



**RAJALAKSHMI  
ENGINEERING COLLEGE**

An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

**DEPARTMENT OF COMPUTER SCIENCE &  
ENGINEERING ACADEMIC YEAR 2024-2025**

**EVEN SEMESTER**



**CS23432 SOFTWARE ENGINEERING LAB**

**LAB MANUAL**

**SECOND YEAR**

**FOURTH SEMESTER**

**2024- 2025**

**EVEN SEMESTER**

Ex No	List of Experiments
1	Study of Azure DevOps
2	Designing Project using AGILE-SCRUM Methodology.
3	Agile Planning
4	User stories – Creation
5	Architecture Diagram Using AZURE
6	Designing Usecase and Class Diagram
7	Designing Interaction Diagrams
8	Design Interface
9	Implementation – Design a Web Page based on Scrum Methodology
10	Testing using Azure.
11	Deployment

Requirements	
Hardware	Intel i3, CPU @ 1.20GHz 1.19 GHz, 4 GB RAM, 32 Bit Operating System
Software	StarUML , Azure

## LAB PLAN

### CS19442-SOFTWARE ENGINEERING LAB

Ex No	Date	Topic	Page No	Sign
1		Study of Azure DevOps		
2		Writing Problem Statement		
3		Designing Project using AGILE-SCRUM Methodology by using Azure.		
4		Agile Planning		
5		User stories – Creation		
6		Architecture Diagram Using AZURE		
7		Designing Usecase Diagram using StarUML		
8		Designing Activity Diagrams using StarUML		
9		Designing Sequence Diagrams using StarUML		
10		Design Class Diagram		
10		Design User Interface		
11		Implementation – Design a Web Page based on Scrum Methodology		
12		Testing		
13		Deployment		

## 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: “-”

## **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:**

**Employee Leave Management System (ELMS) :**

Many organizations still rely on manual methods such as emails, spreadsheets, and paper forms to manage employee leave, leading to frequent errors, approval delays, lack of transparency, and difficulties in enforcing leave policies. This fragmented approach results in miscommunication, inaccurate leave tracking, and inefficient HR operations. The absence of a centralized system also makes it challenging to generate reports or ensure policy compliance. To address these issues, a digital Employee Leave Management System (ELMS) is essential for streamlining leave requests, automating approvals, maintaining accurate records, and improving overall organizational efficiency.

**Result:**

The problem statement was written successfully.



**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.

**CREATE USER STORIES****Aim:**

To create User Stories

**THEORY**

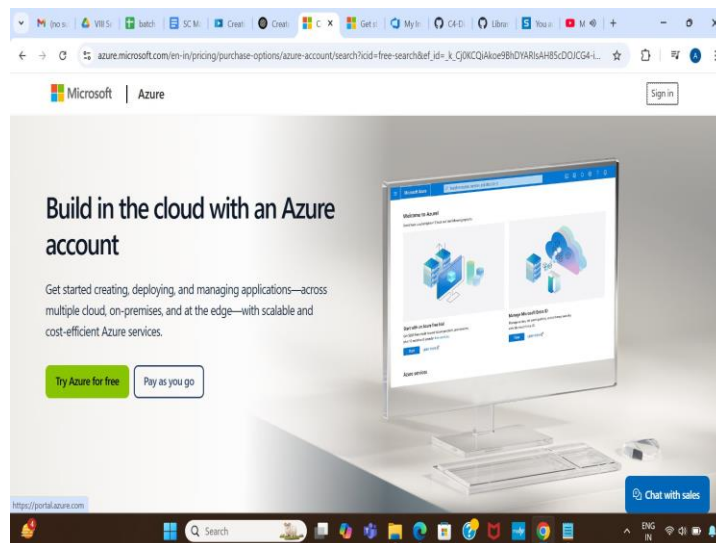
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

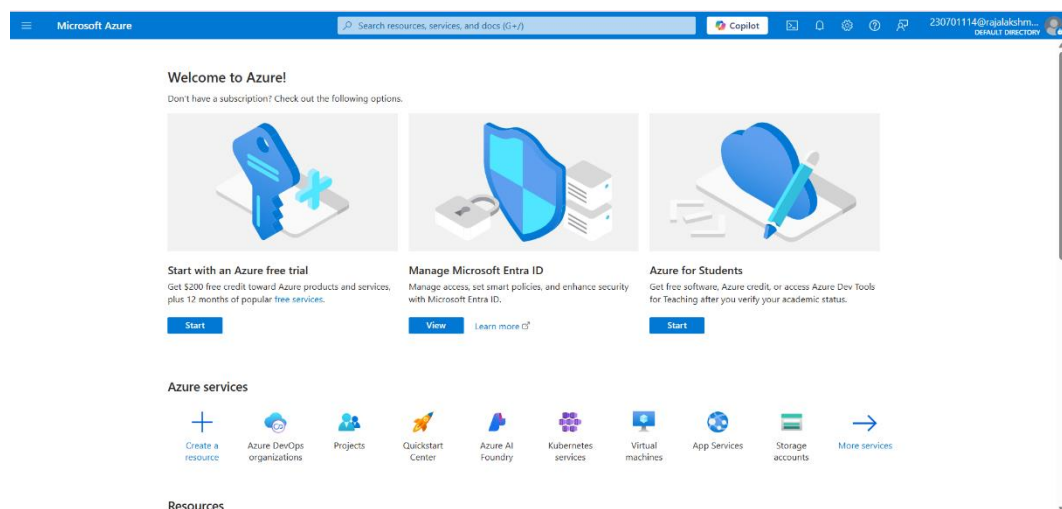
"As a [role], I [want to], [so that]."

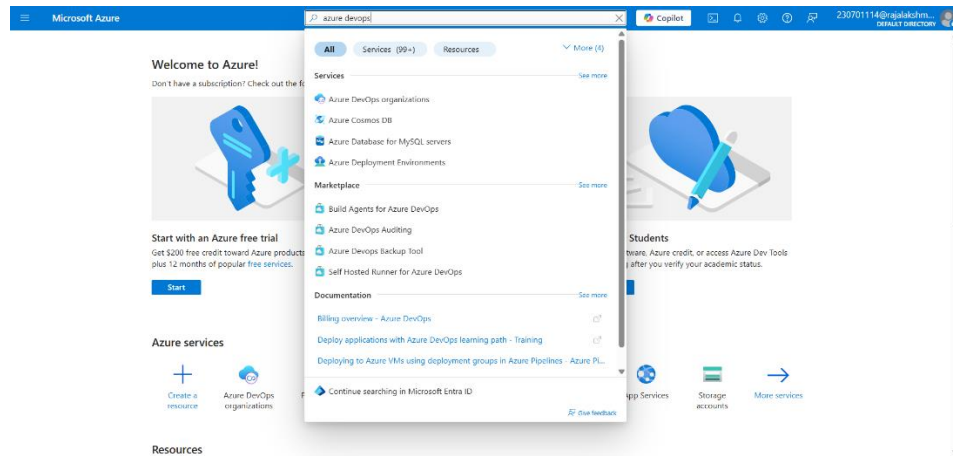
**Procedure:**

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>



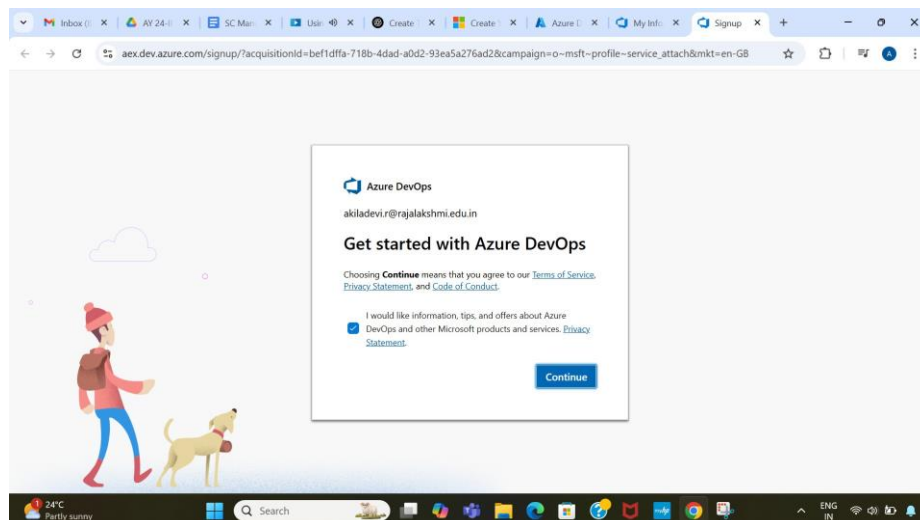
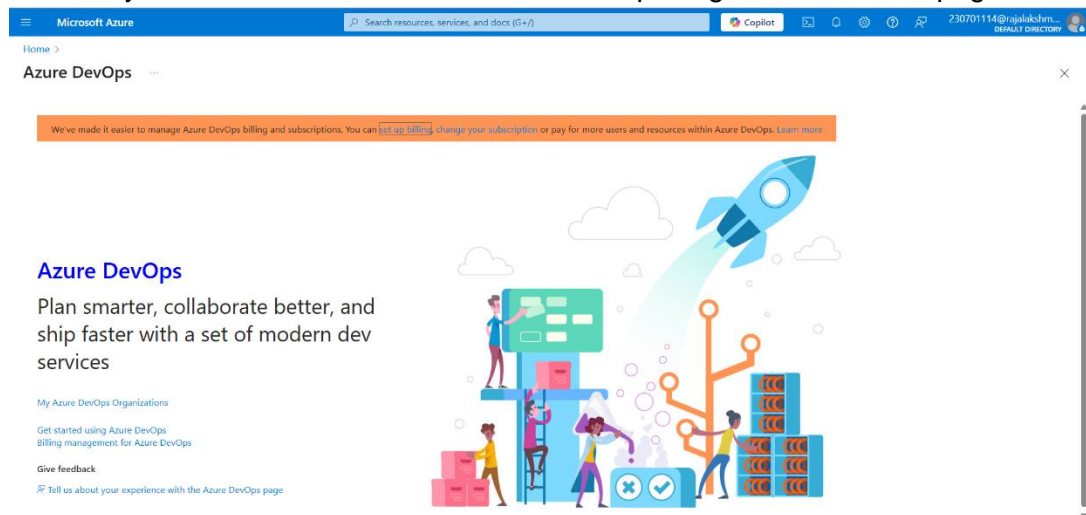
## 3. Azure home page

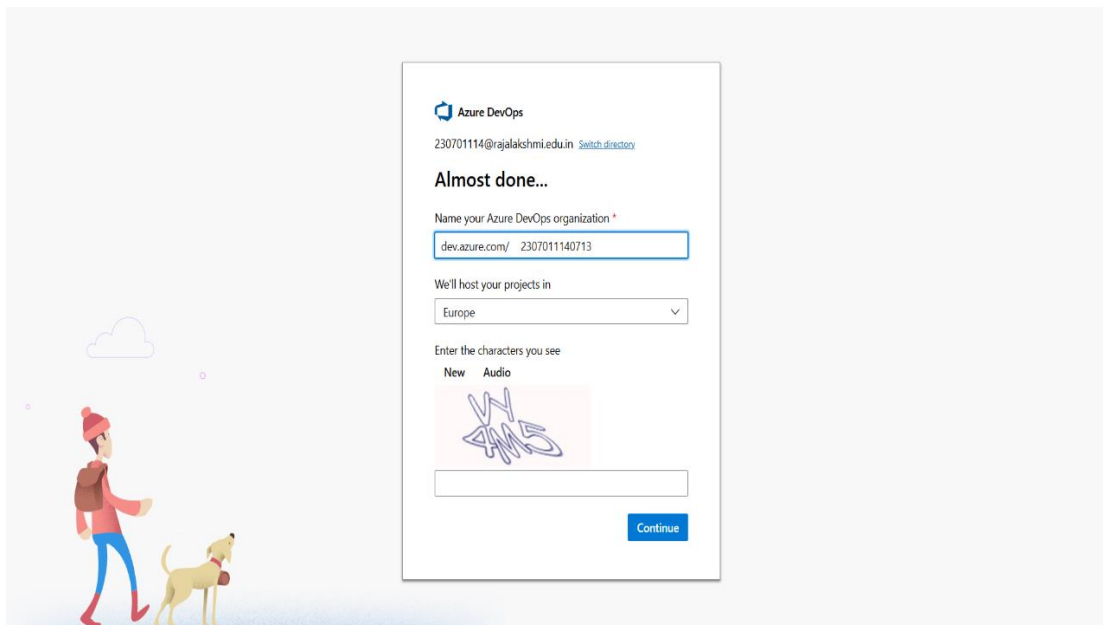




4. Open DevOps environment in the Azure platform by typing Azure DevOps Organizations in the search bar.

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.

Microsoft

Himeshwar N Sign out



Himeshwar N

230701116@rajalakshmi.edu.in

Microsoft account

India

230701116@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/230701114 (Member)

Projects

Online Leave Management System

Actions

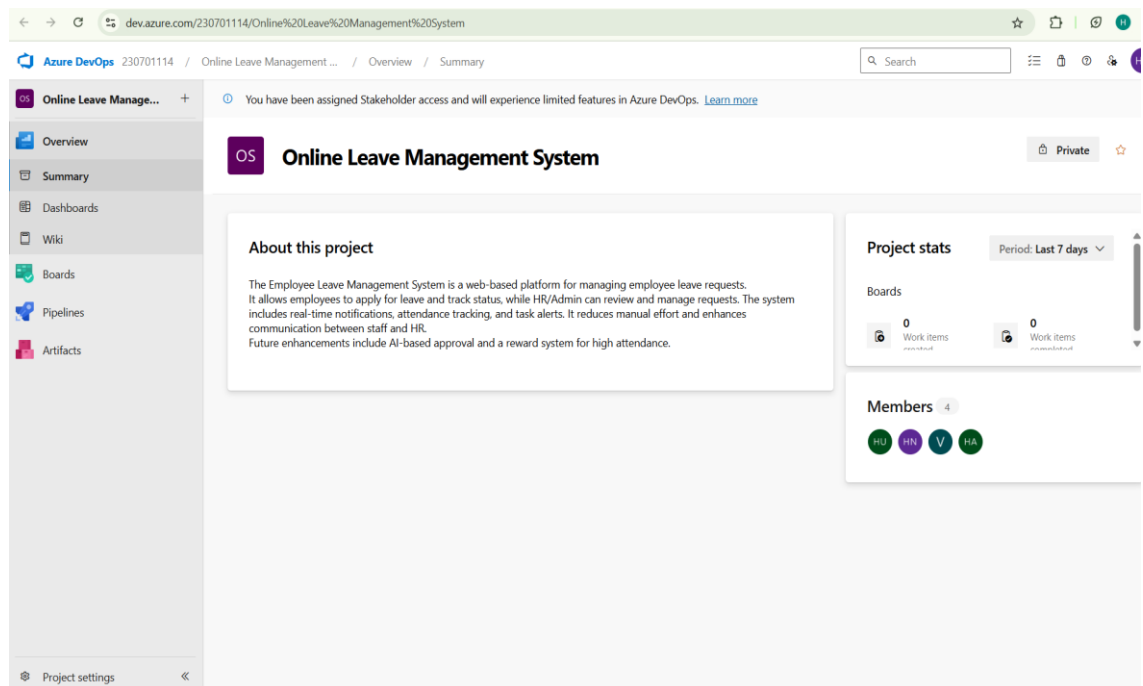
Open in Visual Studio

Manage security

Browse extensions

Leave

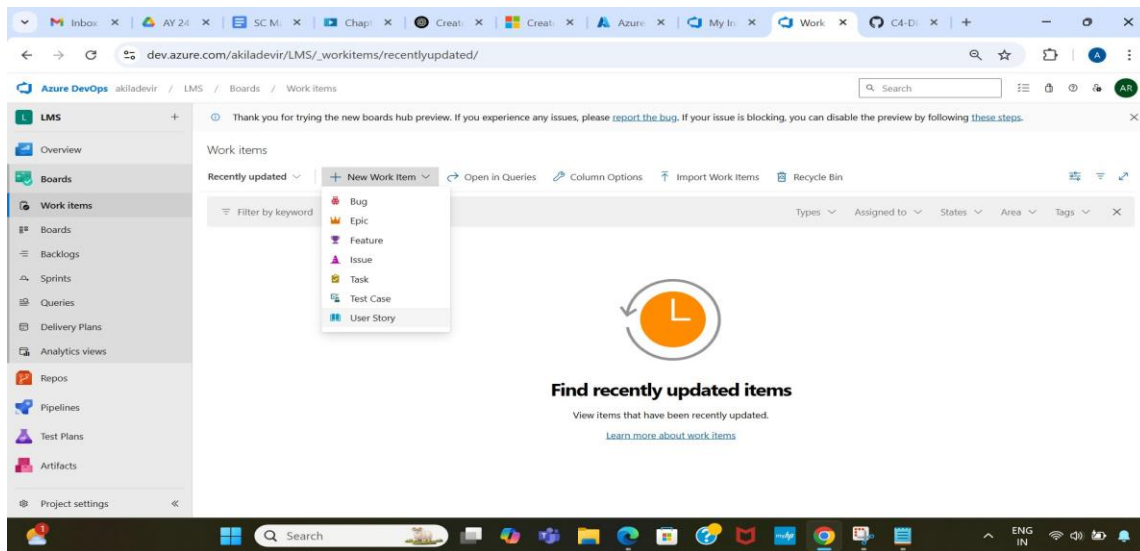
## 8. Project dashboard



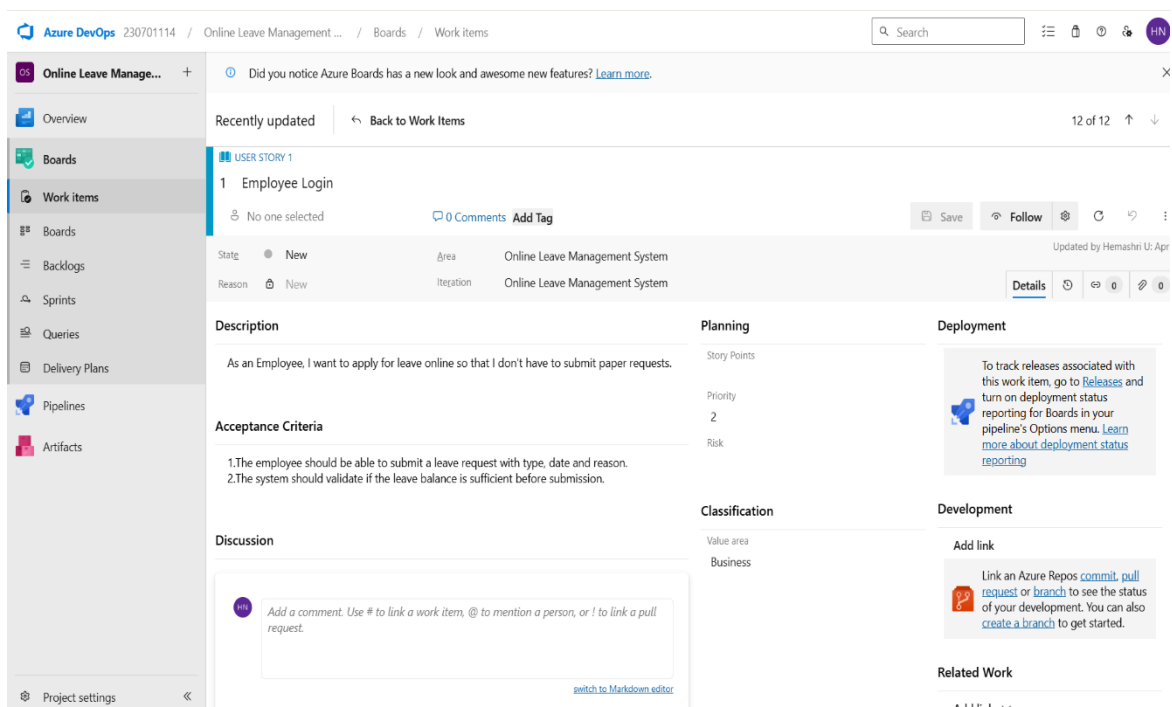
The screenshot shows the Azure DevOps interface for a project named 'Online Leave Management System'. The left-hand navigation menu includes Overview, Summary, Dashboards, Wiki, Boards, Pipelines, and Artifacts. The main content area displays 'About this project' with a description of the system, 'Project stats' for the last 7 days showing 0 work items, and a 'Members' section with 4 members (HN, V, HA, and another HN).

9. 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.



## 10. Fill in User Story Details



## Result:

The user story was written successfully.

**EX NO: 5**

## **SEQUENCE DIAGRAM**

### **Aim:**

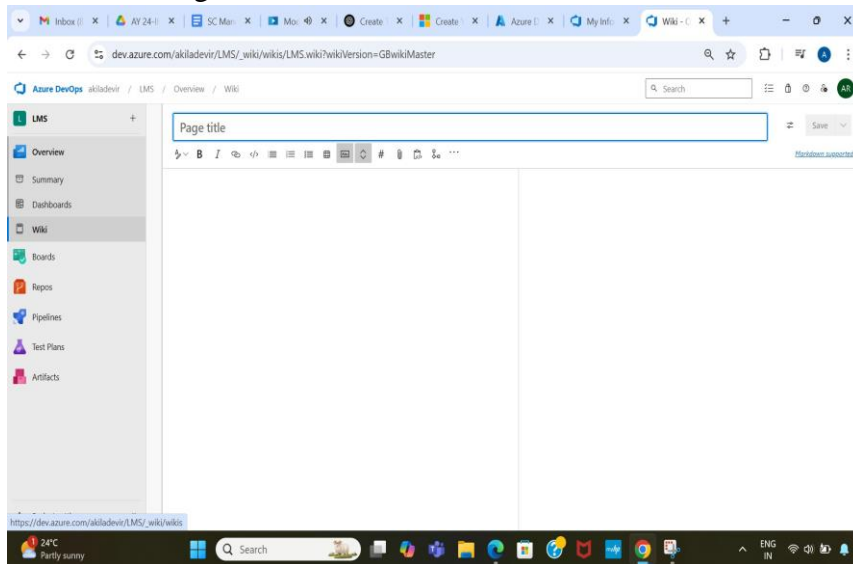
To design a Sequence Diagram by using Mermaid.js

### **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.

```
sequenceDiagram
```

SuperAdmin ->> System: Login

SuperAdmin ->> System: Add departments

SuperAdmin ->> System: Add HRs and Employees

SuperAdmin ->> System: Define/Update leave  
categories

Employee ->> System: Login

Employee ->> System: View profile (department,  
leave balance)

Employee ->> System: Submit leave request (type,  
duration, reason)

System ->> System: Validate request

System ->> HR: Send leave request to respective HR

```

%% HR processes the request
HR ->> System: Login
HR ->> System: View pending requests
HR ->> System: Review leave request details
HR ->> System: Approve/Reject request
System ->> Employee: Notify decision
(Approved/Rejected)
System ->> System: Update leave status
System ->> System: Update attendance and leave
balance

%% Alternative path: Task alert during leave
alt Important task during leave period
    System ->> Employee: Send alert (task scheduled)
end

%% Alternative path: Monthly/Quarterly check
alt Monthly/Quarterly check
    System ->> Employee: Award badge if attendance
    > 95%
end
:::

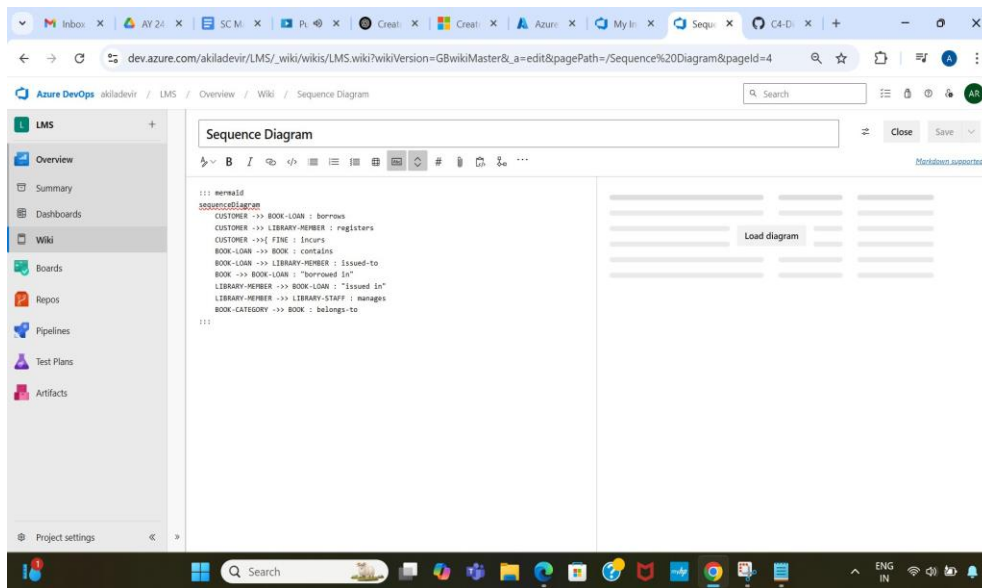
```

### **Explanation:**

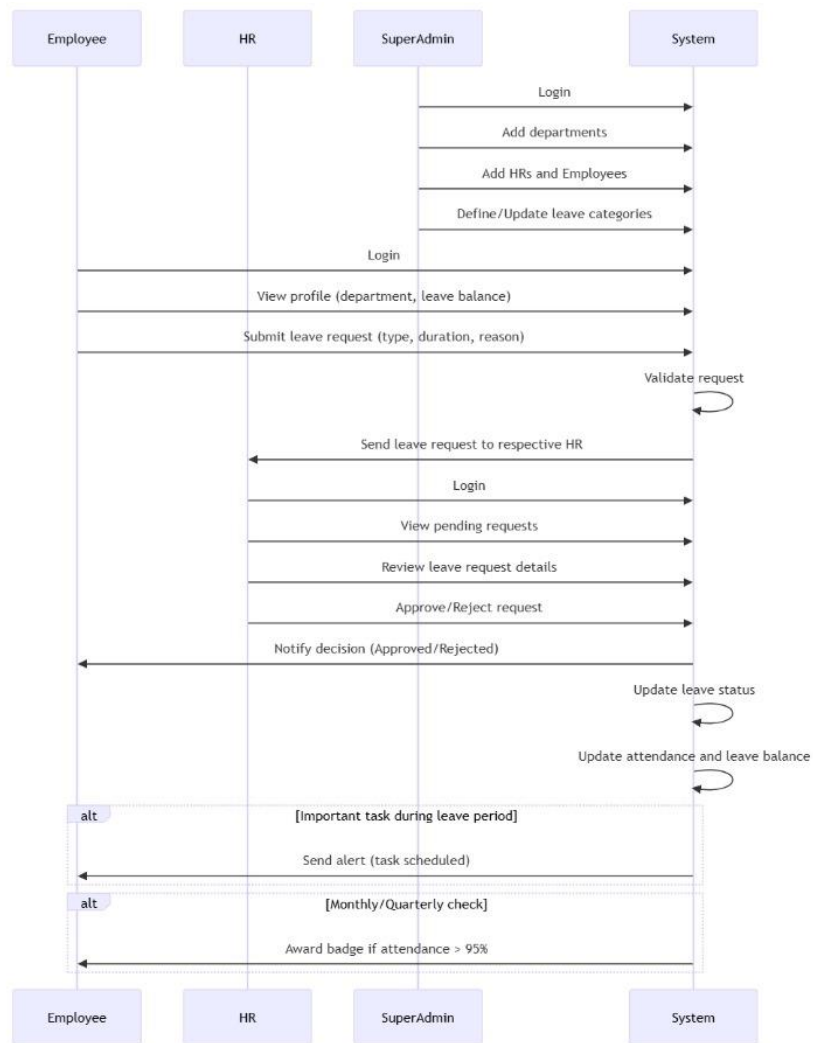
participant defines the entities involved.  
 ->> represents a direct message.  
 -->> represents a response message.  
 + after ->> activates a participant.



- 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

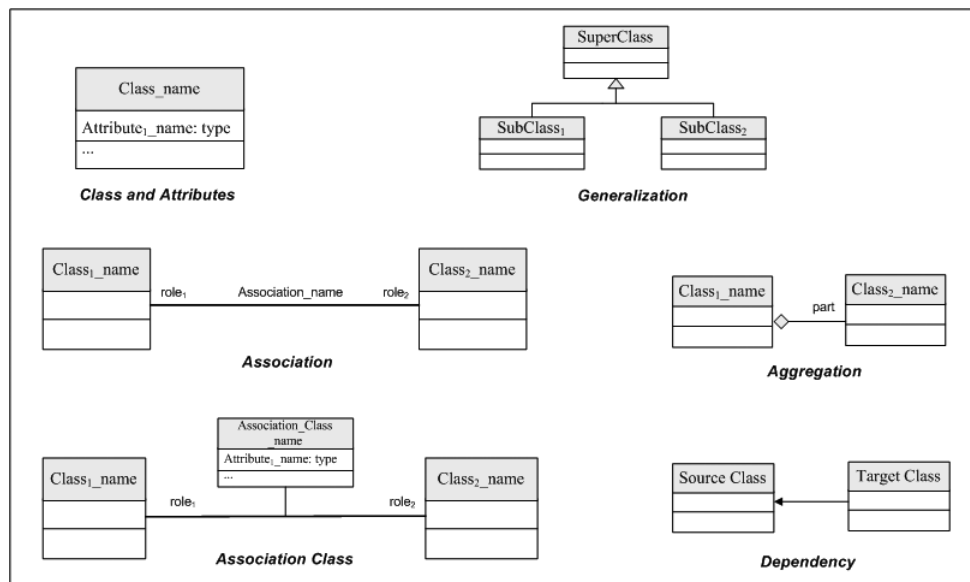
### CLASS DIAGRAM

#### AIM :-

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

#### THEORY

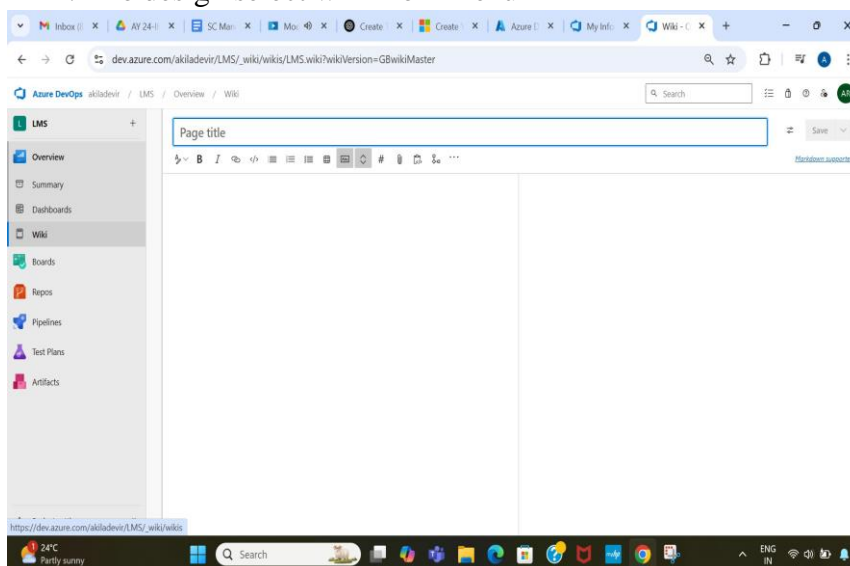
A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.



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

classDiagram

```
class Role {  
    +int role_id  
    +string role_title  
    +string role_description  
    +add_roles()  
    +edit_roles()  
    +delete_roles()  
    +search_roles()  
    +assign_roles()  
}
```

```
class User {  
    +string user_id  
    +int user_role_id  
    +string user_name  
    +string user_email  
    +addUser()  
    +editUser()  
    +deleteUser()  
    +searchUser()  
}
```

```
class Login {  
    +string user_id  
    +string password  
    +string role  
    +authentication()  
    +getUserrole()  
    +login()  
    +logout()
```

```
}
```

```
class Employee {  
    +string employee_id  
    +string name  
    +string email  
    +string department  
    +string designation  
    +int leavebalance  
    +float attendance  
    +applyLeave()  
    +checkLeaveBalance()  
    +viewHistory()  
}
```

```
class HRAdmin {  
    +string manager_id  
    +string manager_name  
    +string manager_email  
    +string department  
    +approveLeave(): Boolean  
    +rejectLeave(): Boolean  
    +viewLeaveRequest(): list  
    +generateLeaveReport(): report  
}
```

```
class SuperAdmin {  
    +string superadmin_id  
    +string superadmin_name  
    +string superadmin_email  
    +addHREmployee()  
    +reviewLeaveRequest()
```

```
+createPolicy()  
  
+createLeaveCatagory()  
  
}
```

```
class LeaveRequest {  
  
    +string leave_id  
  
    +string employee_id  
  
    +string leave_type  
  
    +date start_date  
  
    +date end_date  
  
    +string status  
  
    +string reason  
  
    +submitRequest()  
  
    +cancelRequest()  
  
}
```

```
class LeavePolicy {  
  
    +string policy_id  
  
    +int max_CL  
  
    +int max_SL  
  
    +boolean carry_forward_allowed  
  
    +validateLeaveRequest(): Boolean  
  
}
```

```
class Notification {  
  
    +string notification_id  
  
    +string recipient_id  
  
    +string message  
  
    +string status  
  
    +sendNotification()  
  
    +receiveNotification()  
  
}
```

## %% Relationships

User --> Role : parent

User --> Login : child

User --> HRAdmin : parent

HRAdmin --> SuperAdmin : Extends

HRAdmin --> LeaveRequest : Receives

SuperAdmin --> LeaveRequest : reviewLeaveRequest()

SuperAdmin --> LeavePolicy : Creates

LeaveRequest --> Employee : 1

LeaveRequest --> LeavePolicy : Use

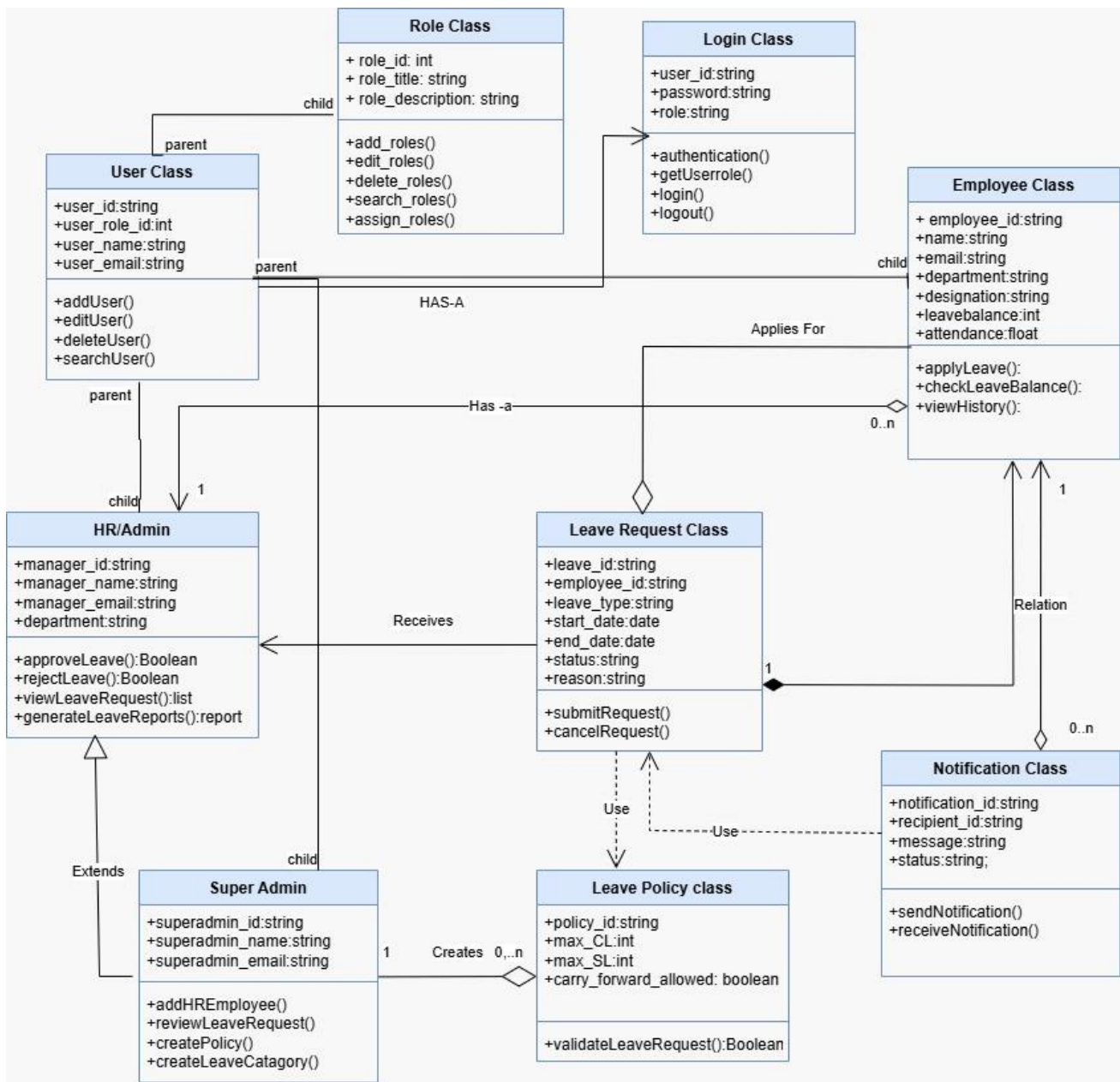
Employee --> LeaveRequest : 0..n

Employee --> Notification : 0..n

Notification --> Employee : 1

## Relationship Types

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



Visit : <https://mermaid.js.org/syntax/classDiagram.html>

## Result:

The use case diagram was designed successfully.



## **USECASE DIAGRAM**

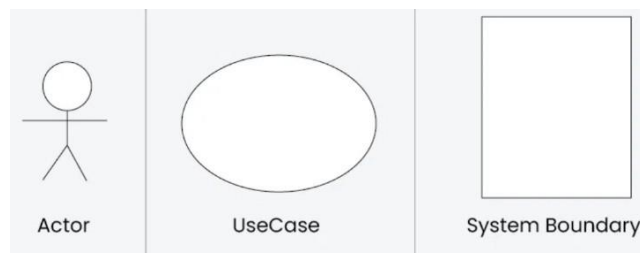
### **Aim:**

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.

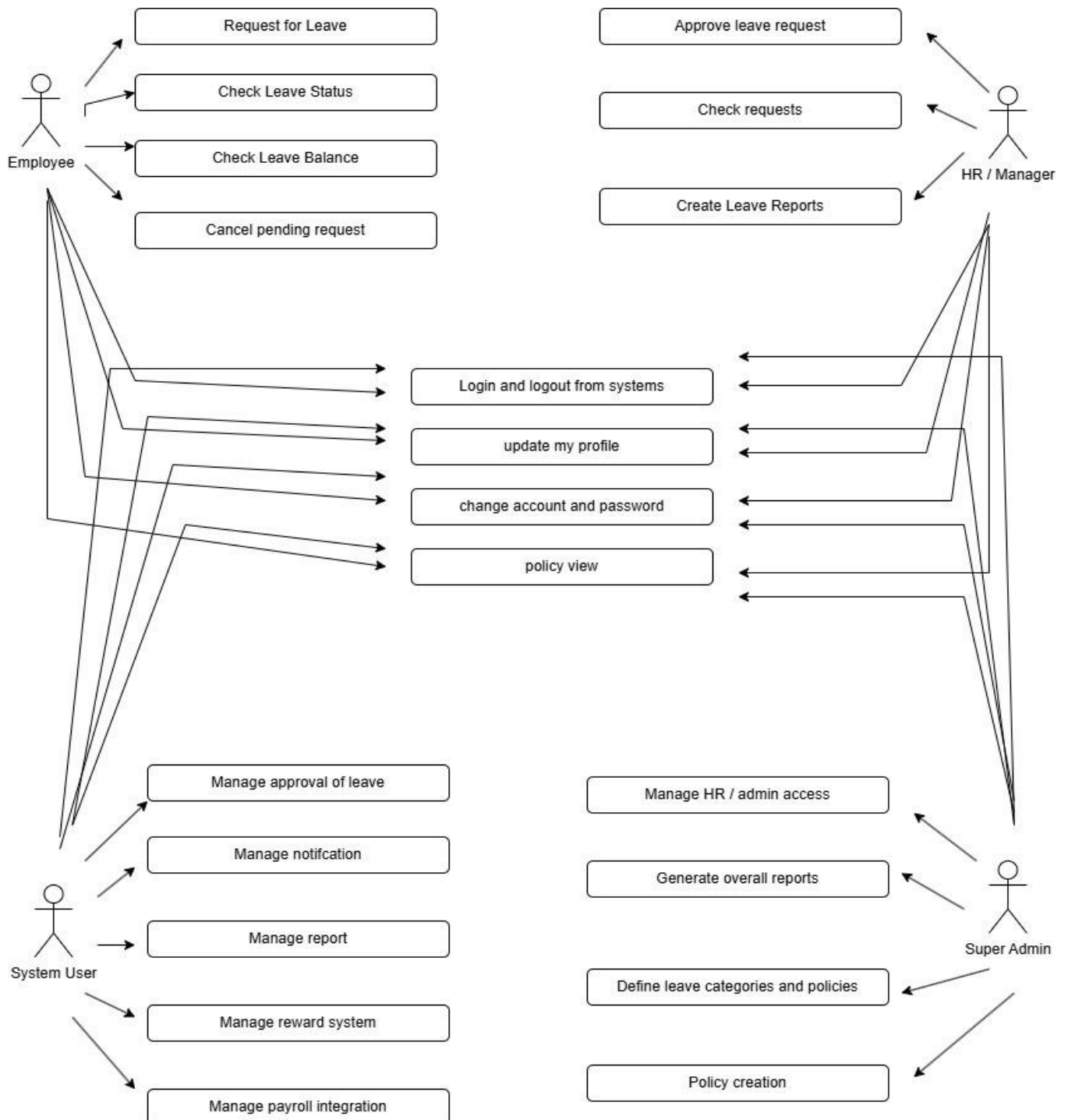
Step 2: Upload the Diagram to Azure DevOps

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



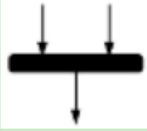








### ACTIVITY DIAGRAM

#### 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 re-introduces 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



**Result:**  
The activity diagram was designed successfully

## EX NO. 9

### ARCHITECTURE DIAGRAM

#### Aim:

Steps to draw the Architecture Diagram using draw.io.

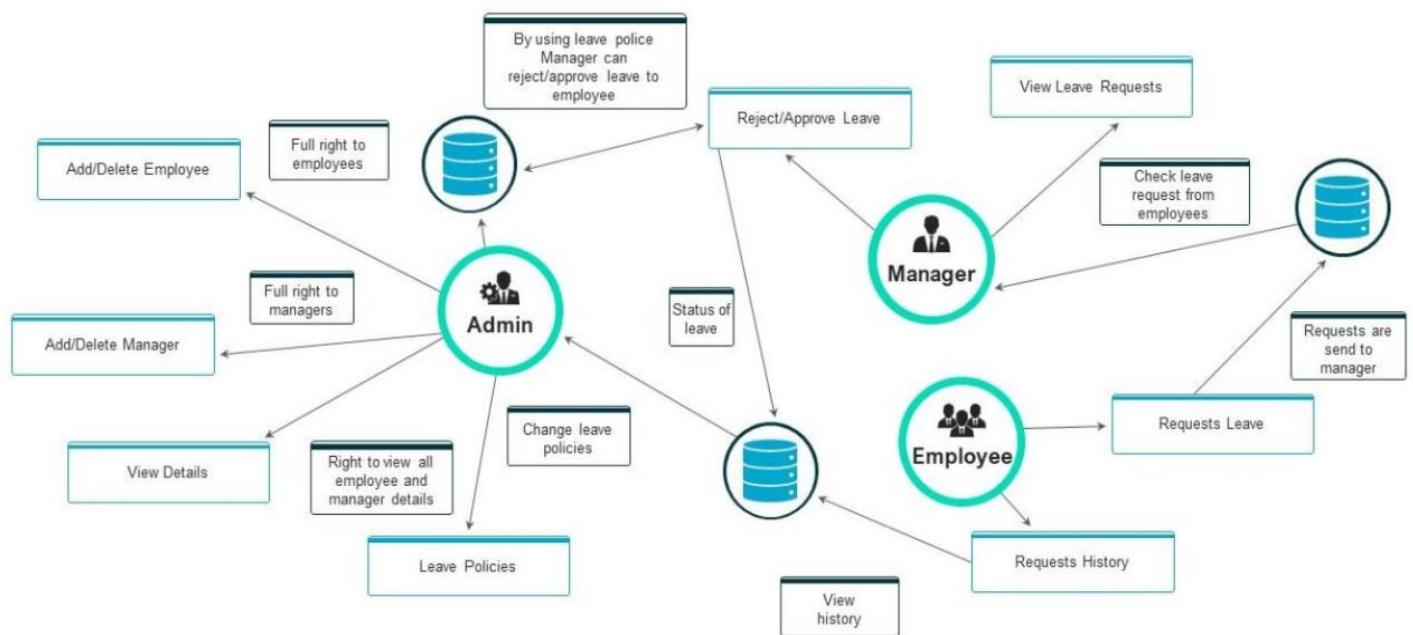
#### Theory:

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

### Login Page (Initial Opening Page)

The login page features a solid blue background. At the top center, the text "Leave Management System" is displayed in a white, sans-serif font. Below this, there are two white input fields for "Username:" and "Password:". Each field has a red label on the left and a white arrow pointing to the right. Below the password field, there is a checkbox labeled "Remember me" and a link "Forgot Password" in a smaller, lighter blue font. To the right of these fields is a dark blue button with the text "Sign In" in white. At the bottom center, there is a red link that says "Register".

### Edit User

The "Edit User" page has a light gray background. On the left, there is a dark gray sidebar with a blue header containing a white menu icon. The sidebar lists several options: "Home", "Manage Users", "Manage Leaves", "Apply Leave", "My Leaves", and "Change Password". The main content area has a blue header with the text "Leave Management System" and a user profile dropdown showing "manager@email.com" with a "[MANAGER]" label. Below the header, there is a breadcrumb trail "Home > Edit User". The main title "Edit User : naveenkumark@email.com" is displayed in a large, blue font. Below this, there are four input fields for "First Name:" (Naveen Kumar), "Last Name:" (K), "E-mail:" (naveenkumark@email.com), and "Password:" (Password). Below these fields, there is a "Role:" section with two radio buttons: "ADMIN" (selected) and "EDITOR". At the bottom, there are two buttons: a blue "SAVE" button and a gray "CANCEL" button.



## Home Page (Options provided for data variation)

Home

Manage Users

Manage Leaves

Apply Leave

My Leaves

Change Password

Leave Management System

manager@email.com  
[MANAGER]

Home >

Pending

Accepted

Rejected

< > today

November 2018

month week list

Sun	Mon	Tue	Wed	Thu	Fri	Sat
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Home

Manage Users

Manage Leaves

Apply Leave

My Leaves

Change Password

Leave Management System

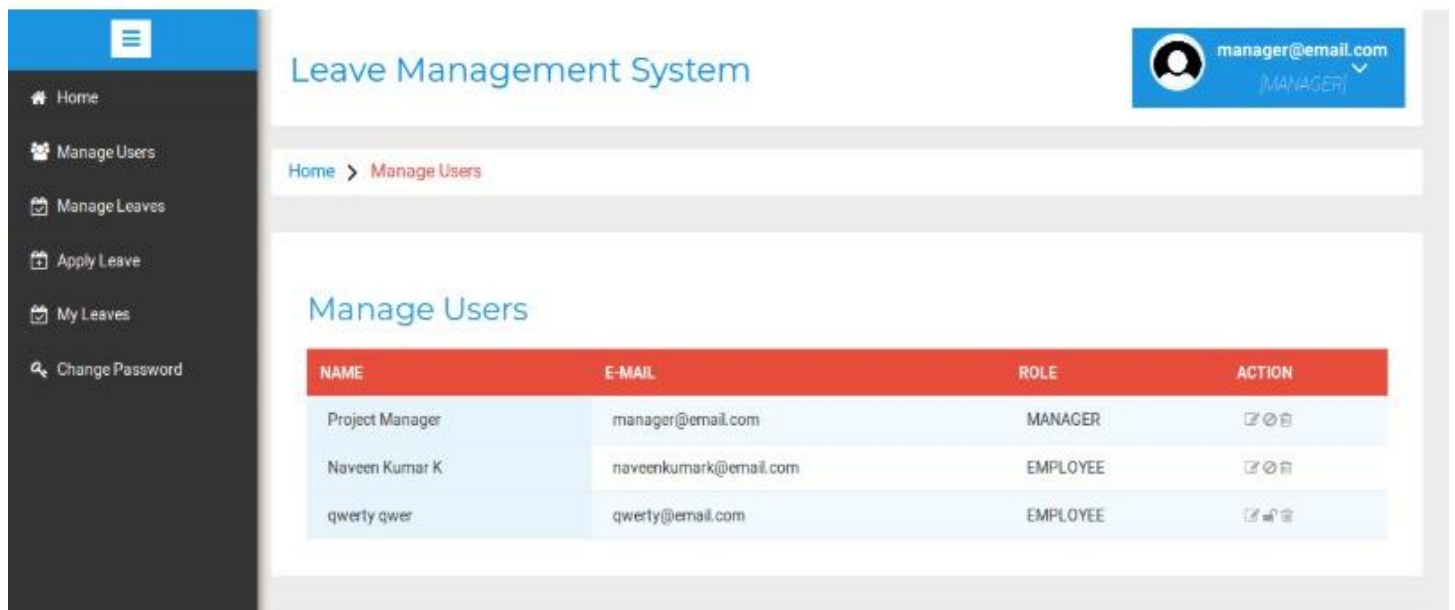
manager@email.com  
[MANAGER]

Home > My Leaves

My Leaves

FROM DATE	TO DATE	NO OF DAYS	LEAVE TYPE	REASON	STATUS
03 Jan 2018	04 Jan 2018	1	CASUAL LEAVE	qwerqwer	ACCEPTED
11 Jan 2018	15 Jan 2018	0	CASUAL LEAVE	just for fun	REJECTED
12 Jan 2018	12 Jan 2018	0	CASUAL LEAVE	Just like that...	PENDING
12 Jan 2018	12 Jan 2018	0	SICK LEAVE	I'm sick	ACCEPTED
12 Jan 2018	12 Jan 2018	0	CASUAL LEAVE	Casual Leave	PENDING
12 Jan 2018	17 Jan 2018	3	CASUAL LEAVE	from 12 jan to 17 jan	REJECTED
15 Jan 2018	19 Jan 2018	3	CASUAL LEAVE	asdasdfasf	REJECTED
01 Jan 2018	09 Jan 2018	3	CASUAL LEAVE	qwerqwer	PENDING

# Manage Users



Result:

The UI was designed successfully.

**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.