

# **RAJALAKSHMI ENGINEERING COLLEGE**

**RAJALAKSHMI NAGAR, THANDALAM - 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CS23432**

**SOFTWARE CONSTRUCTION**

**Laboratory Record Note Book**

Name : LOGESWARAN .....

Year / Branch / Section : ..... II/B.Tech-IT-'AD'

Register No. : ..... 231001101

Semester : ..... IV

Academic Year : ..... 2024-2025



**RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)**  
**RAJALAKSHMI NAGAR, THANDALAM- 602 105**

**BONAFIDE CERTIFICATE**

NAME : LOGESWARAN K      REGISTER NO: 231001101

ACADEMIC YEAR 2024-25 **SEMESTER- IV** **BRANCH:** B. Tech Information

Technology [AD/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

**LAB PLAN**  
**CS23432-SOFTWARE CONSTRUCTION LAB**

<b>Ex No</b>	<b>Date</b>	<b>Topic</b>	<b>Page No</b>	<b>Sign</b>
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		

## **Course Outcomes (COs)**

**Course Name: Software Engineering**

**Course Code: CS23432**

<b>CO 1</b>	Understand the software development process models.
<b>CO 2</b>	Determine the requirements to develop software
<b>CO 3</b>	Apply modeling and modeling languages to design software products
<b>CO 4</b>	Apply various testing techniques and to build a robust software products
<b>CO 5</b>	Manage Software Projects and to understand advanced engineering concepts

### **CO - PO – PSO matrices of course**

<b>PO/PSO CO</b>	<b>PO1</b>	<b>PO2</b>	<b>PO3</b>	<b>PO4</b>	<b>PO5</b>	<b>PO6</b>	<b>PO7</b>	<b>PO8</b>	<b>PO9</b>	<b>PO10</b>	<b>PO11</b>	<b>PO12</b>	<b>PSO1</b>	<b>PSO2</b>	<b>PSO3</b>
<b>CS23432.1</b>	2	2	3	2	2	2	2	2	2	2	3	2	1	3	-
<b>CS23432.2</b>	2	3	1	2	2	1	-	1	1	1	2	-	1	2	-
<b>CS23432.3</b>	2	2	1	1	1	1	1	1	1	1	1	1	2	2	1
<b>CS23432.4</b>	2	2	3	2	2	2	1	0	2	2	2	1	1	2	1
<b>CS23432.5</b>	2	2	2	1	1	1	1	0	2	1	1	1	2	1	-
<b>Average</b>	<b>2.0</b>	<b>2.2</b>	<b>2.0</b>	<b>1.6</b>	<b>1.6</b>	<b>1.4</b>	<b>1.3</b>	<b>1.3</b>	<b>1.6</b>	<b>1.4</b>	<b>1.8</b>	<b>1.3</b>	<b>1.4</b>	<b>2.0</b>	<b>1.0</b>

Correlation levels 1, 2 or 3 are as defined below:

1: Slight (Low)    2: Moderate (Medium)    3: Substantial (High)    No correlation: “-“

## **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:**

**E-commerce Uploader:**

In the rapidly evolving world of digital commerce, e-commerce platforms are witnessing a massive influx of products across various categories to meet the dynamic demands of consumers. Sellers, ranging from small businesses to large enterprises, are required to upload and manage thousands of product listings regularly. However, the traditional manual method of uploading product data — including titles, descriptions, prices, categories, images, inventory details, and specifications — is often tedious, error-prone, and time-consuming.

Many sellers struggle with inconsistencies, formatting errors, missing information, and redundant work while uploading or updating their product catalogs. These issues not only hinder operational efficiency but also affect the customer experience due to inaccurate or incomplete product data. A slow and inefficient product listing process can delay the time-to-market, impacting the overall business performance.

To overcome these challenges, there is a need for a smart and scalable E-commerce Product Uploader Tool that simplifies and automates the product listing workflow. This tool should support features such as bulk uploads via CSV/Excel files, real-time data validation, image preview and compression, auto-category detection, and error highlighting — thereby ensuring faster, more accurate, and hassle-free uploads.

By implementing such a system, sellers can manage their product inventory more efficiently, minimize manual errors, and focus more on their core business strategies. The tool also ensures that end-users receive accurate and complete product information, leading to improved customer satisfaction, reduced returns, and increased trust in the platform.

Overall, this project aims to build a reliable and user-friendly product uploader tool that not only optimizes the seller's workflow but also contributes to the success and scalability of e-commerce platforms.

**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

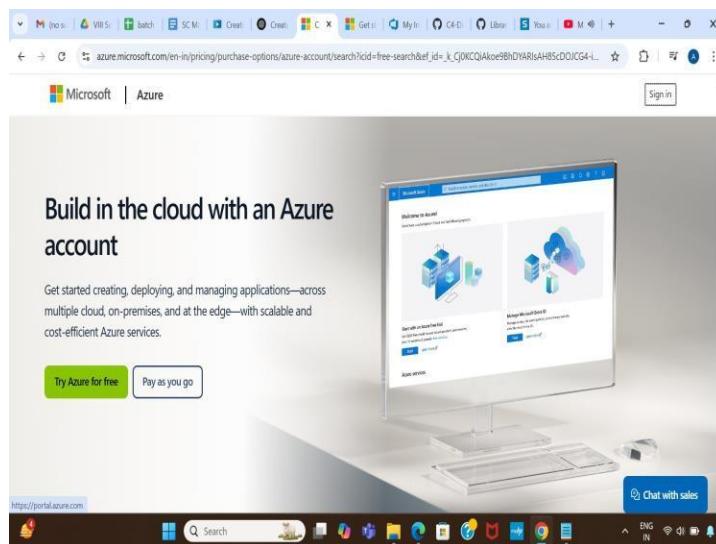
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

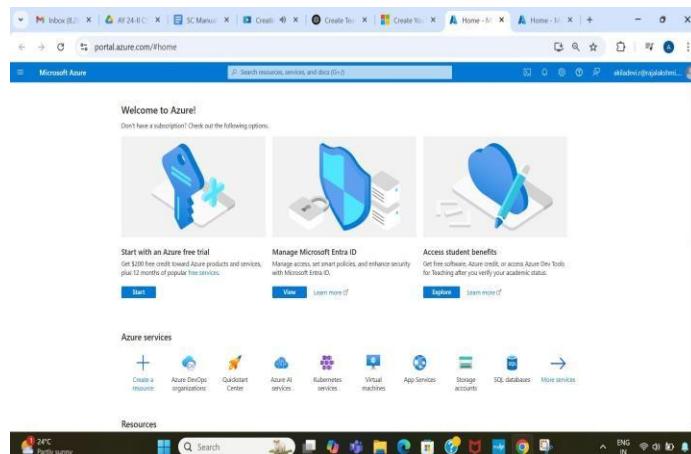
"As a [role], I [want to], [so that]."

#### Procedure:

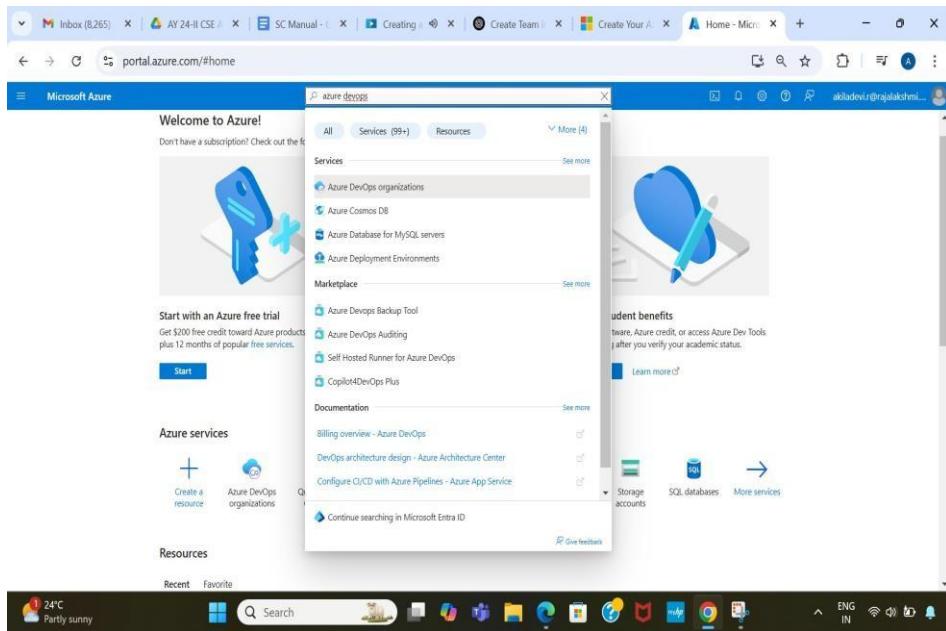
1. Open your web browser and go to the Azure website:  
<https://azure.microsoft.com/en-in> Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for  
<https://signup.live.com/?lic=1>



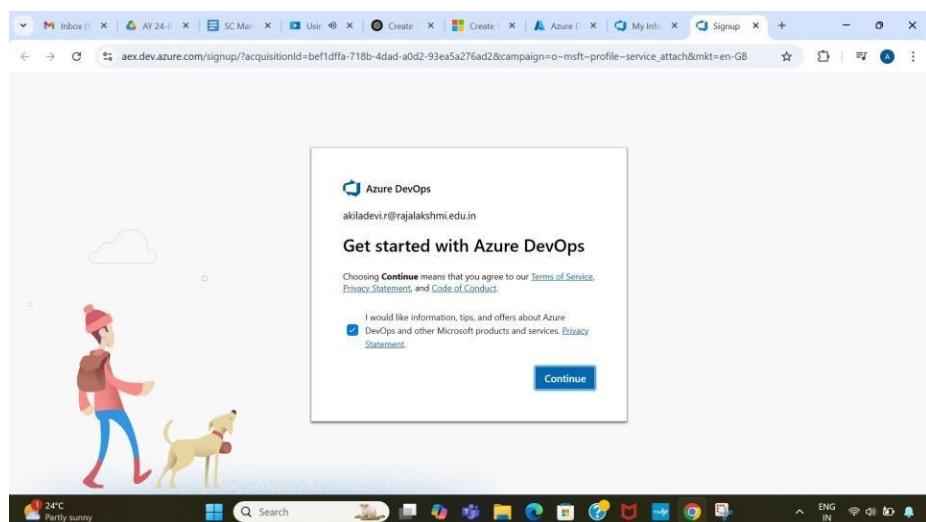
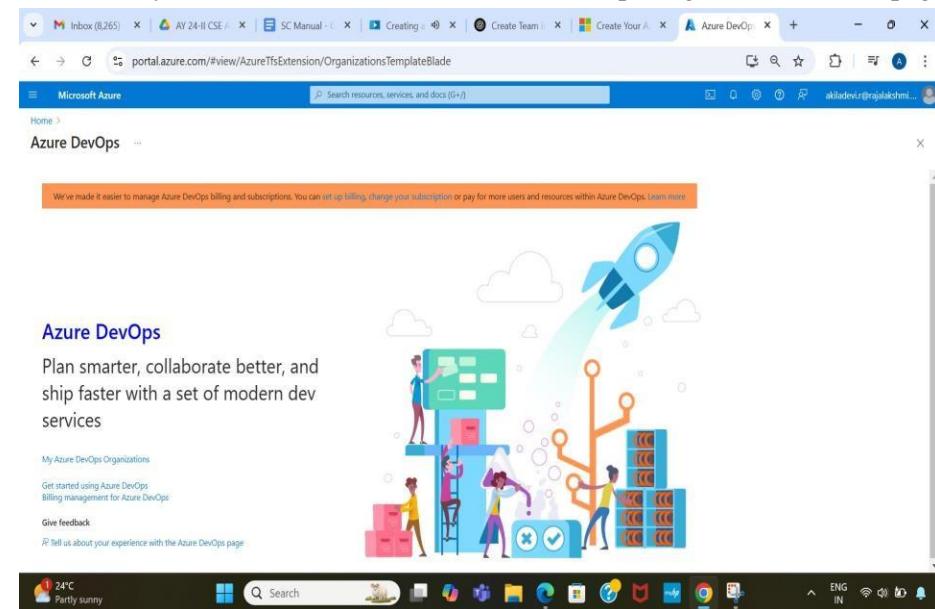
3. Azure home page

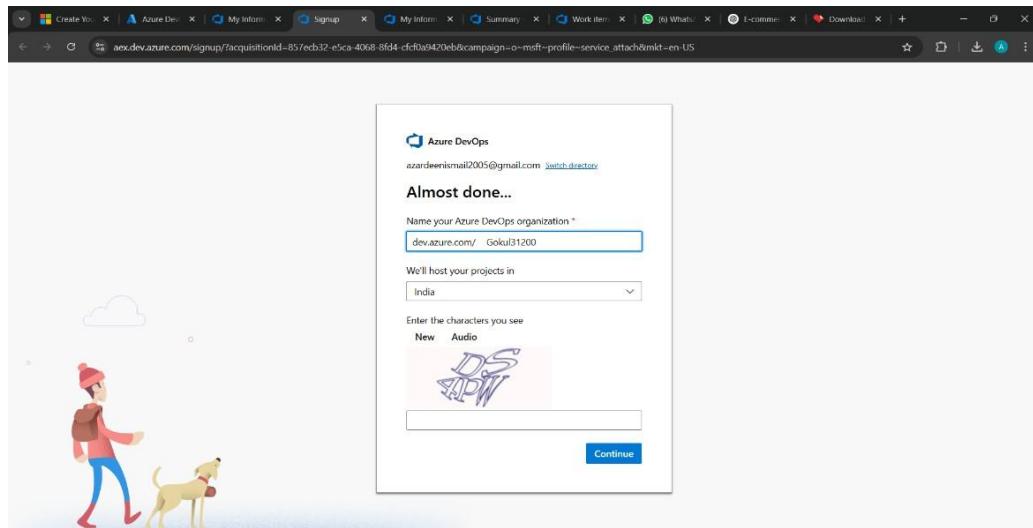


4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.



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.

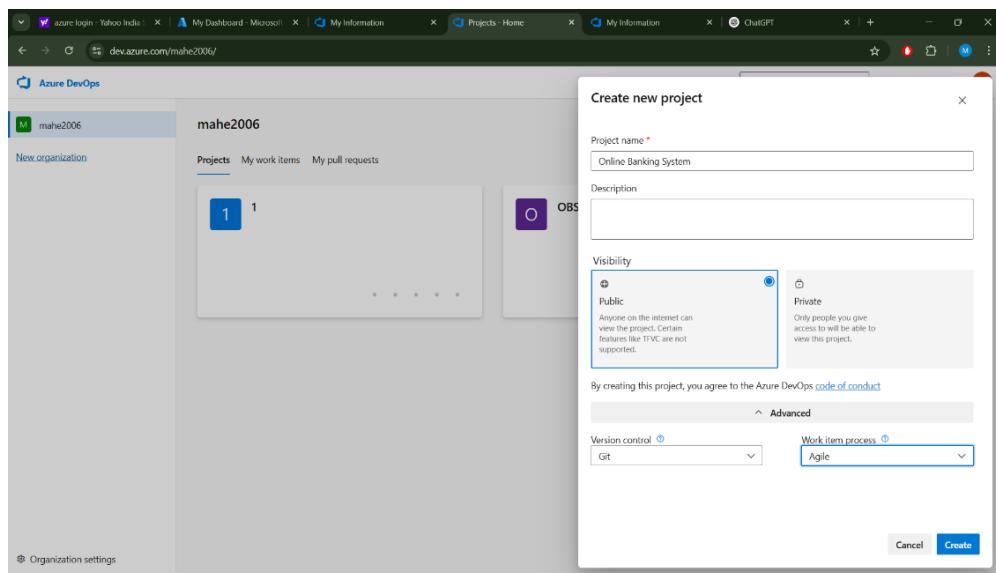




## 6. Create the First Project in Your Organization

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.

- i. On the organization's **Home page**, click on the **New Project** button.
- ii. Enter the project name, description, and visibility options:
  - o **Name:** Choose a name for the project (e.g., **LMS**).
  - o **Description:** Optionally, add a description to provide more context about the project.
  - o **Visibility:** Choose whether you want the project to be **Private** (accessible only to those invited) or **Public** (accessible to anyone).
- iii. Once you've filled out the details, click **Create** to set up your first project.



7. 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.

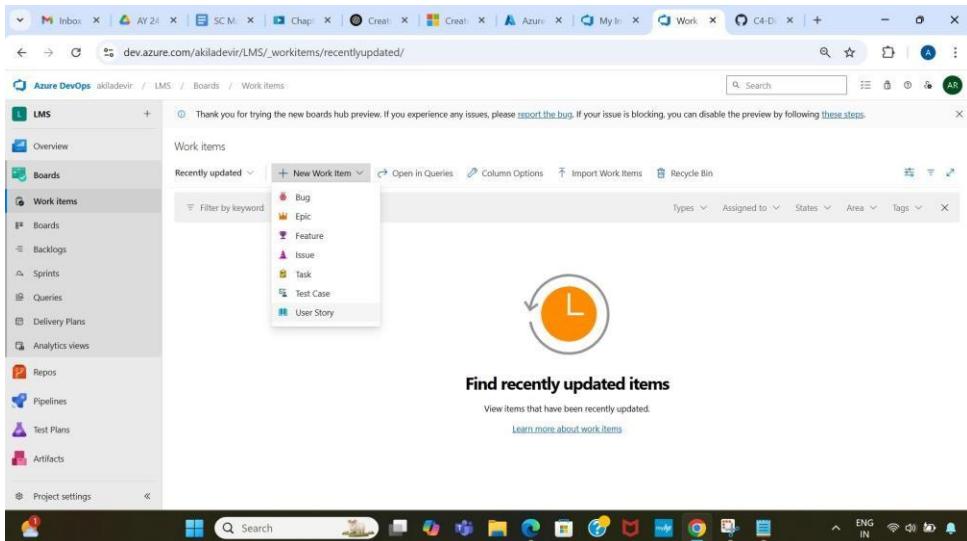
The screenshot shows the Azure DevOps Organizations dashboard. At the top, there's a profile picture of Mahesh Babu with initials 'MB' and his name. The top navigation bar includes 'Microsoft', 'Mahesh Babu', and 'Sign out'. Below the profile, Mahesh's details are listed: 'Mahesh Babu', 'Edit profile', 'maheshbabu2006@gmail.com', 'Microsoft account', 'India', and his email 'maheshbabu2006@gmail.com'. A 'Visual Studio Dev Essentials' section is present with a link to get everything needed for building and deploying apps. On the right, the 'Azure DevOps Organizations' header has a 'Create new organization' button. Under 'Projects', there are two entries: 'Online Banking System' and 'OBS', with a 'New project' option. An 'Actions' section contains a 'Open in Visual Studio' link.

## 8. Project dashboard

The screenshot shows the Azure DevOps project dashboard for 'Online Banking System'. The left sidebar has a navigation menu with options: Overview, Summary (which is selected), Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main content area is titled 'Online Banking System'. It features a 'About this project' section with a placeholder for 'Add Project Description' and a 'Help others to get on board!' section. To the right, there's a 'Project stats' card showing '0 Work items' and a 'Members' card listing 'MB' and 'AG'. The top navigation bar shows the URL 'dev.azure.com/mahe2006/OBS' and includes 'Search', 'Public', 'Invite', and a profile icon.

## 9. 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.



## 10. Fill in User Story Details

A screenshot of the Azure DevOps User Story details page for 'User Story 19 Transfer Funds Between Accounts and to External Banks'. The page has a header with tabs for Overview, Boards, Work items, and more. The main content area shows the user story title, description, acceptance criteria, planning details (Story Points: 2, Priority: 2, Risk: 1), classification (Value area: Business), deployment information, and development details (Add link). There are also sections for Comments, Add Tag, and a save button.

## Result:

The user story was written successfully.

## EX NO: 5

### SEQUENCE DIAGRAM

#### Aim:

To design a Sequence Diagram by using Mermaid.js

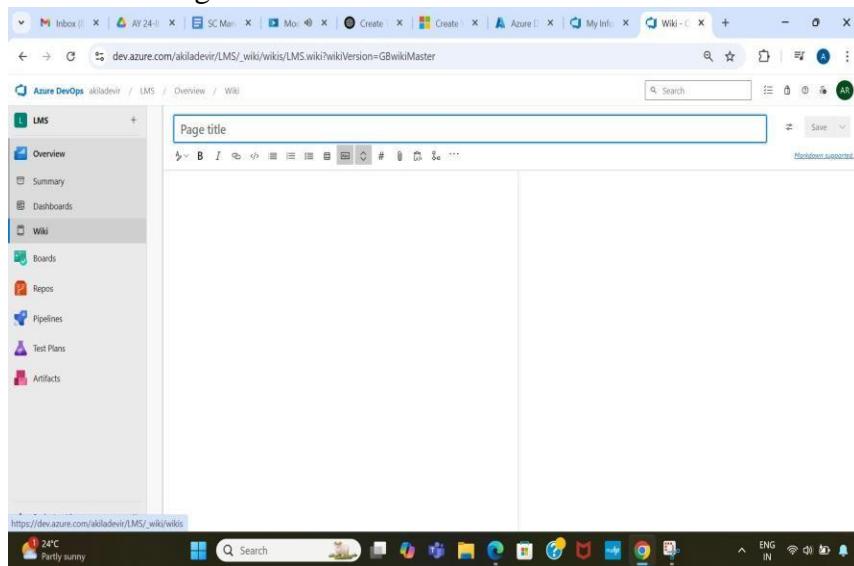
#### THEORY:

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 U as User
    participant S as System
    participant A as Account
    participant T as Transaction
```

U->>S: Login with username and password

S->>S: Validate user credentials

alt Success

S->>U: Display dashboard

else Failure

S->>U: Display error message

end

```

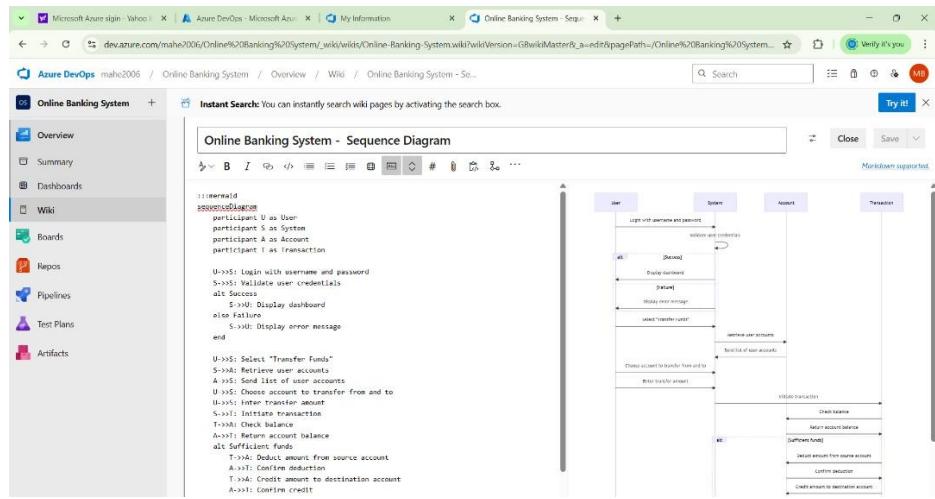
U->>S: Select "Transfer Funds"
S->>A: Retrieve user accounts
A->>S: Send list of user accounts
U->>S: Choose account to transfer from and to
U->>S: Enter transfer amount
S->>T: Initiate transaction
T->>A: Check balance
A->>T: Return account balance
alt Sufficient funds
    T->>A: Deduct amount from source account
    A->>T: Confirm deduction
    T->>A: Credit amount to destination account
    A->>T: Confirm credit
    T->>S: Display transfer success
else Insufficient funds
    T->>S: Display error message
end
U->>S: Log out
S->>U: Display log-out confirmation

```

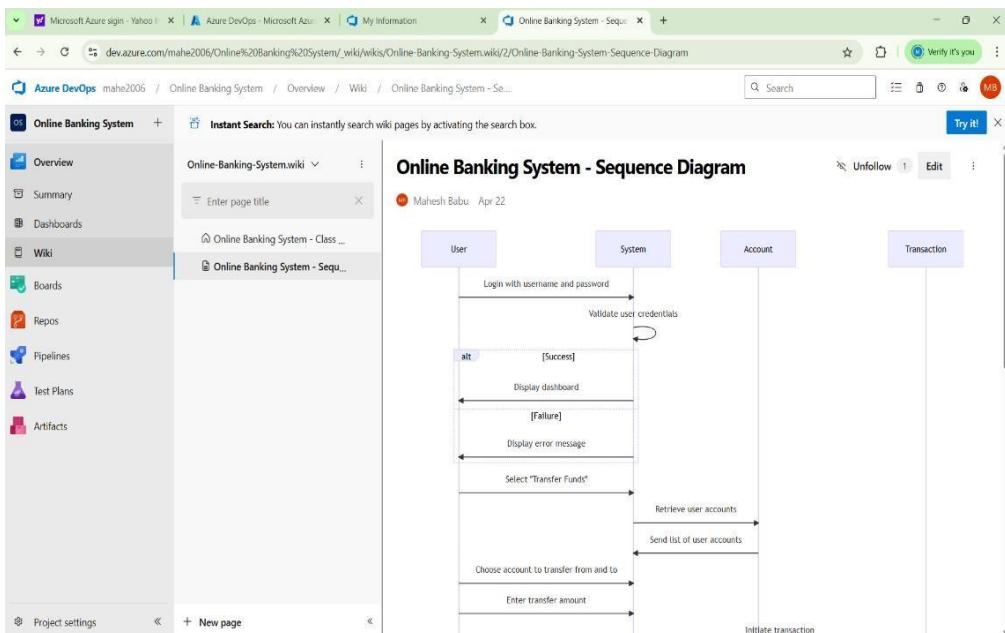
**Explanation:**

participant defines the entities involved.  
->> represents a direct message.  
-->> represents a response message.  
+ after ->> activates a participant  
- after -->> deactivates a participant. alt  
/ else for conditional flows loop can be used for repeated actions.

->	Solid line without arrow
-->	Dotted line without arrow
->>	Solid line with arrowhead
-->>	Dotted line with arrowhead
<<->>	Solid line with bidirectional arrowheads (v11.0.0+)
<<->>	Dotted line with bidirectional arrowheads (v11.0.0+)
-x	Solid line with a cross at the end
--x	Dotted line with a cross at the end
-)	Solid line with an open arrow at the end (async)
--)	Dotted line with an open arrow at the end (async)



4. click wiki menu and select the page



## Result:

The sequence diagram was drawn successfully.

## EX NO. 6

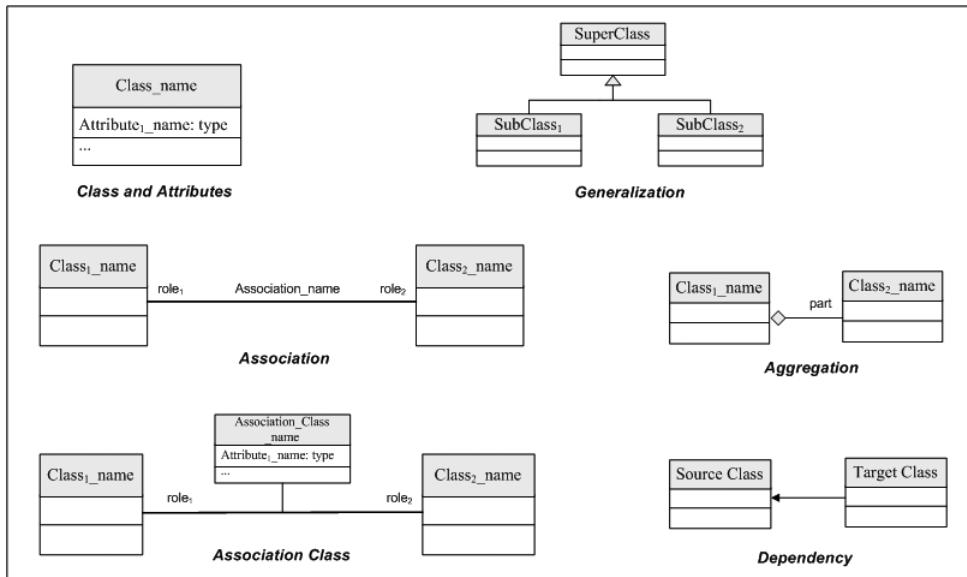
### CLASS DIAGRAM

#### 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.

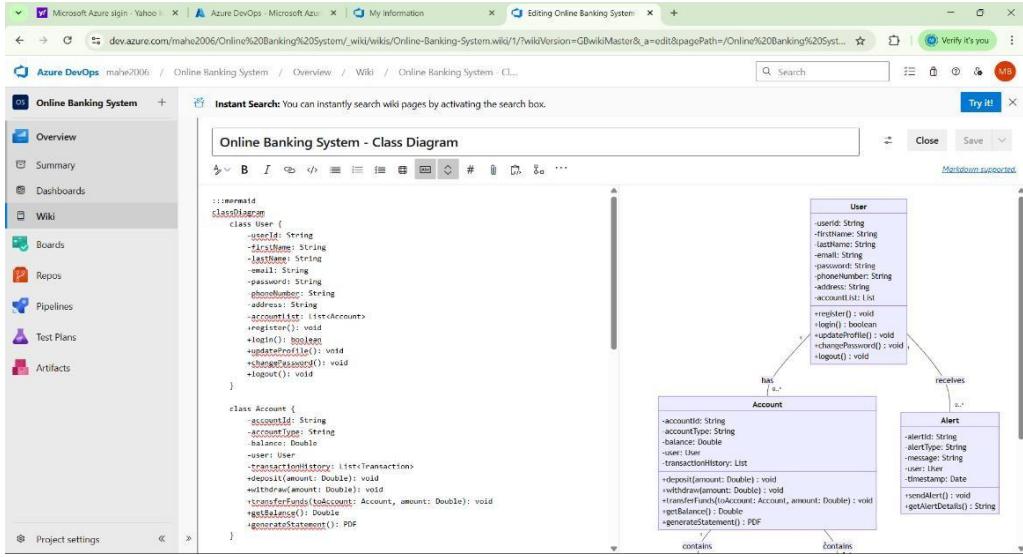


Notations in class diagram

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 User {
    -userId: String
    -firstName: String
    -lastName: String
    -email: String
    -password: String
    -address: String
    +register(): void
    +login(): boolean
    +updateProfile(): void
    +changePassword(): void
    +logout(): void
}
class Account {
    -accountId: String
    -accountType: String
    -balance: Double
    -user: User
    -transactionHistory: List<Transaction>
    +deposit(amount: Double): void
    +withdraw(amount: Double): void
    +transferFunds(toAccount: Account, amount: Double): void
    +getBalance(): Double
    +generateStatement(): PDF
}
class Alert {
    -alertId: String
    -alertType: String
    -message: String
    -user: User
    -timestamp: Date
    +sendAlert(): void
    +getAlertDetails(): String
}
User "1..>" Account : has
Alert "1..>" Account : receives
  
```

```
class Account {  
    -accountId: String  
    -accountType: String  
    -balance: Double  
    -user: User  
    -transactionHistory: List<Transaction>  
    +deposit(amount: Double): void  
    +withdraw(amount: Double): void  
    +transferFunds(toAccount: Account, amount: Double): void  
    +getBalance(): Double  
    +generateStatement(): PDF  
}
```

```
class Transaction {  
    -transactionId: String  
    -transactionType: String  
    -amount: Double  
    -transactionDate: Date  
    -status: String  
    -account: Account  
    +initiateTransaction(): boolean  
    +getTransactionDetails(): String  
}
```

```
class Payment {  
    -paymentId: String  
    -billerName: String  
    -dueDate: Date  
    -amount: Double  
    -user: User
```

```

    -status: String

    +payBill(): boolean

    +getPaymentDetails(): String

    +schedulePayment(): void

}

```

```

class Alert {

    -alertId: String

    -alertType: String

    -message: String

    -user: User

    -timestamp: Date

    +sendAlert(): void

    +getAlertDetails(): String

}

```

```

User "1" -- "0..*" Account : has

Account "1" -- "0..*" Transaction : contains

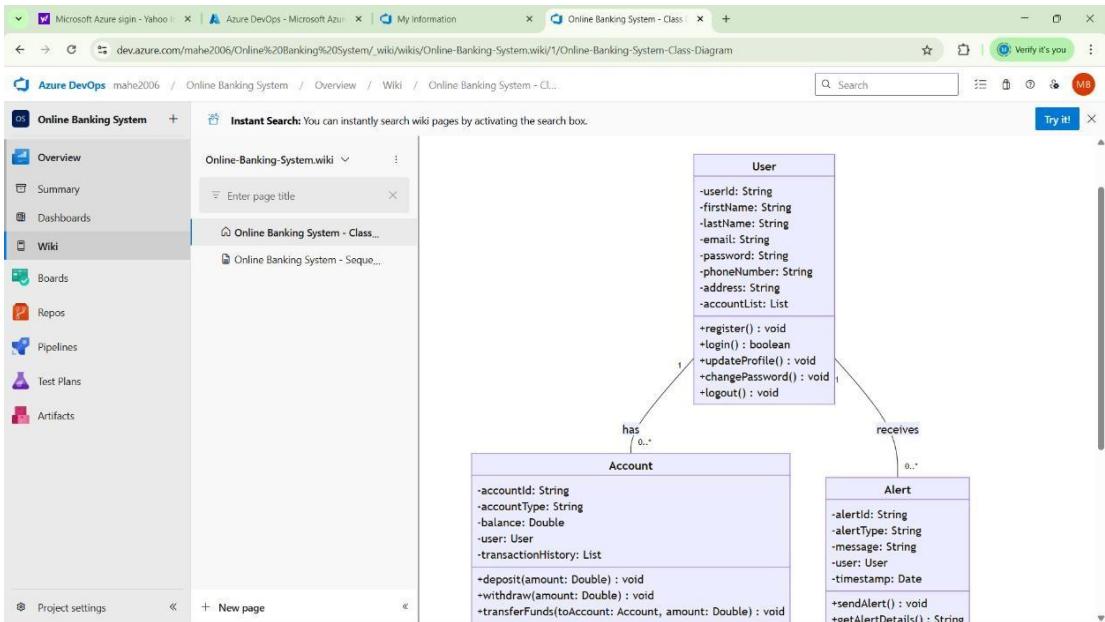
Account "1" -- "0..*" Payment : contains

User "1" -- "0..*" Alert : receives

```

## Relationship Types

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



## Result:

The use case diagram was designed successfully.

## EX NO: 7

### USECASE DIAGRAM

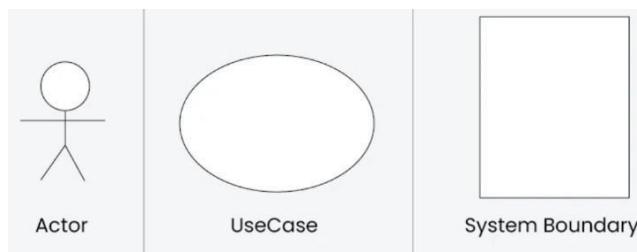
#### Aim:

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.

## USE CASE DIAGRAM:



## Result:

The use case diagram was designed successfully

## EX NO. 8

### ACTIVITY DIAGRAM

#### AIM :-

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

#### THEORY

Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

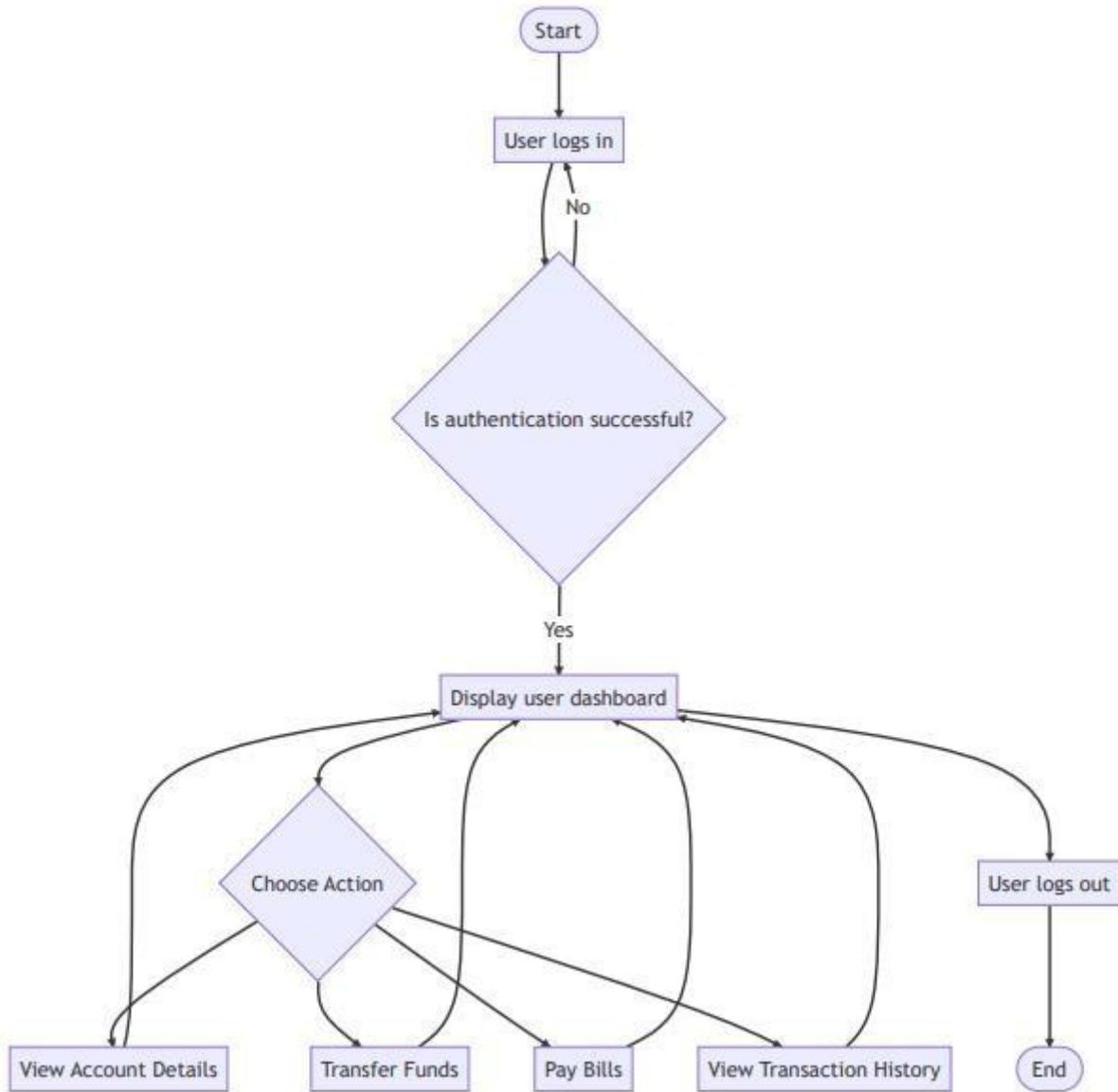
Notations	Symbol	Meaning
Start		Shows the beginning of a process
Connector		Shows the directional flow, or control flow, of the activity
Joint symbol		Combines two concurrent activities and reintroduces them to a flow where one activity occurs at a time
Decision		Represents a decision
Note		Allows the diagram creators to communicate additional messages
Send signal		Show that a signal is being sent to a receiving activity
Receive signal		Demonstrates the acceptance of an event
Flow final symbol		Represents the end of a specific process flow
Option loop		Allows the creator to model a repetitive sequence within the option loop symbol
Shallow history pseudostate		Represents a transition that invokes the last active state.
End		Marks the end state of an activity and represents the completion of all flows of a process

#### Procedure

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

# Online Banking System Activity Diagram

Last updated by | Mahesh Babu | May 13, 2025 at 8:54 PM GMT+5:30

**Result:**

The activity diagram was designed successfully

## EX NO. 9

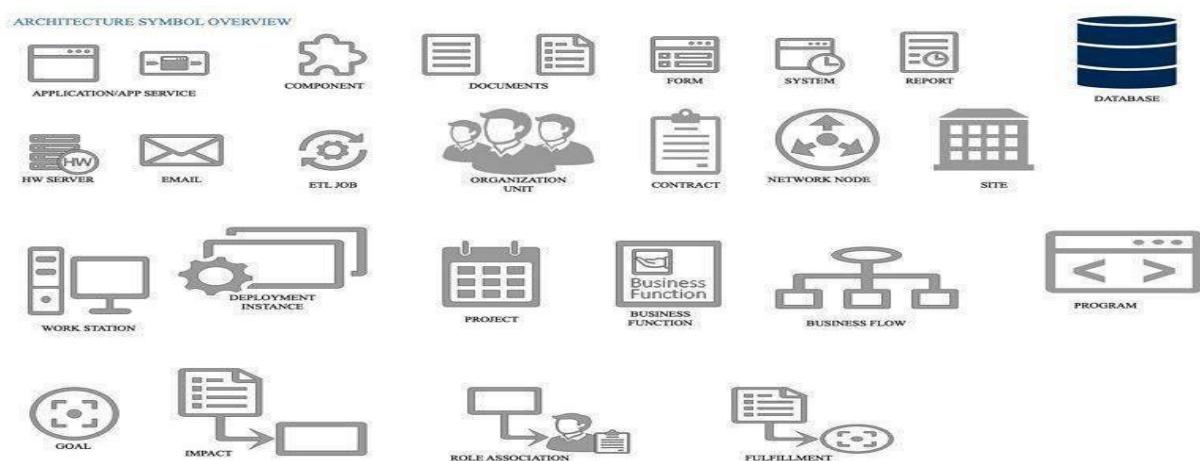
### ARCHITECTURE DIAGRAM

#### Aim:

Steps to draw the Architecture Diagram using draw.io.

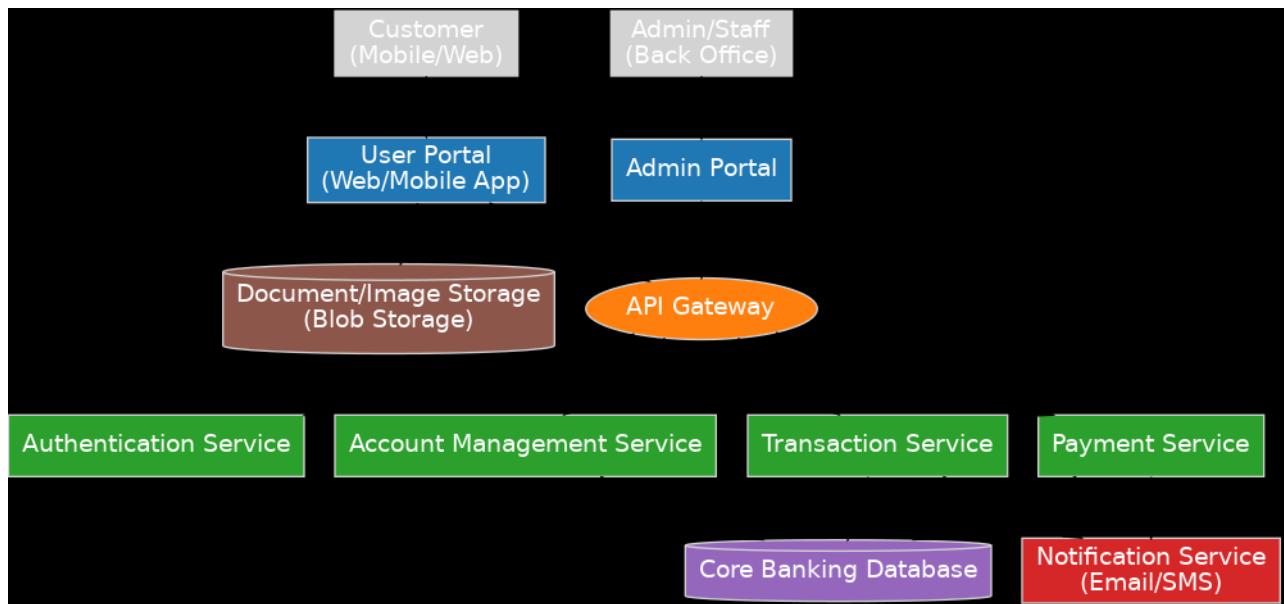
#### Theory:

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. 10

### USER INTERFACE

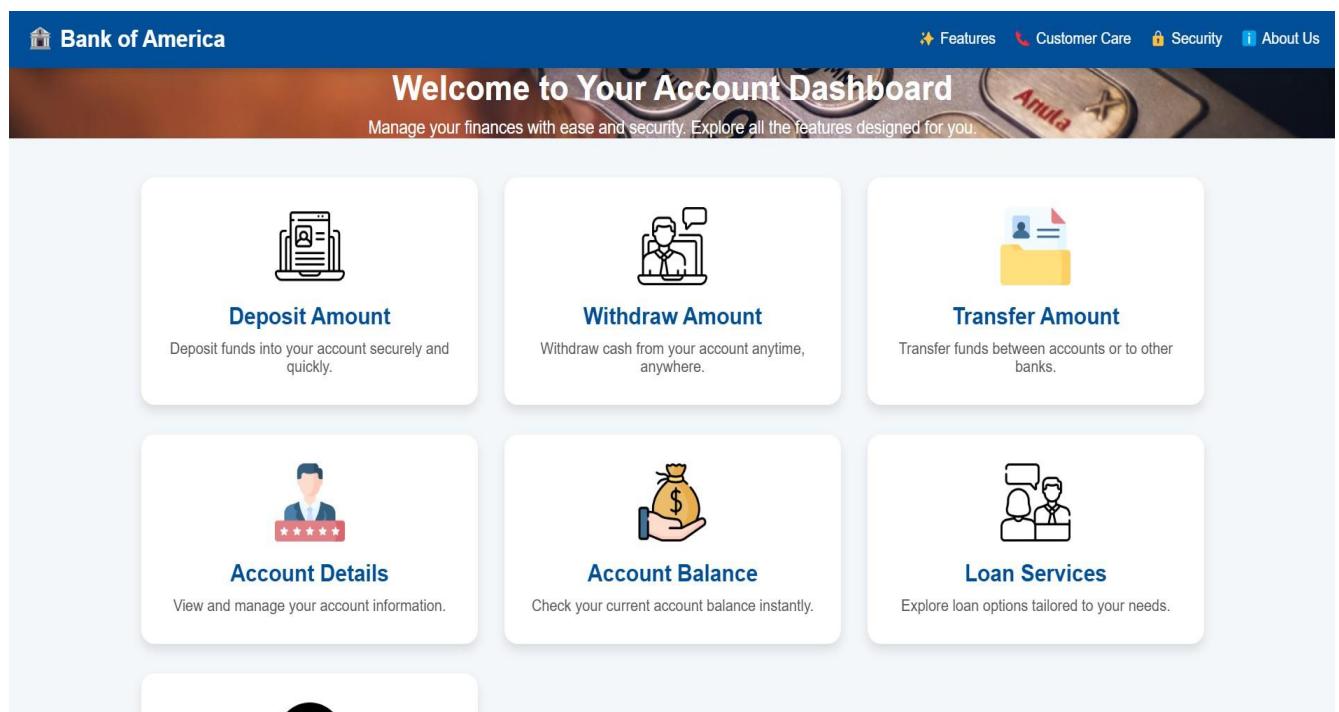
#### Aim:

Design User Interface for the given project

#### User Interface:

User Interface (UI) refers to the visual layout and interactive elements of a software application or website that allow users to interact with the system. It includes components like buttons, menus, input fields, icons, colors, typography, and the overall screen layout.

A well-designed UI ensures that users can easily and efficiently navigate, understand, and use the application to achieve their goals.



**Welcome to Bank of America**

Your trusted partner in banking. Explore our features designed to make your financial life easier and more secure.

## Our Features

**Advanced Security**  
We use cutting-edge technology to protect your accounts and personal

**24/7 Customer Care**  
Our dedicated support team is available round the clock to assist

**Mobile Banking**  
Access your accounts, transfer funds, and pay bills anytime,

**Online Banking**  
Manage your finances seamlessly with our secure and easy-to-use

Result:

The UI was designed successfully.

## **EX NO. 11**

### **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):

```
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.

## **EX NO. 12**

### **TESTING**

#### **a) TESTING-TEST PLANS & 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 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 behaviour (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

The screenshot shows the 'New Test Plan' creation interface in Azure DevOps. The left sidebar lists project navigation options like Overview, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main area is titled 'New Test Plan' and contains three required fields: 'Name' (set to 'Online Banking System'), 'Area Path' (set to 'Online Banking System'), and 'Iteration' (set to 'Online Banking System'). At the bottom right are 'Create' and 'Cancel' buttons.

### 2. Test suite

The screenshot shows the 'Test Suites' page for the 'Login' suite. The left sidebar shows the project navigation. The main area displays the 'Login (1)' suite under the 'Define' tab. It lists one test case: 'Online Banking System-User Login'. The table includes columns for Title, Order, Test Case Id, Assigned To, and Status.

Title	Order	Test Case Id	Assigned To	Status
Online Banking System-User Login	1	14	azar ismail	De

### **3. Test case**

Give two test cases for at least five user stories showcasing the happy path and error scenarios in Azure DevOps platform.

#### **E-Commerce Product Uploader – Test Plans**

#### **USER STORIES**

- As a seller, I want to upload a new product with complete details (ID: 101).
- As a seller, I should be able to see all my listed products (ID: 102).
- As a seller, I should be notified of upload success or failure (ID: 103).
- As a seller, I should be able to edit product information (ID: 104).
- As a seller, I should not be able to upload a product with missing mandatory fields (ID: 105).

#### **Test Suites**

##### **Test Suit: TS01 – Product Upload (ID: 106)**

###### **1. TC01 – Successful Product Upload**

- **Action:**
  - Go to the product upload page.
  - Fill in product name, description, price, image, category, and stock quantity.
  - Click “Upload Product”.
- **Expected Results:**
  - Product form is submitted successfully.
  - Notification "Product uploaded successfully" is displayed.
  - Product appears in seller's product list.
- **Type:** Happy Path

###### **2. TC02 – Upload with Missing Fields**

- **Action:**
  - Go to the product upload page.
  - Leave the “Product Name” and “Price” fields empty.
  - Click “Upload Product”.
- **Expected Results:**
  - Validation fails.
  - Error message "Product Name and Price are required" is shown.
  - Product is not uploaded.
- **Type:** Error Path

###### **3. TC03 – Upload with Invalid Image Format**

- **Action:**
  - Upload a text file instead of a product image.
  - Click “Upload Product”.
- **Expected Results:**
  - Image validation fails.
  - Error “Only image formats (jpg, png) are allowed” is shown.
- **Type:** Error Path

#### **4. TC04 – Upload with Duplicate Product Name**

- **Action:**
  - Enter a product name that already exists in the seller's list.
  - Fill out the remaining details and upload.
- **Expected Results:**
  - System accepts submission.
  - Warning “This product already exists, do you want to continue?” is shown.
- **Type:** Error Path (with optional override)

**Test Suit: TS02 – View & Edit Products (ID: 107)**

#### **1. TC05 – View Uploaded Products**

- **Action:**
  - Log in as a seller.
  - Navigate to “My Products”.
- **Expected Results:**
  - All uploaded products are listed with name, price, and image.
- **Type:** Happy Path

#### **2. TC06 – Edit Existing Product**

- **Action:**
  - Select a product and click “Edit”.
  - Change the price and stock quantity.
  - Click “Save Changes”.
- **Expected Results:**
  - Product updates are saved.
  - Message “Product updated successfully” is shown.
- **Type:** Happy Path

#### **3. TC07 – Edit with Invalid Price**

- **Action:**
  - Edit a product and enter a negative number in the Price field.
  - Click “Save Changes”.
- **Expected Results:**
  - Validation fails.
  - Error “Price must be a positive number” is shown.
- **Type:** Error Path

**Test Suit: TS03 – Upload Notifications (ID: 108)**

#### **1. TC08 – Upload Failure Notification**

- **Action:**
  - Simulate backend failure (e.g., disconnect from server).
  - Try uploading a product.
- **Expected Results:**
  - Upload fails.
  - Message “Upload failed. Please try again later.” is shown.
- **Type:** Error Path

## Test Cases

The screenshot shows the Azure DevOps interface for a test plan. The left sidebar has 'Online' selected under 'Test'. A test case titled '14. Online Banking System-User Login' is displayed. It shows the following details:

- State:** Design
- Area:** Online Banking System
- Reason:** New
- Iteration:** Online Banking System

The 'Steps' section contains the following steps with their expected results:

1. Navigate to the Online Banking System login page (/login) - Expected result: Login page loads with fields for username and password.
2. Enter valid username in the "Username" field - Expected result: Username is accepted (no validation error shown).
3. Enter valid password in the "Password" field - Expected result: Password is accepted (no validation error shown).
4. Click the "Login" button - Expected result: System processes credentials.
5. System authenticates credentials with backend - Expected result: If credentials are correct, user is logged in.
6. Redirect to user dashboard - Expected result: Dashboard loads showing account summary.

Below the steps, there is a 'Parameter values' section. On the right side of the screen, there are sections for 'Deployment', 'Development', and 'Related Work'.

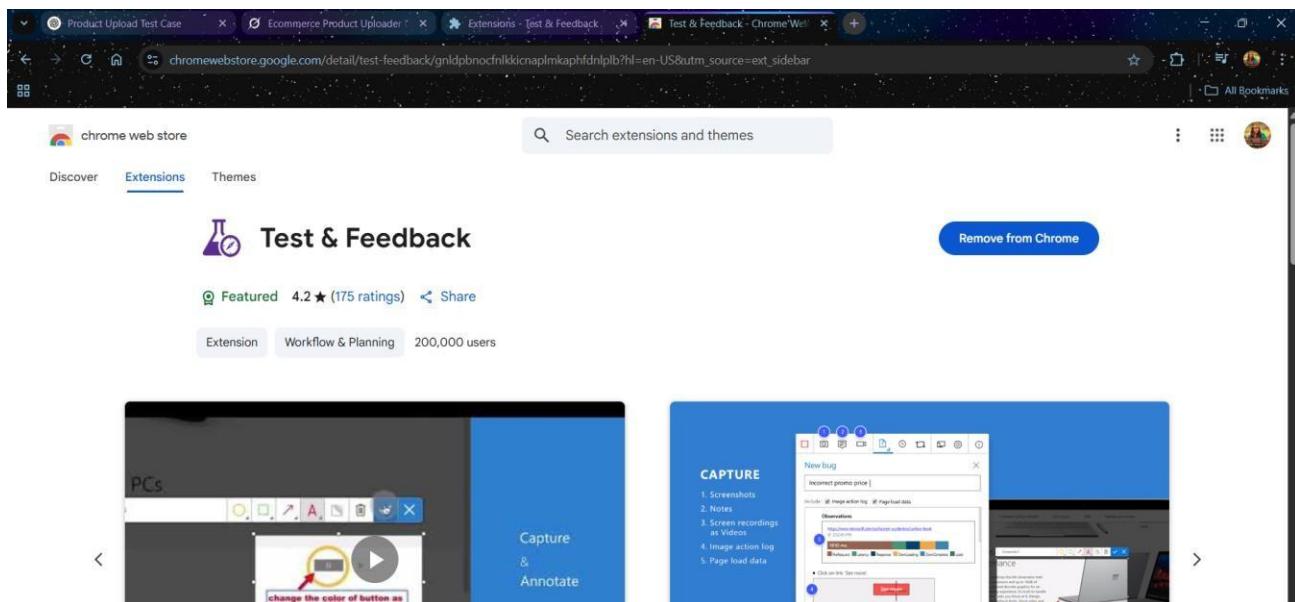
## 4. Installation of test

The screenshot shows the Chrome Web Store page for the 'Test & Feedback' extension. The extension has a rating of 4.2 stars from 175 ratings and over 200,000 users. It is categorized under 'Extension' and 'Workflow & Planning'. The page features two main screenshots:

- PCs:** Shows a screenshot of a computer monitor displaying a web application with a red annotation highlighting a button and the text 'Change the color of button as per UX mockups'.
- Capture:** Shows a screenshot of a software interface with a toolbar at the top and a central window labeled 'CAPTURE' containing a list of items: 1. Screenshots, 2. Notes, 3. Screen recording, 4. Image action log, and 5. Page load data. Below this is a preview of a browser window showing a webpage with annotations.

## Test and feedback

### Showing it as an extension



A screenshot of a Microsoft Edge browser window displaying an Azure DevOps Test Plan. The URL is 'dev.azure.com/Goku312005/E-Commerce%20Product%20Uploader/\_testPlans/define?planId=49&amp;suiteId=50'. The main content area shows a 'TEST CASE 51\*' card with steps for 'TC1:Successful upload of product'. To the right of the card, a sidebar titled 'Extensions' is open, showing 'Full access' and a list of extensions: 'React Developer Tools' and 'Test &amp; Feedback'. Below the extensions, there are sections for 'Deployment', 'Development', 'Related Work', and 'Status'. At the bottom of the sidebar, there's a link to 'Parameter values'. The system tray at the bottom of the screen shows various icons and the date '05-05-2025'.

## 5. Running the test cases

The screenshot shows the Azure DevOps Test Plan interface. On the left, there's a sidebar with project navigation options like Overview, Boards, Repos, Pipelines, Test Plans, Test plans, Progress report, Parameters, Configurations, Runs, and Artifacts. The 'Test Plans' section is currently selected. In the main area, a 'Test Suites' section displays a suite named 'Login' (ID: 13). Below it, a 'Test Cases (1 item)' table lists a single entry:

Title	Order	Test Case Id	Assigned To	Status
Online Banking System-User Login	1	14	azar ismail	De

## 6. Recording the test case

The screenshot shows the 'Runner - Test Plans - Google Chrome' window. It displays a recorded test case for '14: Online Banking System-User Login'. The test steps are as follows:

1. Navigate to the Online Banking System login page (/login)  
EXPECTED RESULT: Login page loads with fields for username and password
2. Enter valid username in the "Username" field  
EXPECTED RESULT: Username is accepted (no validation error shown)
3. Enter valid password in the "Password" field  
EXPECTED RESULT: Password is accepted (no validation error shown)
4. Click the "Login" button  
EXPECTED RESULT: System processes credentials
5. System authenticates credentials with backend  
EXPECTED RESULT: If credentials are correct, user is logged in
6. Redirect to user dashboard  
EXPECTED RESULT: Dashboard loads showing account summary (balance, recent transactions)
7. Display welcome message  
EXPECTED RESULT: "Welcome, John Doe!" message appears

## 7.Creating the bug

The screenshot shows a Microsoft Edge browser window titled "Runner - Test Plans - Google Chrome" with the URL "dev.azure.com/azardeenismail2005/Online%20Banking%20System/\_testExecution/Index". The page displays a test plan for "14: Online Banking System-User Login". A modal dialog box is open, titled "BUG-001: System Didn't Login Even Enter The Correct Username And Password". The dialog includes fields for State (New), Area (Online Banking System), Reason (New), Iteration (Online Banking System), and a "Repro Steps" section. The "Repro Steps" section contains two steps: 1. "None" (Navigate to the Online Banking System login page (/login)) and 2. "None" (Enter valid username in the "Username" field). The "Planning" and "Deployment" sections are visible on the right.

## 8.Test case results

The screenshot shows a Microsoft Edge browser window with multiple tabs open, including "Create Your Azure Free Account", "Azure DevOps - Microsoft Azure", "My Information", "Test Plan 12 Login - Test Plans", and "Test Plan 12 Login - Test Plans". The main view is the "Test Plans" section for the "Online Banking System" project. On the left, the navigation menu includes "Overview", "Boards", "Repos", "Pipelines", "Test Plans", "Test plans", "Progress report", "Parameters", "Configurations", "Runs", and "Artifacts". The "Test Plans" section is currently selected. In the center, a "Login (ID: 13)" card is displayed, showing "May 14 - May 21" and "50% run, 100% passed". Below it, a "Test Suites" section lists "Login (2)". To the right, a "Test Case Results" table is shown for the "Online Banking System-Deposit Amount" test point. The table has columns for "Outcome", "TimeStamp", "Configuration", "Run by", "Tester", and "Test". Four rows show "Passed" outcomes for different runs, all performed by "azar ismail" on "Windows 10".

Outcome	TimeStamp	Configuration	Run by	Tester	Test
Passed	Just now	Windows 10	azar ismail	azar ismail	Login
Passed	3h ago	Windows 10	azar ismail	azar ismail	Login
Passed	3h ago	Windows 10	azar ismail	azar ismail	Login
Passed	3h ago	Windows 10	azar ismail	azar ismail	Login

## 9.Test report summary

The screenshot shows the Azure DevOps Work Items page for a project named "E-Commerce Product Uploader". A specific bug, "BUG-001: Product not uploaded even after entering valid details", has been resolved by "Gokul Krishna R". The bug was initially marked as "New" and is now "Resolved". The "Repro Step" section contains three steps: 1. Passed (Navigate to the "Upload Product" page, Expected Result: Product upload form should be displayed), 2. Passed (Enter product name, Expected Result: Product name field is filled), and 3. Failed (Enter product description). The "Planning" and "Deployment" sections provide details about the bug's status and history. The "Development" section includes links to Azure Repos and branches. The "Related Work" section lists a linked work item.

- Assigning bug to the developer and changing state

## 10.Progress report

The screenshot shows the Azure DevOps Test Plans Progress report for the "Online Banking System". The left sidebar highlights the "Test Plans" and "Progress report" sections. The main area displays a "Summary" card with 1 test plan and 2 test points, showing 50% run completion. Below it is a "Pass rate" card at 100% (1/1). To the right is an "Outcome trend" chart for the last 14 days, showing a sharp increase in tests run starting around May 13th, with a legend indicating "Not run" (grey) and "Passed" (green).

The screenshot shows the 'All processes' section of the Azure DevOps Settings - Process page. The left sidebar is collapsed, and the main area displays a table of process templates:

Name	Description	Team projects
Basic (default)	This template is flexible for any process and great for teams getting started with Azure DevOps.	1
Agile	This template is flexible and will work great for most teams using Agile planning methods, including those practicing Scrum.	1
GokuB12005 Agile	This template is for teams who follow the Scrum framework.	1
Scrum	This template is for teams who follow the Scrum framework.	0
CMMI	This template is for more formal projects requiring a framework for process improvement and an auditable record of decisions.	0

## 11. Changing the test template

The screenshot shows the 'All processes' section of the Azure DevOps Settings - Process page. The left sidebar is collapsed, and the main area displays a table of process templates:

Name	Description	Team projects
Basic (default)	This template is flexible for any process and great for teams getting started with Azure DevOps.	1
Agile	This template is flexible and will work great for most teams using Agile planning methods, including those practicing Scrum.	1
GokuB12005 Agile	This template is for teams who follow the Scrum framework.	1
Scrum	This template is for teams who follow the Scrum framework.	0
CMMI	This template is for more formal projects requiring a framework for process improvement and an auditable record of decisions.	0

## 12. View the new test case template

The screenshot shows the 'Add a field to Test Case' dialog box over a background of the Azure DevOps 'Process' settings page. The dialog has tabs for 'Definition', 'Options', and 'Layout'. Under 'Definition', the 'Create a field' option is selected, showing fields for 'Name' (Acceptance Criteria), 'Type' (Text (single line)), and 'Description'. A large text area on the right is labeled 'Add a field ...'.

The screenshot shows the 'Test Case' process template in the Azure DevOps 'Process' settings. It includes sections for 'Steps' (Text (multiple lines)), 'Recent test results' (Recent test case results), 'Deployment' (Deployments), 'Development' (Links), 'Related Work' (Links), and 'Status' (Priority: Integer, Automation status: Text (single line)). A large text area on the right is labeled 'Add a field ...'.

The screenshot shows the Azure DevOps Settings - Process page. The URL in the browser is [dev.azure.com/Gokul312005/\\_settings/process?process-name=Gokul312005%20Agile&a=projects](https://dev.azure.com/Gokul312005/_settings/process?process-name=Gokul312005%20Agile&a=projects). The left sidebar is titled "Organization Setti..." and lists various settings categories: General, Security, Boards, Pipelines, and Process (which is selected). The main content area is titled "All processes > Gokul312005 Agile" and shows a table with one item: "E-Commerce Product Uploader". The table columns are "Name" and "Description". The "Name" column shows "E-Commerce Product Uploader" and the "Description" column shows "The E-Commerce Product Uploader is a tool that allows sellers to effortlessly add and manage products on their online store. It supports bulk uploads, image management, and automated data ...".

## Result:

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

## **b) Load Testing and Performance Testing**

### **Aim:**

To create an Azure Load Testing resource and run a load test to evaluate the performance of a target 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:

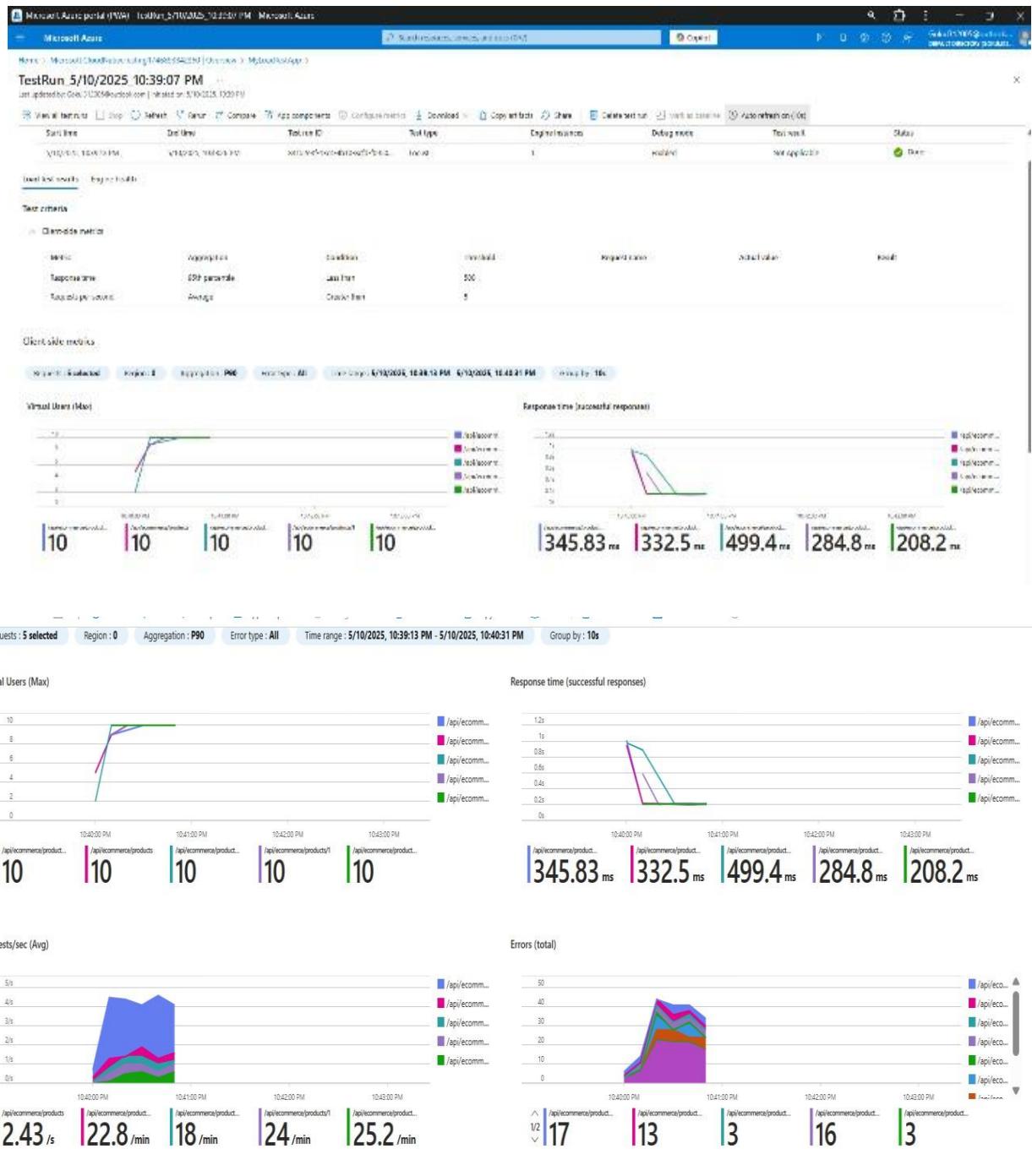
1. Sign in to Azure Portal  
Go to <https://portal.azure.com> and log in.
2. Create the Resource
  - o Go to *Create a resource* → Search for “Azure Load Testing”.
  - o Select Azure Load Testing and click Create.
3. Fill in the Configuration Details
  - o *Subscription*: Choose your Azure subscription.
  - o *Resource Group*: Create new or select an existing one.
  - o *Name*: Provide a unique name (no special characters).
  - o *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.

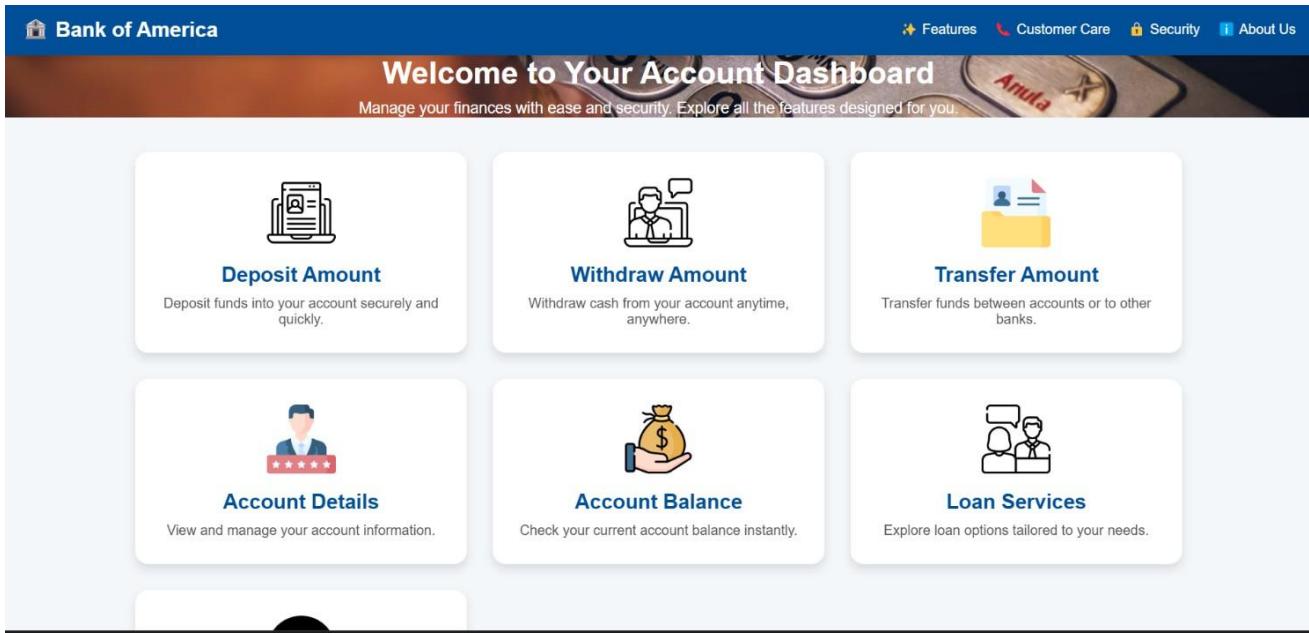
#### **Steps to Create and Run a Load Test:**

Once your resource is ready:

1. Go to your Azure Load Testing resource and click Add HTTP requests > Create.
2. Basics Tab

## Load Testing





**Result:**

Successfully created the Azure Load Testing resource and executed a load test to assess the performance of the specified endpoint.