**AZURE DEVOPS ENVIRONMENT SETUP**

**EXP NO: 1**

**Aim:**

Create Epic, Features, User Stories, Task

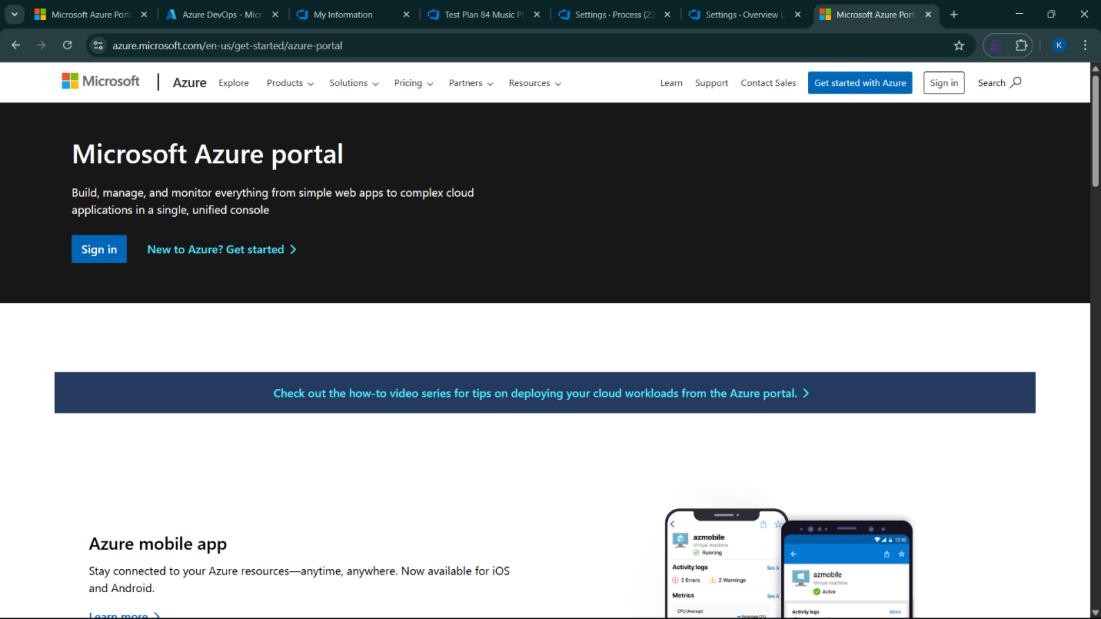
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

**INSTALLATION**

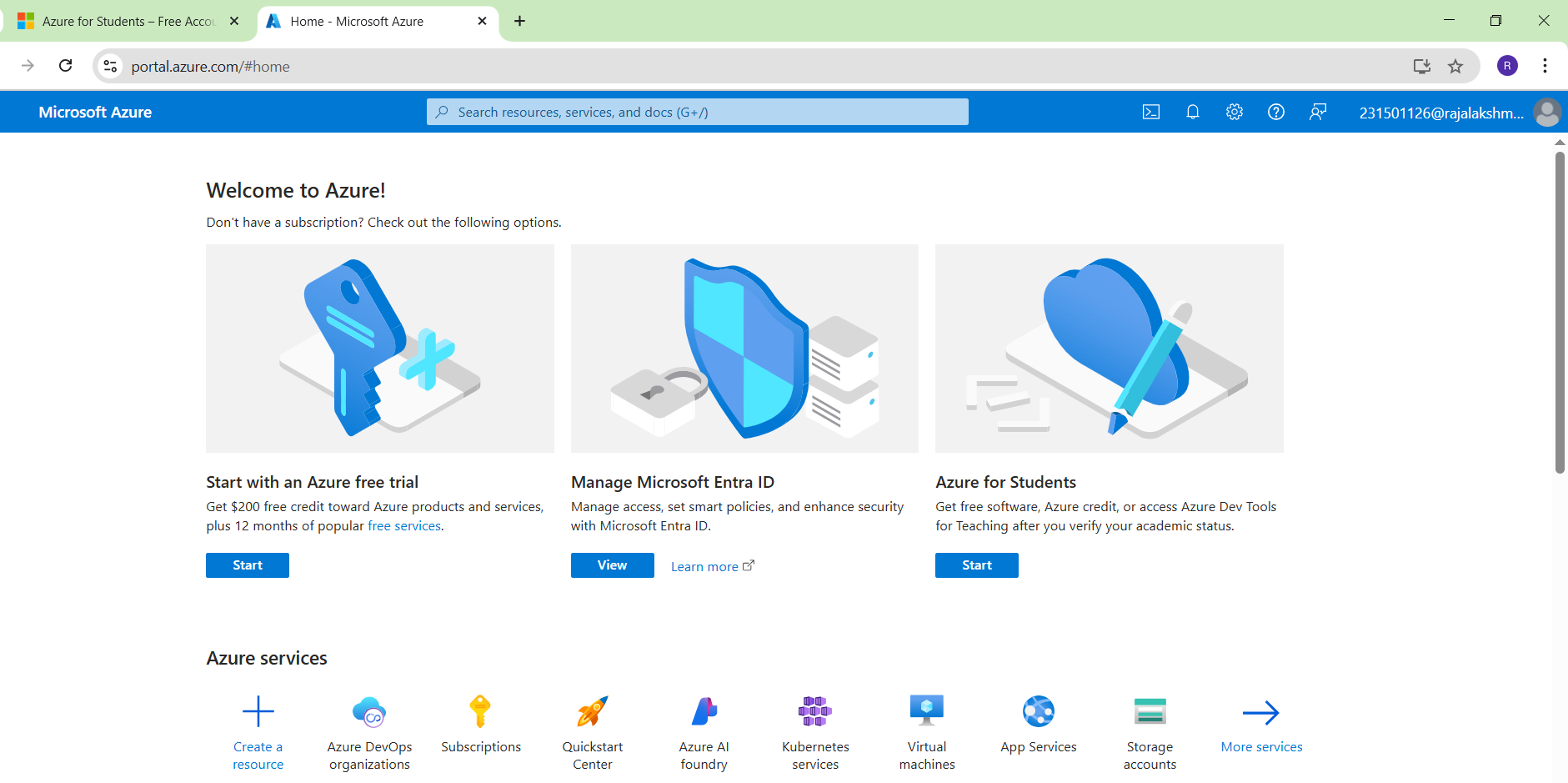
1. Open your web browser and go to the Azure website: [https://azure.microsoft.com/en-us/get-](https://azure.microsoft.com/en-us/get-started/azure-portal) [started/azure-portal](https://azure.microsoft.com/en-us/get-started/azure-portal).

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>



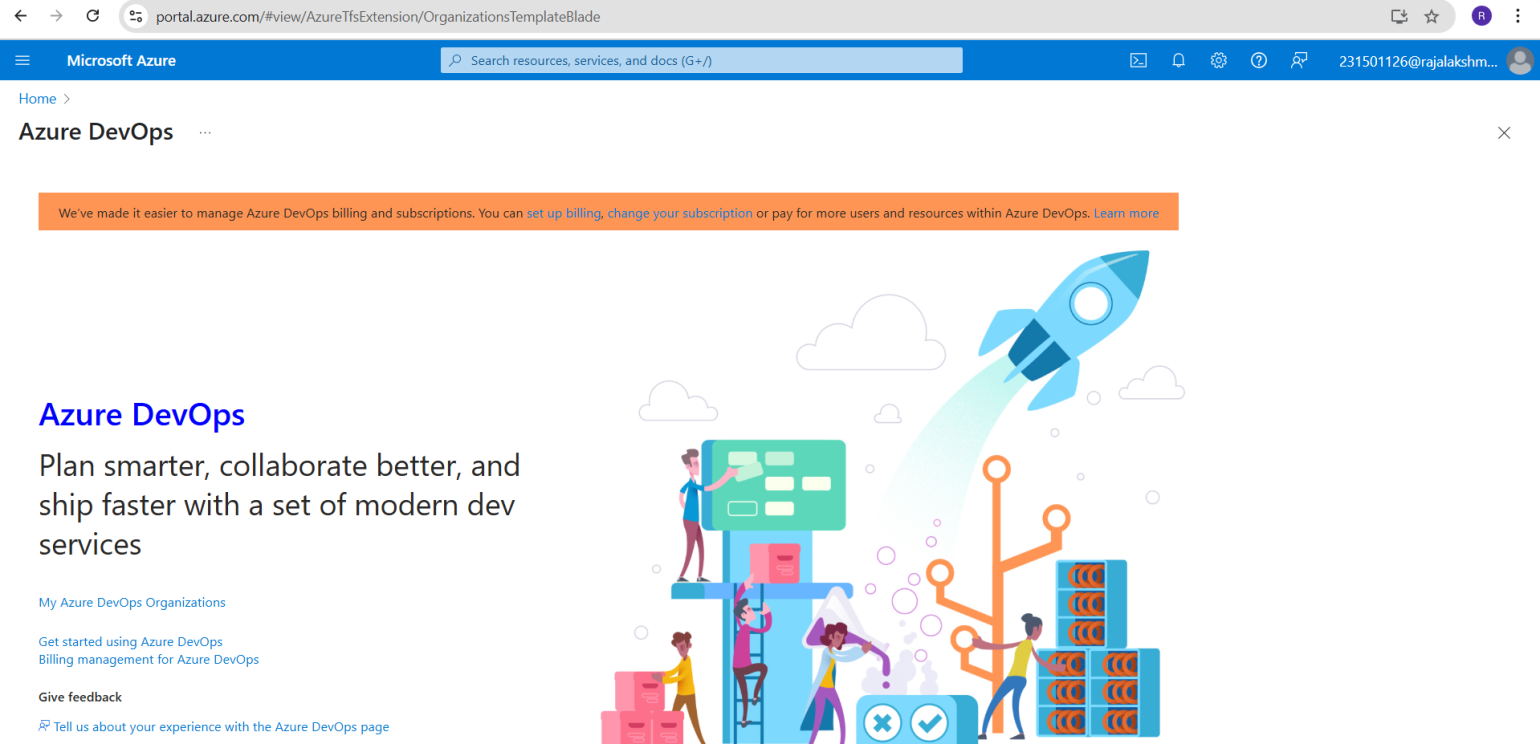
1. Azure home page



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**

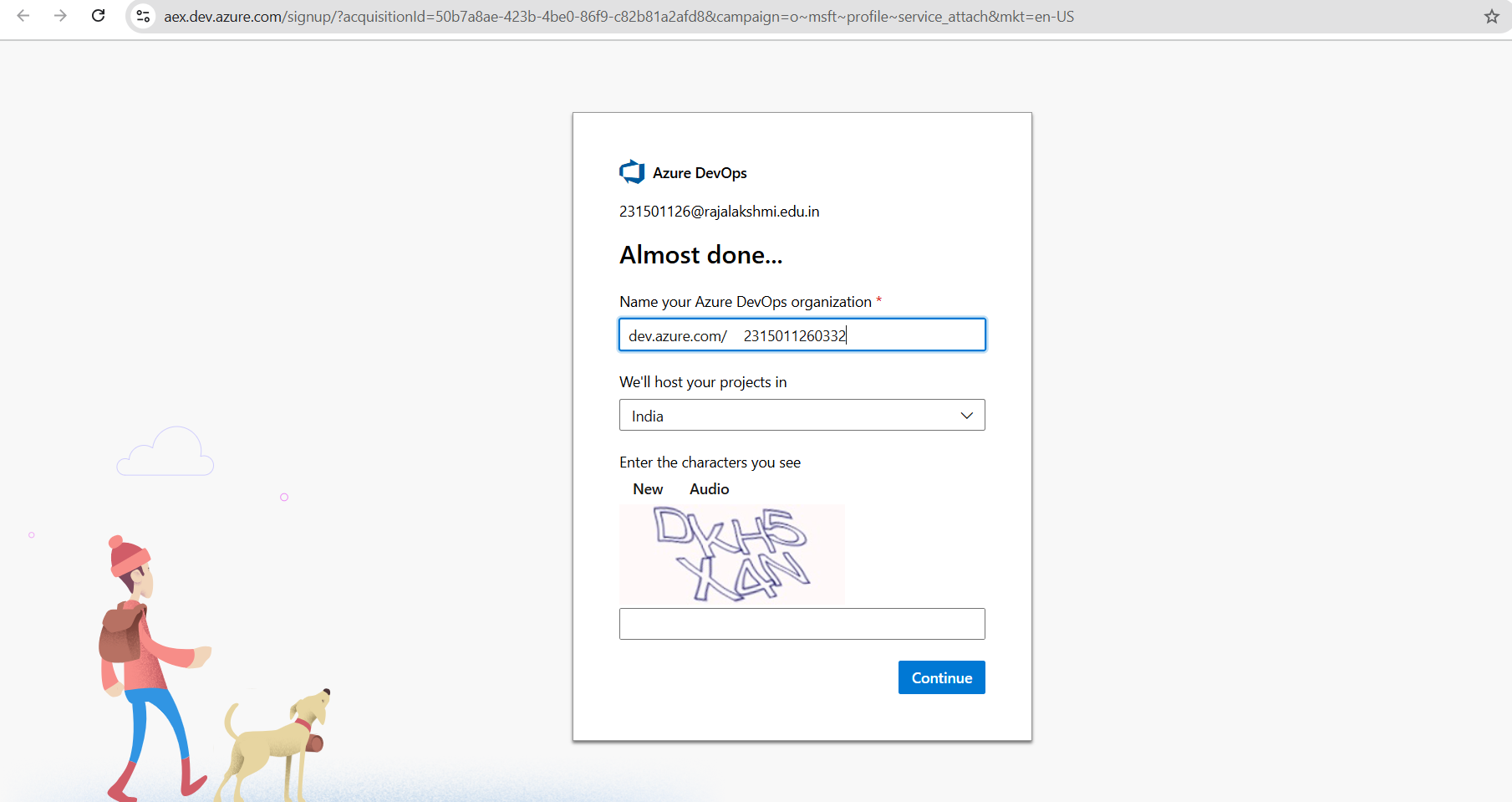
**EXP NO: 2**

**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

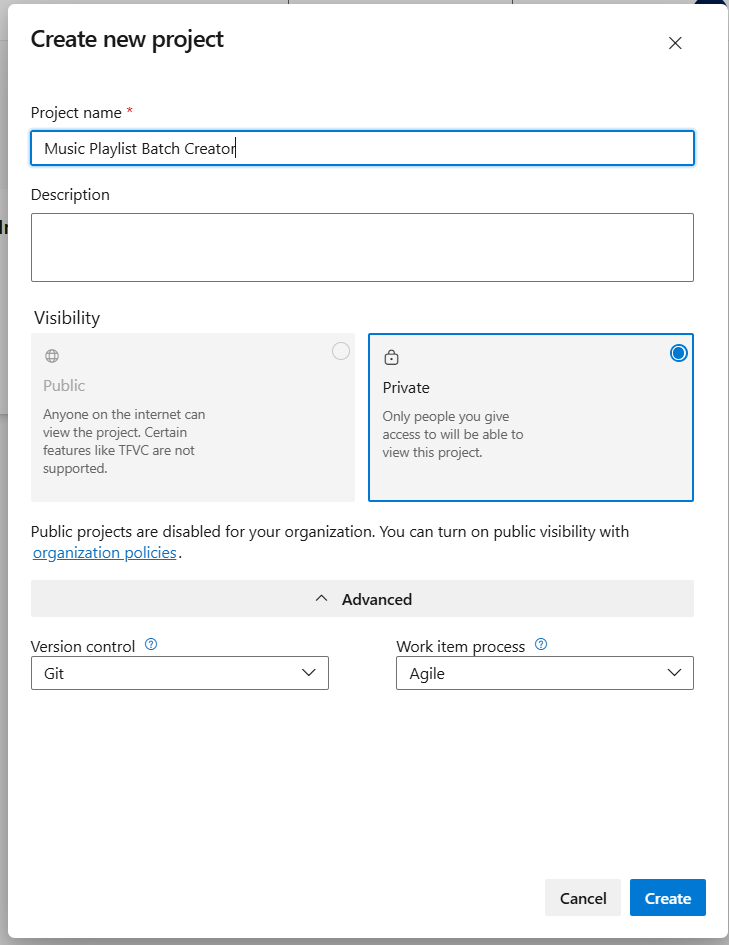


1. Create the First Project in Your Organization
   1. After the organization is set up, you’ll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.
   2. On the organization’s **Home page**, click on the **New Project** button.
   3. Enter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

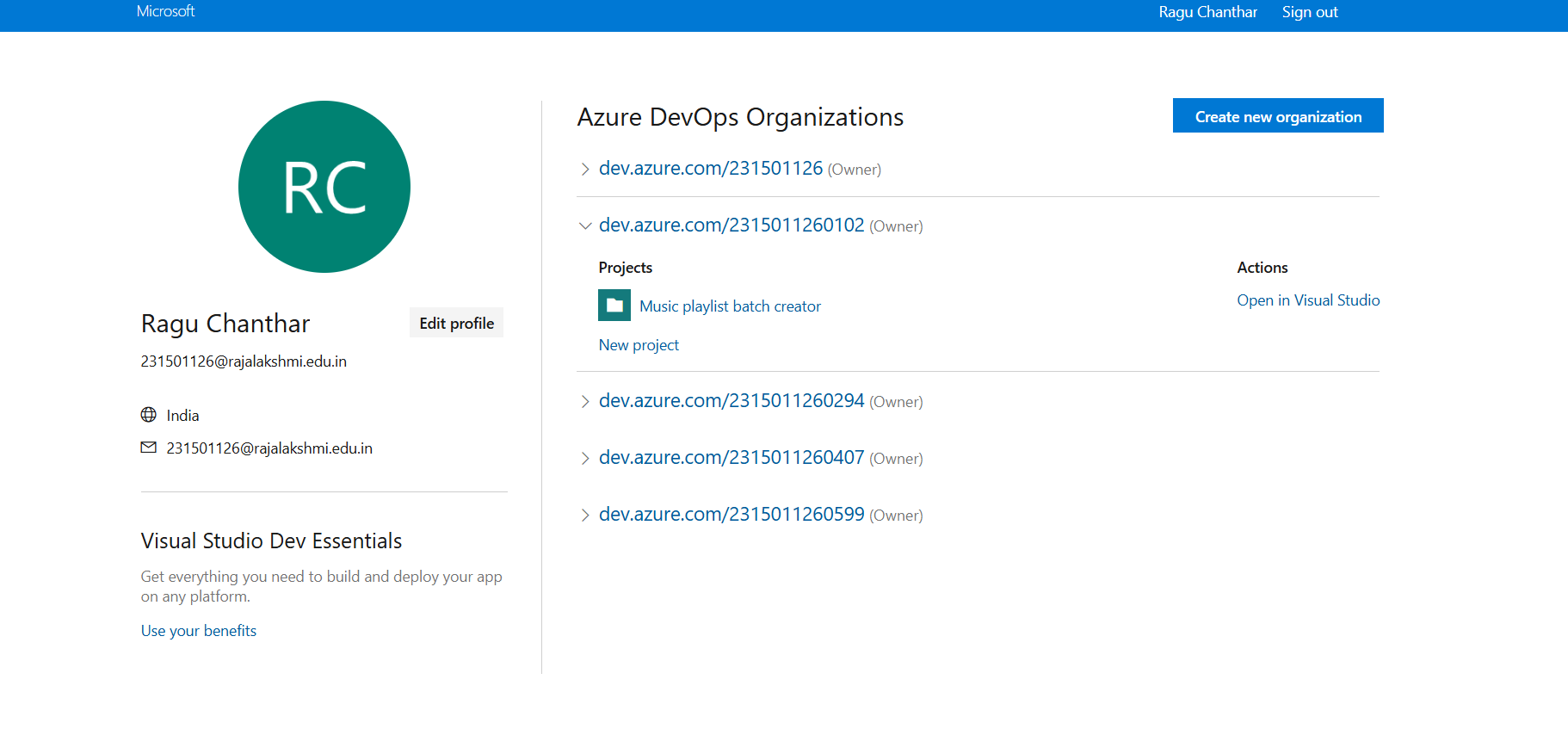
***Name****:* Choose a name for the project (e.g., LMS).

***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).

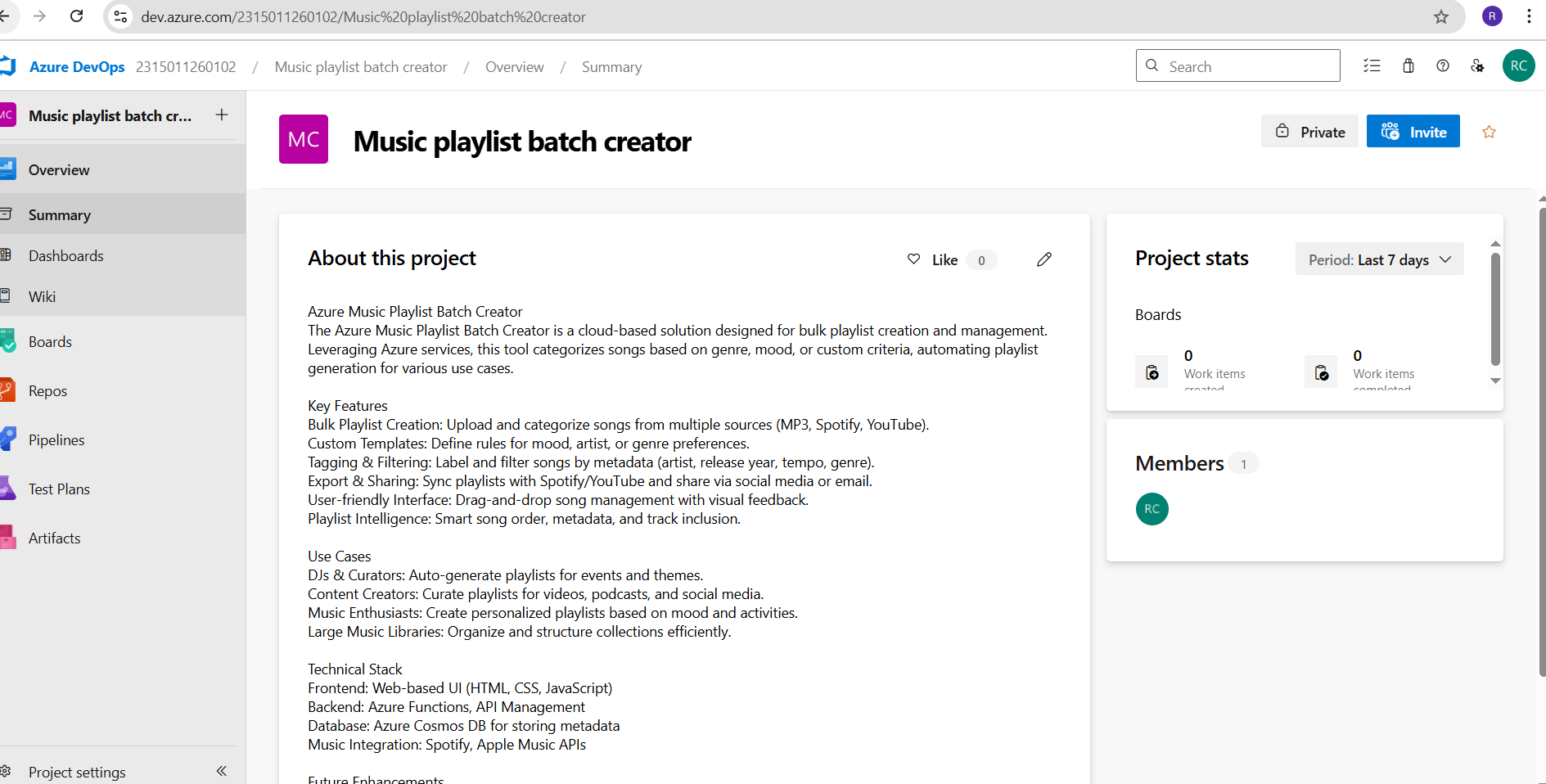
* 1. Once you’ve filled out the details, click **Create** to set up your first project.



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.



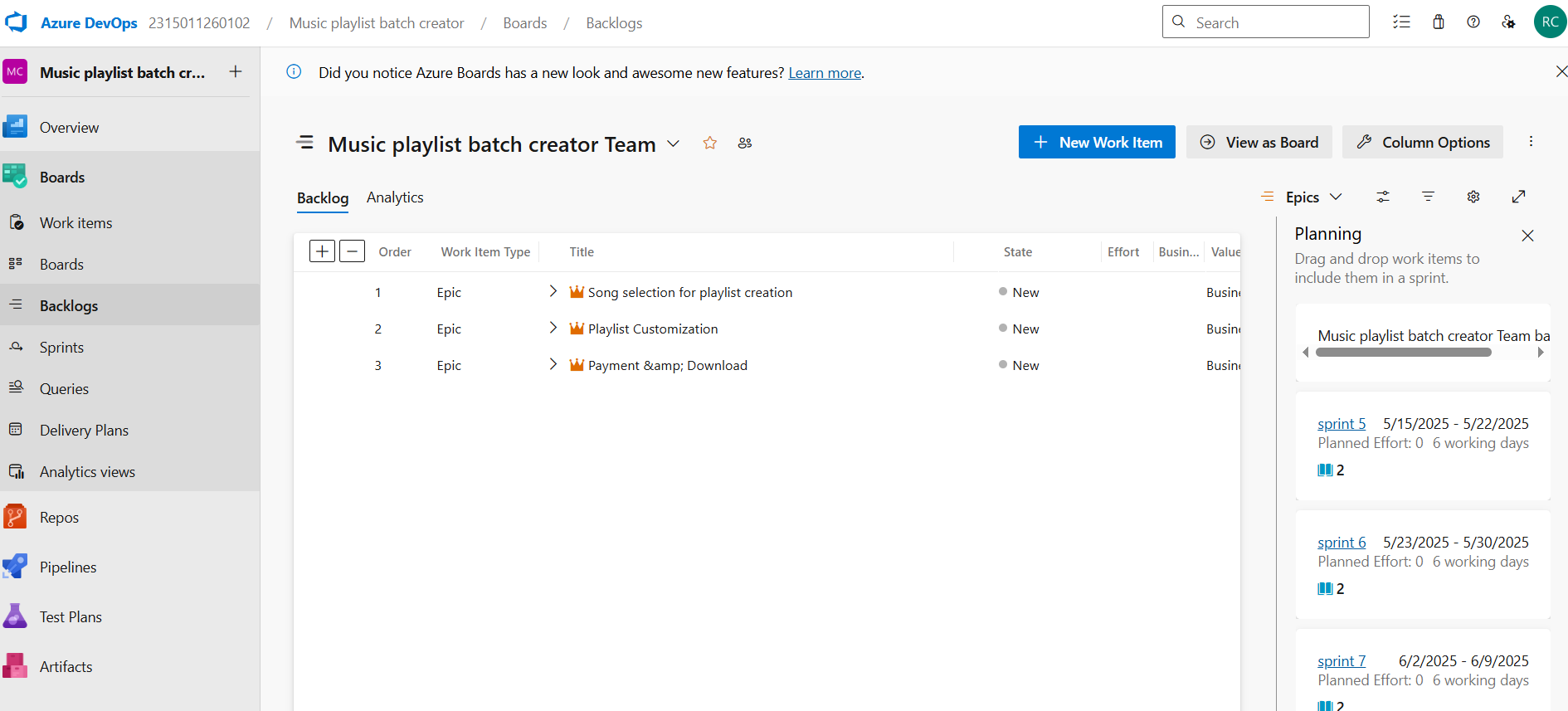
1. Project dashboard

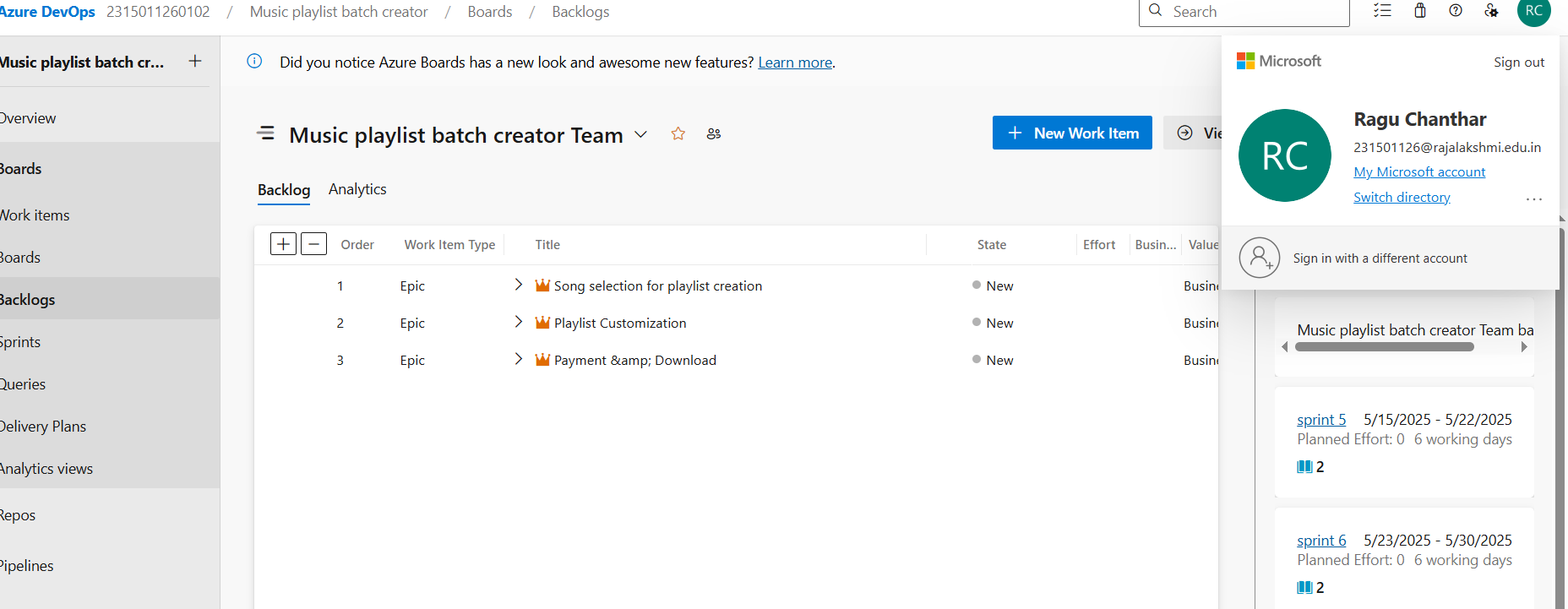


1. To manage user stories:
   1. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards**

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**

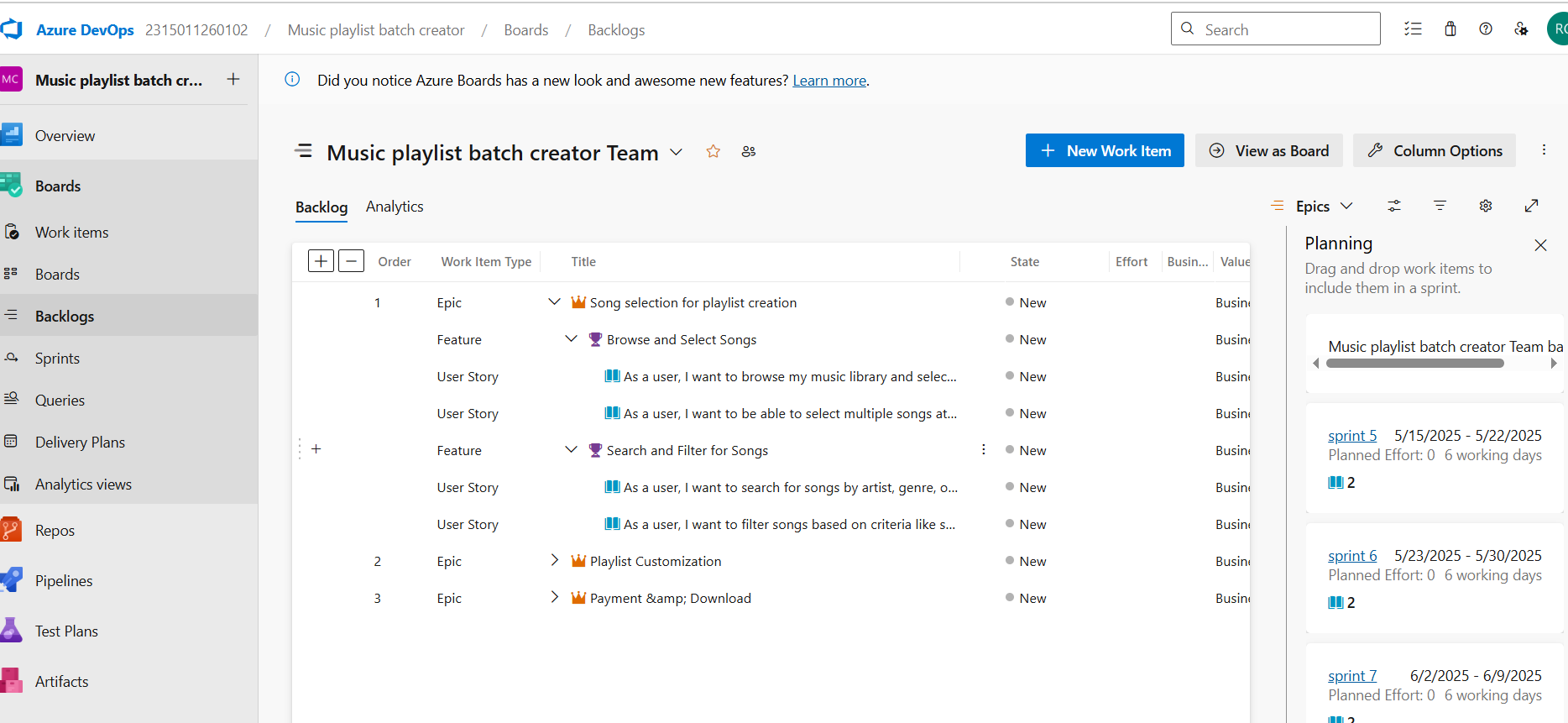
**EXP NO: 3**

**Aim:**

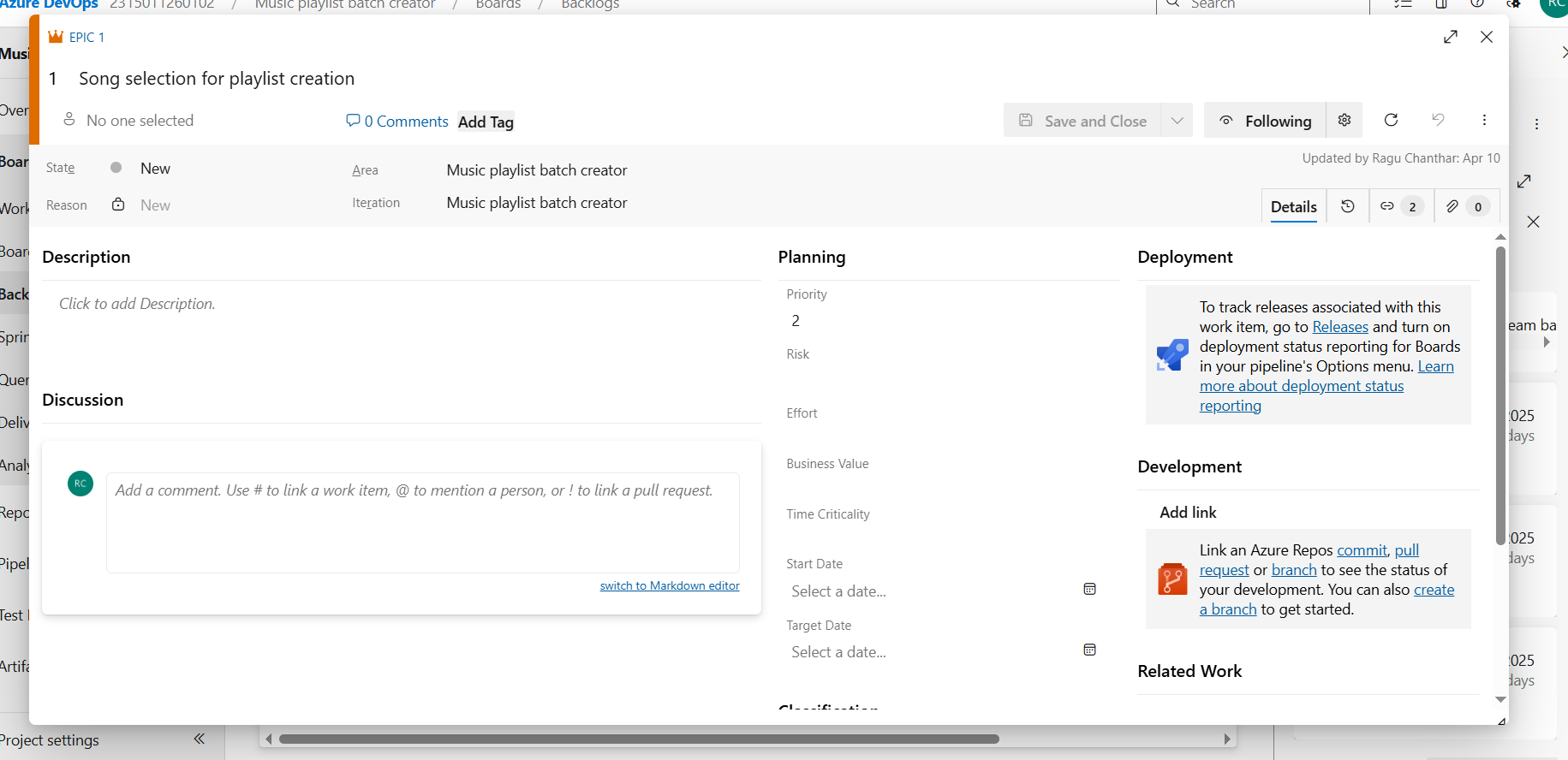
Create Epic, Features, User Stories, Task

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

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

****

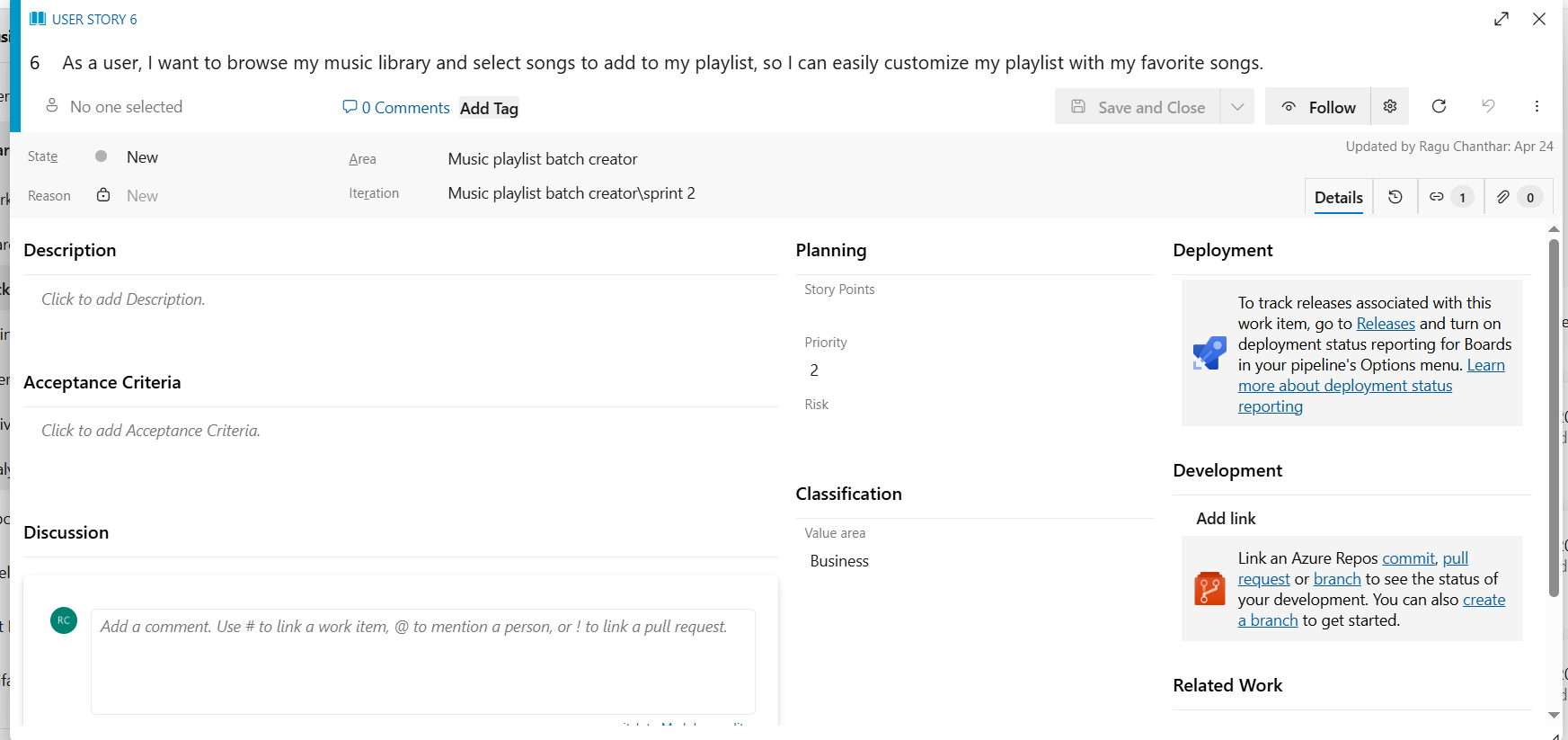
1. **Fill in Epics**

****

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**

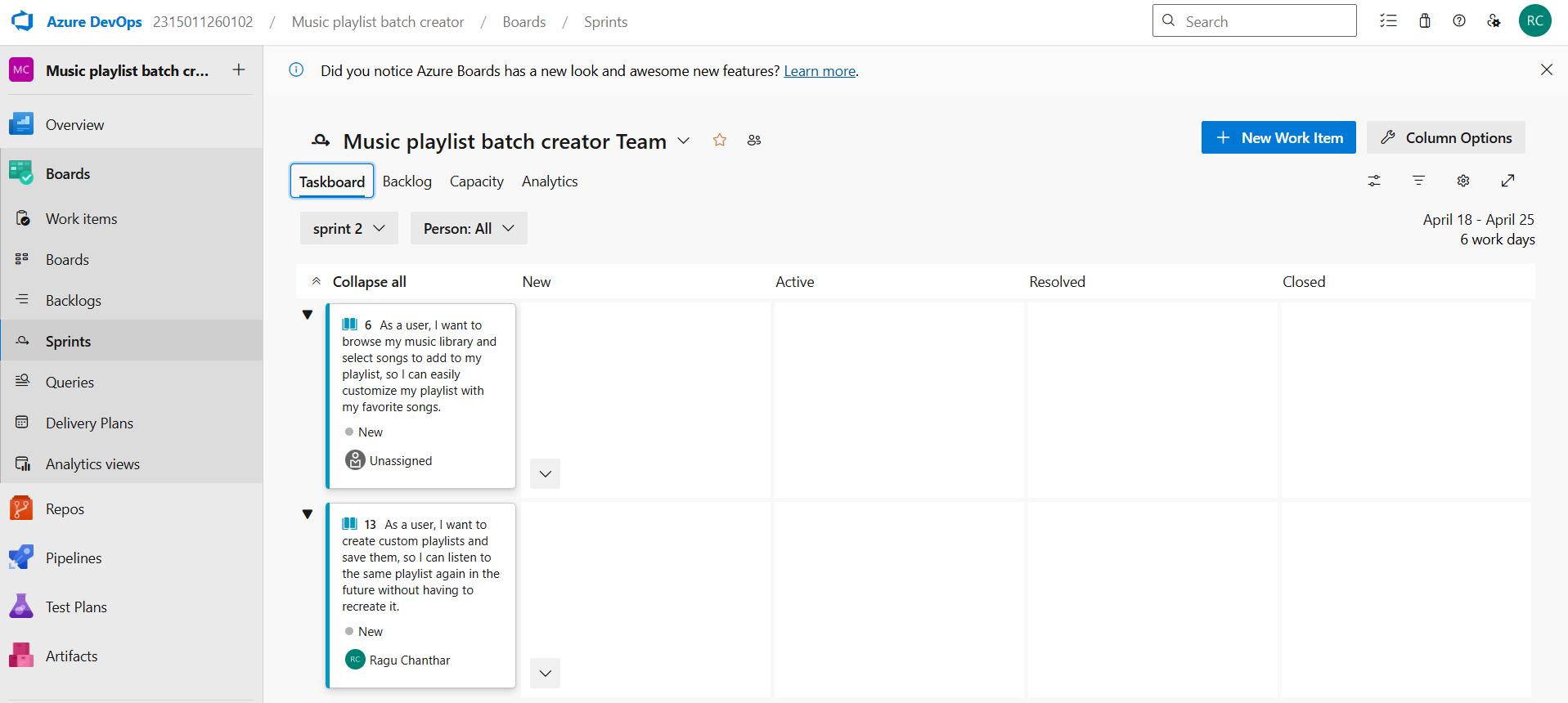
**EXP NO: 4**

**Aim:**

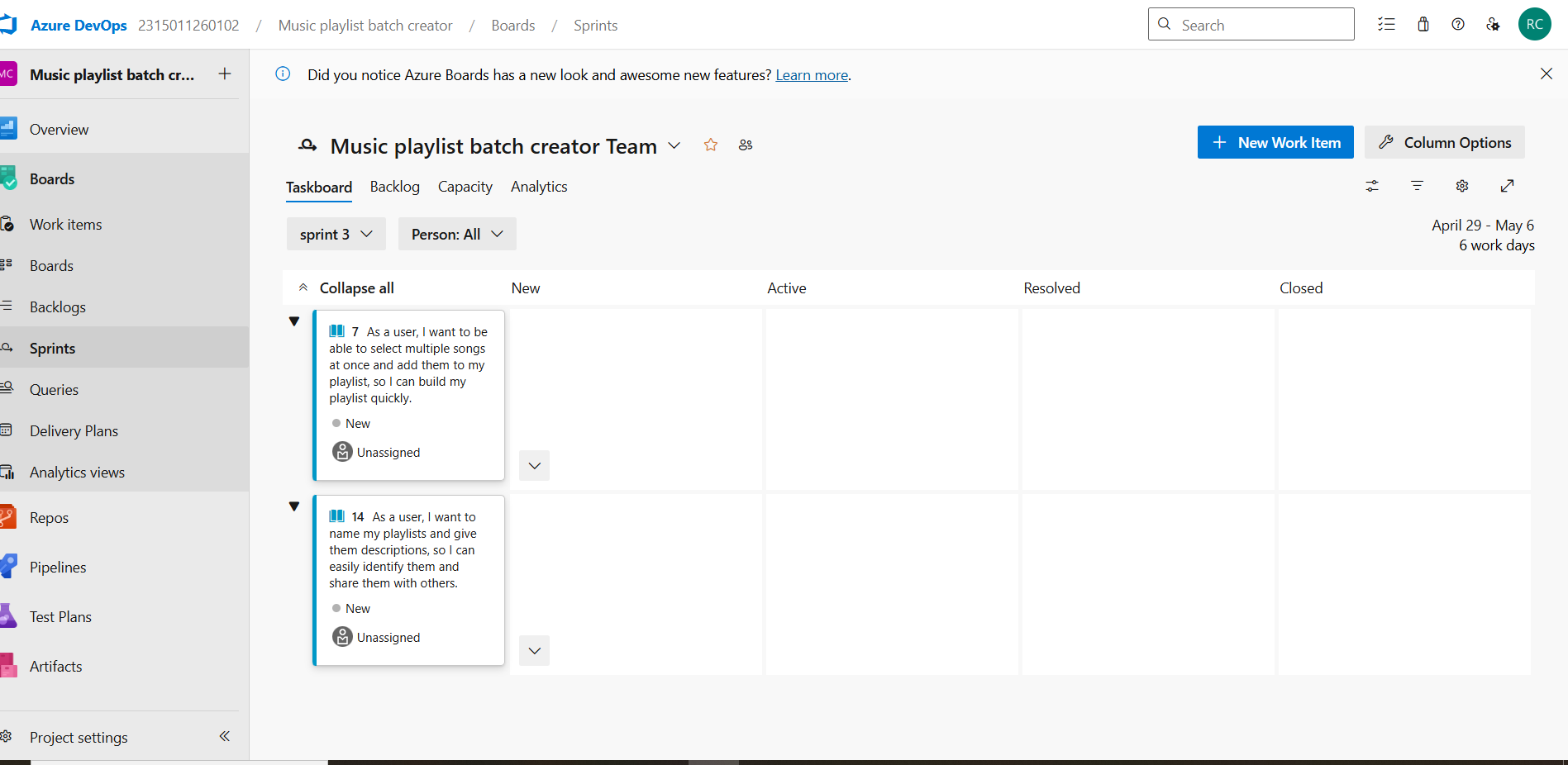
Create Epic, Features, User Stories, Task

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

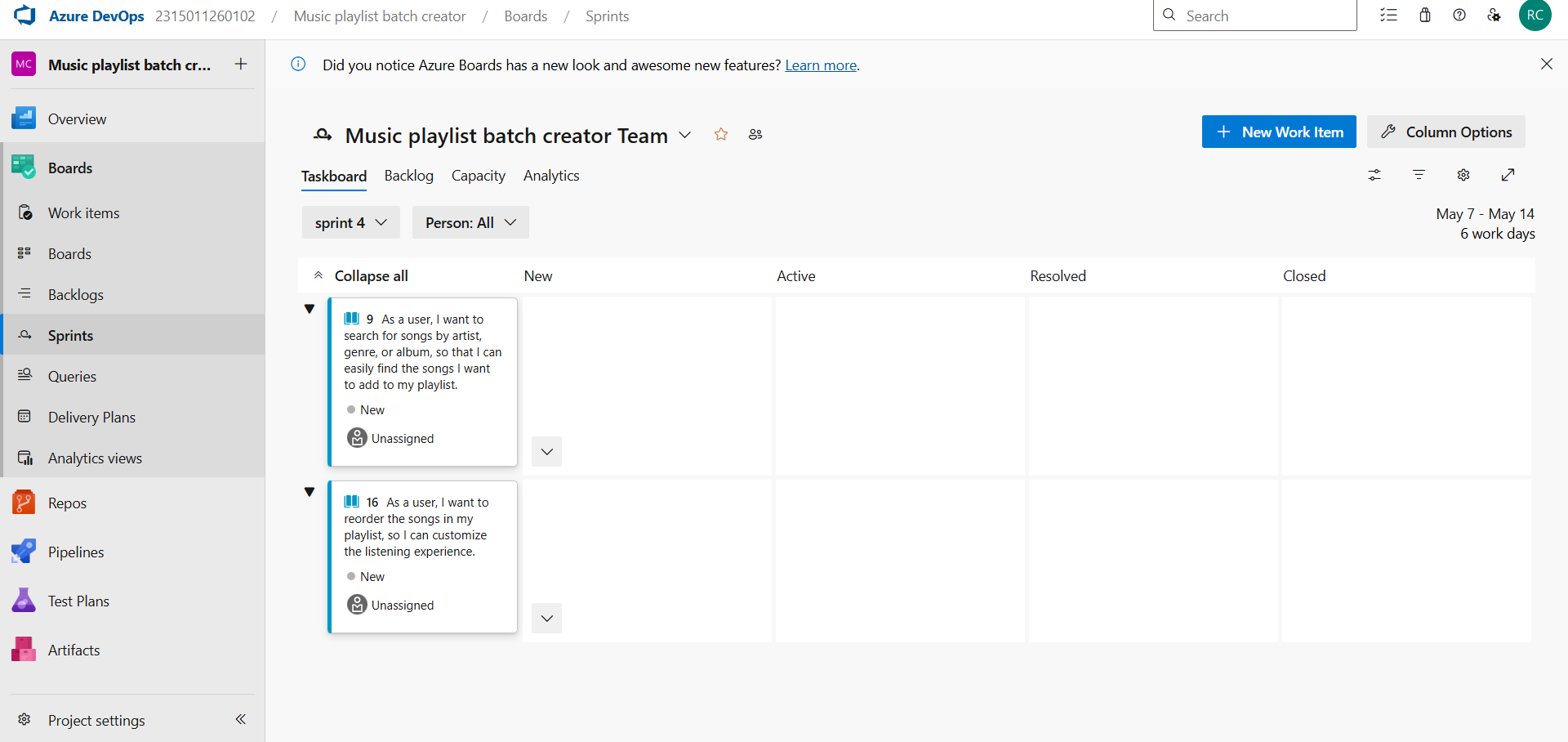
**Sprint Planning Sprint 1**

****

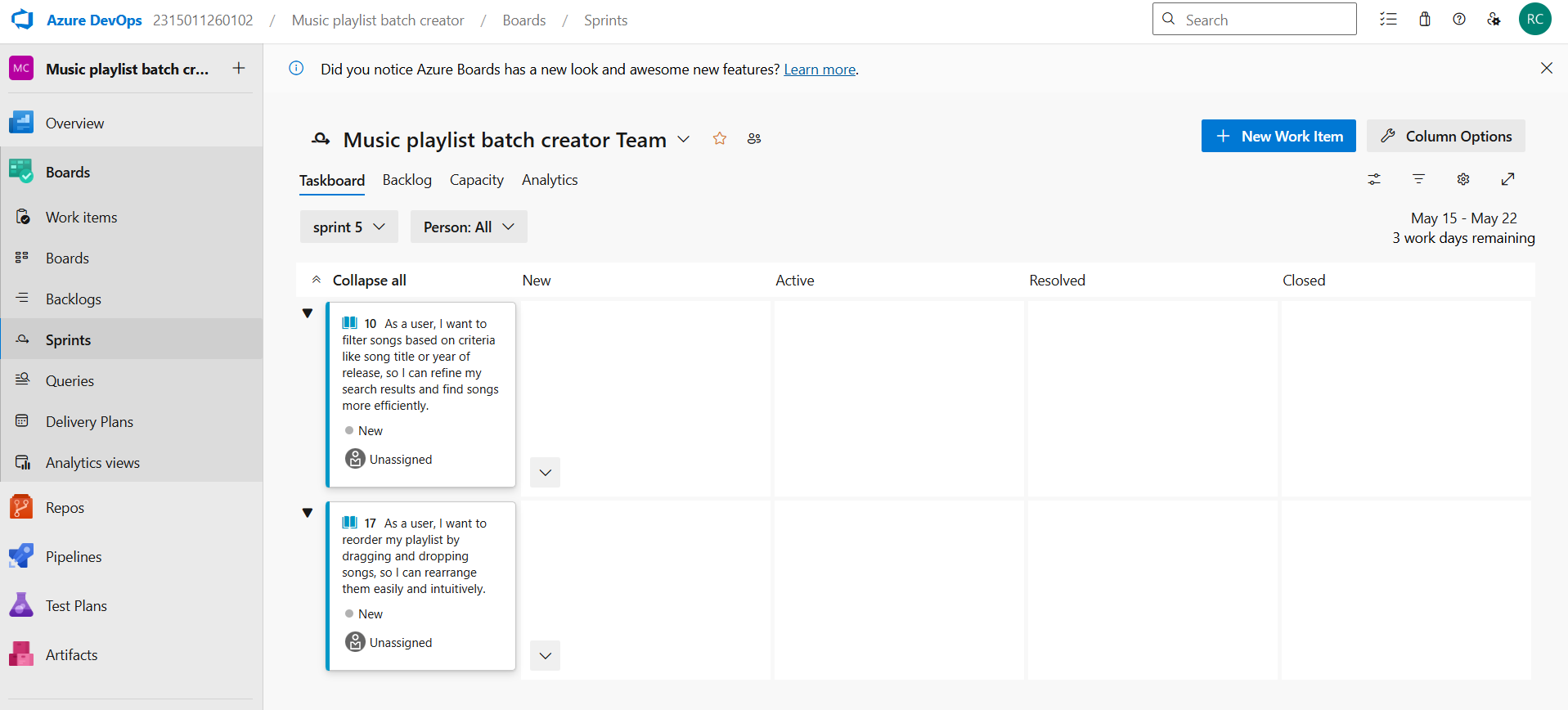
**Sprint 2**

****

**Sprint 3**

****

**Sprint 4**

****

**Result:**

The Sprints are created for the Music Playlist Batch Creator Project.

**POKER ESTIMATION**

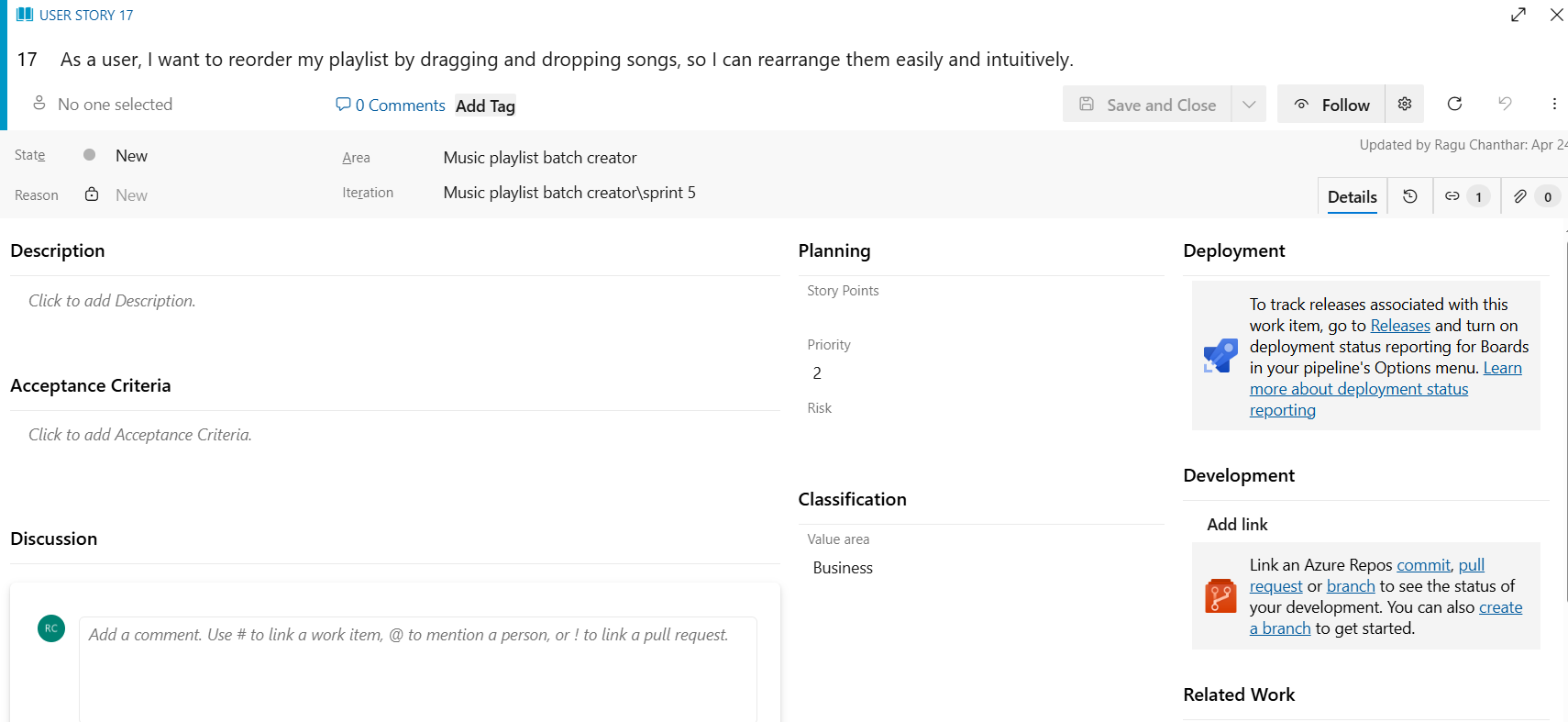
**EXP NO: 5**

**Aim:**

Create Epic, Features, User Stories, Task

Create Poker Estimation for the user stories - Music Playlist Batch Creator Project.

**Poker Estimation**

****

**Result:**

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

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

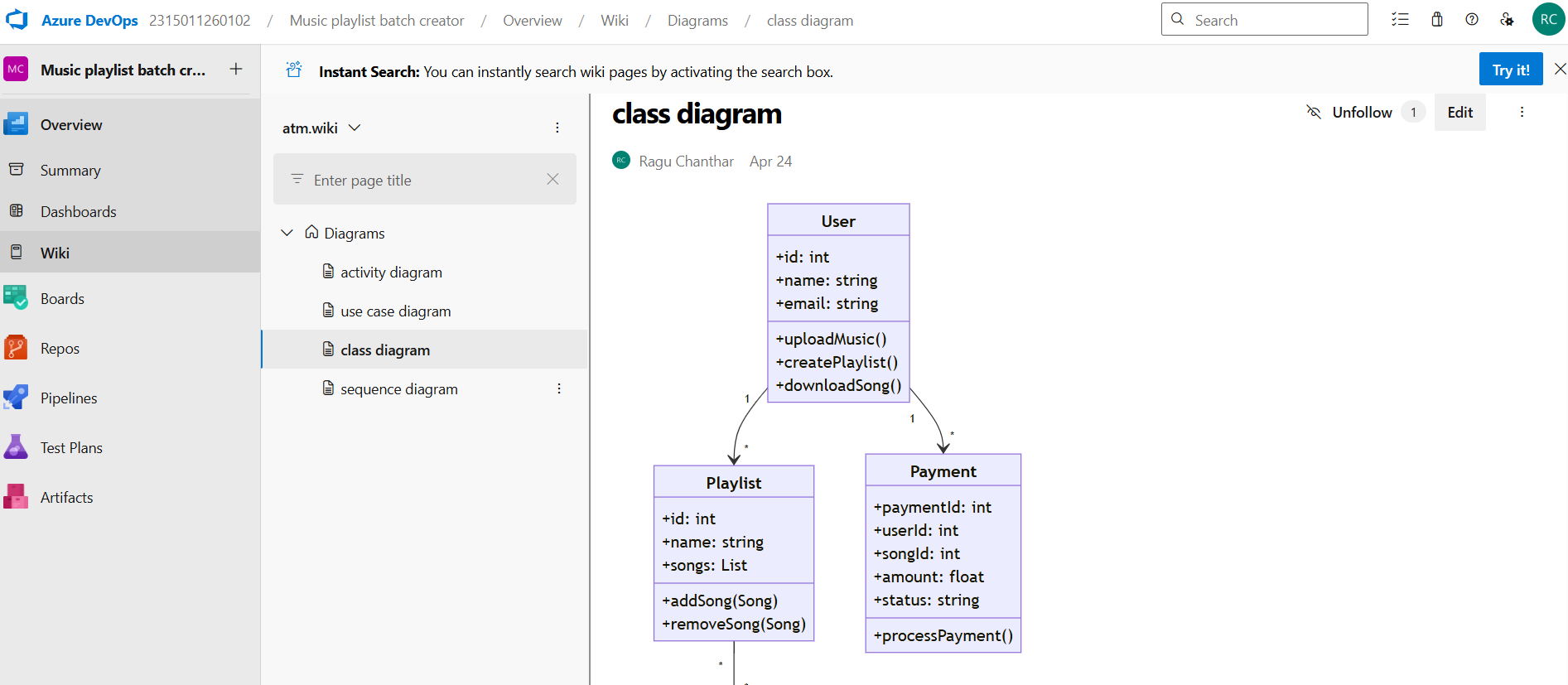
**EXP NO: 6**

**Aim:**

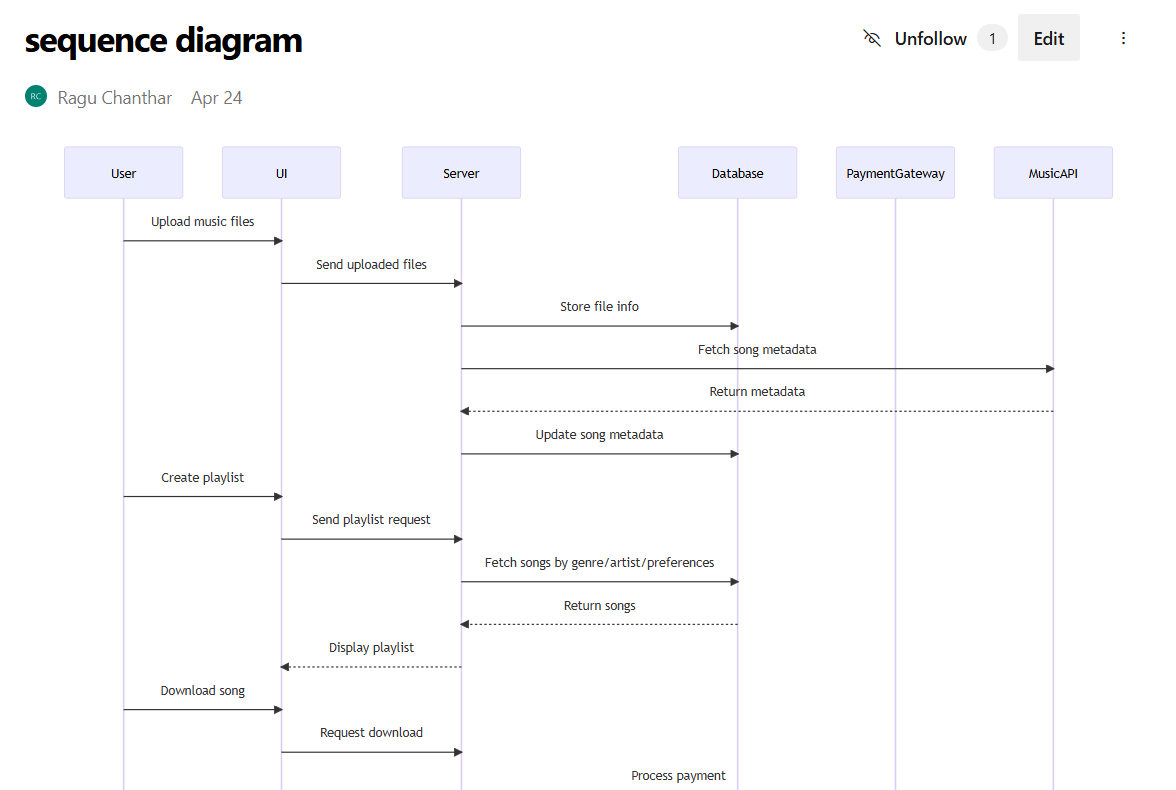
Create Epic, Features, User Stories, Task

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 USE CASE AND ACTIVITY DIAGRAMS FOR PROJECT STRUCTURE**

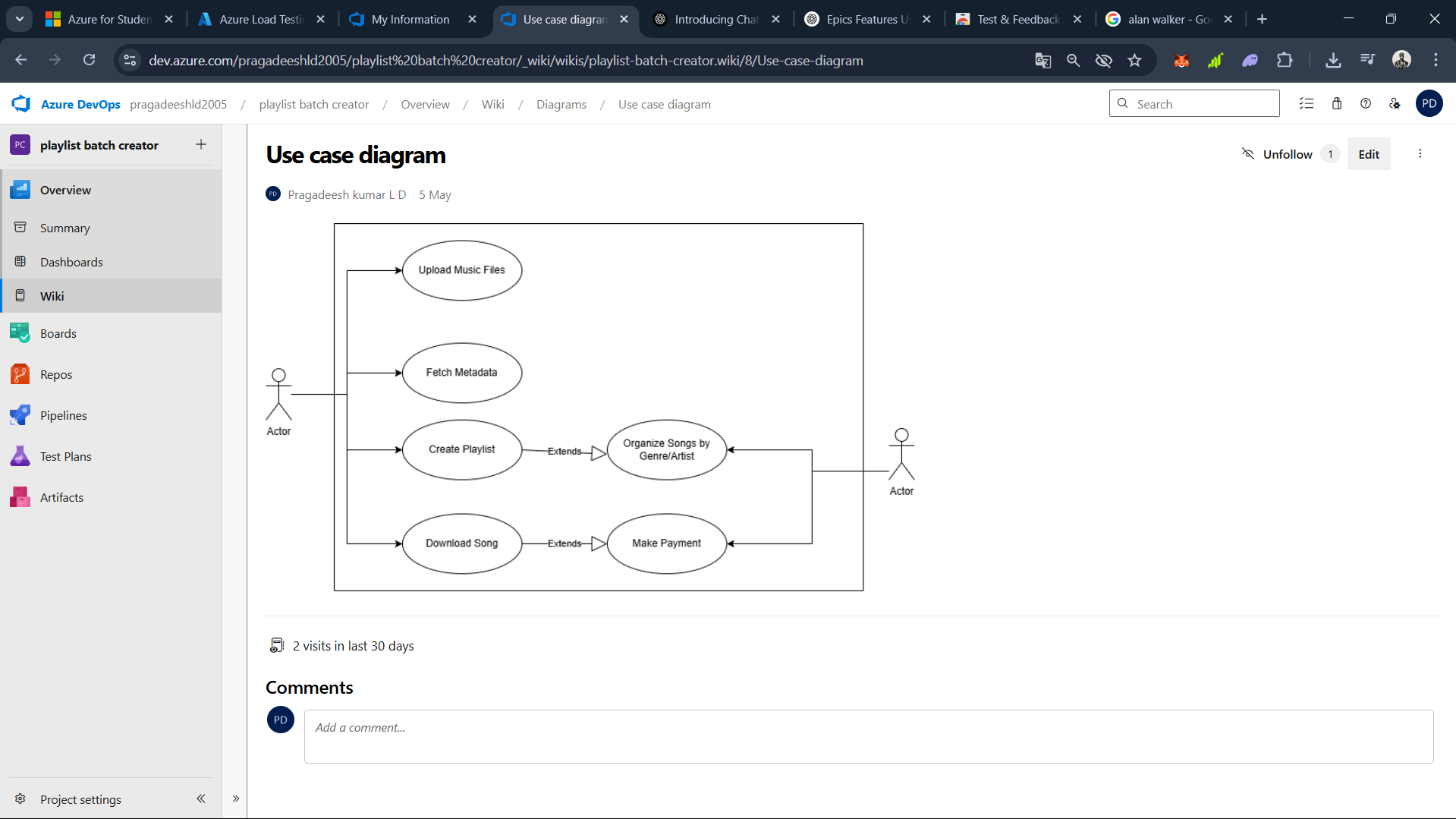
**EXP NO: 7**

**Aim:**

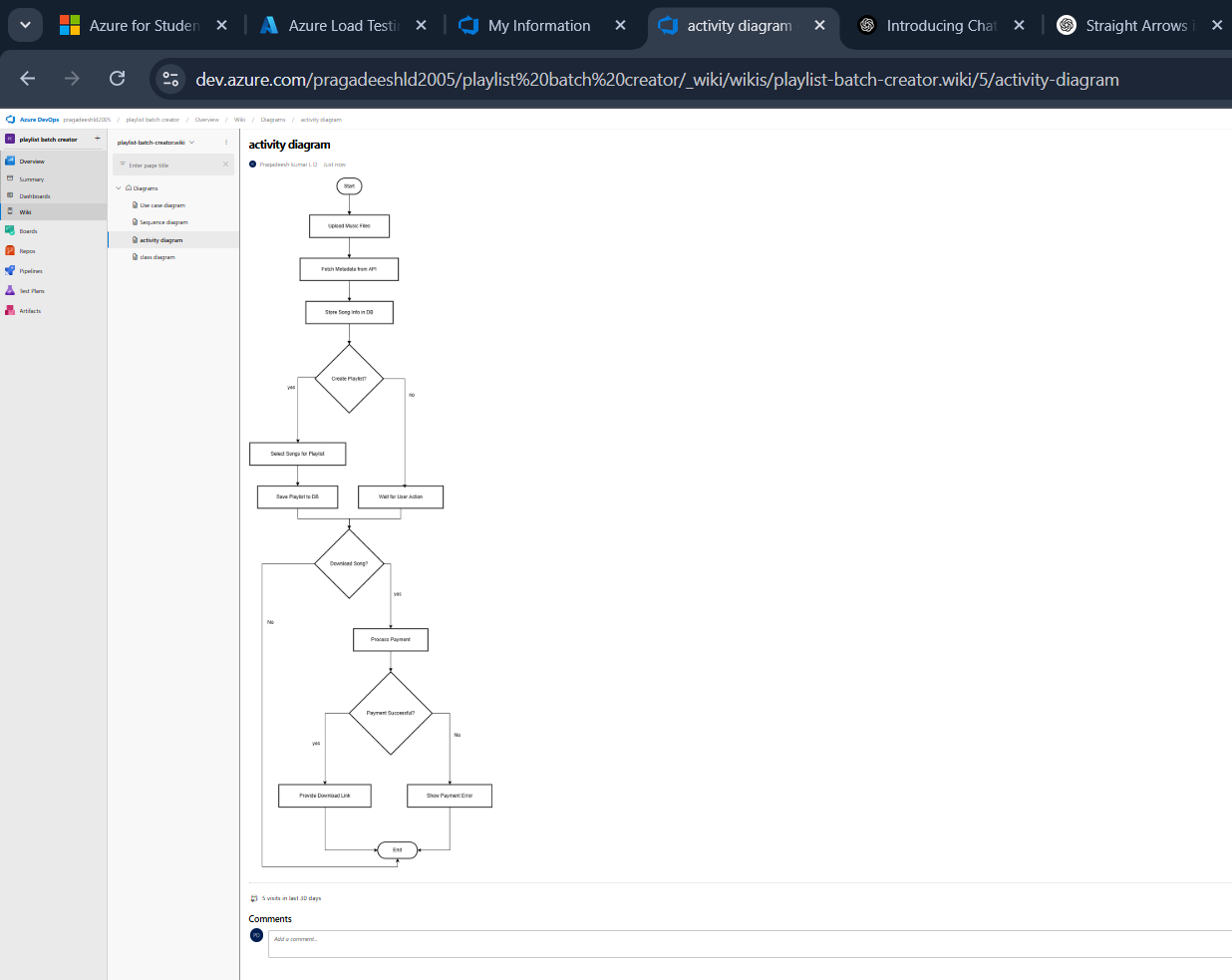
Create Epic, Features, User Stories, Task

To Design an Use Case Diagram and Activity Diagram for the given Project.

**7A. Use Case Diagram**

****

**7B.Activity Diagram**

****

**Result:**

Creator

The Use Case Diagram and Activity Diagram is designed Successfully for the Music Playlist Batch

**TESTING – TEST PLANS AND TEST CASES**

**EXP NO: 8**

**Aim:**

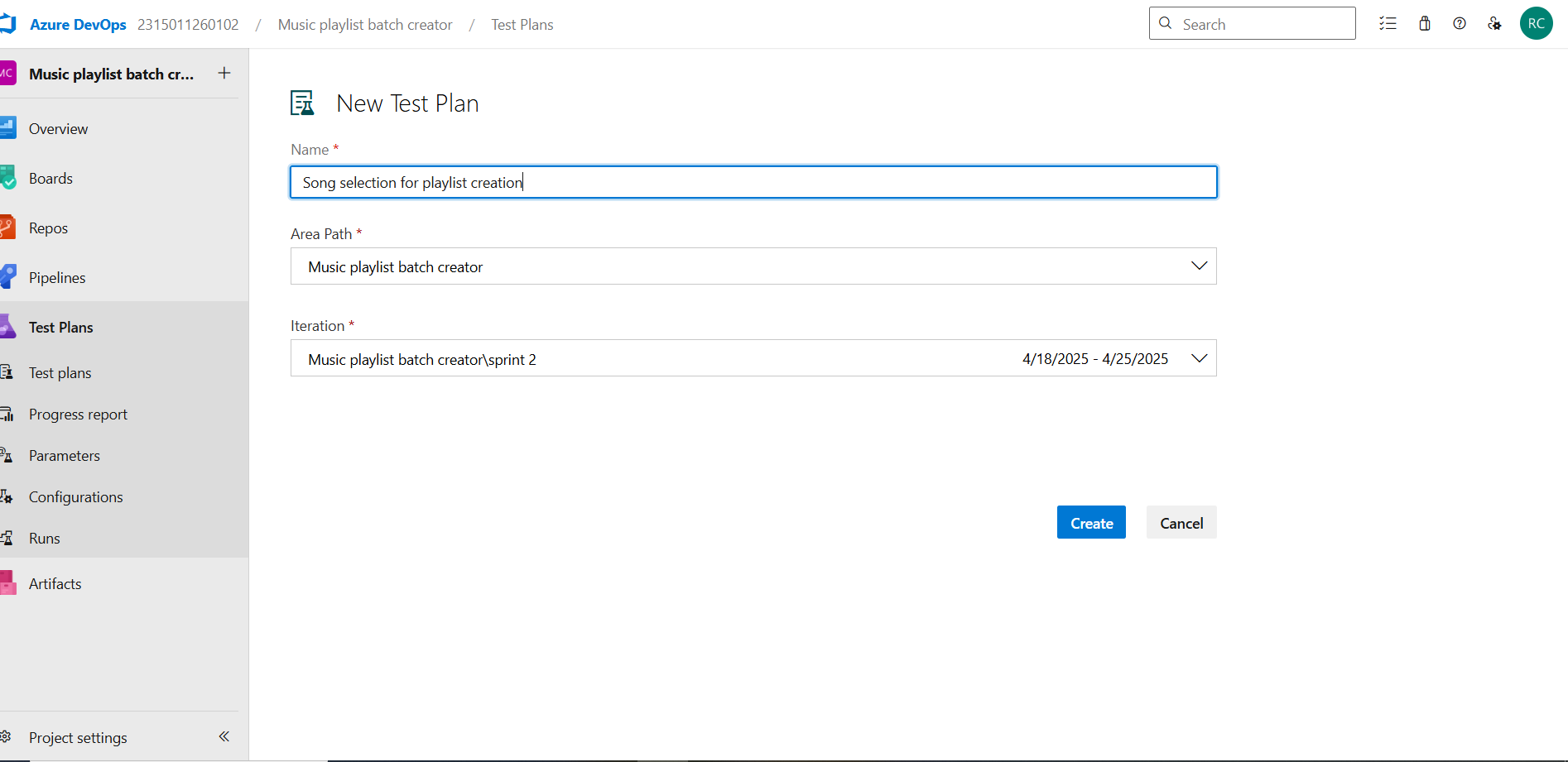
Create Epic, Features, User Stories, Task

Test Plans and Test Case and write two test cases for at least five user stories showcasing the

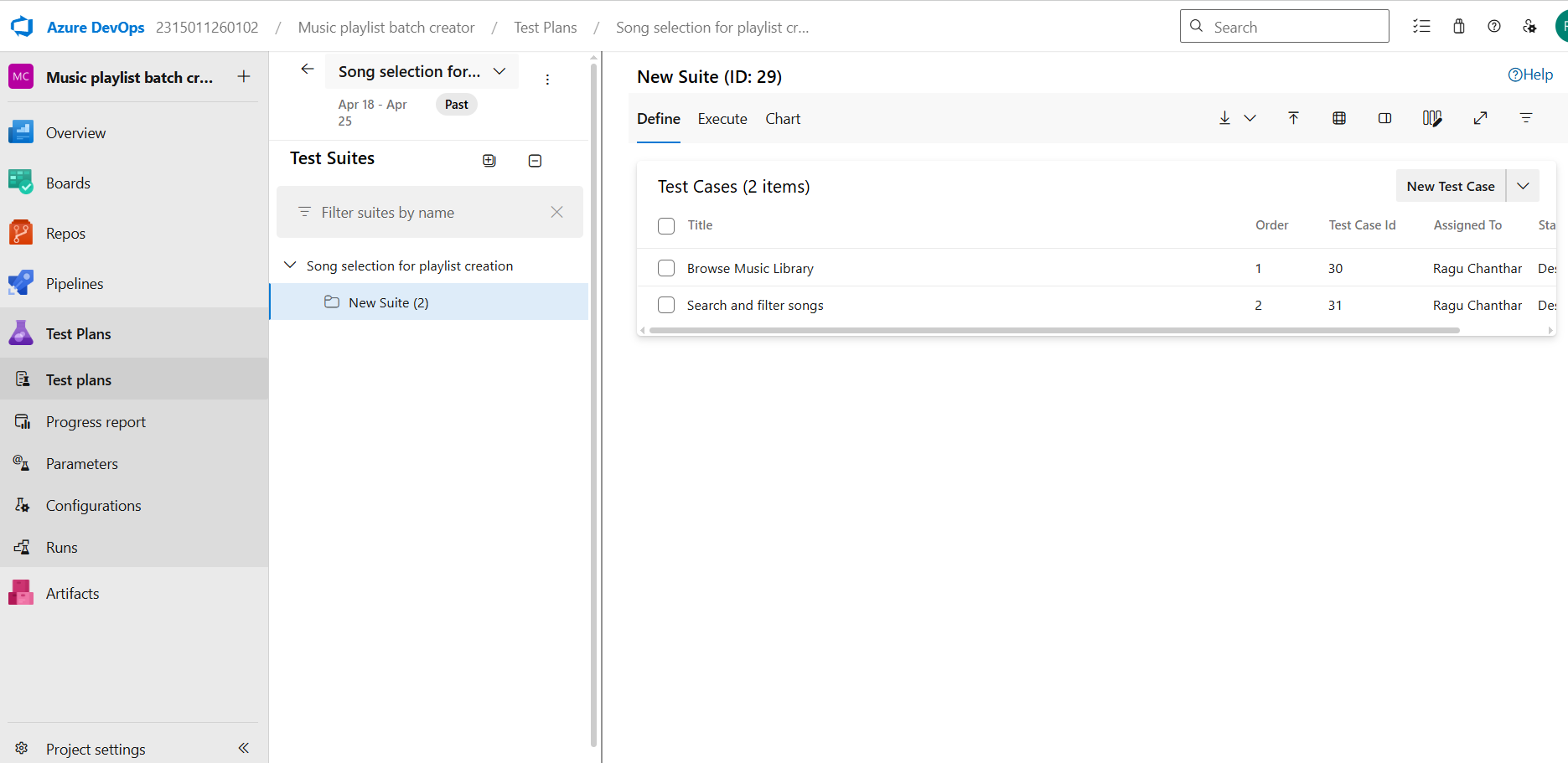
happy path and error scenarios in azure DevOps platform.

**Test Planning and Test Case Test Case Design Procedure**

1. **Understand Core Features of the Application**
   * User Signup & Login
   * Viewing and Managing Playlists
   * Fetching Real-time Metadata
   * Editing playlists (rename, reorder, record)
   * Creating smart audio playlists based on categories (mood, genre, artist, etc.)
2. **Define User Interactions**
   * Each test case simulates a real user behaviour (e.g., logging in, renaming a playlist, adding a song).
3. **Design Happy Path Test Cases**
   * Focused on validating that all features function as expected under normal conditions.
   * Example: User logs in successfully, adds item to playlist, or creates a category-based playlist.
4. **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.
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 (e.g., TC01 – Successful Login, TC10 – Save Playlist Fails).
   * Helps in quick identification and linking to user stories or features.
7. **Separate Test Suites**
   * Grouped test cases based on functionality (e.g., Login, Playlist Editing, Recommendation System).
   * 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.
9. **New test plan**

****

1. **Test suite**

****

1. **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**

1. As a user, I want to browse my music library and select songs to add to a playlist.
2. As a user, I want to be able to select multiple songs at once.
3. As a user, I want to search for songs by artist, genre, or title.
4. As a user, I want to filter songs based on criteria like style or release year.
5. As a user, I want to create custom playlists and save them.

**Test Suites**

**Test Suite: TS06 – Song Selection for Playlist Creation**

**Feature:** Browse and Select Songs, Search and Filter for Songs

**🔹 TC15 – Browse Music Library**

* **Objective:** Verify the user can browse the music library.
* **Steps:**
  1. Login to the app.
  2. Navigate to the music library section.
  3. Scroll through available songs.
* **Expected Result:** Songs should be visible and scrollable.
* **Type:** Happy Path

**🔹 TC16 – Multi-select Songs**

* **Objective:** Verify multiple songs can be selected.
* **Steps:**
  1. Navigate to the music library.
  2. Select multiple songs using checkboxes or multi-select options.
* **Expected Result:** Multiple songs should be selected without losing previous selections.
* **Type:** Happy Path

**🔹 TC17 – Search and Filter Songs**

* **Objective:** Verify search and filtering functionality.
* **Steps:**
  1. Enter a keyword in the search bar (e.g., artist name).
  2. Apply filters like genre or year.
* **Expected Result:** Only relevant songs should be displayed based on search/filter criteria.
* **Type:** Happy Path

**✅ Test Suite: TS07 – Playlist Customization**

**Feature:** Create and Save Playlists, Edit Playlist Order

**🔹 TC18 – Create Playlis**

* **Objective:** Verify the user can create a custom playlist.
* **Steps:**
  1. Click "Create Playlist."
  2. Add songs.
  3. Name the playlist and save.
* **Expected Result:** Playlist should be created and saved successfully.
* **Type:** Happy Path

**🔹 TC19 – Name and Save Playlist**

* **Objective:** Ensure playlists can be named and saved with a custom title.
* **Steps:**
  1. Enter a custom name.
  2. Click “Save.”
* **Expected Result:** Playlist should be saved with the correct name.
* **Type:** Happy Path

**🔹 TC20 – Reorder Songs in Playlist**

* **Objective:** Test drag-and-drop or manual reordering.
* **Steps:**
  1. Open an existing playlist.
  2. Reorder songs using drag-and-drop or arrow buttons.
* **Expected Result:** Songs should rearrange correctly and persist after saving.
* **Type:** Happy Path

**✅ Test Suite: TS08 – Payment & Download**

**Feature:** Paid Song Downloads, Download Access Control

**🔹 TC21 – Purchase Song**

* **Objective:** Verify users can purchase individual songs.
* **Steps:**
  1. Select a song to purchase.
  2. Proceed to payment.
  3. Complete the transaction.
* **Expected Result:** Payment should be successful and the song marked as purchased.
* **Type:** Happy Path

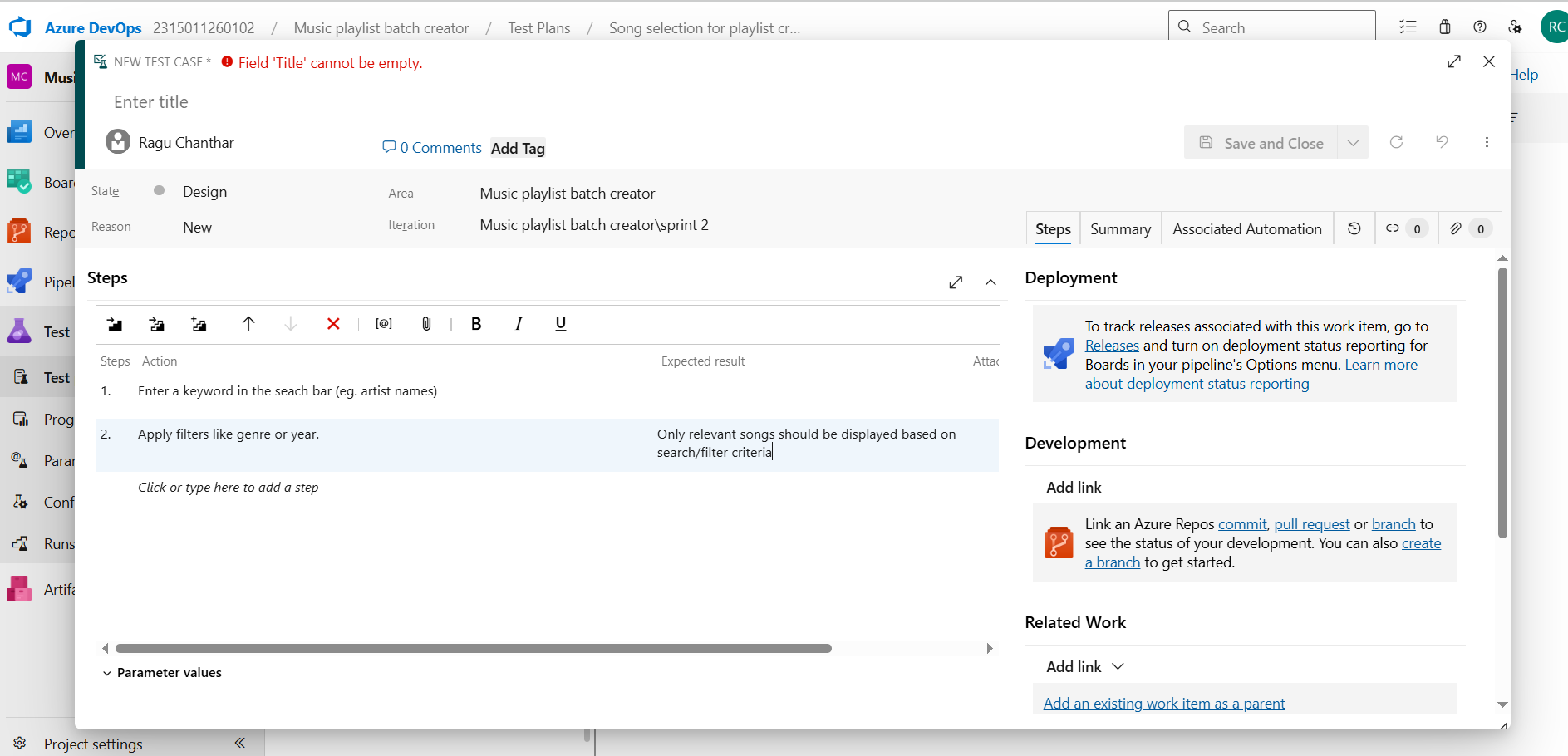
**🔹 TC22 – Secure Payment Flow**

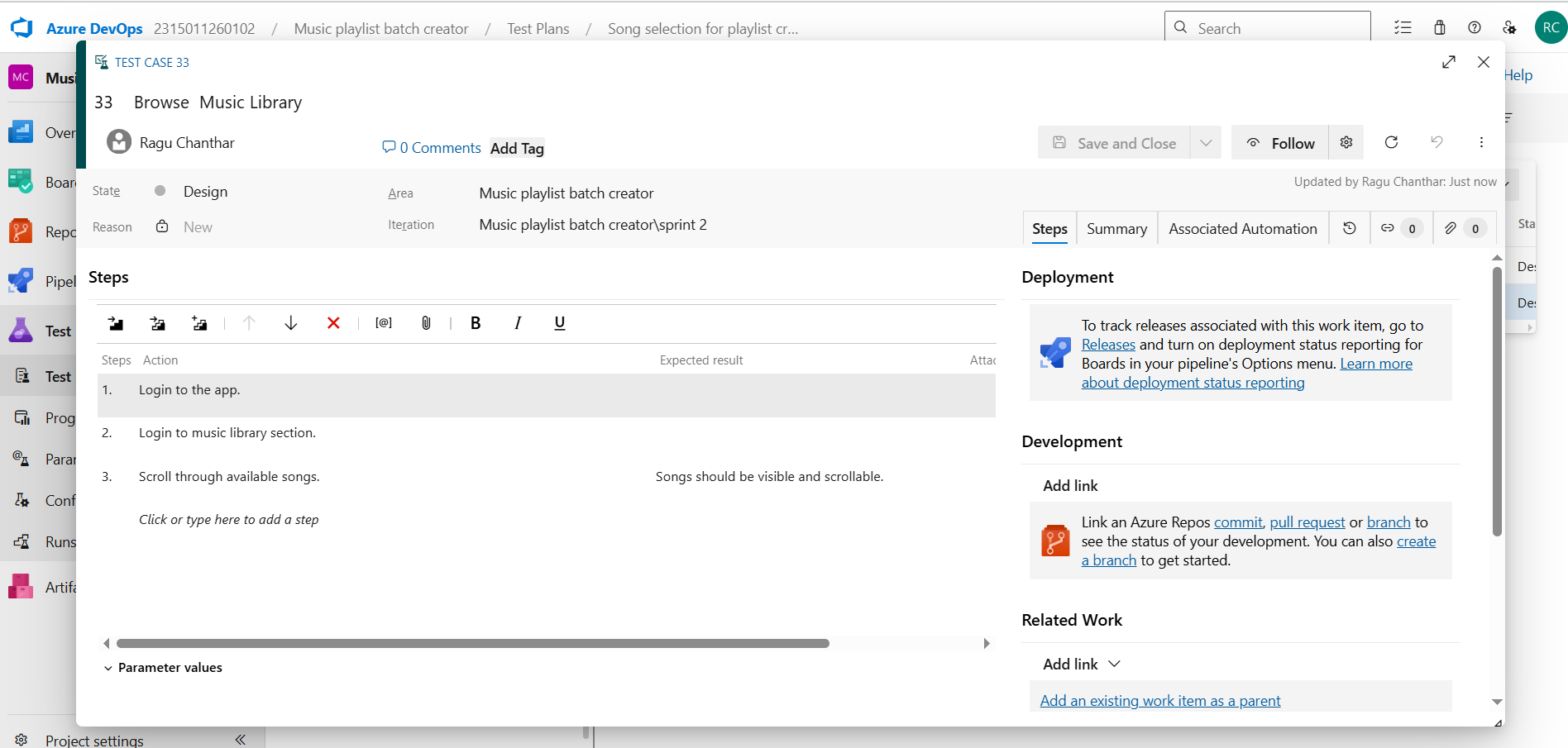
* **Objective:** Ensure payment is handled securely.
* **Steps:**
  1. Initiate payment.
  2. Enter card or payment details.
* **Expected Result:** Data is transmitted securely (e.g., via HTTPS), and the payment gateway processes the transaction without errors.
* **Type:** Happy Path

**🔹 TC23 – Download Purchased Songs**

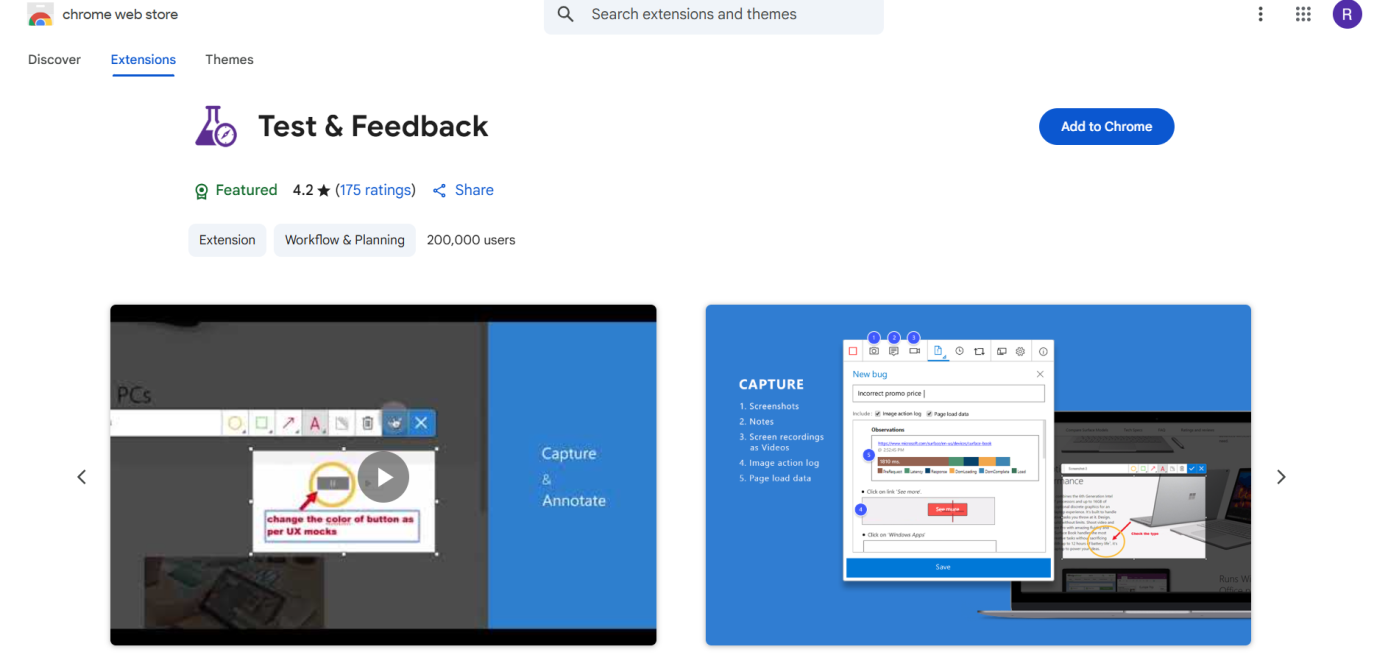
* **Objective:** Confirm users can download purchased songs.
* **Steps:**
  1. Navigate to purchased songs.
  2. Click “Download.”
* **Expected Result:** Download should begin, and a confirmation email should be sent.
* **Type:** Happy Path

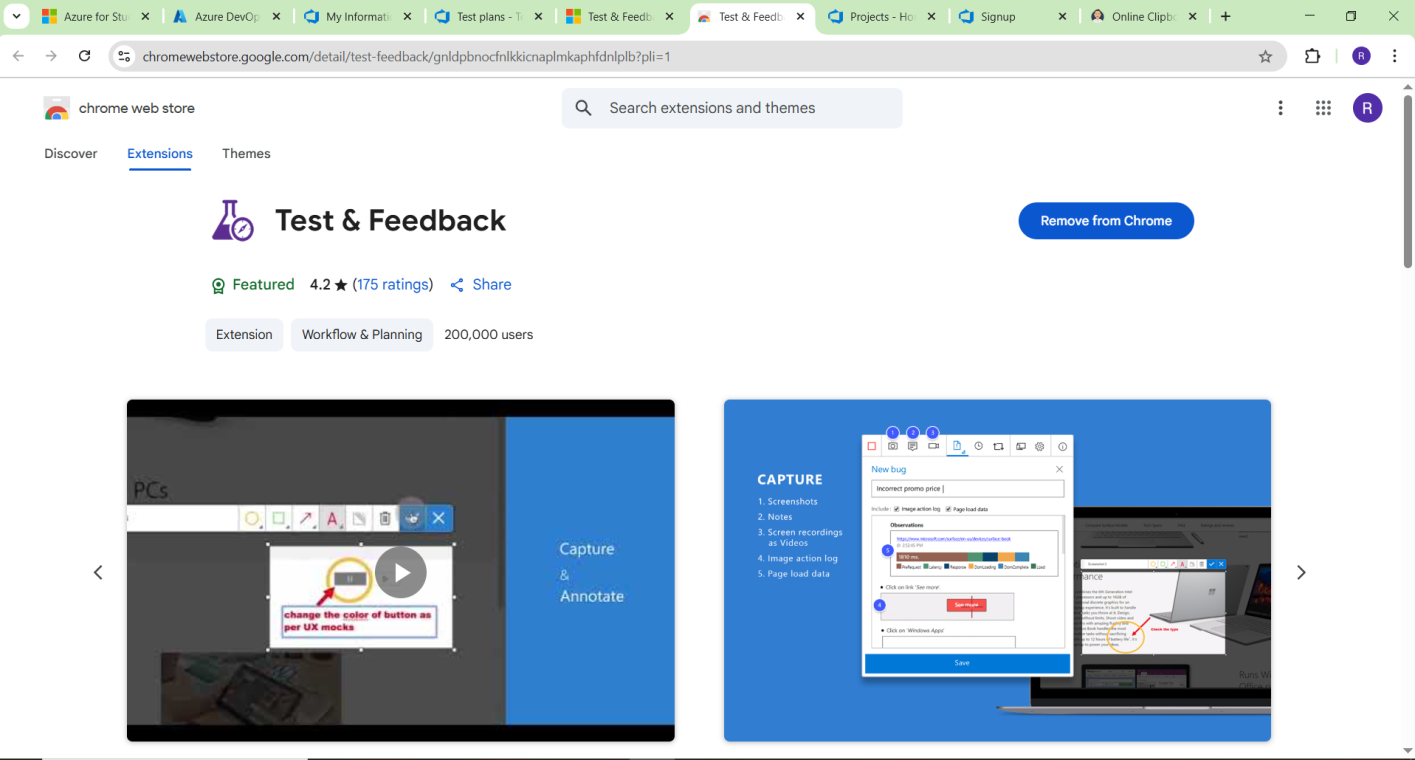
**Test Cases**

****

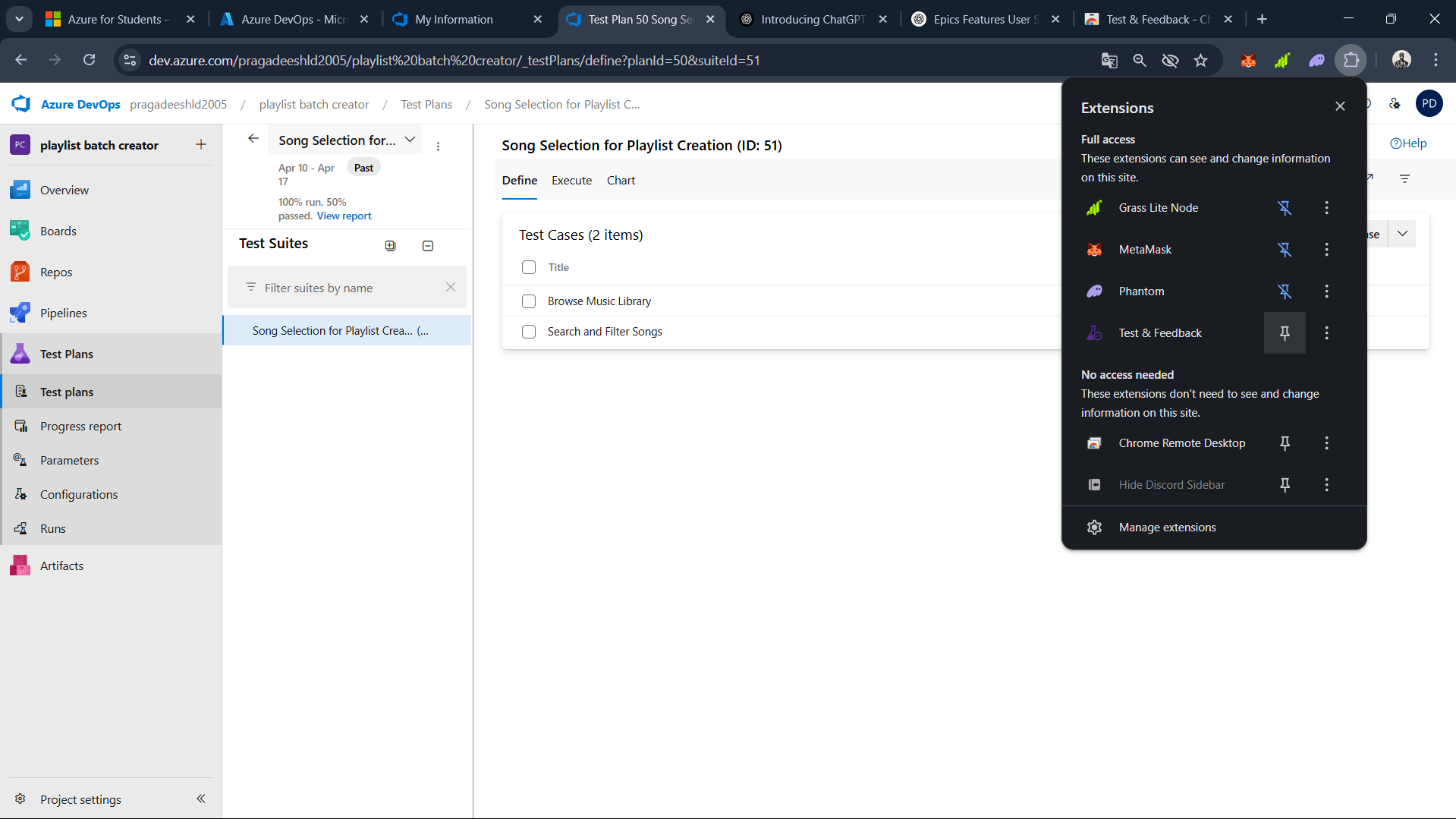
****

1. **Installation of test**

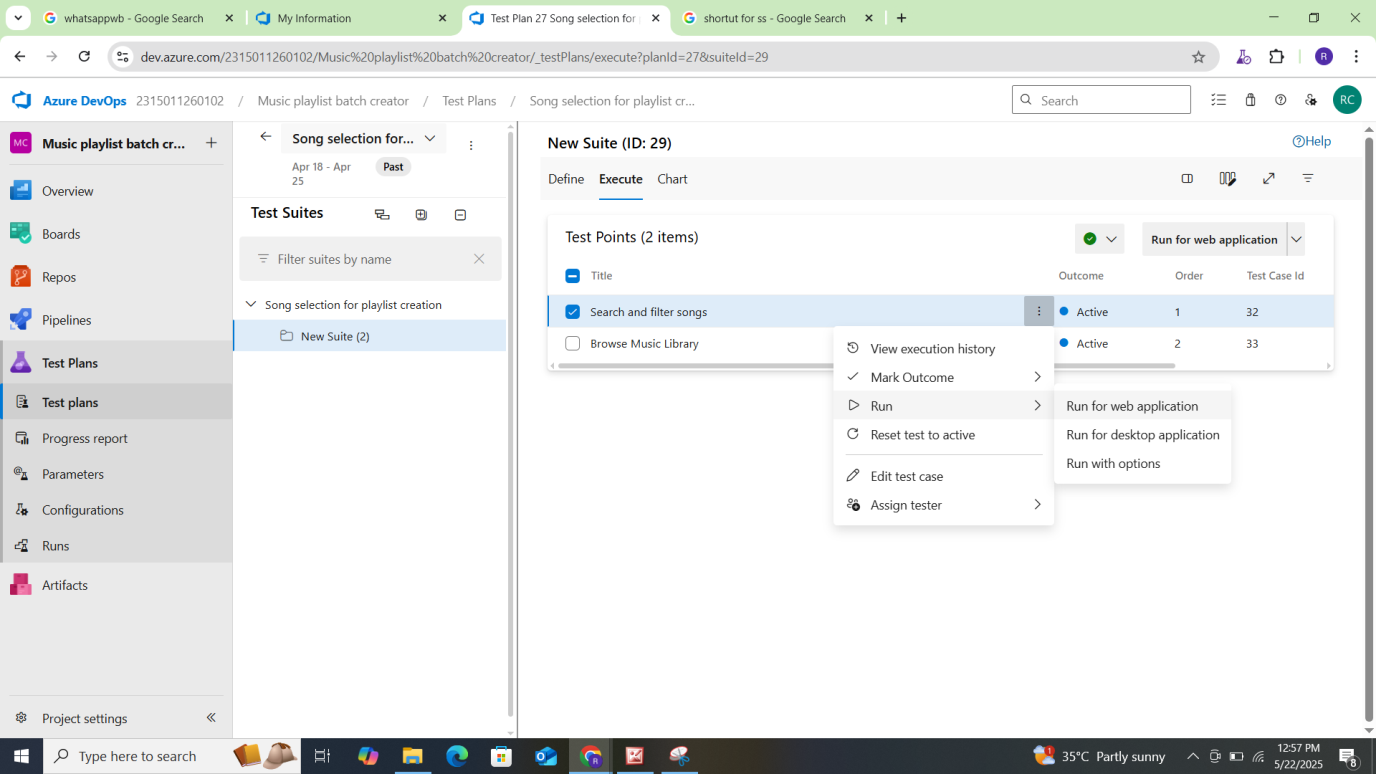
****

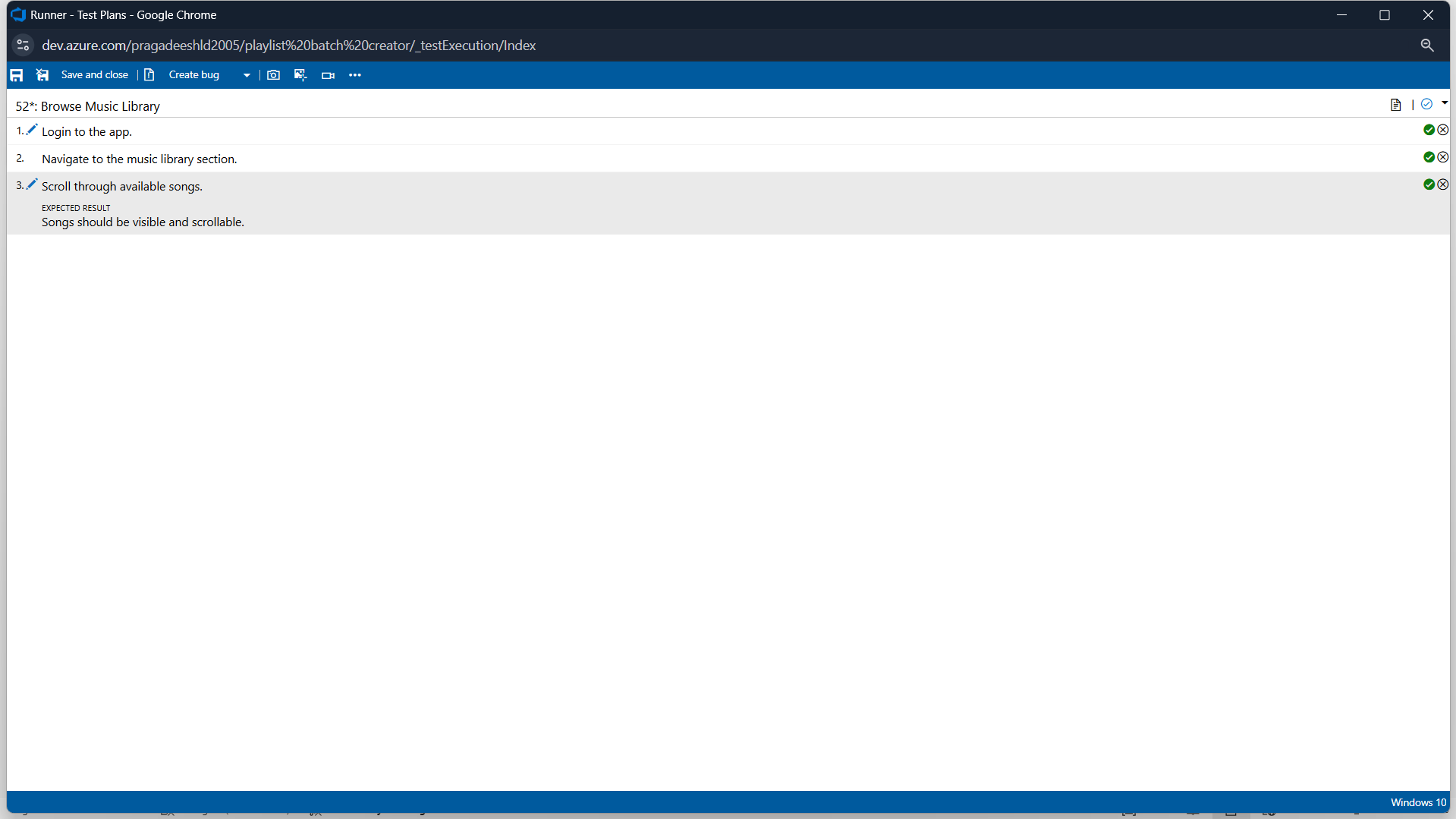


Test and feedback Showing it as an extension

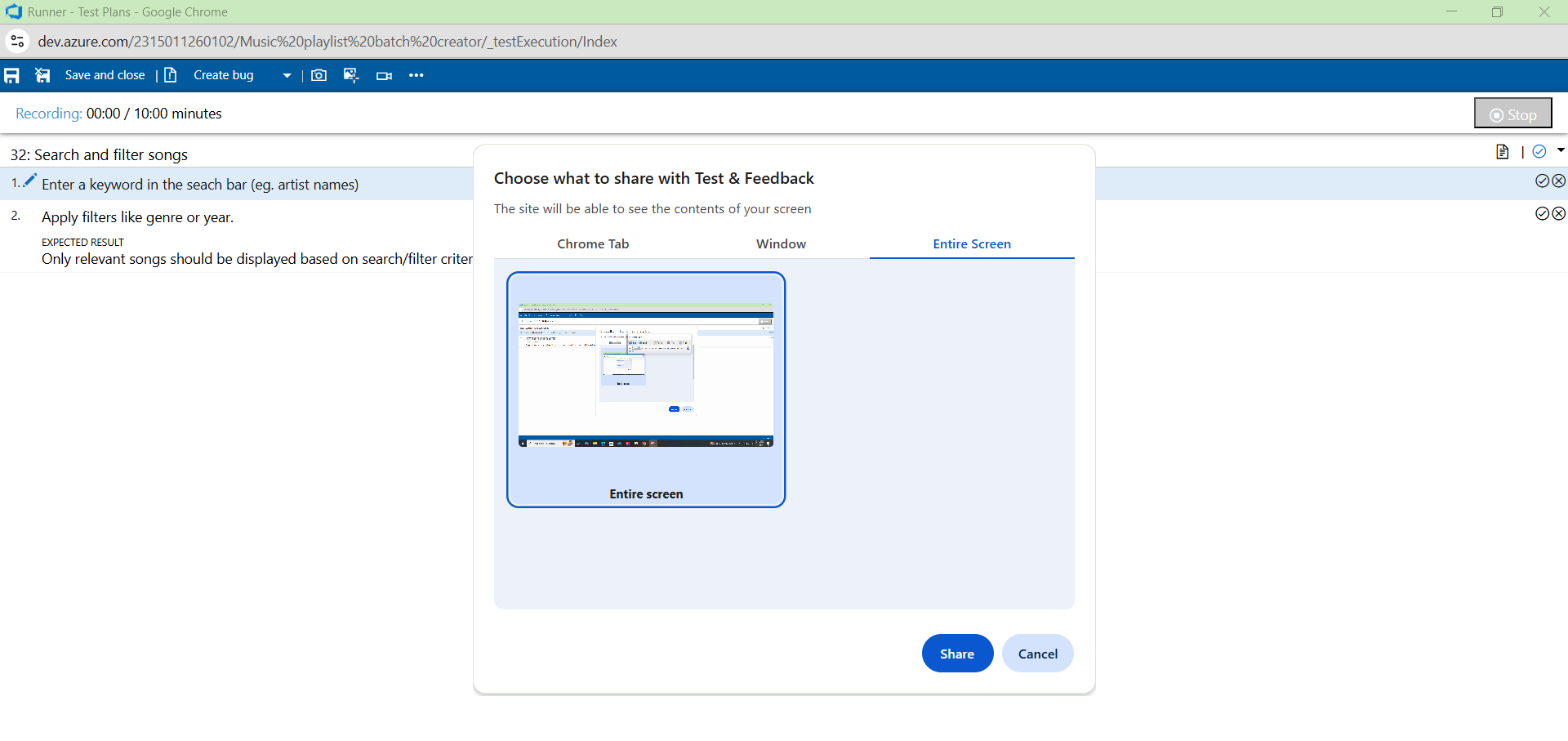


1. **Running the test cases**

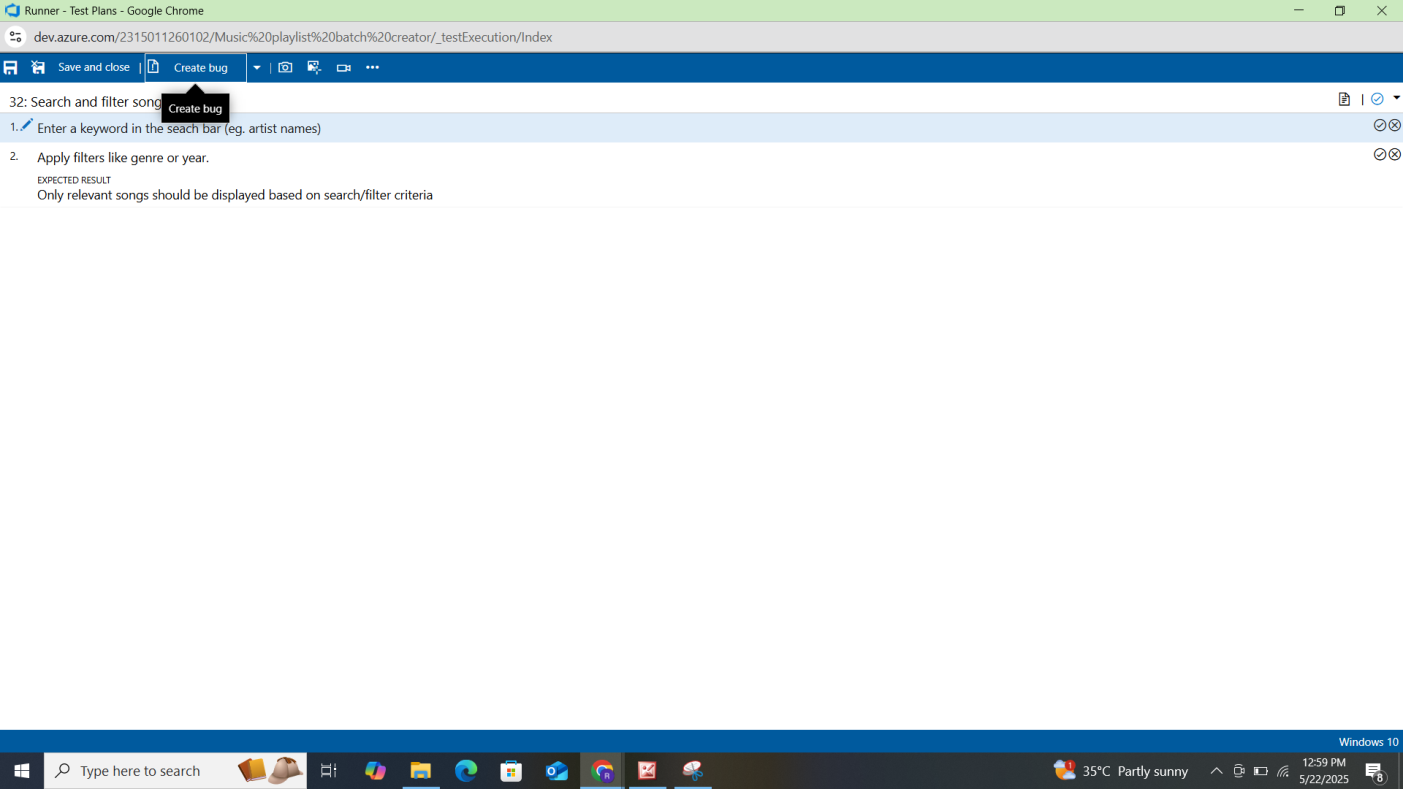
****

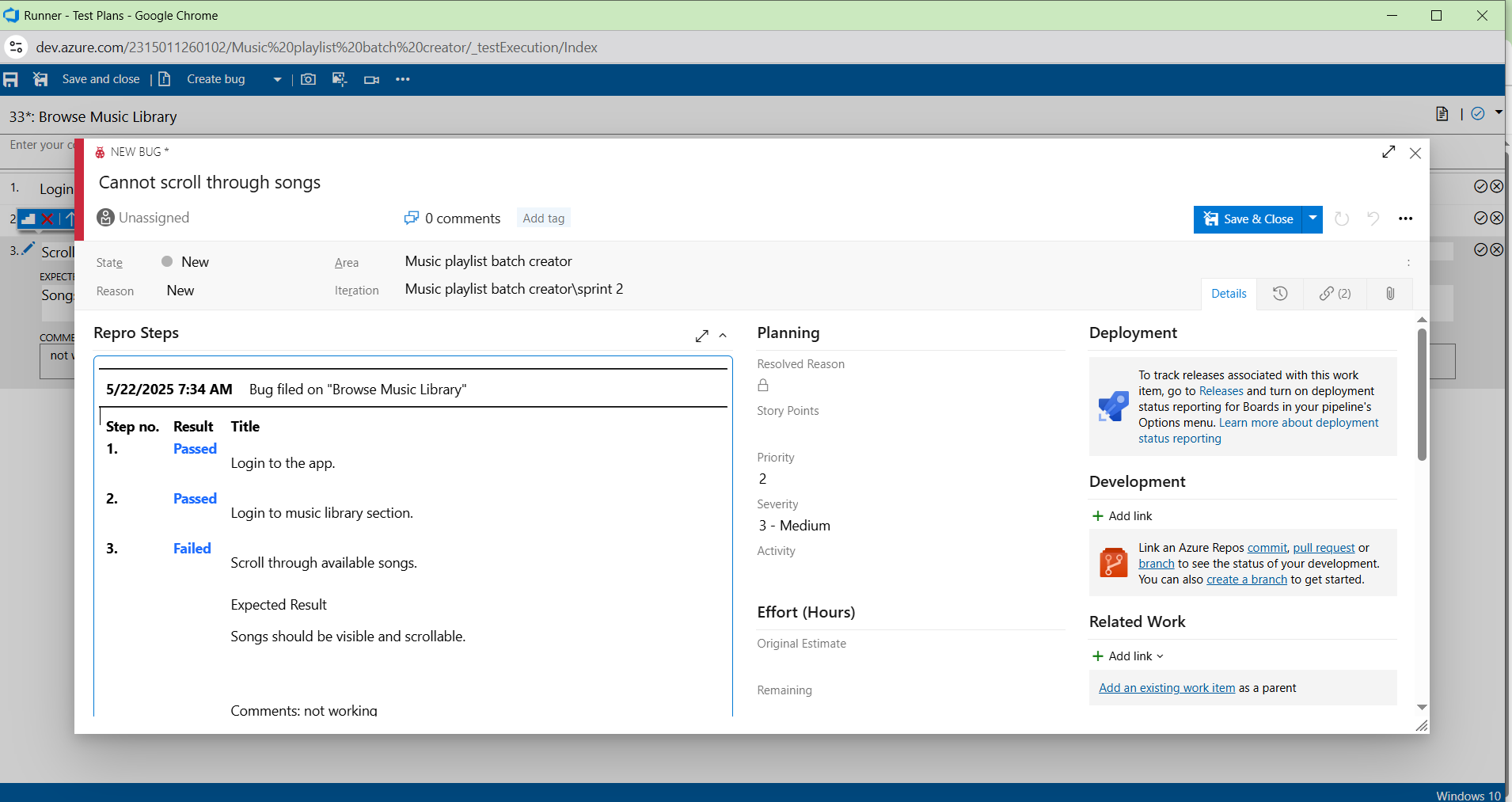
****

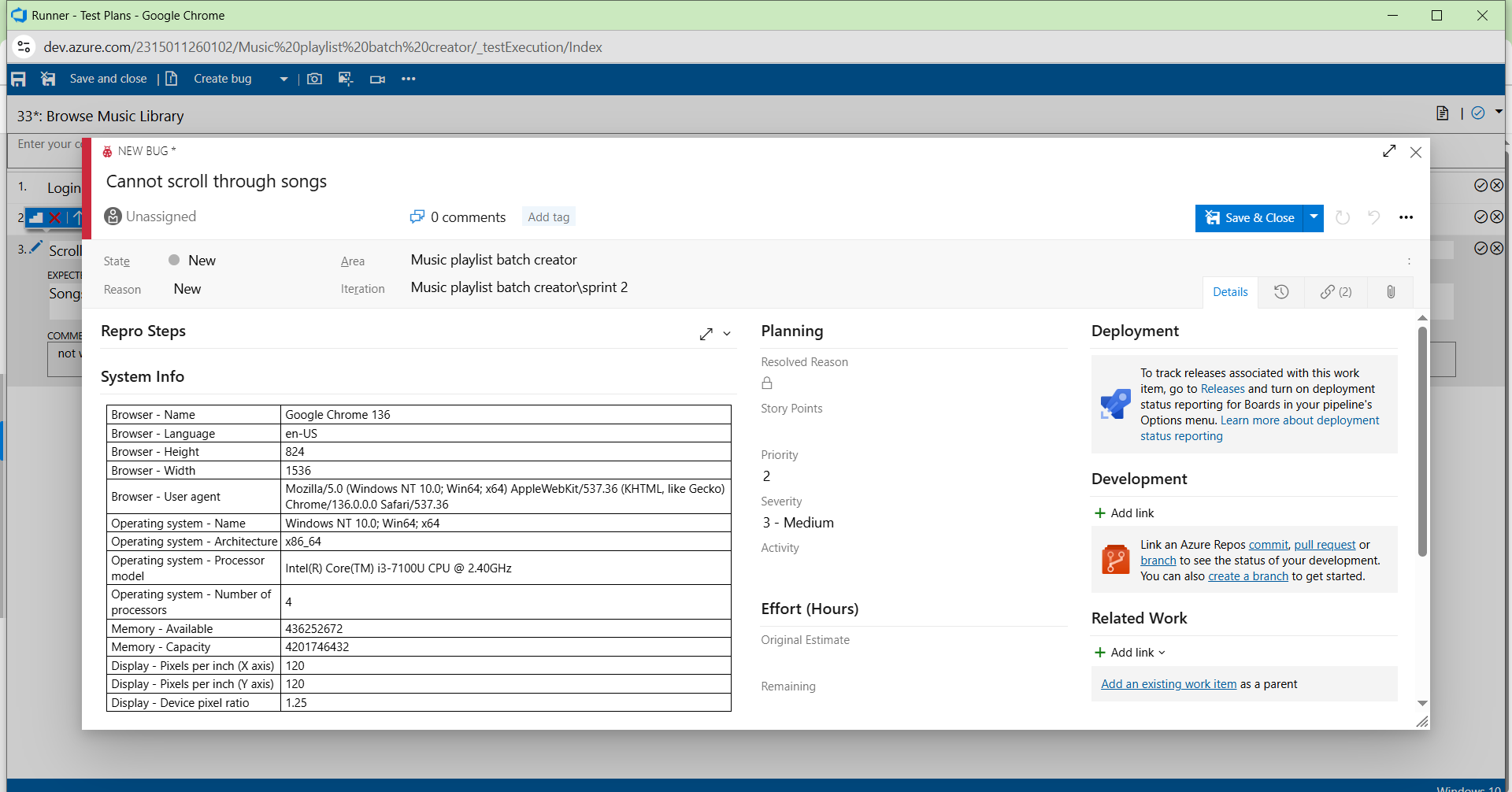
1. **Recording the test case**

****

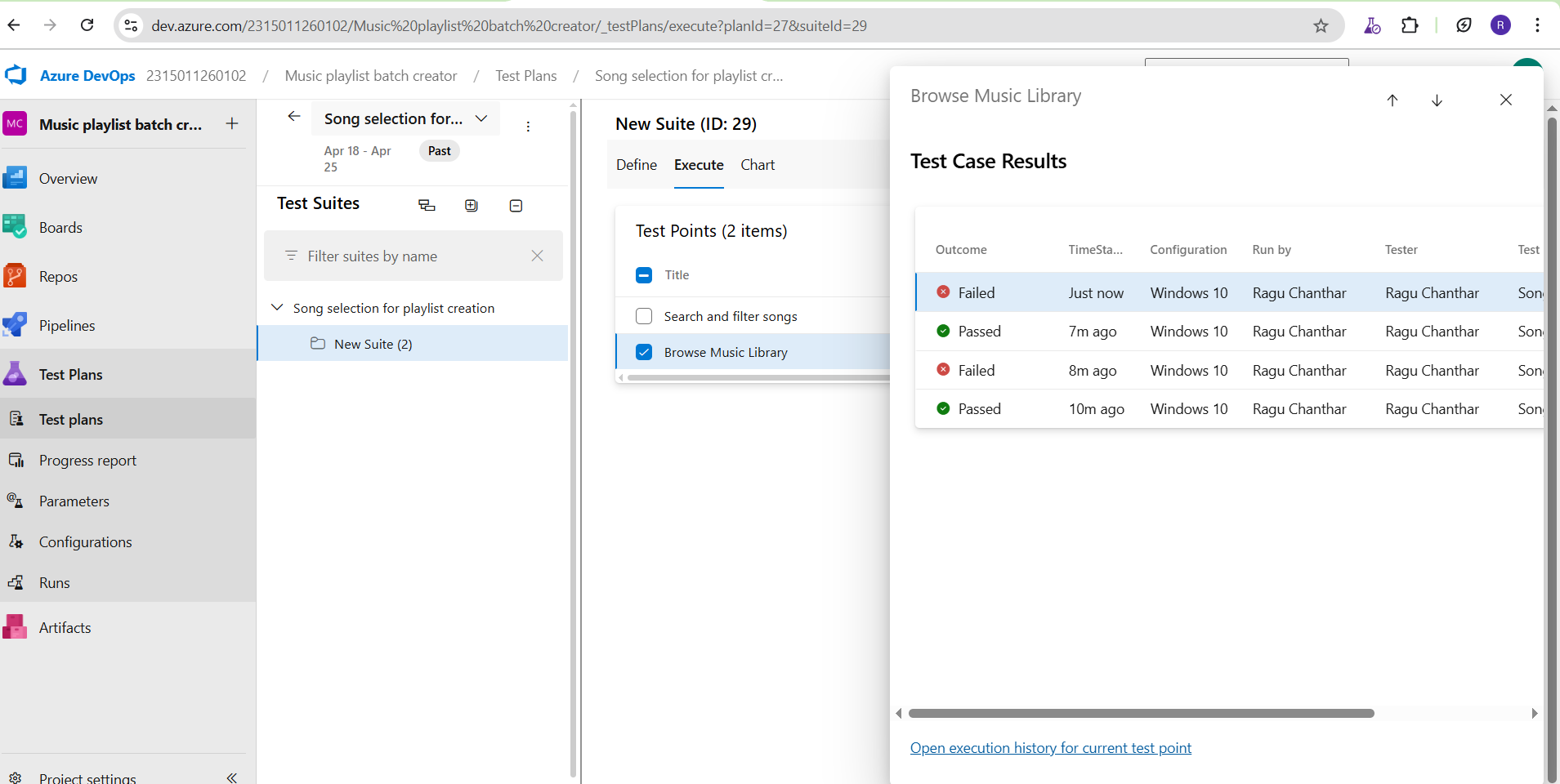
1. **Creating the bug**

****

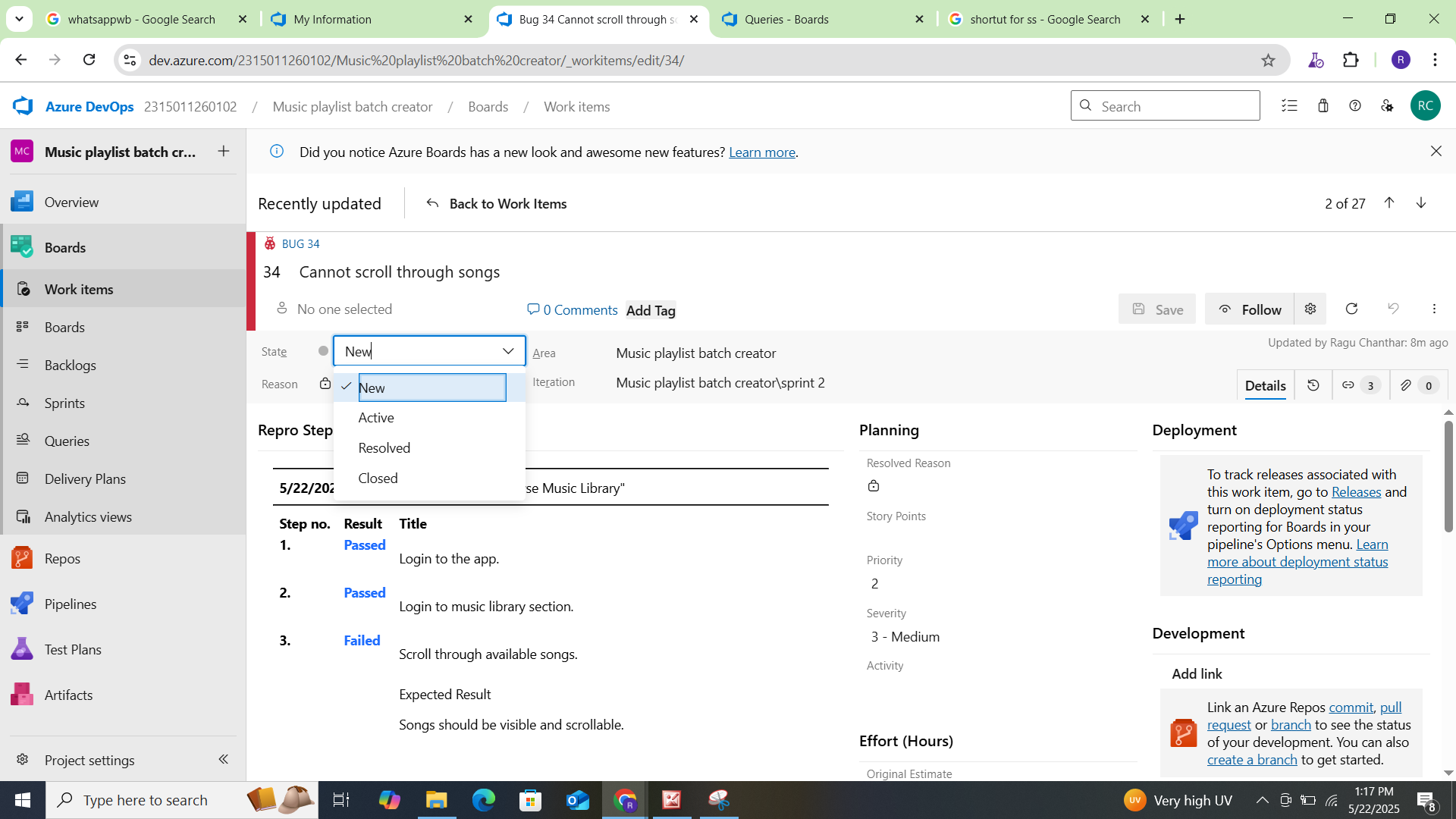




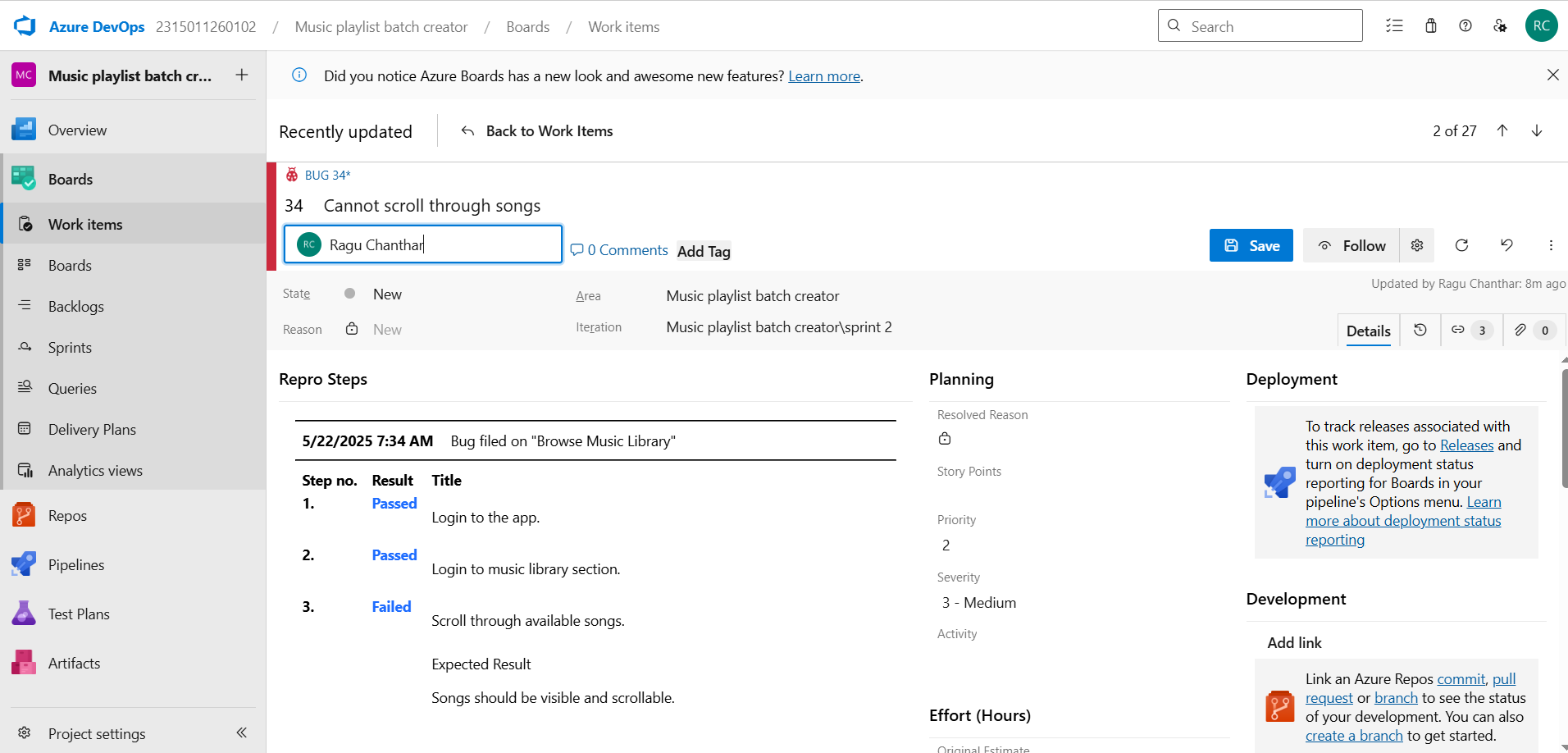
1. **Test case results**

****

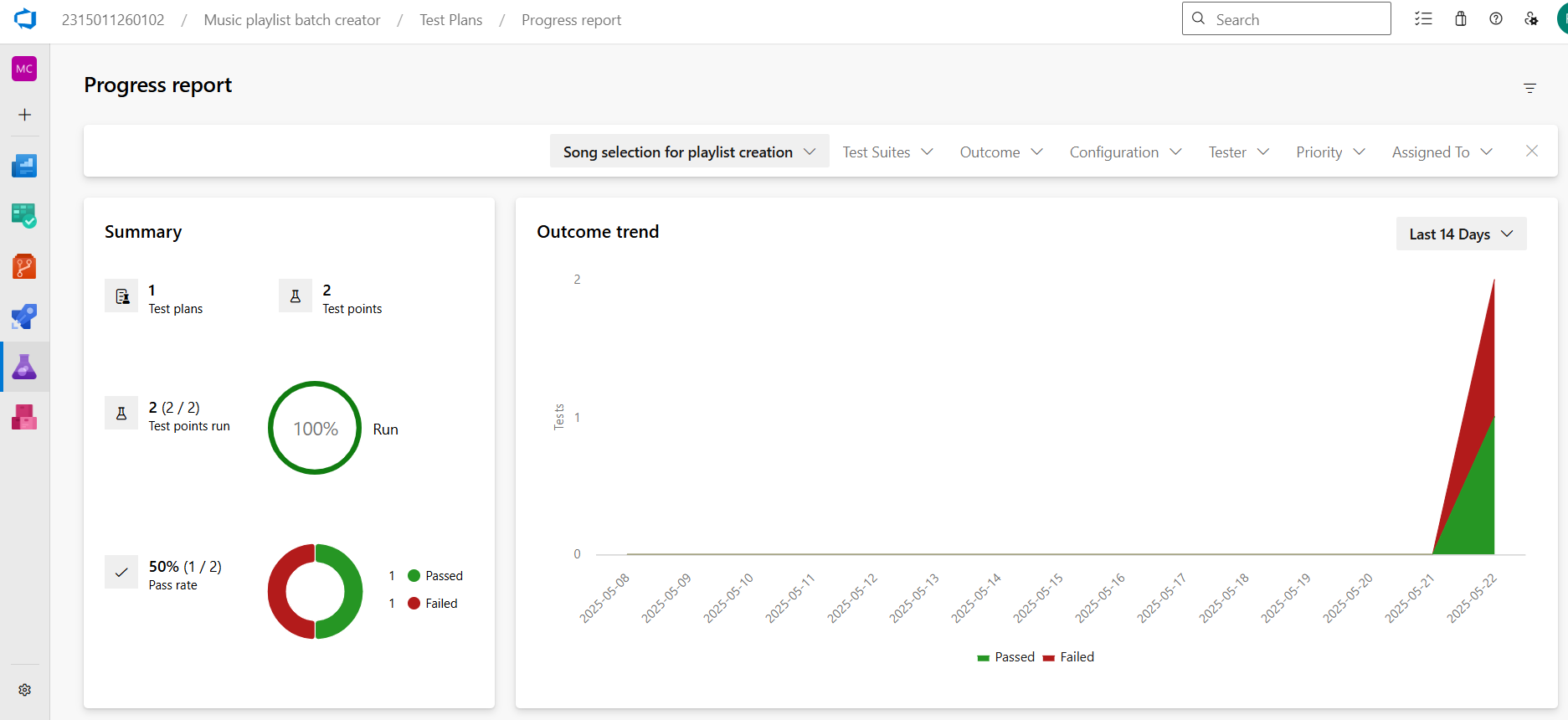
1. **Test report summary**

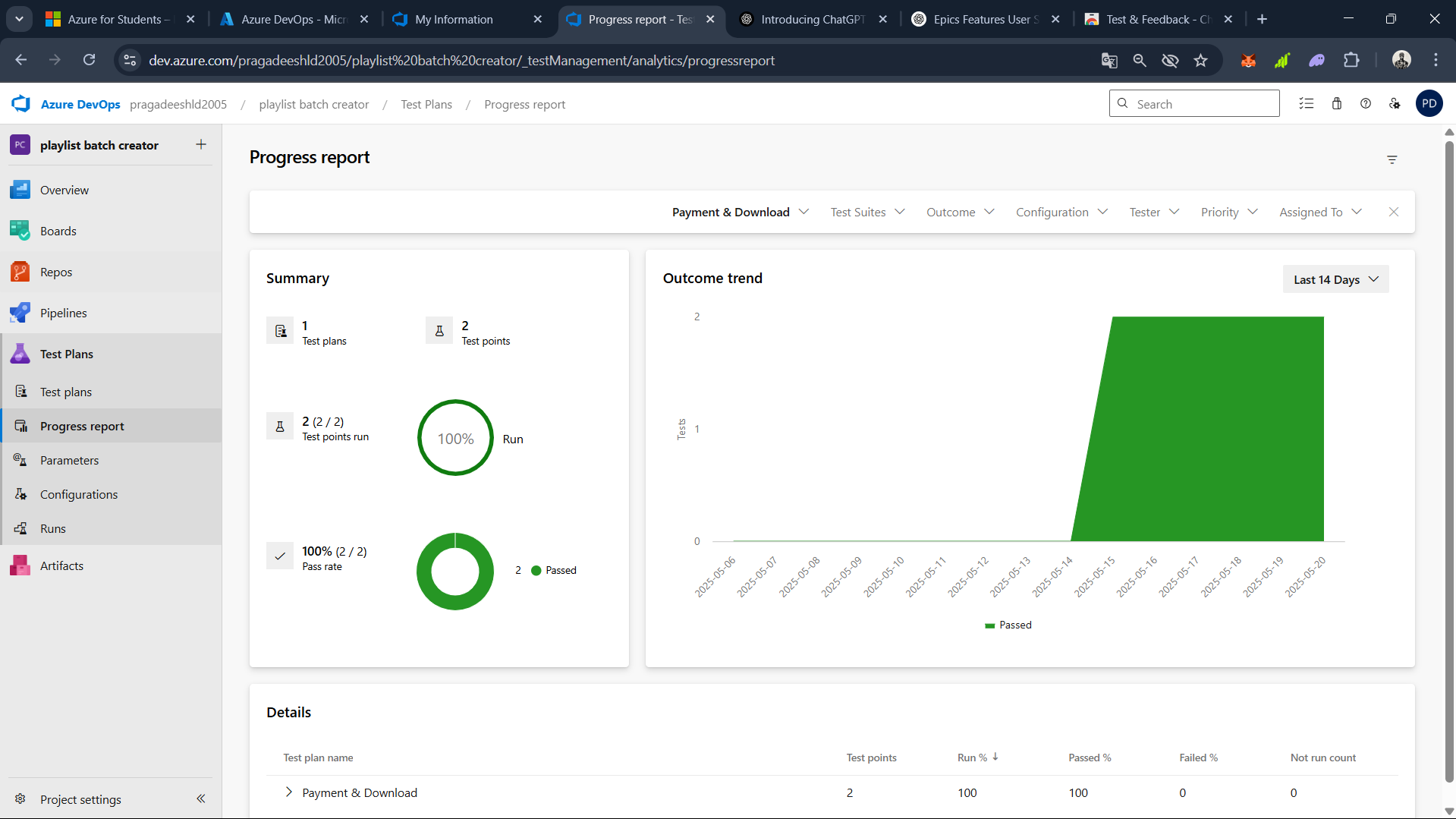
****

* + Assigning bug to the developer and changing state

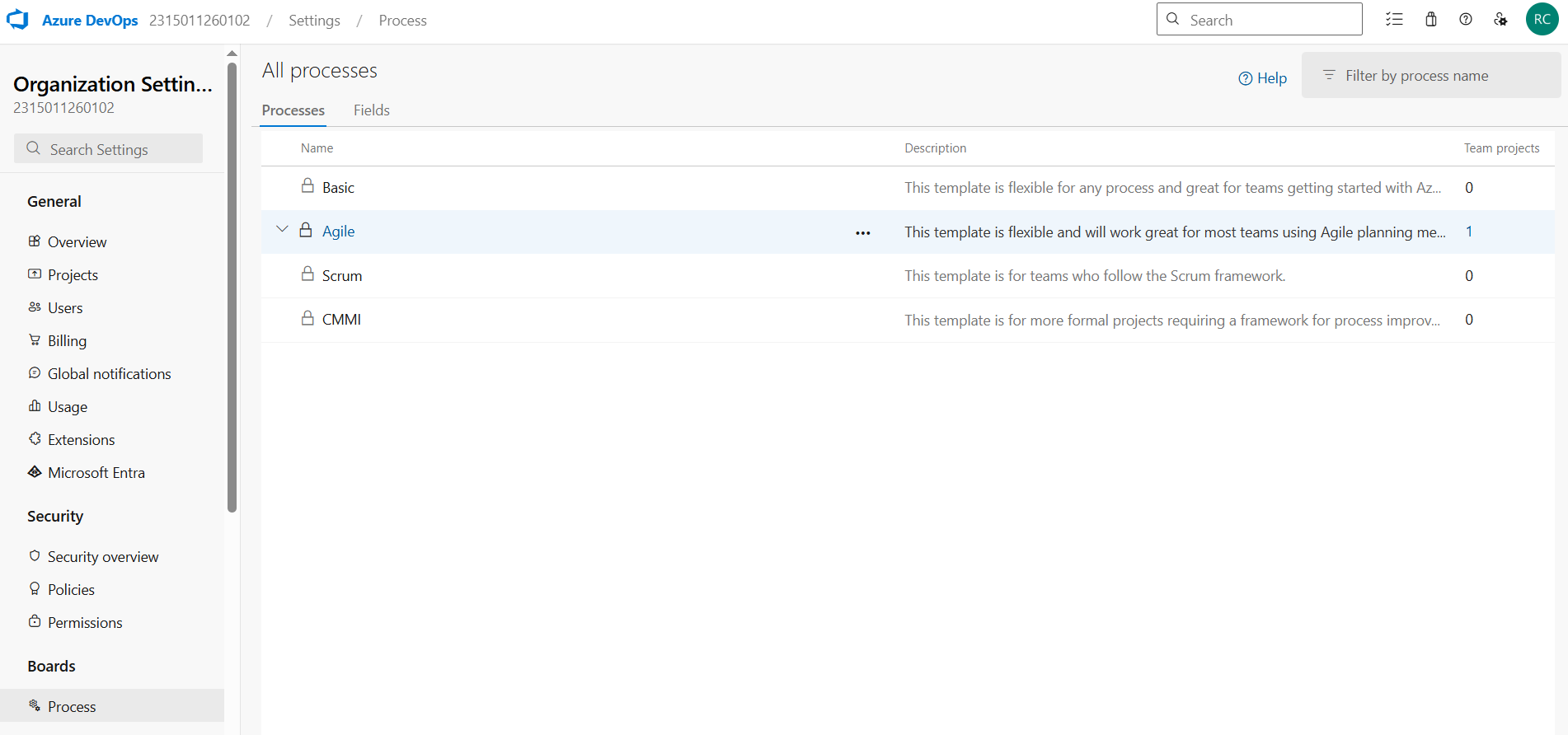


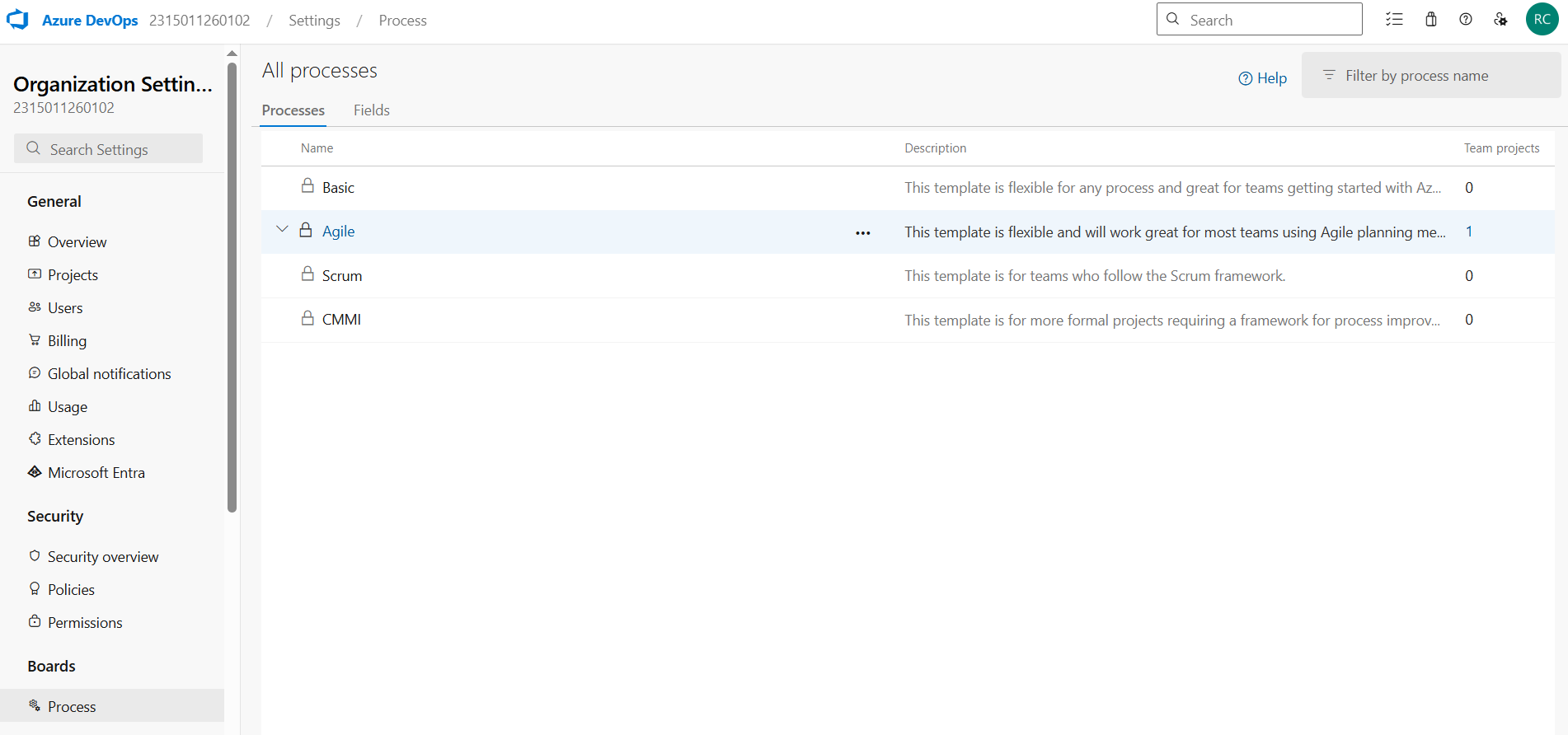
1. **Progress report**

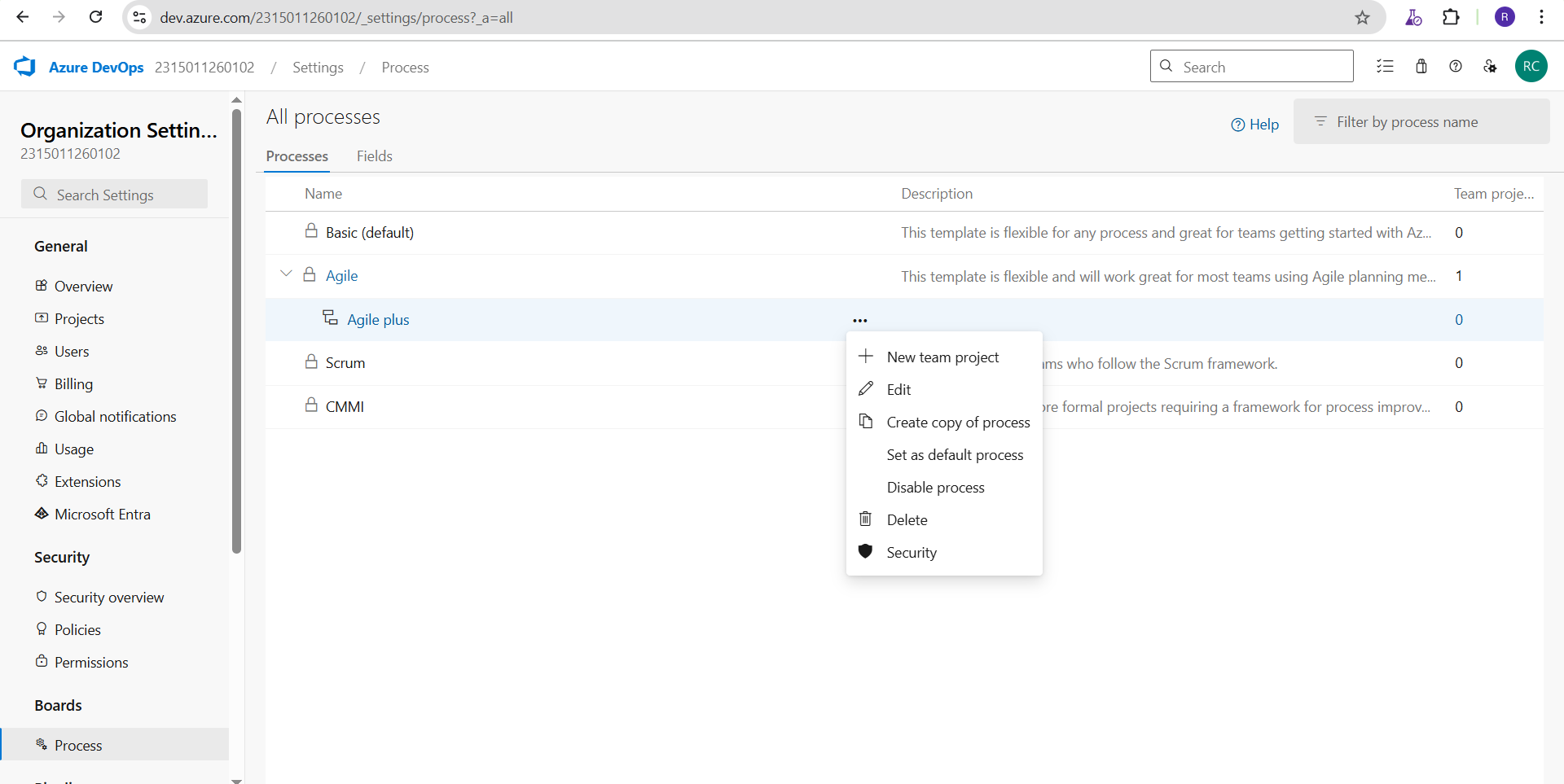
****



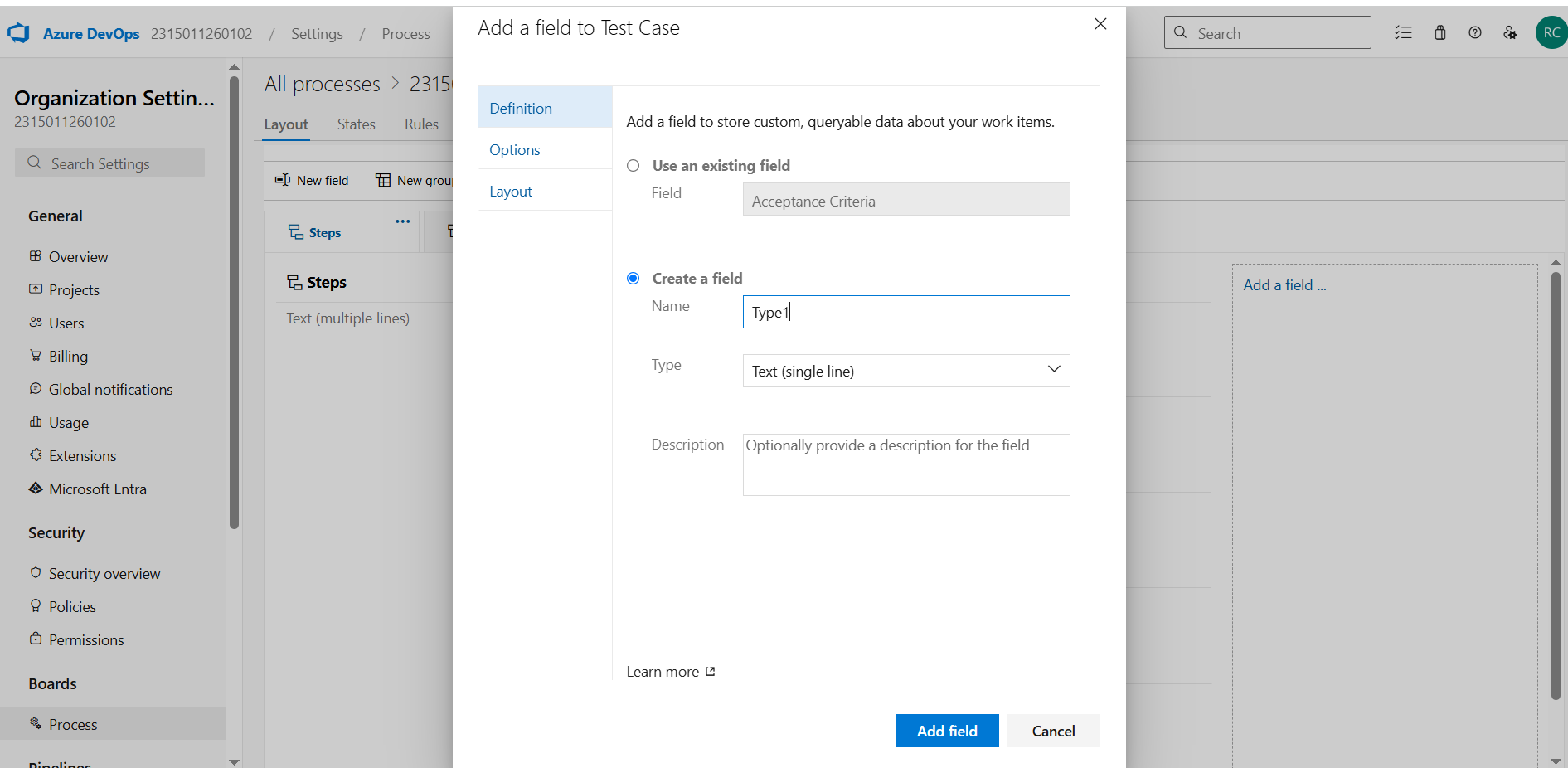
1. **Changing the test template**

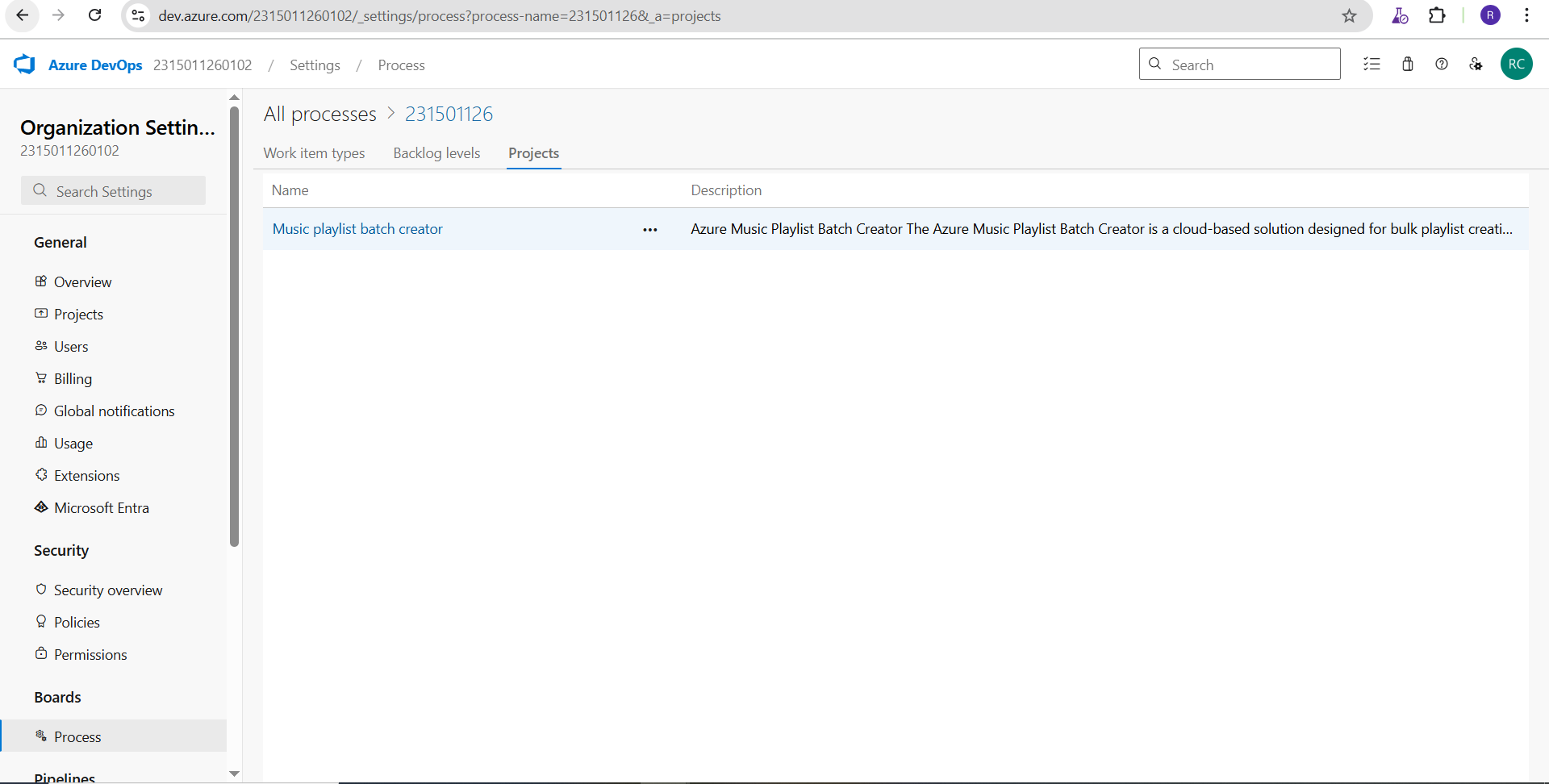
****

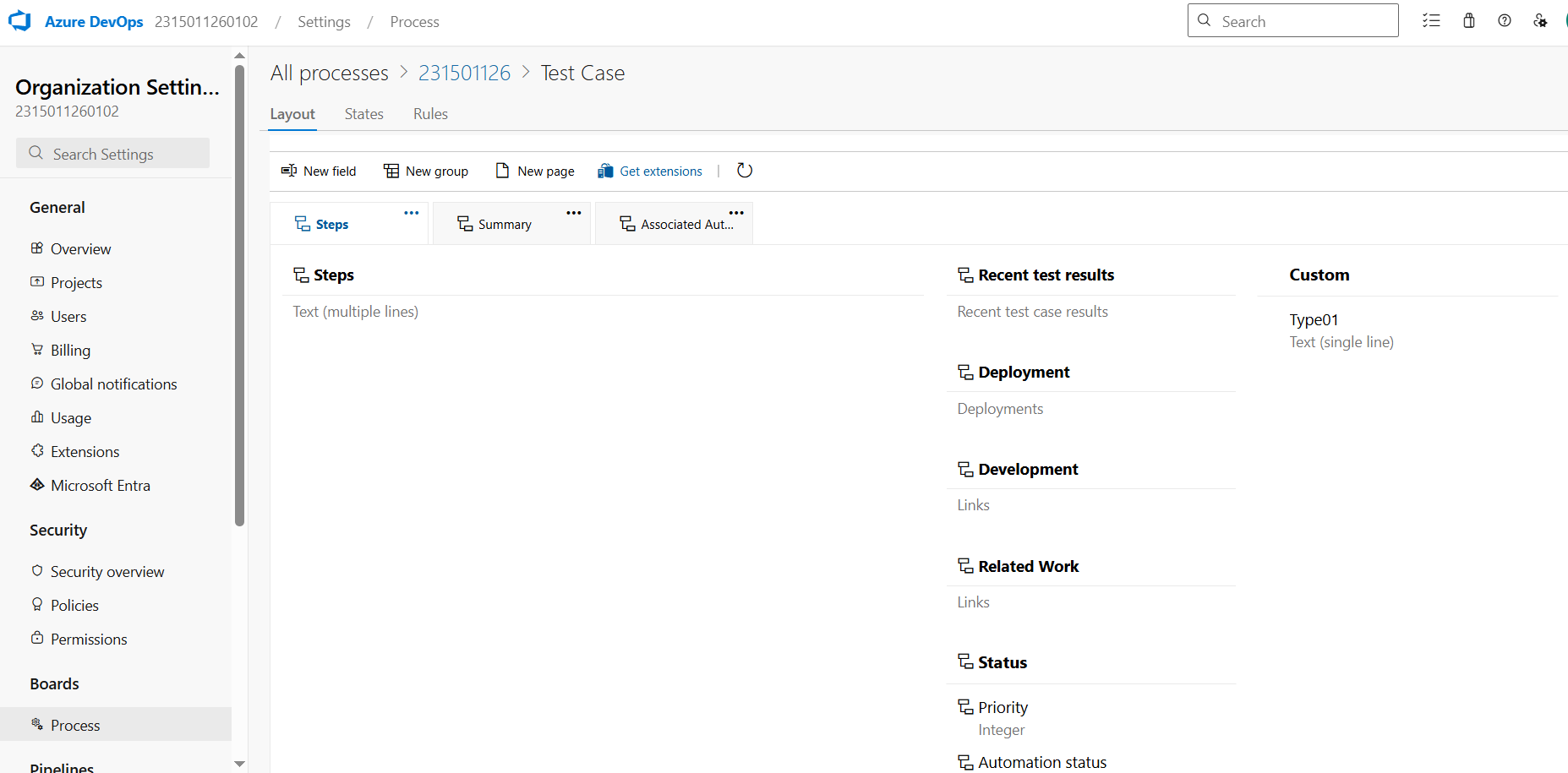


****

1. **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 DATE:** | **Ci/CD PIPELINES IN AZURE** |

# AIM:

To implement a Continuous Integration and Continuous Deployment (CI/CD) pipeline in Azure DevOps for automating the build, testing, and deployment process of the Student Management System, ensuring faster delivery and improved software quality.

# PROCEDURE:

**Steps to Create and implement pipelines in Azure:**

1. Sign in to Azure DevOps and Navigate to Your Project

Log in to [dev.azure.com](https://dev.azure.com/), select your organization, and open the project where your Student Management System code resides.

1. Connect a Code Repository (Azure Repos or GitHub)

Ensure your application code is stored in a Git-based repository such as Azure Repos or GitHub. This will be the source for triggering builds and deployments in your pipeline.

1. Create a New Pipeline

Go to the Pipelines section on the left panel and click “Create Pipeline”.

Choose your source (e.g., Azure Repos Git or GitHub), and then select the repository containing your project code.

1. Choose the Pipeline Configuration

You can select either the YAML-based pipeline (recommended for version control and automation) or the Classic Editor for a GUI-based setup.

If using YAML, Azure DevOps will suggest a template or allow you to define your own.

1. Define Build Stage (CI - Continuous Integration) from YAML file
2. Install dependencies (e.g., npm install, dotnet restore)
3. Build the application (dotnet build, npm run build)
4. Run unit tests (dotnet test, npm test)
5. Publish build artifacts to be used in the release stage
6. Save and Run the Pipeline for the First Time

Save the YAML or build definition and click “Run”.

Azure will fetch the latest code and execute the defined build and test stages.

1. Configure Continuous Deployment (CD)

Navigate to the Releases tab under Pipelines and click “New Release Pipeline”. Add an Artifact (from the build stage) and create a new Stage (e.g., Development, Production).

1. Configure the CD stage with deployment tasks such as deploying to Azure App Service, running database migrations or scripts, and restarting services using the Azure App Service Deploy task linked to your subscription and app details.
2. Set Triggers and Approvals

Enable continuous deployment trigger so the release pipeline runs automatically after a successful build.

For production environments, configure pre-deployment approvals to ensure manual verification before release.

1. Monitor Pipelines and Manage Logs

View all pipeline runs under the Runs section.

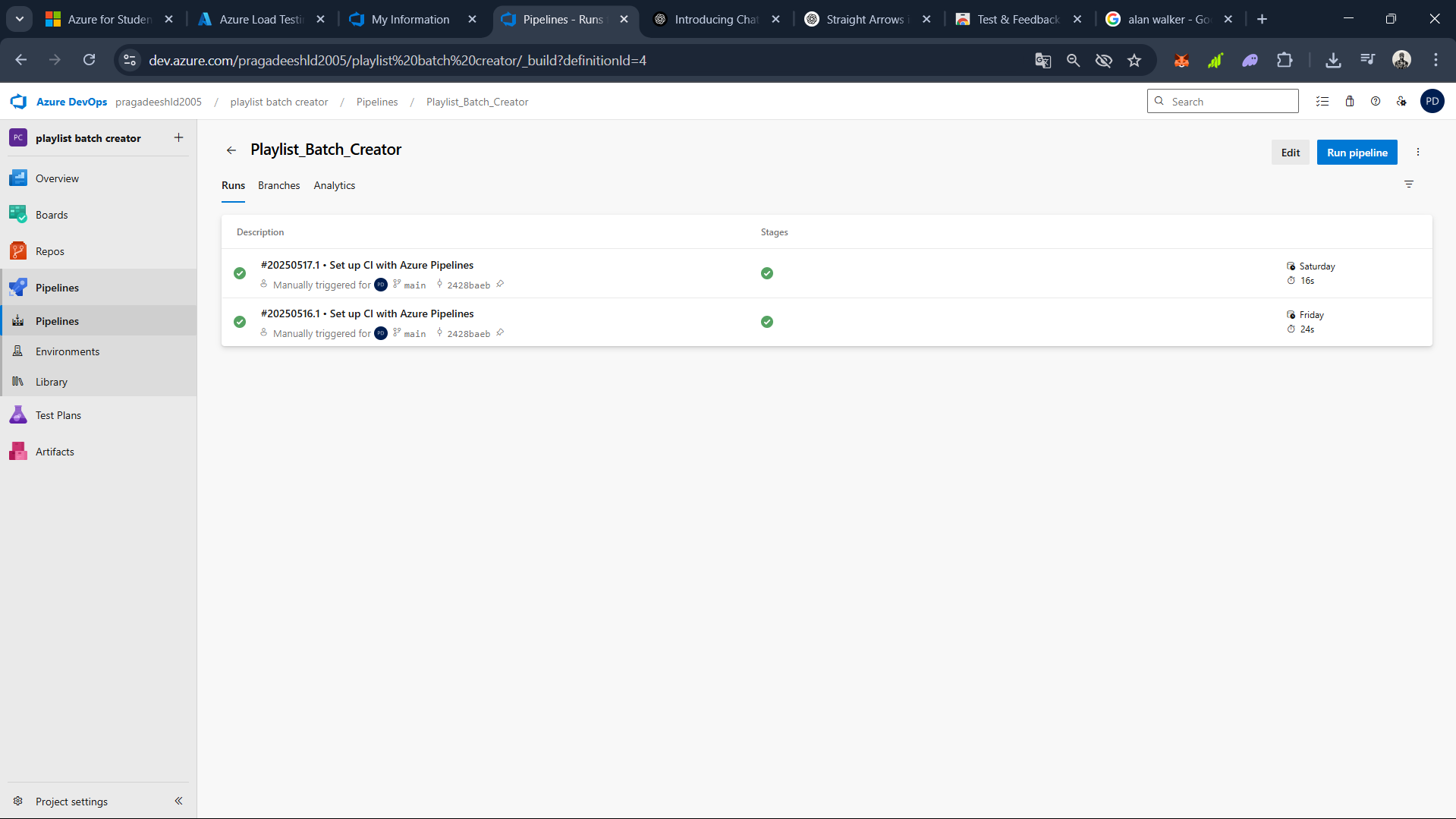
Check logs for build/test/deploy stages to debug any errors.

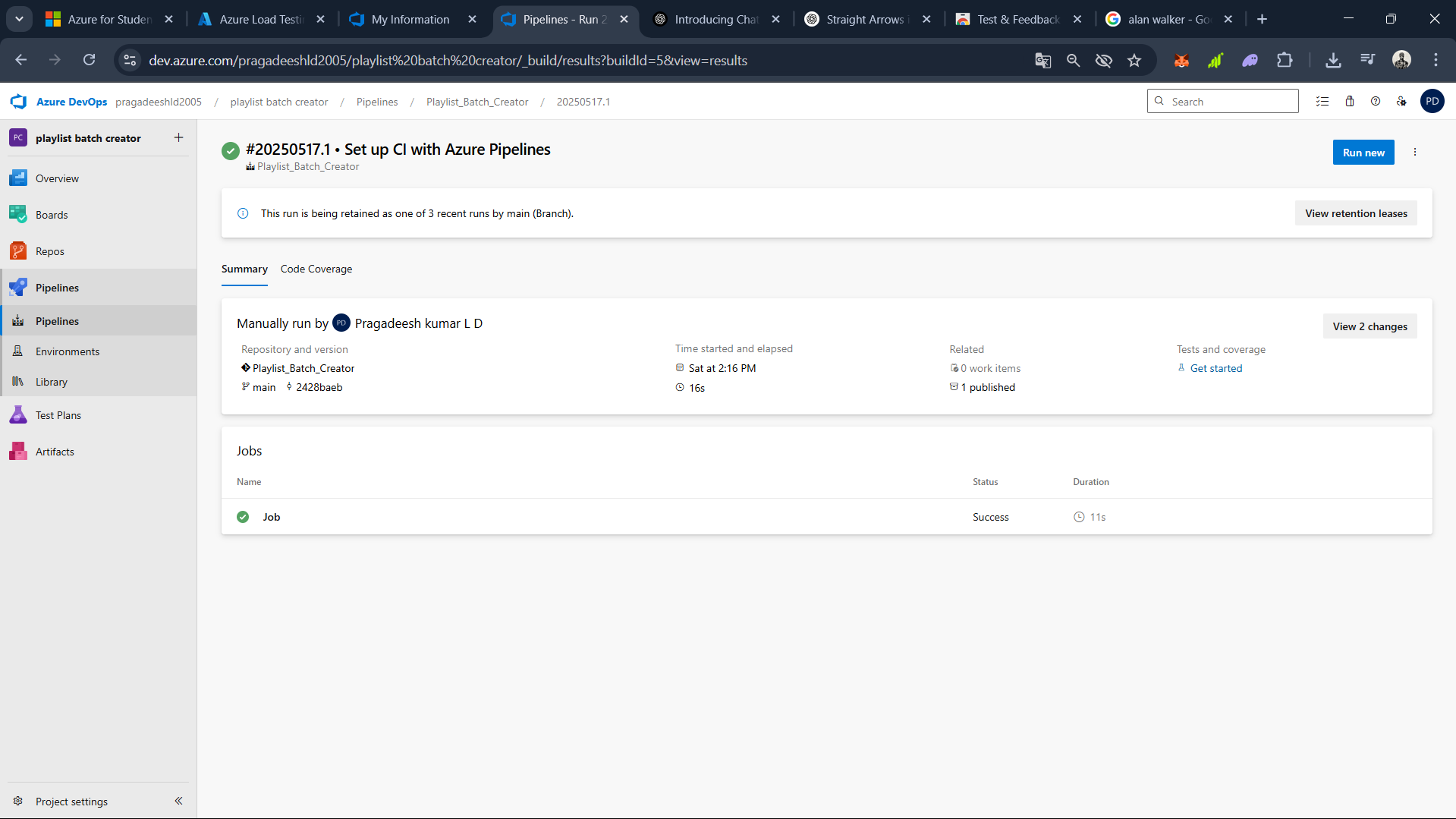
You can also integrate email alerts or Microsoft Teams notifications for build failures.

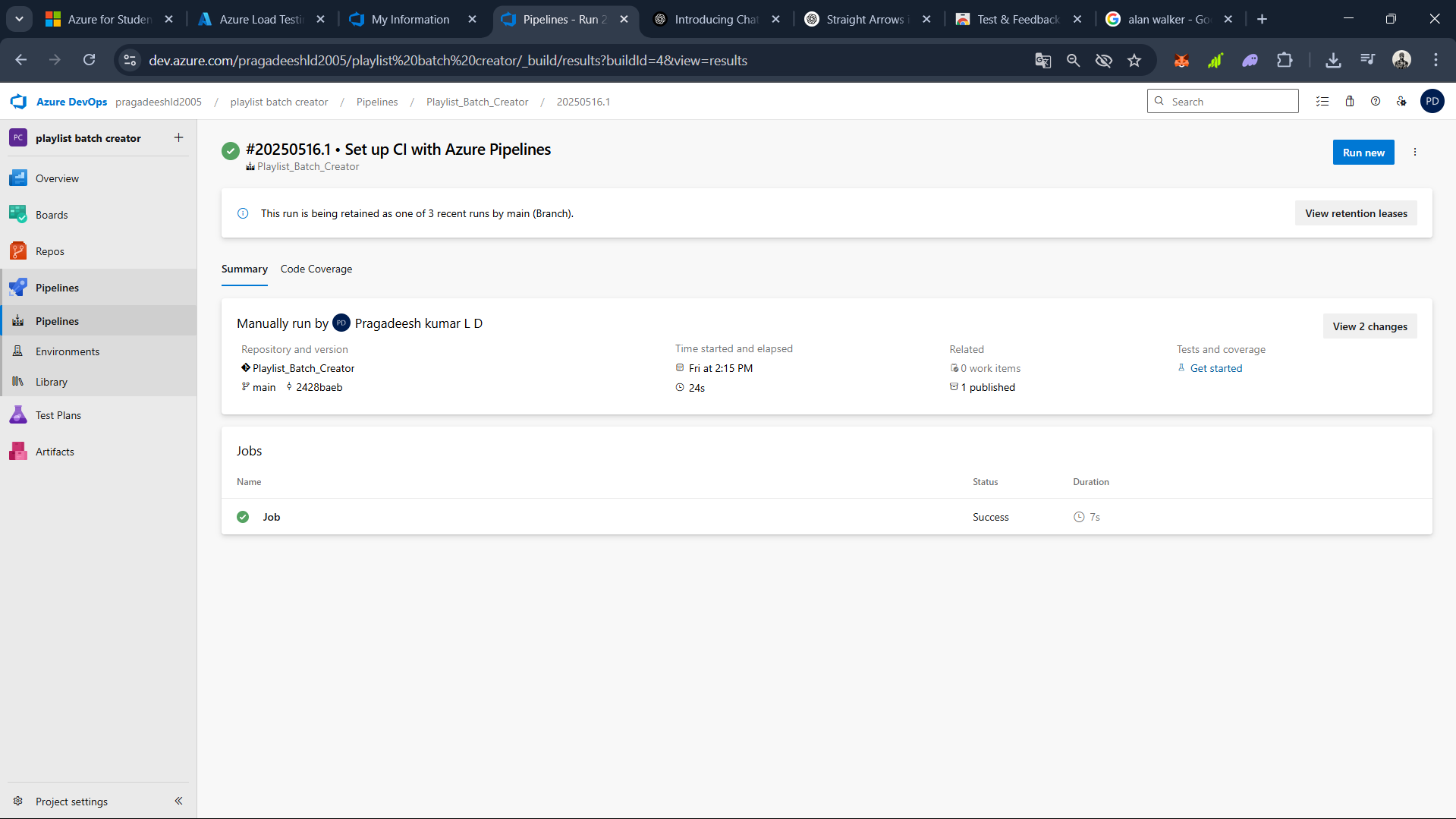
1. Review and Maintain Pipelines

Regularly update your pipeline tasks or YAML configurations as your application grows. Ensure pipeline runs are clean and artifacts are stored securely.

Integrate quality gates and code coverage policies to maintain code quality.







# RESULT:

Thus the pipelines for the given project “Student Management System has been executed successfully

**GITHUB: PROJECT STRUCTURE & NAMING CONVENTIONS**

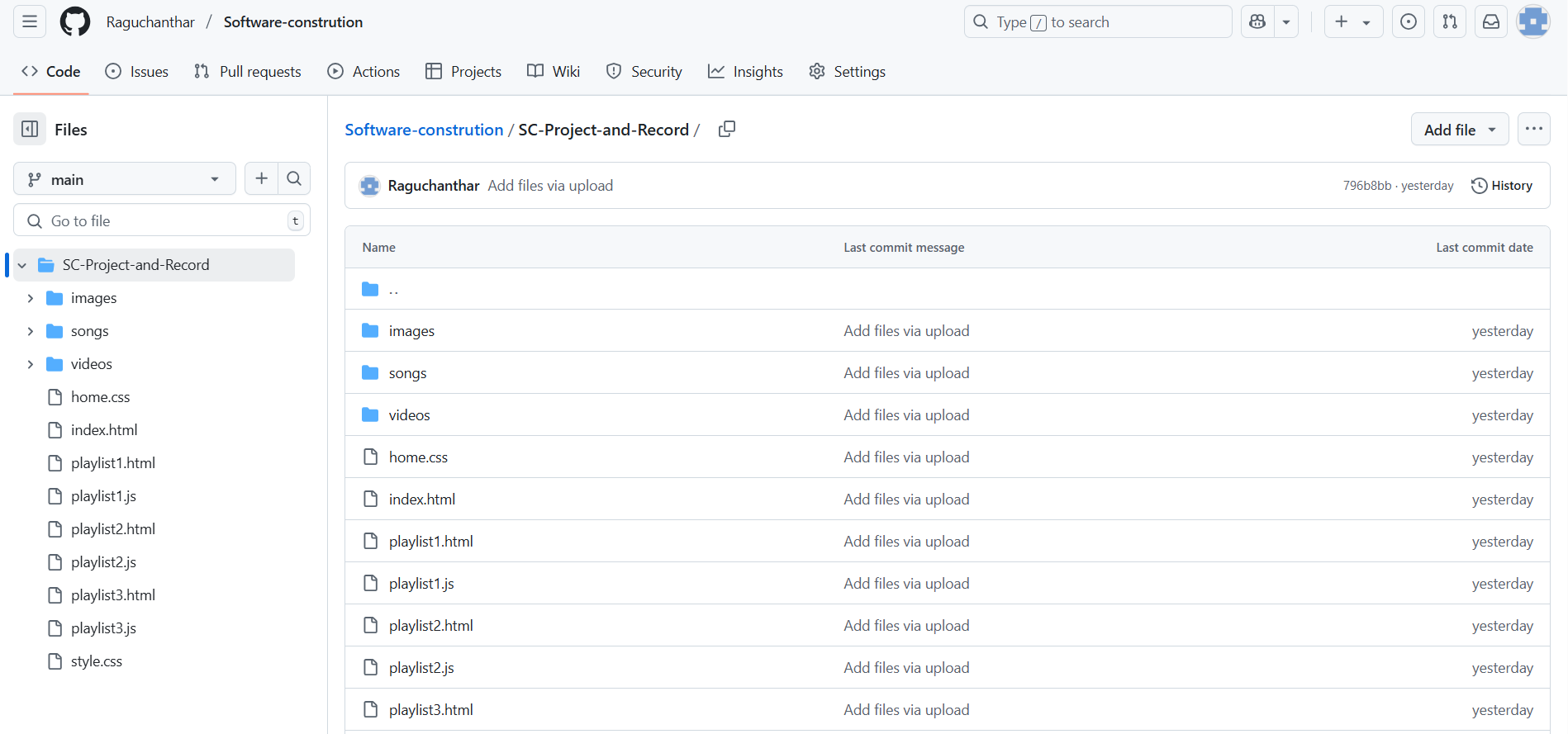
**EXP NO: 10**

**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 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