**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 Use-Case and Activity Diagrams for Project Architecture** |
| **8.** | **25/3/25** | **Testing – Test Plans and Test Cases** |
| **9.** | **15/4/25** | **CI/CD Pipelines in Azure** |
| **10.** | **22/4/25** | **GitHub: Project Structure & Naming Convention** |

|  |  |
| --- | --- |
| **EXP NO:** **1** | **AZURE DEVOPS ENVIRONMENT SETUP** |
|  |

# AIM

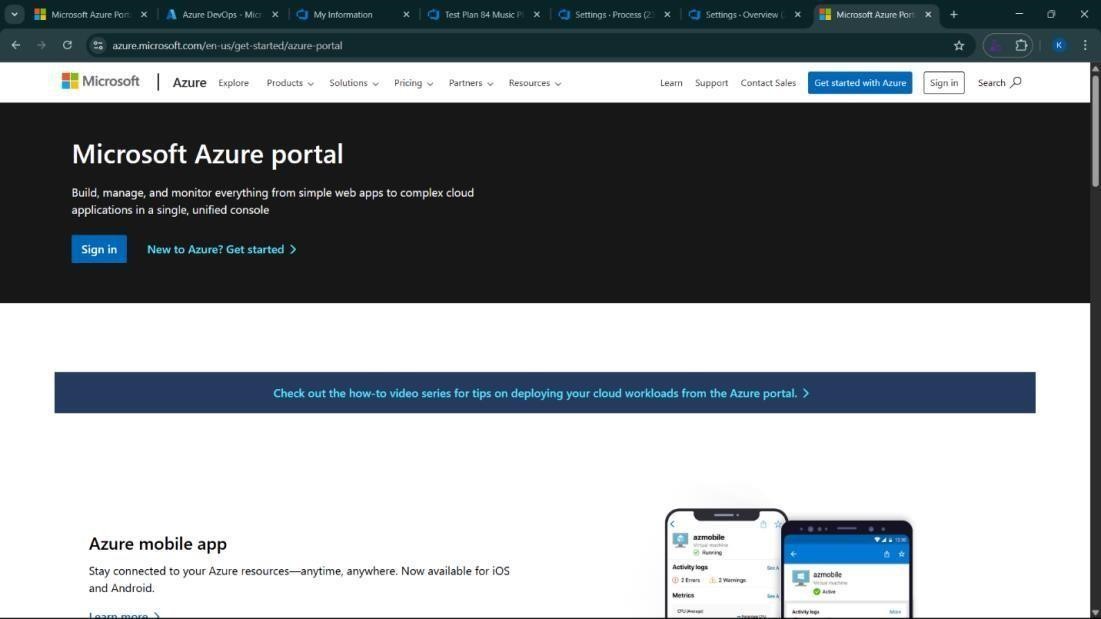
To set up and access the Azure DevOps environment by creating an organization through the Azure portal.

# INSTALLATION

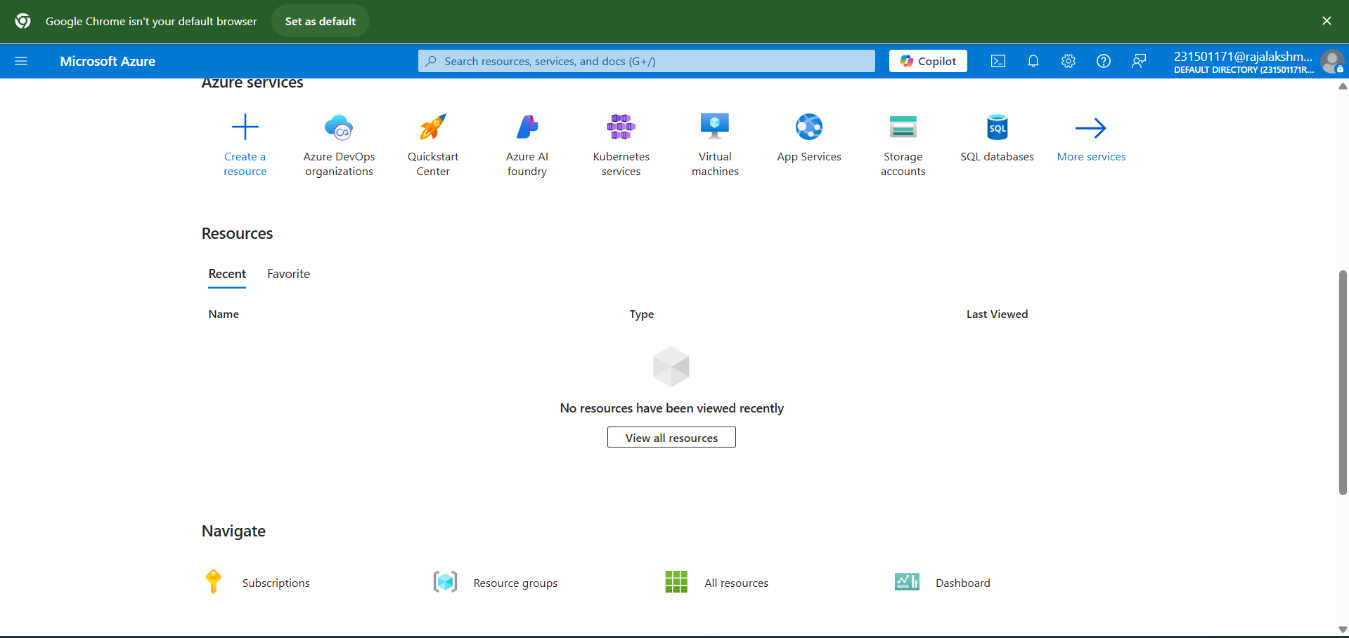
1.Open your web browser and go to the Azure website: [https://azure.microsoft.com/enus/getstarted/azureportal.](https://azure.microsoft.com/en-us/get-started/azure-portal)

Sign in using your Microsoft account credentials.

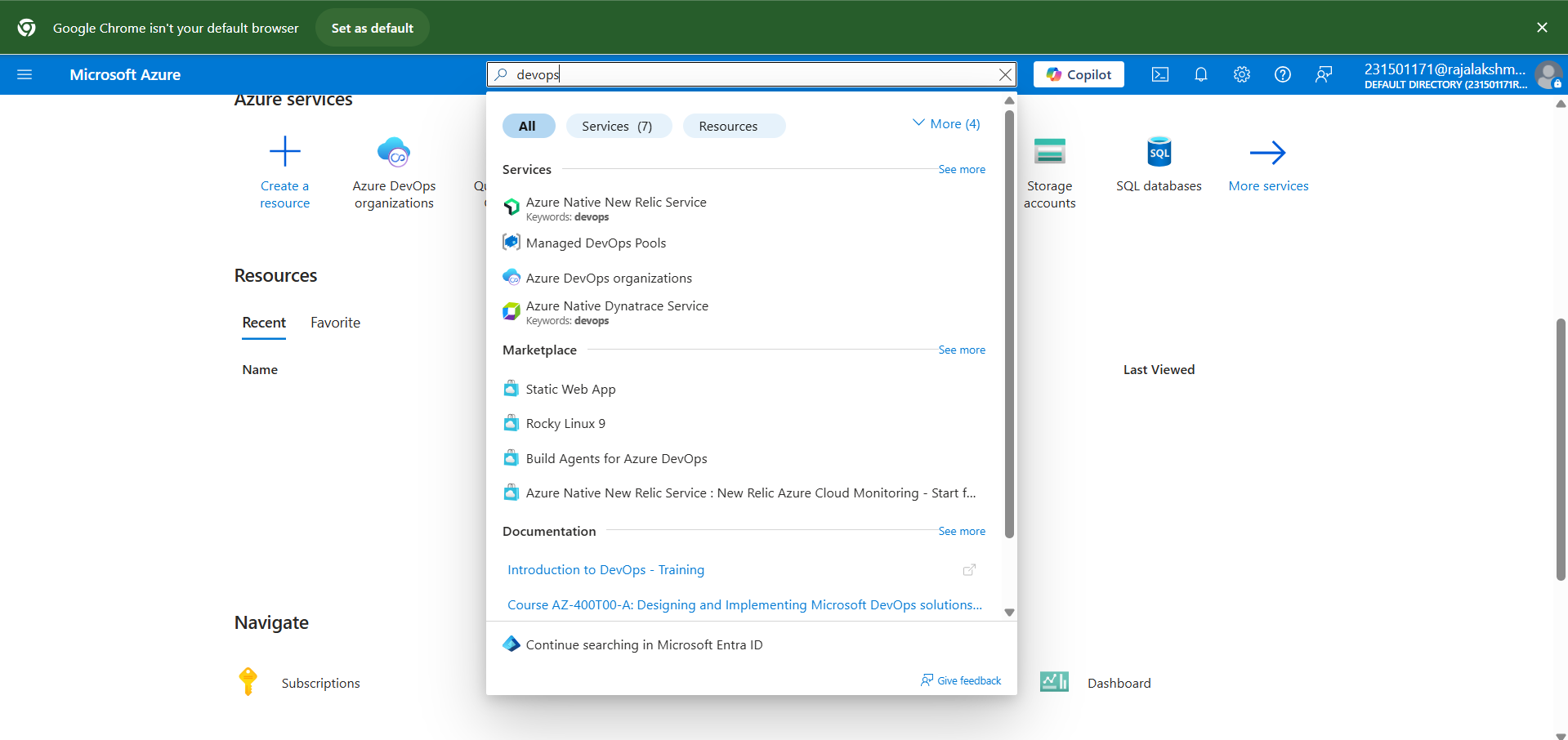
If you don't have a Microsoft account, you can create one here: <https://signup.live.com/?lic=1>



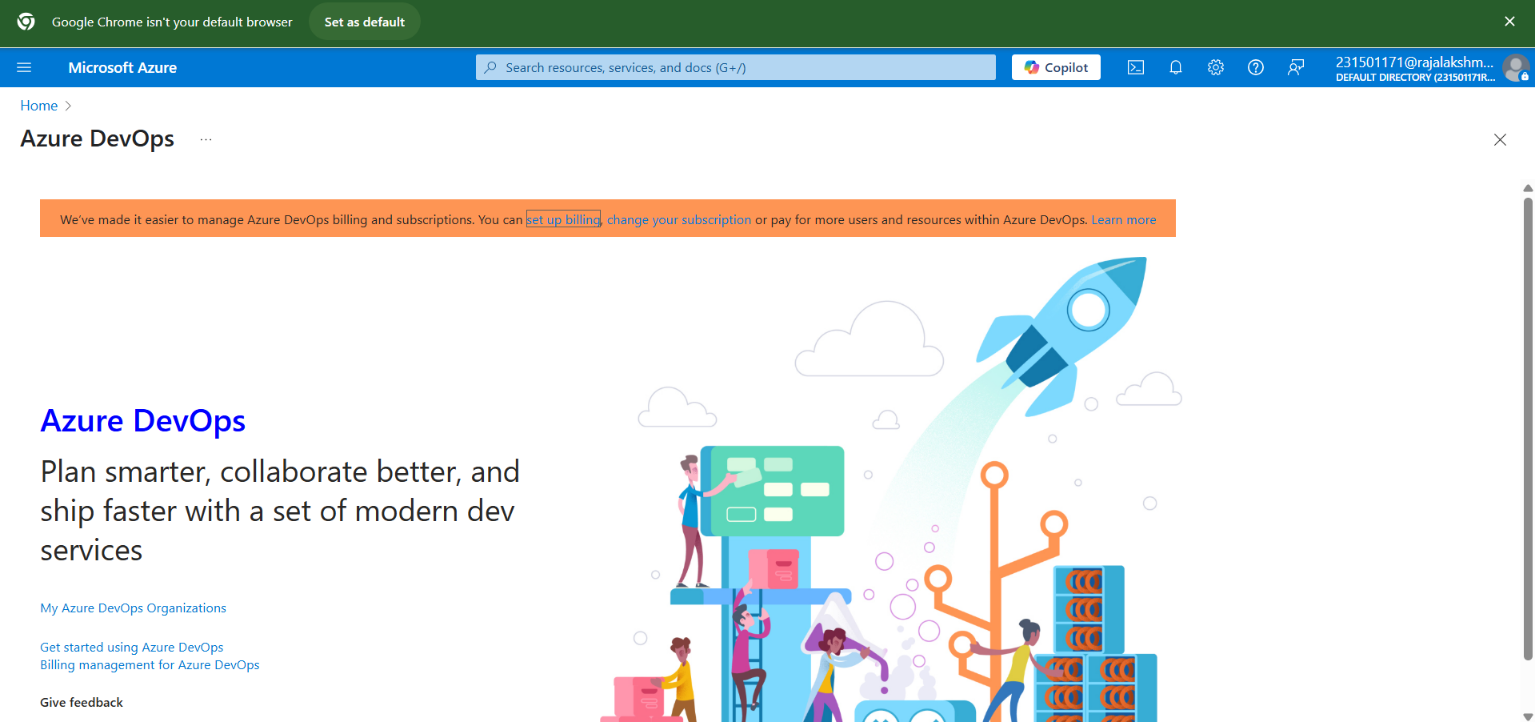
2.Azure home page



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**

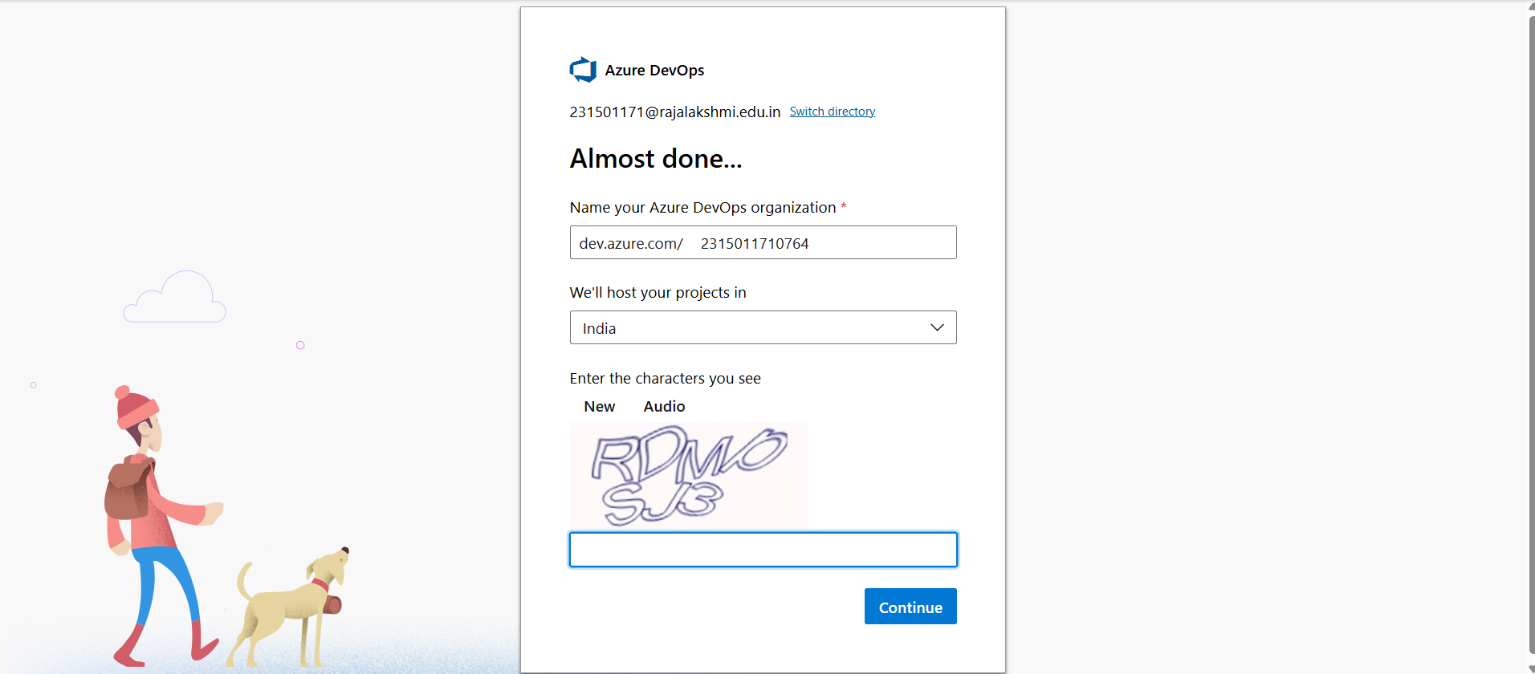
Successfully accessed the Azure DevOps environment and created a new organization through the Azure portal.

|  |  |
| --- | --- |
| **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

1. After the organization is set up, you’ll need to create your first **project**. This is where you'll

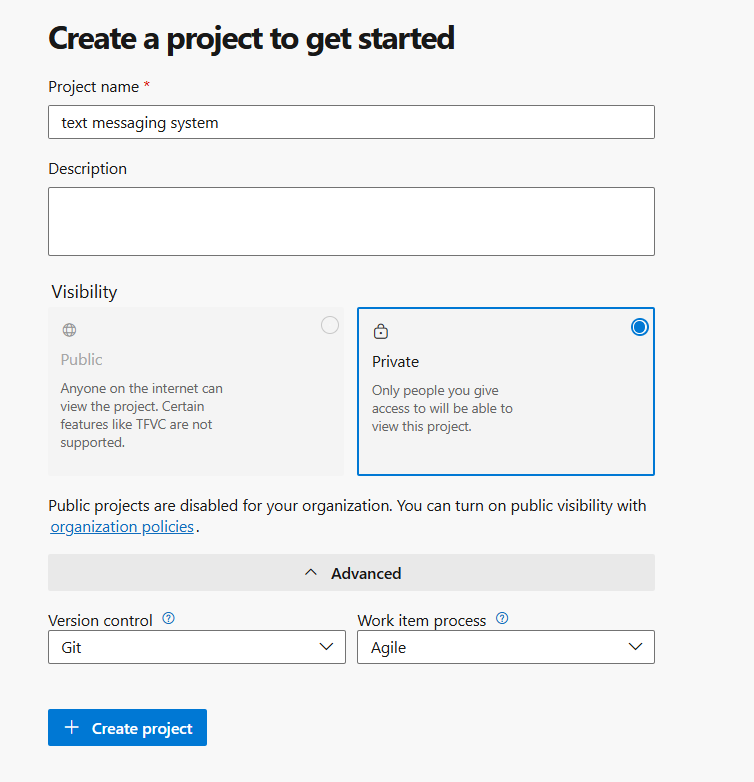
begin to manage code, pipelines, work items, and more.

1. On the organization’s **Home page**, click on the **New Project** button.
2. Enter the project name, description, and visibility options:

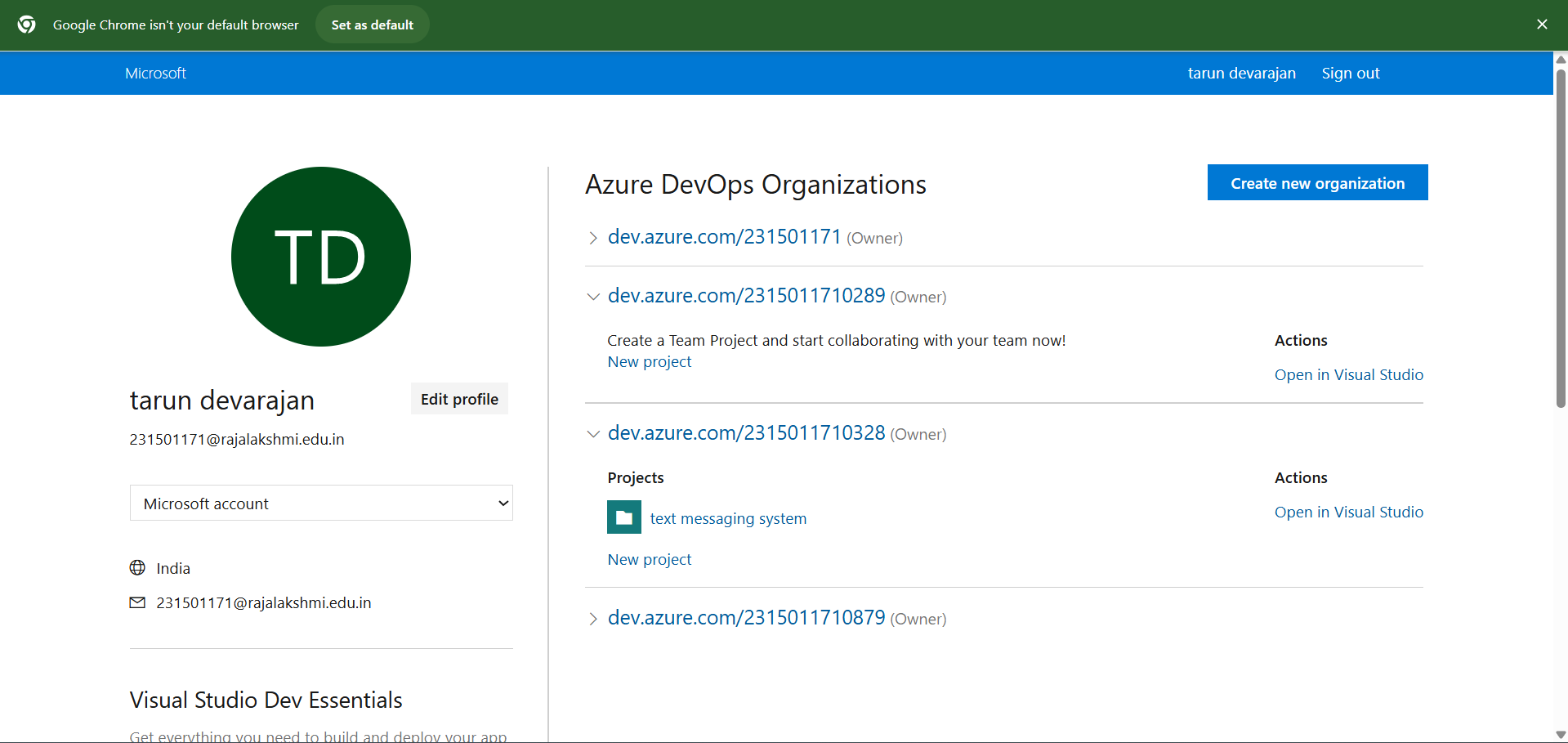
***Name****:* Choose a name for the project (e.g., LMS).

***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.



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:

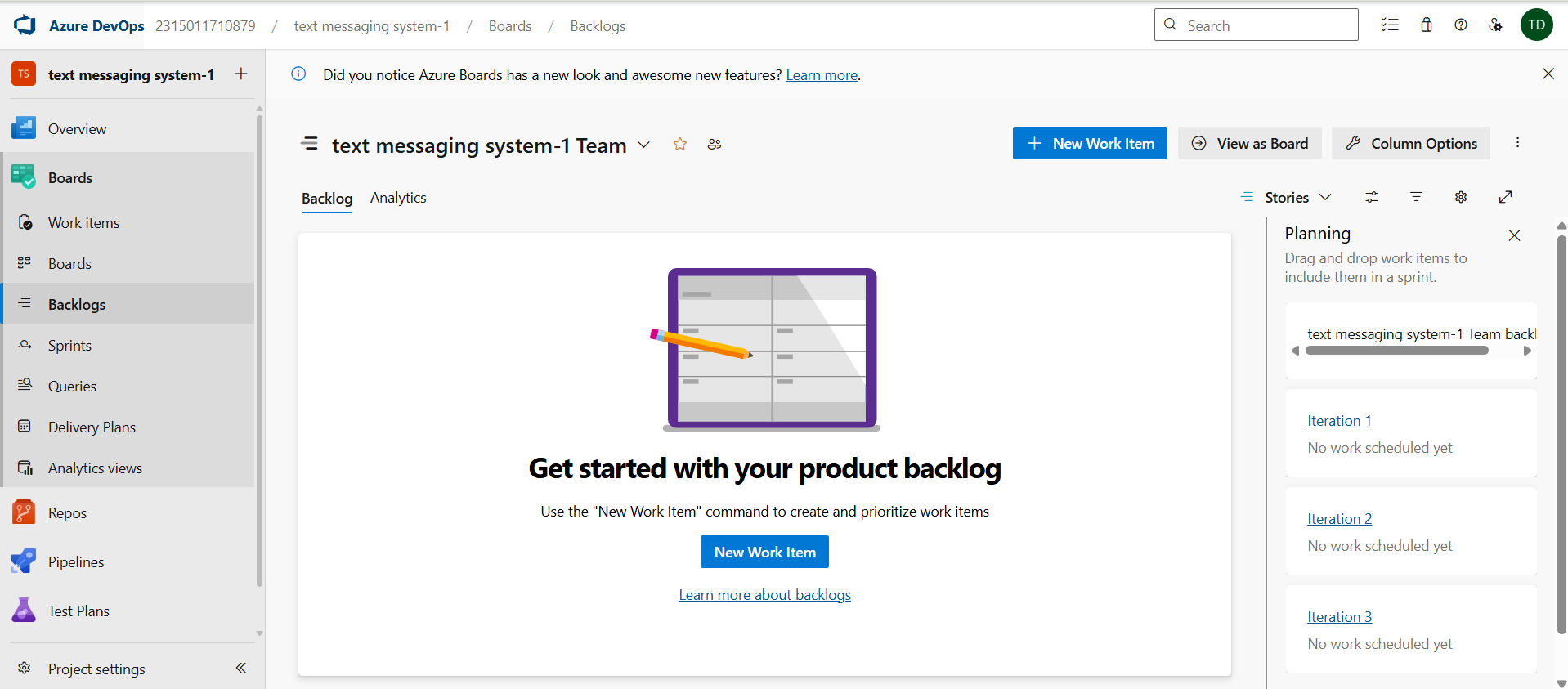
1. From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards**

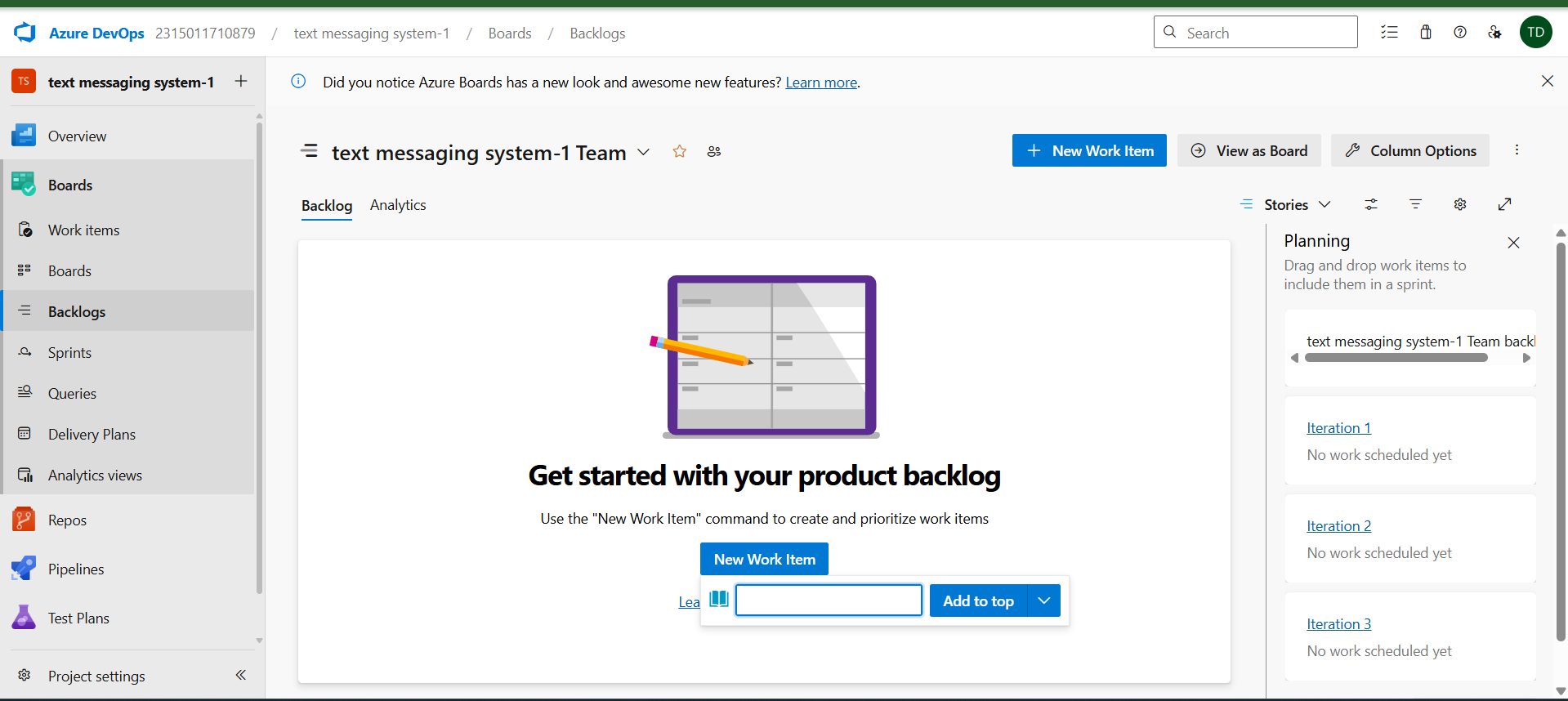
page, where you can manage work items, backlogs, and sprints.

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

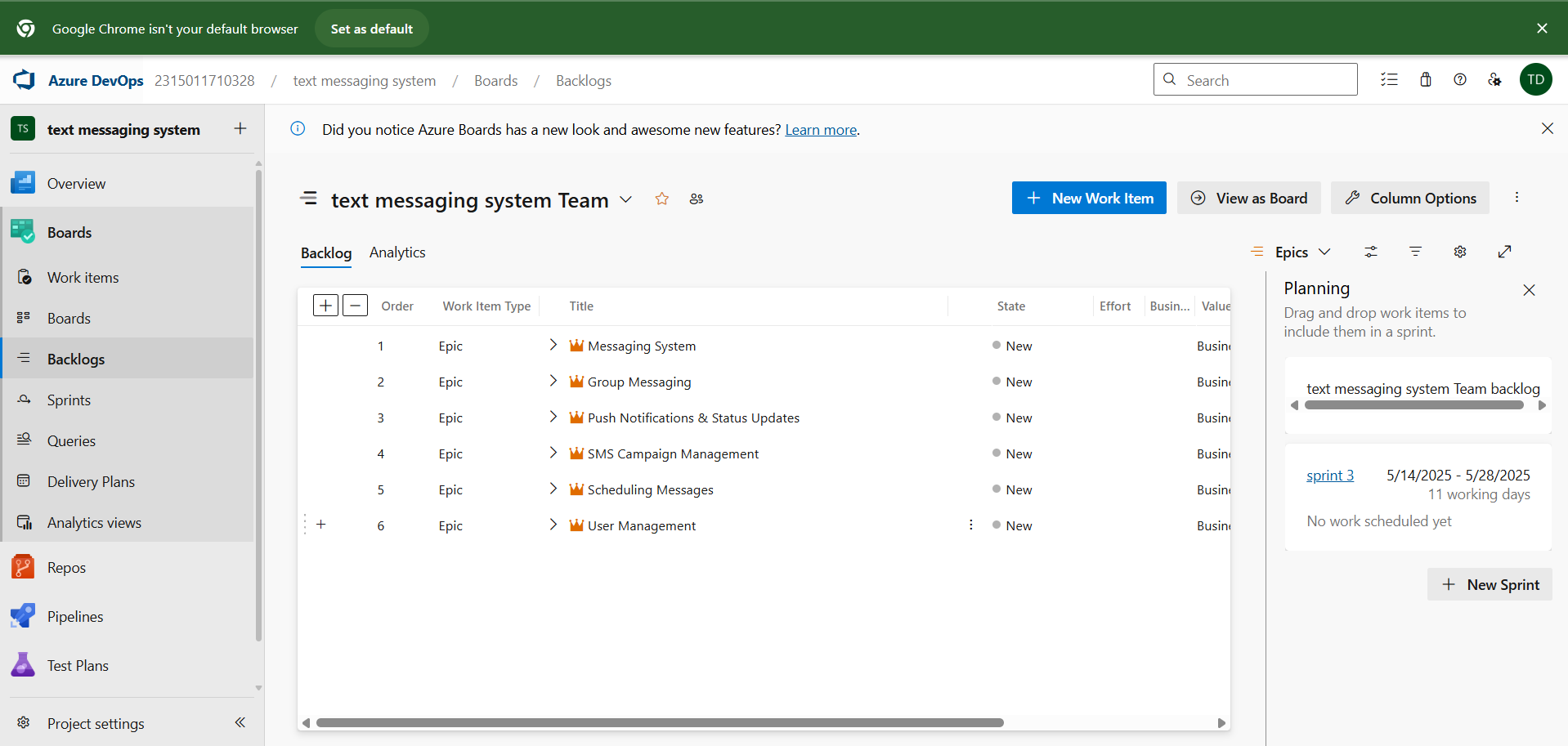
Successfully created an Azure DevOps project with user story management and agile workflow setup.

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

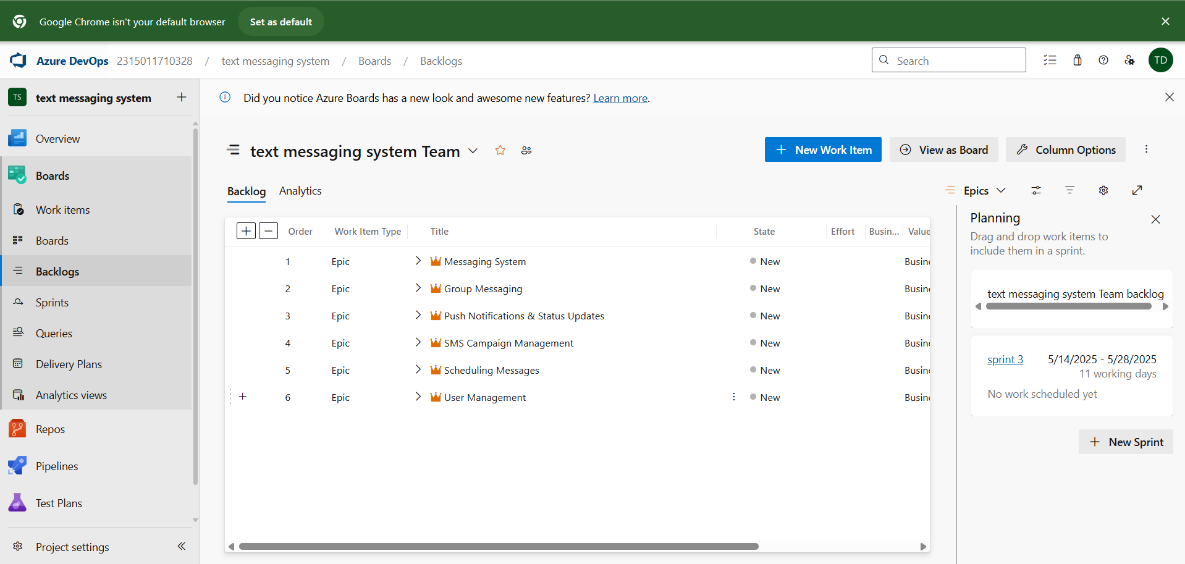
**AIM**

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

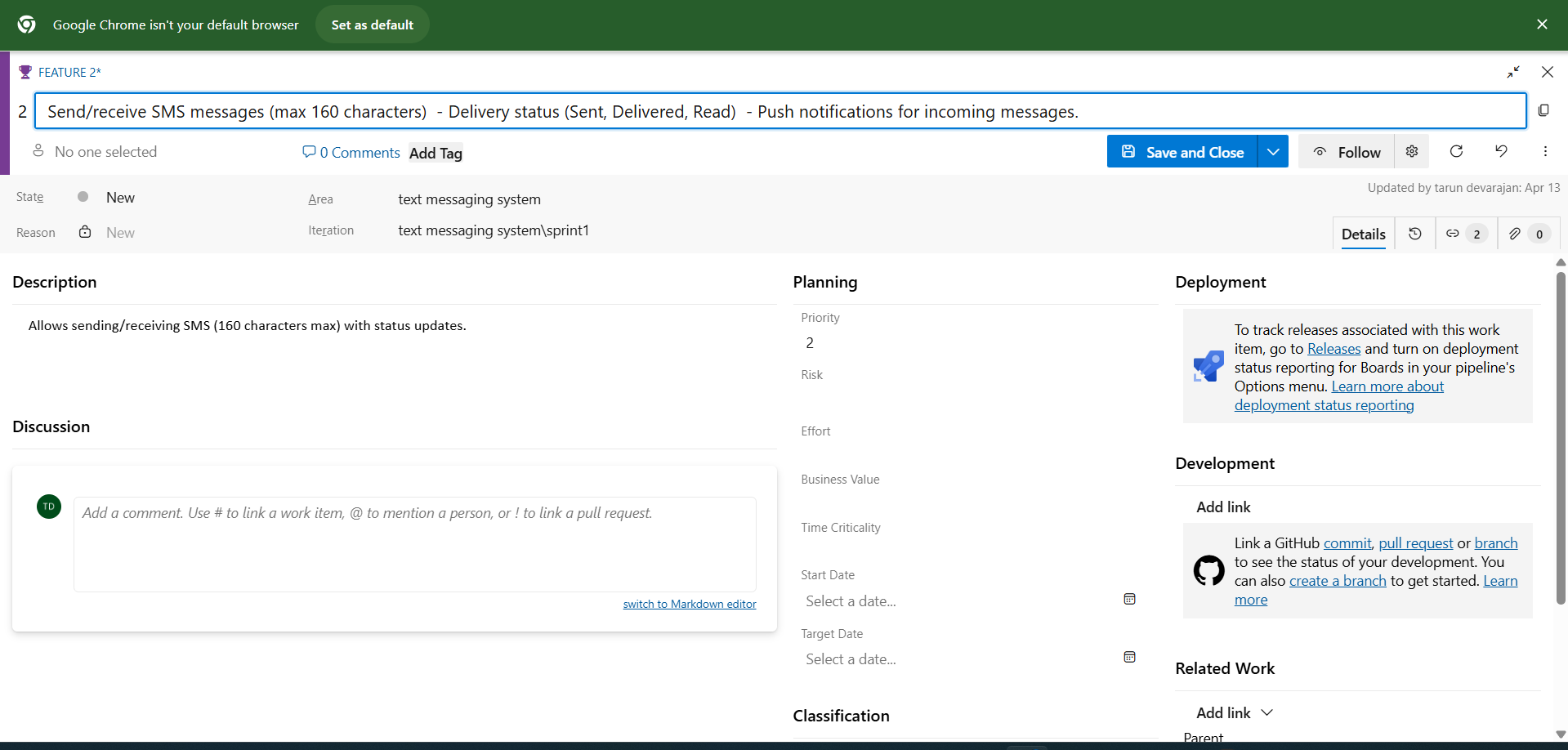
# Create Epic, Features, User Stories, Task



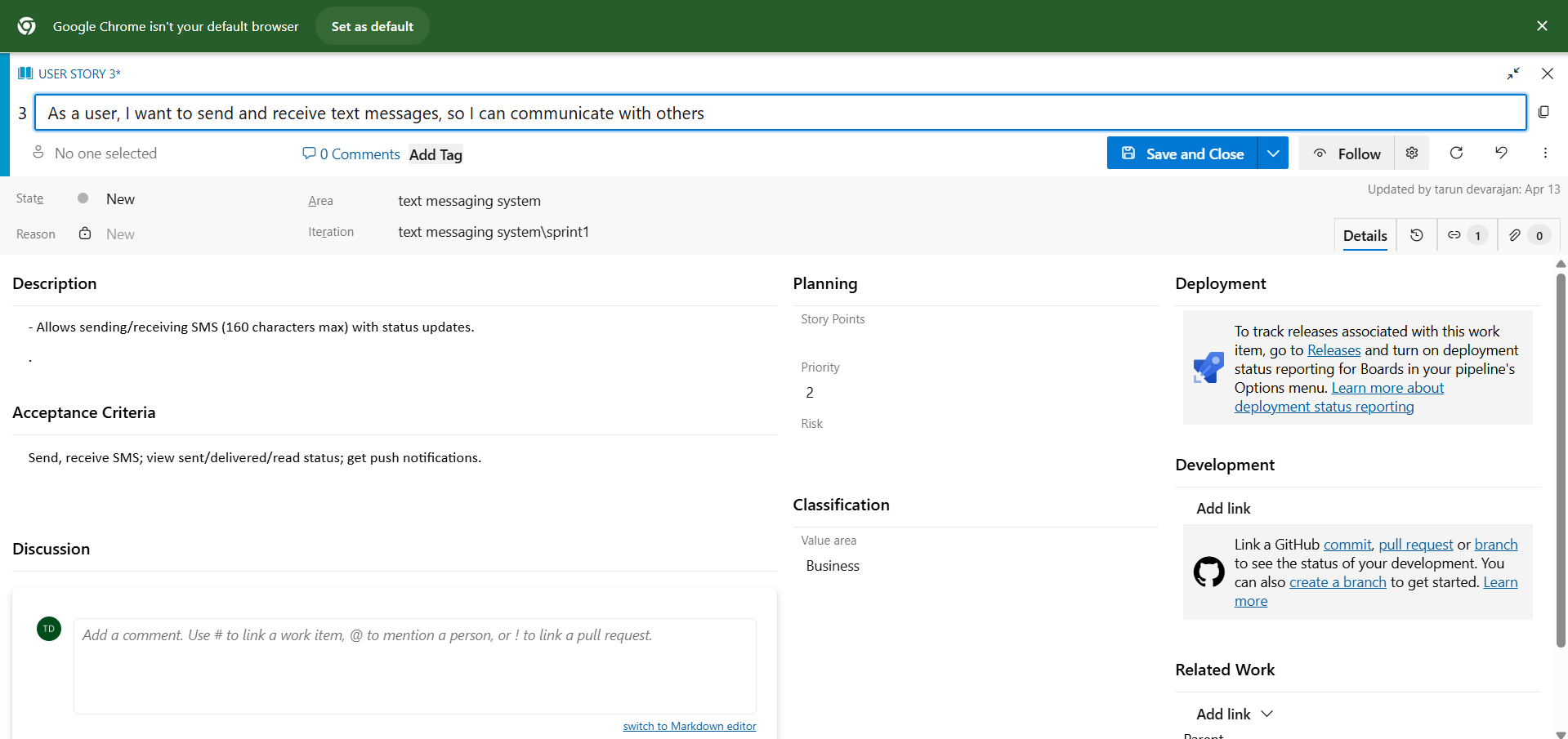
1.Fill in Epics



## 2.Fill in Features



# 3.Fill in User Story Details



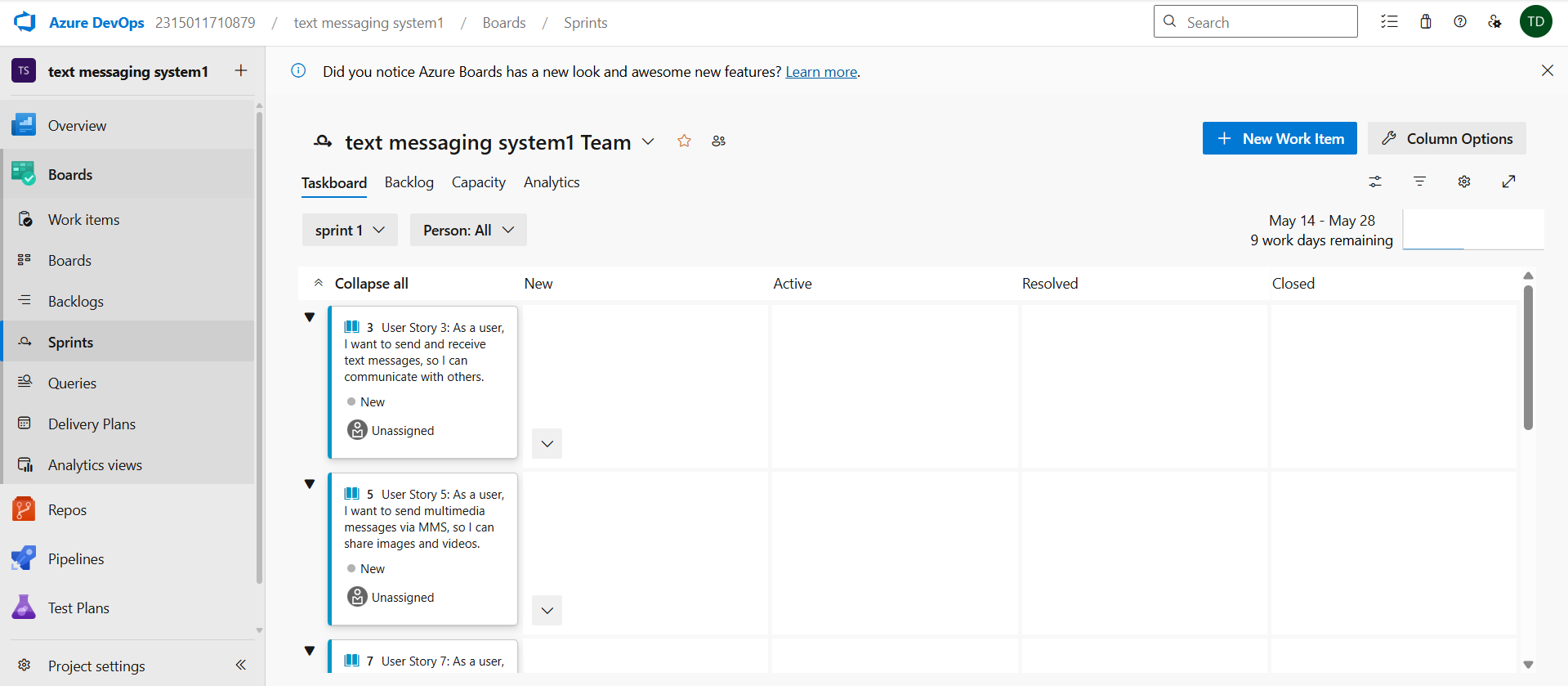
RESULT :Thus, the creation of epics, features, user story and task has been created successfully.

|  |  |
| --- | --- |
| **EXP NO:** **4** | **SPRINT PLANNING** |

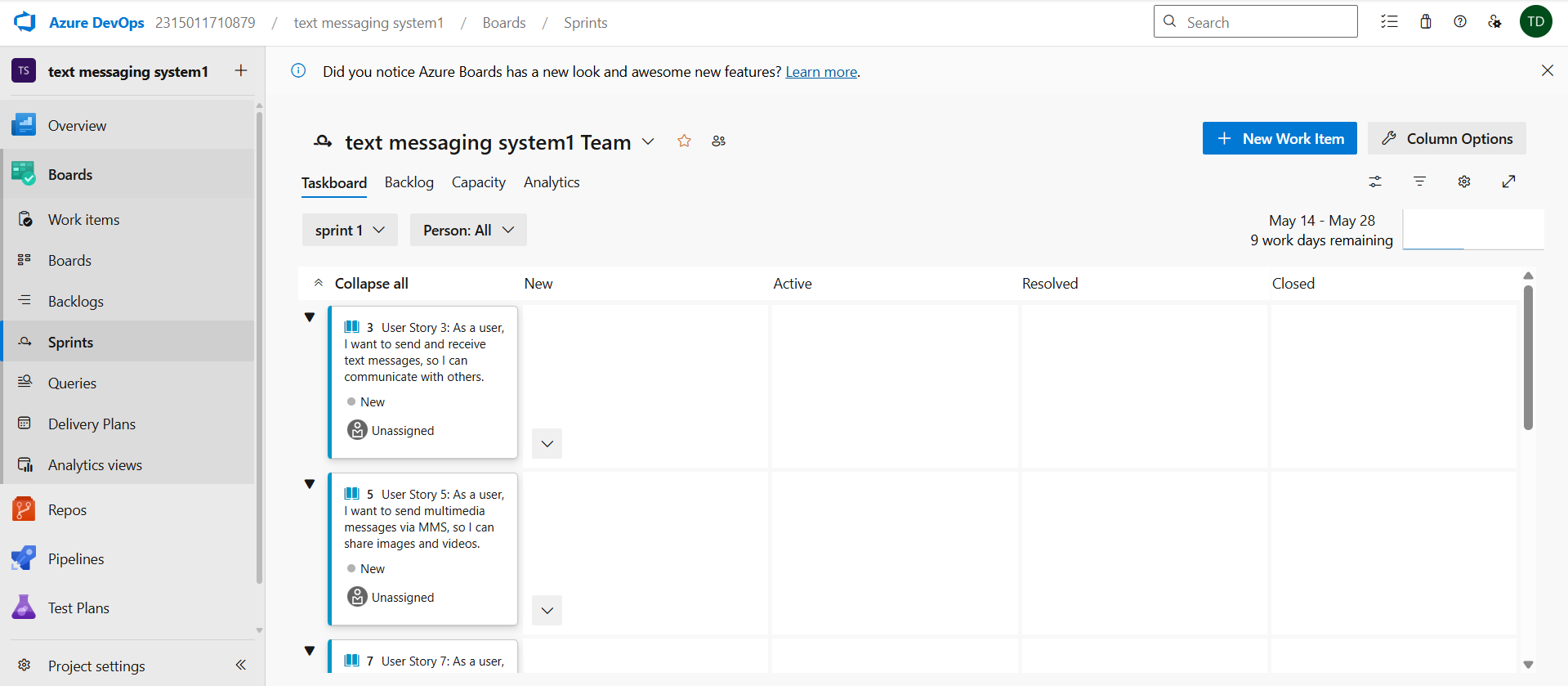
**AIM**

To assign user story to specific sprint for the Text /sms messaging system System.

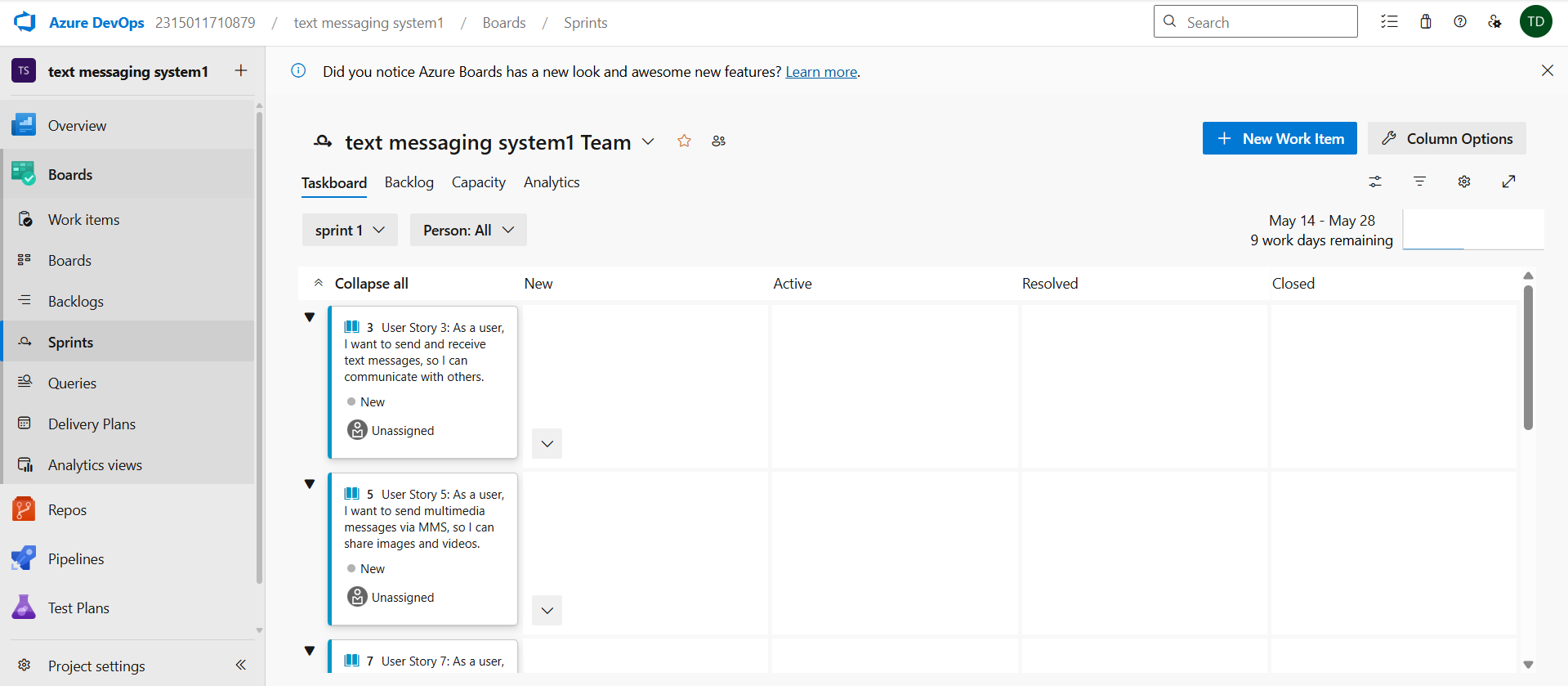
## Sprint Planning Sprint 1



## Sprint 2



## Sprint 3



## RESULT

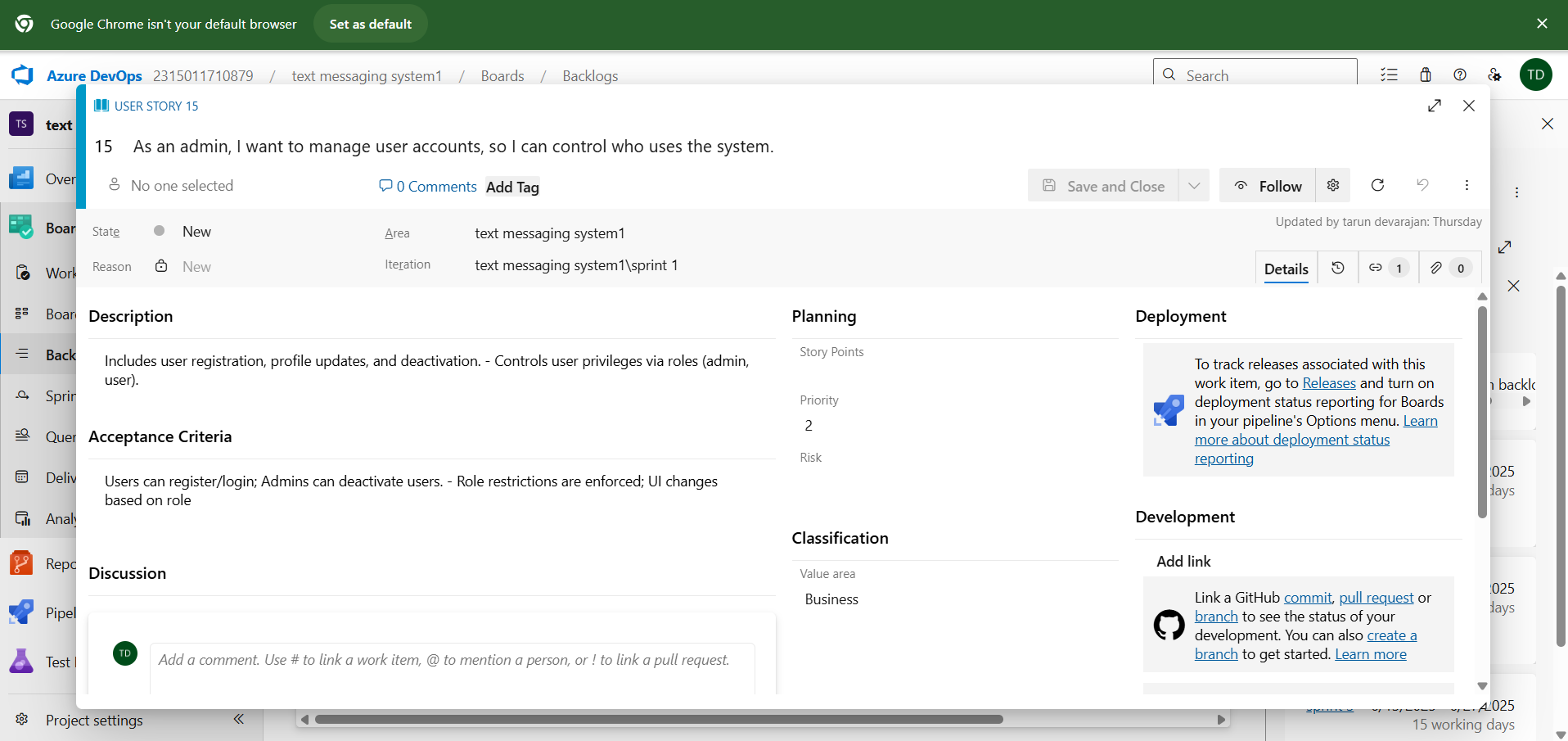
The Sprints have been created for Text/sms messaging system System.

|  |  |
| --- | --- |
| **EXP NO:5** | **POKER ESTIMATION** |

**AIM**

Create Poker Estimation for the user stories – Online Quiz System.

## Poker Estimation



## RESULT

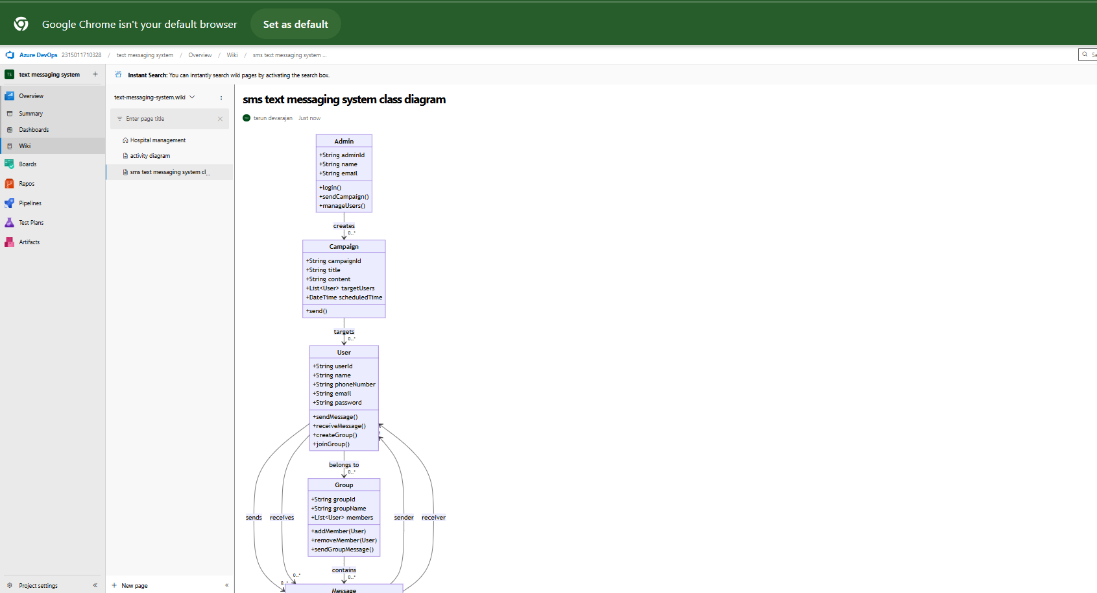
The Estimation/Story Points is created for the project using Poker Estimation.

|  |  |
| --- | --- |
| **EXP NO: 6** | **DESIGNING CLASS AND SEQUENCE DIAGRAMS FOR**    **PROJECT ARCHITECTURE** |

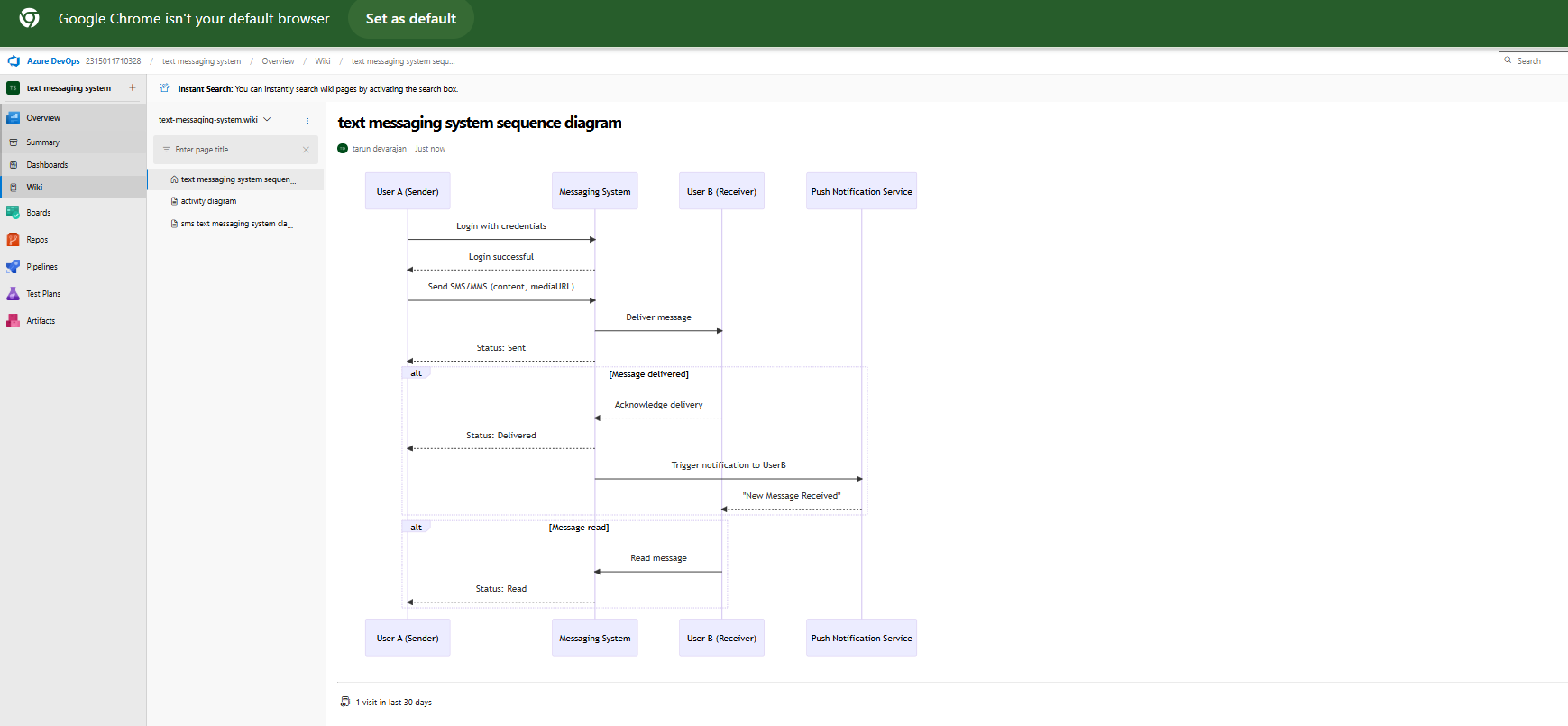
**AIM**

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 Text/sms messaging System.

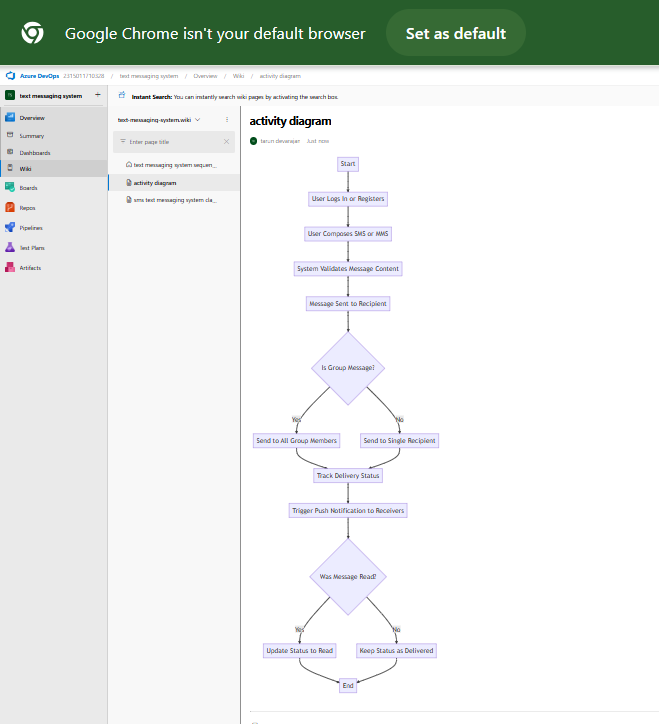
|  |  |
| --- | --- |
| **EXP NO:** **7** | **DESIGNING USE-CASE AND ACTIVITY DIAGRAMS**    **FOR PROJECT STRUCTURE** |

## AIM

To Design an Use-Case Diagram and Activity Diagram for the given Project. **7A. Use-Case Diagram**



## 7B. Activity Diagram



**RESULT**

The Use-Case Diagram and Activity Diagram is designed Successfully for the text/sms messaging System.

|  |  |
| --- | --- |
| **EXP 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

## User Authentication (Login/Register), Send/Receive Messages (SMS, MMS), Delivery Tracking (sent, delivered, read), Group Messaging (group creation, member management), SMS Campaign Management (admin-side)

## 2. Define User Interactions

Register a user, Send SMS/MMS, Receive messages with delivery confirmation, Create a group and send group message, Admin initiates a campaign to target users

## 3. Design Happy Path Test Cases

Successful user login and message sending, Proper delivery status updates, Group chat works with all members receiving messages, Admin sends a campaign successfully

## 4. Design Error Path Test Cases

Login with incorrect credentials, Exceeding character limit in messages,Adding user to group who doesn’t exist, Admin campaign with empty target segment

## 5. Break Down Steps and Expected Results

## Test Steps (clear actions),Expected Results (output, UI updates, data changes)

## 6. Use Clear Naming and IDs

Example: TC\_SMS\_001 – Send Text Message Successfully, Example: TC\_GRP\_002 – Add Invalid Member to Group

## 7. Separate Test

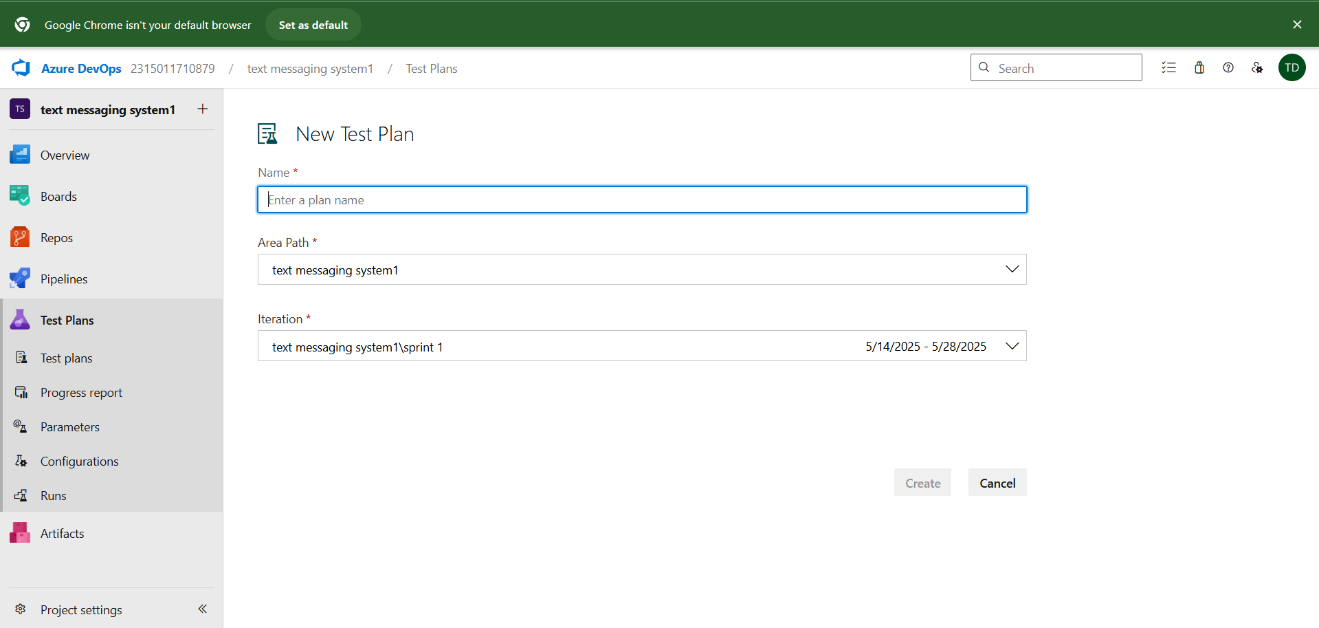
UserAuthentication, Messaging, DeliveryTracking, GroupMessaging, CampaignManagement

## 8. Prioritize and Review

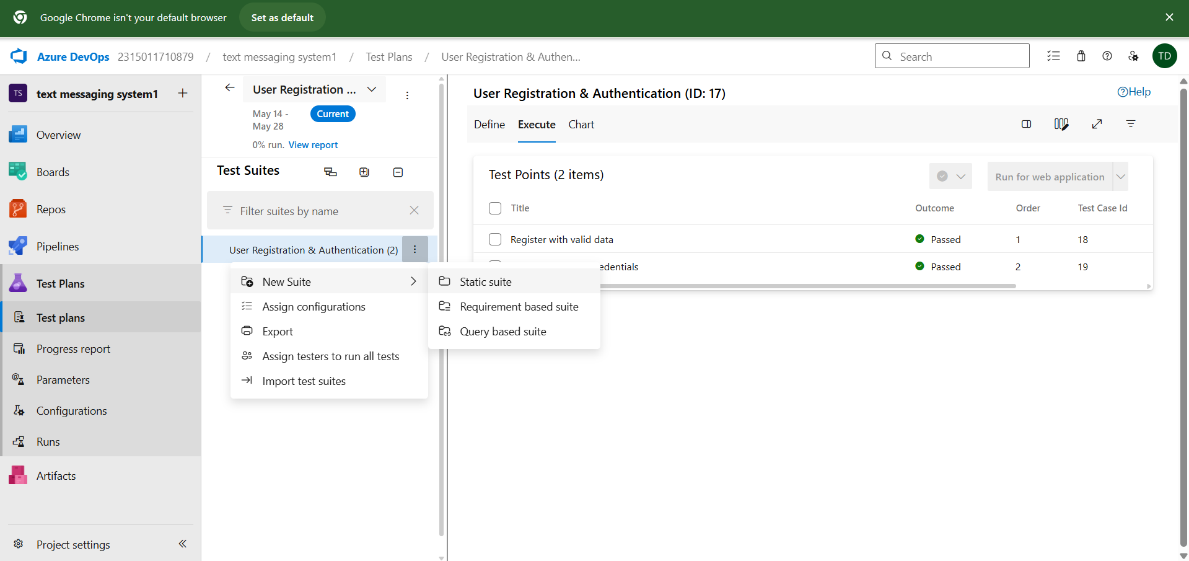
Happy path test cases marked High Priority, Each test case is linked to its corresponding User StoryID in Azure DevOps, Regular review for coverage and completeness

.

## 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.

Text/sms management system– Test Plans

## USER STORIES

* As a user, I want to send and receive text messages, so I can communicate with others.
* As a user, I want to create group chats, so I can communicate with multiple people at once.
* As a user, I want to be notified when I receive a new message.
* As an admin, I want to send SMS campaigns to users, so I can notify them efficiently
* As a user, I want to schedule my messages, so they are sent at a specific time

**TEST SUITES**

**Test Suite: TS01 – User Registration & Authentication**

**TC01 – Register with Valid Data**

Action:  
▪ Open the SMS/MMS web or mobile app  
▪ Navigate to "Sign Up" page  
▪ Fill in valid data: Username, Phone Number, Password  
▪ Click "Register

Expected Result:  
▪ Success message: “Registration Successful. Please log in.”

Type: Happy Pat

TC02 – Show Validation Error for Missing Fields

Action:  
▪ Open "Sign Up" page  
▪ Leave the Phone Number and Password fields empty  
▪ Click "Register"

Expected Result:  
▪ Error message: “Phone Number and Password are required.”

Type: Error Path

## 🔹 Test Suite: TS02 – Messaging Functionality (SMS/MMS)

## TC03 – Send Text Message Successfully

## Action: ▪ Log in as a registered user ▪ Navigate to “New Message” ▪ Enter a valid recipient number ▪ Type a message under 160 characters ▪ Click “Send”

## Expected Result: ▪ Message status changes to “Sent” and later to “Delivered”

## Type: Happy Path

## TC04 – Error for Message Exceeding 160 Characters

## Action: ▪ Open "New Message" ▪ Type a message of more than 160 characters ▪ Click “Send”

## Expected Result: ▪ Error message: “Message exceeds 160 character limit.”

## Type: Error Path

## 🔹 Test Suite: TS03 – Group Messaging

## TC05 – Send Message to Group Successfully

## Action: ▪ Log in and create a group with at least two contacts ▪ Open group chat ▪ Send a message

## Expected Result: ▪ All group members receive the message

## Type: Happy Path

## TC06 – Add Invalid Member to Group

## Action: ▪ Open an existing group ▪ Try to add a user with a non-existent phone number

## Expected Result: ▪ Error message: “User does not exist.”

## Type: Error Path

## 🔹 Test Suite: TS04 – Message Delivery Tracking

## TC07 – Track Delivery Status of Message

## Action: ▪ Send a message to an active recipient ▪ Open the message thread

## Expected Result: ▪ Message status updates: Sent → Delivered → Read

## Type: Happy Path

## TC08 – Handle Failed Delivery (Offline User)

## Action: ▪ Send a message to a number that is turned off or unreachable

## Expected Result: ▪ Message status shows: “Failed to Deliver”

## Type: Error Path

## 🔹 Test Suite: TS05 – SMS Campaign Management (Admin)

## TC09 – Send Campaign Message to User Segment

## Action: ▪ Log in as Admin ▪ Choose a user segment (e.g., based on tags or region) ▪ Compose and send campaign message

## Expected Result: ▪ Campaign sent and summary report generated

## Type: Happy Path

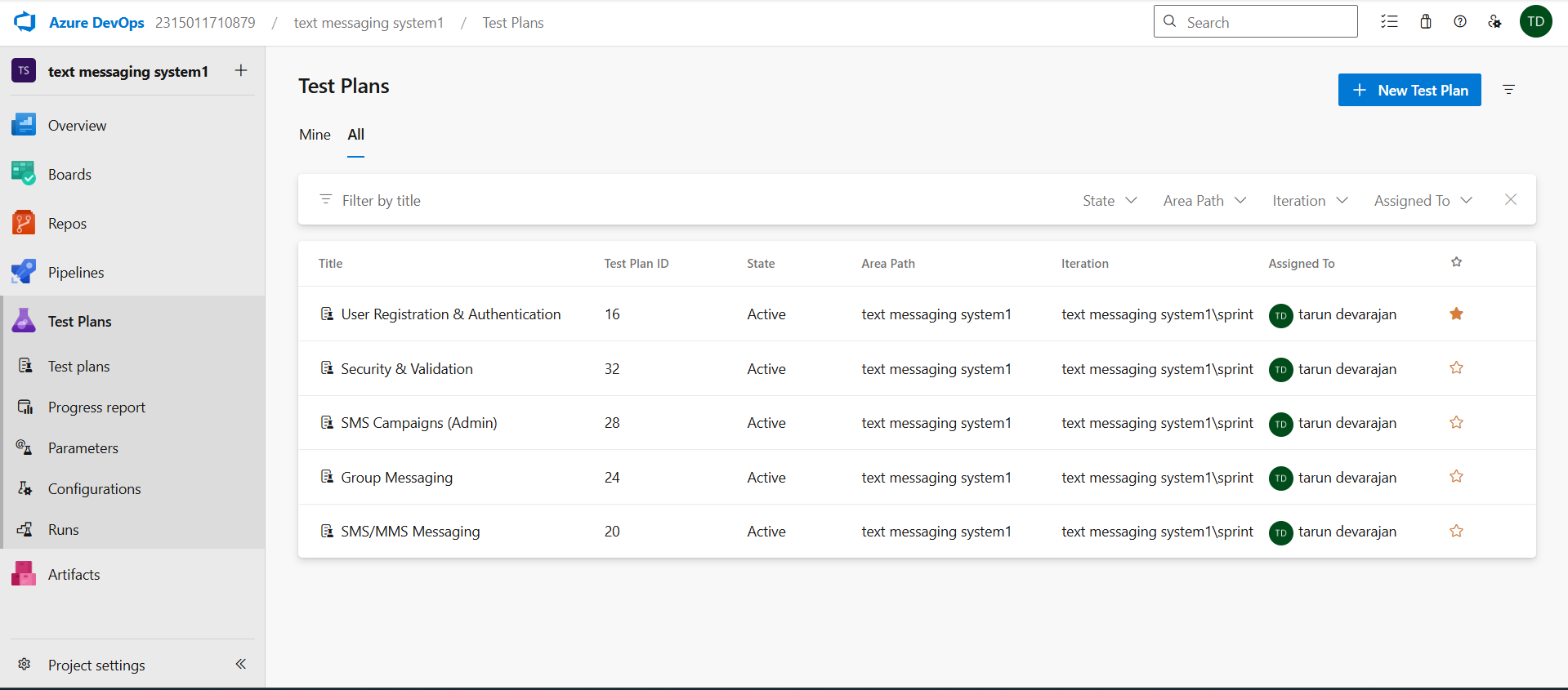
## TC10 – Attempt Campaign Without Target Segment

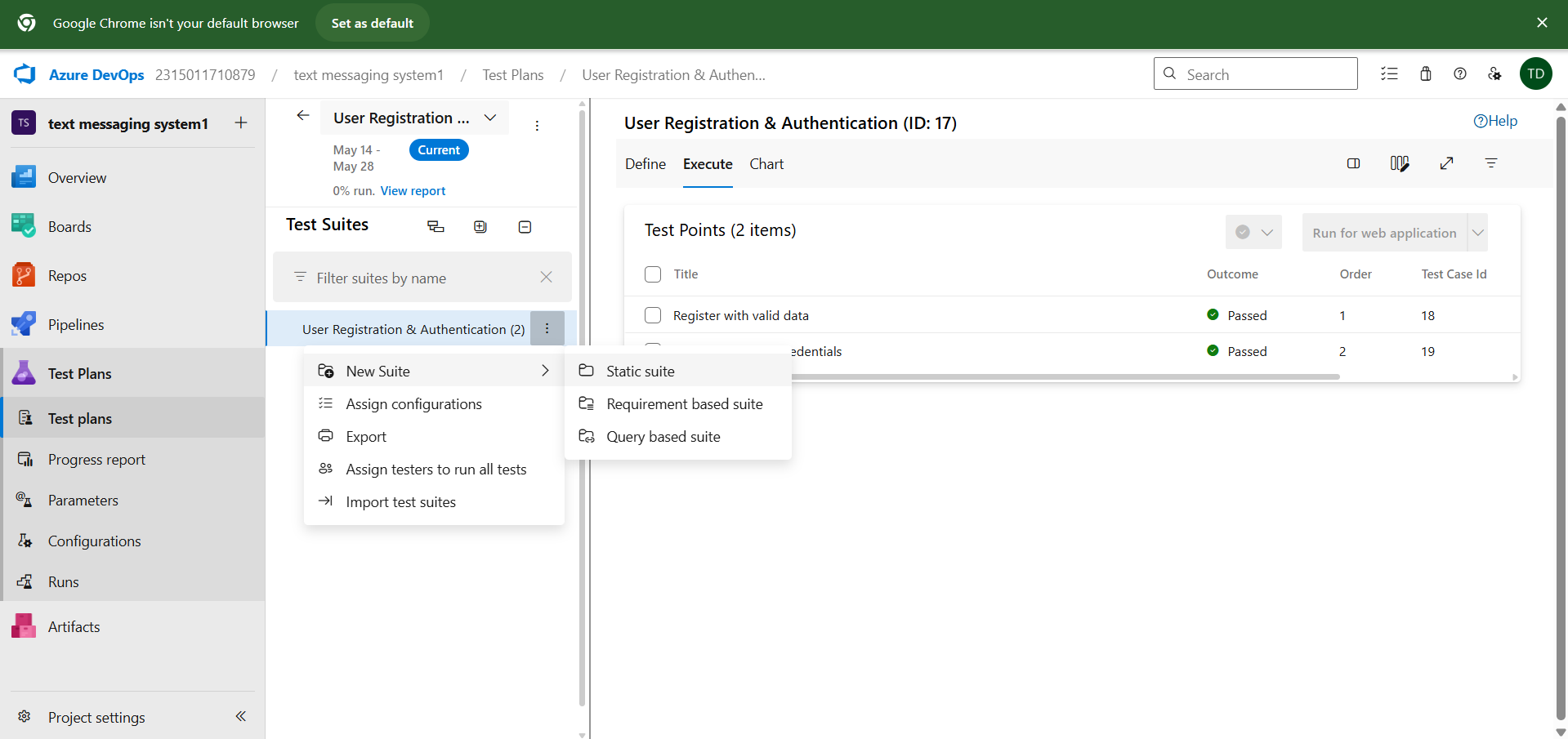
## Action: ▪ Open campaign interface ▪ Skip selecting the target audience ▪ Click “Send”

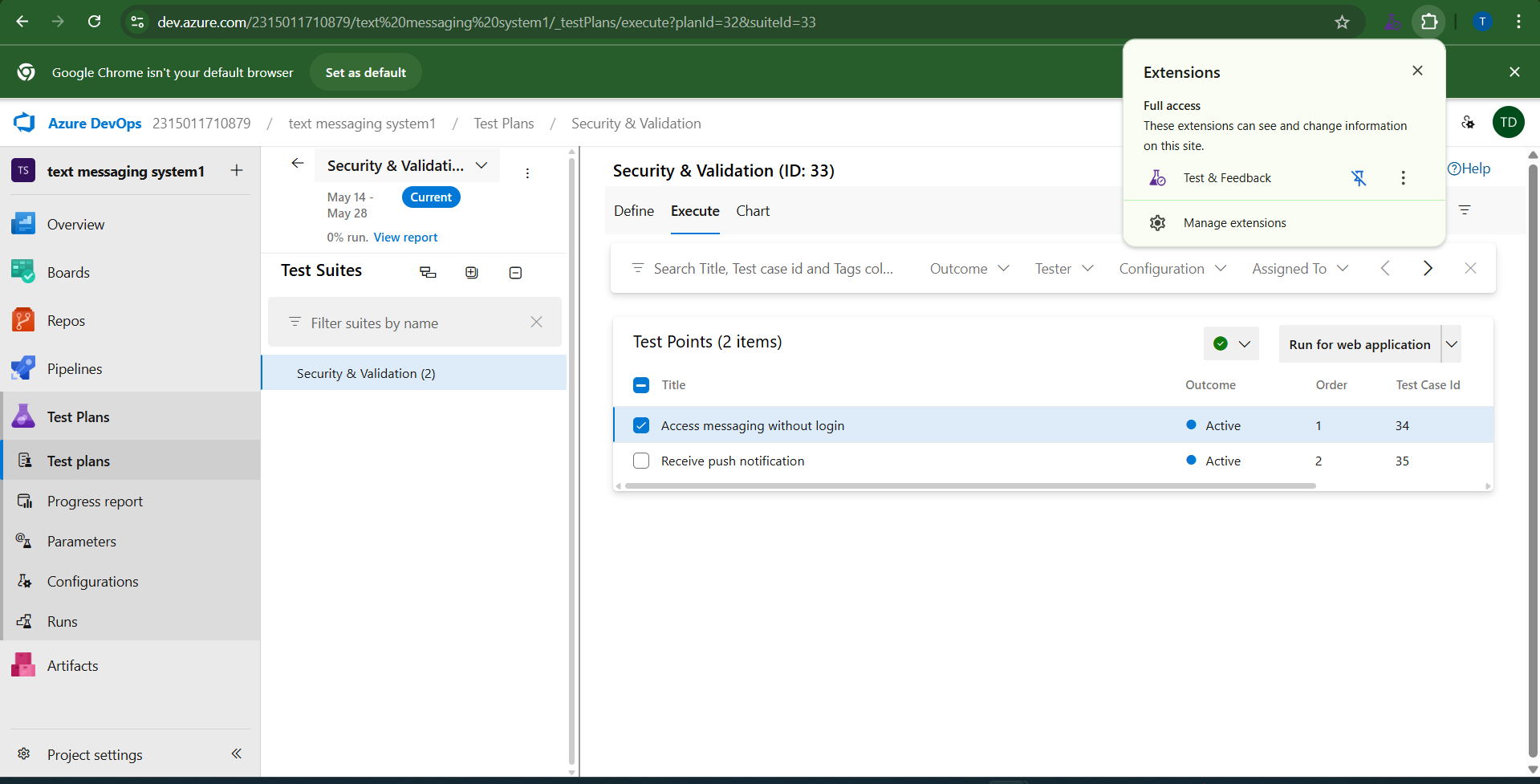
## Expected Result: ▪ Error: “No recipients selected for campaign.”

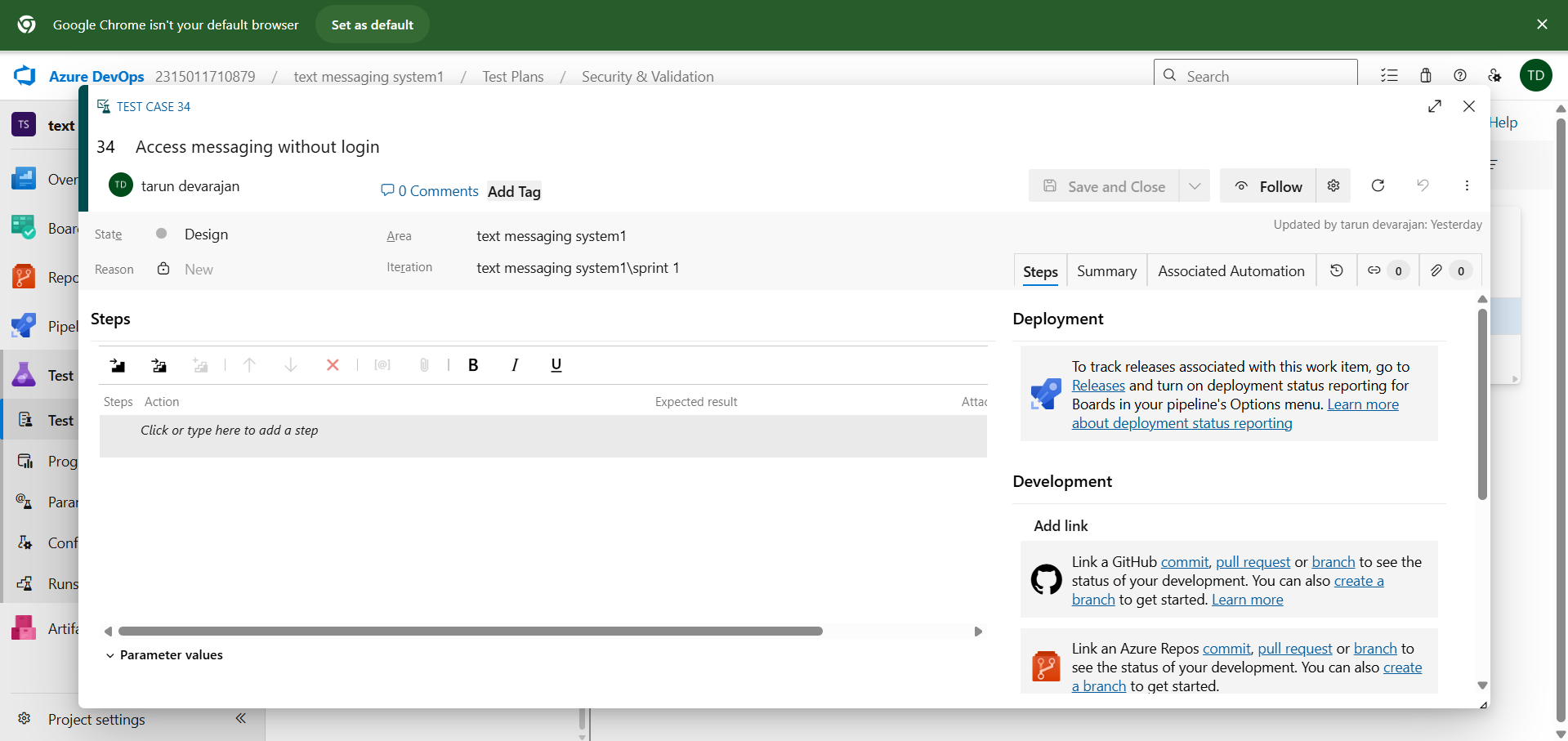
## Type: Error Path

## Test Cases

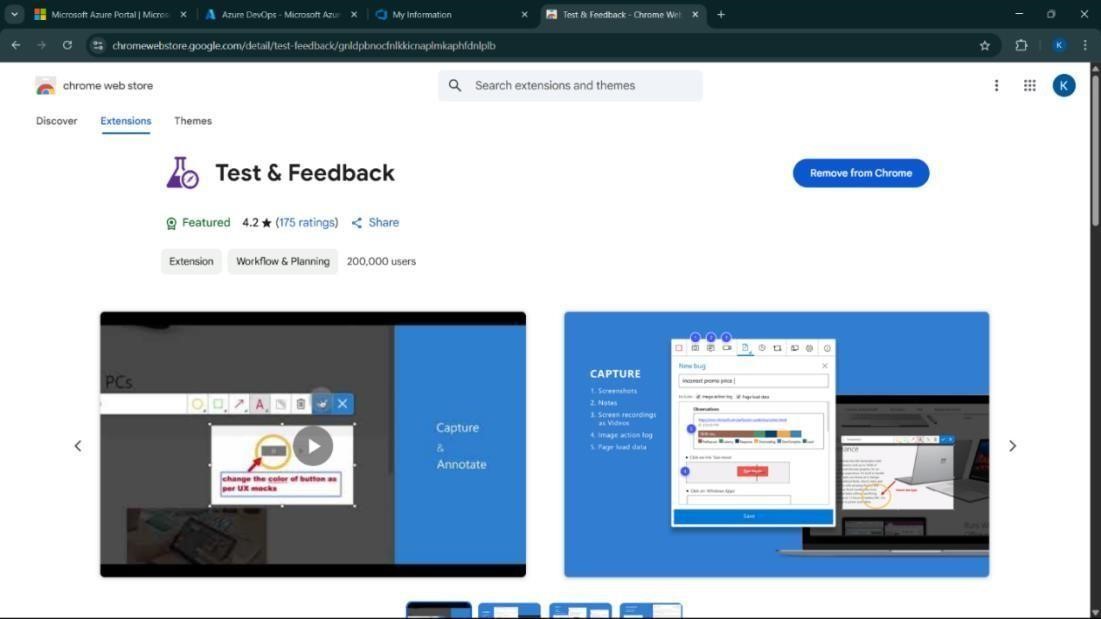
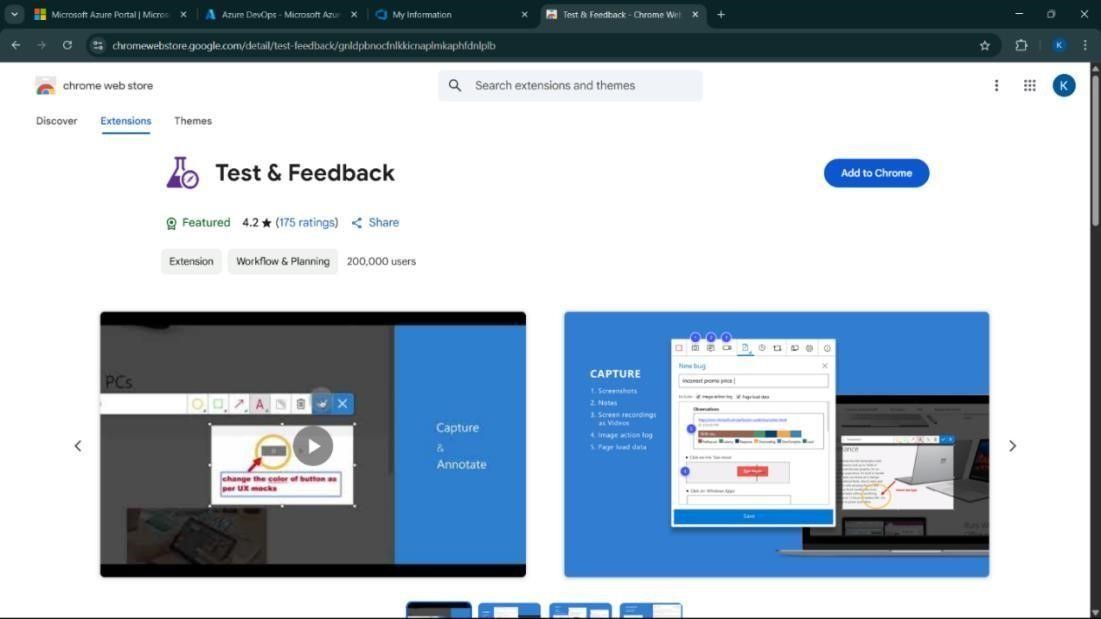




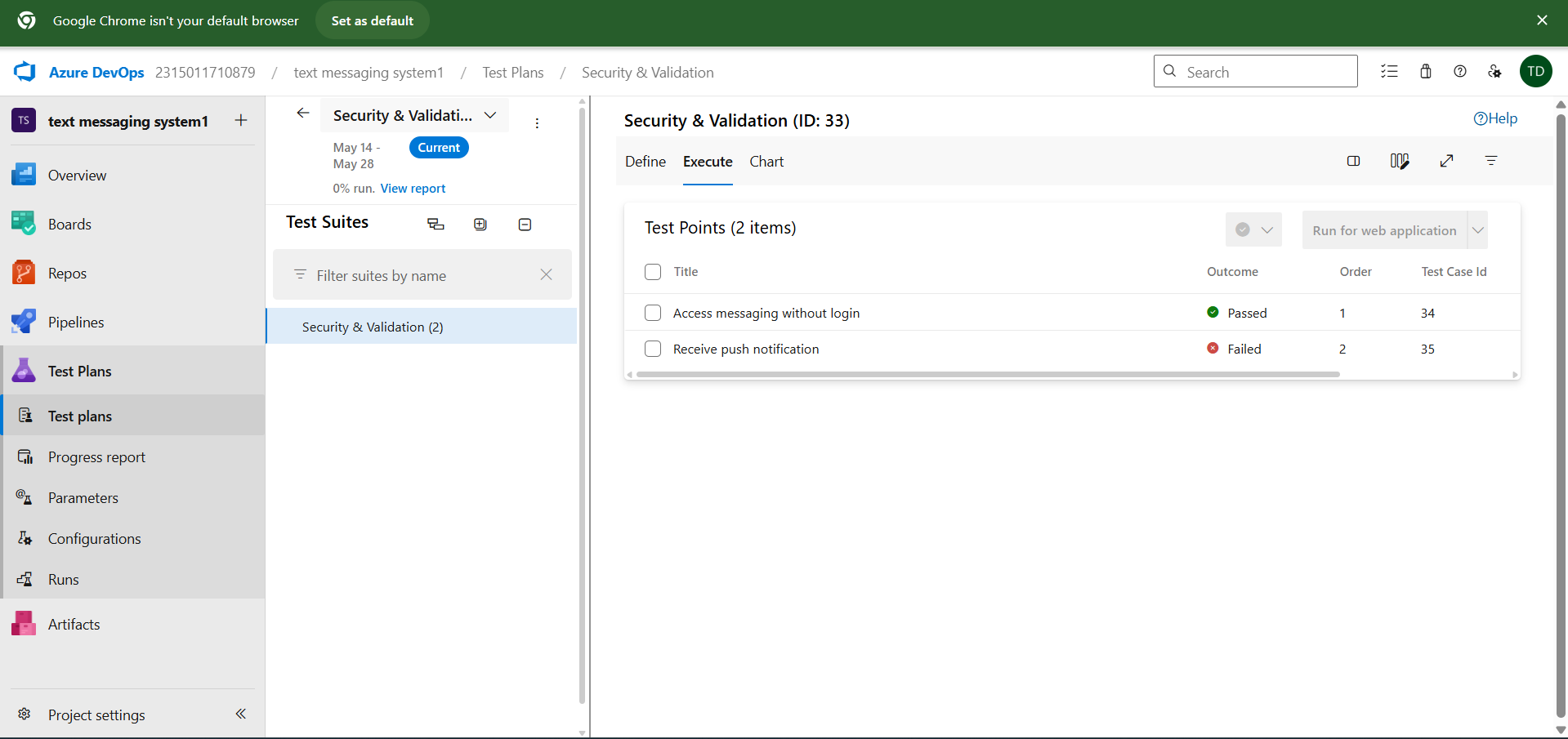




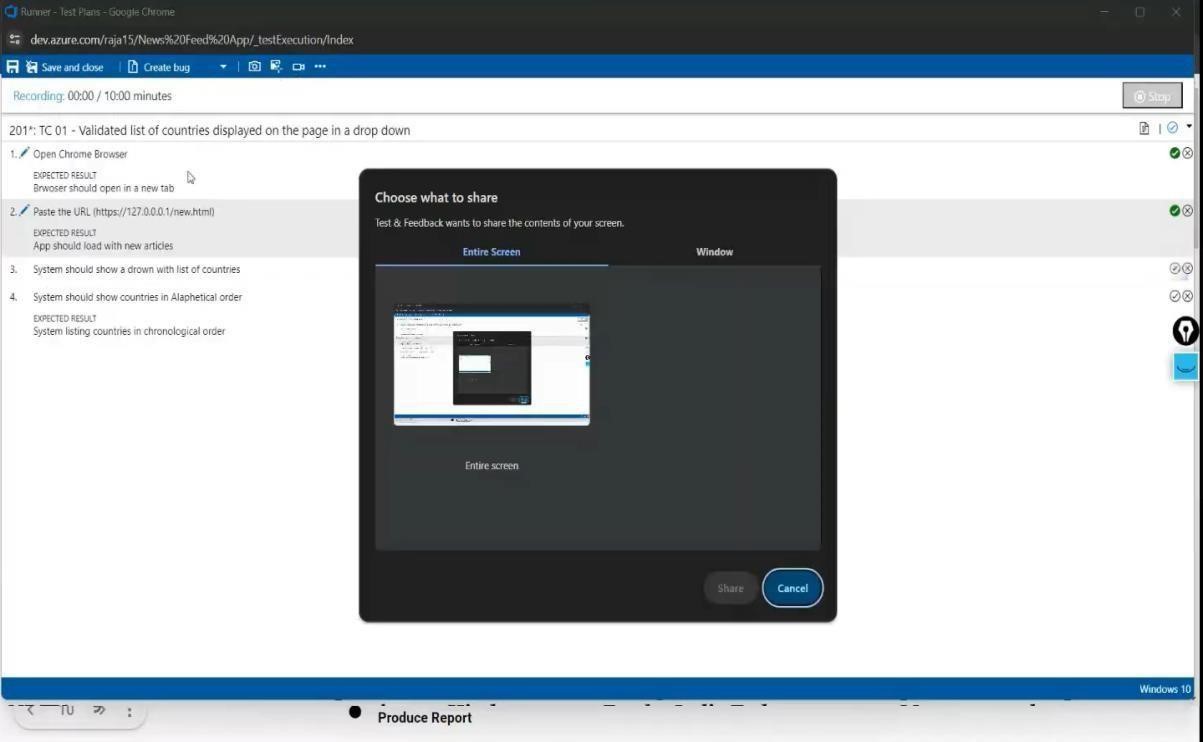
## 4.Installation of test



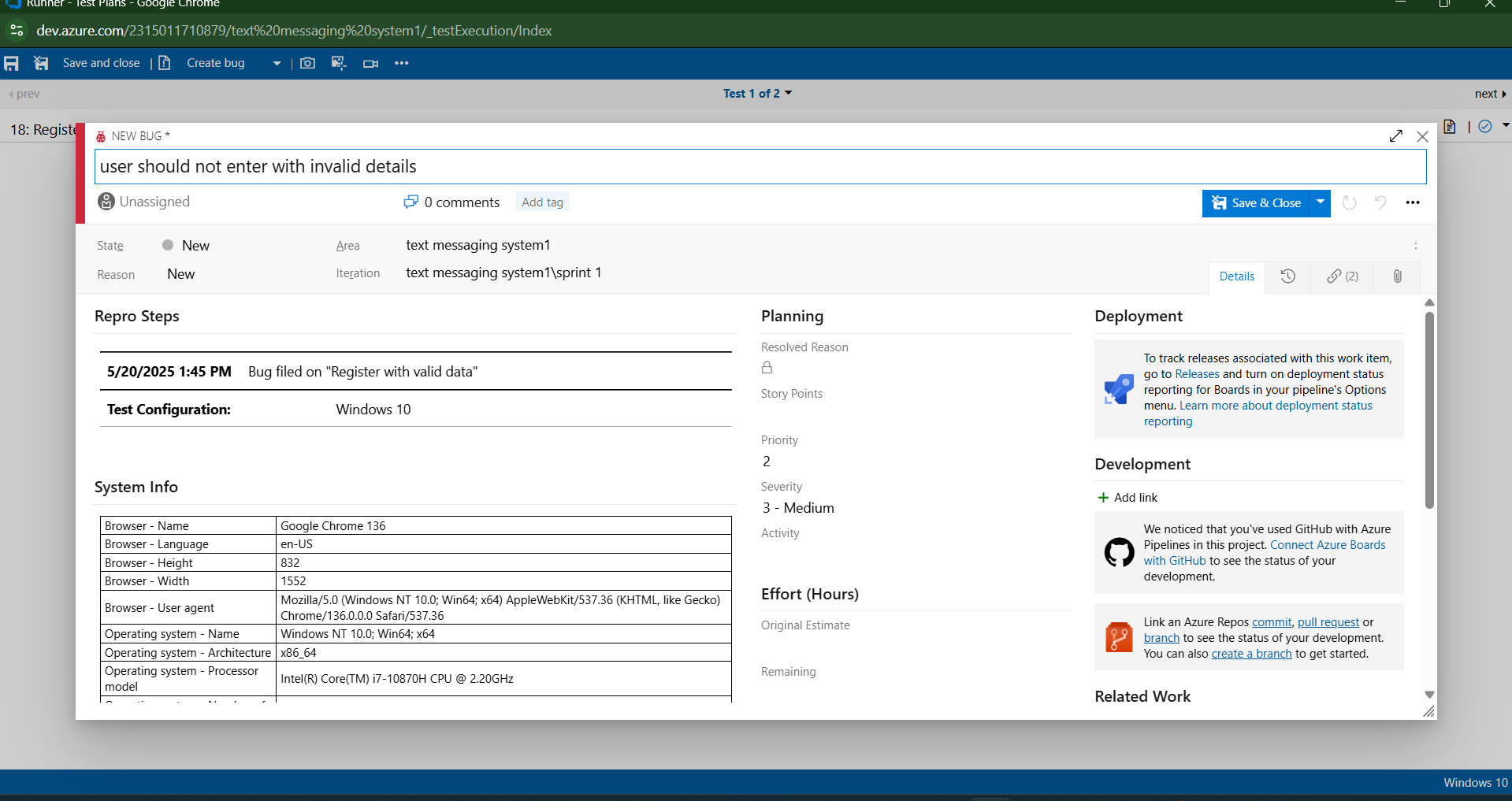
## 5.Running the test cases

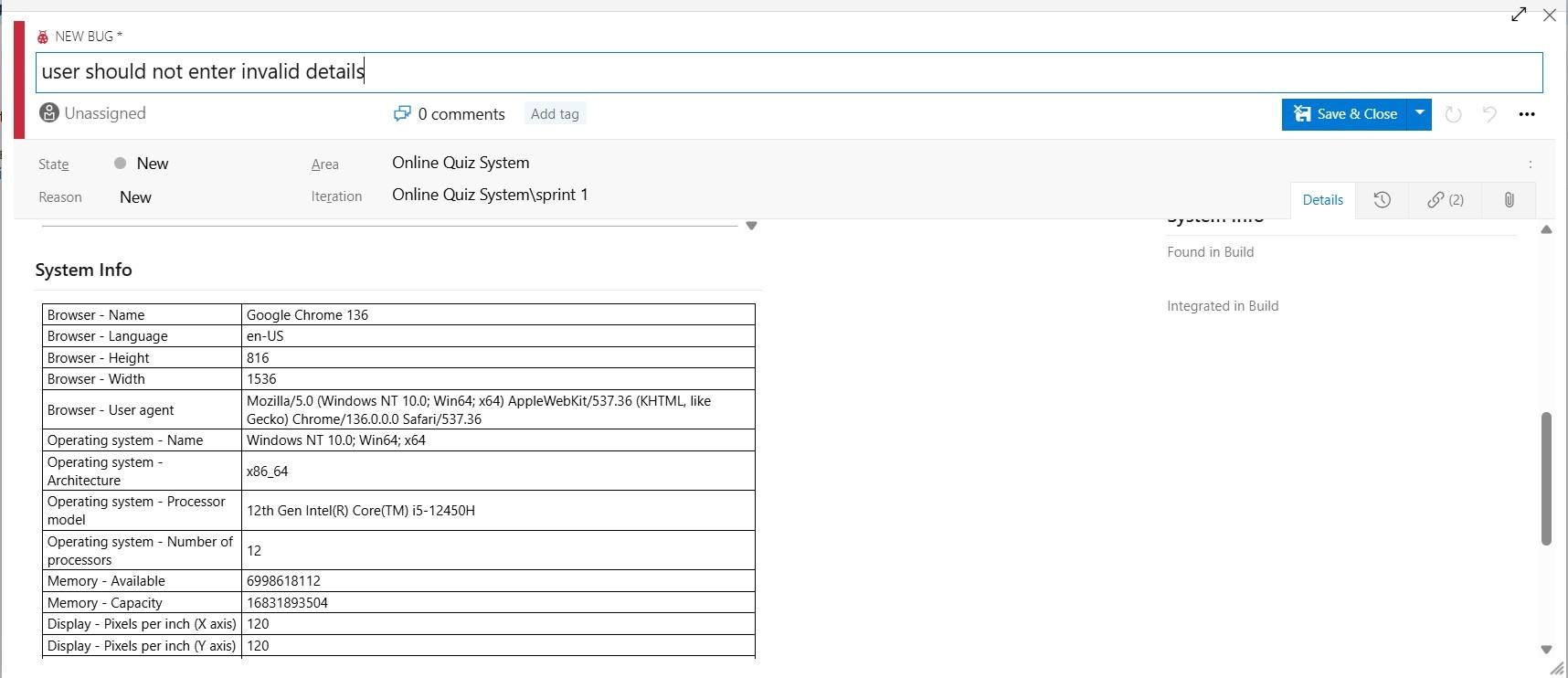


6.Recording the test case \



## 7.Creating the bug

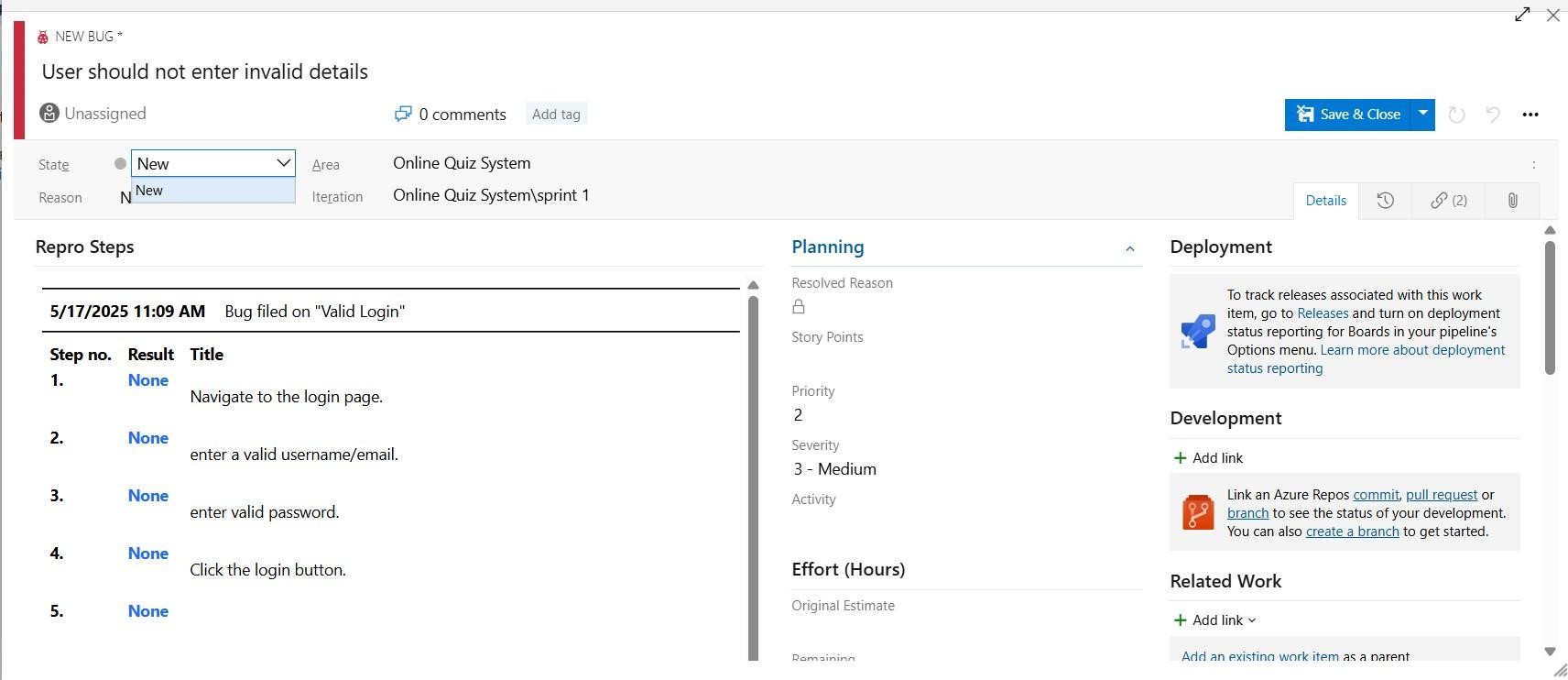




## 8.Test case results

## 

## 9.Test report summary

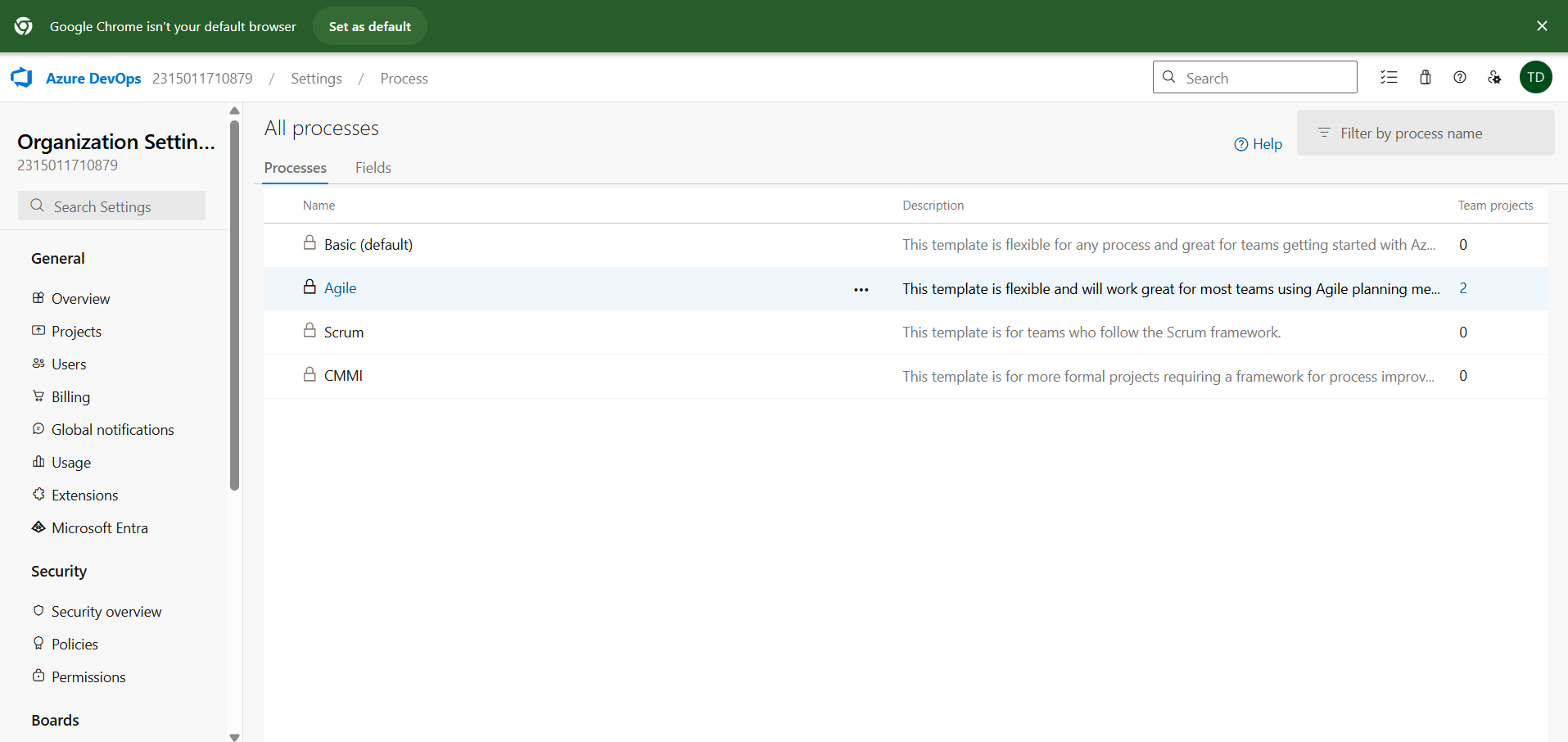


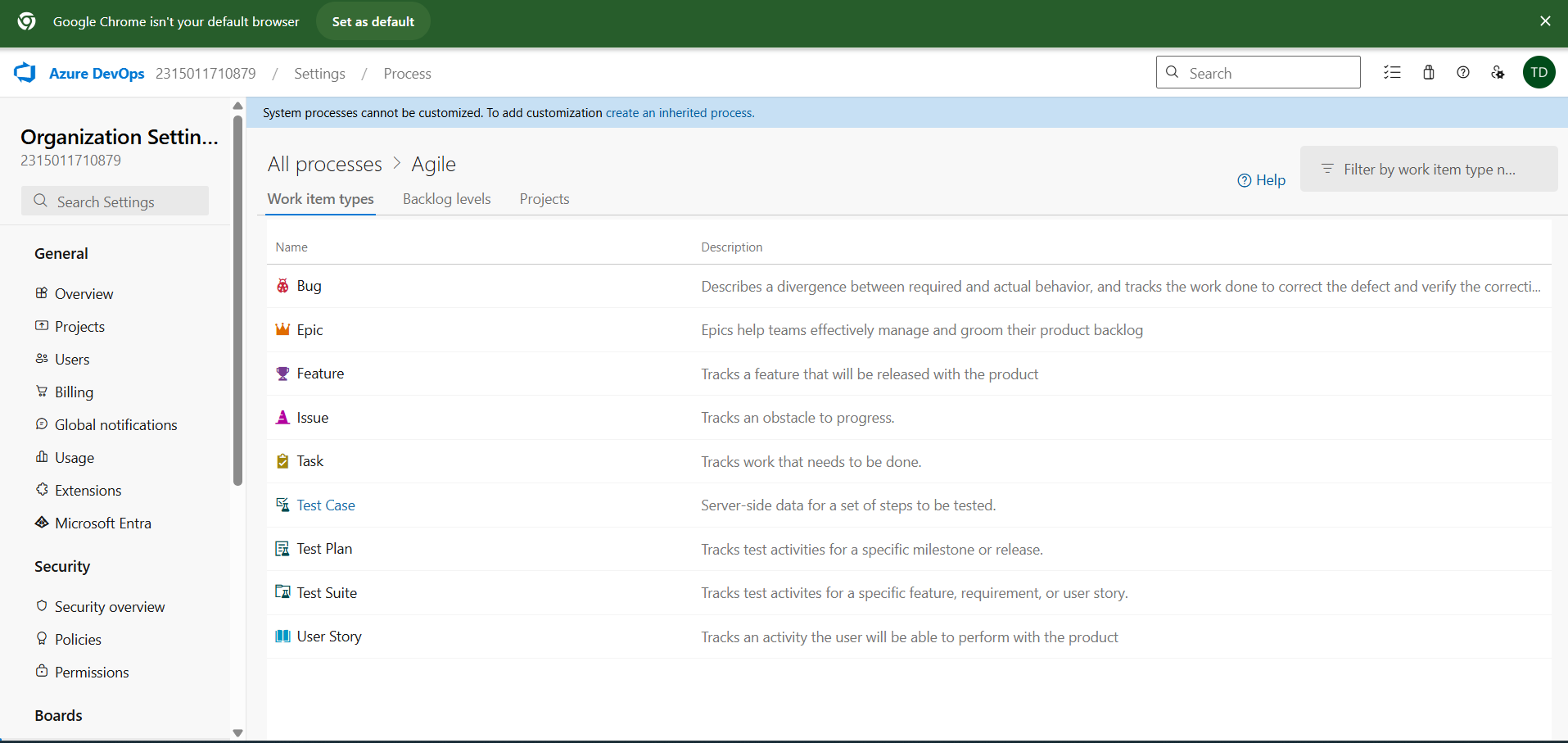
• Assigning bug to the developer and changing state

## 10.Progress report



## 11.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.

|  |  |
| --- | --- |
| **EXP 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 text/sms messaging 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.

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

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

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

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

1. 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.

1. 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.

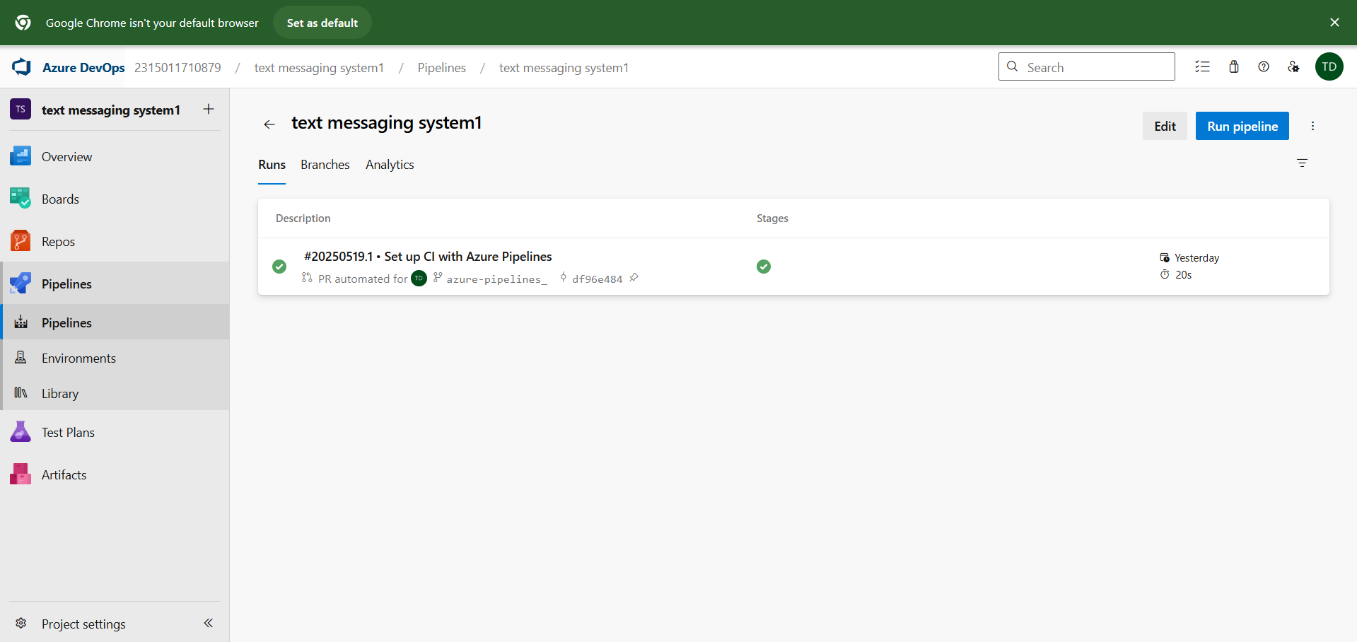
You can also integrate email alerts or Microsoft Teams notifications for build failures.

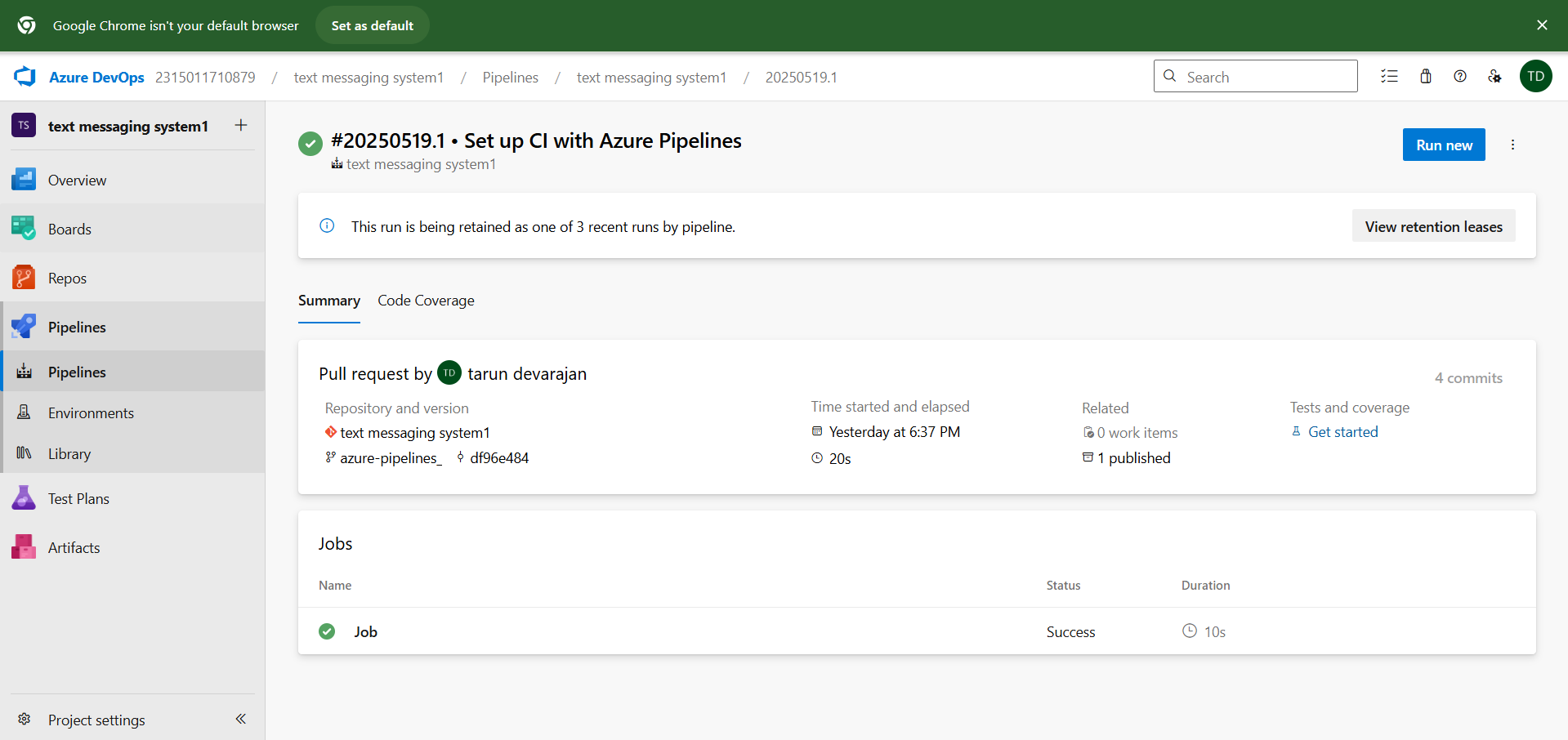
1. Review and Maintain Pipelines

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

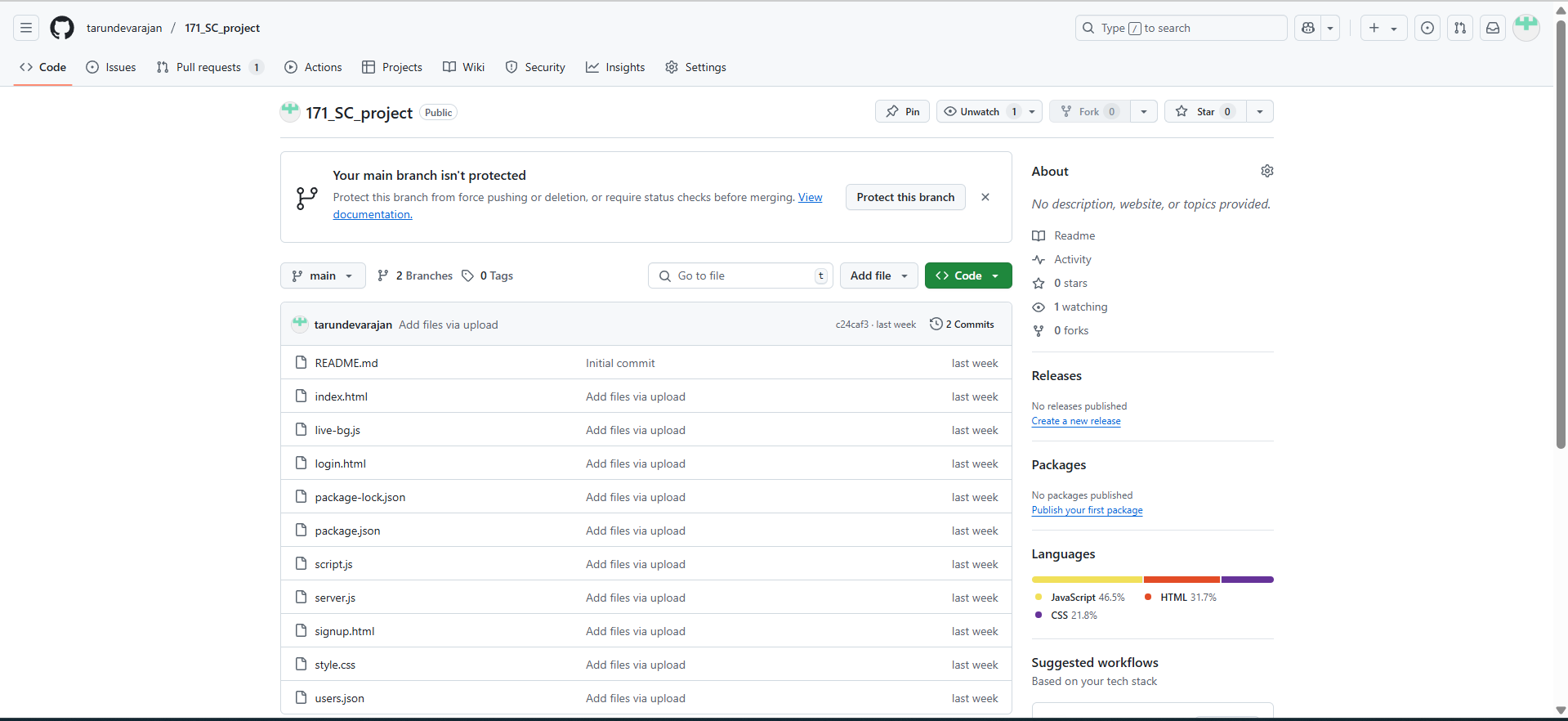
Thus, the pipelines for the given project **“Text/sms messaging system”** has been executed successfully.

|  |  |
| --- | --- |
| **EXP NO: 10** | **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 Text/sms messaging System.

## 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.