INDEX

EXP NO: 1	AZURE DEVOPS ENVIRONMENT SETUP

Aim:

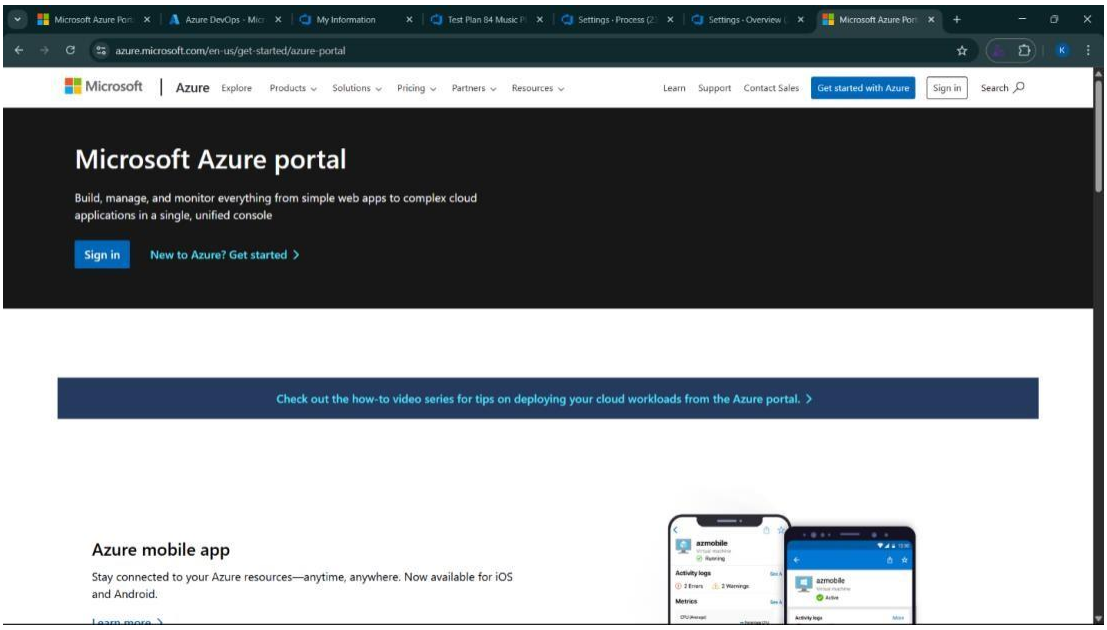
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

INSTALLATION

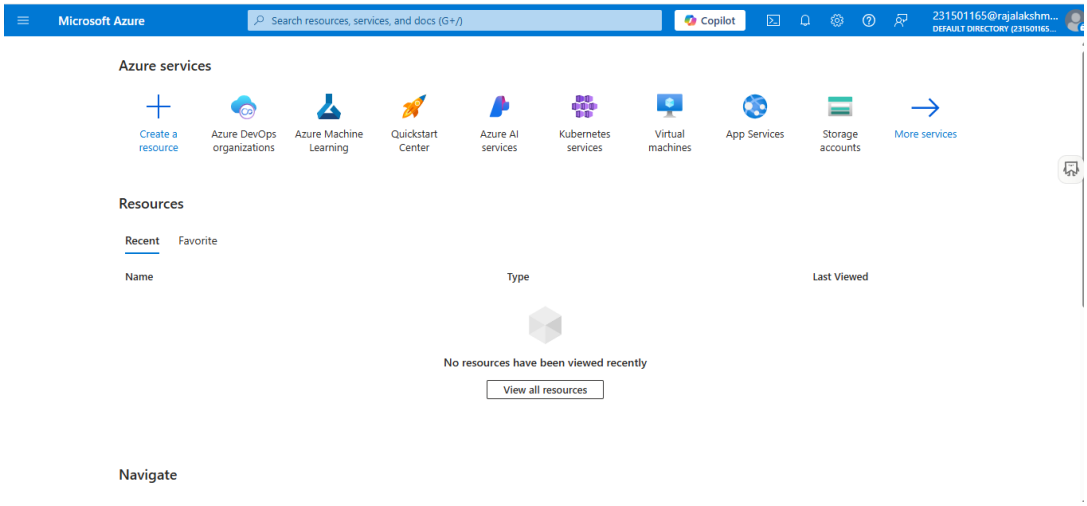
1. Open your web browser and go to the Azure website: https://azure.microsoft.com/en-us/get-started/azure-portal.

Sign in using your Microsoft account credentials.

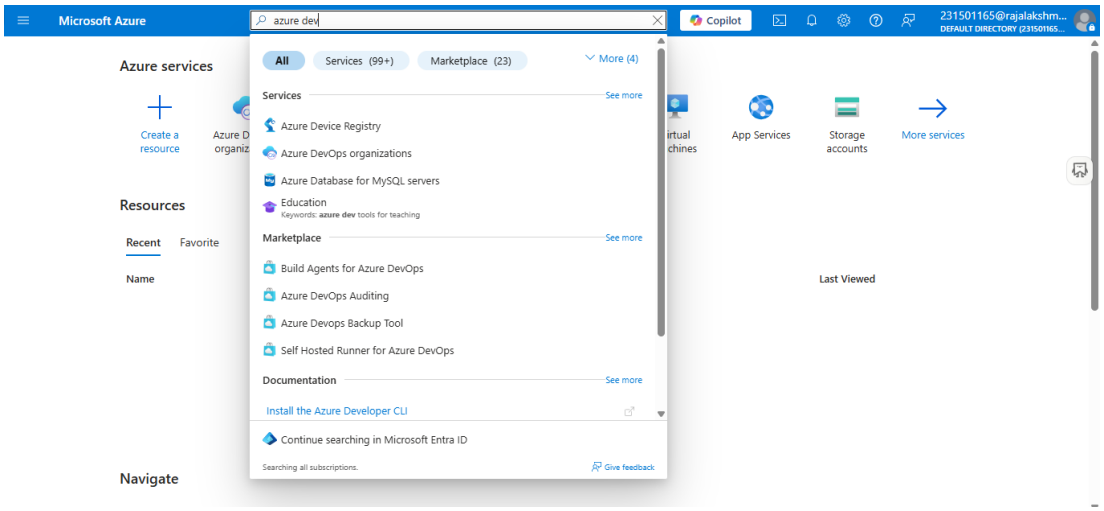
If you don't have a Microsoft account, you can create one here: https://signup.live.com/?lic=1



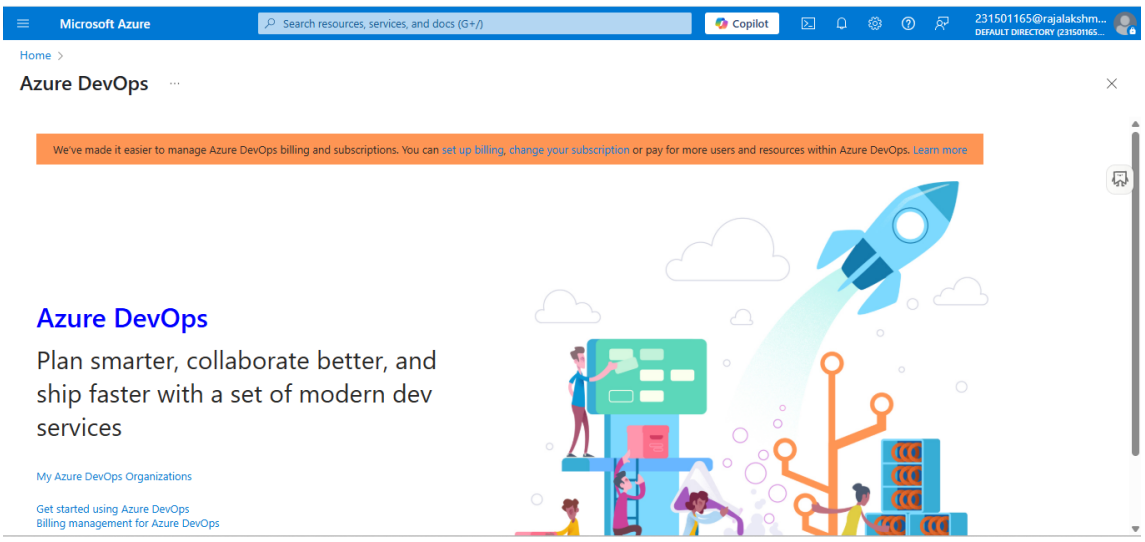
2. Azure home page



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:

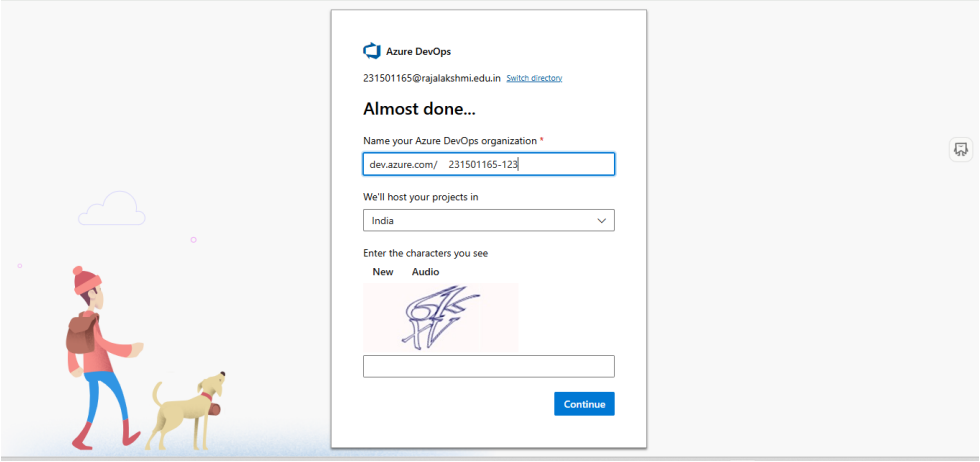
Successfully accessed the Azure DevOps environment and created a new organization through the Azure portal.

EXP 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.

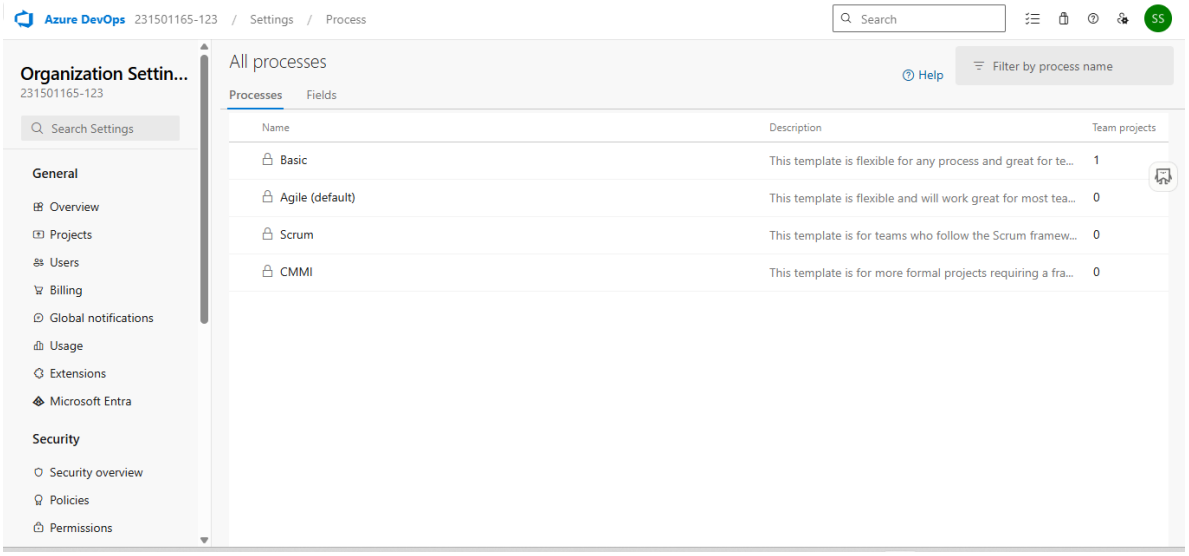
c. Enter the project name, description, and visibility options:

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

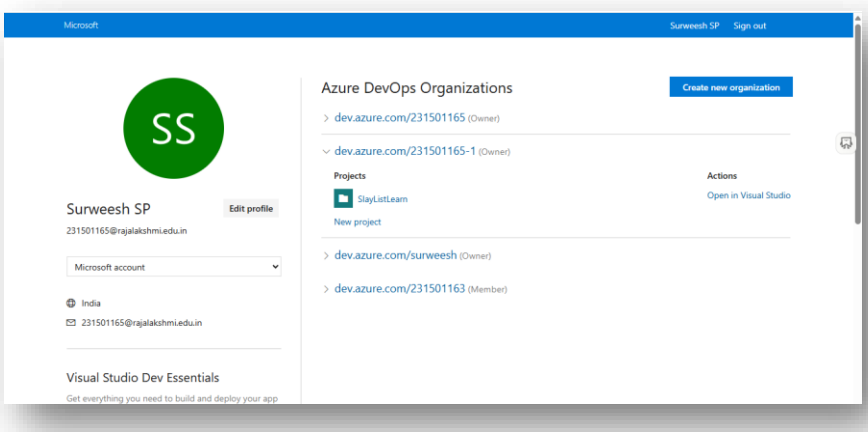
Description: Optionally, add a description to provide more context about the project.

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

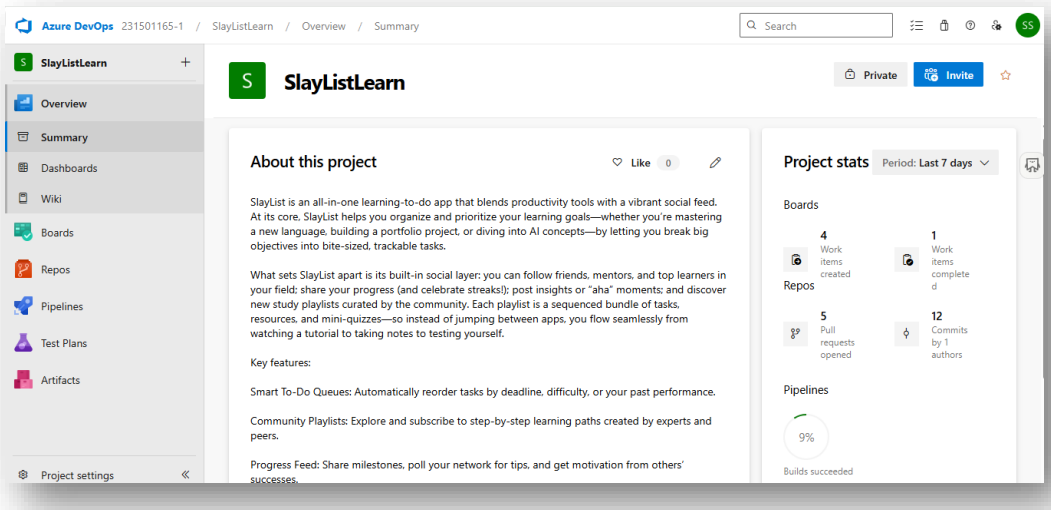
d. Once you've filled out the details, click **Create** to set up your first project.



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

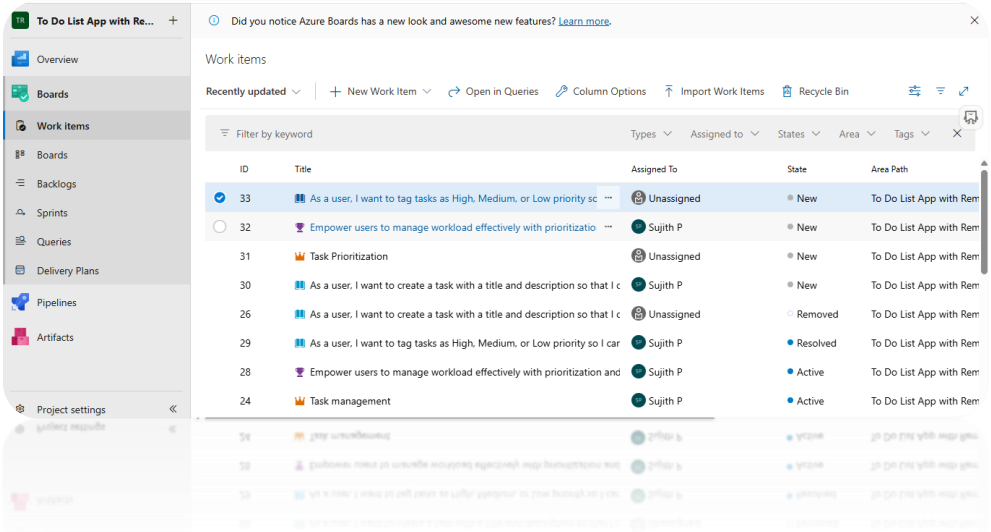


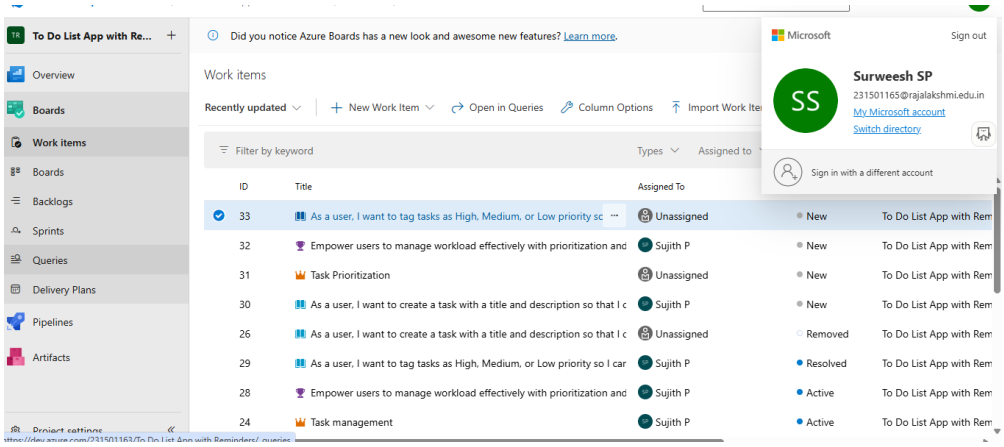
5. To manage user stories:

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.

c.





Results:

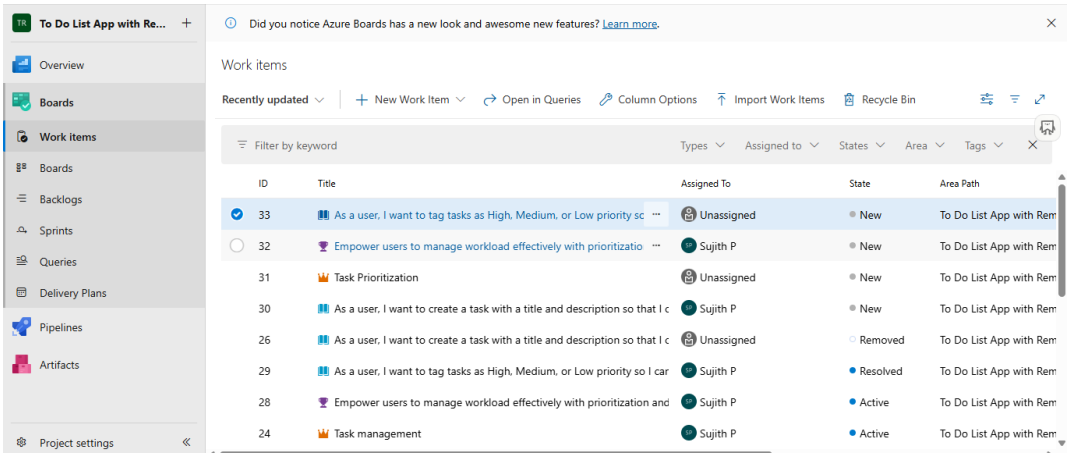
Successfully created an Azure DevOps project with user story management and agile workflow setup.

EXP NO: 3	SETTING UP EPICS, FEATURES, AND USER STORIES FOR PROJECT PLANNING

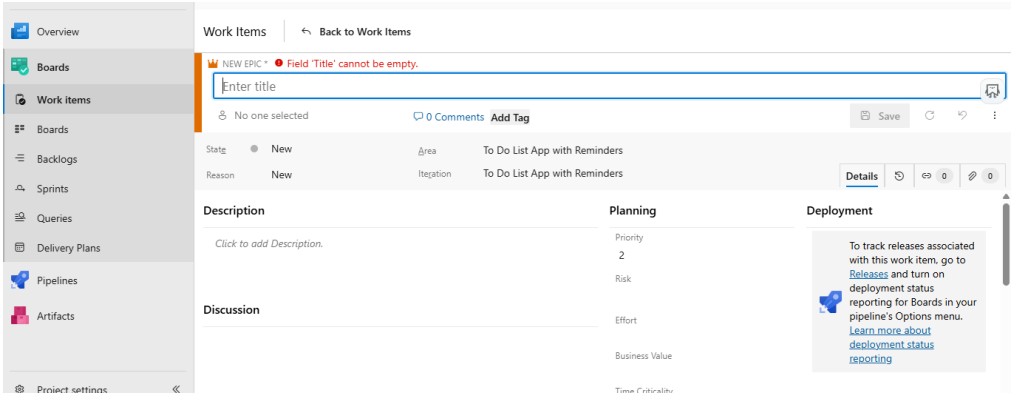
Aim:

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

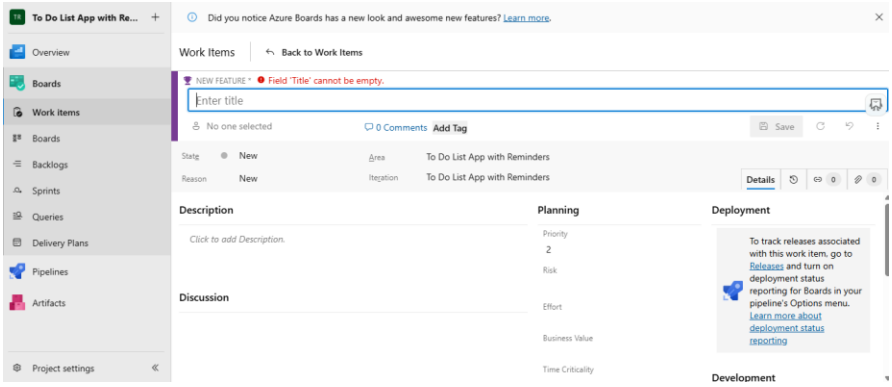
Create Epic, Features, User Stories, Task



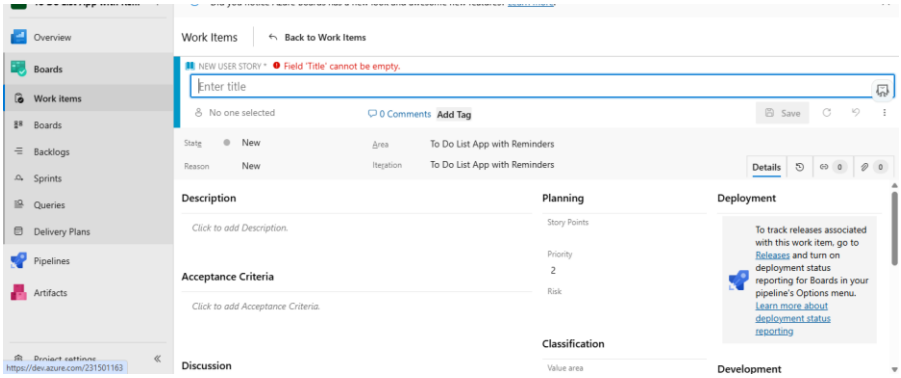
1. Fill in Epics



2. Fill in Features



3. Fill in User Story Details



Result:

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

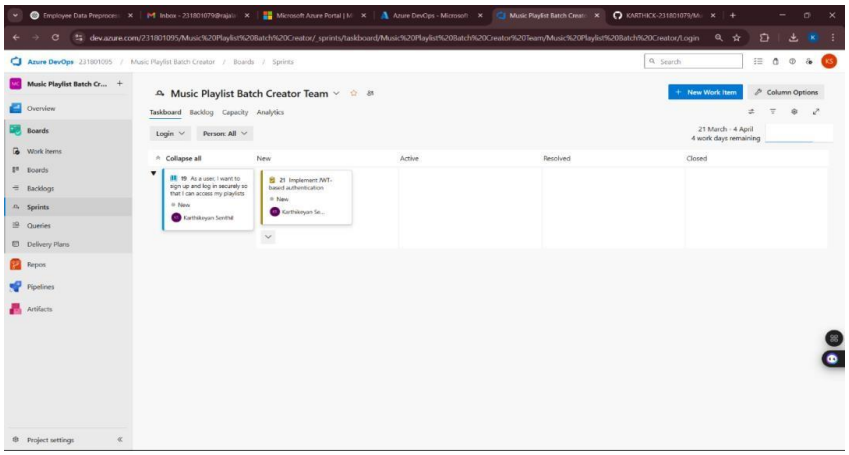
EXP NO: 4	SPRINT PLANNING

Aim:

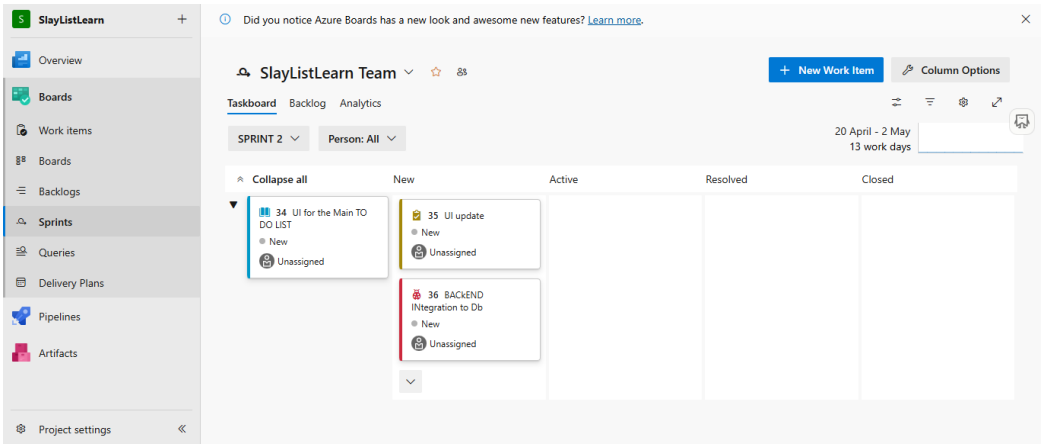
To assign user story to specific sprint for the TO DO List Project.

Sprint Planning:

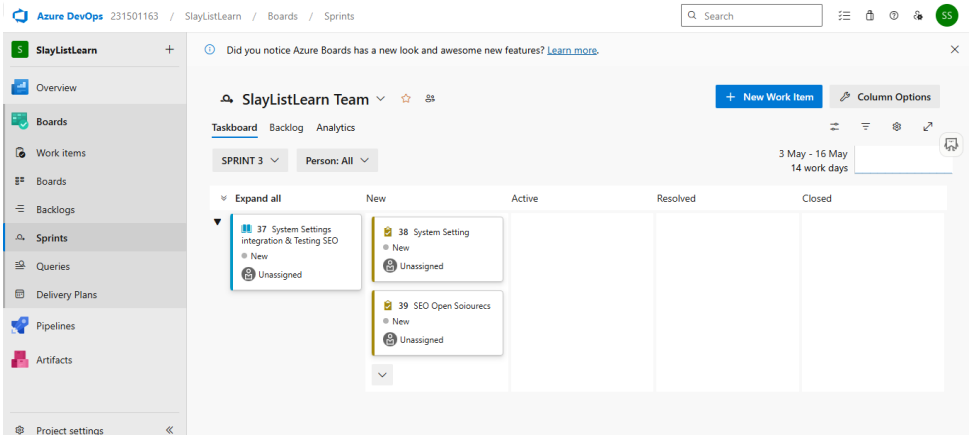
Sprint 1:



Sprint 2



Sprint 3



Result:

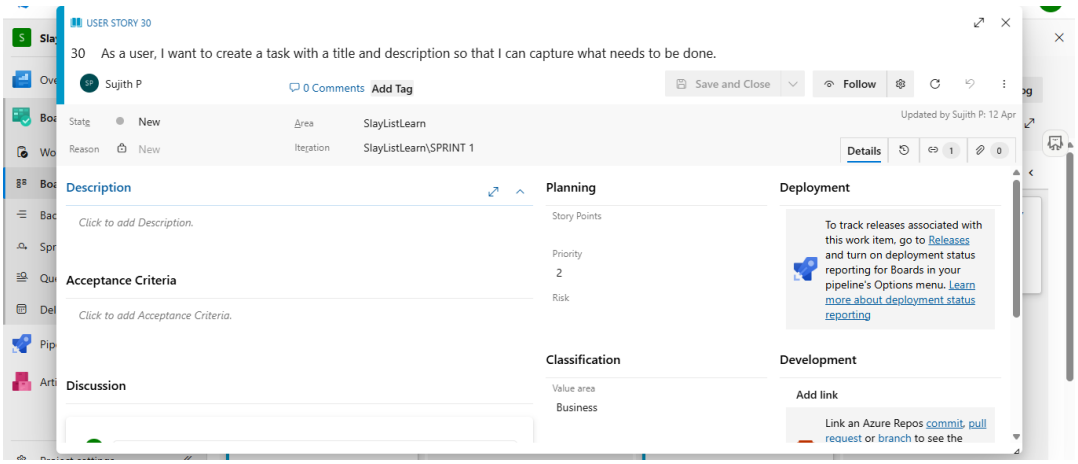
The Sprints are created for the To Do List Project.

EXP NO: 5	POKER ESTIMATION

Aim:

Create Poker Estimation for the user stories – TO DO List Project.

Poker Estimation



Result:

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

EXP 6	FUNCTIONAL AND NON-FUNCTIONAL REQUIEMENTS FOR THE PROJECT

Aim:

To Design a Functional and Non-Functional |Requirement's for the given Project.

Functional Requirements (What the system should do)

1. User Management

- Users can register, log in, and log out.
- Support for user authentication and session management.
- Password encryption and security features.

2. Task Management

- Users can create, read, update, and delete to-do items.
- Tasks can have attributes: title, description, deadline, priority, tags.
- Tasks can be marked as completed or pending.

3. Intelligent AI Features (AMI Integration)

- AI can categorize tasks (e.g., Work, Personal, Health) using NLP.
- Emotional Analysis: System captures mood/emotion from text (e.g., "I feel tired") using sentiment analysis.
- AMI (Artificial Mental Interface) suggests tasks based on emotional input or productivity pattern.
- Predictive input: Auto-suggest task titles based on previous entries or context.
- Reminders and follow-ups powered by ML-based priority and time analysis.

4. Notifications and Reminders

- Set and receive task reminders via email or in-app notification.
- Smart reminders based on urgency and frequency of missed deadlines.

5. Data Storage and Retrieval

- Securely store to-do items in a database (e.g., Supabase, MongoDB).
- Fetch and display past entries with filters (by date, tag, emotion, status).

6. Visualization & Insights

- Graphical dashboard showing:
  - Task completion rates
  - Mood trends over time
  - Productivity heatmaps
- Time tracking for tasks and productivity scoring.

Non-Functional Requirements (How the system should behave)

1. Performance

- The system should respond to user interactions (e.g., create/edit task) within 2 seconds.
- AI model inference (e.g., mood detection) should complete within 3–5 seconds.

2. Scalability

- The system should support at least 100 concurrent users in its experimental phase.
- Backend should handle task and emotion analysis efficiently as user base grows.

3. Security

- Use HTTPS for all communications.
- Ensure secure storage of user credentials with hashing (e.g., bcrypt).
- Implement token-based authentication (e.g., JWT).

4. Availability & Reliability

- System should be available 99% of the time during working hours (9 AM to 9 PM).
- Auto-recovery from crashes (e.g., via containerized deployment or auto-scaling).

5. Maintainability

- Code should follow modular and clean architecture.
- Easy logging and error tracking for debugging (e.g., using Winston or Sentry).

6. Usability

- User interface should be simple, intuitive, and responsive.
- The AI and emotional suggestions should be non-intrusive and user-controllable.

7. Compliance & Ethics

- Ensure transparency in AI decisions (explainable AI).
- Follow data privacy regulations (e.g., GDPR compliance for EU users).
- Log AI feedback loops for accountability in AMI interactions.

Result:

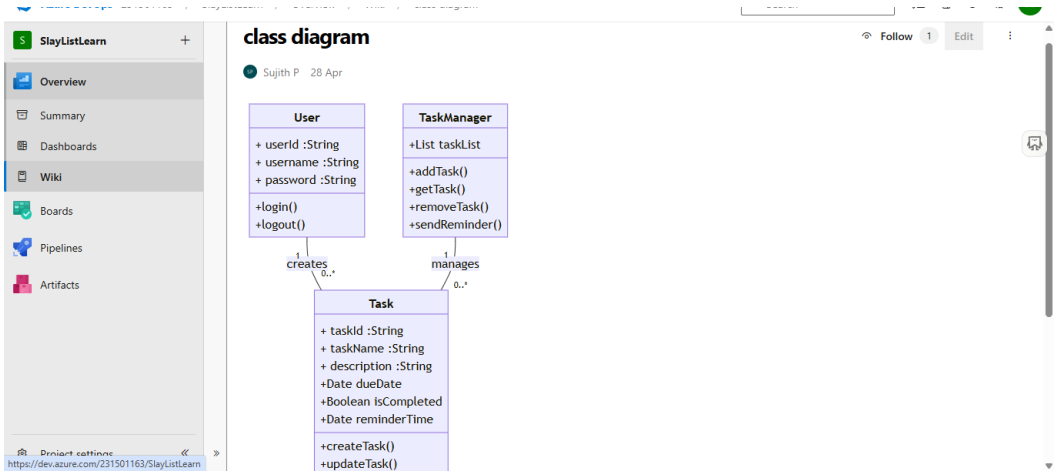
The Estimation/Story Points is created for the project using Poker Estimation.

EXP NO: 7	DESIGNING CLASS AND SEQUENCE DIAGRAMS FOR PROJECT ARCHITECTURE

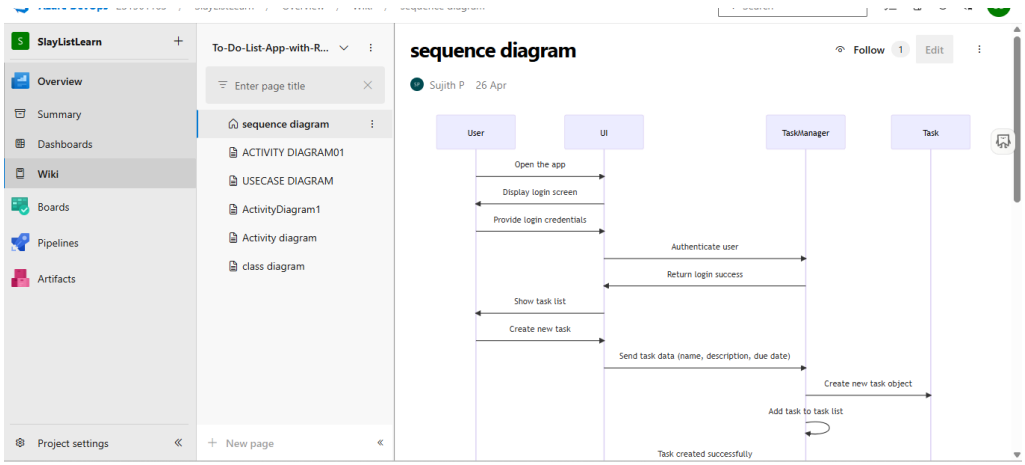
Aim:

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

7A. Class Diagram



7B. Sequence Diagram



Result:

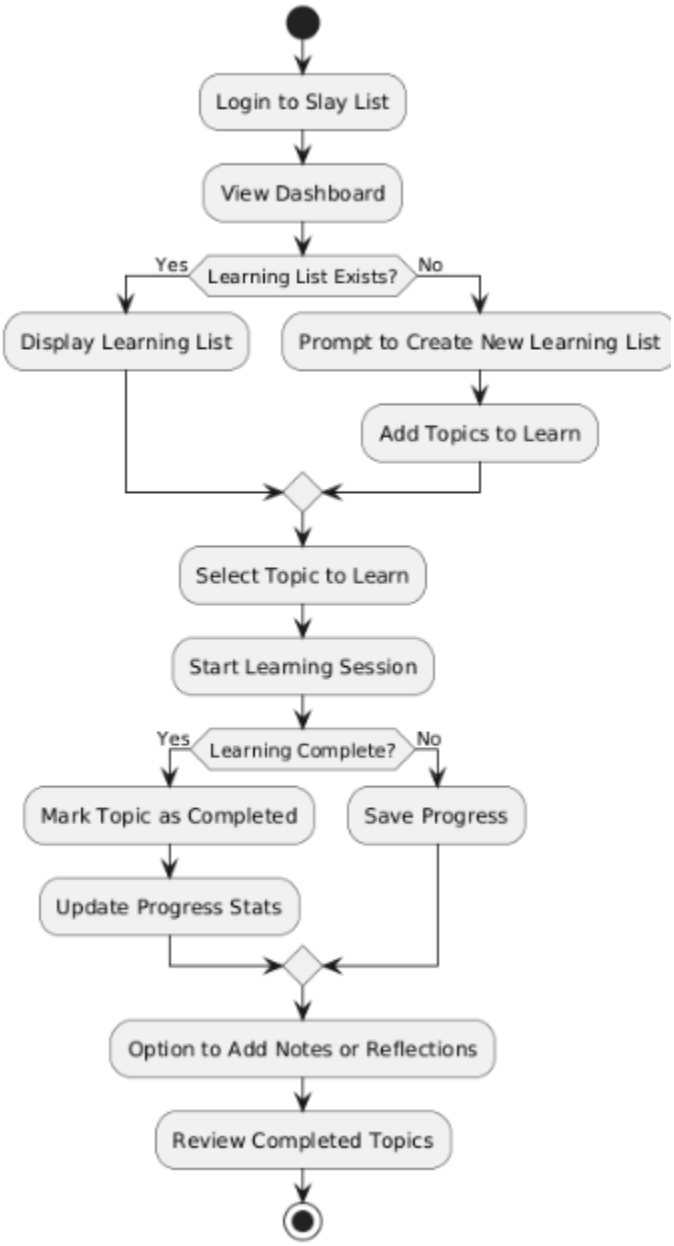
The Class Diagram and Sequence Diagram is designed Successfully for the TO DO List.

EXP NO: 8	DESIGNING ACTIVITY AND USE CASE DIAGRAMS FOR PROJECT ARCHITECTURE

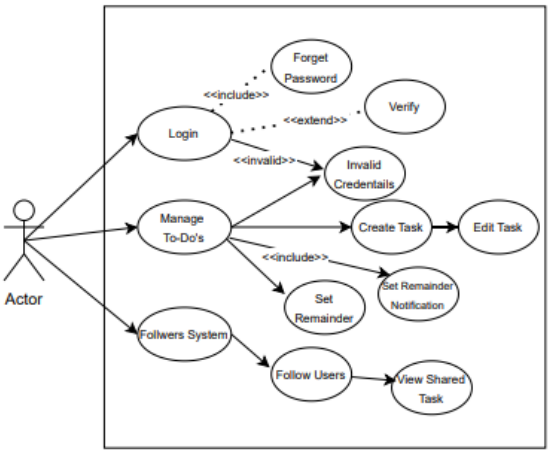
Aim:

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

8A. Activity Diagram



8B. Use Case Diagram



Result:

The Activity and the Use Case Diagram for the given Project is successfully executed.

EXP NO: 9	TESTING – TEST PLANS AND TEST CASES

Aim:

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

Test Planning and Test Case
Test Case Design Procedure

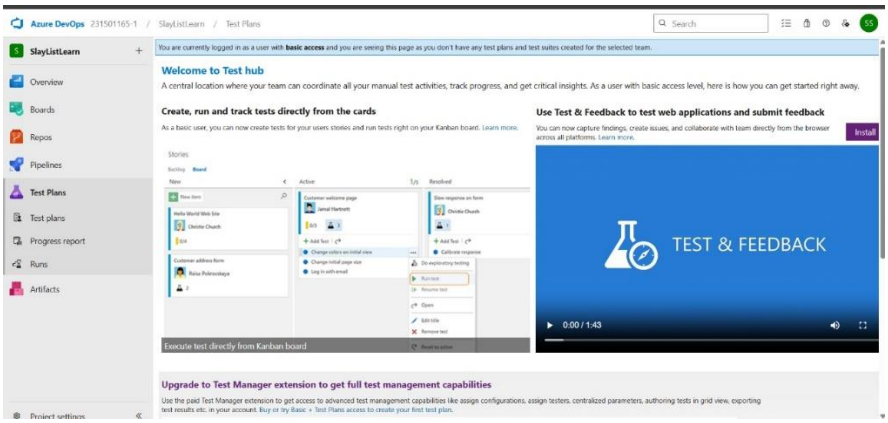
1. **Understand Core Features of the Application**
   - User Signup & Login
   - Viewing and Managing Playlists
   - Fetching Real-time Metadata
   - Editing playlists (rename, reorder, record)
   - Creating smart audio playlists based on categories (mood, genre, artist, etc.)
2. **Define User Interactions**
   - Each test case simulates a real user behavior (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.

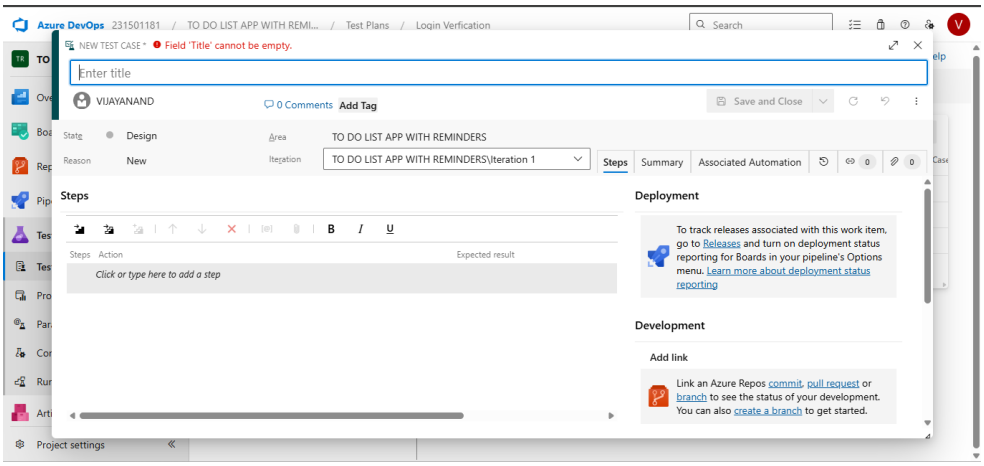
8. Prioritize and Review

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

1.New test plan



2.Test suite



3. Test case

Give two test cases for at least five user stories showcasing the happy path and error scenarios in azure DevOps platform.
Music Playlist Batch Creator – Test Plans

USER STORIES

- As a user, I want to sign up and log in securely so that I can access my playlists (ID: 79).
- As a user, I need to see my playlist in one place (ID: 76).
- As a user, I should be able to create an audio playlist as needed (ID: 73).
- As a user, I should be able to rename, record, and change the playlist (ID: 68).
- As a user, I need to have real-time metadata (ID: 65).

Test Suites

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

1. **TC01 – Successful Sign Up**
    - **Action:**
        - Go to the Sign-Up page.
        - Enter valid name, email, and password.
        - Click "Sign Up".
    - **Expected Results:**
        - Sign-Up form is displayed.
        - Fields accept values without error.
        - Account is created, and the user is redirected to the dashboard.
    - **Type**: Happy Path
2. **TC02 – Secure Login**
    - **Action:**
        - Go to the Login page.
        - Enter valid email and password.
        - Click on "Login".
    - **Expected Results:**
        - Login form is displayed.
        - Fields accept data without error.
        - User is logged in and redirected to the dashboard.
    - **Type:** Happy Path
3. **TC03 – Sign Up with Existing Email**
    - **Action:**
        - Go to the Sign-Up page.
        - Enter a name and an already registered email.
        - Click on "Sign Up".
    - **Expected Results:**

        - Fields accept data.
        - Error message "Email already registered" is displayed.
    - **Type:** Error Path

4. **TC04 – Login with Wrong Password**
    - **Action:**
        - Go to the Login page.
        - Enter valid email and incorrect password.
        - Click on "Login".
    - **Expected Results:**
        - Input is accepted.
        - Error message "Invalid username or password" is shown.
    - **Type:** Error Path

Test Suit: TS02 – Add List (ID: 87)

1. **TC05 – View TO DO List Page**
    - **Action:**
        - Log in successfully.
        - Navigate to "My " section.
    - **Expected Results:**
        - All created playlists are displayed clearly.
    - **Type:** Happy Path
2. TC06 –List Failure
    - **Action:**
        - Disconnect from the internet.
        - Navigate to "List".
    - **Expected Results:**
        - Network is offline.
        - Error message "Unable to load playlists" is shown.
    - **Type:** Error Path

Test Suit: TS03 – Real Time TO DO Socket (ID: 88)

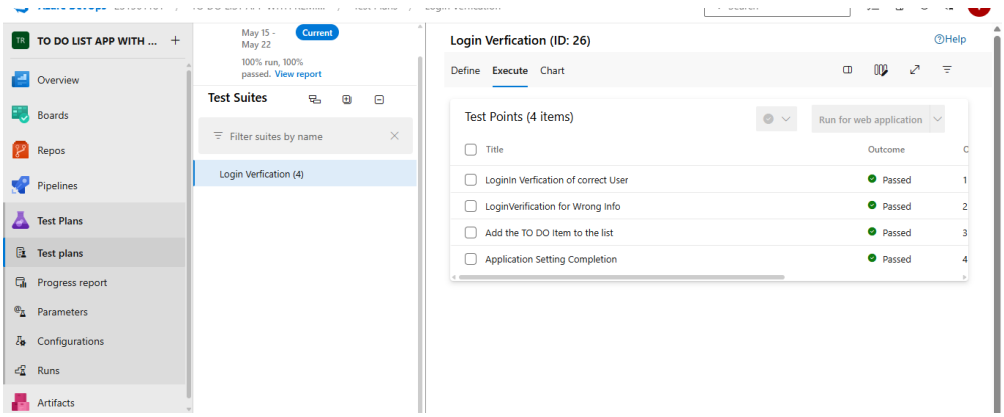
1. **TC07 – Real-Time Metadata Display**
    - **Action:**
        - Display the list panel.
        - Observe the metadata panel.
    - **Expected Results:**
        - Metadata (title, user, list, duration) is displayed and updates in real time.
    - **Type:** Happy Path
2. TC08 – Metadata Not Updating
    - **Action:**

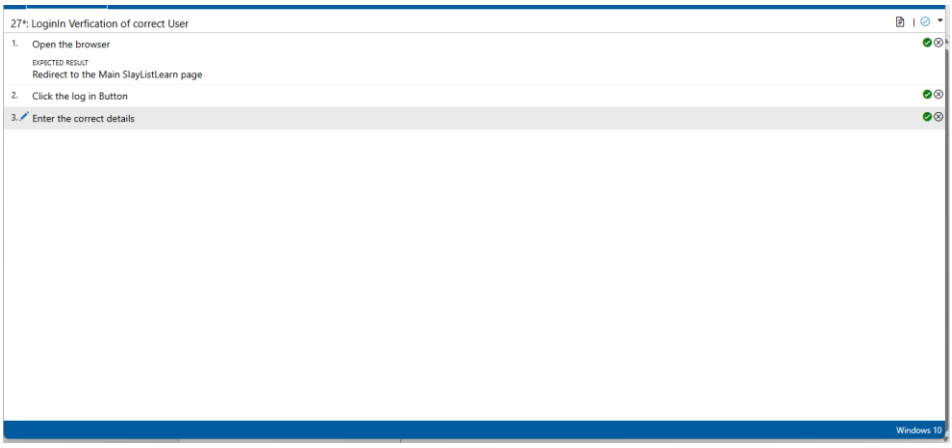
        - Add different attribute.
        - Observe the metadata panel.
    - **Expected Results:**
        - Metadata remains static or shows default/fallback message.
    - **Type:** Error Path

Test Suit: TS04 - Editing (ID: 89)

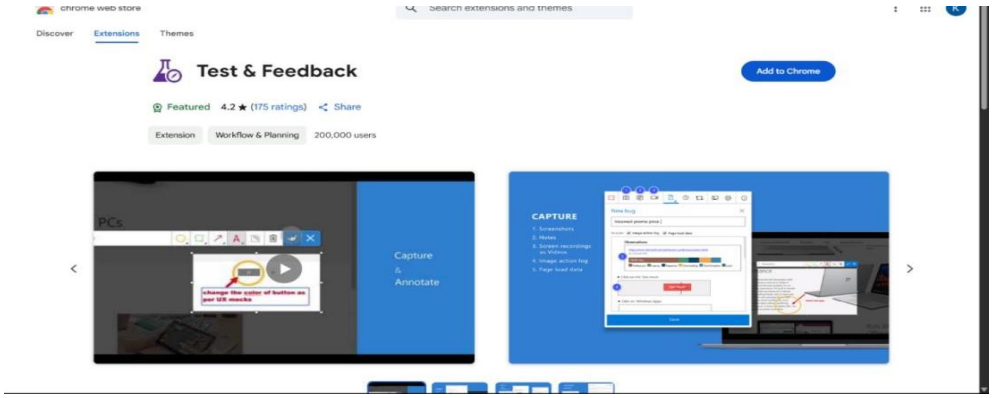
1. **TC09 – Rename Edit Successfully**
    - **Action:**
        - Navigate to “Rename the List ".
        - Click "Rename" next to a playlist.
        - Enter a new name and click "Save".
    - **Expected Results:**
        - List details / name updates successfully.
    - **Type:** Happy Path
2. **TC10 – Rename with Blank Name**
    - **Action:**
        - Click "Rename" on a playlist.
        - Leave the field blank.
        - Click "Save".
    - **Expected Results:**
        - Error message "List name cannot be empty" is shown.
    - **Type:** Error Path
3. **TC11 – Change Priority Order**
    - **Action:**
        - Open a List.
        - Drag and drop list to reorder.
        - Click "Save".
    - **Expected Results:**
        - Playlist order is updated and saved.
    - **Type:** Happy Path
4. **TC12 – Change Priority Order Fails**
    - **Action:**
        - Login and go to “My Playlists”.
        - Select a List.
        - Go offline or simulate server error.
        - Reorder songs and click “Save Order”.
    - **Expected Results:**
        - Error message: "Failed to update order. Please check your connection".
    - **Type:** Error Path

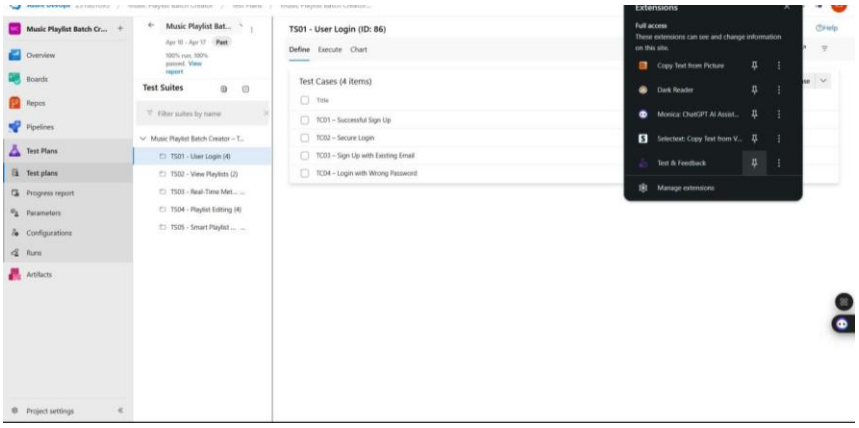
Test Cases


TO DO LIST APP WITH ...
Overview
Boards
Repos
Pipelines
Test Plans
Test plans
Progress report
Parameters
Configurations
Runs
Artifacts
May 15 - May 22
Current
100% run, 100% passed. View report
Test Suites
Filter suites by name
Login Verfication (4)
Login Verfication (ID: 26)
Help
Define
Execute
Chart
Test Points (4 items)
Run for web application
Title
Outcome
LoginIn Verfication of correct User
Passed
LoginVerification for Wrong Info
Passed
Add the TO DO Item to the list
Passed
Application Setting Completion
Passed

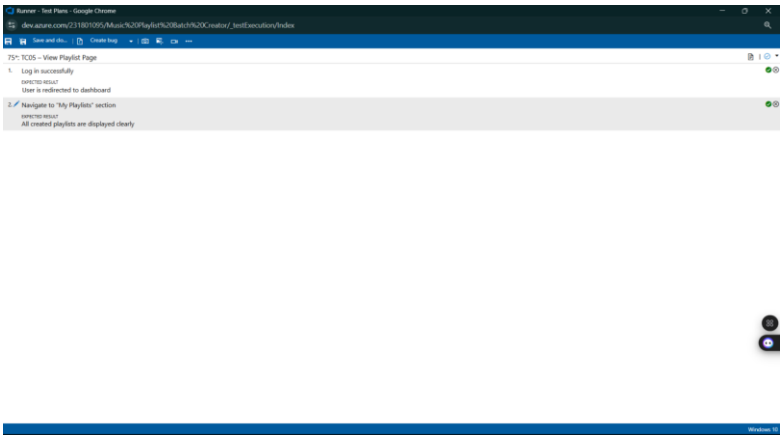

27*: LoginIn Verfication of correct User
1. Open the browser
EXPECTED RESULT
Redirect to the Main SlayListLearn page
2. Click the log in Button
3. Enter the correct details
Windows 10

4. Installation of test

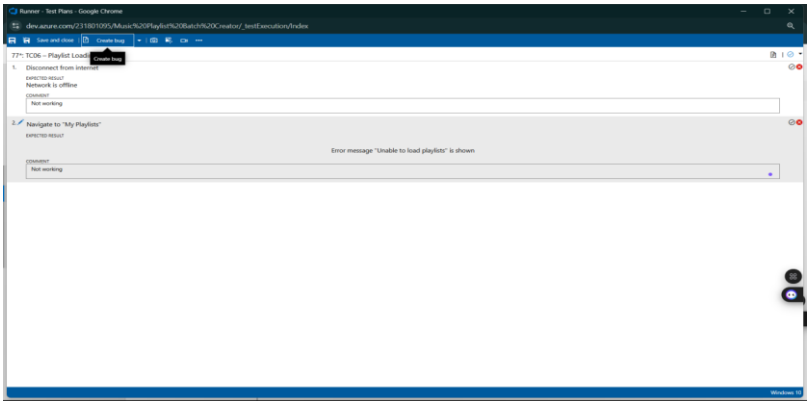


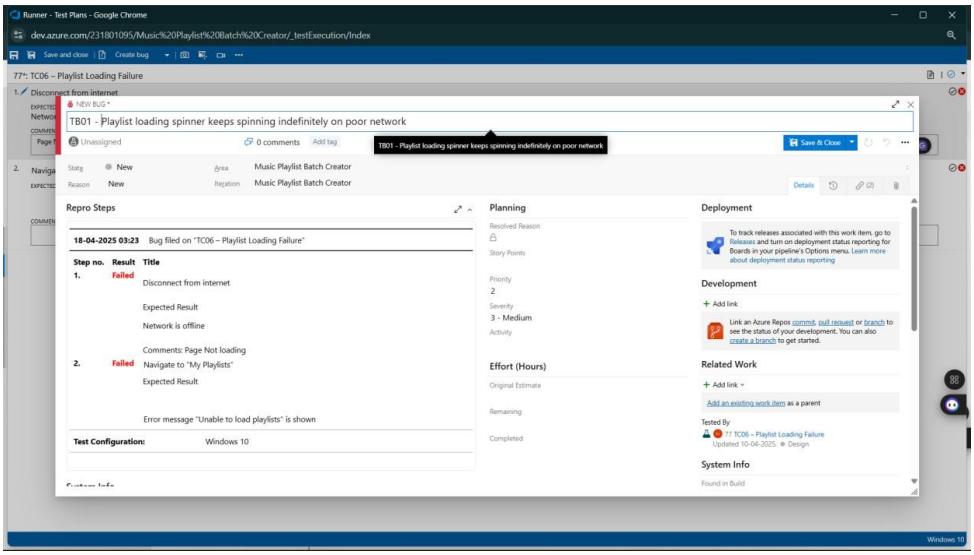


5. Running the test cases

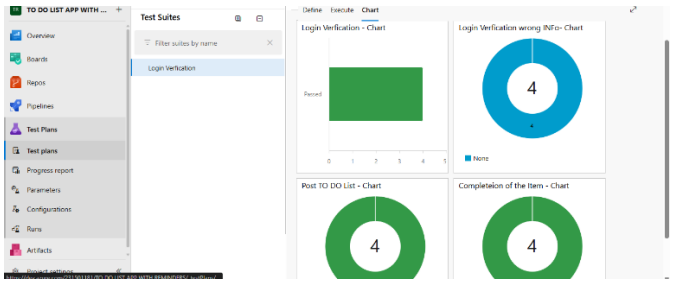


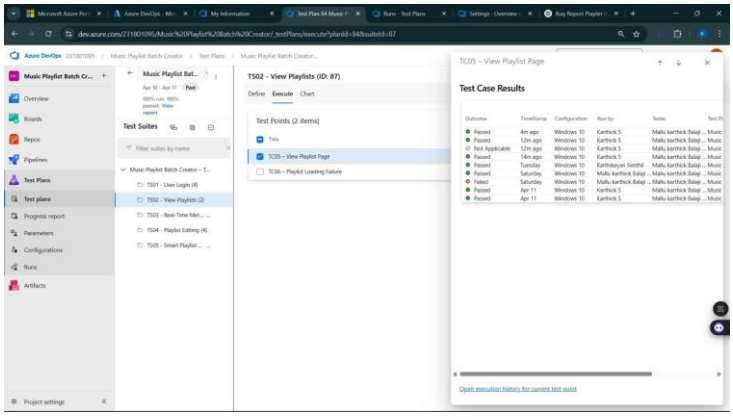
6. Creating the bug





Test Case Results:





Result:

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

EXP NO: 10	PIPE LINES IN AZURE DEVOP'S

Aim:

To create an Azure Pipeline resource and run the performance and integrate the targeted endpoint.

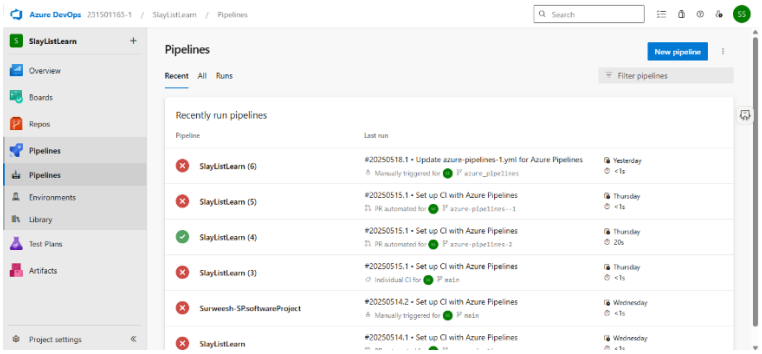
Load Testing

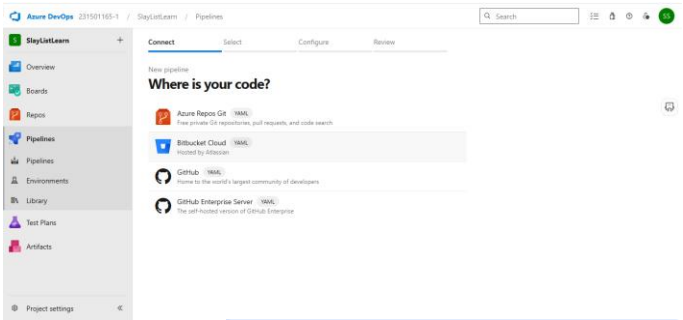
Steps to Create an Azure Load Testing Resource:

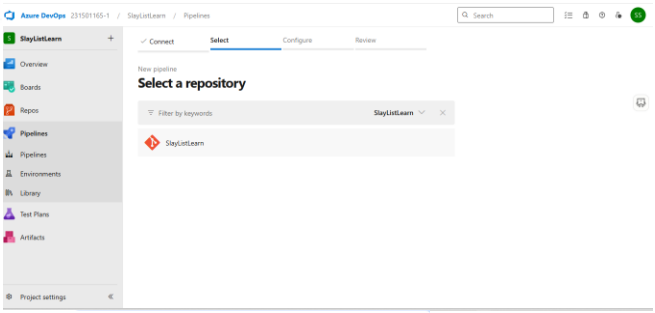
Before you run your first test, you need to create the Azure Load Testing resource:

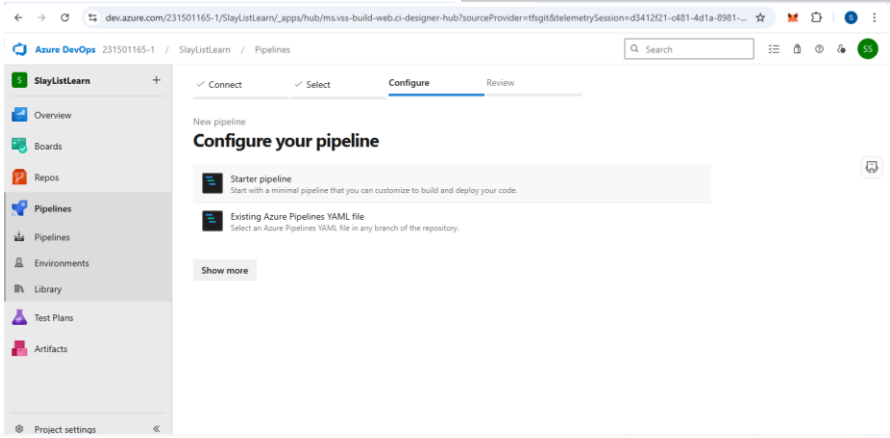
1. **Sign in to Azure DevOps Portal**
   Go to https://dev.azure.com and log in with your Microsoft account.
2. **Navigate to Your Project**
3. Select your organization and project
4. If you haven't created a project yet, click **"New Project"**, fill in the details, and create one.
5. **Create a New Pipeline**
6. On the left sidebar, go to **Pipelines**.
7. Click **"New Pipeline"** at the top right.
8. **Select Your Repository**
9. Choose where your code is stored (e.g., Azure Repos Git, GitHub, Bitbucket, etc.).
10. Authenticate and select the specific repository you want to build.
11. **Configure the Pipeline**
12. Choose a pipeline configuration method (YAML or classic editor).
    a. **YAML**: Recommended for modern DevOps workflows.
    b. **Classic editor**: Useful if you prefer a GUI-based configuration.
13. **Set Up Your Build Pipeline**
14. If using YAML:
    a. Either let Azure generate a template based on your code, or write your own `azure-pipelines.yml` file.
15. If using classic:
    a. Add tasks such as build, test, and deploy using the step-by-step editor.
16. **Save and Run the Pipeline**
17. Click **"Save and run"** to execute your pipeline for the first time.
18. Review the configuration and click **"Run"**.
19. **Monitor the Pipeline Execution**
20. You'll be redirected to the run summary where you can see logs and results for each job and step.
21. *(Optional)* **Add Triggers and Variables**
22. Set up CI/CD triggers, environment variables, and other configurations to fine-tune your pipeline behavior.

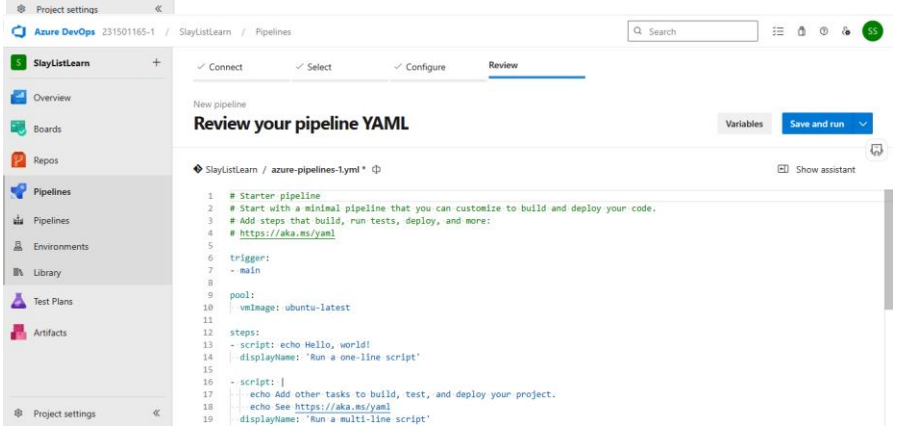
Creating Pipelines:

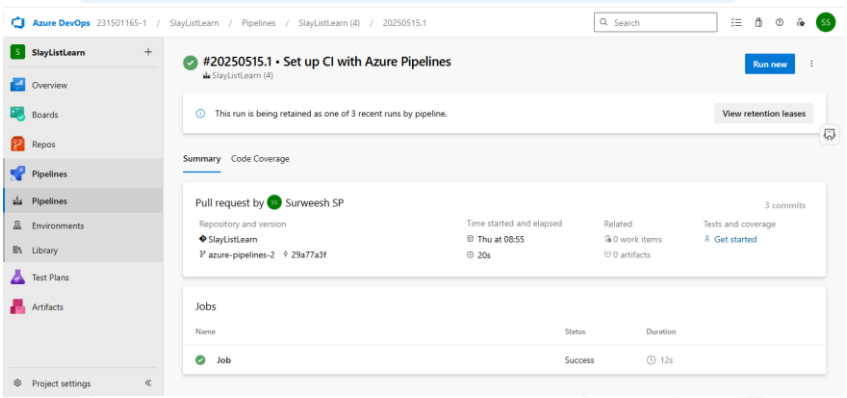


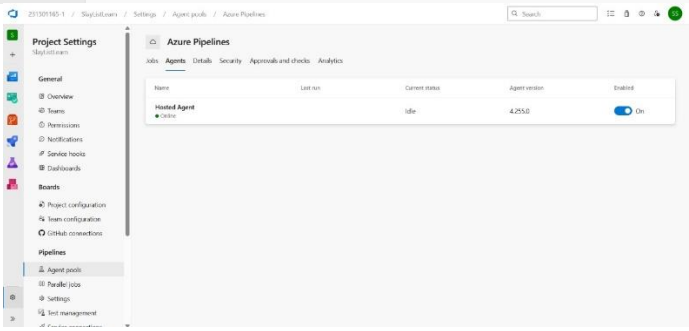


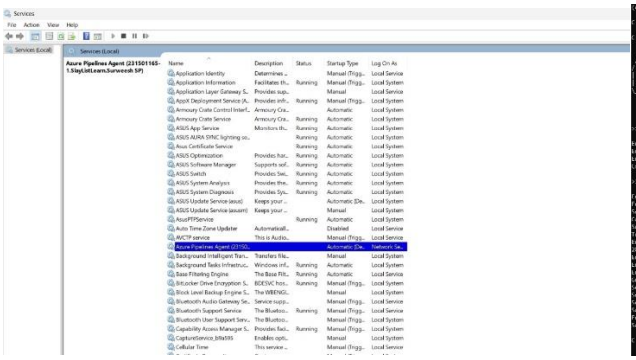


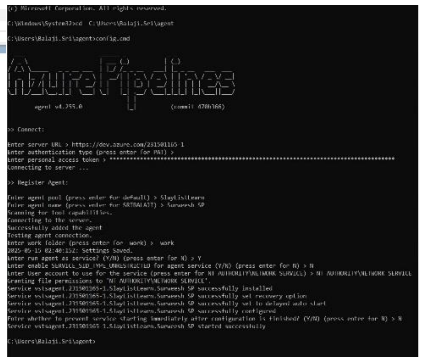












Result:

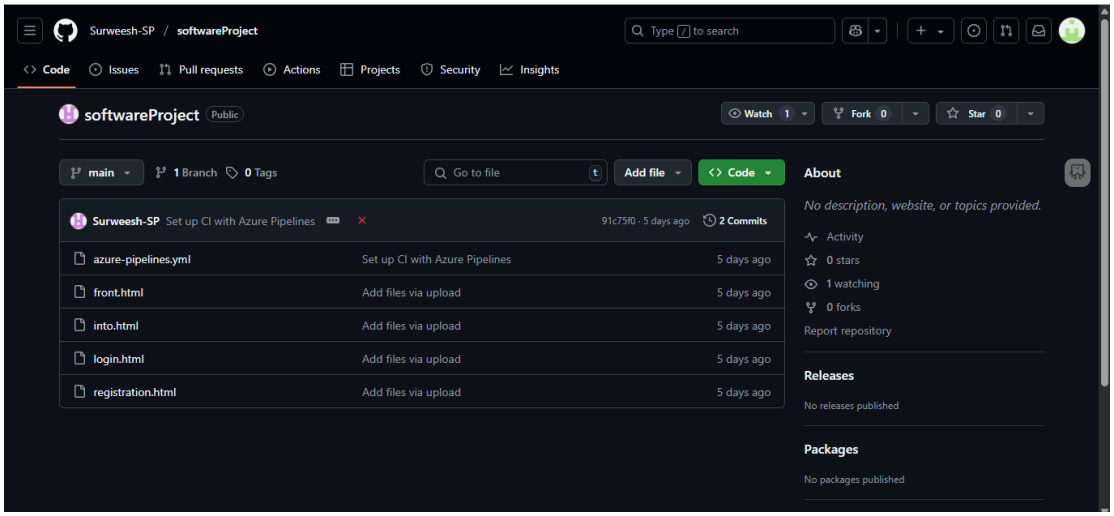
Successfully created the Azure Pipelines resource and executed a performance of the specified endpoint.

EXP NO: 11	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 TO DO List 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.