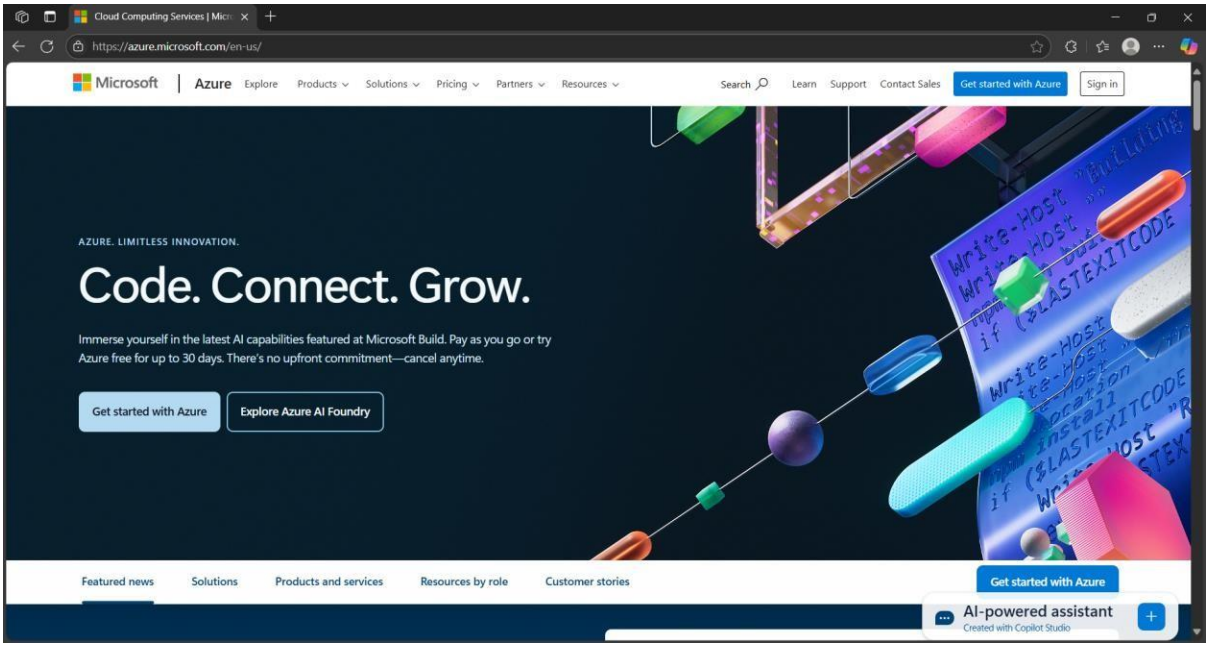
Expr no 1	Azure DevOps Environment Setup

Aim:

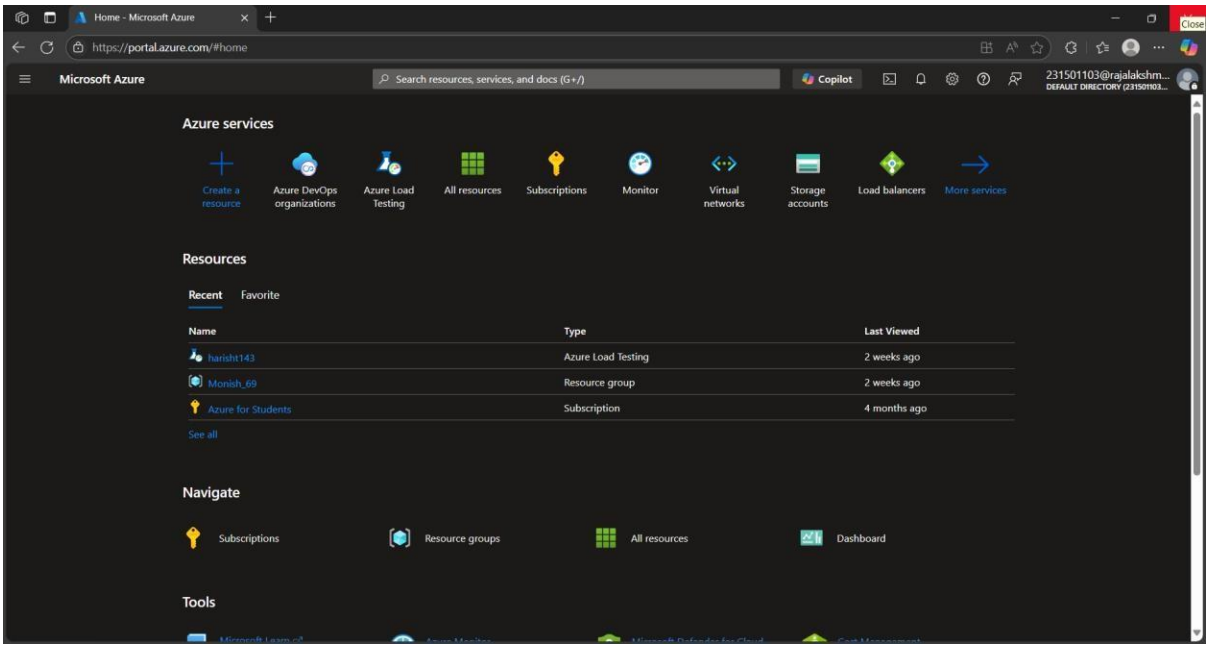
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

Installation:

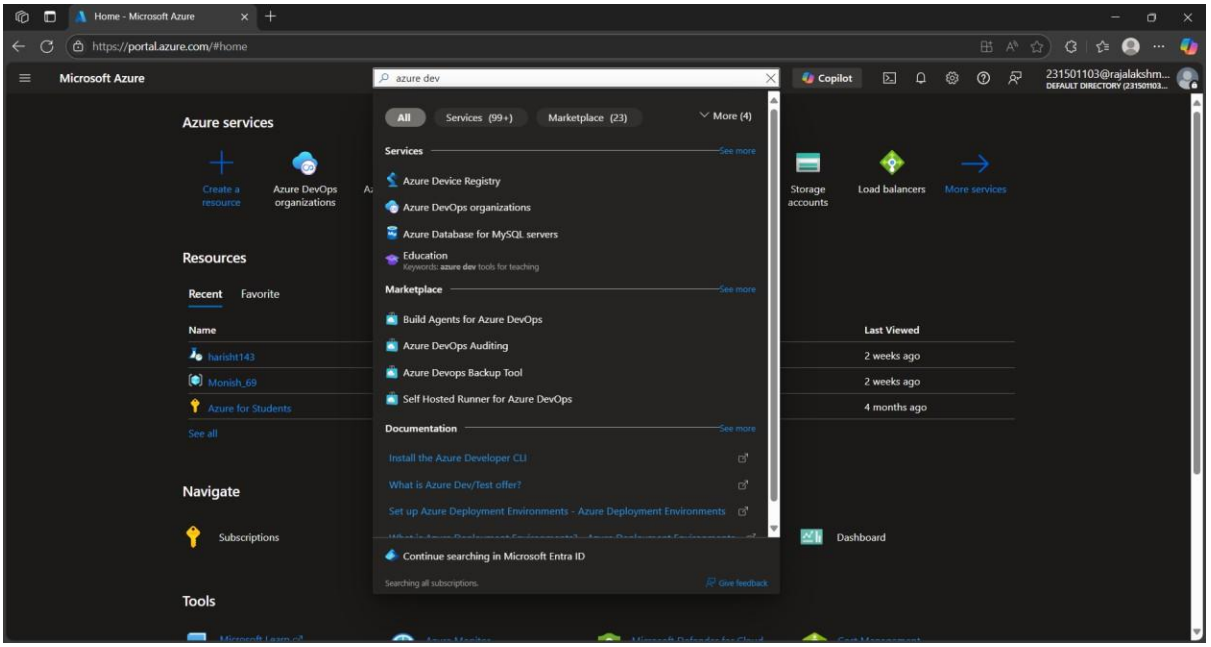
1. Open your web browser and go to the Azure website: https://azure.microsoft.com/en-us/get started/azure-portal. Sign in using your Microsoft account credentials. If you don't have a Microsoft account, you can create one here: https://signup.live.com/?lic=1



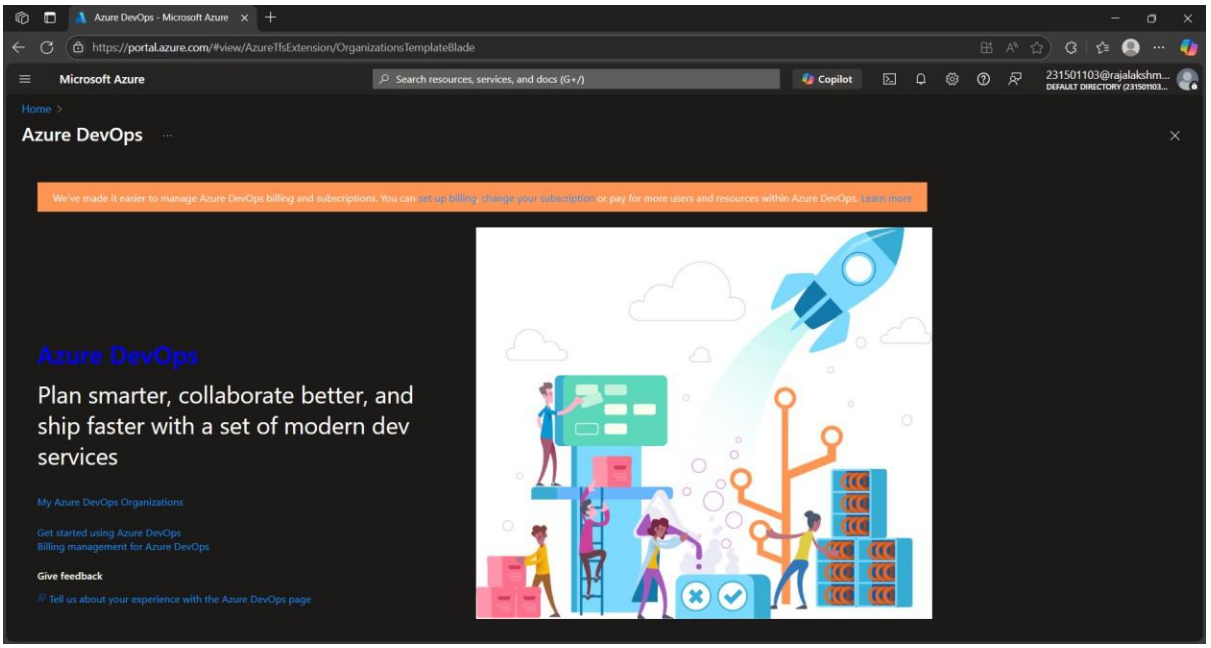
2. Azure Home Page



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:

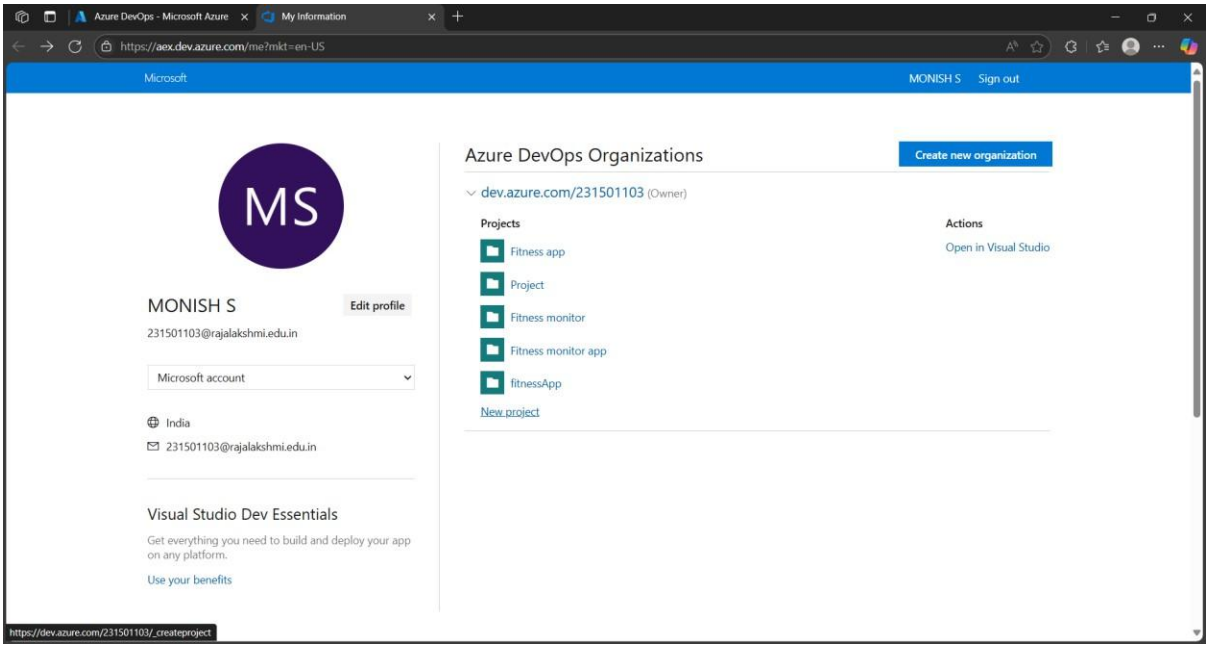
Successfully accessed the Azure DevOps environment and created a new organization through the Azure portal.

Expr no 2	AZURE DEVOPS PROJECT SETUP AND USER STORY MANAGEMENT

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.

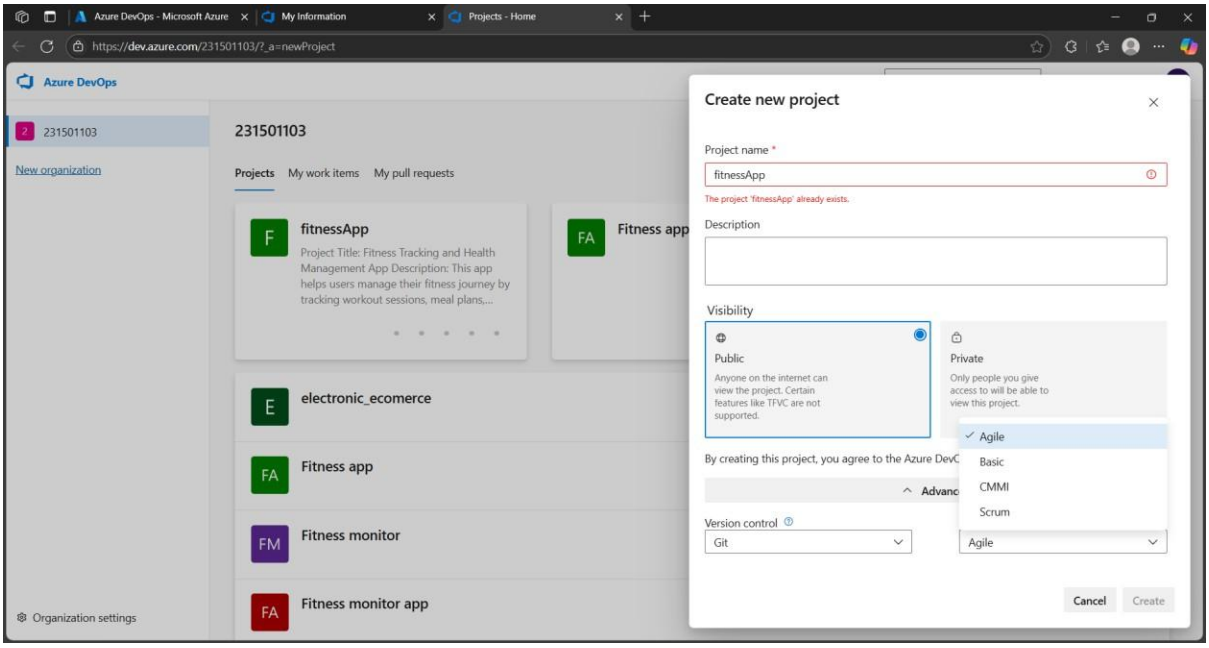
c. Enter the project name, description, and visibility options:

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

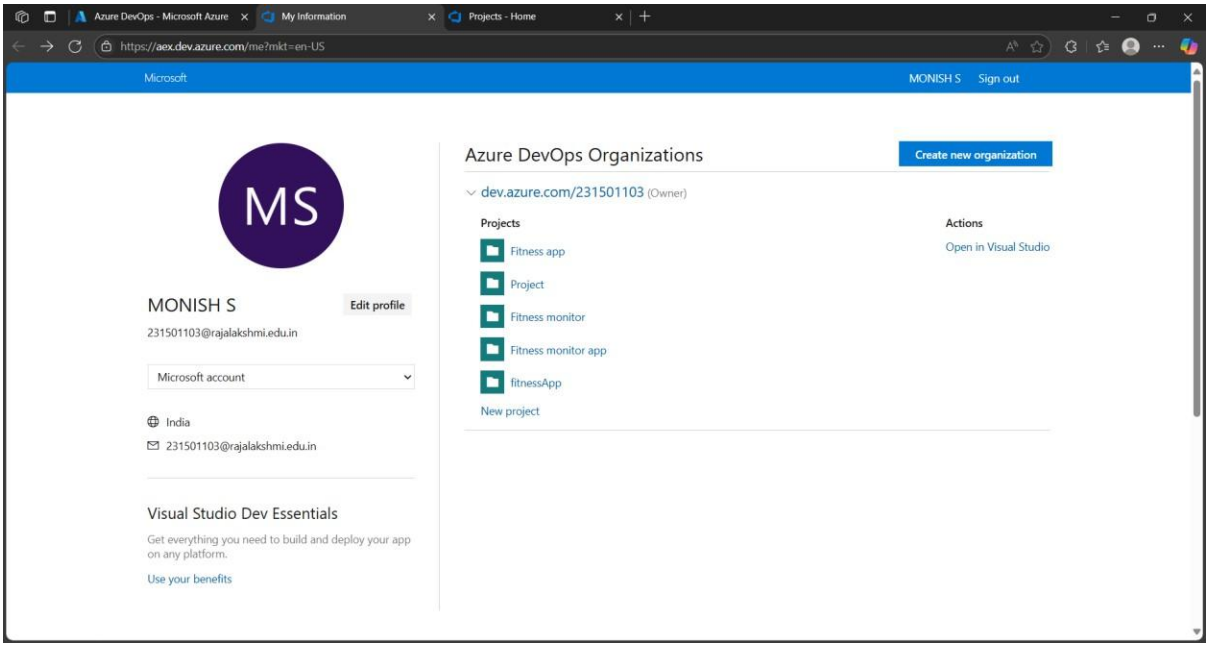
Description: Optionally, add a description to provide more context about the project.

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

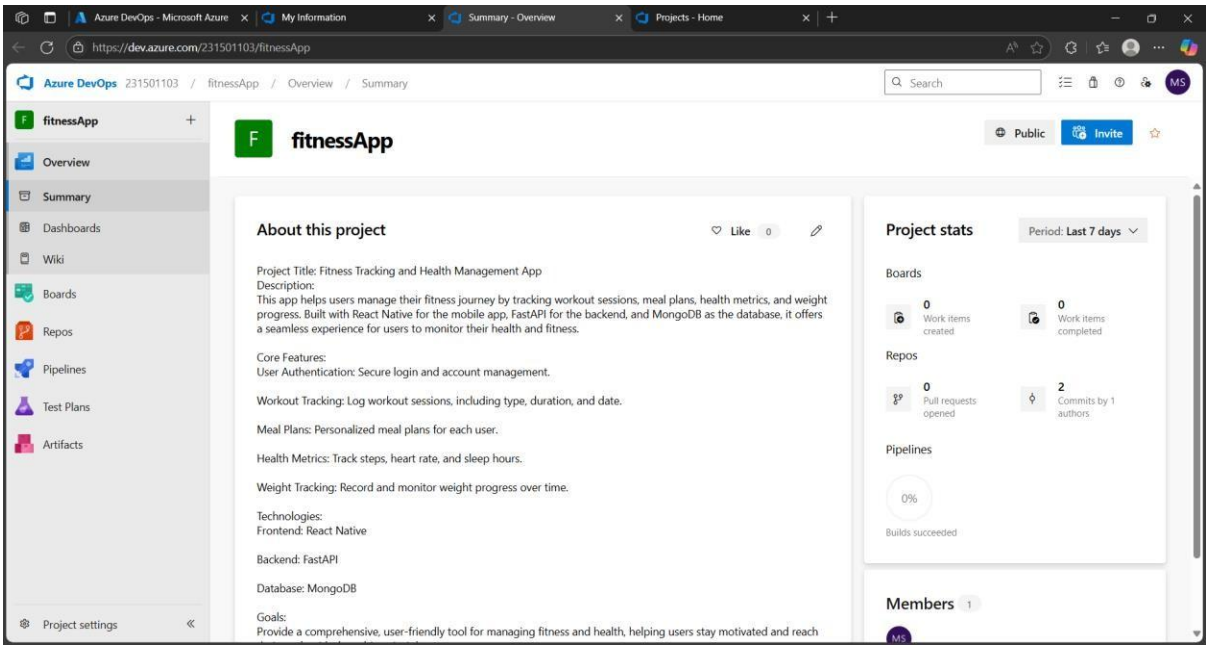
d. Once you've filled out the details, click **Create** to set up your first project.



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.



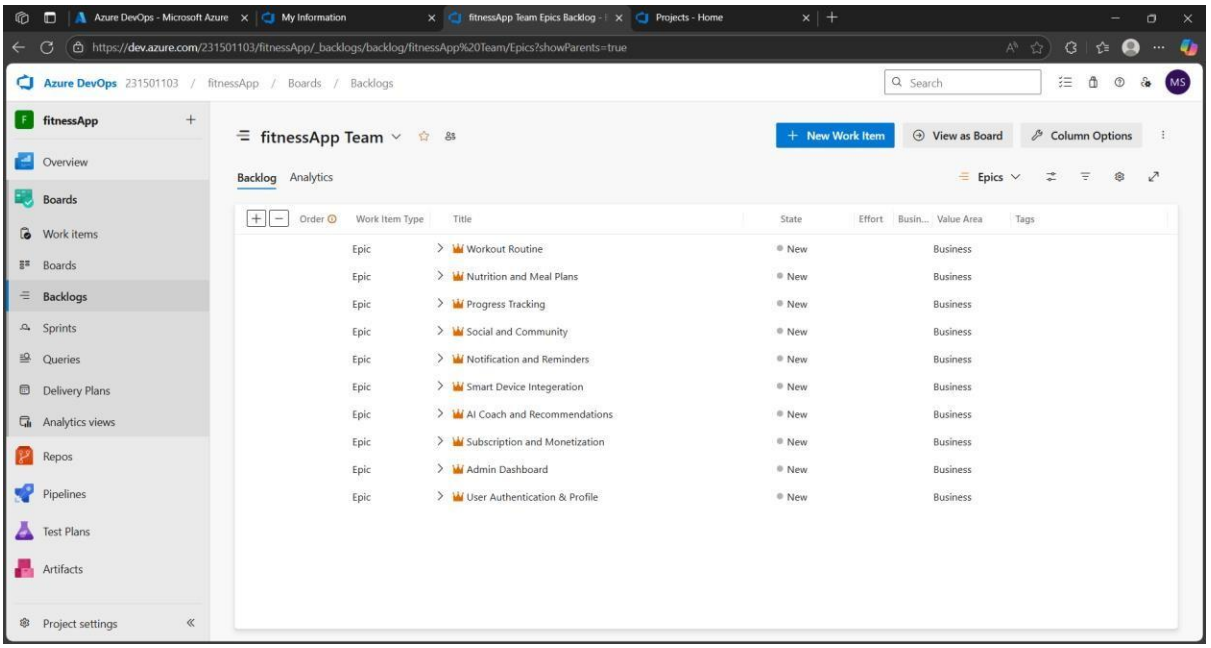
4. Project dashboard

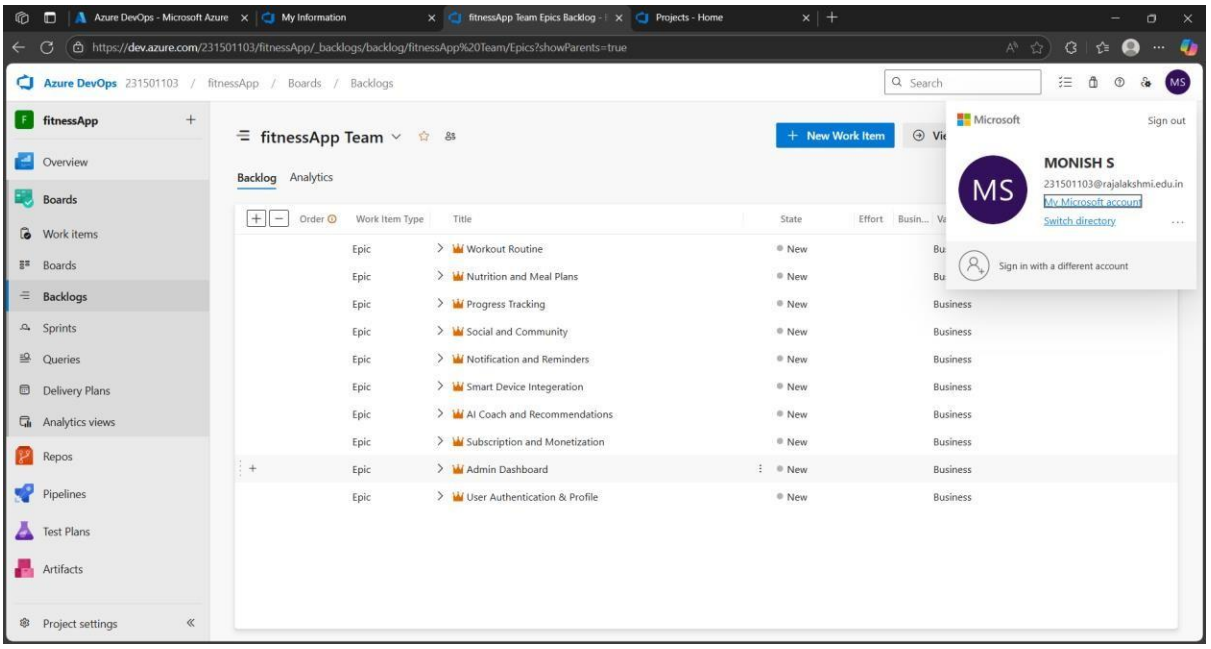


5. To manage user stories:

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:

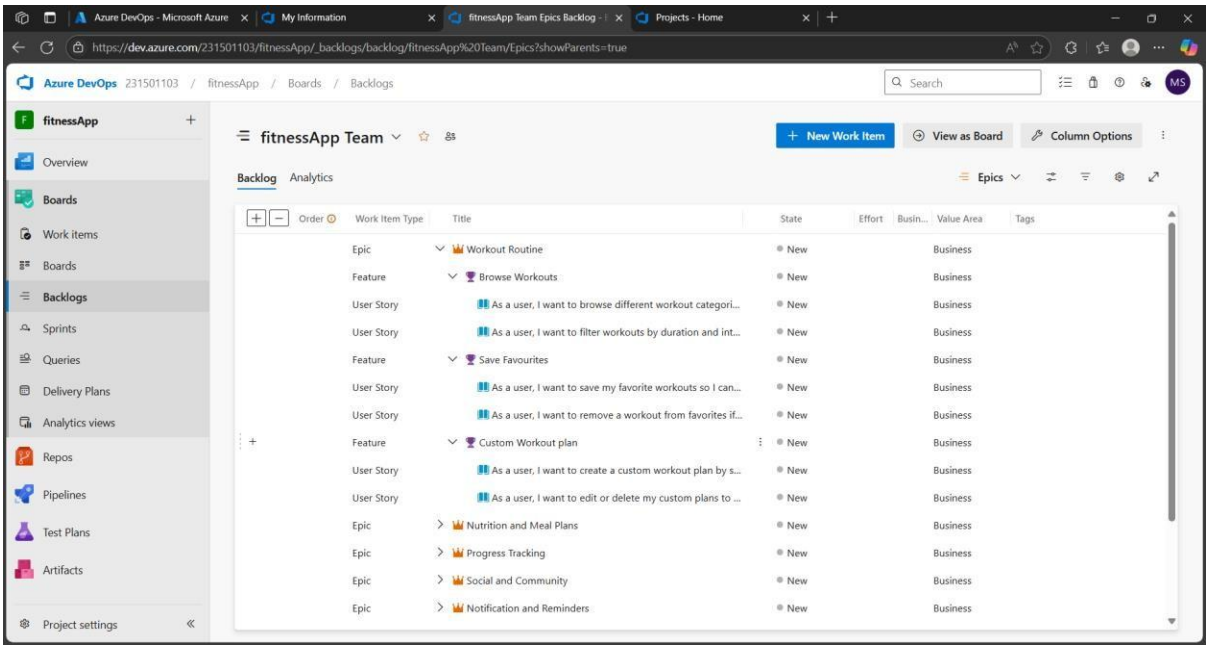
Successfully created an Azure DevOps project with user story management and agile workflow setup.

Expr no 3	SETTING UP EPICS, FEATURES, AND USER STORIES FOR PROJECT PLANNING

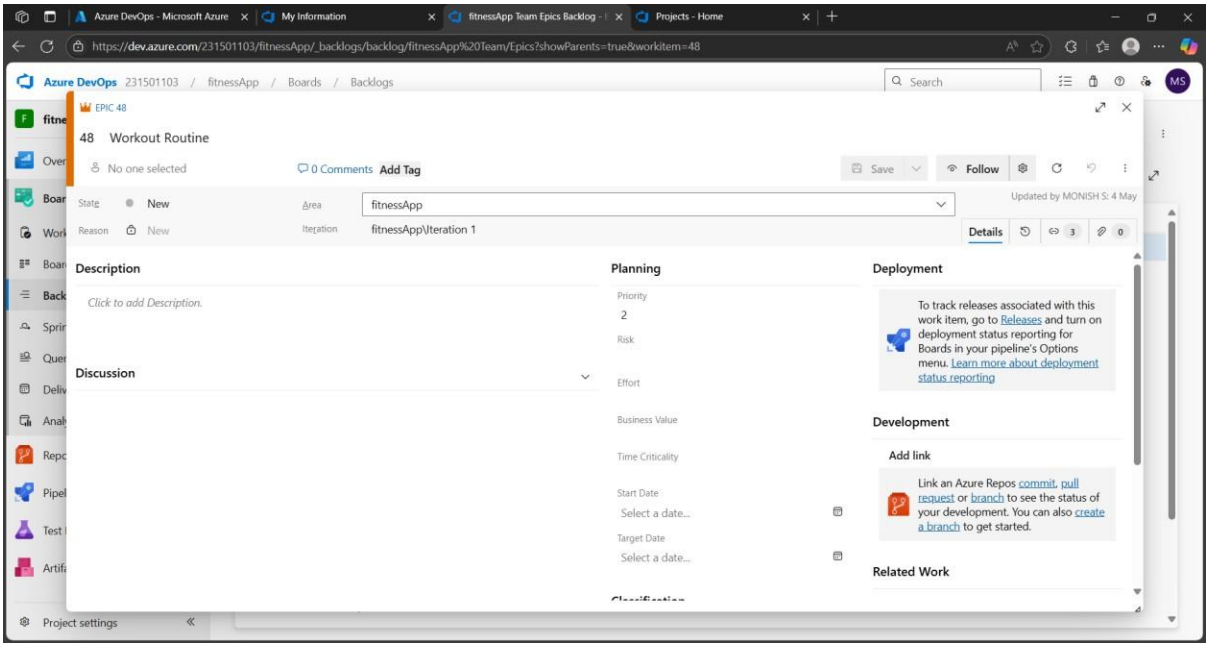
Aim:

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

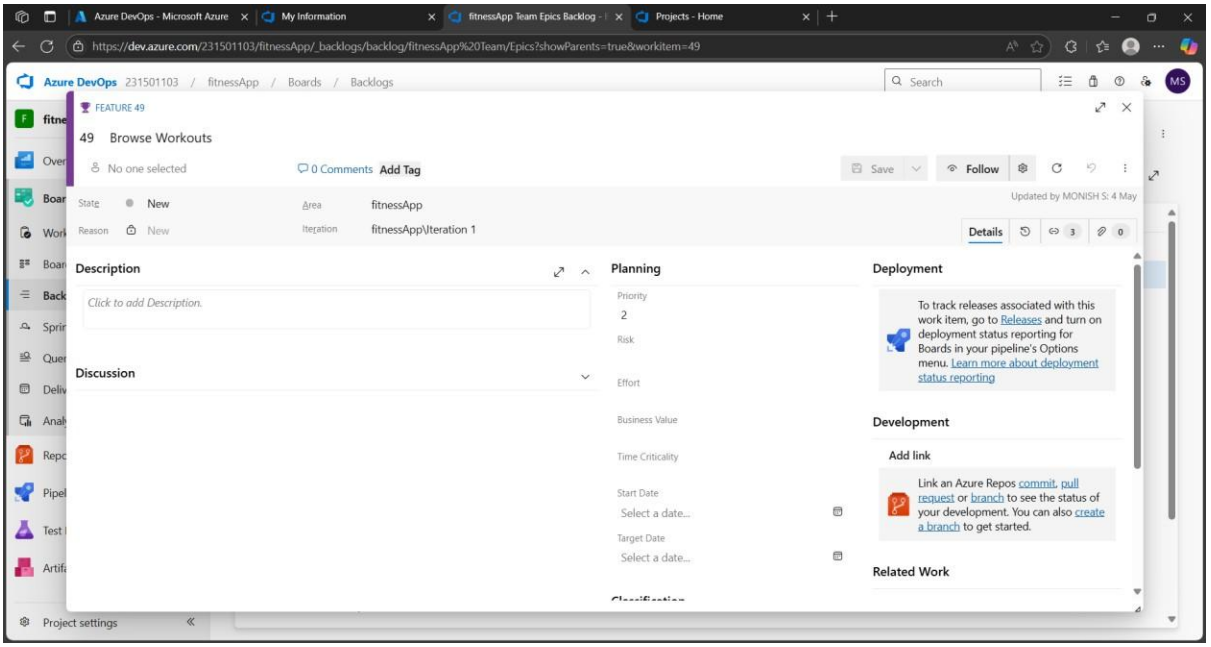
Create Epic, Features, User Stories, Task



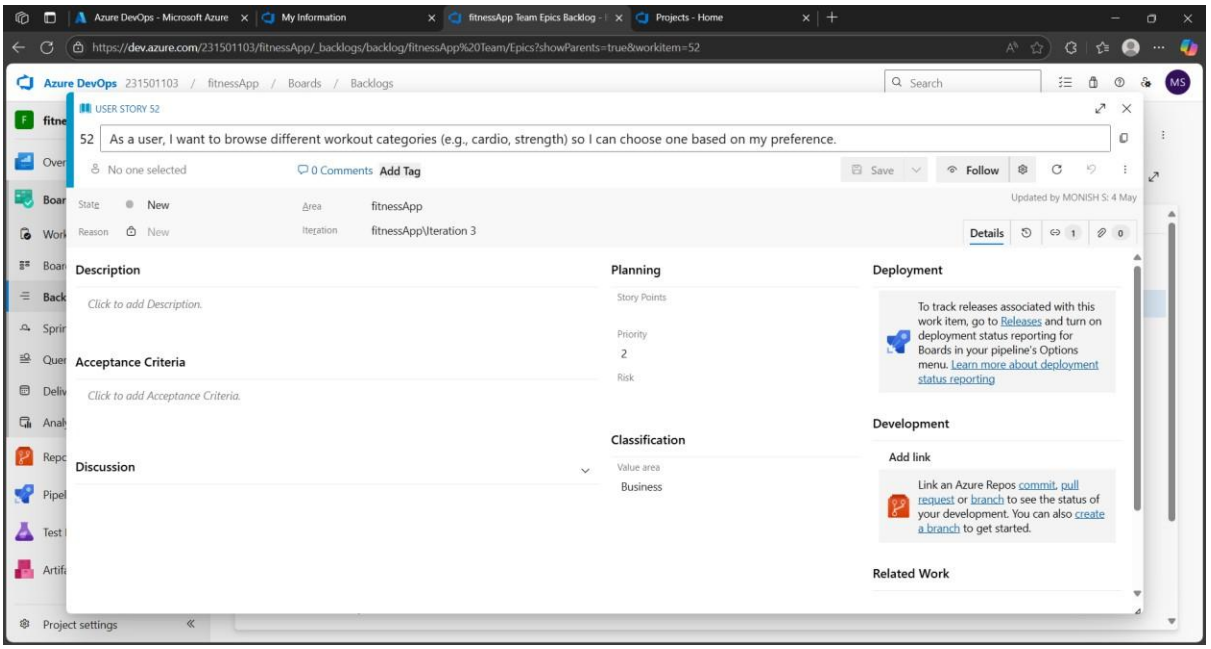
1. Fill in Epics



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.

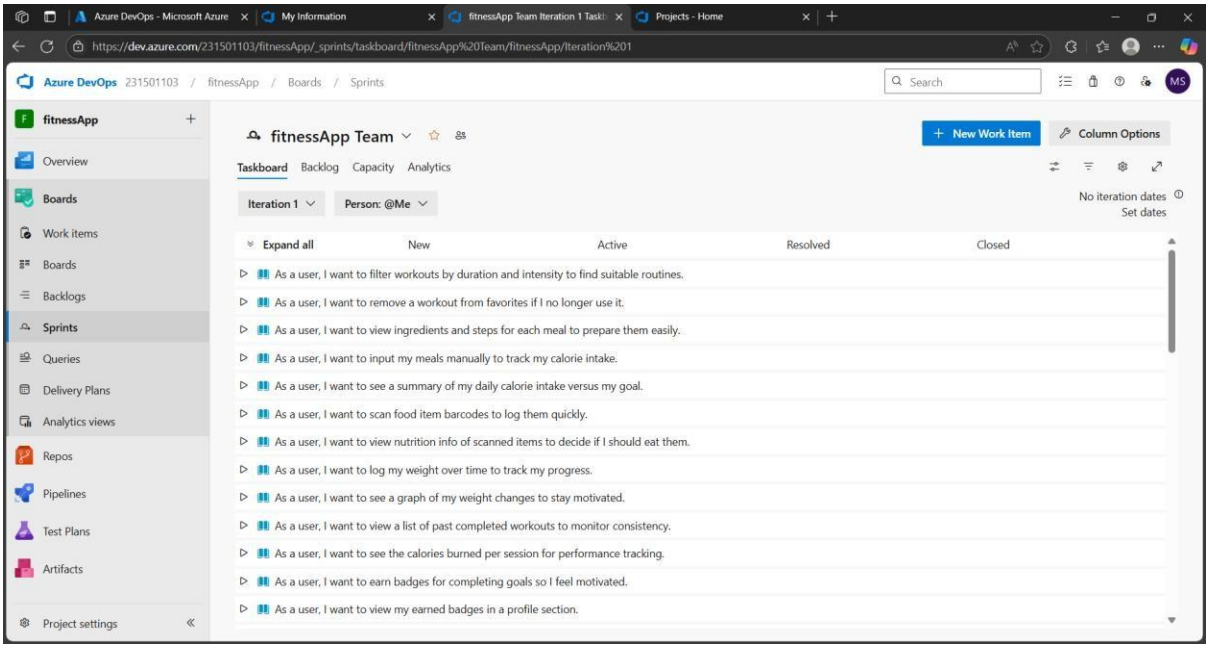
Expr no 4	Sprint Planning

Aim:

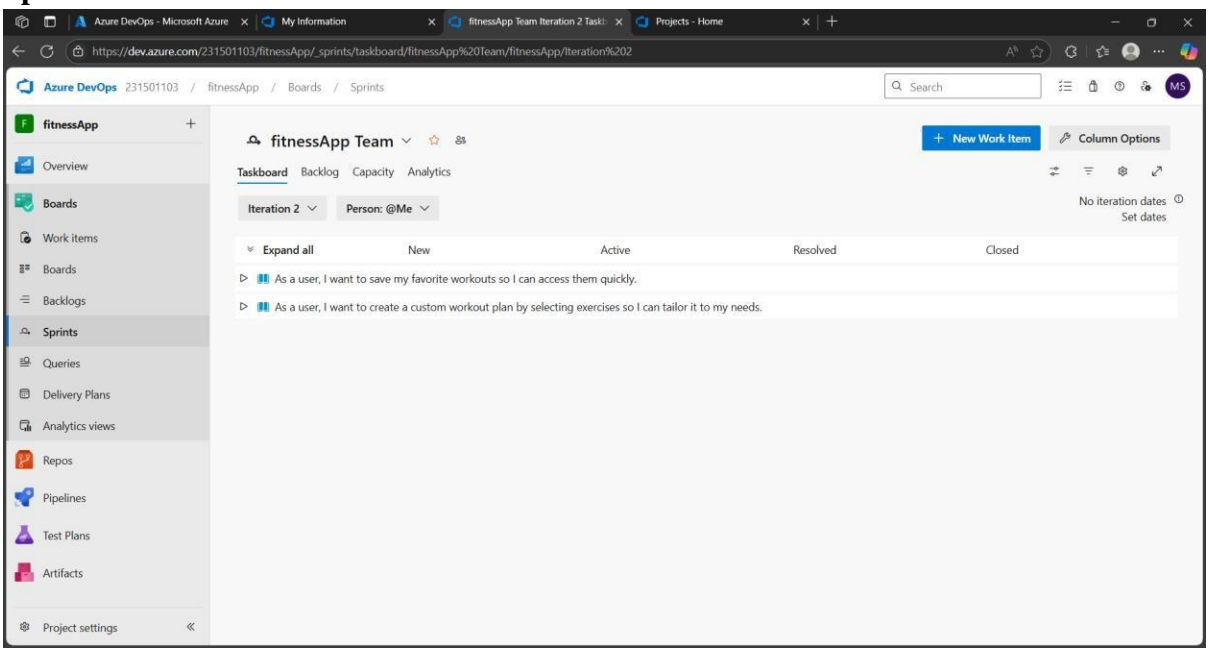
To assign user story to specific sprint for the Music Playlist Batch Creator Project.

Sprint Planning

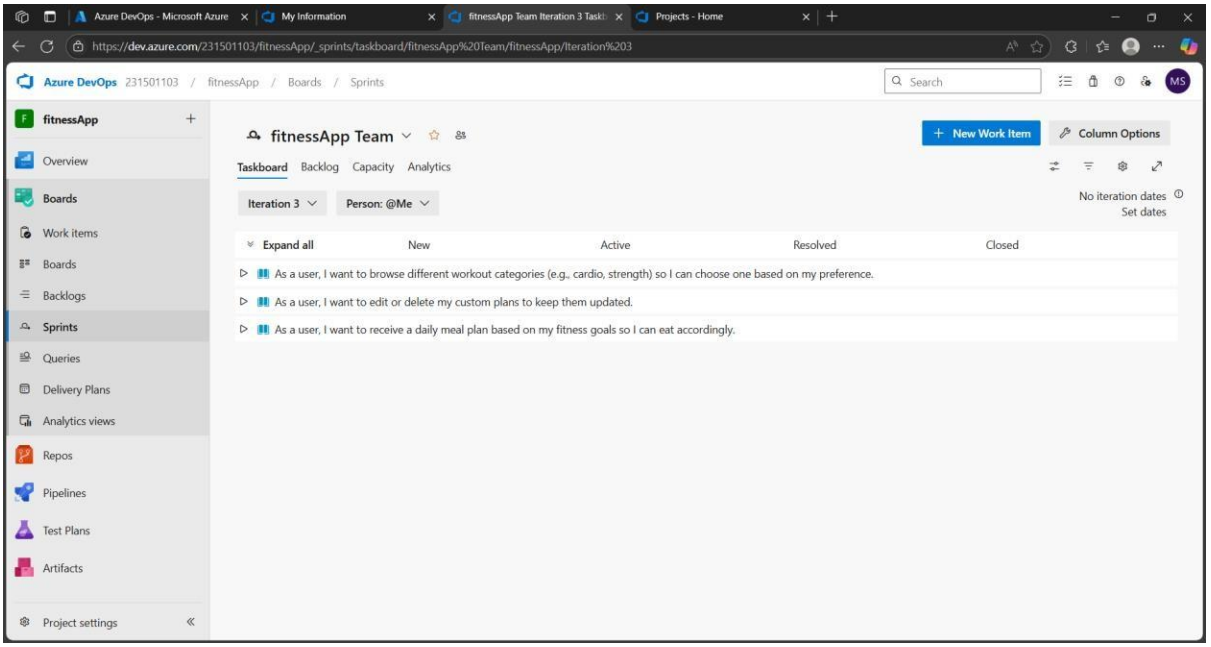
Sprint 1



Sprint 2



Sprint 3



Result:

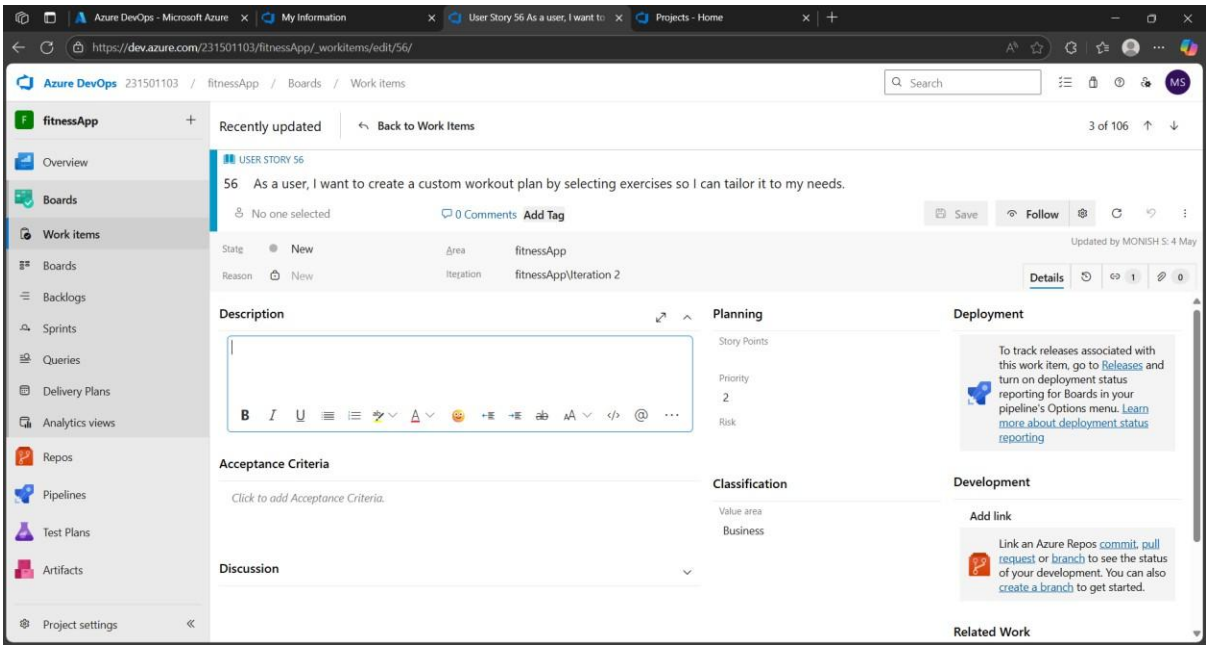
The Sprints are created for the Music Playlist Batch Creator Project.

Expr no 5	Poker Estimation

Aim:

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

Poker Estimation



Result:

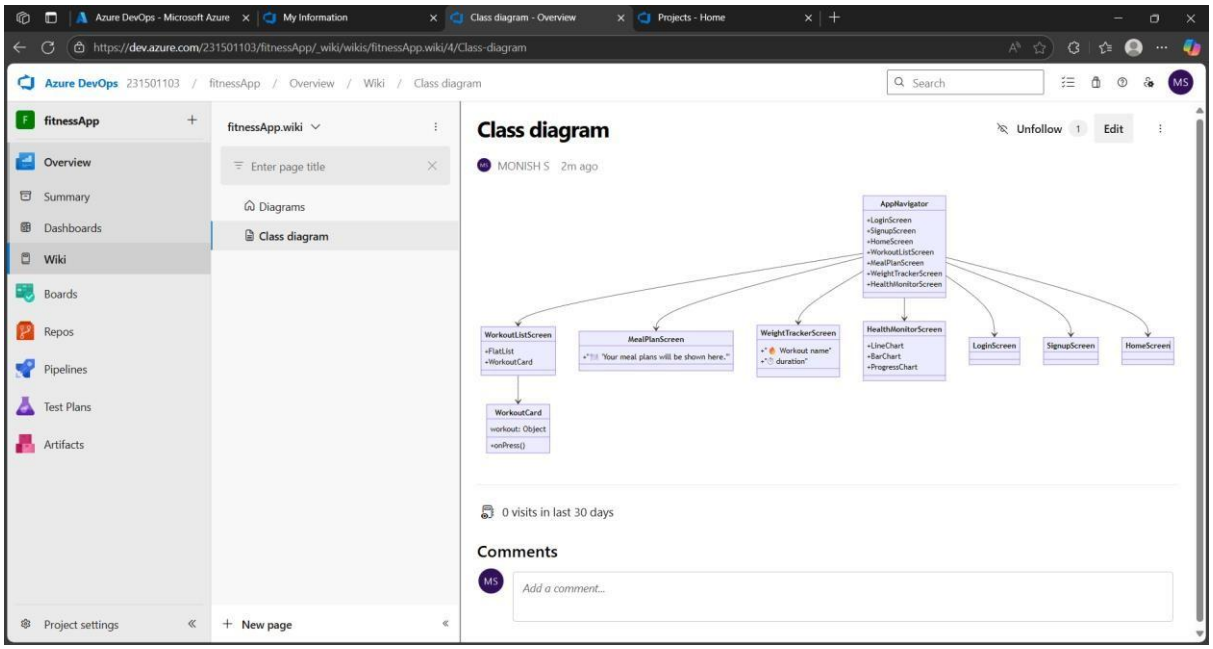
The Estimation/Story Points is created for the project using Poker Estimation.

Expr no 6	DESIGNING CLASS AND SEQUENCE DIAGRAMS FOR PROJECT ARCHITECTURE

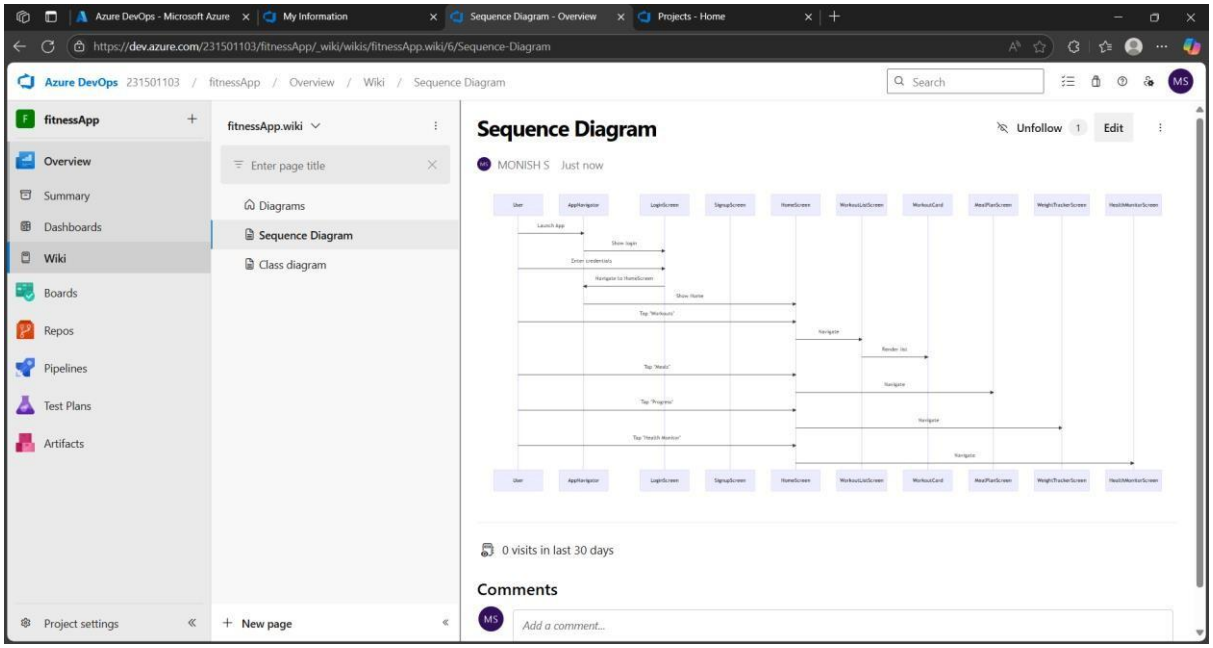
Aim

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

A. Class Diagram



B. Sequence Diagram



Result:

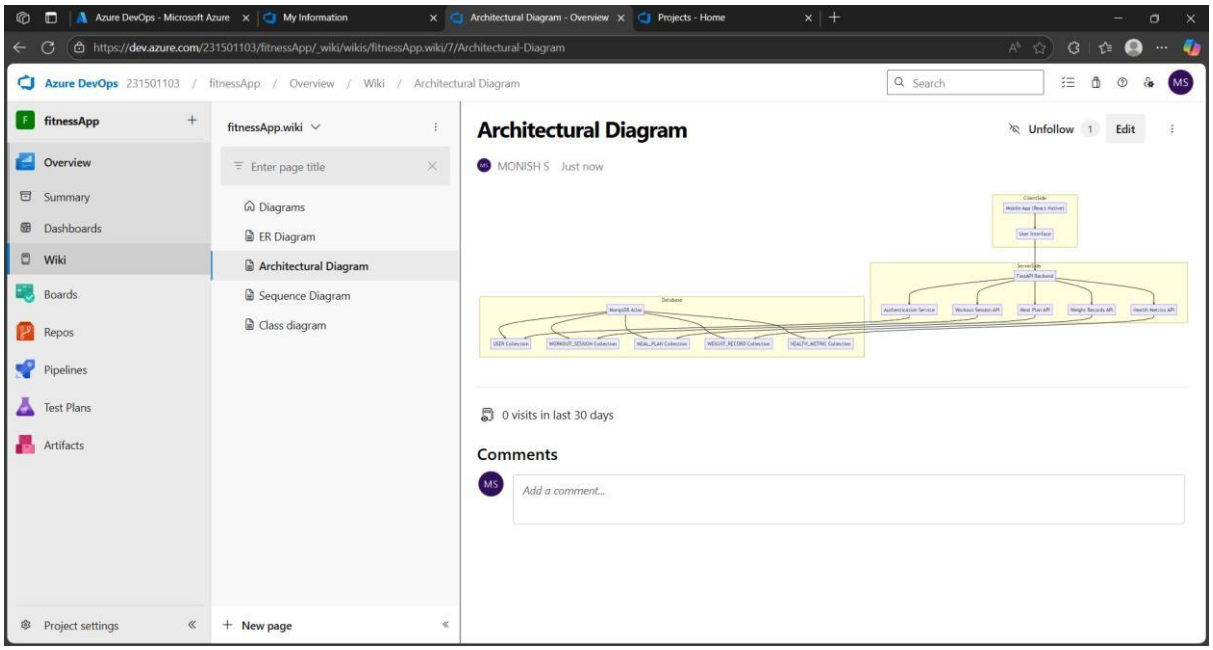
The Class Diagram and Sequence Diagram is designed Successfully for the Music Playlist Batch Creator.

Expr no 7	DESIGNING ARCHITECTURAL AND ER DIAGRAMS FOR PROJECT STRUCTURE

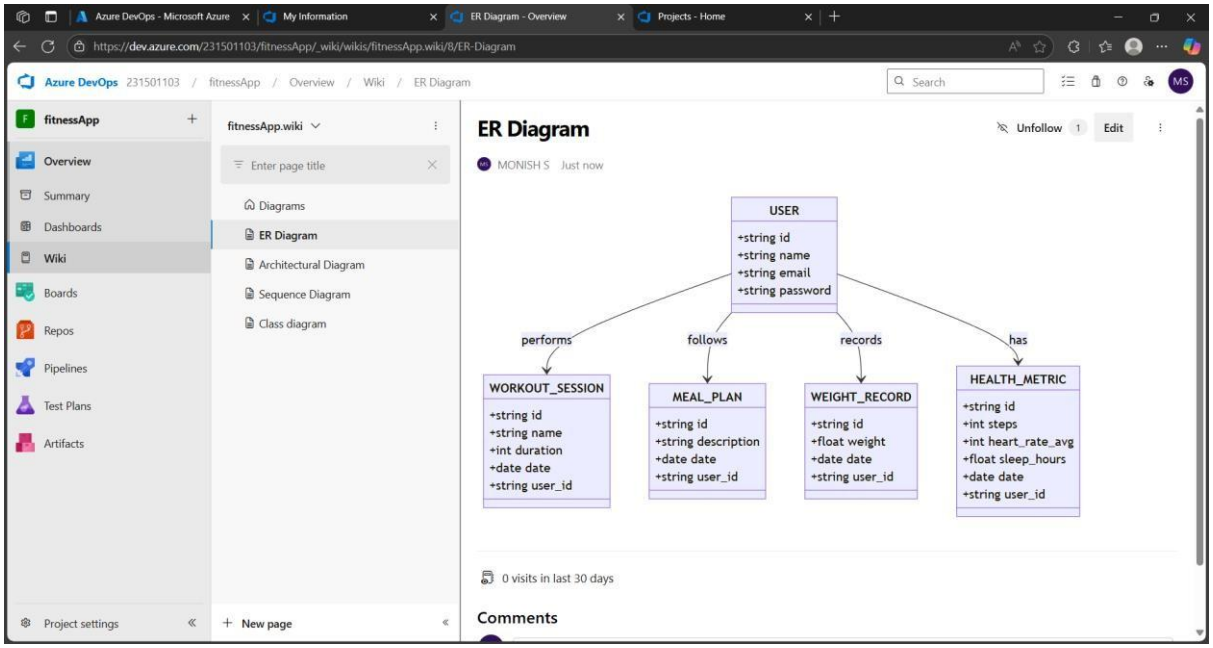
Aim:

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

A. Architectural Diagram



B. ER Diagram



Result:

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

Expr no 8	TESTING – TEST PLANS AND TEST CASES

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

- o User Signup & Login
- o Viewing and Managing Playlists
- o Fetching Real-time Metadata
- o Editing playlists (rename, reorder, record)
- o Creating smart audio playlists based on categories (mood, genre, artist, etc.)

2. Define User Interactions

- o Each test case simulates a real user behaviour (e.g., logging in, renaming a playlist, adding a song).

3. Design Happy Path Test Cases

- o Focused on validating that all features function as expected under normal conditions.
- o Example: User logs in successfully, adds item to playlist, or creates a category-based playlist.

4. Design Error Path Test Cases

- o Simulate negative or unexpected scenarios to test robustness and error handling.
- o Example: Login fails with invalid credentials, save fails when offline, no recommendations found.

5. Break Down Steps and Expected Results

- o Each test case contains step-by-step actions and a corresponding expected outcome.
- o Ensures clarity for both testers and automation scripts.

6. Use Clear Naming and IDs

- o Test cases are named clearly (e.g., TC01 – Successful Login, TC10 – Save Playlist Fails).
- o Helps in quick identification and linking to user stories or features.

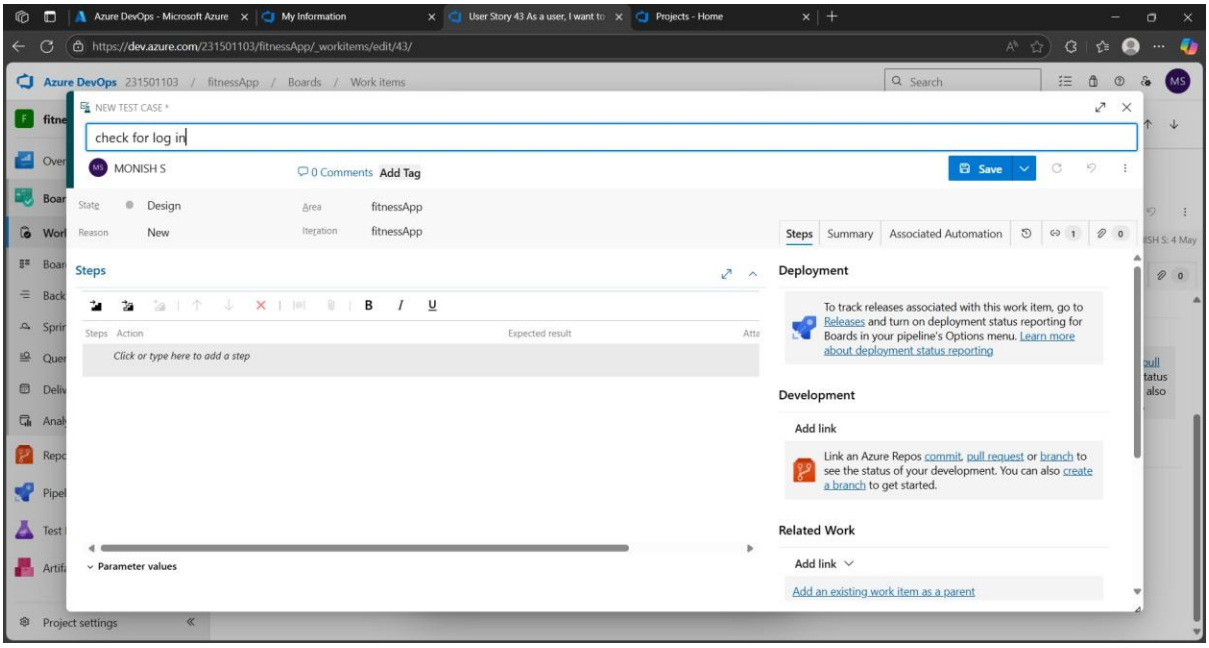
7. Separate Test Suites

- o Grouped test cases based on functionality (e.g., Login, Playlist Editing, Recommendation System).
- o Improves organization and test execution flow in Azure DevOps.

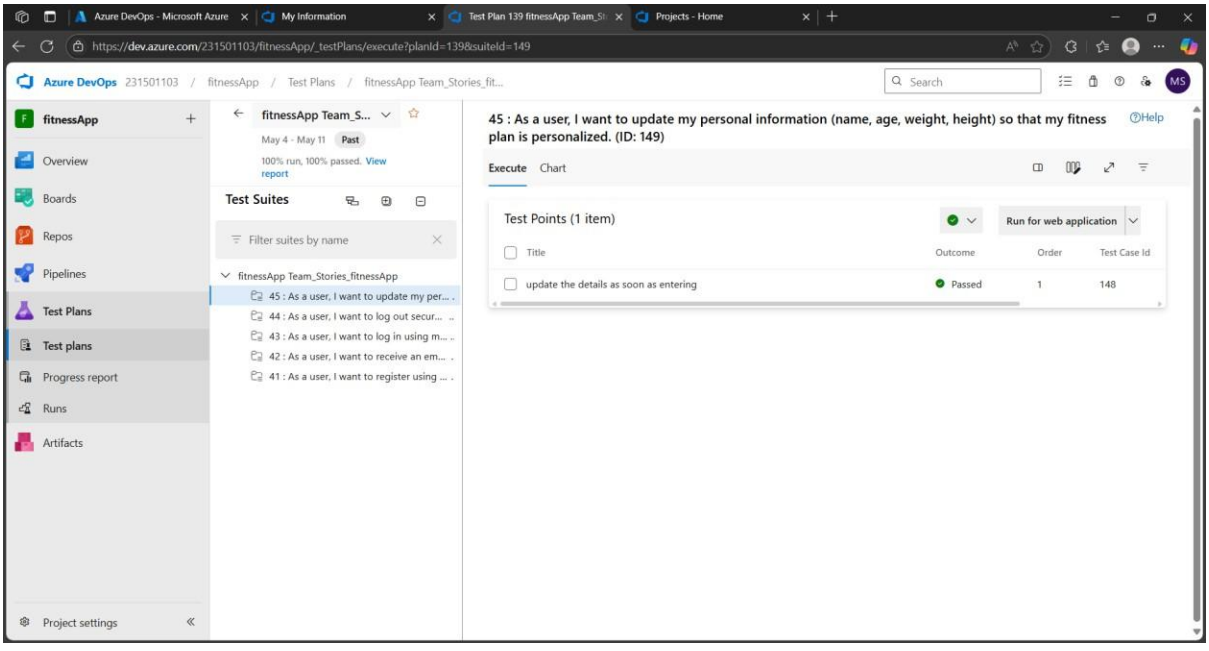
8. Prioritize and Review

o Critical user actions are marked high-priority.
o Reviewed for completeness and traceability against feature requirements.

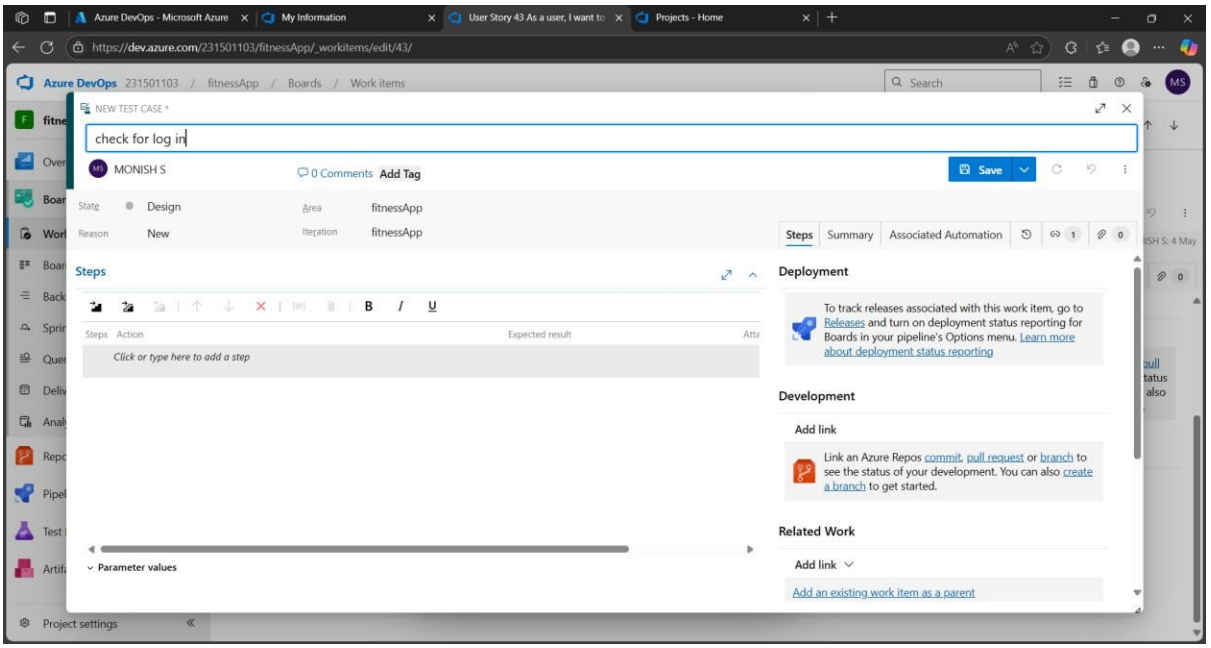
1.New test plan



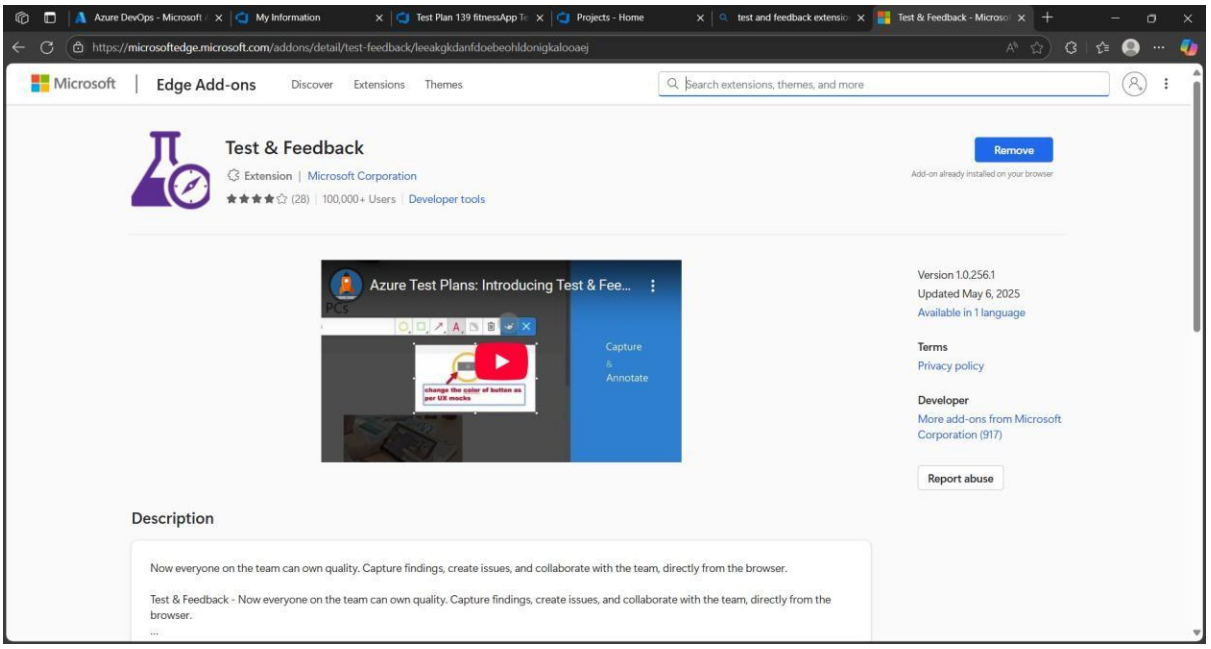
2.Test suite



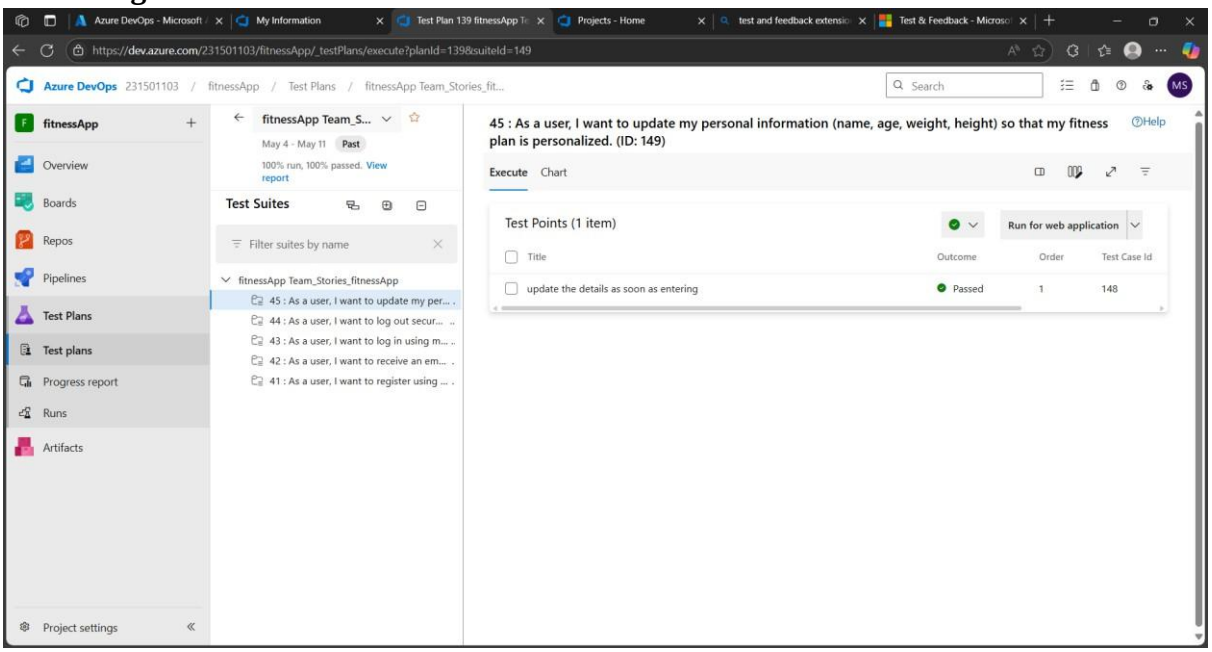
3.Test Cases

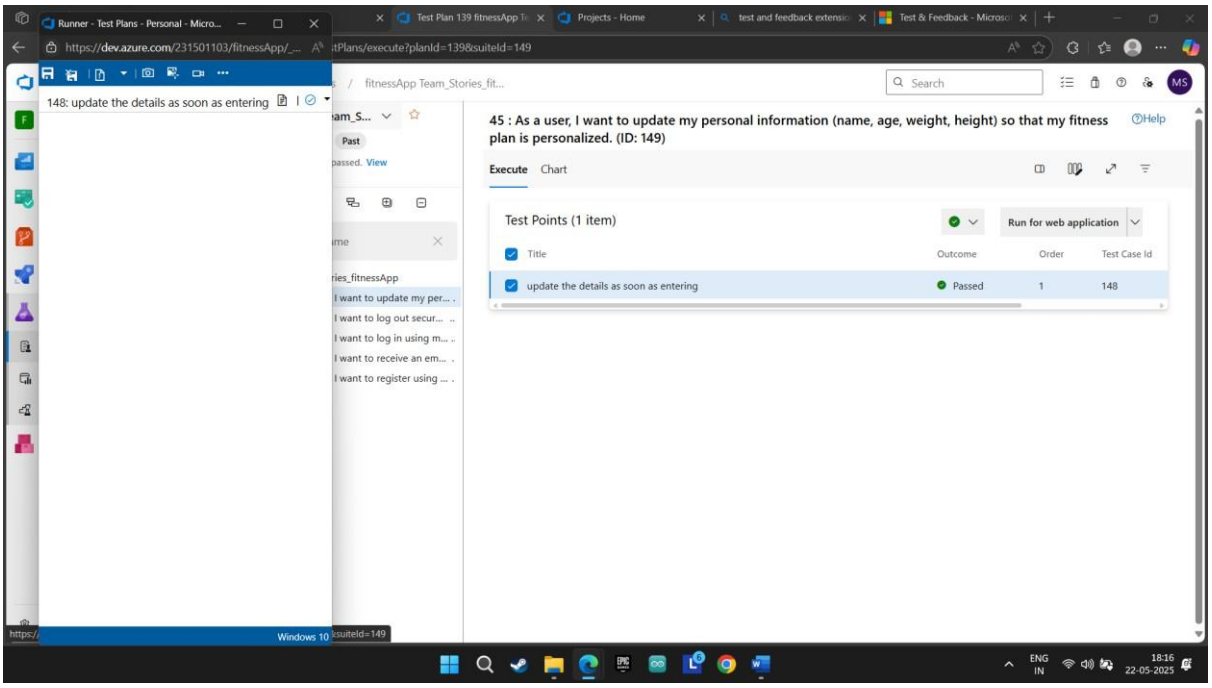


4. Installation of test

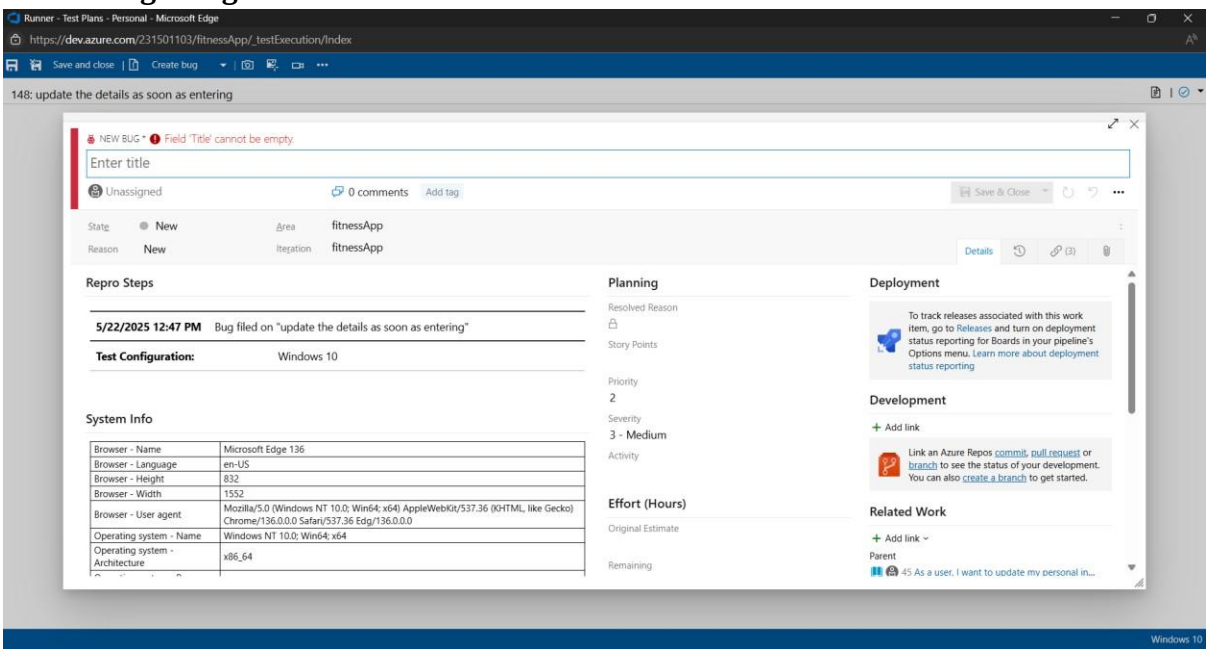


5. Running the test cases

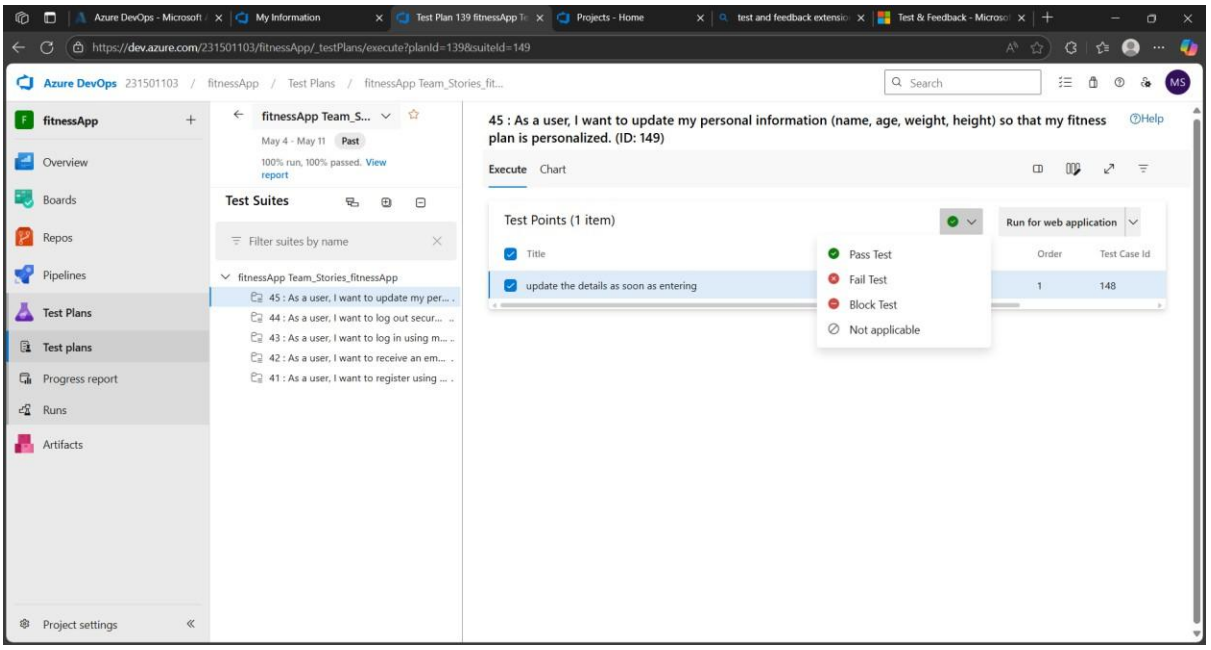




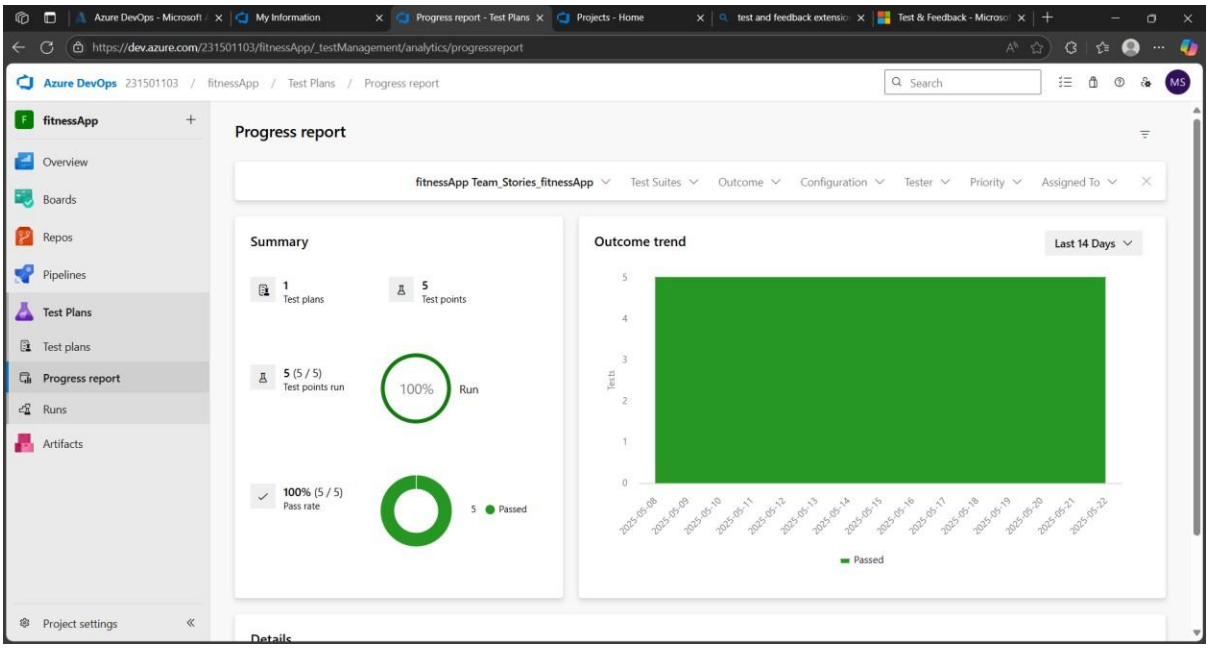
6. Creating a bug



7. Test Case Result



8. Progress Report



Result:

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

Expr no 9	**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, select your organization, and open the project where your Student Management System code resides.

2. 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.

3. 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.

4. 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.

5. Define Build Stage (CI - Continuous Integration) from YAML file.

6. Install dependencies (e.g., npm install, dotnet restore).

7. Build the application (dotnet build, npm run build).

8. Run unit tests (dotnet test, npm test).

9. Publish build artifacts to be used in the release stage.

10. 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.

11. 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).

12. 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.

13. Set Triggers and Approvals

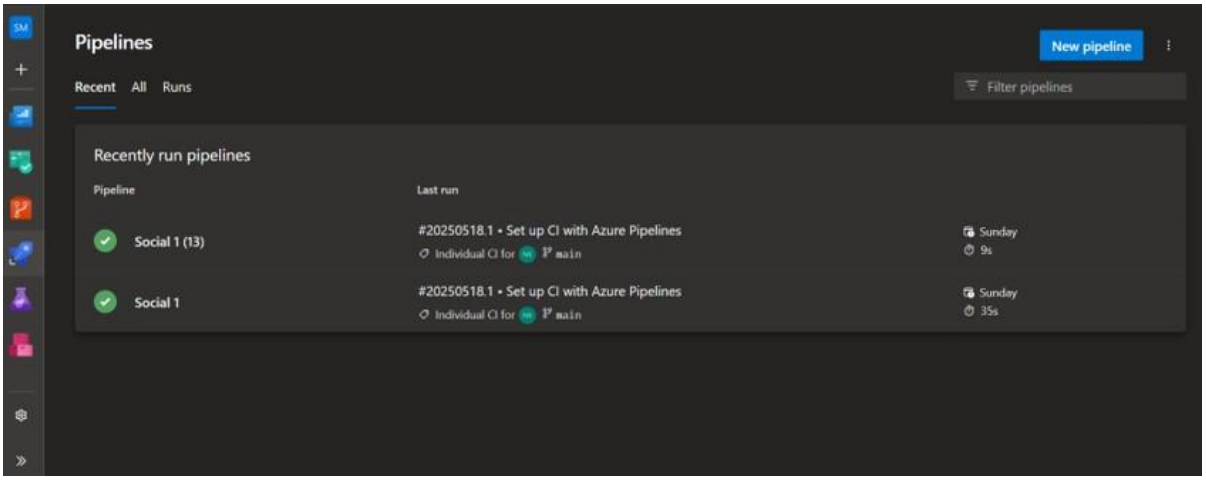
Enable continuous deployment trigger so the release pipeline runs automatically after a successful build. For production environments, configure pre-deployment approvals to ensure manual verification before release.

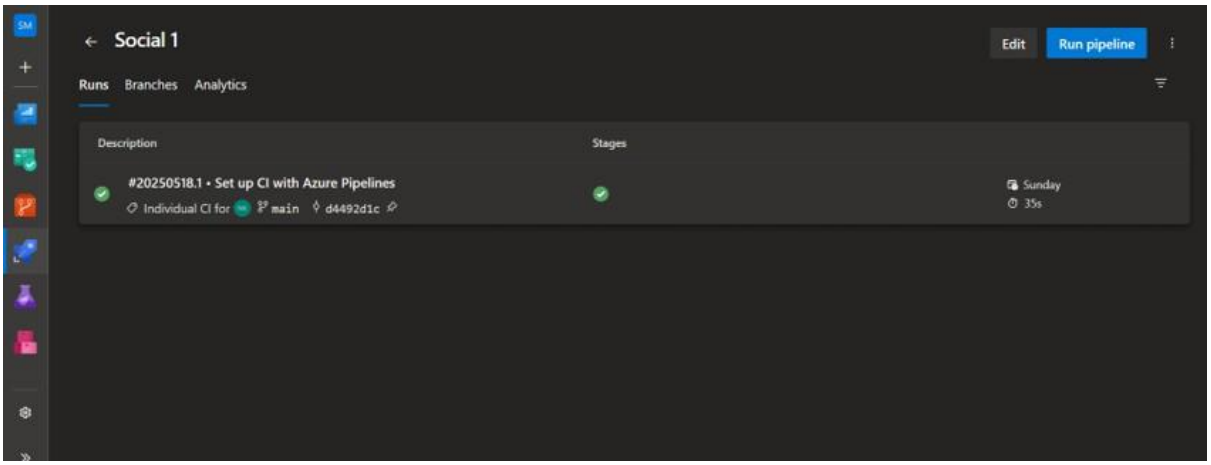
14. Monitor Pipelines and Manage Logs

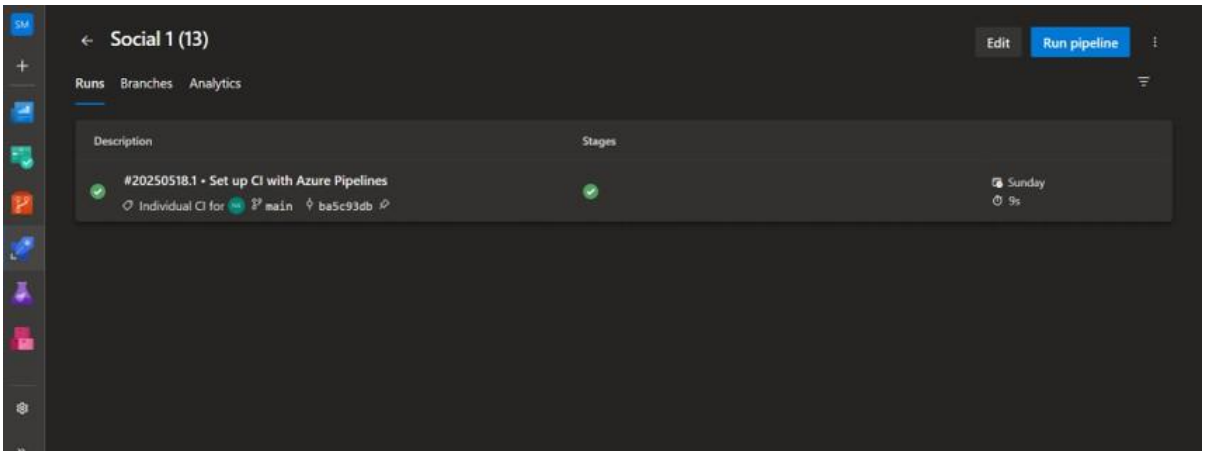
View all pipeline runs under the Runs section.
Check logs for build/test/deploy stages to debug any errors.
You can also integrate email alerts or Microsoft Teams notifications for build failures.

15. 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:

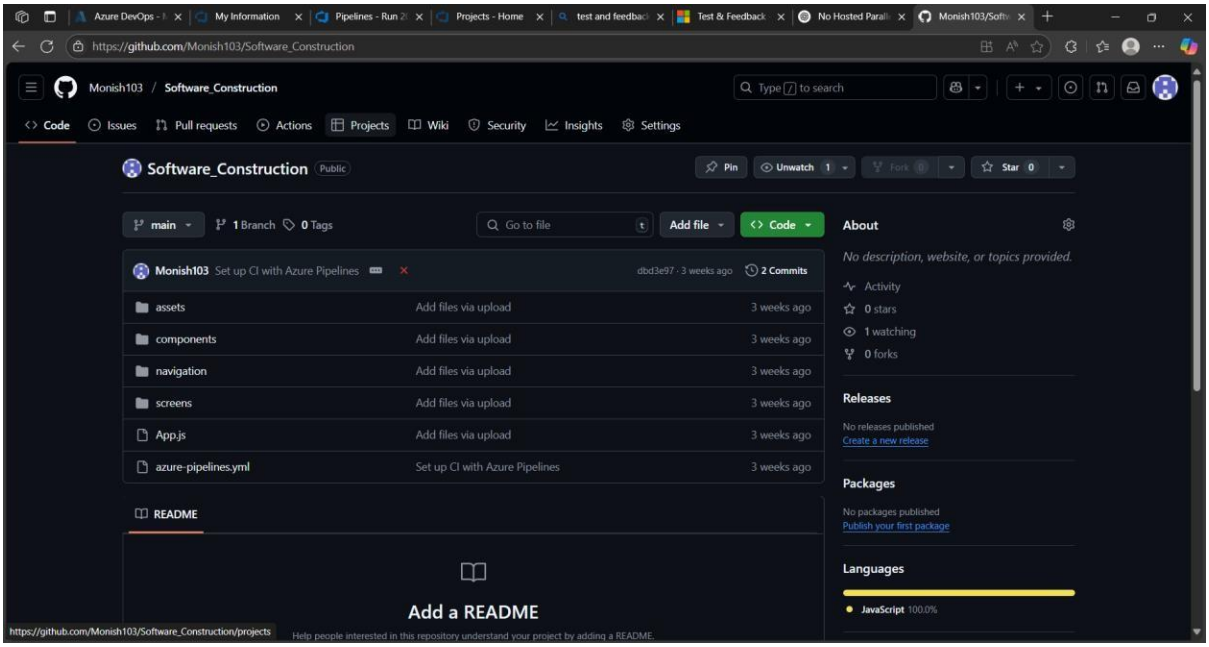
Thus, the pipelines for the given project “Online Quiz System” has been executed successfully.

Expr no10	GITHUB: PROJECT STRUCTURE & NAMING CONVENTIONS

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.