

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

**CS23432 – Software Construction**

**(REGULATION 2023)**

**RAJALAKSHMI ENGINEERING COLLEGE**

**Thandalam, Chennai-602015**

Name: BALAJI R

Register No: 231801018

Year / Branch / Section: 2nd / AI&DS / AC 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. |

**AZURE DEVOPS ENVIRONMENT SETUP**

EXP NO: 1

**Aim:**

Create Epic, Features, User Stories, Task

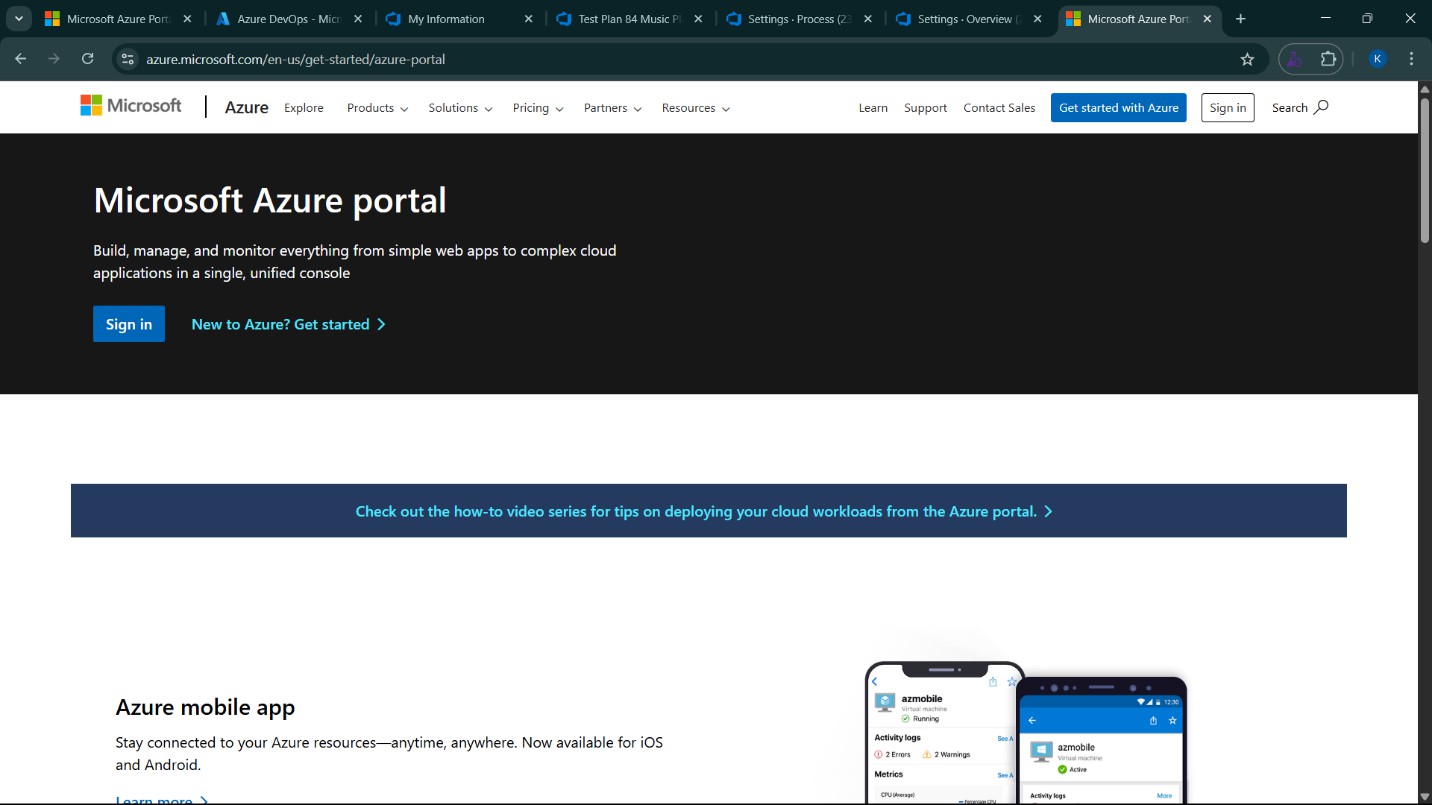
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

**INSTALLATION**

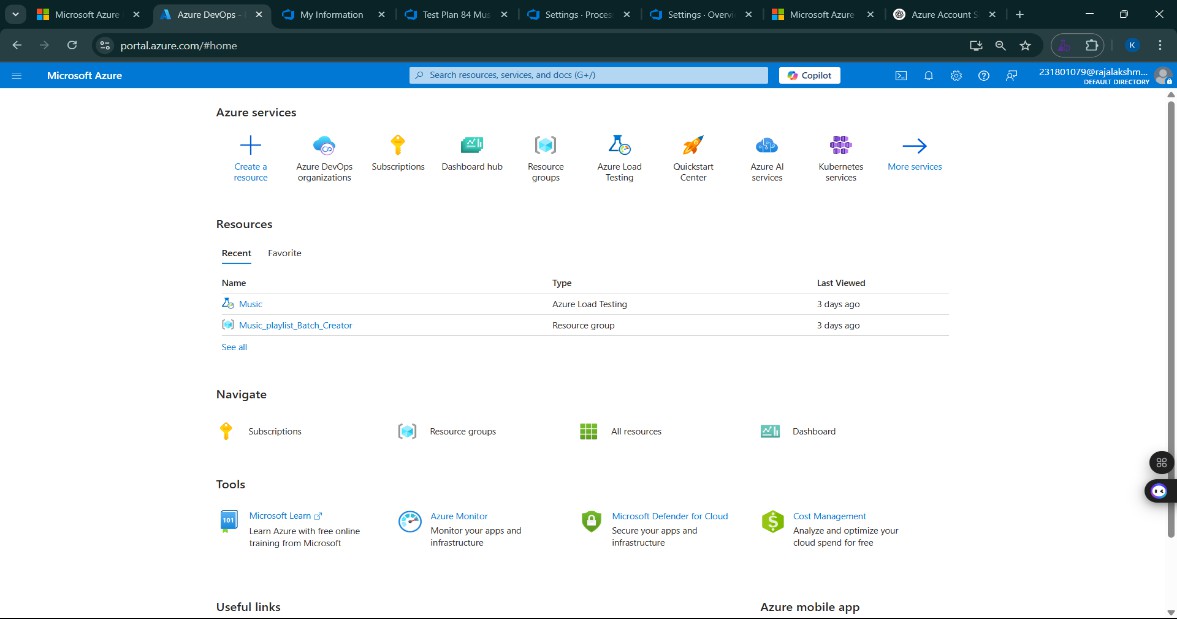
1. Open your web browser and go to the Azure website: [https://azure.microsoft.com/en-us/get-](https://azure.microsoft.com/en-us/get-started/azure-portal) [started/azure-portal](https://azure.microsoft.com/en-us/get-started/azure-portal).

Sign in using your Microsoft account credentials.

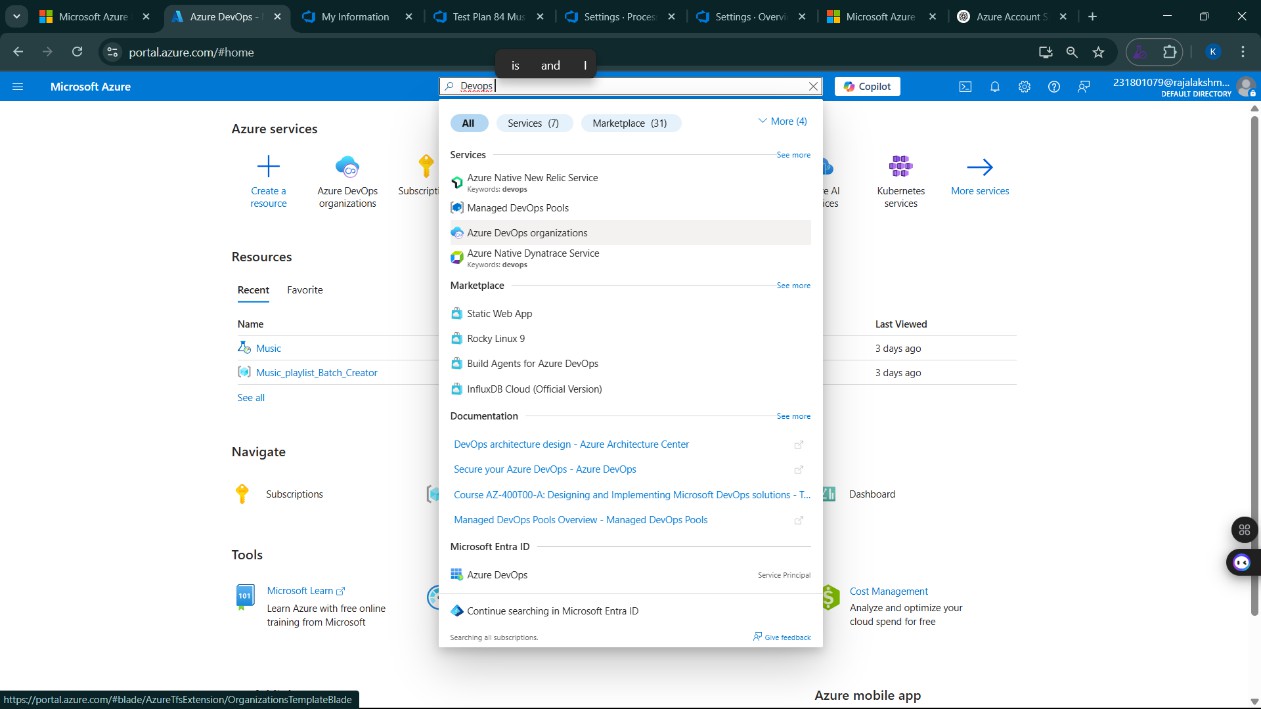
If you don't have a Microsoft account, you can create one here: [https://signup.live.com/?lic=](https://signup.live.com/?lic=1)1



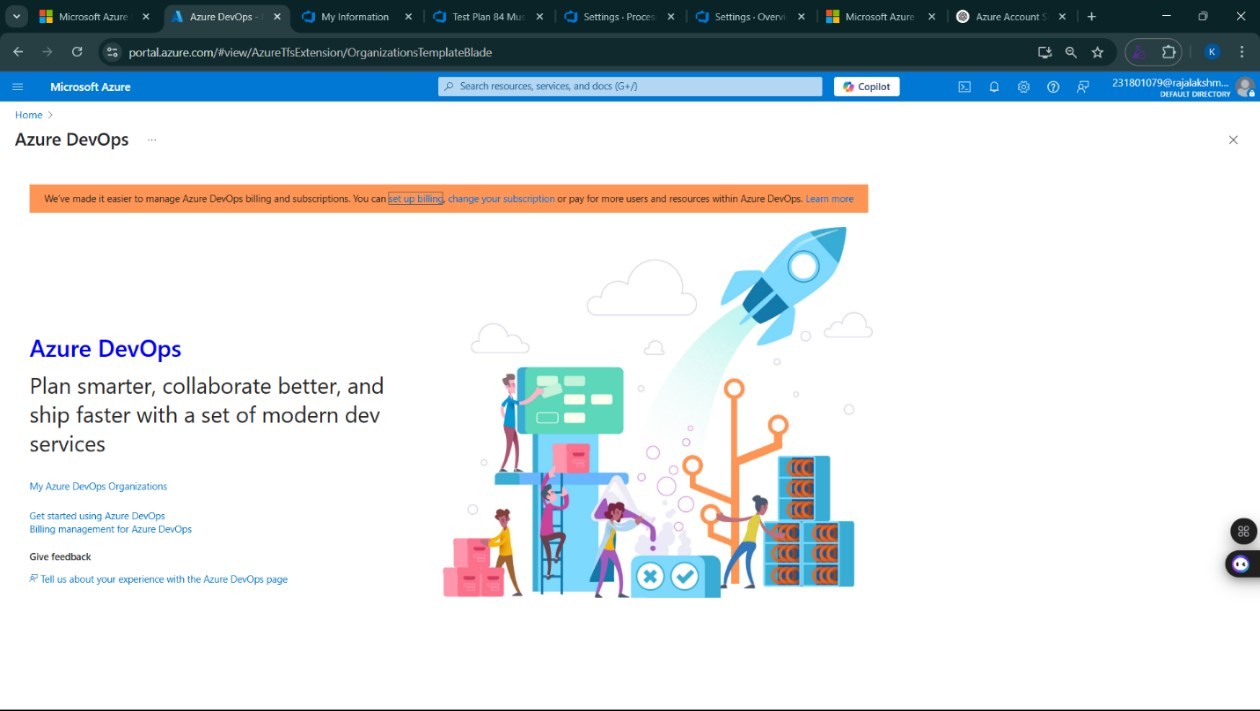
1. Azure home page



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**

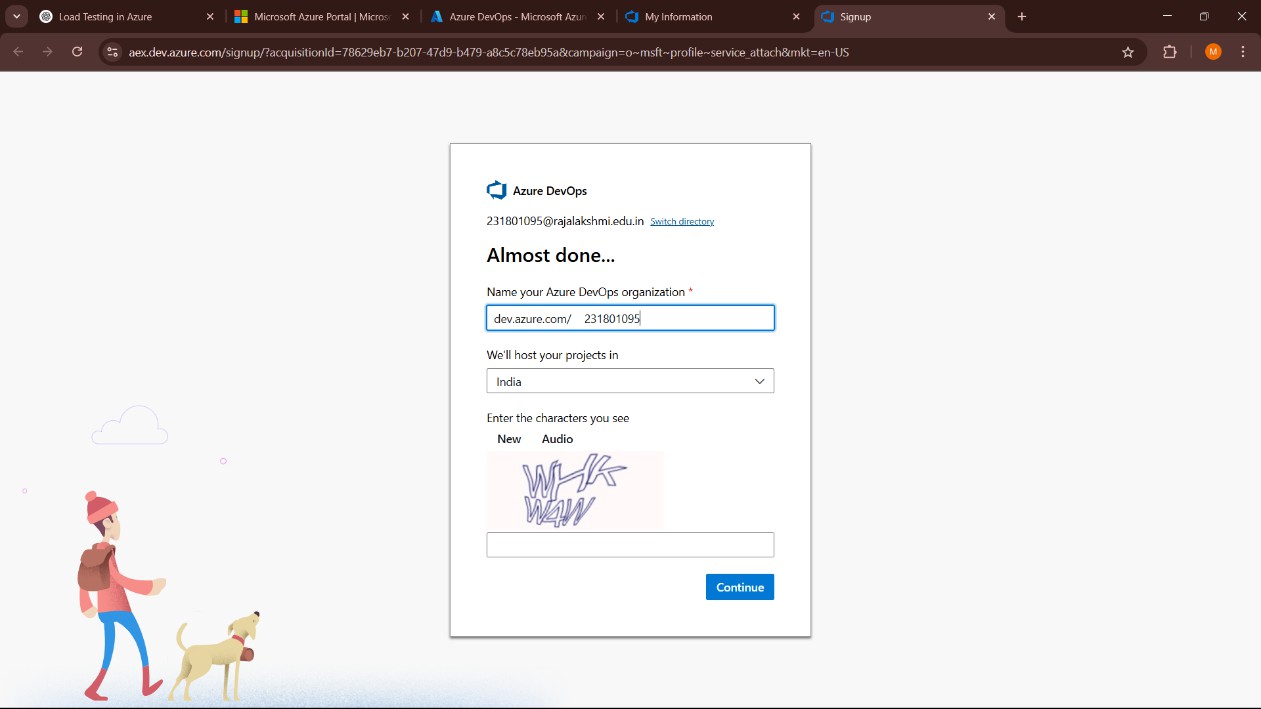
**EXP NO: 2**

**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.

A screenshot of a computer

AI-generated content may be incorrect.

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.

A screenshot of a computer

AI-generated content may be incorrect.

1. Project dashboard

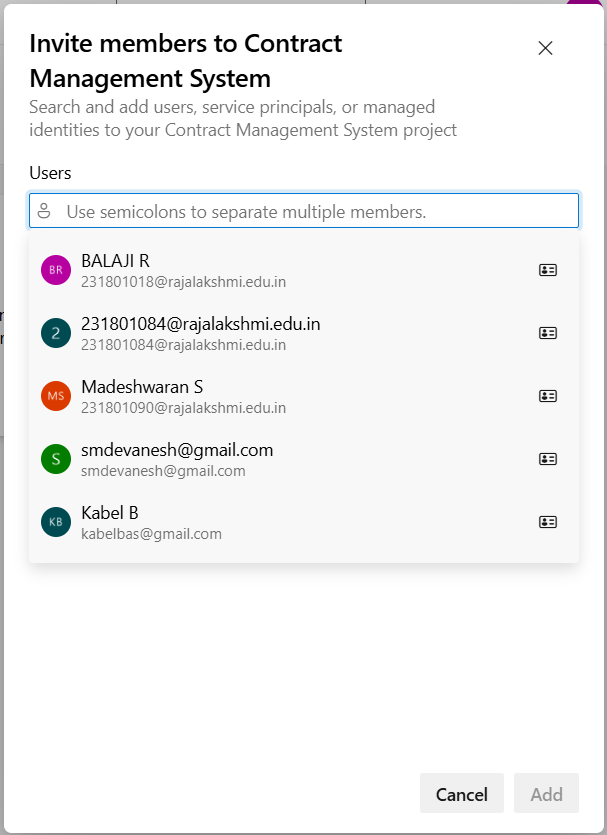
A screenshot of a computer

AI-generated content may be incorrect.

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

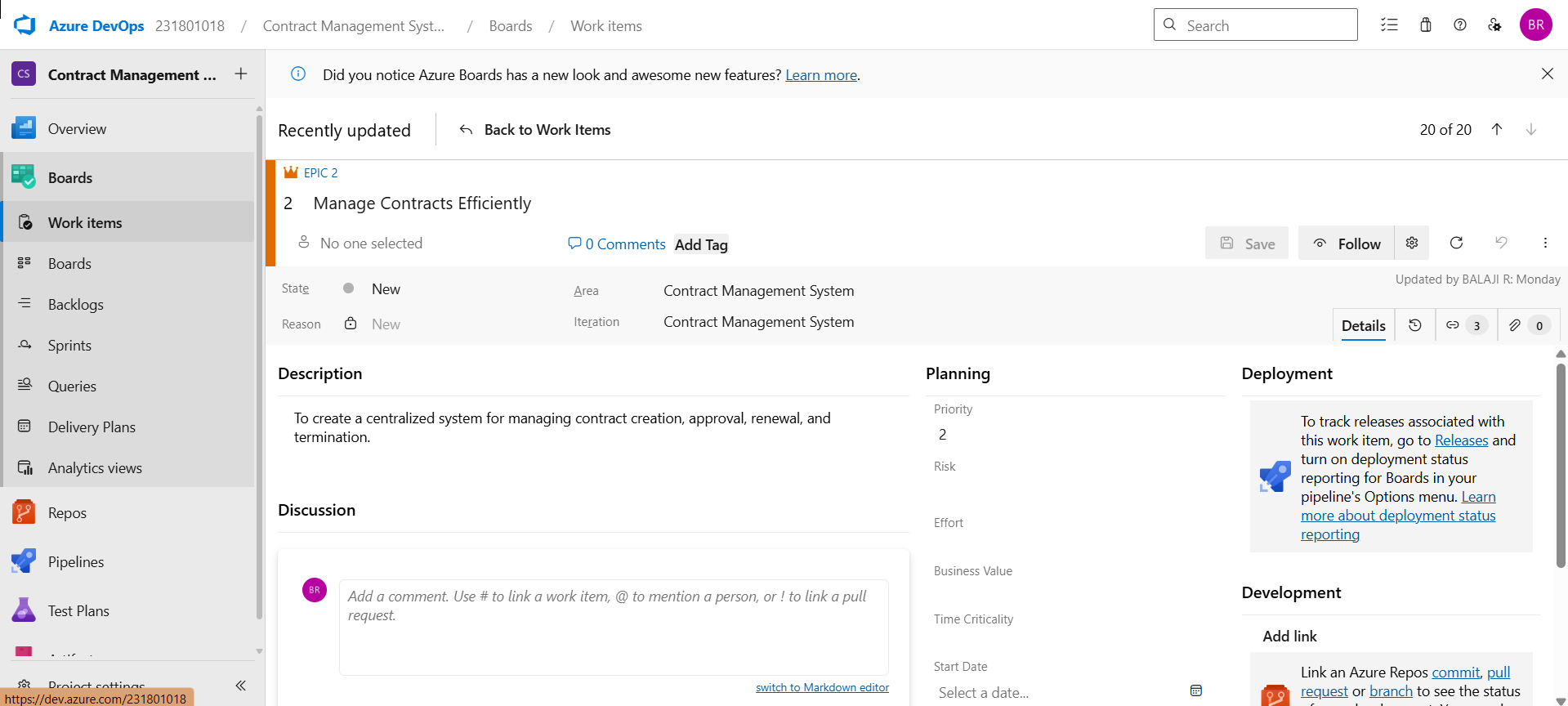
To learn about how to create epics, user story, features, backlogs for your assigned project.

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

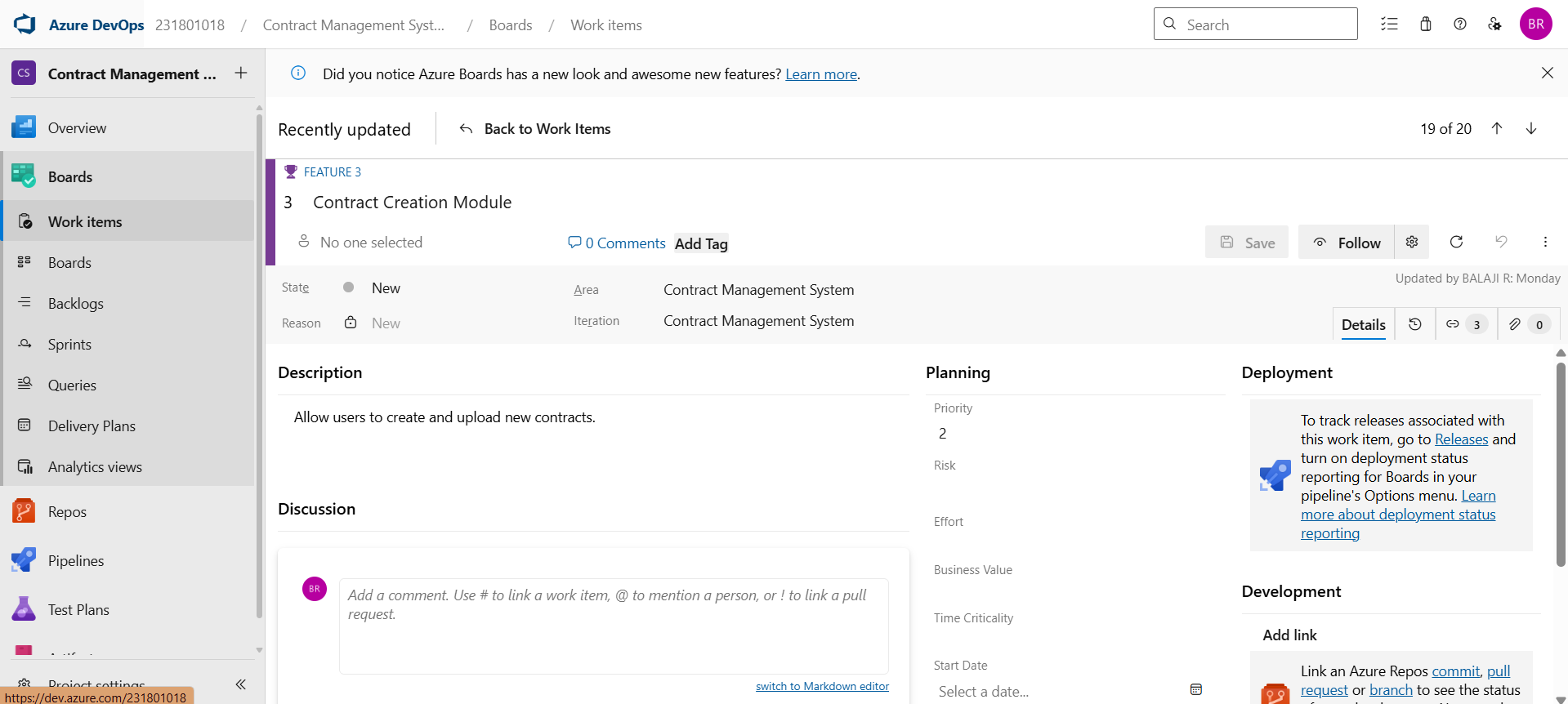
**A screenshot of a computer

AI-generated content may be incorrect.**

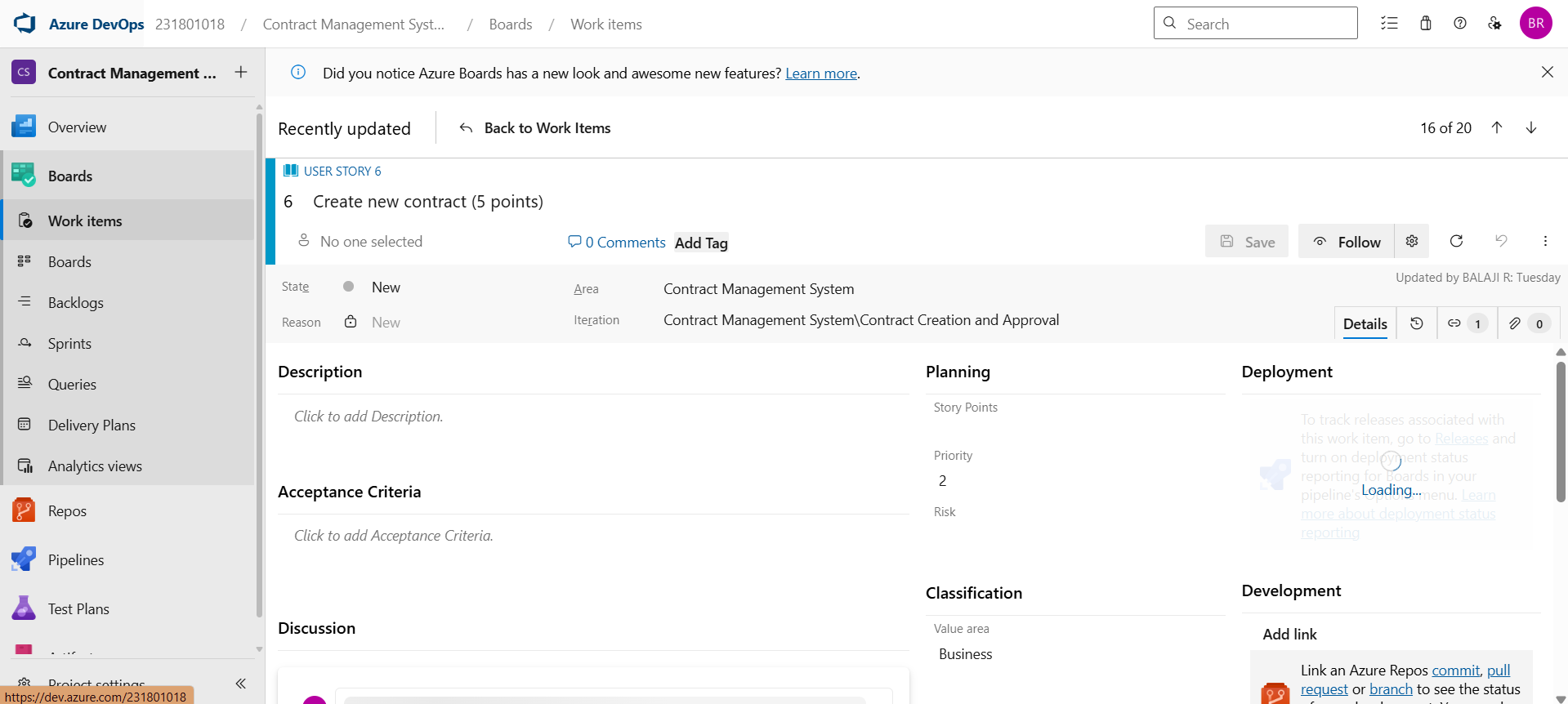
1. **Fill in Epics**

****

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**

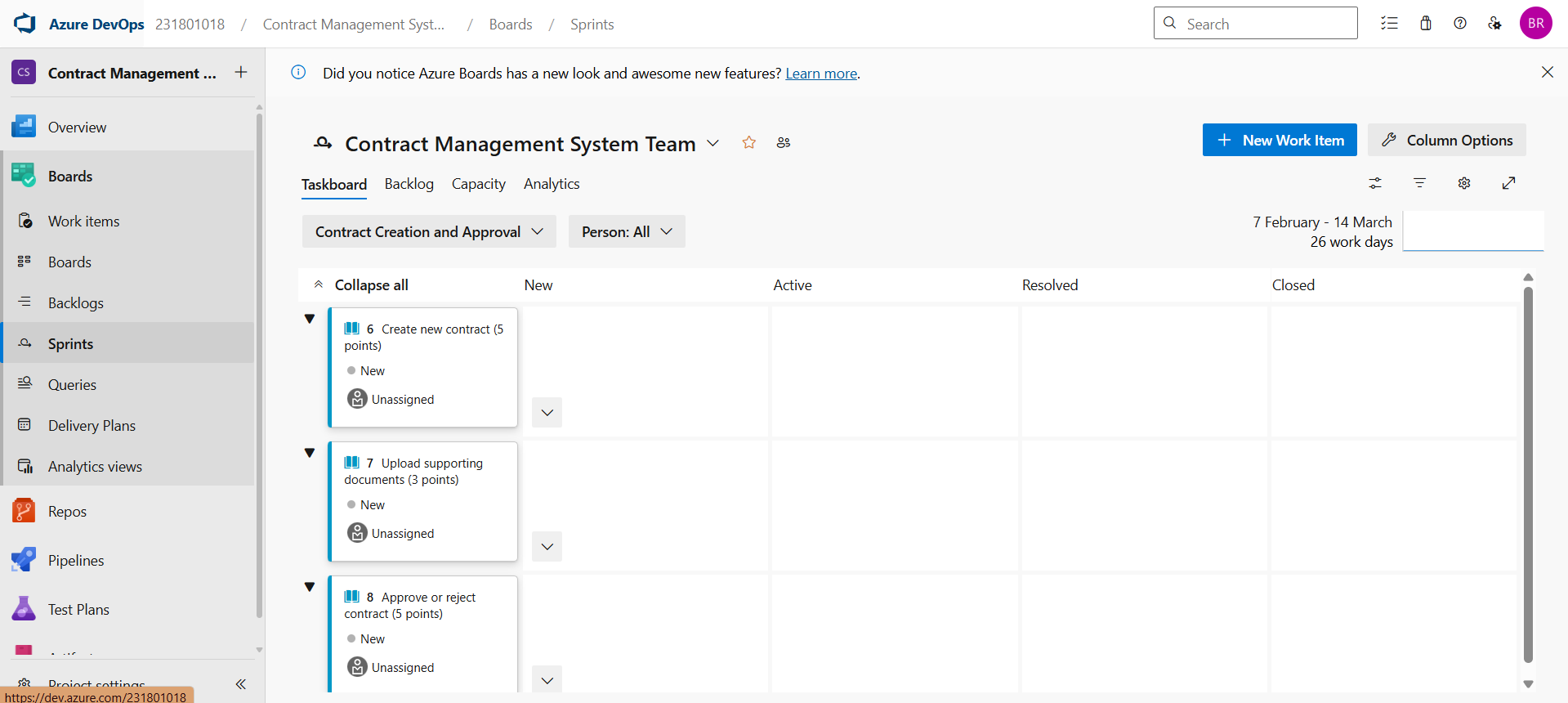
**EXP NO: 4**

**Aim:**

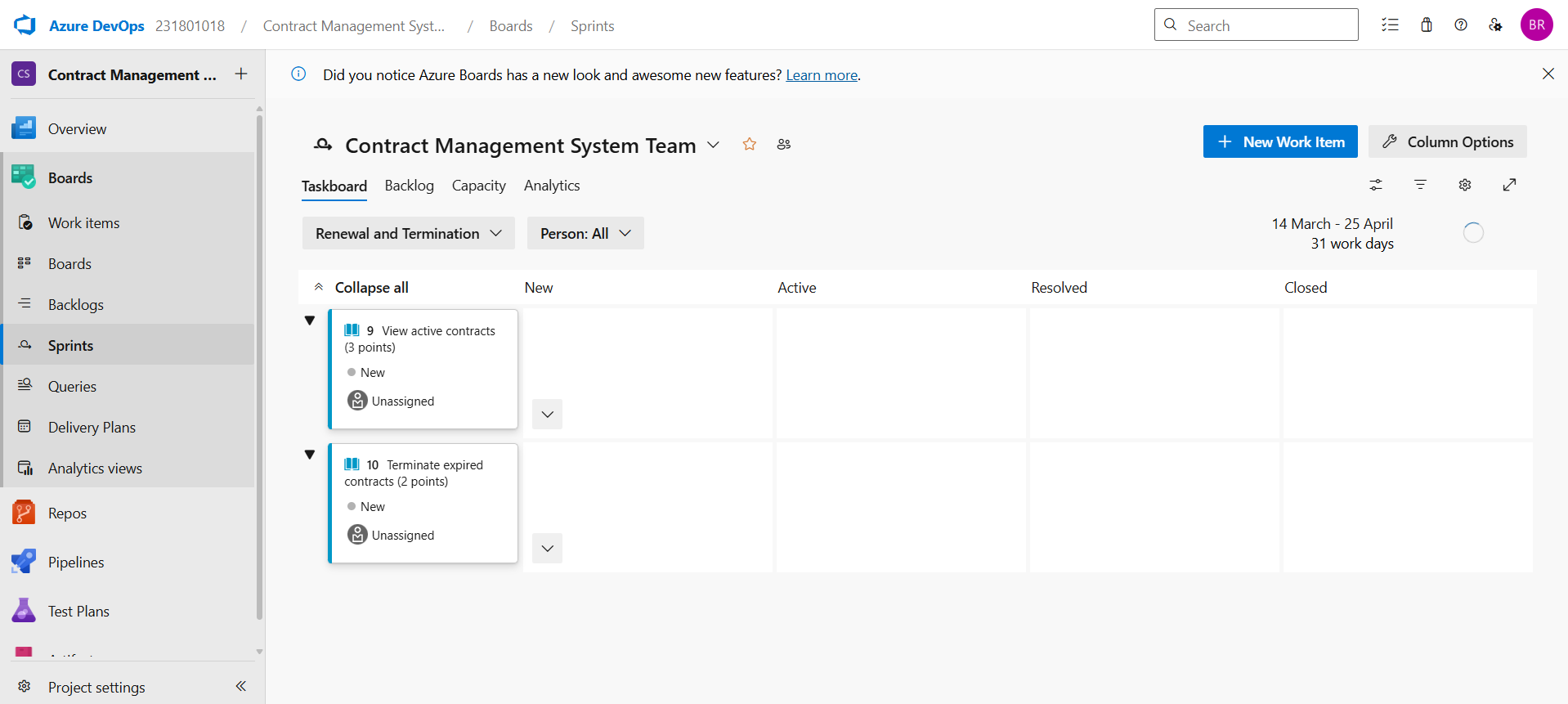
Create Epic, Features, User Stories, Task

To assign user story to specific sprint for the Contract Management System Project.

**Sprint Planning Sprint 1**

****

**Sprint 2**

****

**Result:**

The Sprints are created for the Contract Management System Project.

**POKER ESTIMATION**

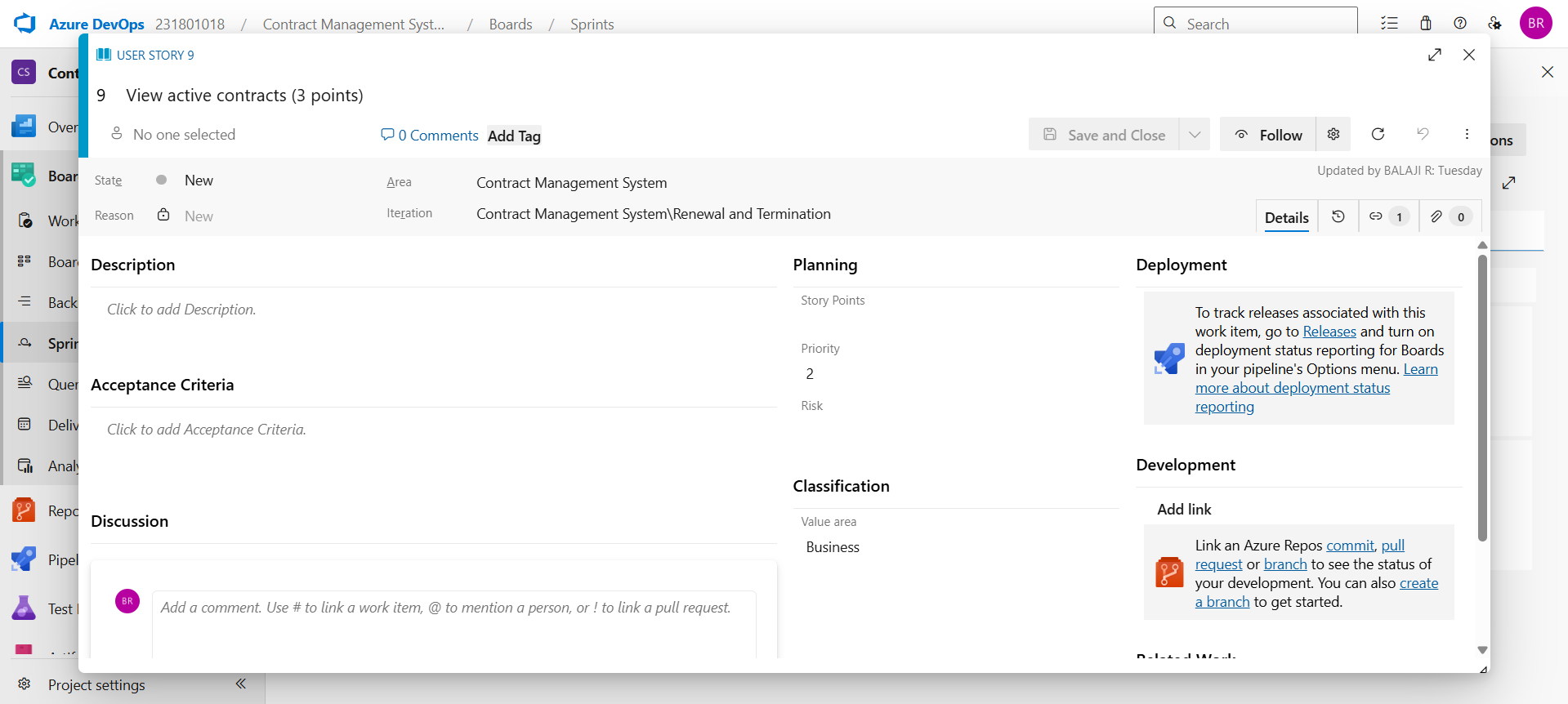
**EXP NO: 5**

**Aim:**

Create Epic, Features, User Stories, Task

Create Poker Estimation for the user stories – Contract Management System Project.

**Poker Estimation**

****

**Result:**

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

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

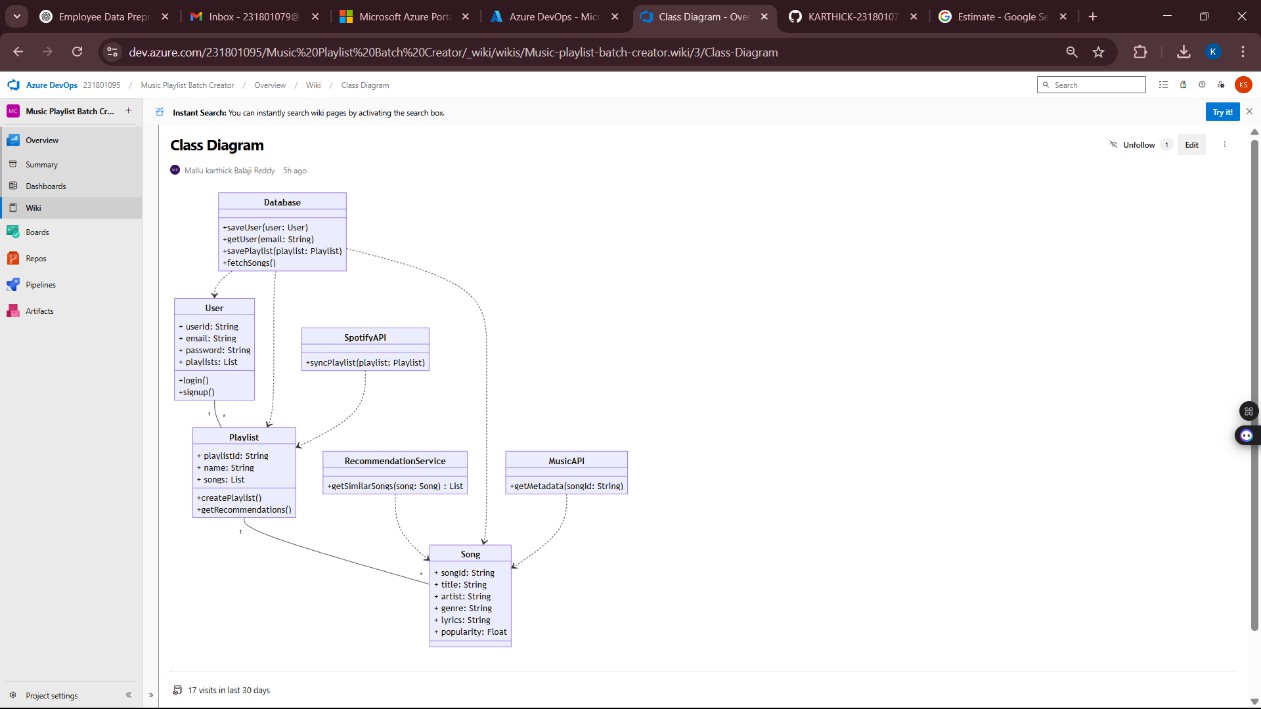
**EXP NO: 6**

**Aim:**

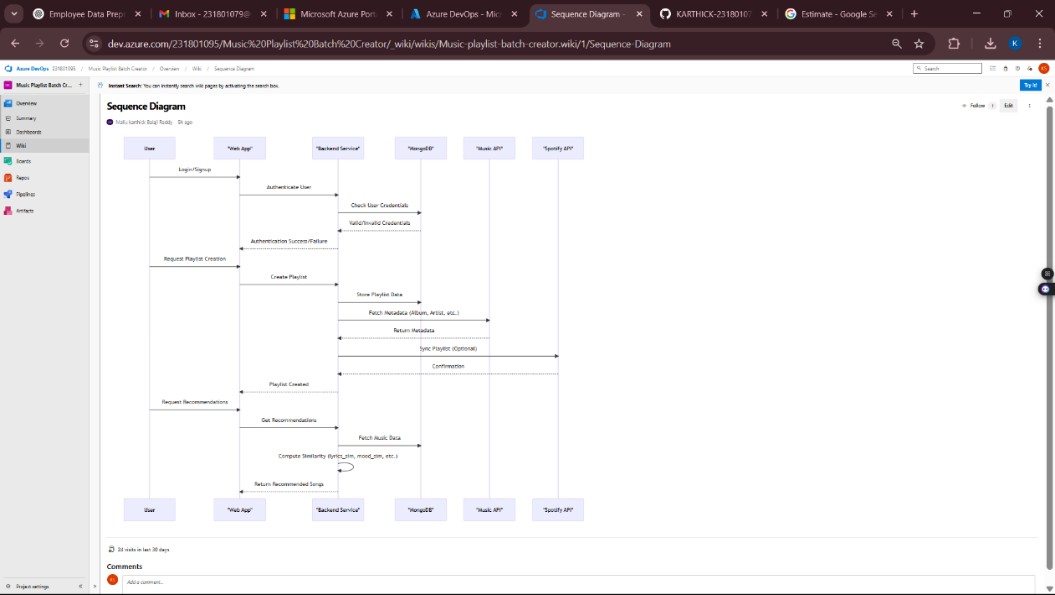
Create Epic, Features, User Stories, Task

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 Contract Management System.

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

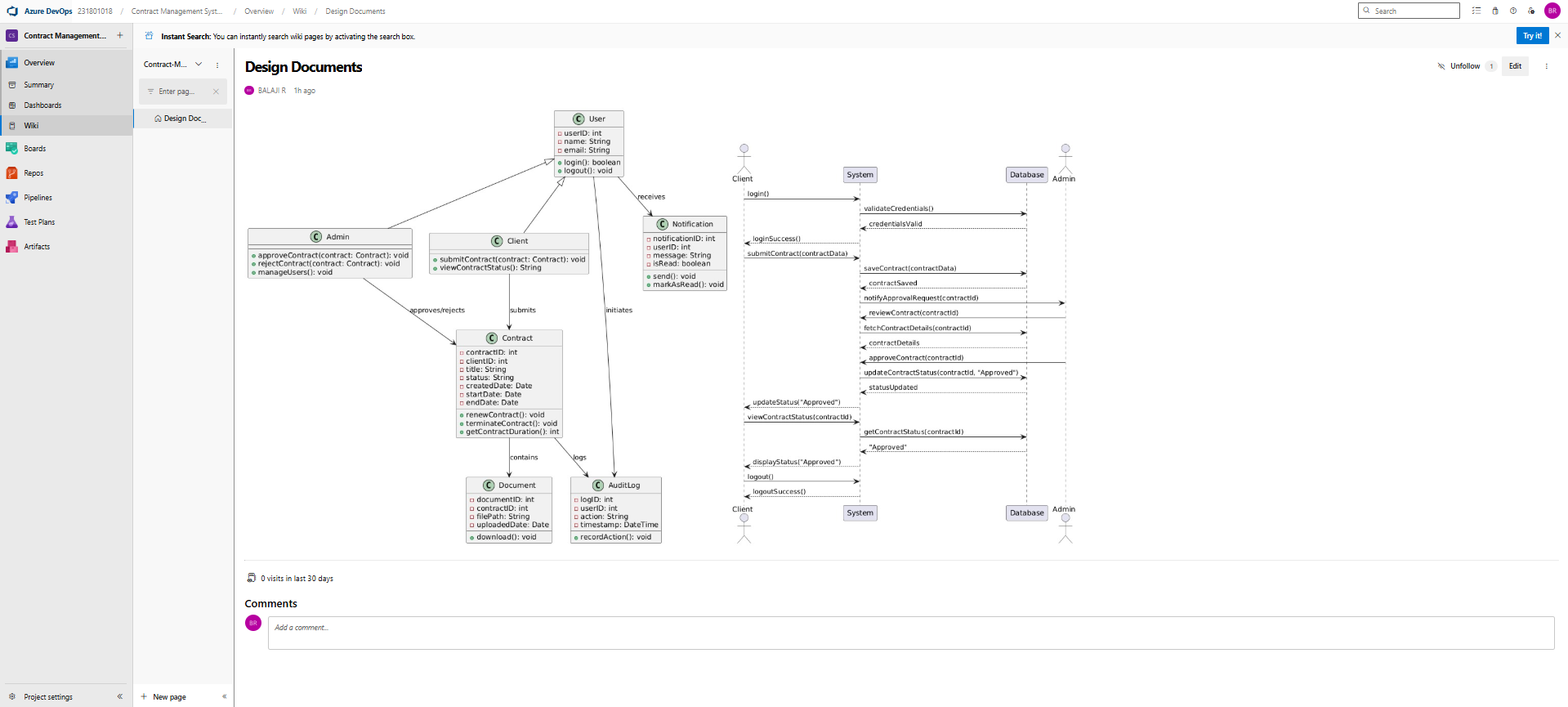
**EXP NO: 7**

**Aim:**

Create Epic, Features, User Stories, Task

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

**7A. Architectural Diagram**

****

**Result:**

The Architecture Diagram and ER Diagram is designed Successfully for the Contract Management System.

**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**

**Test Case Design Procedure – *Contract Management System***

**1. Understand Core Features of the Application**

* **User Login & Access Control**
* **Create New Contract**
* **Upload Supporting Documents**
* **Approve / Reject Contracts**
* **View Active & Expired Contracts**
* **Terminate Expired Contracts**

**2. Define User Interactions**

**Each test case will simulate real-world usage:**

* **Logging in as an authorized user**
* **Creating a new contract**
* **Uploading valid/invalid files**
* **Approving contracts with or without reason**
* **Viewing contract lists (logged in or not)**
* **Terminating contracts based on status**

**3. Design Happy Path Test Cases**

**Focus on verifying that features work correctly under normal conditions:**

* **TC01 – Login with valid credentials**
* **TC02 – Successfully create contract**
* **TC03 – Upload valid PDF**
* **TC04 – Approve contract with reason**
* **TC05 – View active contracts after login**
* **TC06 – Terminate expired contract**

**4. Design Error Path Test Cases**

**Simulate incorrect inputs and unexpected behavior:**

* **TC07 – Login with incorrect password**
* **TC08 – Create contract with missing fields**
* **TC09 – Upload unsupported file type (e.g., .exe)**
* **TC10 – Reject contract without reason**
* **TC11 – View contracts without login**
* **TC12 – Attempt to terminate active contract**

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

**Each test case contains:**

* **Clear action steps (e.g., "Click Submit")**
* **Expected result (e.g., "Contract created", or "Error message displayed")  
  This format ensures both manual testers and automation engineers can follow easily.**

**6. Use Clear Naming and IDs**

**Test Cases follow a consistent ID pattern:**

* **Format: TC0X – Description**
* **Example: TC03 – Upload Valid PDF File  
  This helps link cases to corresponding User Stories or Features.**

**7. Separate Test Suites**

**Organized by function for better manageability:**

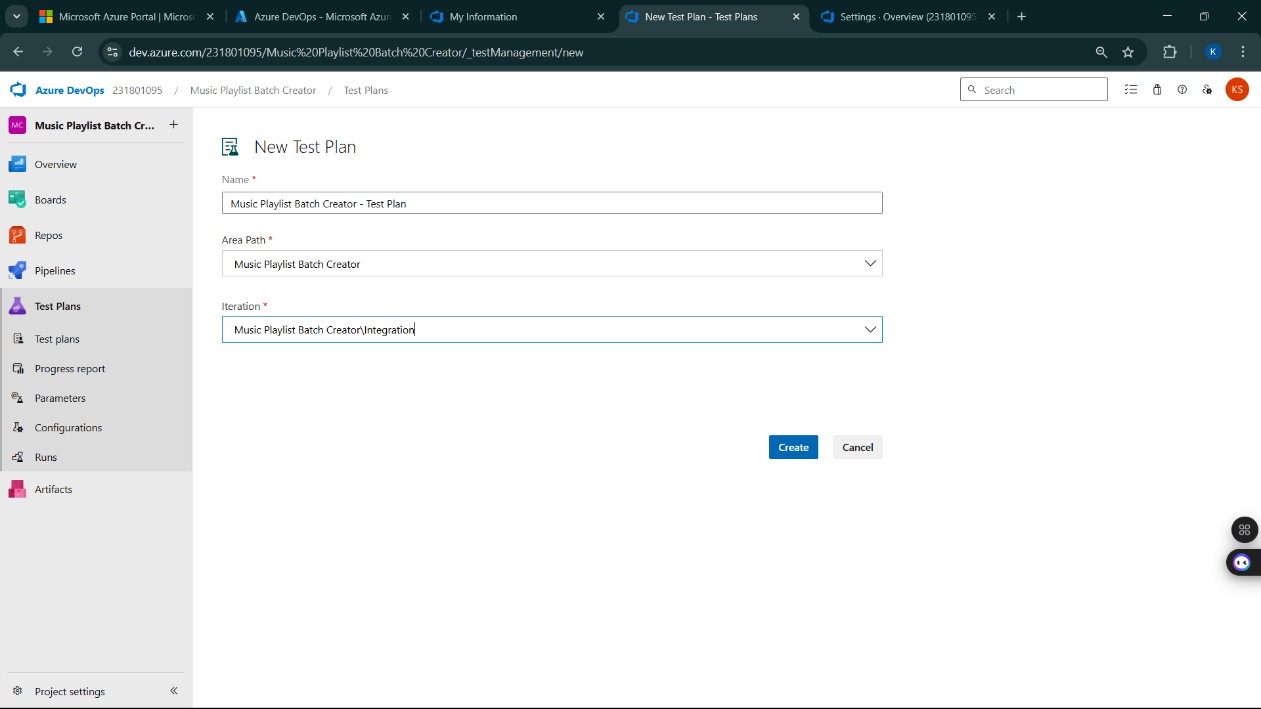
* **🔹 Login & Access**
* **🔹 Contract Creation**
* **🔹 File Uploads**
* **🔹 Approval/Review Workflow**
* **🔹 Contract Viewing & Termination**

**These suites can be directly created in Azure DevOps Test Plans for smooth tracking.**

**8. Prioritize and Review**

* **Critical flows like Login, Contract Creation, and Approval are marked as High Priority**
* **All test cases are reviewed for:**
  + **Feature coverage**
  + **Traceability to requirements**
  + **Completeness and accuracy**

1. **New test plan**

****

1. **Test case**

**Test Suite TS01 - User Authentication**

**User Story ID**: US01  
**Title**: As a user, I want to securely sign up and log in so that I can manage contracts

**Test Case TC01** – Successful Sign Up  
Type: Happy Path  
Steps:

1. Navigate to the Sign-Up page
2. Enter valid name, email, and password
3. Click on Sign Up  
   Expected Result: Account is created and user is redirected to dashboard

**Test Case TC02** – Sign Up with Existing Email  
Type: Error Path  
Steps:

1. Navigate to the Sign-Up page
2. Enter a registered email
3. Click on Sign Up  
   Expected Result: Error message is displayed saying email is already in use

**Test Suite TS02 - Create Contract**

**User Story ID**: US02  
**Title**: As a user, I want to create a new contract with essential fields

**Test Case TC03** – Create Contract Successfully  
Type: Happy Path  
Steps:

1. Log in
2. Click on Create Contract
3. Fill all required fields
4. Click Save  
   Expected Result: Contract is created and visible in contract list

**Test Case TC04** – Contract Creation with Missing Data  
Type: Error Path  
Steps:

1. Log in
2. Click on Create Contract
3. Leave mandatory fields blank
4. Click Save  
   Expected Result: Validation error message shown and contract is not created

**Test Suite TS03 - View Contract Details**

**User Story ID**: US03  
**Title**: As a user, I want to view contract details to verify status and content

**Test Case TC05** – View Existing Contract  
Type: Happy Path  
Steps:

1. Log in
2. Go to Contracts list
3. Click on a contract  
   Expected Result: Full contract details are displayed

**Test Case TC06** – Contract Not Found  
Type: Error Path  
Steps:

1. Log in
2. Manually enter an invalid contract ID in the URL  
   Expected Result: Message shown stating contract not found or does not exist

**Test Suite TS04 - Approve or Reject Contract**

**User Story ID**: US04  
**Title**: As a user, I want to approve or reject a contract for review flow

**Test Case TC07** – Approve Contract Successfully  
Type: Happy Path  
Steps:

1. Log in as reviewer
2. Open a pending contract
3. Click on Approve  
   Expected Result: Status updates to Approved

**Test Case TC08** – Try Approving Without Permission  
Type: Error Path  
Steps:

1. Log in as non-reviewer
2. Try to approve a contract  
   Expected Result: Action denied with message stating insufficient permissions

**Test Suite TS05 - Contract Expiry Alerts**

**User Story ID**: US05  
**Title**: As a user, I want to receive alerts before contract expiry

**Test Case TC09** – Receive Expiry Alert  
Type: Happy Path  
Steps:

1. Log in
2. Check dashboard for contracts expiring within seven days  
   Expected Result: Alert is shown for upcoming expiry

**Test Case TC10** – No Alert for Expired Contracts  
Type: Error Path  
Steps:

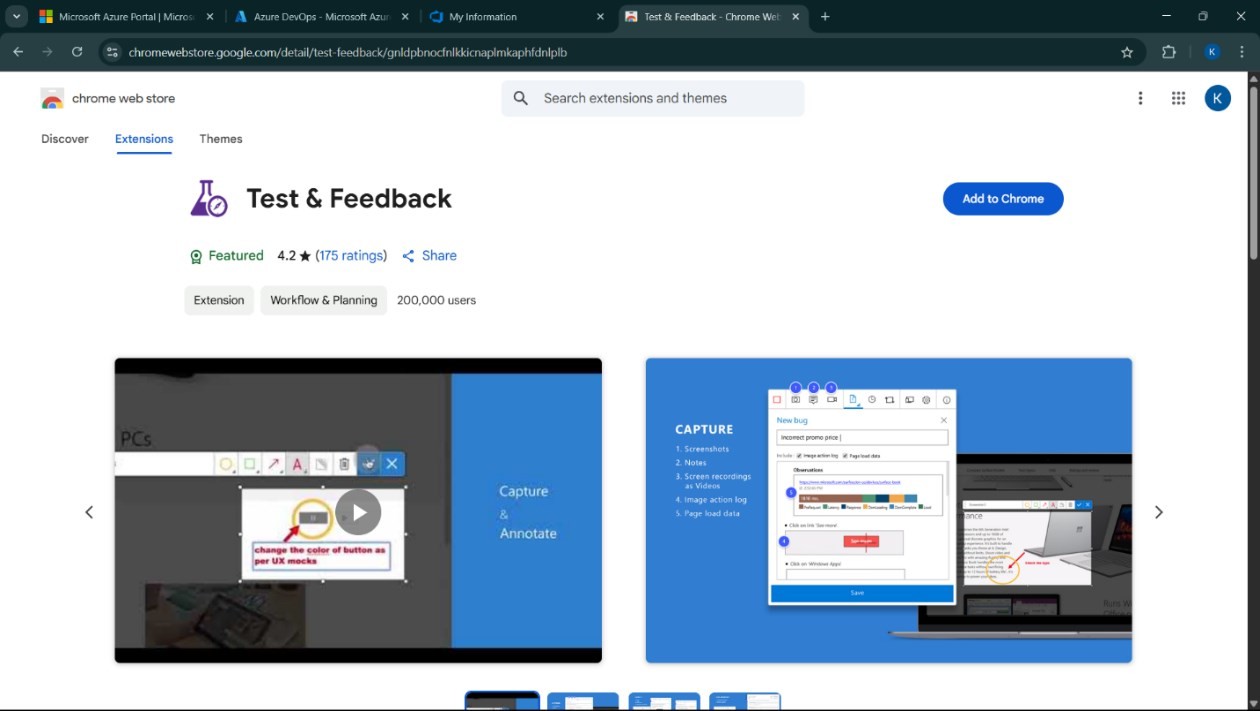
1. Log in after contract has already expired  
   Expected Result: No alert shown and status shows expired

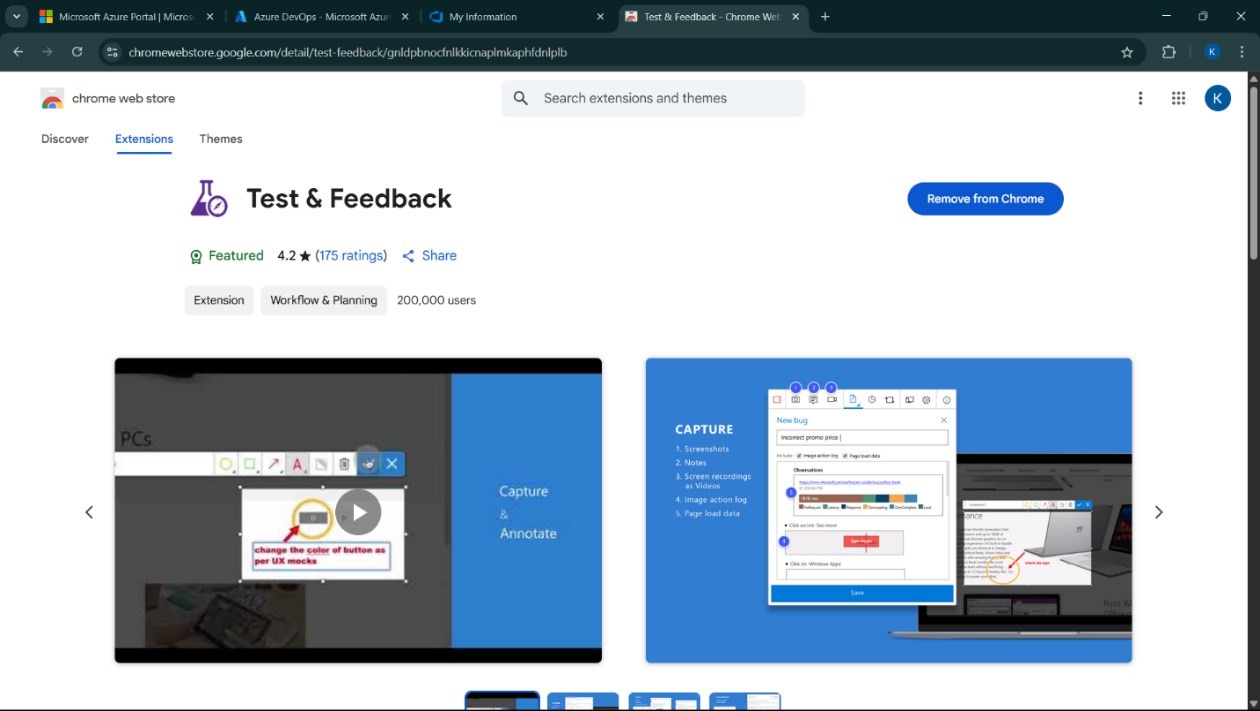
**Test Cases**

**A screenshot of a computer

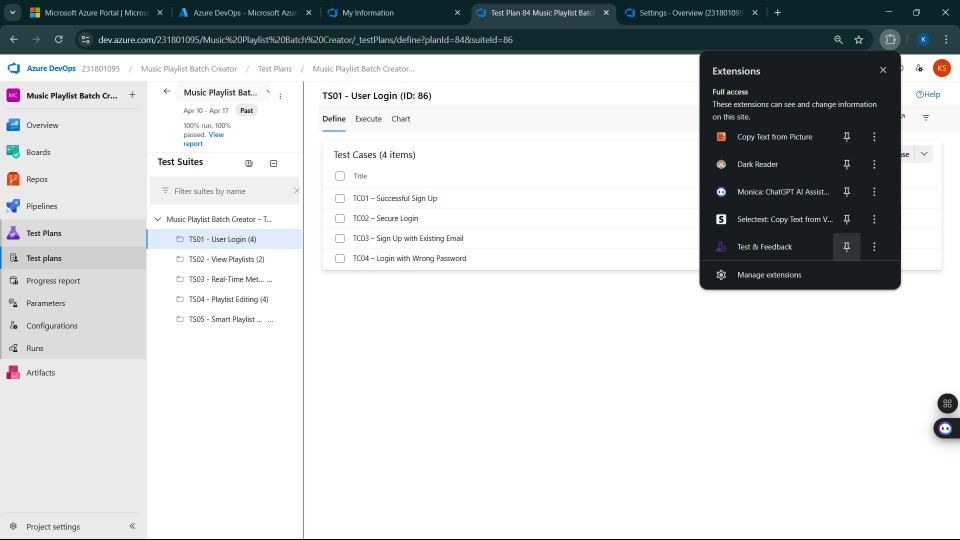
AI-generated content may be incorrect.**

1. **Installation of test**

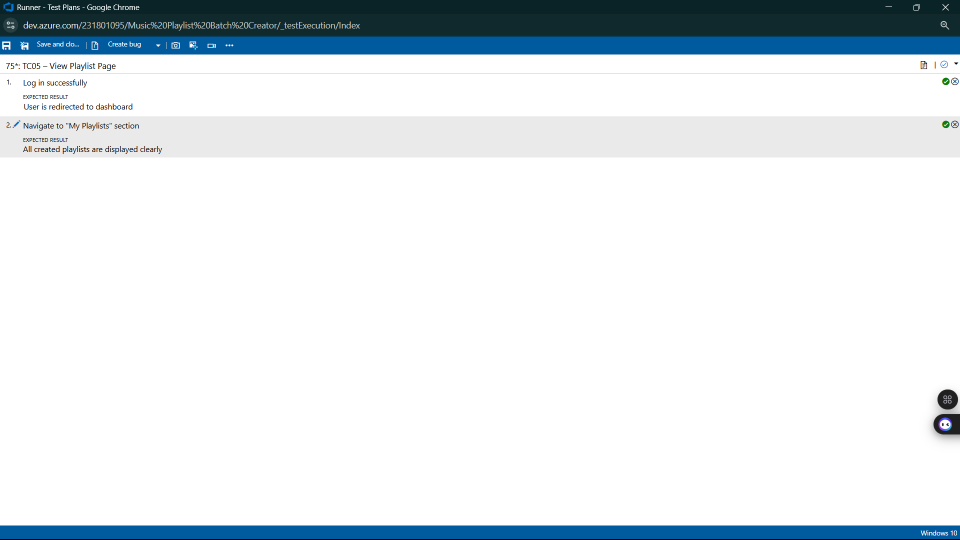
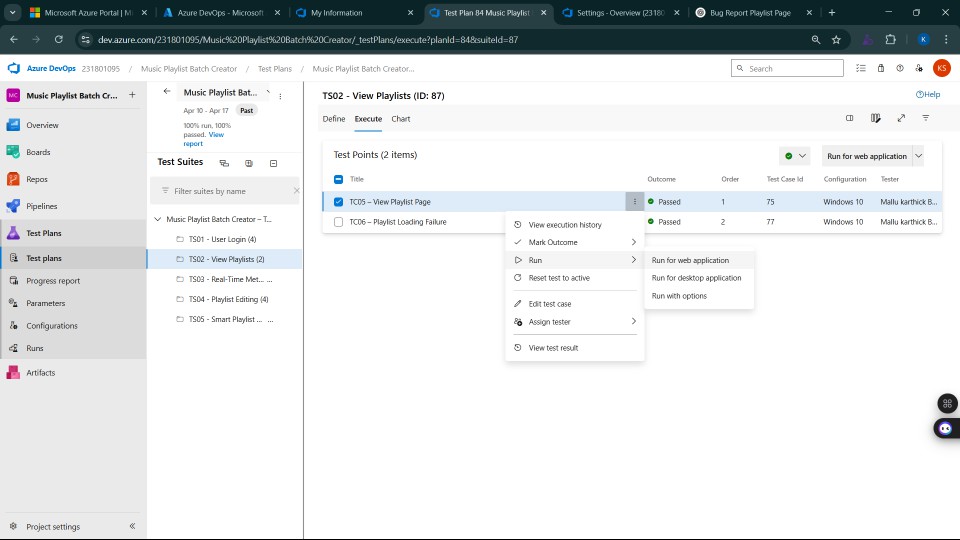
****



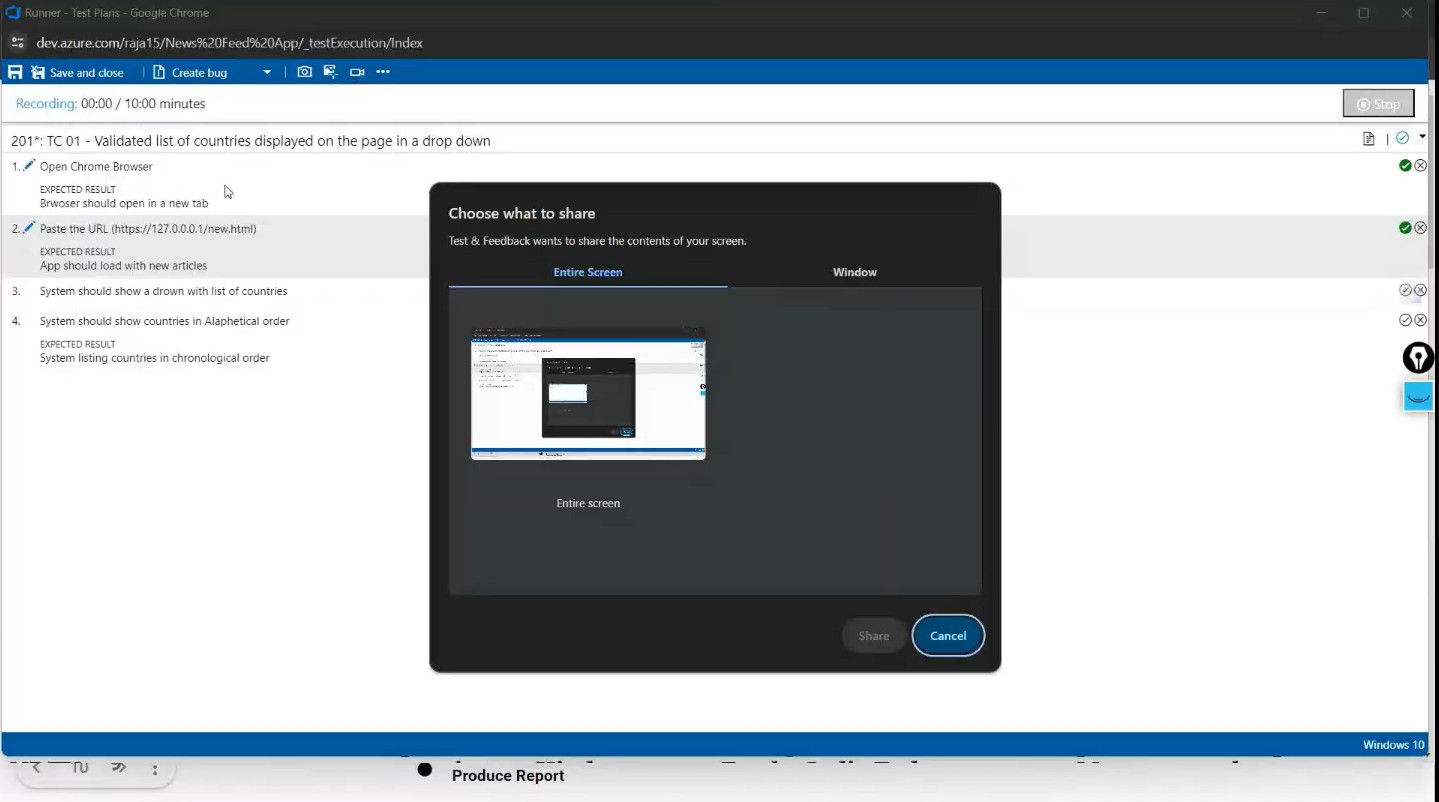
Test and feedback Showing it as an extension



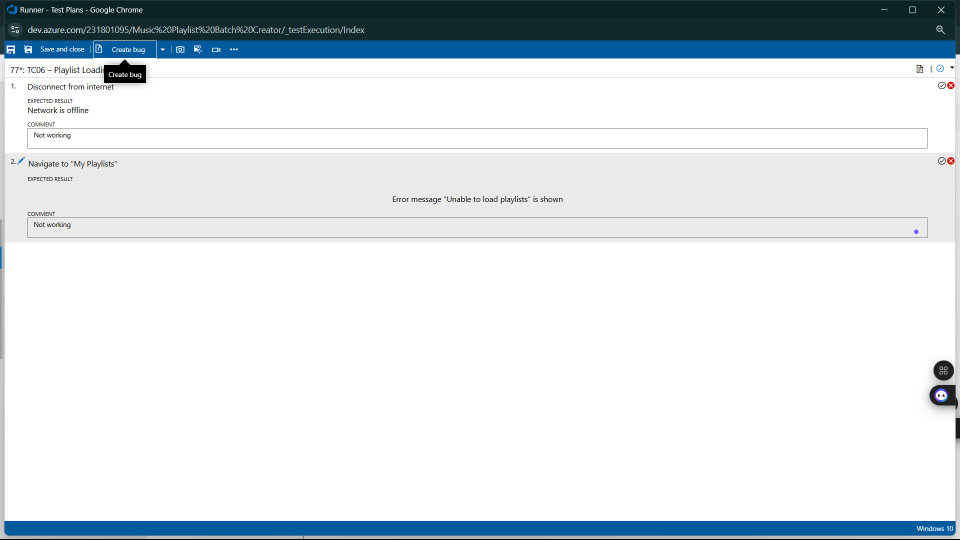
1. **Running the test cases**

****

1. **Recording the test case**

****

1. **Creating the bug**

****

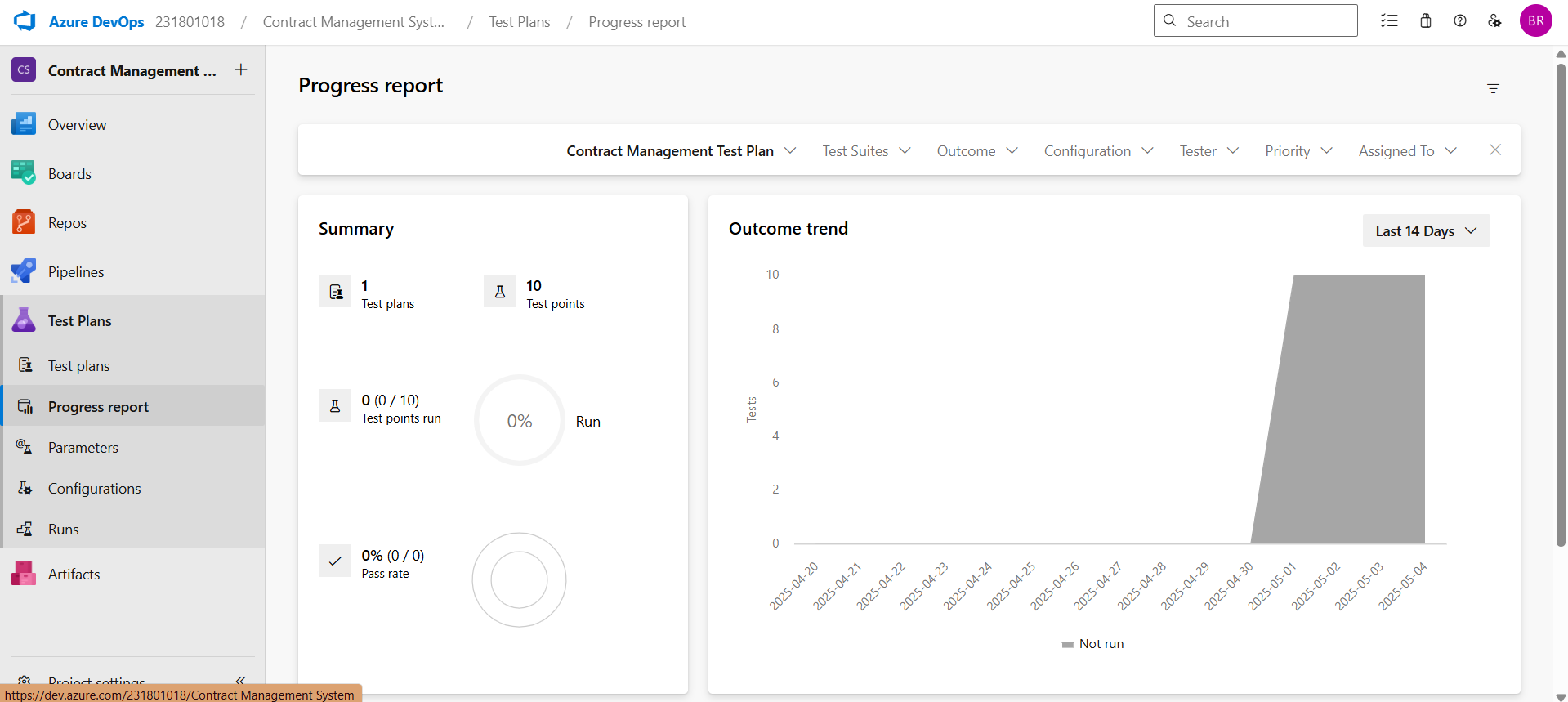
1. **Test report summary**

**A screenshot of a computer

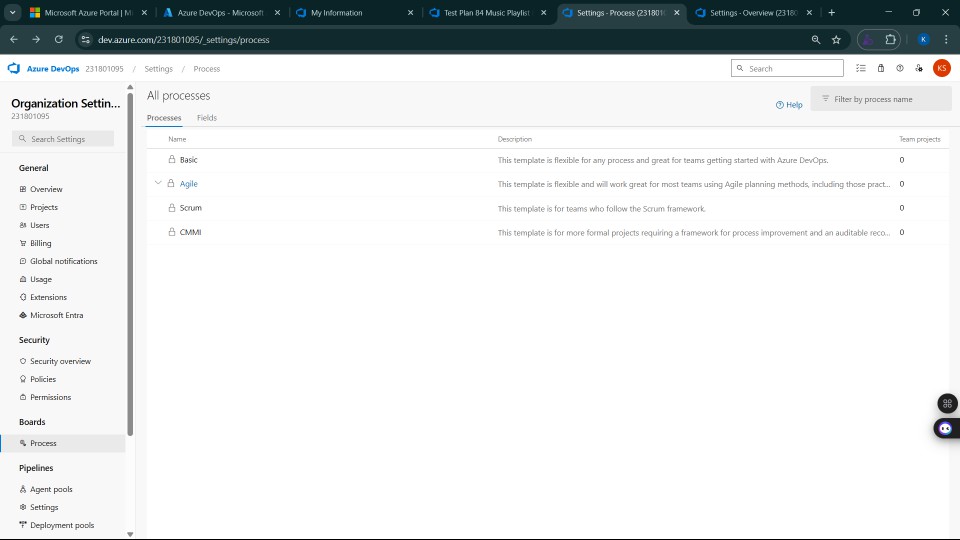
AI-generated content may be incorrect.**

* + Assigning bug to the developer and changing state

1. **Progress report**

****

1. **Changing the test 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 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](https://portal.azure.com/) and log in.

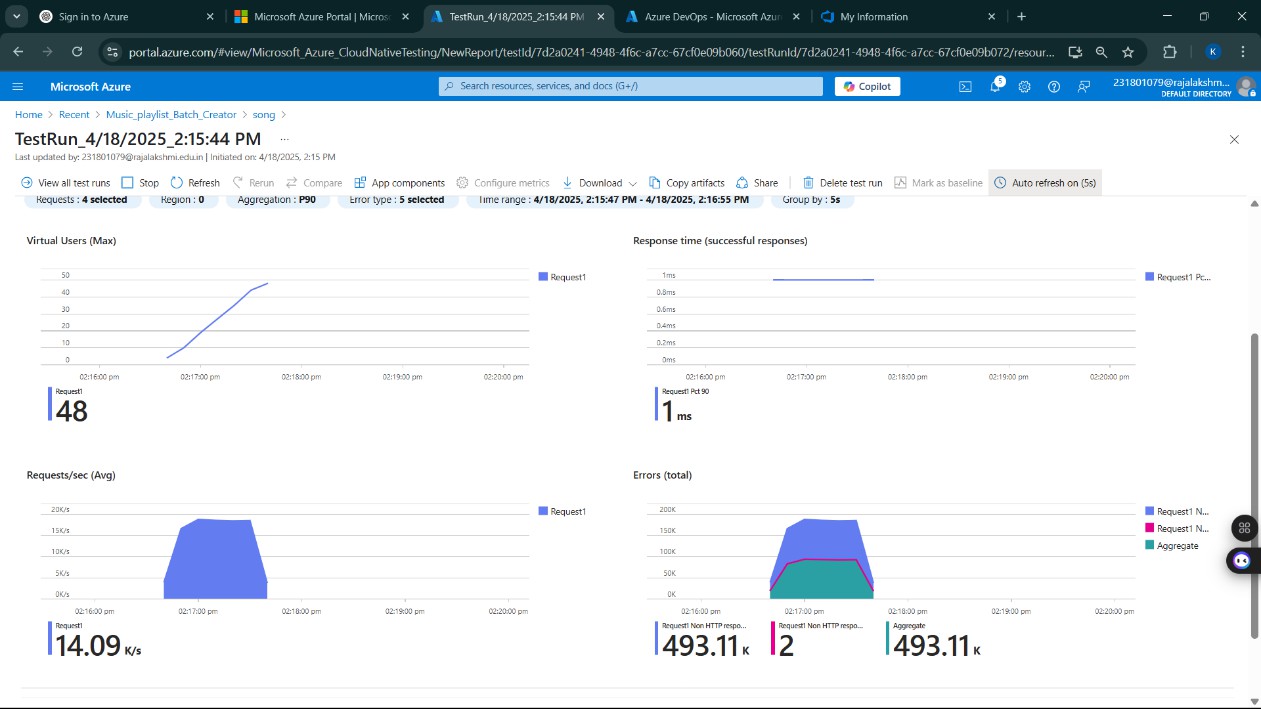
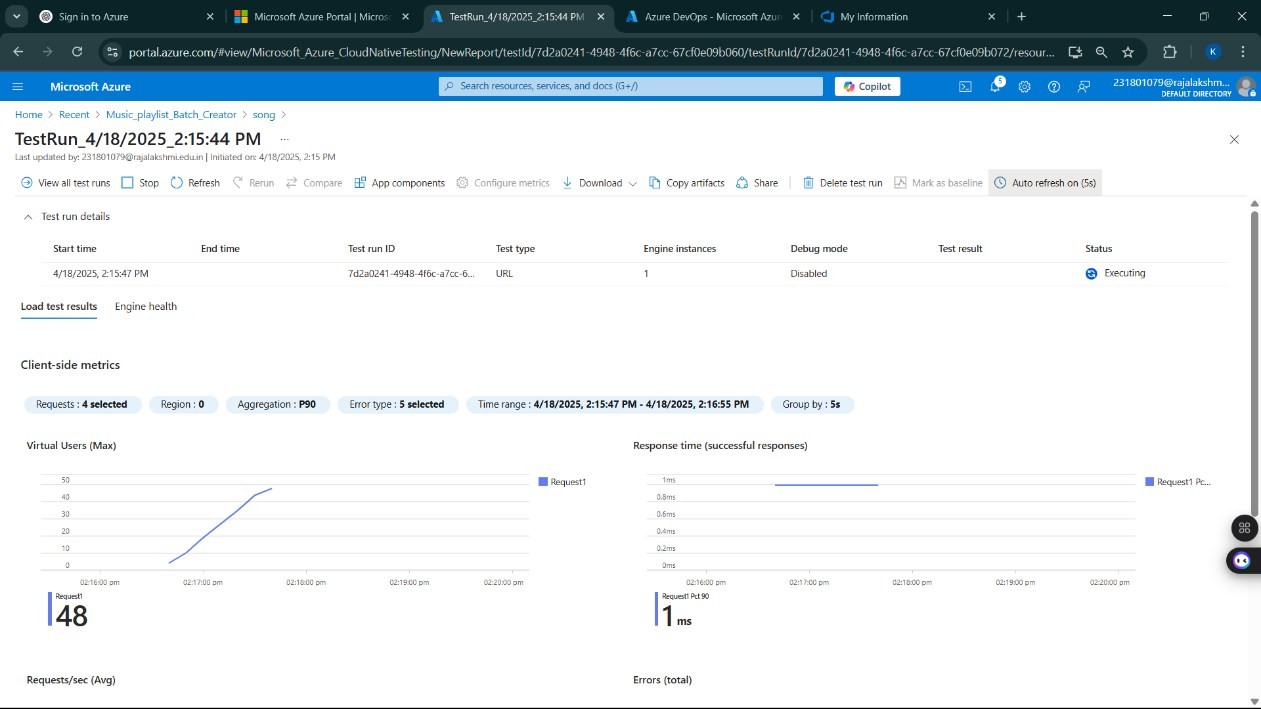
1. Create the Resource
   * Go to *Create a resource* → Search for “Azure Load Testing”.
   * Select Azure Load Testing and click Create.
2. Fill in the Configuration Details
   * Subscriptio*n:* Choose your Azure subscription.
   * *Resource Group:* Create new or select an existing one.
   * *Name:* 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 ch ecks 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

pool:

vmImage: 'ubuntu-latest'

steps:

- checkout: self

- script: echo "✅ Azure Pipeline for Contract Management System is running!"

displayName: 'Run sample 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.
2. 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**

**A screenshot of a computer

AI-generated content may be incorrect.**

**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.

**GITHUB: PROJECT STRUCTURE & NAMING CONVENTIONS**

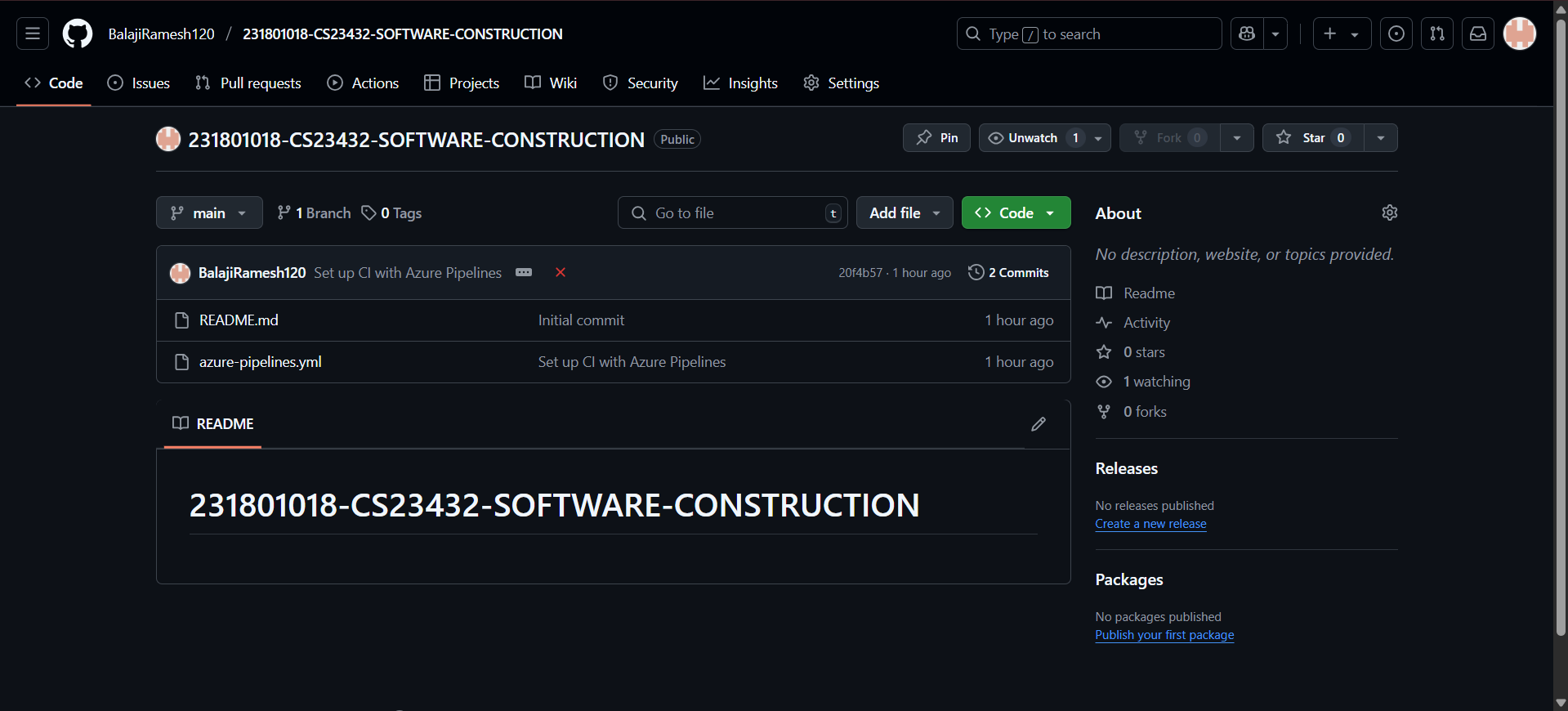
**EXP NO: 10**

**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 Contract Management System 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.