RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23432

SOFTWARE CONSTRUCTION

Laboratory Record Note Book

NAME: VARNIKA S

YEAR/BRANCH/SECTION:II/IT/AE

REGISTER NO:231001236

SEMESTER:IV

ACADEMIC YEAR:2025-2026

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM 602 105

BONAFIDE CERTIFICATE

NAME VARNIKA.S REGISTER NO. 231001236

ACADEMIC YEAR 2024-25 SEMESTER- IV BRANCH: B. Tech Information Technology [AE]. This Certification is the Bonafide record of work done by the above student in the CS23432- Software Construction Laboratory during the year 2024-2025.

Signature of Faculty -in Charge

Submitted for the Practical Examination held on _____

Internal Examiner External Examiner

Ex No	Date	Topic	Page No	Sign
1	21/01/2025	Study of Azure DevOps		
2	28/01/2025	Problem Statement		
3	04/02/2025	Agile Planning		
4	18/02/2025	Create User stories with Acceptance Criteria		
5	25/02/2025	Designing Sequence Diagrams using Azure DevOps-WIKI		
6	04/03/2025	Designing Class Diagram using Azure DevOps-WIKI		
7	11/03/2025	Designing Use case Diagram using Azure DevOps-WIKI		
8	18/03/2025	Designing Activity Diagrams using Azure DevOps-WIKI		
9	25/03/2025	Designing Architecture Diagram Using Star UML		
10	01/04/2025	Design User Interface		
11	08/04/2025	Implementation – Design a Web Page based on Scrum Methodology		

12	15/04/2025	Testing-Test Plan, Test Case and Load Testing		

EX NO:1

Study of Azure DevOps

AIM:

To study how to create an agile project in Azure DevOps environment.

STUDY:

Azure DevOps is a cloud-based platform by Microsoft that provides tools for DevOps practices, including CI/CD pipelines, version control, agile planning, testing, and monitoring. It supports teams in automating software development and deployment.

1. Understanding Azure DevOps

Azure DevOps consists of five key services:

1.1 Azure Repos (Version Control)

Supports Git repositories and Team Foundation Version Control (TFVC).

Provides features like branching, pull requests, and code reviews.

1.2 Azure Pipelines (CI/CD)

Automates build, test, and deployment processes.

Supports multi-platform builds (Windows, Linux, macOS).

Works with Docker, Kubernetes, Terraform, and cloud providers (Azure, AWS, GCP).

1.3 Azure Boards (Agile Project Management)

Manages work using Kanban boards, Scrum boards, and dashboards.

Tracks user stories, tasks, bugs, sprints, and releases.

1.4 Azure Test Plans (Testing)

Provides manual, exploratory, and automated testing.

Supports test case management and tracking.

1.5 Azure Artifacts (Package Management)

Stores and manages NuGet, npm, Maven, and Python packages.

Enables versioning and secure access to dependencies.

Getting Started with Azure DevOps

Step 1: Create an Azure DevOps Account Visit

Azure DevOps.

Sign in with a Microsoft Account.

Create an Organization and a Project.

Step 2: Set Up a Repository (Azure Repos) Navigate

to Repos.

Choose Git or TFVC for version control.

Clone the repository and push your code.

Step 3: Configure a CI/CD Pipeline (Azure Pipelines) Go

to Pipelines → New Pipeline.

Select a source code repository (Azure Repos, GitHub, etc.).
Define the pipeline using YAML or the Classic Editor.

Run the pipeline to build and deploy the application.

Step 4: Manage Work with Azure Boards Navigate

to Boards.

Create work items, user stories, and tasks.

Organize sprints and track progress.

Step 5: Implement Testing (Azure Test Plans) Go

to Test Plans.

Create and run test cases

View test results and track bugs.

Result:

The study was successfully completed.

EX NO:2

PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

Problem Statement:

Salary Management System

In modern organizations, efficient and accurate salary management is crucial for maintaining employee satisfaction, compliance with labor laws, and overall operational efficiency. Manual salary calculation and record-keeping often lead to errors, delays, and difficulty in tracking payroll history, tax deductions, bonuses, and other financial components.

A Salary Management System is required to automate and streamline the process of managing employee salaries. This system should handle employee data, compute monthly salaries based on attendance, leaves, allowances, and deductions, and generate payslips. It should also allow HR and finance departments to manage tax calculations, overtime, bonuses, and maintain historical salary records in a secure and efficient manner.

Result:

The problem statement was written successfully.

EX NO:3

AGILE PLANNING

Aim:

To prepare an Agile Plan.

THEORY

Agile planning is a part of the Agile methodology, which is a project management style with an incremental, iterative approach. Instead of using an in-depth plan from the start of the project—which is typically product-related—Agile leaves room for requirement changes throughout and relies on constant feedback from end users.

With Agile planning, a project is broken down into smaller, more manageable tasks with the ultimate goal of having a defined image of a project's vision. Agile planning involves looking at different aspects of a project's tasks and how they'll be achieved, for example:

· Roadmaps to guide a product's release ad schedule

· Sprints to work on one specific group of tasks at a time

· A feedback plan to allow teams to stay flexible and easily adapt to change

User stories, or the tasks in a project, capture user requirements from the end user's perspective Essentially, with Agile planning, a team would decide on a set of user stories to action at any given time, using them as a guide to implement new features or functionalities in a tool. Looking at tasks as user stories is a helpful way to imagine how a customer may use a feature and helps teams prioritize work and focus on delivering value first.

· Steps in Agile planning process

1. Define vision
2. Set clear expectations on goals
3. Define and break down the product roadmap
4. Create tasks based on user stories
5. Populate product backlog
6. Plan iterations and estimate effort
7. Conduct daily stand-ups
8. Monitor and adapt

Result:

Thus the Agile plan was completed successfully.

EX NO : 4

CREATE USER STORIES

Aim:

To create User Stories

THEORY

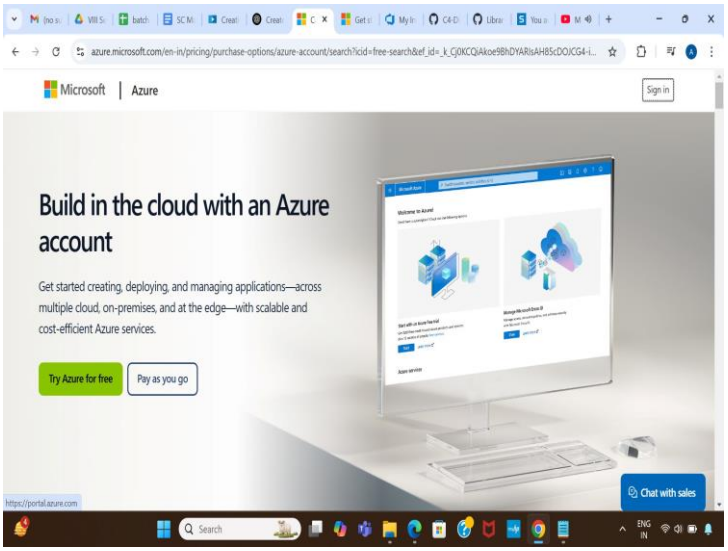
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

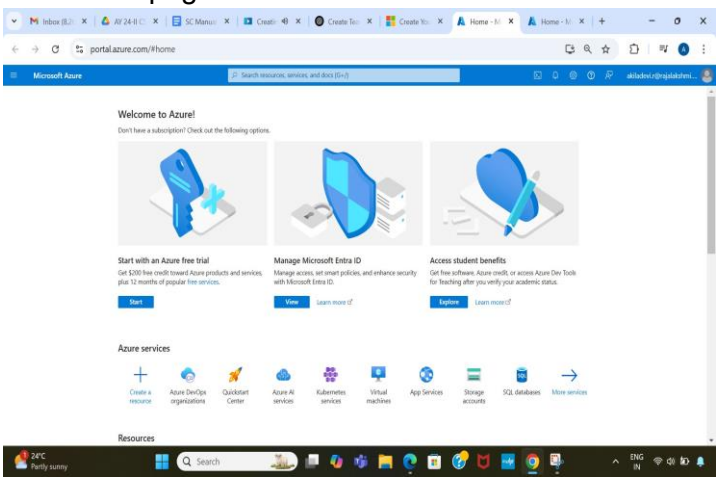
"As a [role], I [want to], [so that]."

Procedure:

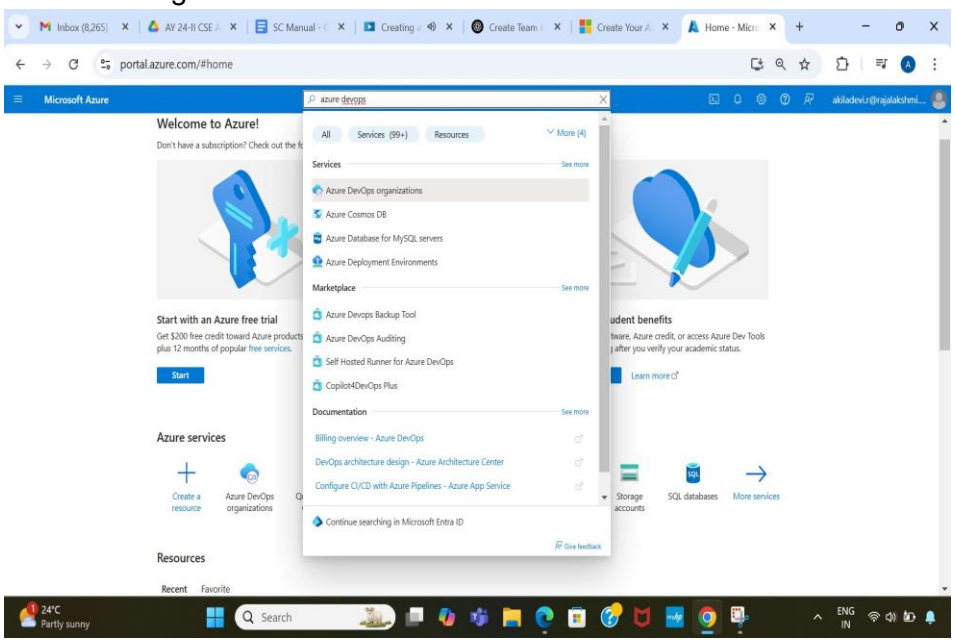
1. Open your web browser and go to the Azure website: *https://azure.microsoft.com/en-in* Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for *https://signup.live.com/?lic=1*



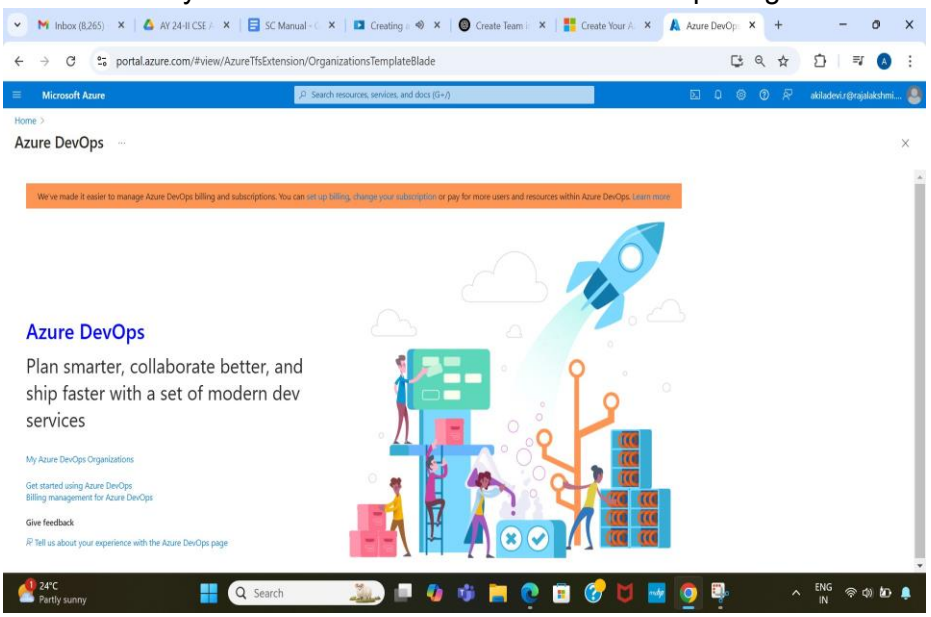
3. Azure home page

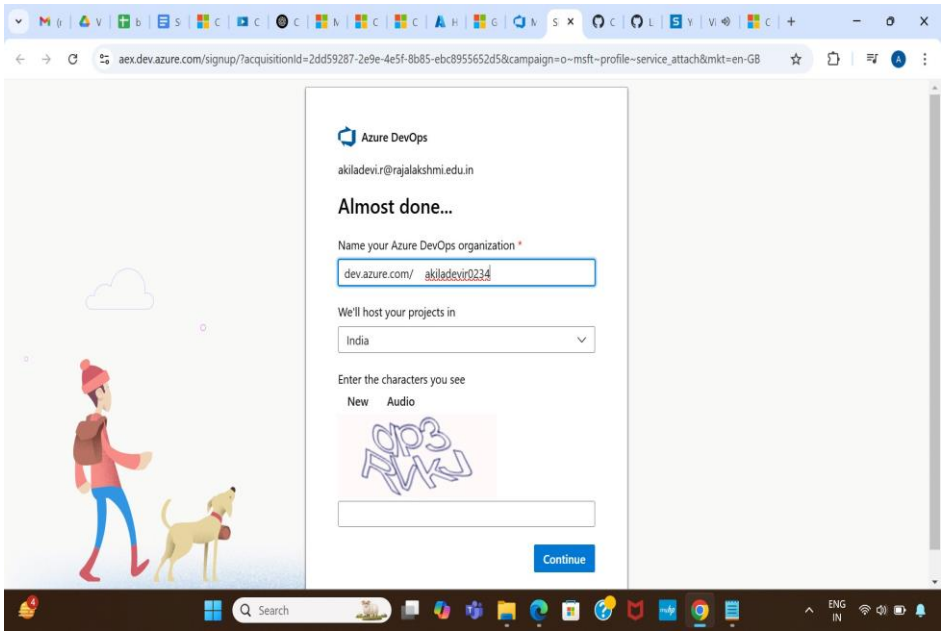


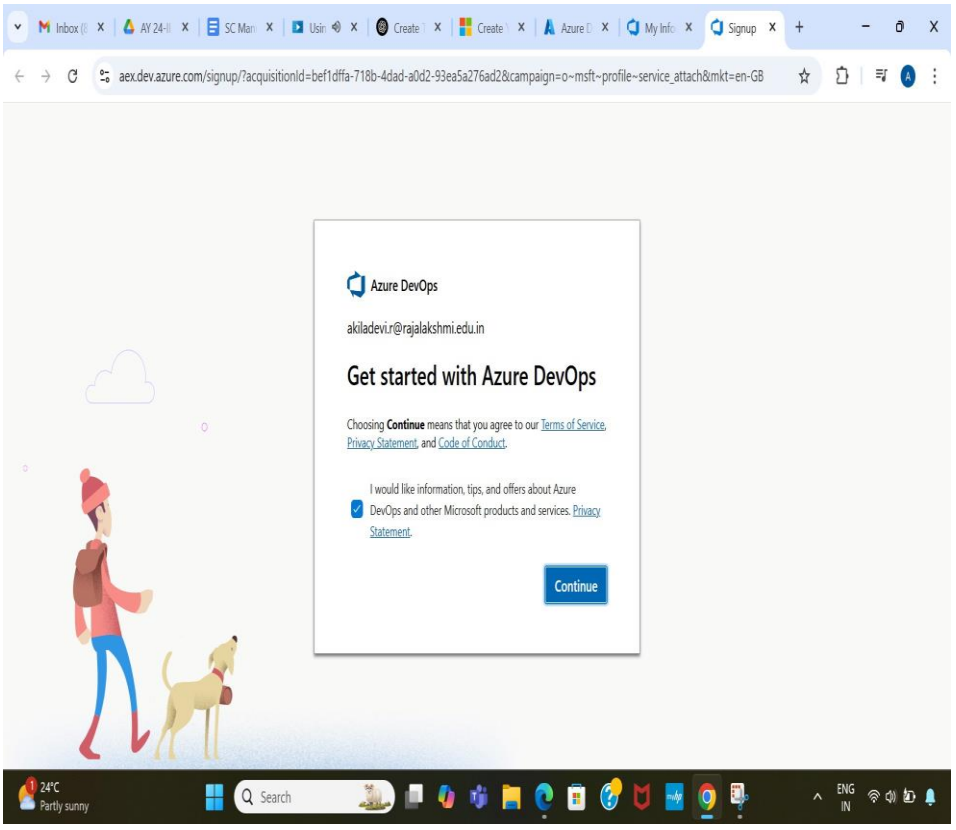
4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



5. Click on the My Azure DevOps Organization link and create an organization and you should be taken to the Azure DevOps Organization Home page.




aex.dev.azure.com/signup/?acquisitionId=2dd59287-2e9e-4e5f-8b85-ebc8955652d5&campaign=o~msft~profile~service_attach&mkt=en-GB
Azure DevOps
akiladevi.r@rajalakshmi.edu.in
Almost done...
Name your Azure DevOps organization *
dev.azure.com/ akiladevir0234
We'll host your projects in
India
Enter the characters you see
New Audio
Continue
Search
ENG
IN

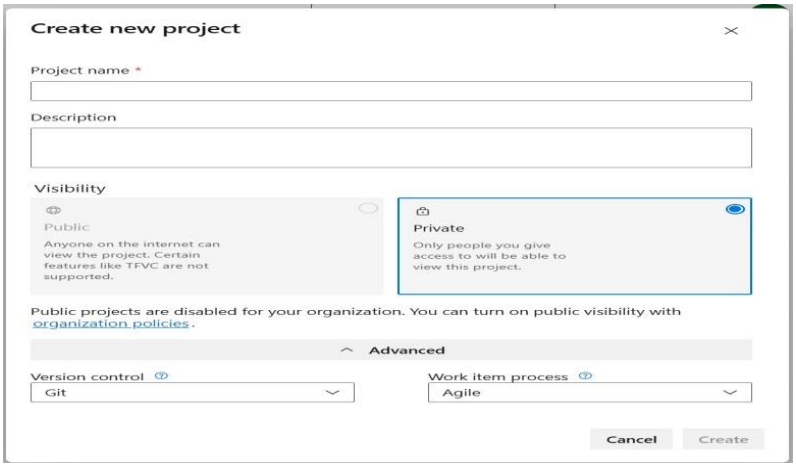

aex.dev.azure.com/signup/?acquisitionId=bef1dffa-718b-4dad-a0d2-93ea5a276ad2&campaign=o~msft~profile~service_attach&mkt=en-GB
Azure DevOps
akiladevi.r@rajalakshmi.edu.in
Get started with Azure DevOps
Choosing Continue means that you agree to our Terms of Service, Privacy Statement, and Code of Conduct.
I would like information, tips, and offers about Azure DevOps and other Microsoft products and services. Privacy Statement.
Continue
24°C
Partly sunny
Search
ENG
IN

1. Create the First Project in Your Organization

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.

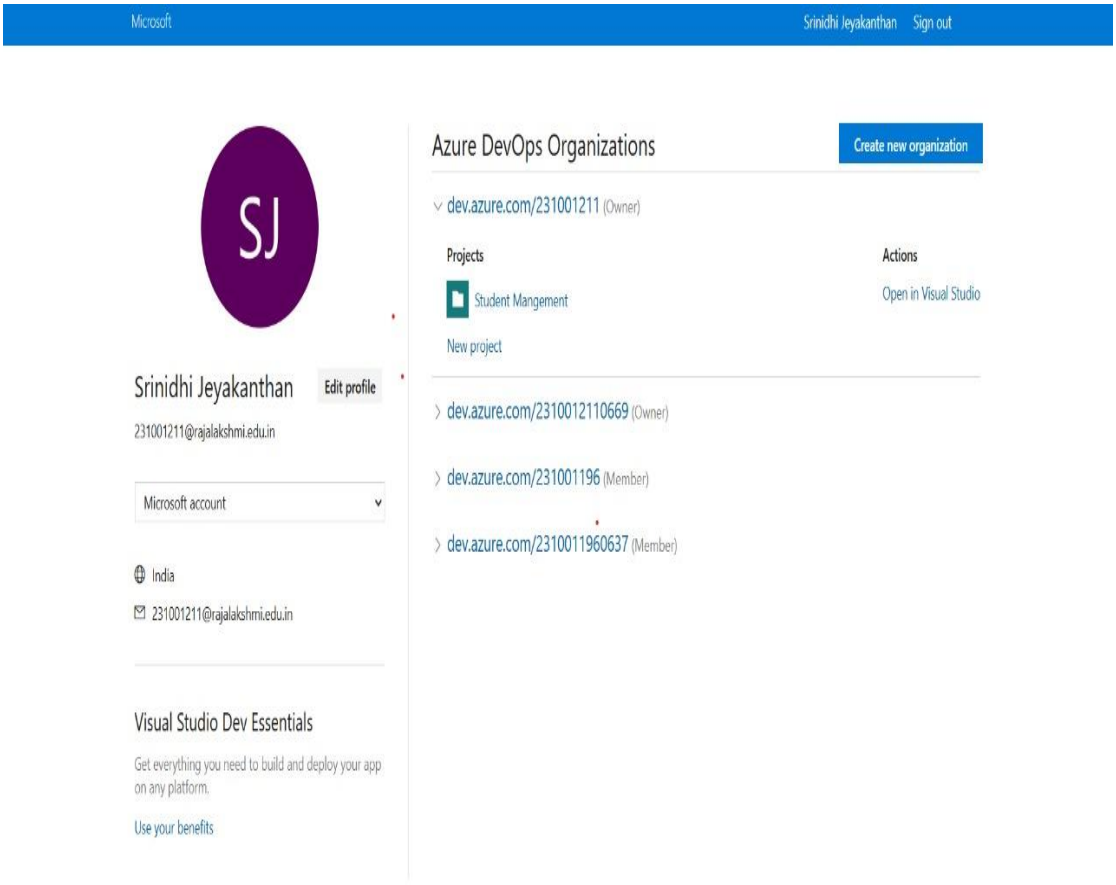
2. On the organization's **Home page**, click on the **New Project** button
3. Enter the project name, description, and visibility options:
    - **Name**: Choose a name for the project (e.g., `LMS`).
    - **Description**: Optionally, add a description to provide more context about the project.
    - **Visibility**: Choose whether you want the project to be **Private** (accessible only to those invited) or **Public** (accessible to anyone).

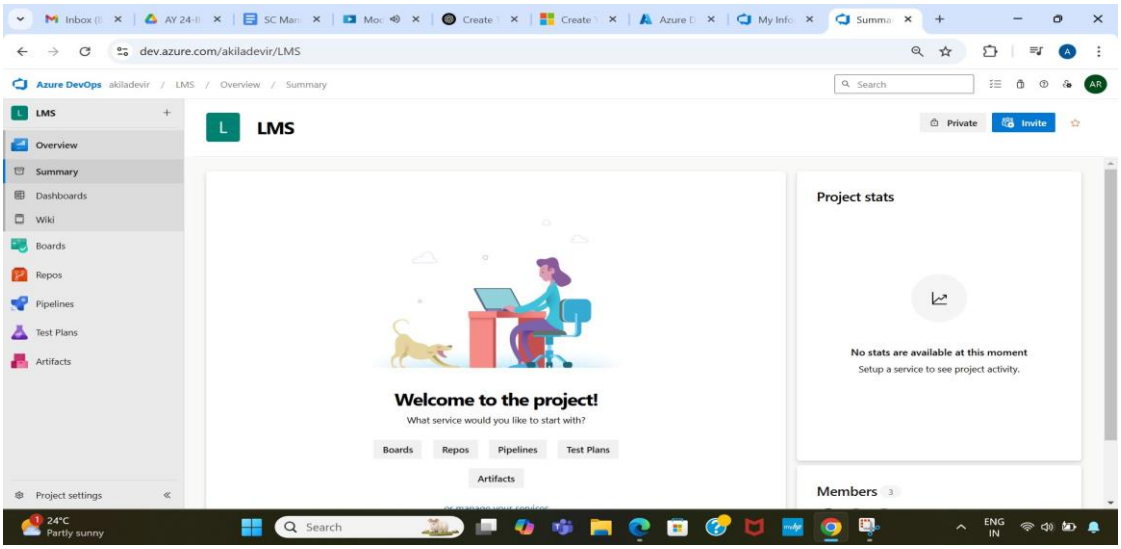
4.Once you've filled out the details, click **Create** to set up your first project.



5.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.

6.Project dashboard

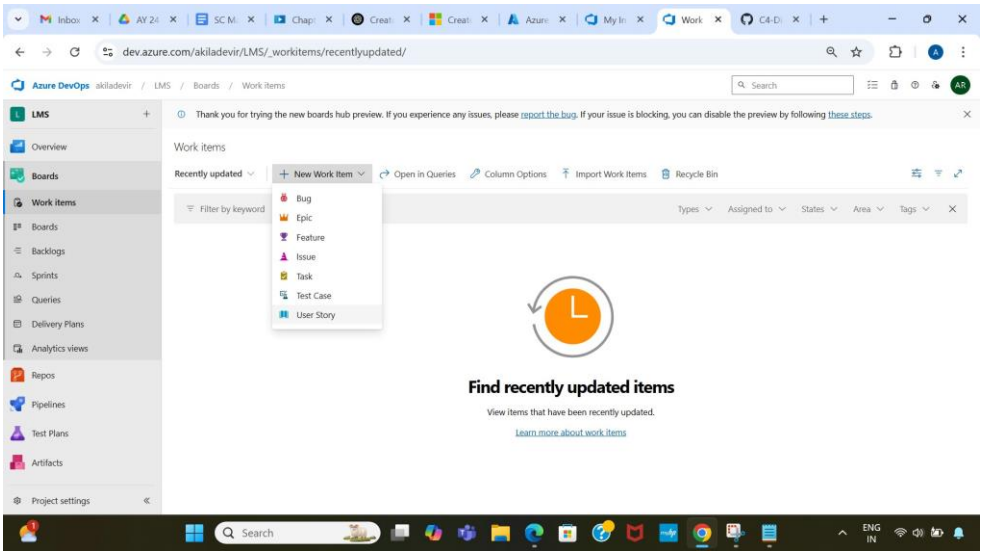

Microsoft
Srinidhi Jeyakanthan Sign out
SJ
Srinidhi Jeyakanthan
Edit profile
231001211@rajalakshmi.edu.in
Microsoft account
India
231001211@rajalakshmi.edu.in
Visual Studio Dev Essentials
Get everything you need to build and deploy your app on any platform.
Use your benefits
Azure DevOps Organizations
Create new organization
dev.azure.com/231001211 (Owner)
Projects
Actions
Student Mangement
Open in Visual Studio
New project
dev.azure.com/2310012110669 (Owner)
dev.azure.com/231001196 (Member)
dev.azure.com/2310011960637 (Member)



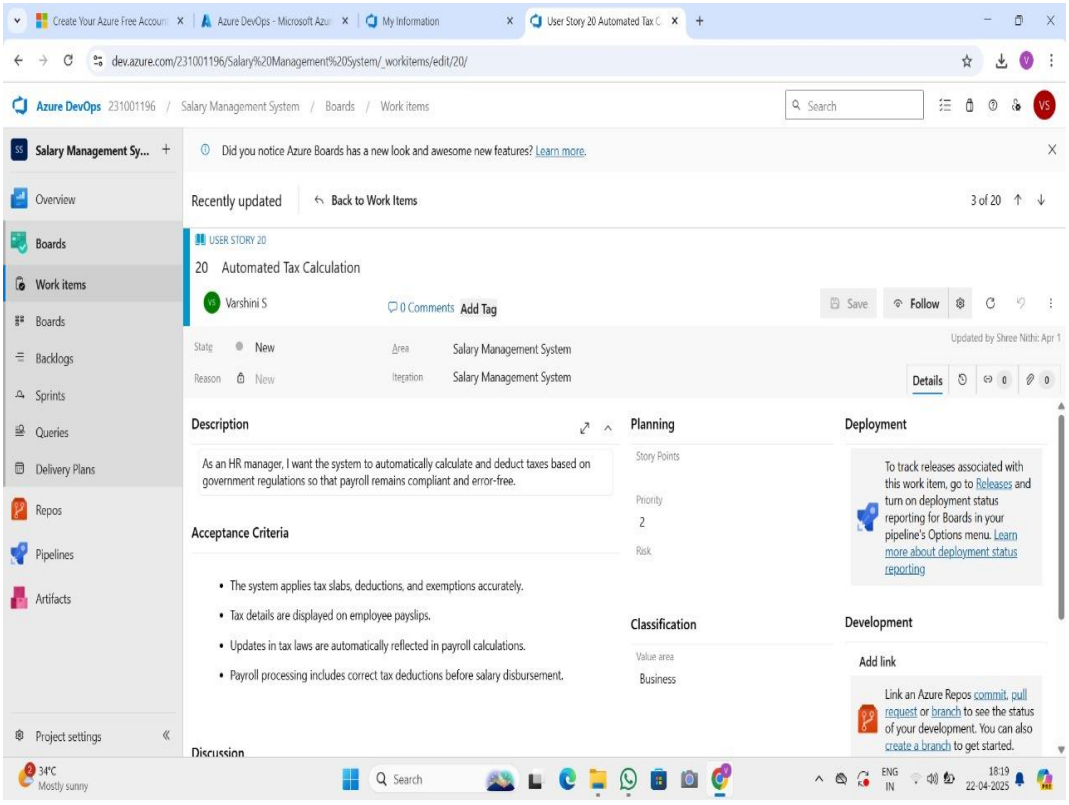
7.To manage user stories

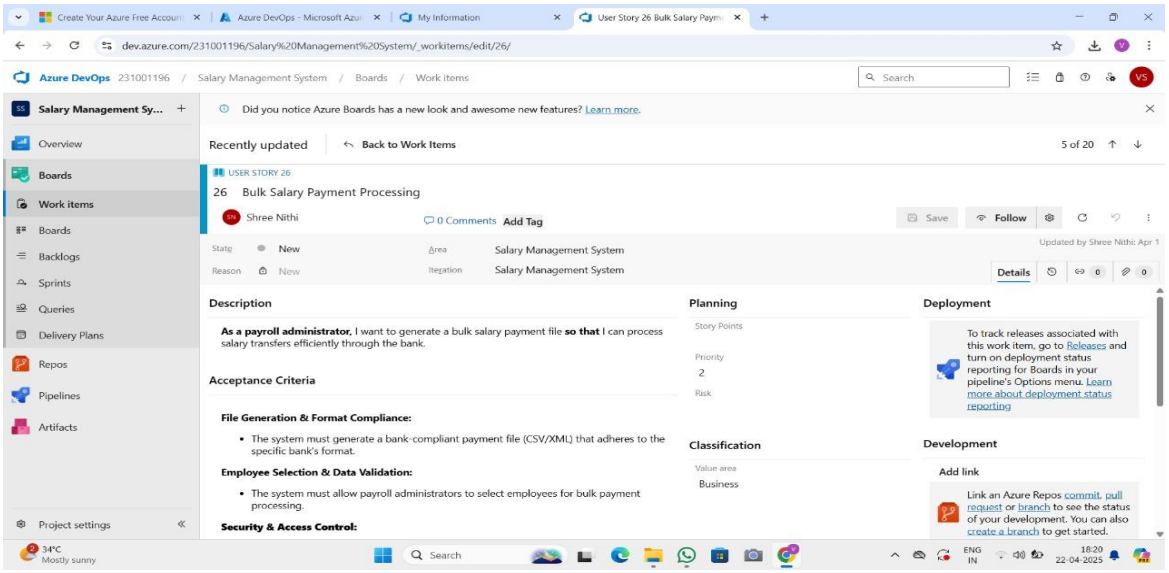
From the **left-hand navigation menu**, click on **Boards**. This will take you to the main **Boards** page, where you can manage work items, backlogs, and sprints.

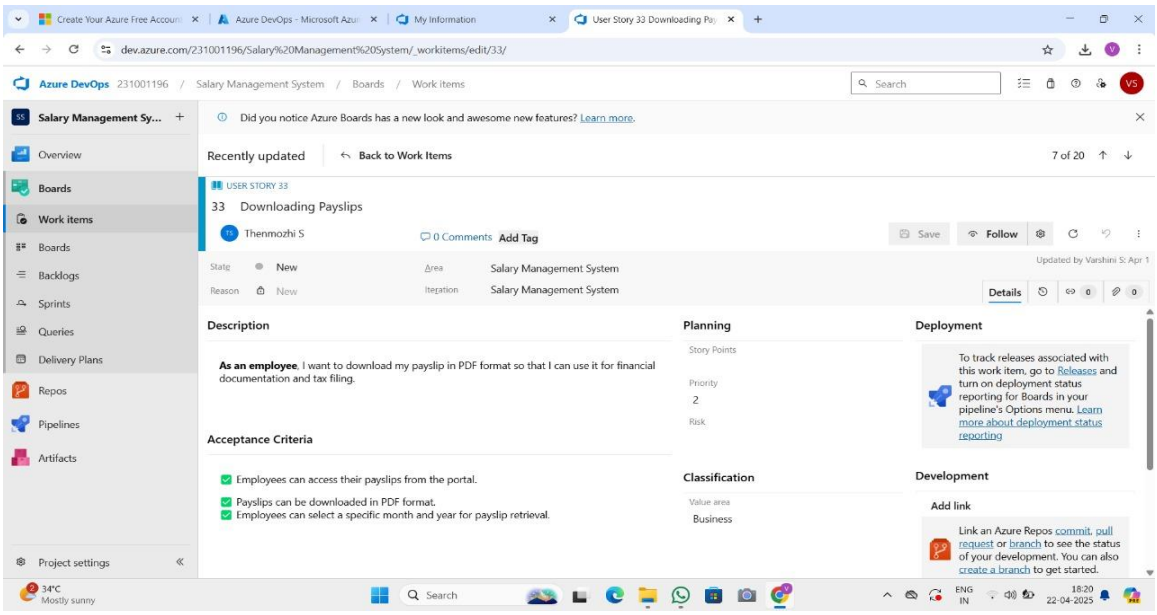
On the **work items** page, you'll see the option to **Add a work item** at the top. Alternatively, you can find a **+** button or **Add New Work Item** depending on the view you're in. From the **Add a work item** dropdown, select **User Story**. This will open a form to enter details for the new User Story.

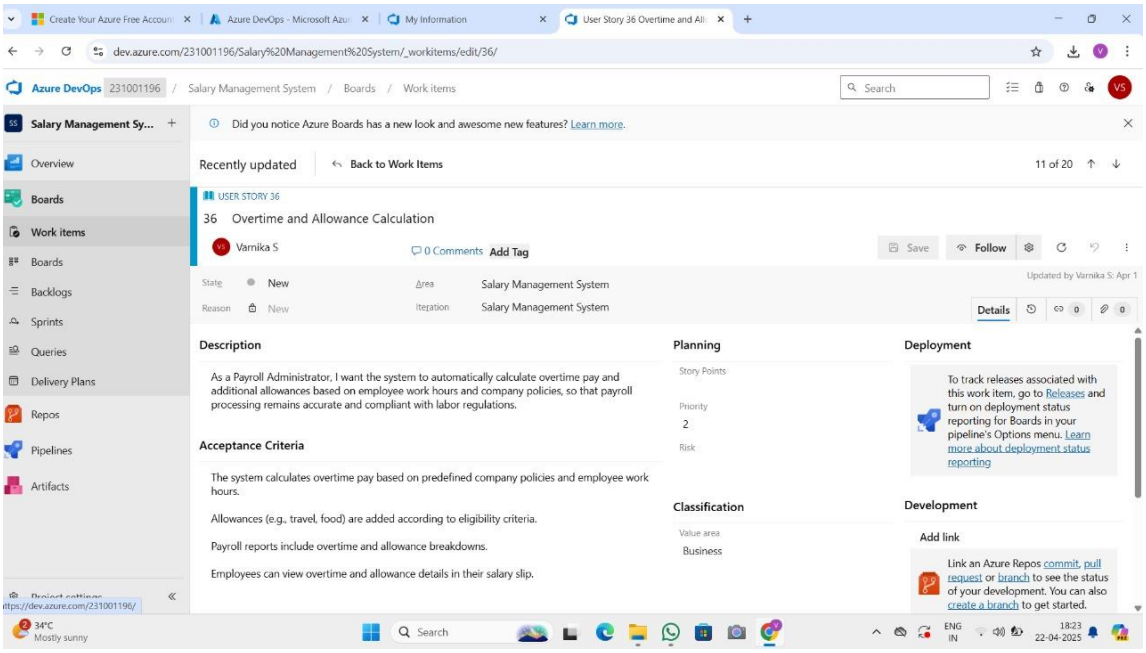


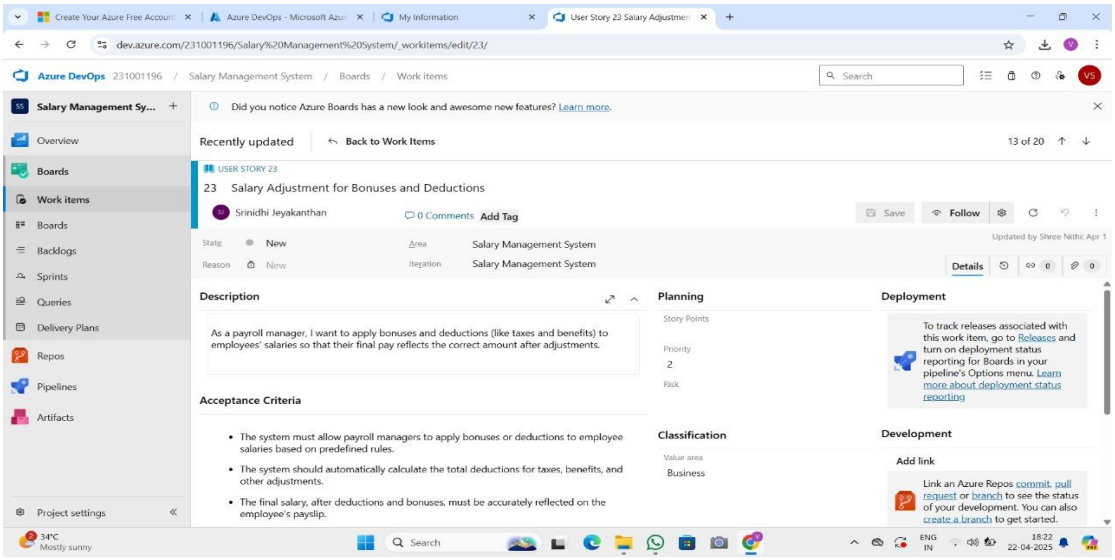
1 Fill in User Story Details




Create Your Azure Free Account
Azure DevOps - Microsoft Azure
My Information
User Story 26 Bulk Salary Payment
dev.azure.com/231001196/Salary%20Management%20System/_workitems/edit/26/
Azure DevOps 231001196 / Salary Management System / Boards / Work items
Search
Salary Management Sy...
Did you notice Azure Boards has a new look and awesome new features? Learn more.
Overview
Boards
Work items
Boards
Backlogs
Sprints
Queries
Delivery Plans
Repos
Pipelines
Artifacts
Project settings
Recently updated
Back to Work Items
5 of 20
USER STORY 26
26 Bulk Salary Payment Processing
Shree Nithi
0 Comments
Add Tag
Save
Follow
Updated by Shree Nithi: Apr 1
State
New
Reason
New
Area
Salary Management System
Iteration
Salary Management System
Details
Description
As a payroll administrator, I want to generate a bulk salary payment file so that I can process salary transfers efficiently through the bank.
Acceptance Criteria
File Generation & Format Compliance:
The system must generate a bank-compliant payment file (CSV/XML) that adheres to the specific bank's format.
Employee Selection & Data Validation:
The system must allow payroll administrators to select employees for bulk payment processing.
Security & Access Control:
Planning
Story Points
Priority
2
Risk
Classification
Value area
Business
Deployment
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting
Development
Add link
Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started.
34°C
Mostly sunny
Search
ENG
IN
18:20
22-04-2025


Create Your Azure Free Account
Azure DevOps - Microsoft Azure
My Information
User Story 33 Downloading Pay
dev.azure.com/231001196/Salary%20Management%20System/_workitems/edit/33/
Azure DevOps 231001196 / Salary Management System / Boards / Work items
Search
Salary Management Sy...
Did you notice Azure Boards has a new look and awesome new features? Learn more.
Overview
Boards
Work items
Boards
Backlogs
Sprints
Queries
Delivery Plans
Repos
Pipelines
Artifacts
Project settings
Recently updated
Back to Work Items
7 of 20
USER STORY 33
33 Downloading Payslips
Thenmozhi S
0 Comments
Add Tag
Save
Follow
Updated by Varshini S: Apr 1
State
New
Reason
New
Area
Salary Management System
Iteration
Salary Management System
Details
Description
As an employee, I want to download my payslip in PDF format so that I can use it for financial documentation and tax filing.
Acceptance Criteria
Employees can access their payslips from the portal.
Payslips can be downloaded in PDF format.
Employees can select a specific month and year for payslip retrieval.
Planning
Story Points
Priority
2
Risk
Classification
Value area
Business
Deployment
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting
Development
Add link
Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started.
34°C
Mostly sunny
Search
ENG
IN
18:20
22-04-2025





Result:

The user story was written successfully.

EX NO: 5

SEQUENCE DIAGRAM

Aim:

To design a Sequence Diagram by using Mermaid.js

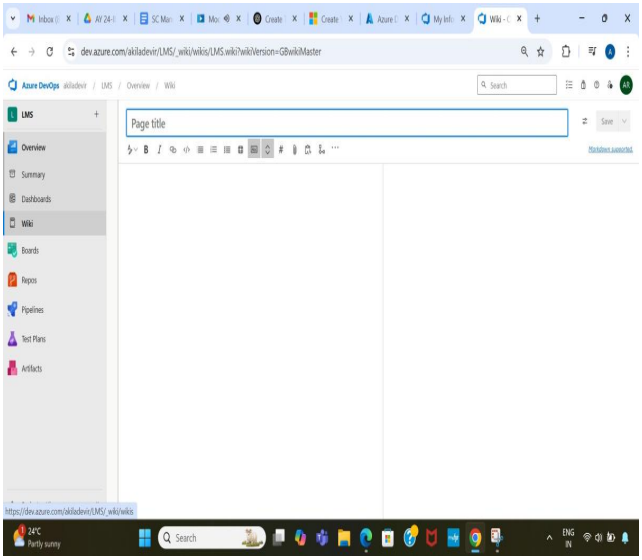
THEORY:

A Sequence Diagram is a key component of Unified Modelling Language (UML) used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modelling dynamic behaviour in a system.

Procedure:

1.Open a project in Azure DevOps Organisations.

2.To design select wiki from menu



3.Write code for drawing sequence diagram and save the code.

```
::: mermaid
  sequenceDiagram
   participant Admin
   participant System
   participant DB as Database

   Admin->>System: Login
   System->>DB: Validate credentials
   DB-->>System: Authentication success
   System-->>Admin: Show Dashboard

   Admin->>System: Add/Update Employee
   System->>DB: Insert/Update Employee record
```

```
    DB-->>System: Confirm changes

    Admin->>System: Add Attendance
    System->>DB: Insert attendance record
    DB-->>System: Confirm attendance entry

    Admin->>System: Generate Salary
    System->>DB: Fetch employee, position, attendance
    DB-->>System: Return data
    System->>System: Calculate salary
    System->>DB: Insert salary record
    DB-->>System: Confirm salary stored

    System-->>Admin: Show confirmation & salary slip
```

:::

4.Explanation:

participant defines the entities involved (Admin, System, Database).

->> represents a direct message sent between participants (e.g., Admin to System).

-->> represents a response message (e.g., Database to System after validation).

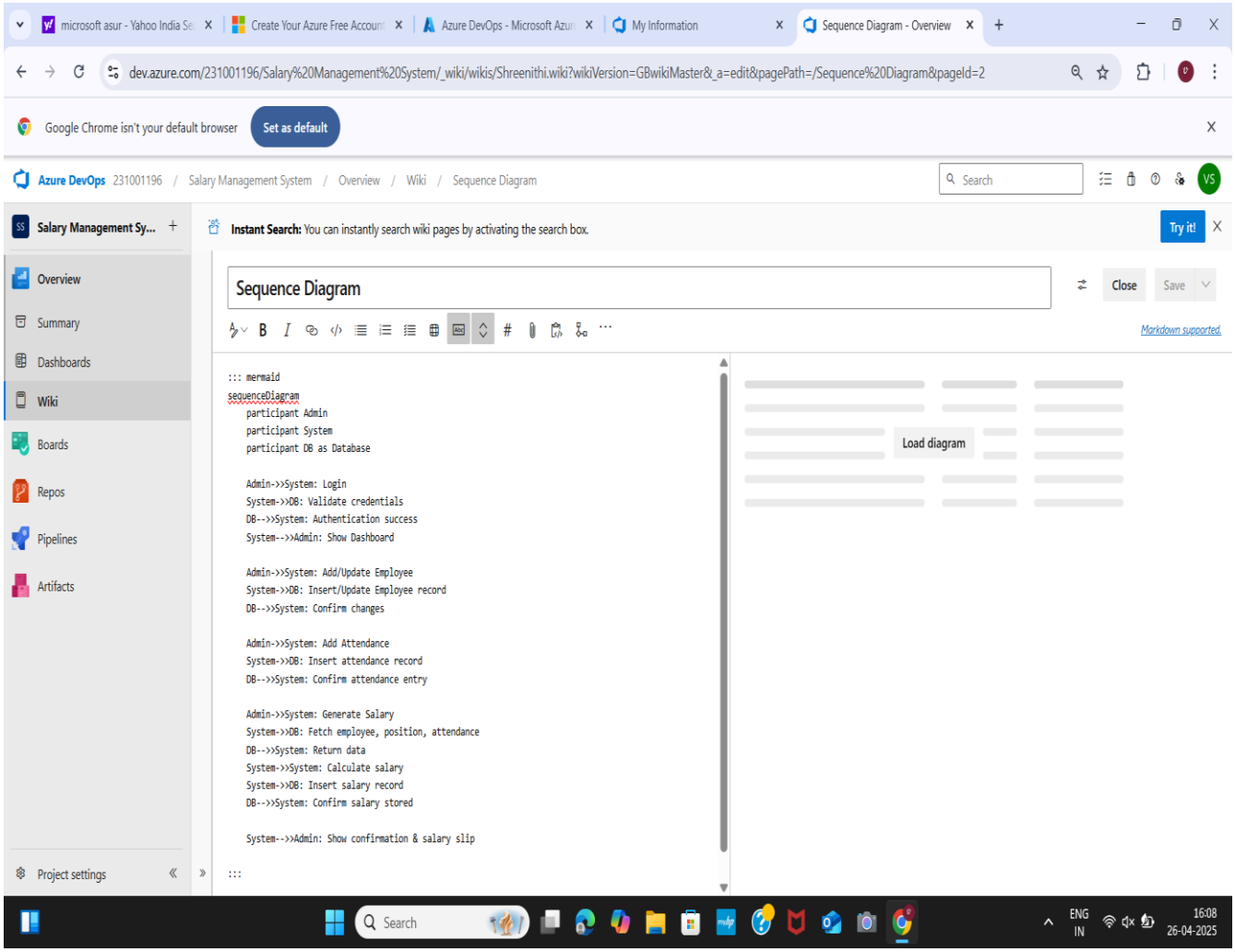
System->>System represents an internal process inside the same participant (self-message).

+ after ->> activates a participant (not used here, but useful if needed).

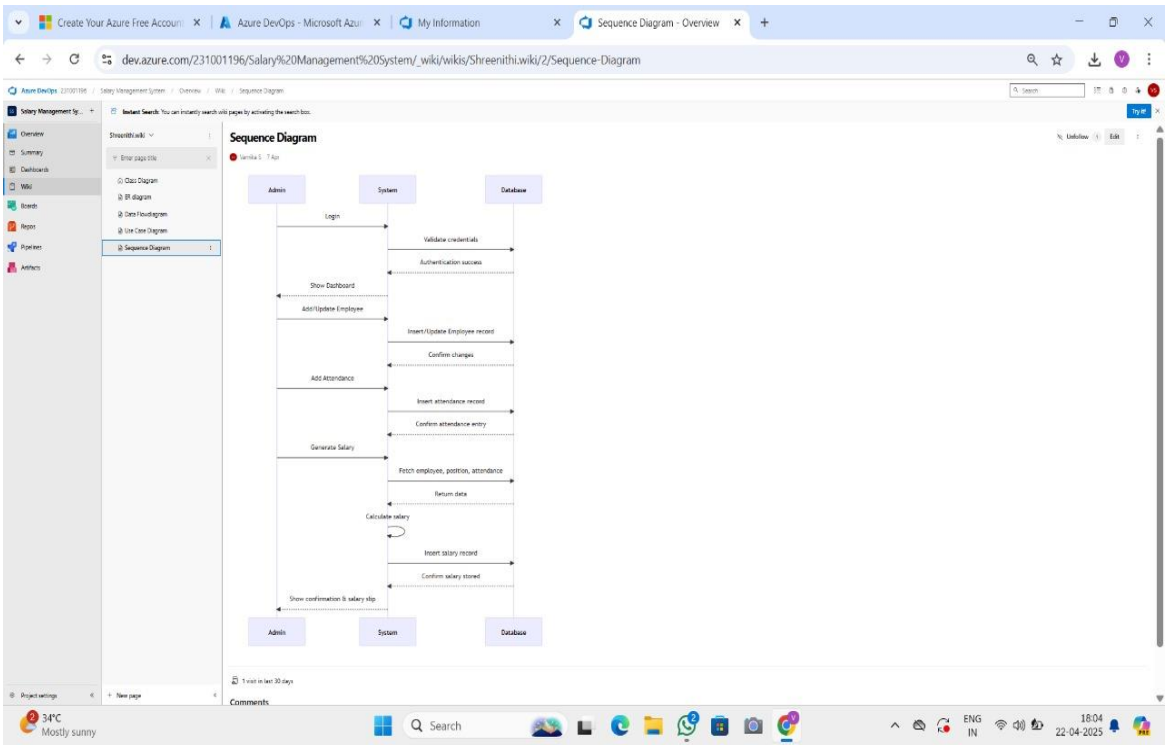
- after -->> deactivates a participant (not used here, but useful if needed).

alt / else is used for conditional branches (not present here, but can be added for "if-else" flows).

loop can be used for repeated actions or iterations (not used here).


Sequence Diagram - Overview
dev.azure.com/231001196/Salary%20Management%20System/_wiki/wikis/Shreenithi.wiki?wikiVersion=GBwikiMaster&_a=edit&pagePath=/Sequence%20Diagram&pageId=2
Google Chrome isn't your default browser
Set as default
Azure DevOps 231001196 / Salary Management System / Overview / Wiki / Sequence Diagram
Search
Salary Management Sy...
Instant Search: You can instantly search wiki pages by activating the search box.
Try it!
Overview
Summary
Dashboards
Wiki
Boards
Repos
Pipelines
Artifacts
Project settings
Sequence Diagram
Close
Save
Markdown supported.
::: mermaid
sequenceDiagram
participant Admin
participant System
participant DB as Database
Admin->>System: Login
System->>DB: Validate credentials
DB-->>System: Authentication success
System-->>Admin: Show Dashboard
Admin->>System: Add/Update Employee
System->>DB: Insert/Update Employee record
DB-->>System: Confirm changes
Admin->>System: Add Attendance
System->>DB: Insert attendance record
DB-->>System: Confirm attendance entry
Admin->>System: Generate Salary
System->>DB: Fetch employee, position, attendance
DB-->>System: Return data
System->>System: Calculate salary
System->>DB: Insert salary record
DB-->>System: Confirm salary stored
System-->>Admin: Show confirmation & salary slip
:::
Load diagram
Search
ENG
IN
16:08
26-04-2025

5. click wiki menu and select the page



Result:

The sequence diagram was drawn successfully.

EX NO. 6

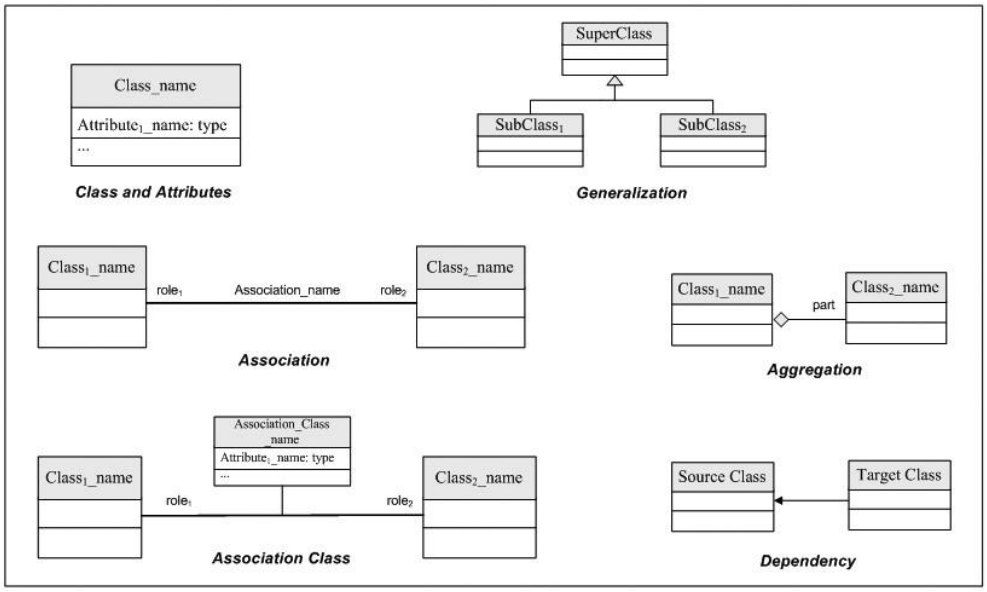
CLASS DIAGRAM

AIM :-

To draw a sample class diagram for your project or system.

THEORY

A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.

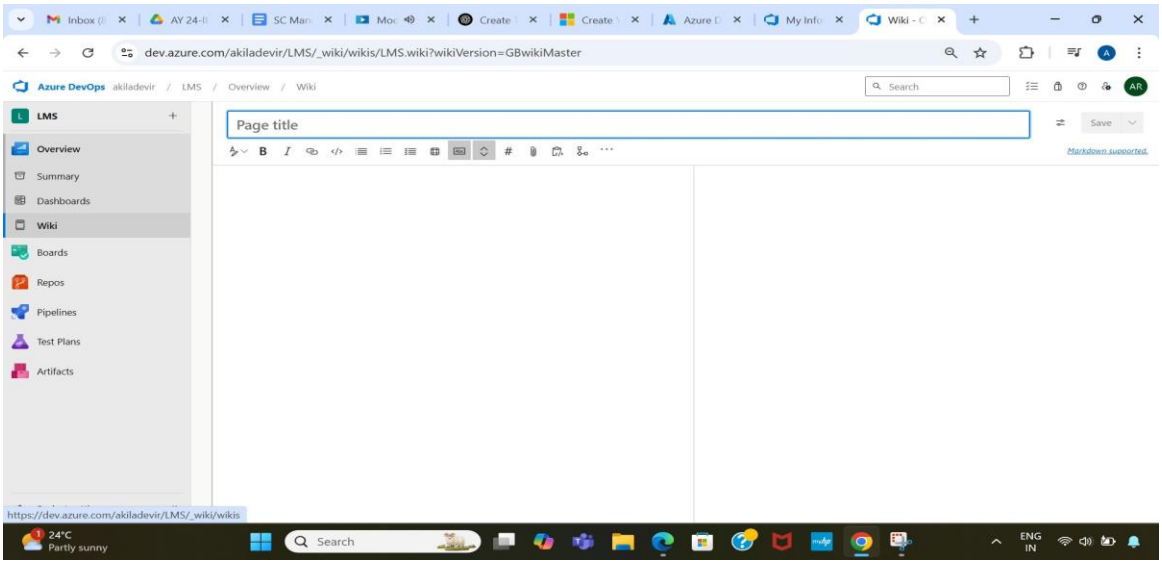


Notations in class diagram

Procedure:

1.Open a project in Azure DevOps Organisations.

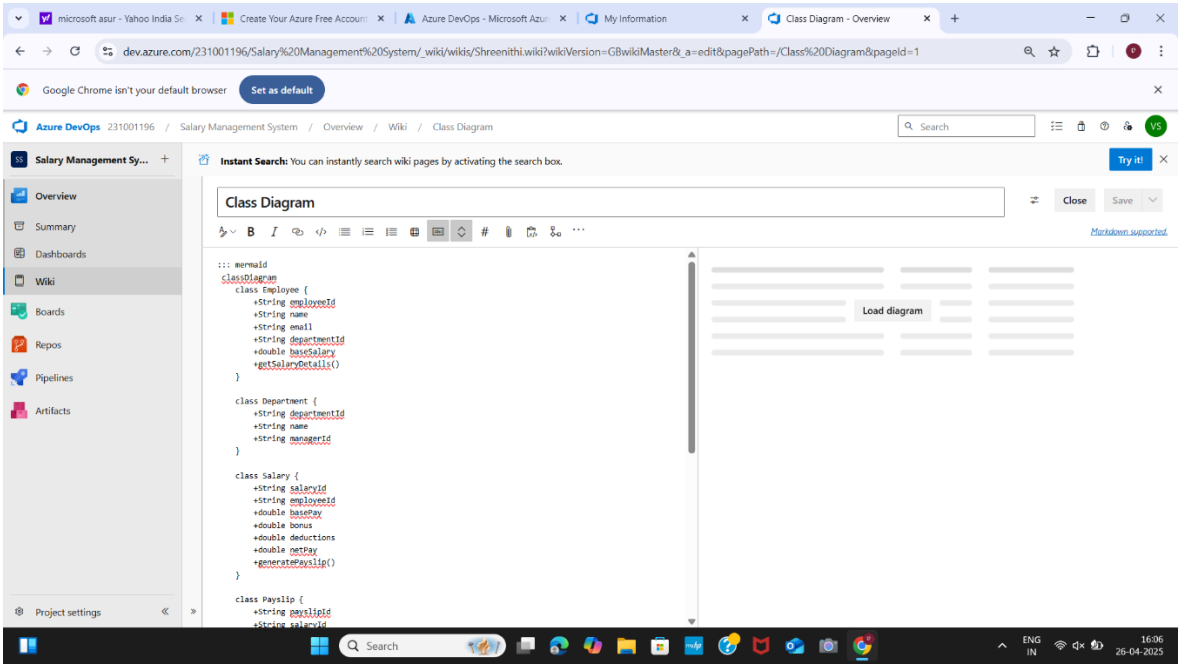
2..To design select wiki from menu

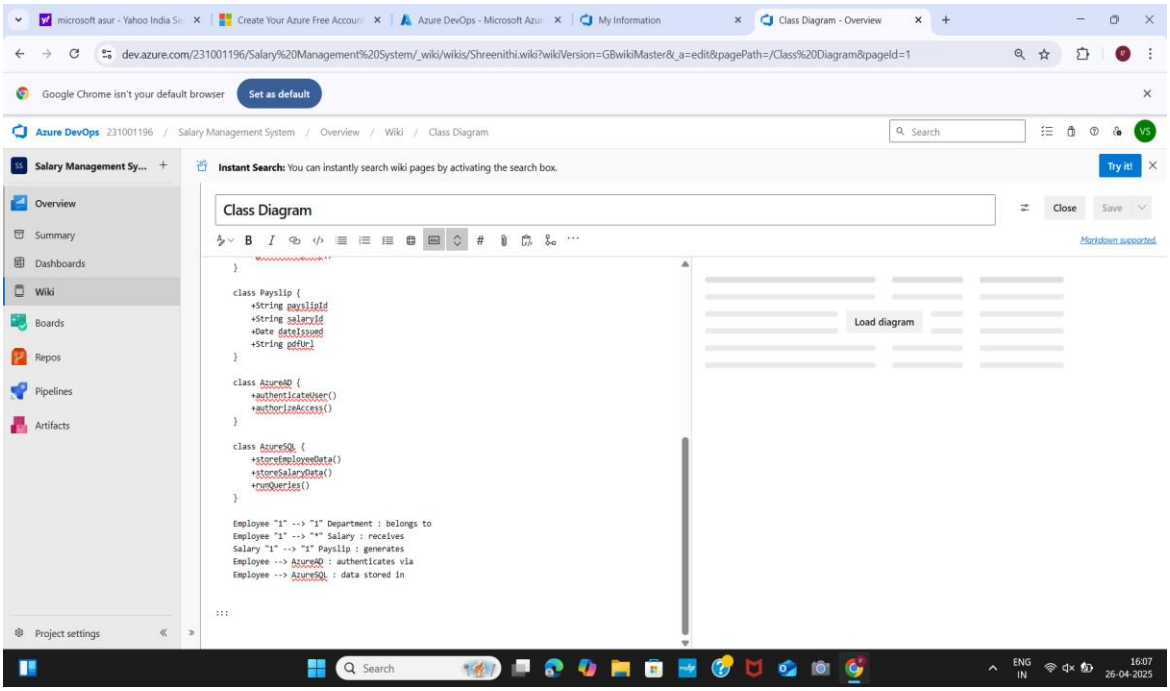


3.Write code for drawing class diagram and save the code

```
::: mermaid
classDiagram
  class Employee {
    +String employeeId
    +String name
    +String email
    +String departmentId
    +double baseSalary
    +getSalaryDetails()
  }

  class Department {
    +String departmentId
    +String name
    +String managerId
  }

  class Salary {
    +String salaryId
    +String employeeId
    +double basePay
    +double bonus
    +double deductions
    +double netPay
    +generatePayslip()
  }

  class Payslip {
    +String payslipId
    +String salaryId
```

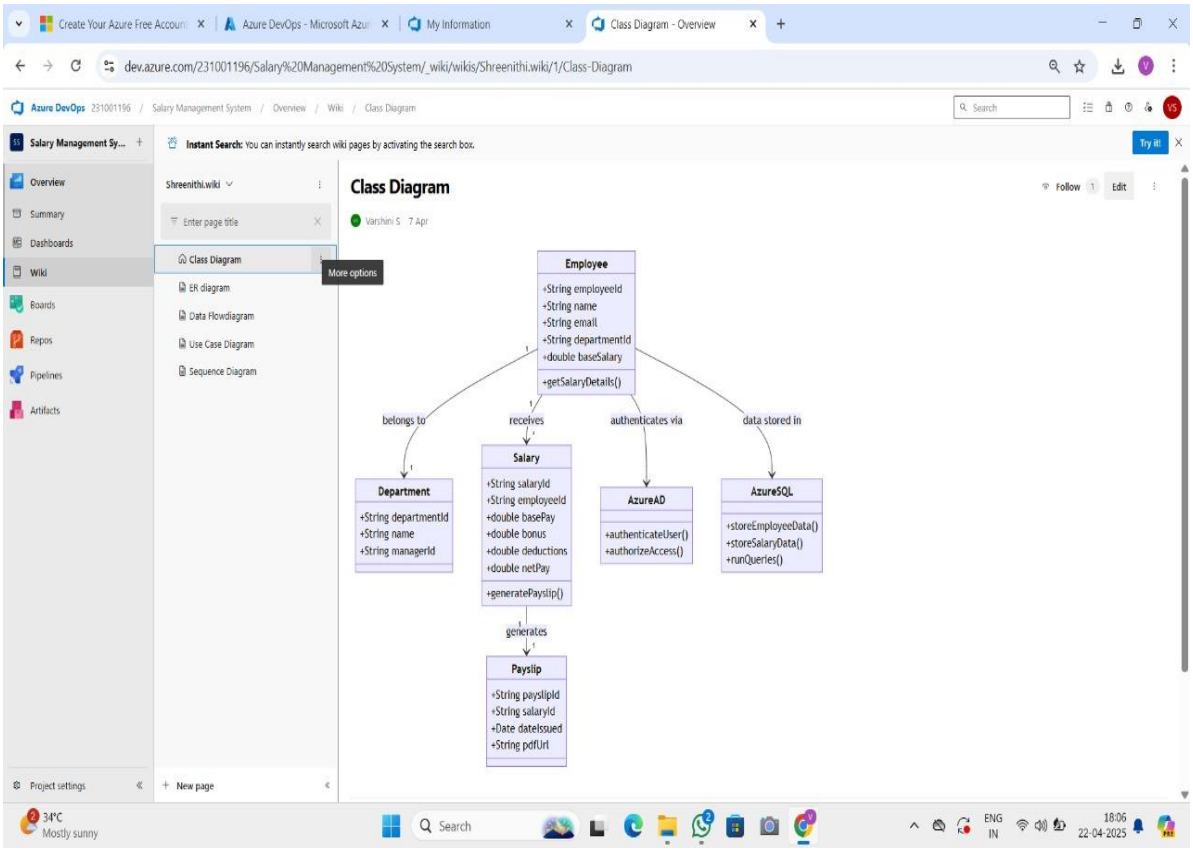
```
    +Date dateIssued
    +String pdfUrl
  }

  class AzureAD {
    +authenticateUser()
    +authorizeAccess()
  }

  class AzureSQL {
    +storeEmployeeData()
    +storeSalaryData()
    +runQueries()
  }

  Employee "1" --> "1" Department : belongs to
  Employee "1" --> "*" Salary : receives
  Salary "1" --> "1" Payslip : generates
  Employee --> AzureAD : authenticates via
  Employee --> AzureSQL : data stored in
:::
```





Relationship Types

Type	Description
<|	Inheritance
*	Composition
o	Aggregation
>	Association
<	Association
|>	Realization



Result:

The class diagram was designed successfully.

EX NO:7

USECASE DIAGRAM

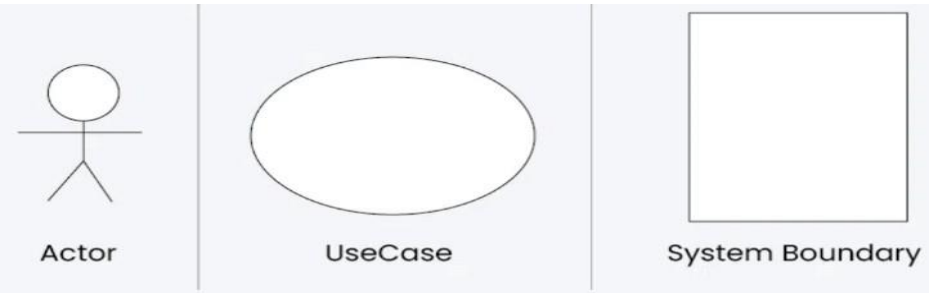
Aim:

Steps to draw the Use Case Diagram using draw.io

Theory:

UCD shows the relationships among actors and use cases within a system which Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model and communicate the scope of a development project

- **Use Cases**
- **Actors**
- **Relationships**
- **System Boundary Boxes**



Procedure

Step 1: Create the Use Case Diagram in Draw.io

- Open Draw.io (diagrams.net).
- Click "Create New Diagram" and select "Blank" or "UML Use Case" template.
- Add Actors (Users, Admins, External Systems) from the UML section.
- Add Use Cases (Functionalities) using ellipses.
- Connect Actors to Use Cases with lines (solid for direct interaction, dashed for <<include>> and <<extend>>).
- Save the diagram as .drawio or export as PNG/JPG/SVG.

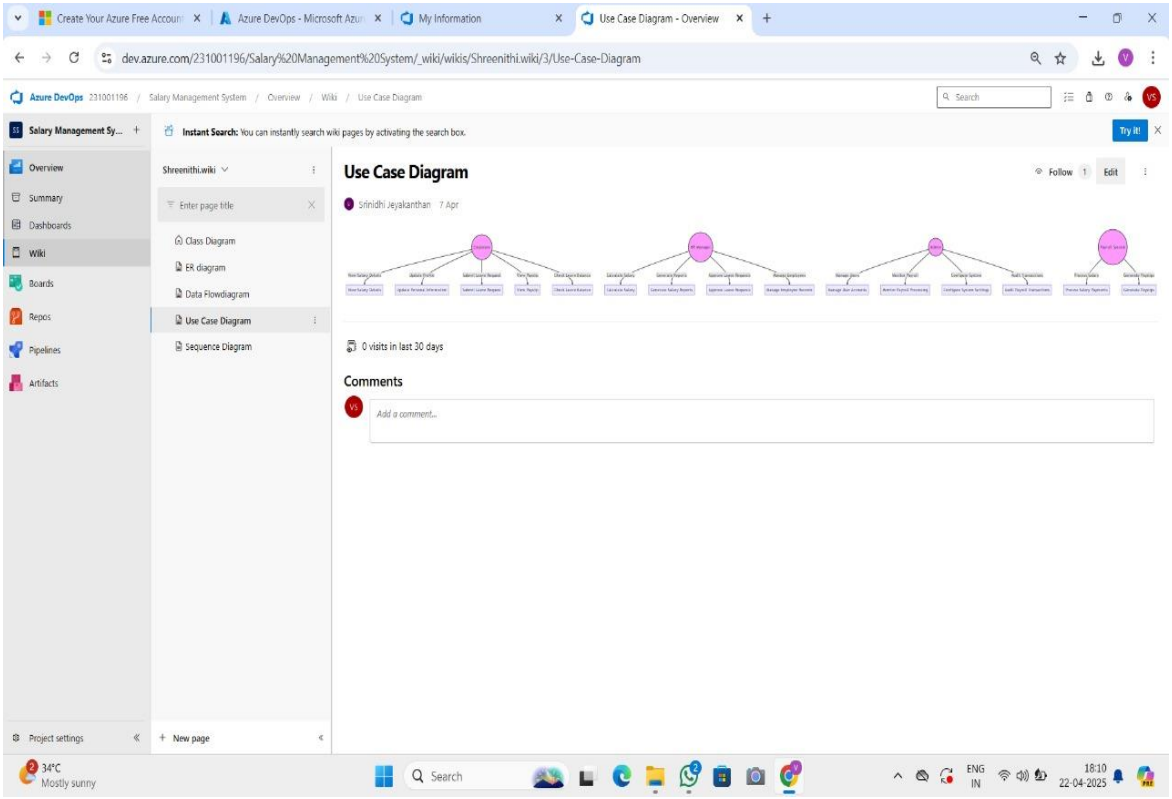
Step 2: Upload the Diagram to Azure DevOps

Option 1: Add to Azure DevOps Wiki

- Open Azure DevOps and go to your project.
- Navigate to Wiki (Project > Wiki).
- Click "Edit Page" or create a new page.
- Drag & Drop the exported PNG/JPG image.
- Use Markdown to embed the diagram:
- ![Use Case Diagram](attachments/use_case_diagram.png)

Option 2: Attach to Work Items in Azure Boards

- Open Azure DevOps → Navigate to Boards (Project > Boards).
- Select a User Story, Task, or Feature.
- Click "Attachments" → Upload your Use Case Diagram.
- Add comments or descriptions to explain the use case.



Result:

The use case diagram was designed successfully

EX NO. 8

ARCHITECTURE DIAGRAM

Aim:

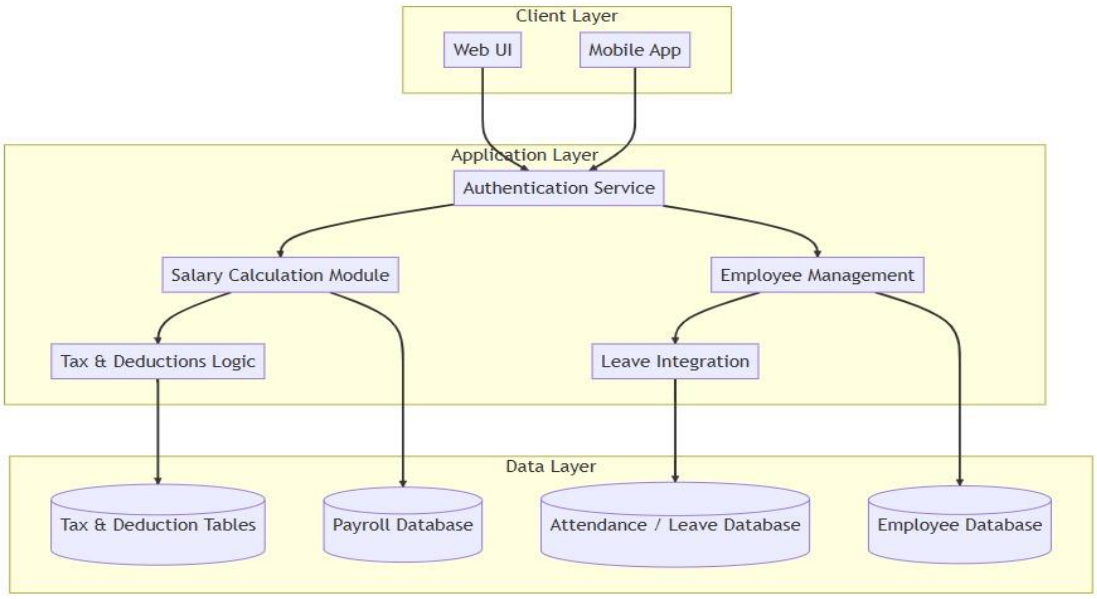
Steps to draw the Architecture Diagram using draw.io.

Theory:

An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.

Procedure

1.Draw diagram in draw.io
2.Upload the diagram in Azure DevOps wiki



Result:

The architecture diagram was designed successfully

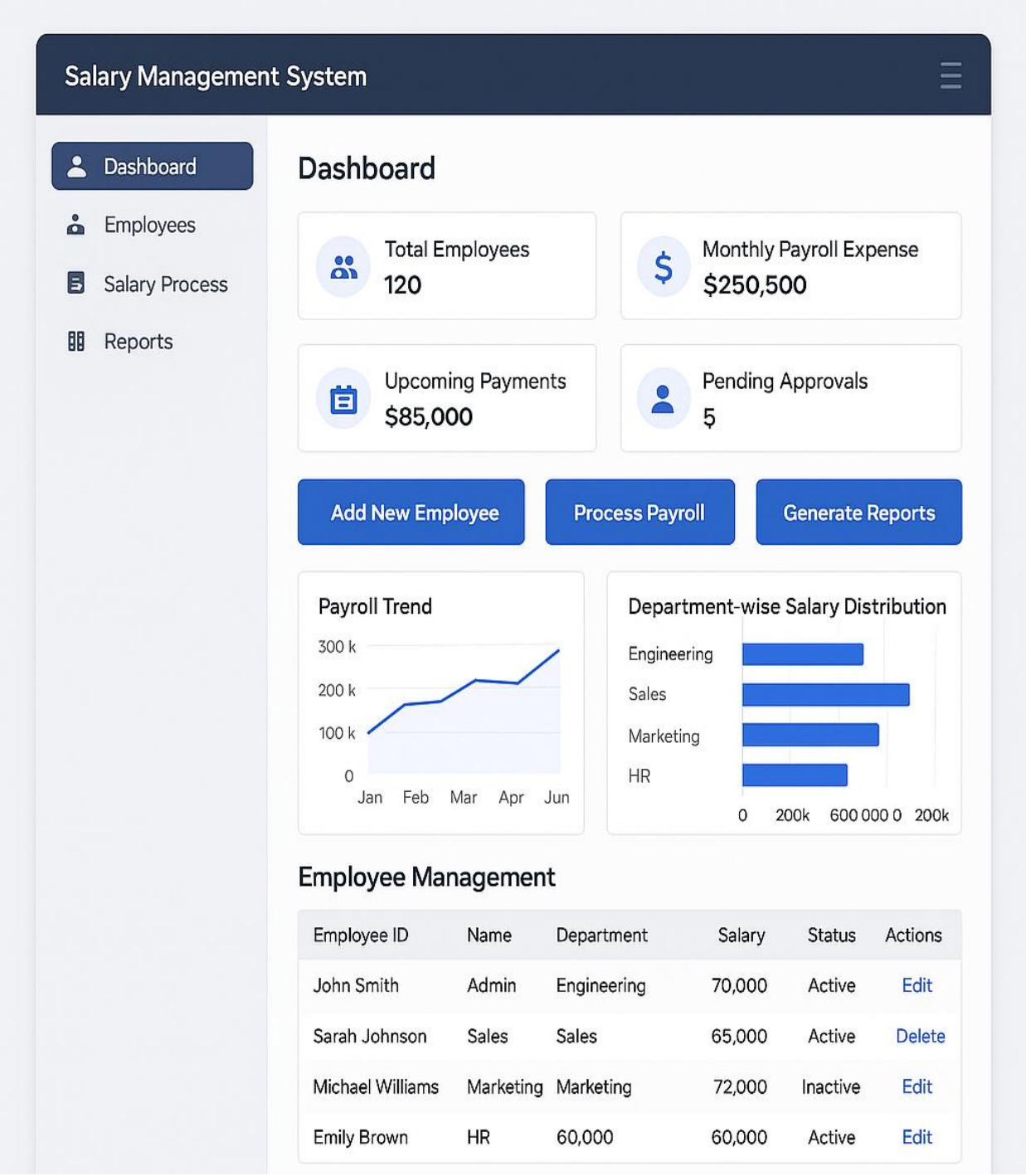
EX NO. 9

USER INTERFACE

Aim:

Design User Interface for the given project

User Interface for Salary Management System:



Result:

The UI was designed successfully.

EX NO. 10

IMPLEMENTATION

Aim:

To implement the given project based on Agile Methodology.

Procedure:

Step 1: Set Up an Azure DevOps Project

- Log in to Azure DevOps.
- Click "New Project" → Enter project name → Click "Create".
- Inside the project, navigate to "Repos" to store the code.

Step 2: Add Your Web Application Code

- Navigate to Repos → Click "Clone" to get the Git URL. ● Open Visual Studio Code / Terminal and run: git clone <repo_url> cd <repo_folder>

- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.). ● Commit & push:

```
git add .
git commit -m "Initial commit" git push origin main
```

Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)

- Navigate to Pipelines → Click "New Pipeline".
- Select Git Repository (Azure Repos, GitHub, or Bitbucket).
- Choose Starter Pipeline or a pre-configured template for your framework.
- Modify the azure-pipelines.yml file (Example for a Node.js app):

```
pool:
  vmImage: 'ubuntu-latest'

steps:
- task: UseNode@1
inputs:
    version: '16.x'

- script: npm install
```

```
displayName:
'Install
dependencies'

- script:
npm run build
displayName:
'Build
application'

- task:
PublishBuildArt
ifacts@1
inputs:

pathToPublish:
'dist'
artifactName:
'drop'
```

Click "Save and Run" → The pipeline will start building app.

Step 4: Set Up Release Pipeline (CD - Continuous Deployment)

- Go to Releases → Click "New Release Pipeline".
- Select Azure App Service or Virtual Machines (VMs) for deployment.
- Add an artifact (from the build pipeline).
- Configure deployment stages (Dev, QA, Production).
- Click "Deploy" to push your web app to Azure.

Result

Thus the application was successfully implemented.

EXP NO : 11

IMPLEMENTATION – DESIGN A WEBPAGE BASED ON SCRUM METHODOLOGY

AIM :

To implement the given project based on Agile Methodology.

PROCEDURE:

Step 1: Set Up an Azure DevOps Project

- Log in to Azure DevOps.
- Click "New Project" → Enter project name → Click "Create".
- Inside the project, navigate to "Repos" to store the code.

Step 2: Add Your Web Application Code

- Navigate to Repos → Click "Clone" to get the Git URL.
- Open Visual Studio Code / Terminal and run:

```
git clone <repo_url>
cd <repo_folder>
```

- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.).
- Commit & push:

```
git add .
git commit -m "Initial commit" git push origin main
```

Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)

- Navigate to Pipelines → Click "New Pipeline".
- Select Git Repository (Azure Repos, GitHub, or Bitbucket).
- Choose Starter Pipeline or a pre-configured template for your framework.
- Modify the azure-pipelines.yml file (Example for a Node.js app)

```
trigger:
- main

pool:
vmImage: 'ubuntu-latest'
steps:
- task: UseNode@1 inputs:
version: '16.x'

- script: npm install
displayName: 'Install dependencies'

-script: npm run build displayName: 'Build application'

- task: PublishBuildArtifacts@1 inputs:
pathToPublish: 'dist' artifactName: 'drop'
```

Click "Save and Run" → The pipeline will start building app.

Step 4: Set Up Release Pipeline (CD - Continuous Deployment)

- Go to Releases → Click "New Release Pipeline".
- Select Azure App Service or Virtual Machines (VMs) for deployment.

- Add an artifact (from the build pipeline).
- Configure deployment stages (Dev, QA, Production).
- Click "Deploy" to push your web app to Azure.

RESULT :

Thus the application was successfully implemented.

EXP NO : 12

TESTING – TEST PLAN, TEST CASE, LOAD TESTING

AIM:

To design and manage structured test plans and test cases in Azure DevOps for validating core user
stories through both happy path and error scenarios and evaluate the performance of the application's endpoint by creating and executing load tests using Azure Load Testing.

PROCEDURE:

TEST CASE DESIGN PROCEDURE

1. Understand Core Features of the Application

- Review requirement documents and user stories.
- Identify all main functionalities of the application.
- Ensure complete coverage of modules before test case creation.

2. Define User Interactions

- Determine common user behaviors based on application flow.
- Translate user actions into testable scenarios.
- Ensure each test case mimics a real user operation.

3. Design Happy Path Test Cases

- Create test cases for expected and correct user actions.
- Ensure each functionality works under normal conditions.
- Add these cases under the relevant Test Suite in Azure DevOps.

4. Design Error Path Test Cases

- Identify edge cases, invalid inputs, and system failures.
- Test how the system handles incorrect or unexpected behavior.
- Add these test cases to the same or a separate Test Suite in Azure DevOps.

5. Break Down Steps and Expected Results

- Write step-by-step instructions in the "Steps" section of the test case.
- Provide expected results for each action.
- Ensure clarity for both manual execution and automation mapping.

6. Use Clear Naming and IDs

- Name test cases clearly using a defined naming convention (e.g., TC01, TC02, etc.).
- Ensure titles reflect the purpose of the test case.
- Azure DevOps auto-generates test case IDs for tracking.

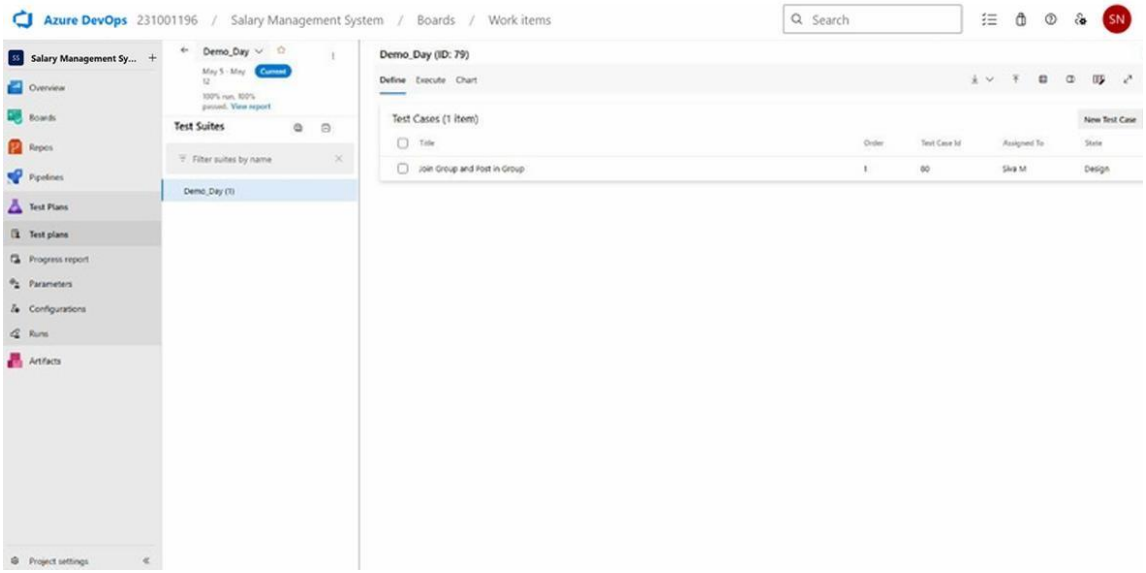
7. Separate Test Suites

- Group test cases based on functionality (e.g., Login, Playlist, Recommendations).
- Use Static, Requirement-based, or Query-based suites in Azure DevOps.
- Improves traceability and execution flow.

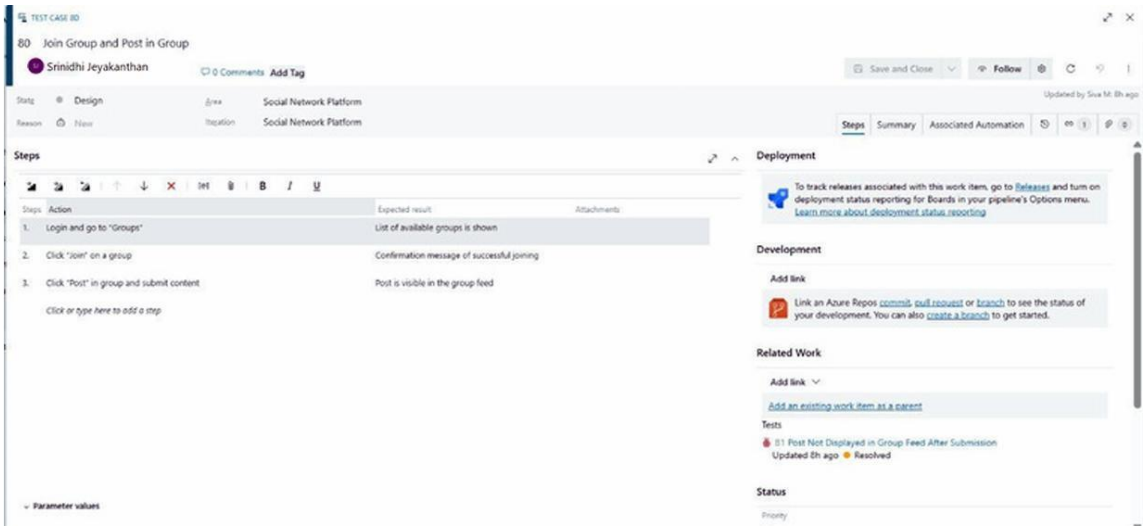
8. Prioritize and Review

- Mark test cases with priority (High, Medium, Low).
- Review test cases for completeness and correctness.
- Ensure alignment with associated user stories or features.

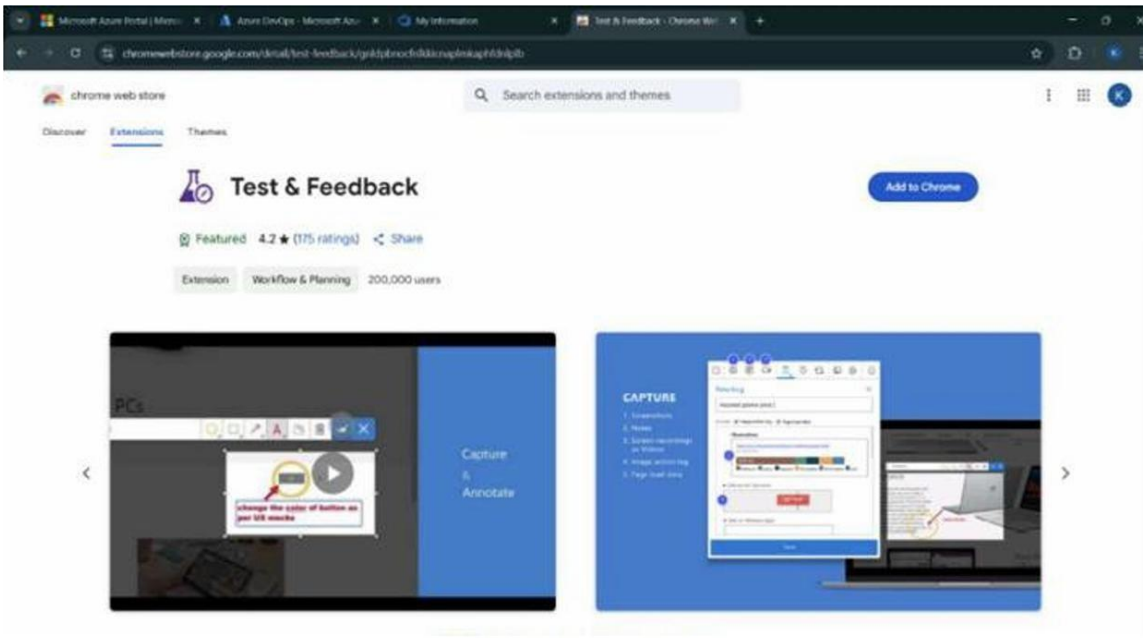
1. **New test plan**



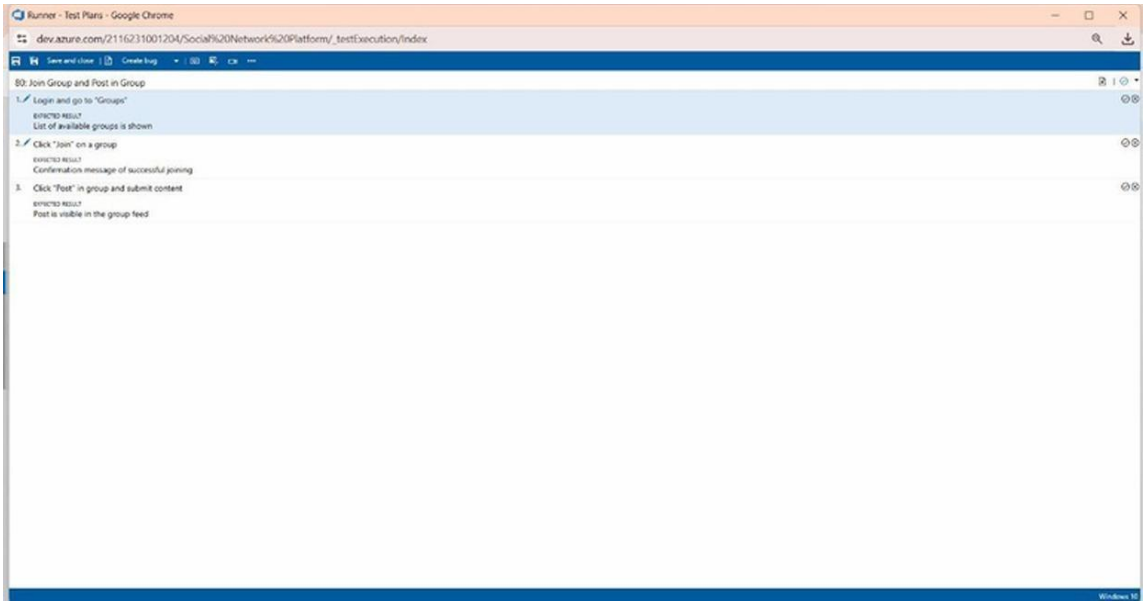
2.Test case



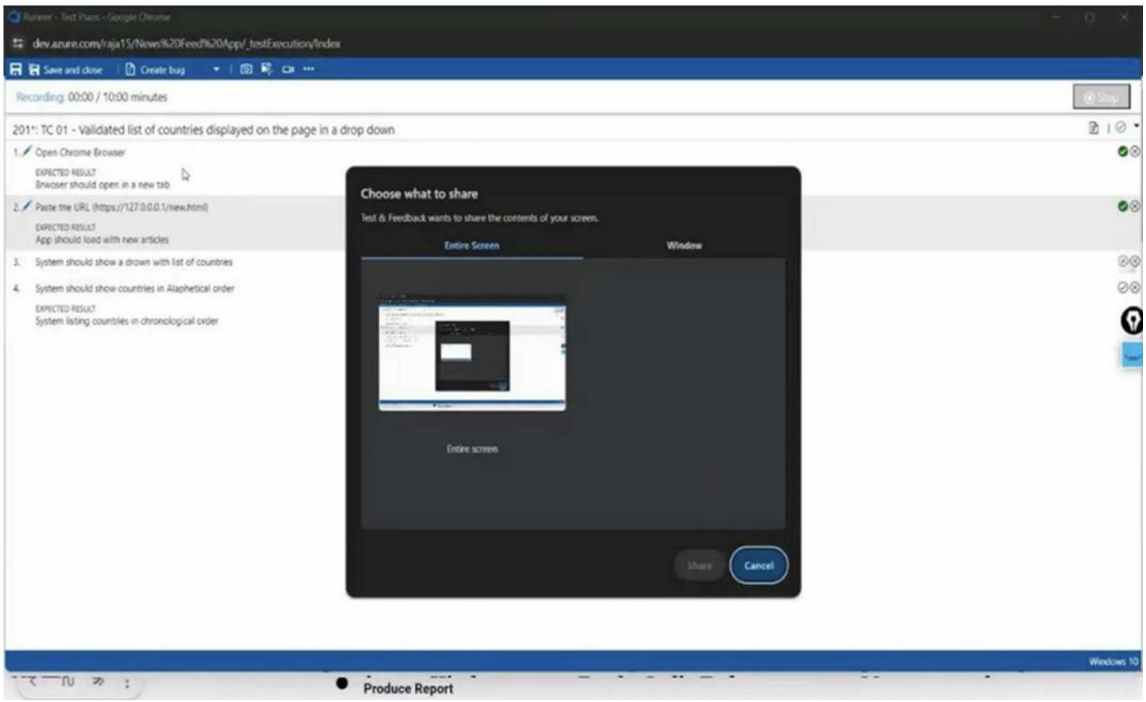
3.Installation of Test



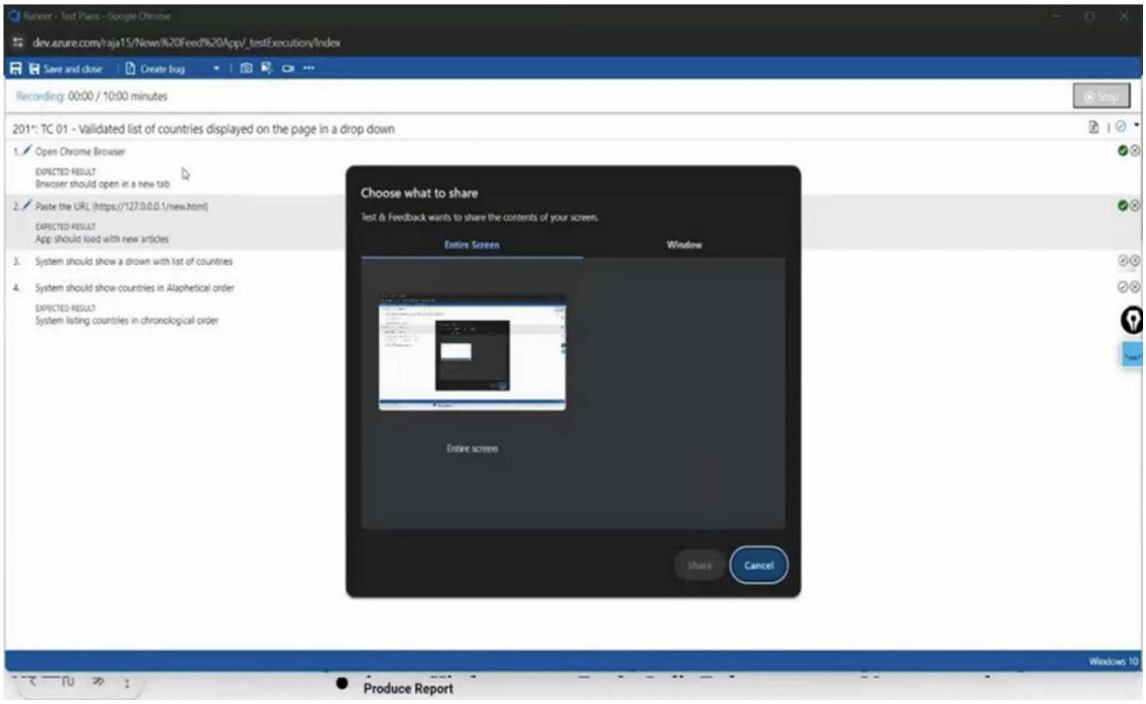
4.Running the Test Cases



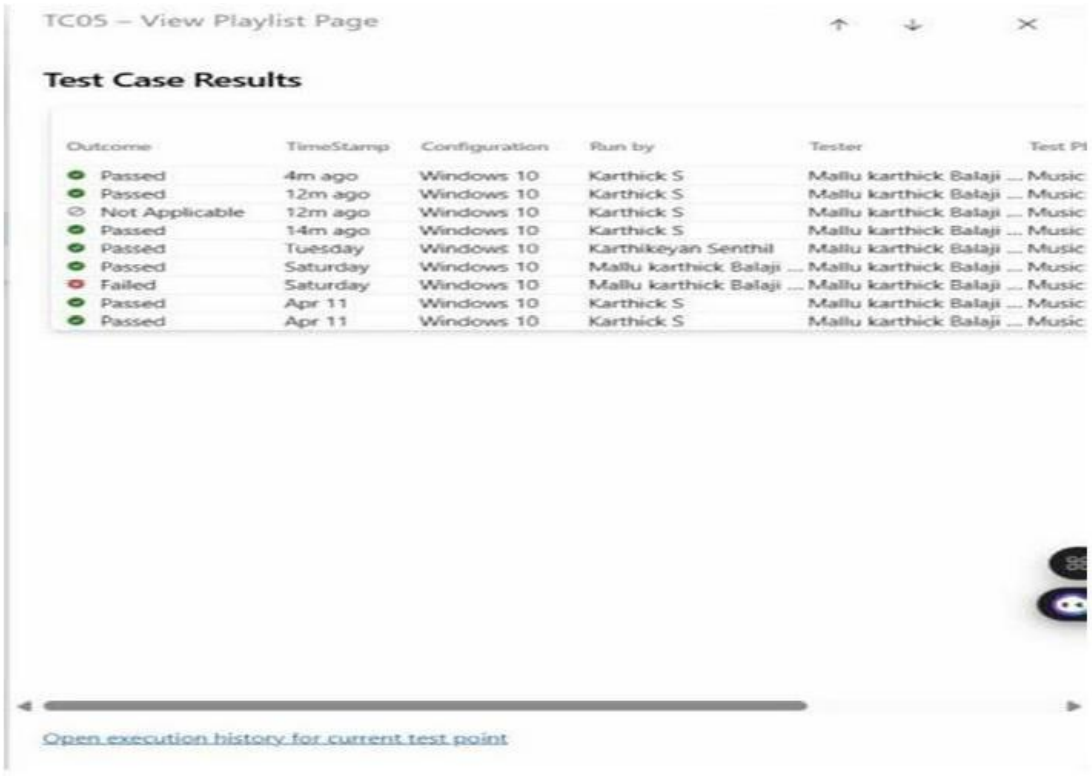
5.Recording the Test Cases



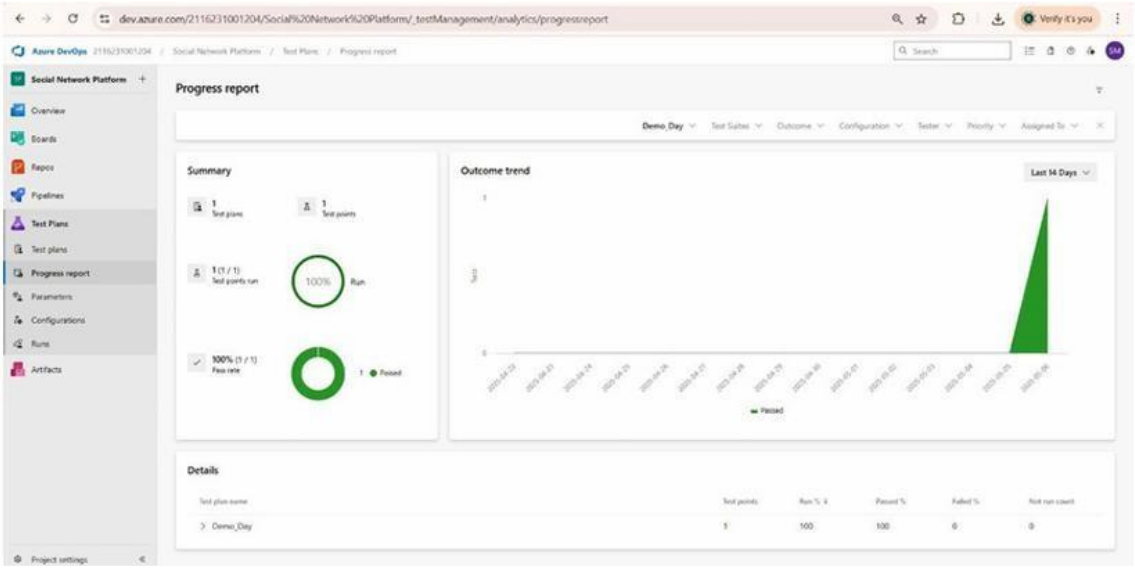
6.Creating Bugs



7.Test Case Results



8.Progress Report



LOAD TESTING PROCEDURE :

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 and log in.

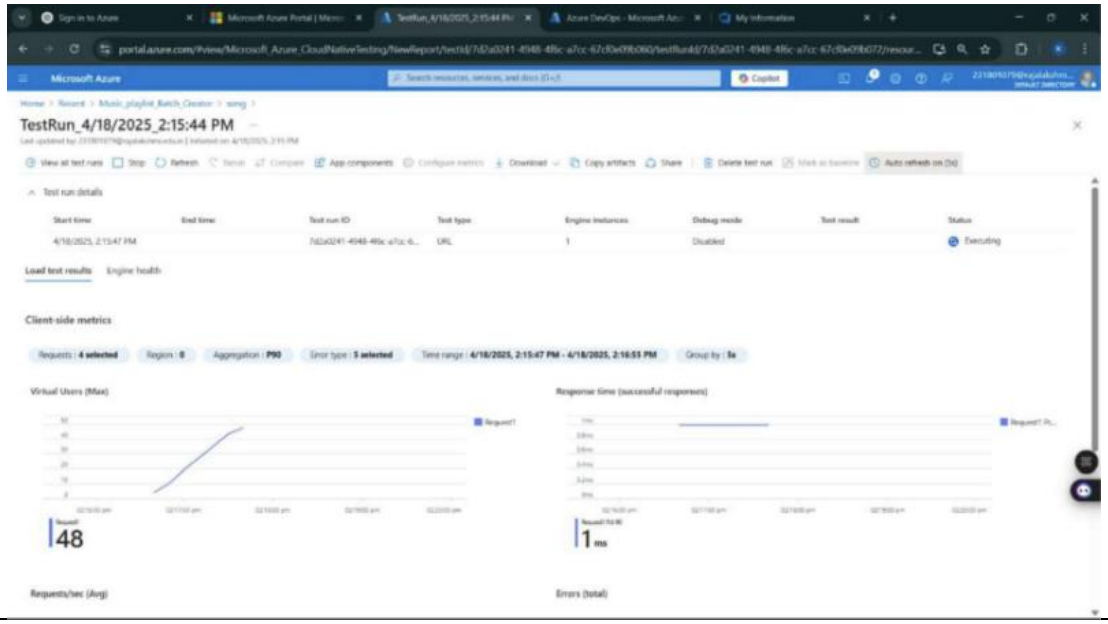
2. Create the Resource
    - Go to Create a resource — Search for “Azure Load Testing”.
    - Select Azure Load Testing and click Create.
3. Fill in the Configuration Details
    - Subscription: Choose your Azure subscription.
    - Resource Group: Create new or select an existing one.
    - Name: Provide a unique name (no special characters).
    - Location: Choose the region for hosting the resource.
4. (Optional) Configure tags for categorization and billing.
5. Click Review + Create, then Create.
6. Once deployment is complete, click Go to resource.

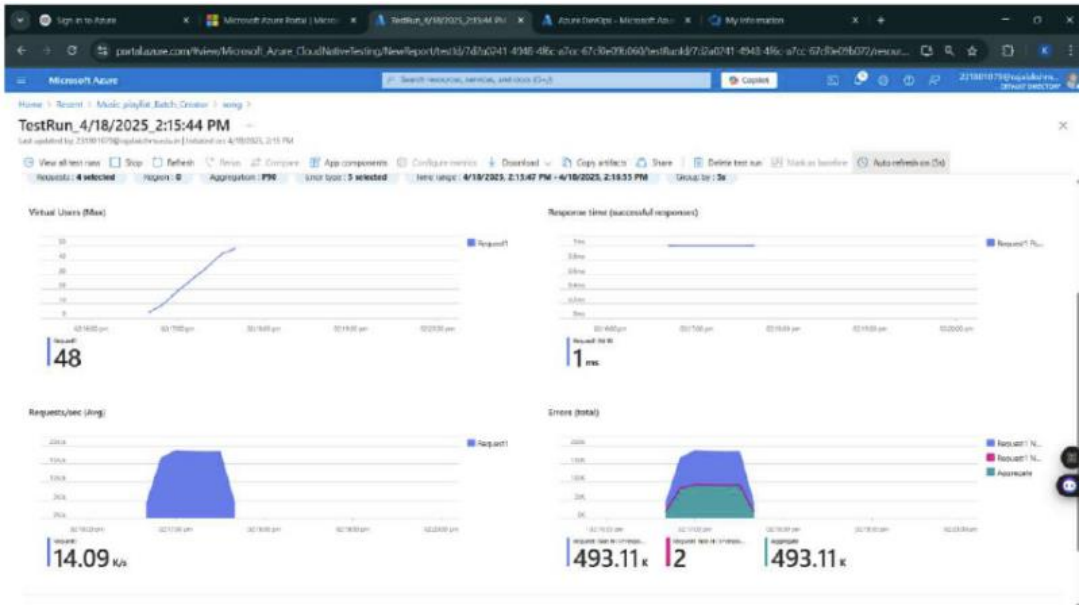
Steps to Create and Run a Load Test:

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





RESULT:

Test plans and test cases for selected user stories were created in Azure DevOps, covering both happy and error paths and an Azure Load Testing resource was also set up, and a load test was successfully run to evaluate the performance of the target en