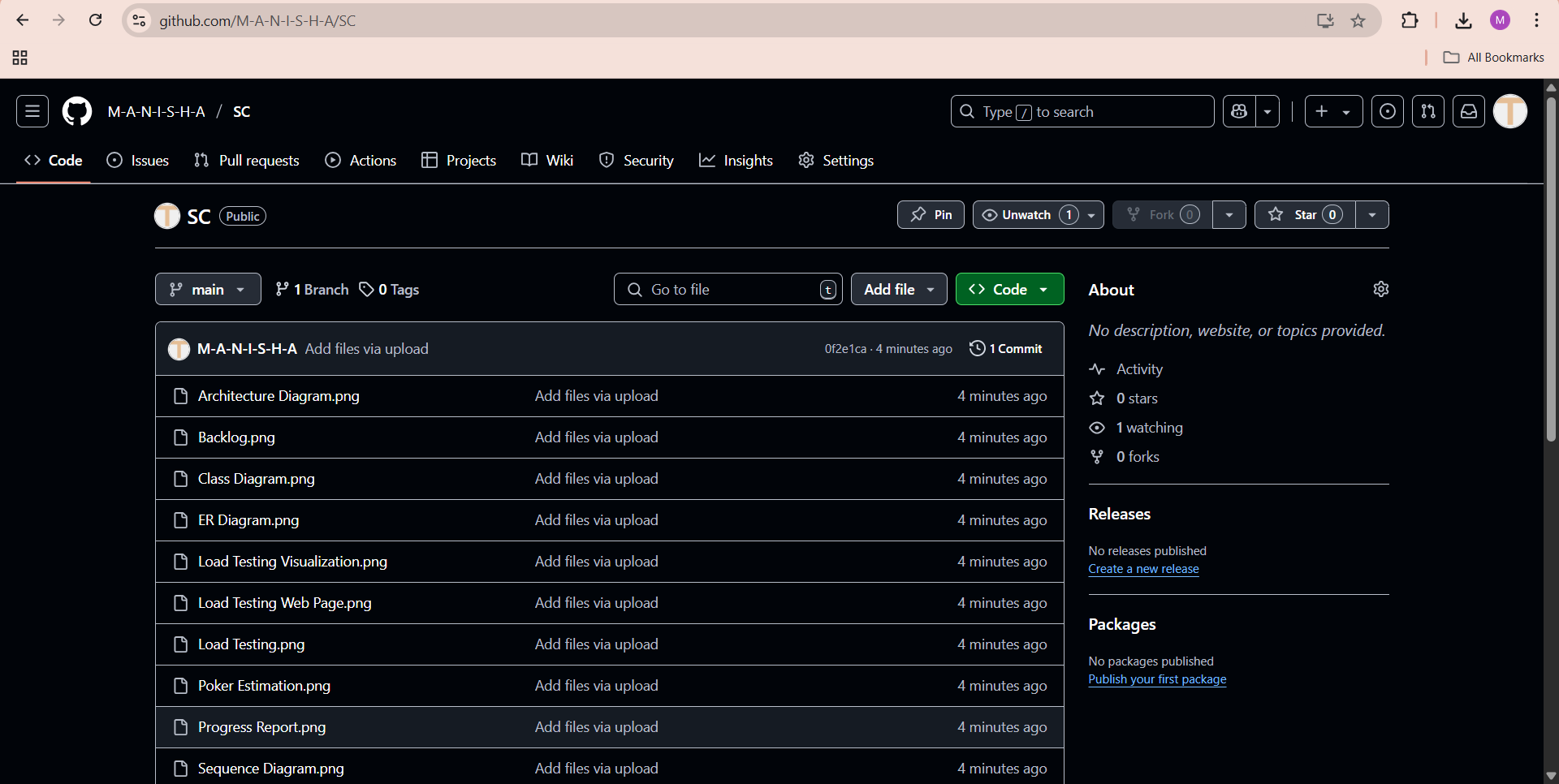
**EXP NO: 10**

**Aim:**

Create Epic, Features, User Stories, Task

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 Batch Data Analysis And Visualization.

**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.