RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23432 SOFTWARE CONSTRUCTION LABORATORY

Laboratory Note Book

Name: NIMAL MG

Year / Branch / Section: 2nd YEAR / AIML/AC

University Register No. :2116231501110

College Roll No: 231501110

Semester: 4rd SEMESTER

Academic Year: 2024-2025

RAJALAKSHMI ENGINEERING COLLEGE

[AUTONOMOUS]

RAJALAKSHMI NAGAR, THANDALAM – 602 105

BONAFIDE CERTIFICATE

Name : . NIMAL MG. . . . . .

Academic Year : … 2024-2025…. Semester : 04 Branch : AIML

Register No. **2116231501110**

Certified that this is the bonafide record of work done by the above student in the **CS23432 – SOFTWARE CONSTRUCTION** during the year 2024 - 2025.

Signature of Faculty in-charge

Submitted for the Practical Examination held on . . . . . .

Internal Examiner **External Examiner**

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:

Thus, the study for the given problem statement was successfully completed.

EX NO: 2 WRITING PROBLEM STATEMENT

AIM:

To prepare PROBLEM STATEMENT for your given project.

PROBLEM STATEMENT:

Salary Management System :

Build a full-stack web application that automates the salary management process for an organization. The system should handle salary calculations, tax deductions, salary slip generation, bank transfers, and reporting. It must offer role-based access for HR, Admin, and Employees and ensure secure data storage and retrieval.

1. **Salary Calculation**
   - HR/Admin can input employee details such as working hours, salary per hour, and tax percentage.
   - The system calculates gross, tax, and net salary for each employee on a monthly and annual basis.
   - The system calculates the total monthly and annual salary payout for the entire organization.
   - Overtime hours are added to the base salary based on configured rules and rates.

2. **Tax Calculation**
   - The system computes tax deductions for each employee on a monthly basis.
   - The system calculates the total annual tax payable by the organization.
   - Admin can set or update tax percentages as per government norms.
   - Tax deductions are shown clearly in the employee's salary slip.

3. **Salary Slip & Bank File Generation**
   - Employees can request and download their monthly salary slips.
   - HR can generate bulk salary slips for all employees.
   - The system generates bank files with employee payment details for bulk salary transfers.
   - The system supports email distribution of salary slips.

4. **User Management & Data Storage**
   - Role-based access: Admin, HR, and Employee dashboards.
   - Employees can log in to view their salary history and slips securely.
   - All salary, tax, and employee data is stored securely in a database (e.g., MongoDB).
   - The system includes data backup and restore functionality for salary records.

5. **Reports & Analytics**
   - Admin and Finance teams can generate monthly/annual salary and tax reports.
   - Employees can view/download individual tax deduction reports for filing returns.
   - The system supports CSV and PDF export of reports.

RESULT:

Thus, the problem statement for the given problem is successfully written.

EX NO: 3 DESIGNING PROJECT USING AGILE-SCRUM METHODOLOGY BY USING AZURE.

AIM:

To plan a agile model for the given problem statement.

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 designing project using agile-scrum methodology by using azure was completed successfully.

EX NO: 4 AGILE PLANNING

AIM:

To build a secure, scalable, and user-friendly web/mobile-based SMS/MMS messaging platform that allows users to send, receive, and manage text messages and multimedia messages, create groups, and run SMS campaigns with delivery tracking and notifications

SCOPE:

MVP (Minimum Viable Product) Features:

- User registration and authentication
- Sending and receiving SMS
- Delivery status (Sent, Delivered, Read)
- Push notifications for new messages
- MMS support (images and multimedia)
- Group chat functionality
- SMS Campaign system for admins

AGILE EPICS & USER STORIES:

Epic 1: User Management

- As a user, I want to register and log in securely so I can access messaging features.
- As a user, I want to update my profile and preferences.

Epic 2: SMS/MMS Messaging

- As a user, I want to send SMS messages to other registered users (max 160 chars).
- As a user, I want to receive messages instantly.
- As a user, I want to see message delivery status (Sent, Delivered, Read).
- As a user, I want to send multimedia files (MMS).

Epic 3: Notifications

- As a user, I want push notifications when I receive new messages.

Epic 4: Group Messaging

- As a user, I want to create a group chat with multiple users.

- As a user, I want to add or remove members from a group I created.
- As a user, I want to send messages to the group and ensure all members receive them.

Epic 5: SMS Campaigns

- As an admin, I want to send broadcast SMS to all or selected users.
- As an admin, I want to segment users for targeted campaigns.
- As an admin, I want to track the delivery status of campaign messages.

SPRINT PLANNING:

Sprint 1: Core Setup & Authentication

- Project setup (frontend & backend)
- User registration/login
- Basic dashboard layout
- JWT authentication

Sprint 2: Basic Messaging

- Send/receive SMS (160 chars)
- Real-time message sync (WebSocket/Firebase)
- Basic chat UI
- Store messages in database

Sprint 3: Delivery Tracking & Notifications

- Add message status (Sent, Delivered, Read)
- Push notifications setup
- Message read receipts

Sprint 4: MMS & Multimedia Support

- File/image upload
- Display multimedia in chat
- Optimize MMS handling

Sprint 5: Group Chat Features

- Group creation
- Add/remove participants
- Group messaging flow

Sprint 6: Admin SMS Campaigns

- Admin dashboard
- Create/send campaigns
- User segmentation
- Campaign tracking

FUTURE ENHANCEMENTS:

- **AI-based message categorization or response suggestion**
- **Schedule messages**
- **Integration with email, WhatsApp, or Telegram**
- **Two-way SMS with bots**
- **Analytics dashboard for users and admins**
- **Voice and video message support**

RESULT:

Thus, the agile plan for the problem statement is completed successfully.

EX NO: 5 USER STORIES – CREATION

AIM:

To create User Stories for the given problem statement.

THEORY:

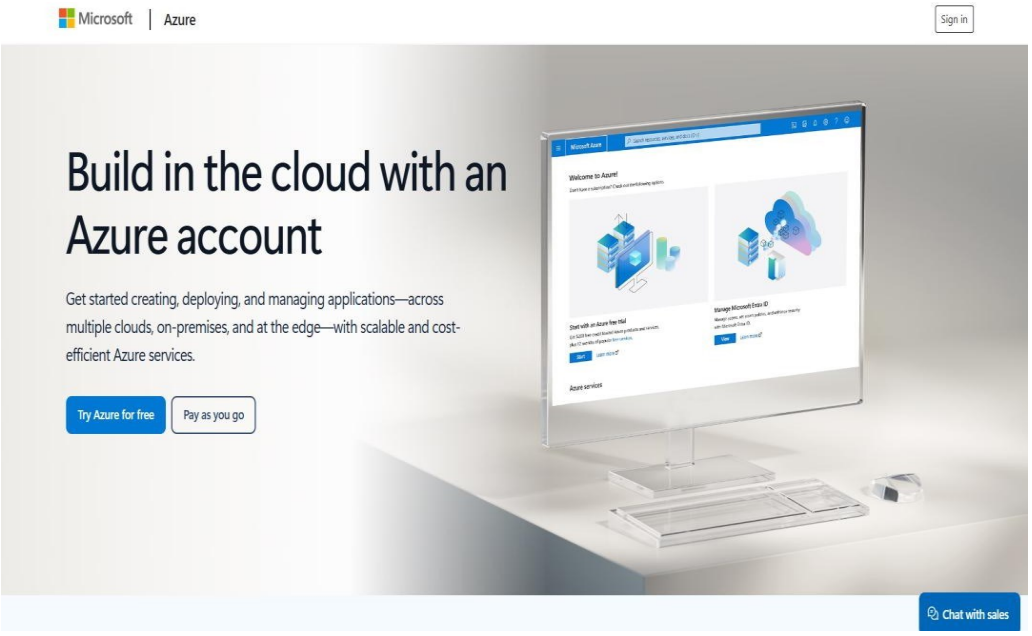
A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.

User story template

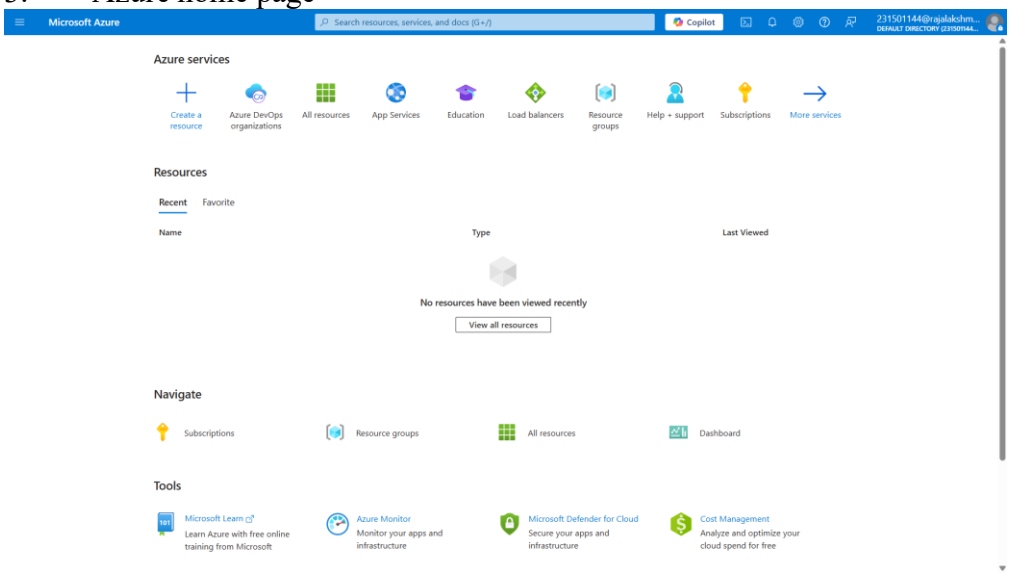
"As a [role], I [want to], [so that]."

PROCEDURE:

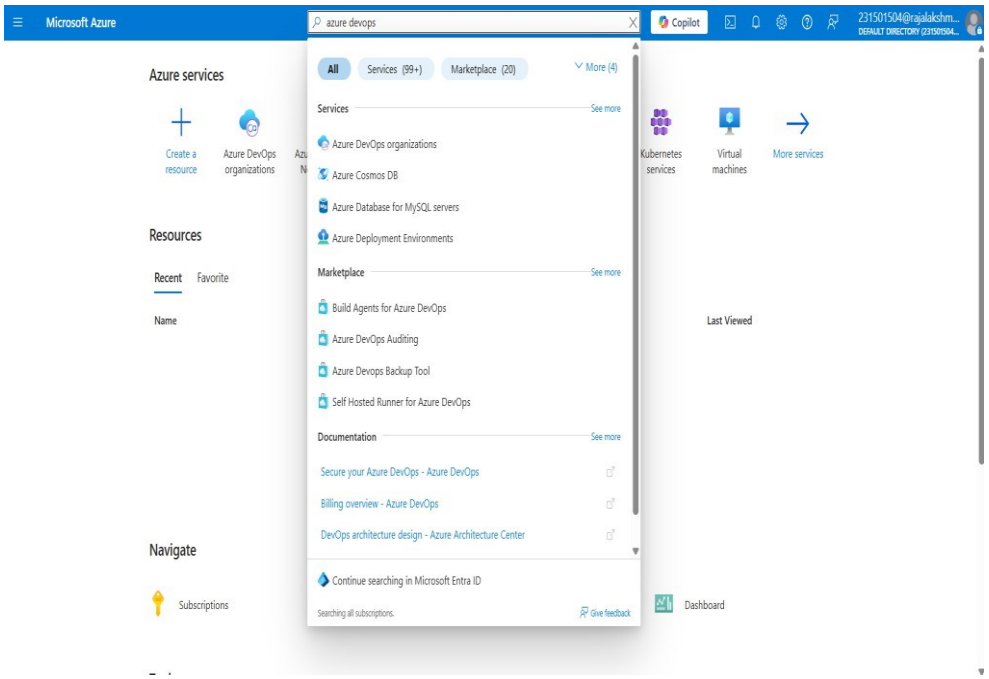
1. Open your web browser and go to the Azure website: *https://azure.microsoft.com/en-in* Sign in using your Microsoft account credentials. If you don't have an account, you'll need to create one.
2. If you don't have a Microsoft account, you can sign up for *https://signup.live.com/?lic=1*



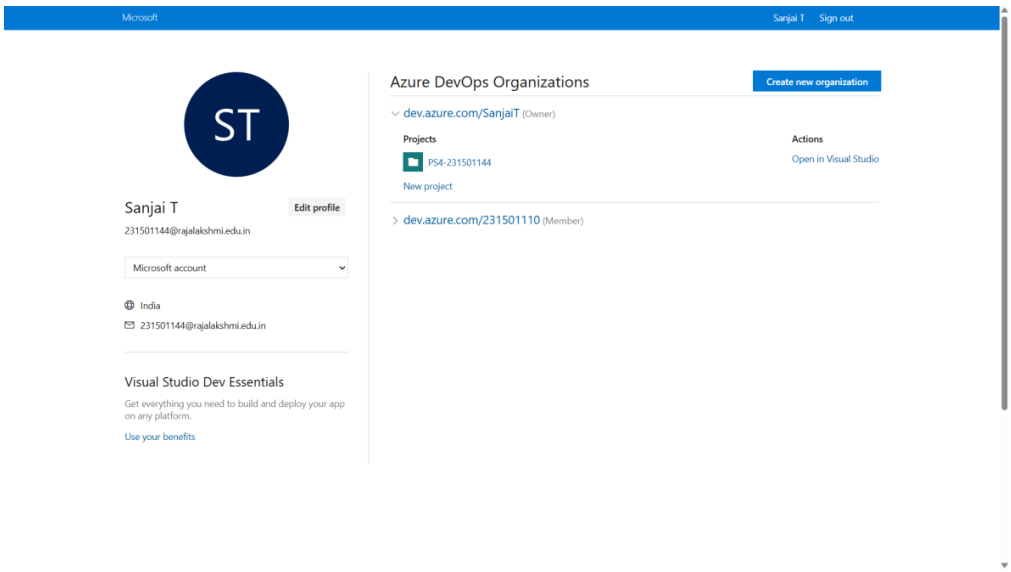
3. Azure home page



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.

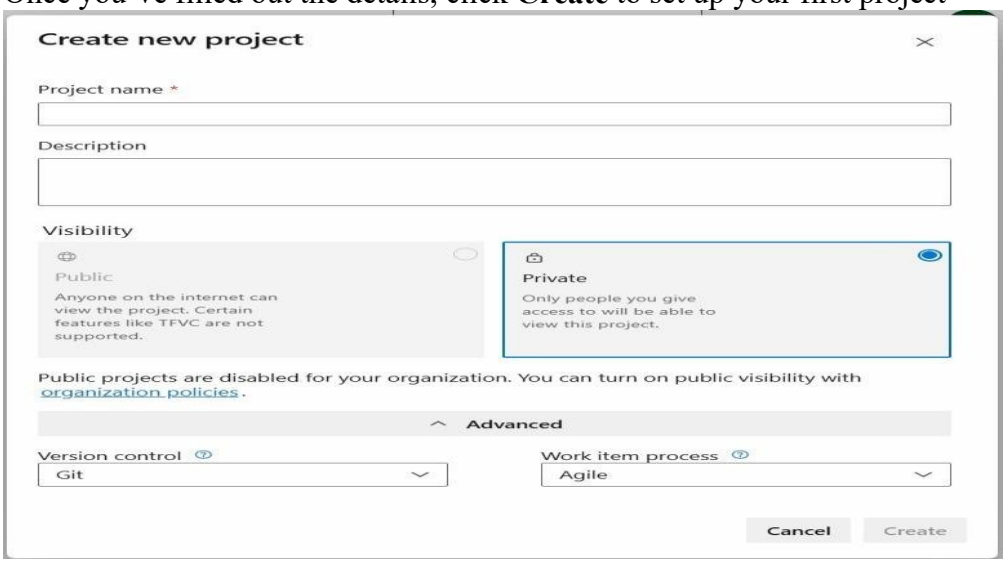


6. Create the First Project in Your Organization

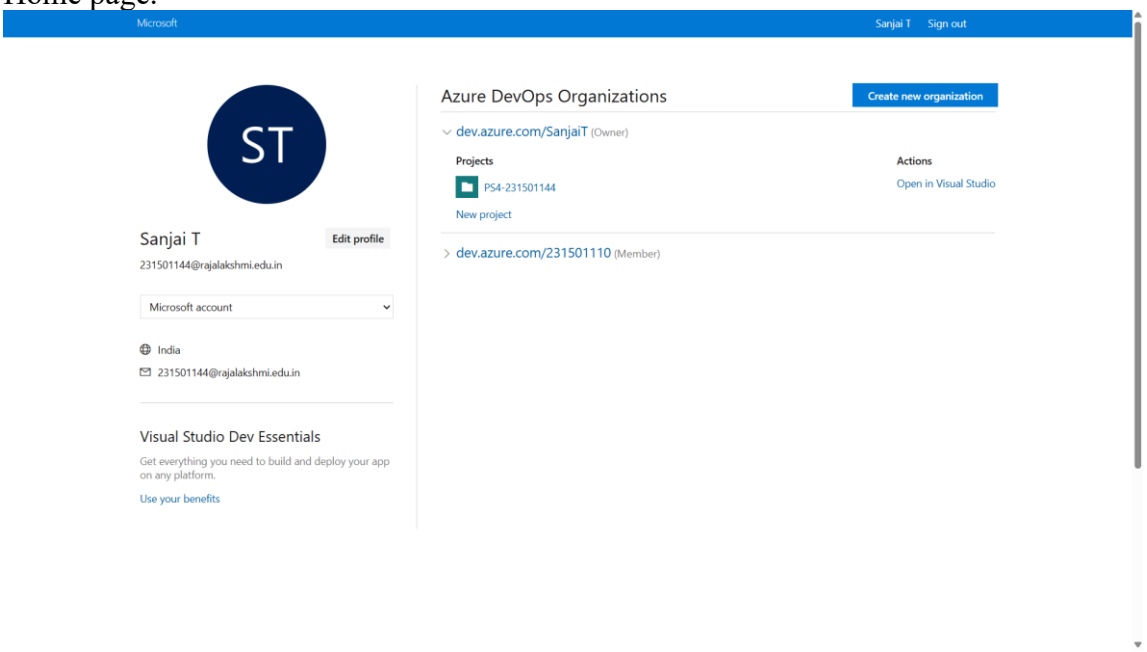
After the organization is set up, you'll need to create your first **project**. This is where you'll begin to manage code, pipelines, work items, and more.

i. On the organization's **Home page**, click on the **New Project** button.
ii. Enter the project name, description, and visibility options:
- **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).

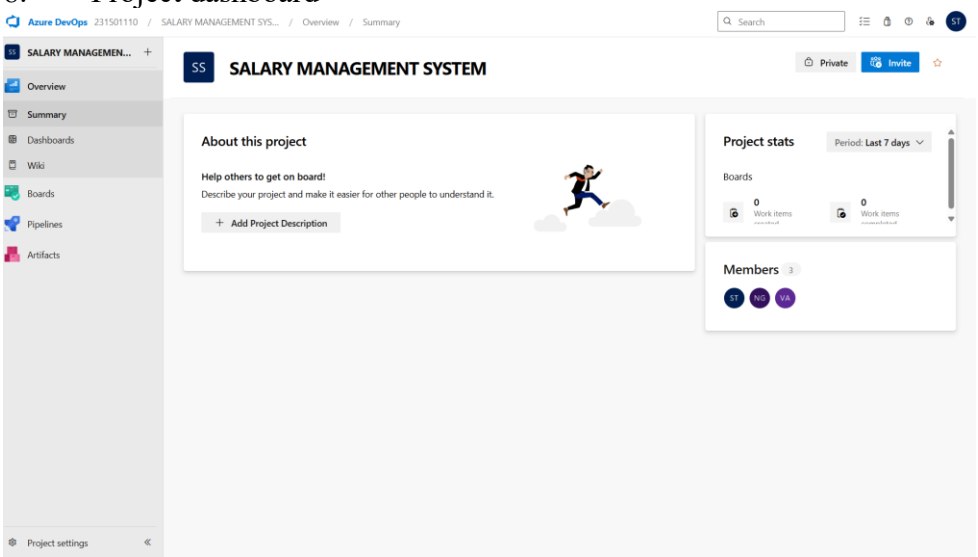
Once you've filled out the details, click **Create** to set up your first project



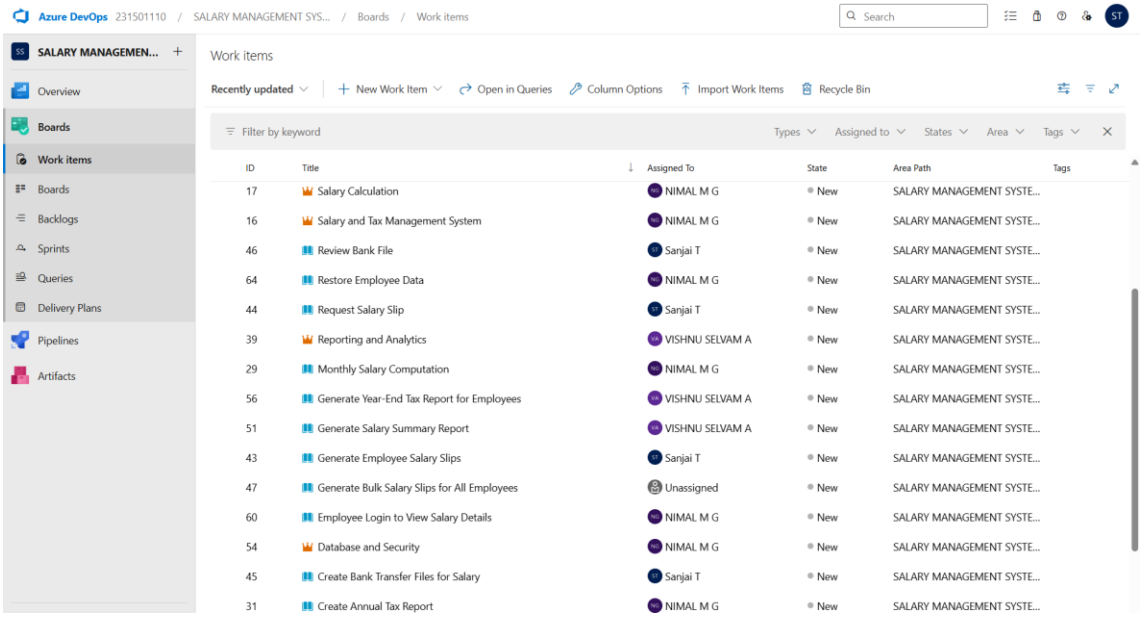
7. Once logged in, ensure you are in the correct organization. If you're part of multiple organizations, you can switch between them from the top left corner (next to your user profile). Click on the Organization name, and you should be taken to the Azure DevOps Organization Home page.



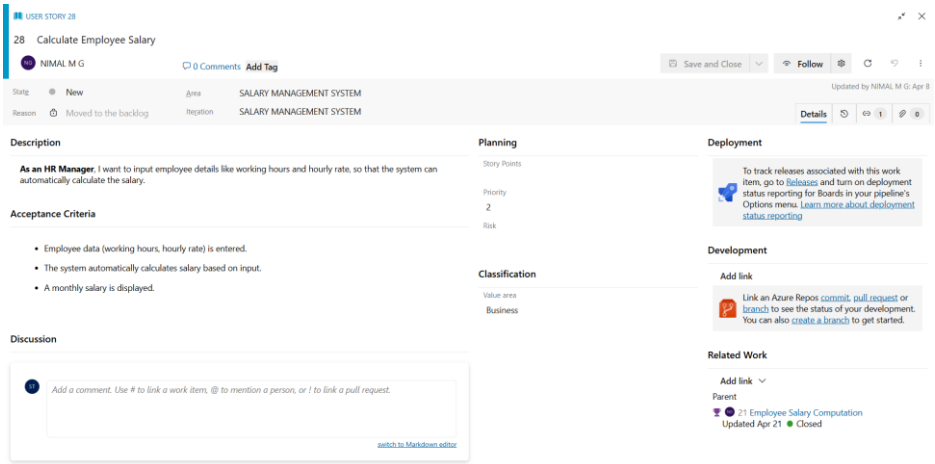
8. Project dashboard



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



RESULT:

The user story for the given problem statement was written successfully.

EX NO: 6 SEQUENCE DIAGRAM

AIM:

To design a Sequence Diagram by using Mermaid.js for the given problem statement.

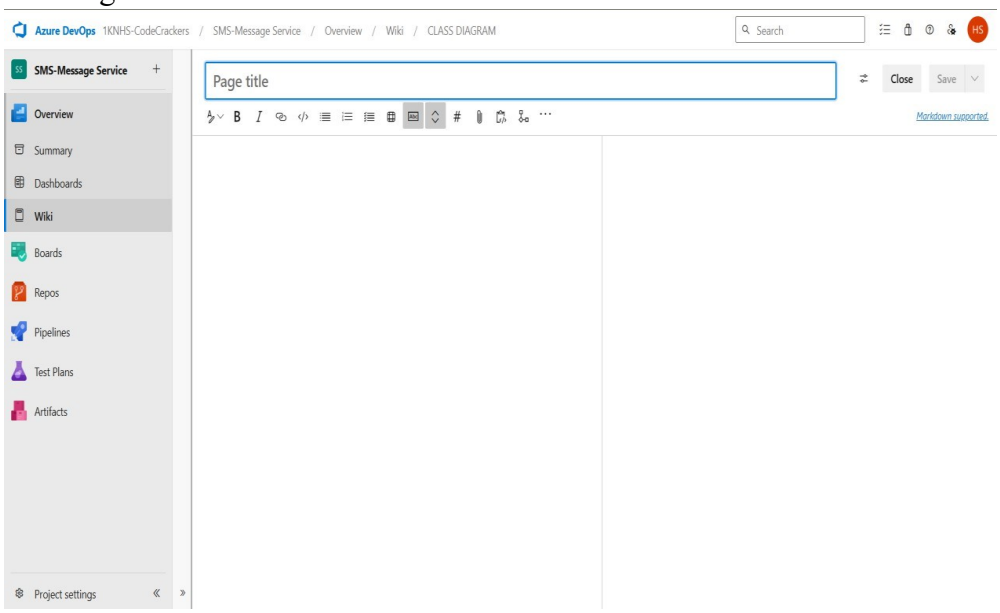
THEORY:

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:

Open a project in Azure DevOps Organisations.

1. To design select wiki from menu



2. Write code for drawing sequence diagram and save the code.::: mermaid sequence Diagram

```
@startuml
actor HR
actor Admin
participant FE as "Frontend"
participant BE as "Backend"
participant FS as "Full Stack"
database DB as "Database"
participant ADO as "Azure DevOps"

HR -> FE : Input hours/rate
FE -> BE : Send input data
BE -> DB : Fetch employee data
DB --> BE : Return employee data
BE -> BE : Calculate Salary
BE -> DB : Save calculated salary
DB --> BE : Confirm save
BE -> FE : Return Salary

HR -> FE : Request bulk salary slips
FE -> BE : Trigger bulk slip generation
BE -> DB : Fetch salary records
DB --> BE : Return records
BE -> BE : Generate slips
BE -> DB : Save slips
DB --> BE : Confirm save
BE -> FE : Notify slips ready
FE -> HR : Display slips

HR -> FE : Confirm Email/Download
```

```
FE -> BE : Trigger Email/Download

Admin -> ADO : Trigger Backup
ADO -> DB : Backup DB
DB --> ADO : Provide backup file
ADO -> FS : Store backup file

@enduml


%% Campaign Message Flow
participant Admin as Administrator
Admin->>System: createCampaign(content, targets)
System->>Receiver: sendCampaignMessage()
Notification->>Receiver: sendPushNotification()
```

EXPLANATION:

Direct Message Flow

This is a one-on-one message from one user to another.

1. **Sender → System:**
   The user (Sender) sends a direct message using sendMessage(to, content).

2. **System → Receiver:**
   The messaging system delivers the message to the recipient (Receiver).

3. **System → Sender:**
   The system sends back an update to the sender confirming the message was **SENT**.

4. **Receiver → System:**
   The receiver acknowledges the message was received.

5. **System → Sender:**
   The system updates the message status to **DELIVERED** for the sender.

6. **Notification → Receiver:**
   A push notification is triggered and sent to the receiver via a Notification Service.

7. **Receiver → System:**
   The receiver reads the message.

8. **System → Sender:**
   The system updates the status to **READ** for the sender to indicate it was seen.

Group Message Flow

This involves sending a message to a group chat.

1. **Sender → System:**
   The user sends a group message using sendGroupMessage(groupID, content).

2. **System → Group:**
   The system identifies the group and distributes the message to all group members.

3. **Group → Receiver:**
   The message is delivered to each individual member of the group.

4. **Notification → Receiver:**
   Each group member receives a push notification from the Notification Service.

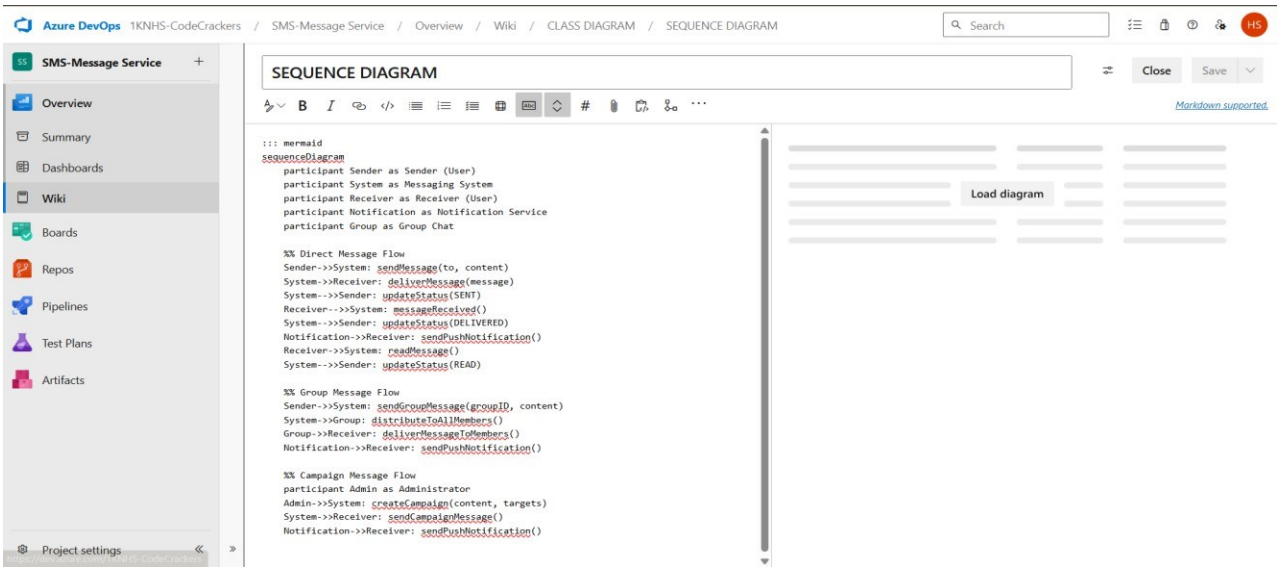
Campaign Message Flow

Used for mass communication, typically by an administrator (e.g., marketing messages or alerts).

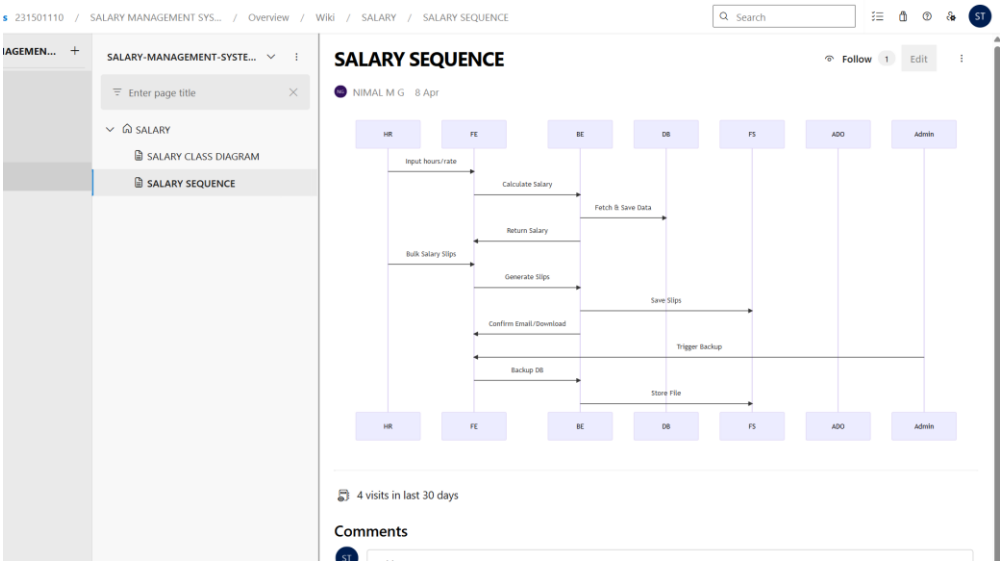
1. **Admin → System:**
   An administrator creates a campaign with content and target recipients using createCampaign(content, targets).

2. **System → Receiver:**
   The system sends the campaign message to each targeted recipient.

3. **Notification → Receiver:**
   A push notification is sent to each recipient.



4. click wiki menu and select the page



RESULT:

Thus, the sequence diagram for the given problem statement was drawn successfully.

EX NO: 7 CLASS DIAGRAM

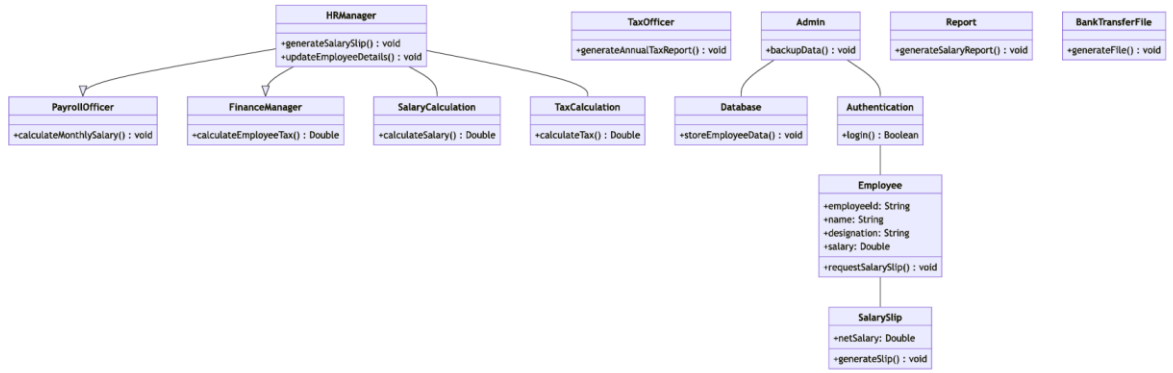
AIM:

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.

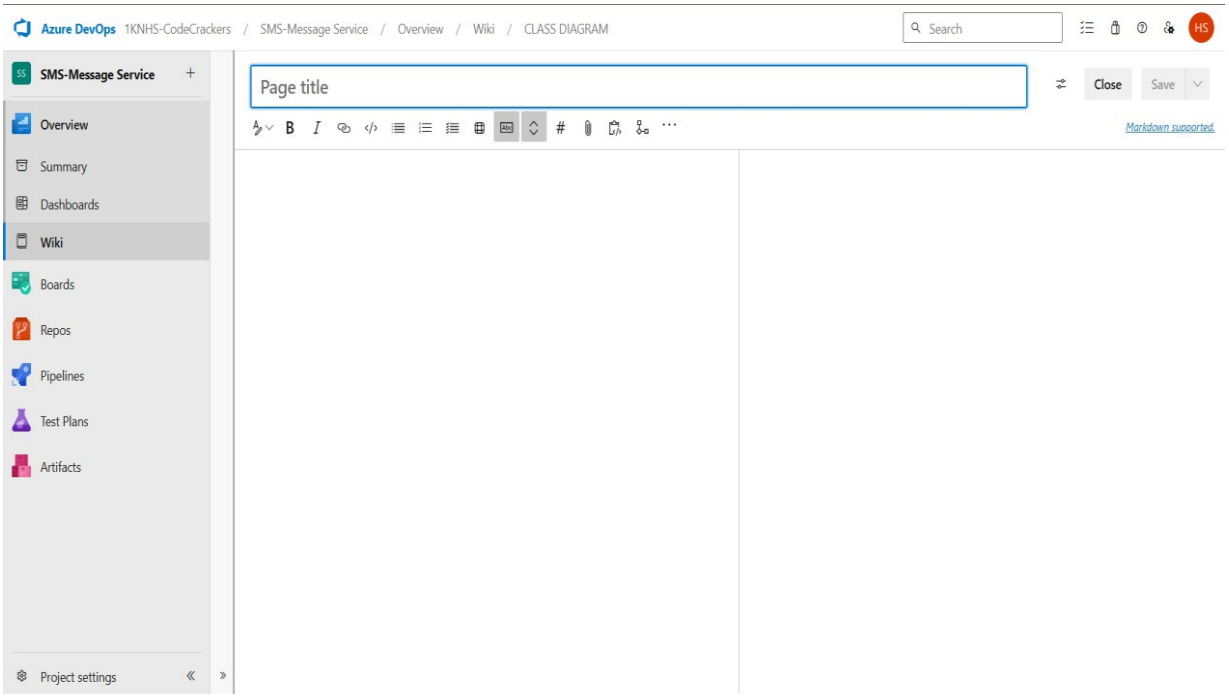
SALARY CLASS DIAGRAM



PROCEDURE:

1. Open a project in Azure DevOps Organisations.

2. To design select wiki from menu



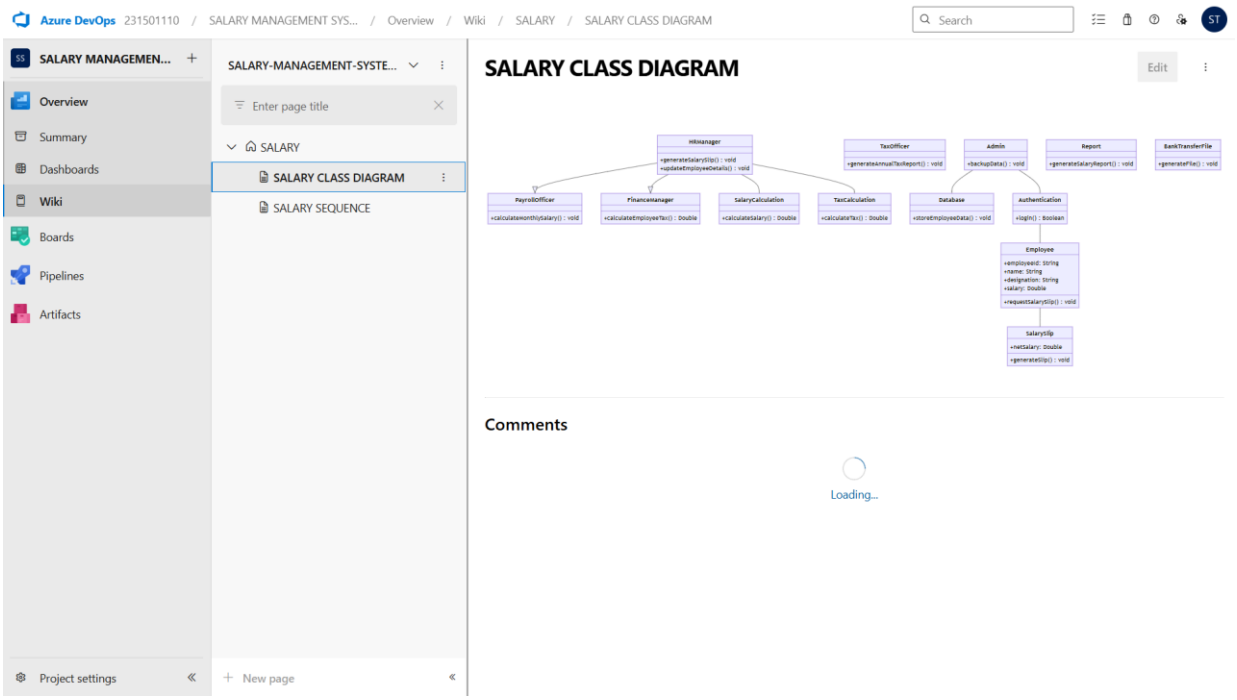
3. Write code for drawing class diagram and save the code

ClassDiagram

::: mermaid

```
classDiagram

class User {
  - userID: String
  - name: String
  - phoneNumber: String
  + sendMessage(to: User, content: String): void
  + receiveMessage(message: Message): void
}

class Message {
  - messageID: String
  - sender: User
  - recipient: User
  - content: String
  - timestamp: DateTime
  - status: MessageStatus
  - type: MessageType
}

class Group {
  - groupID: String
  - groupName: String
  - members: List~User~
  + addMember(user: User): void
  + removeMember(user: User): void
  + sendGroupMessage(message: Message): void
}

class Campaign {
  - campaignID: String
  - content: String
  - targetUsers: List~User~
  + send(): void
}

class NotificationService {
  + sendPushNotification(user: User, message: String): void
}

class MessageStatus {
  <<enum>>
  SENT
  DELIVERED
  READ
```

```
}

class MessageType {
  <<enum>>
  SMS
  MMS
}

User --> Message : sends >
Message --> User : delivered to >
User --> Group : member of >
Group --> Message : contains >
Campaign --> User : targets >
NotificationService --> User : notifies >
```



RESULT:

Thus, the use case diagram for the given problem statement was designed successfully.

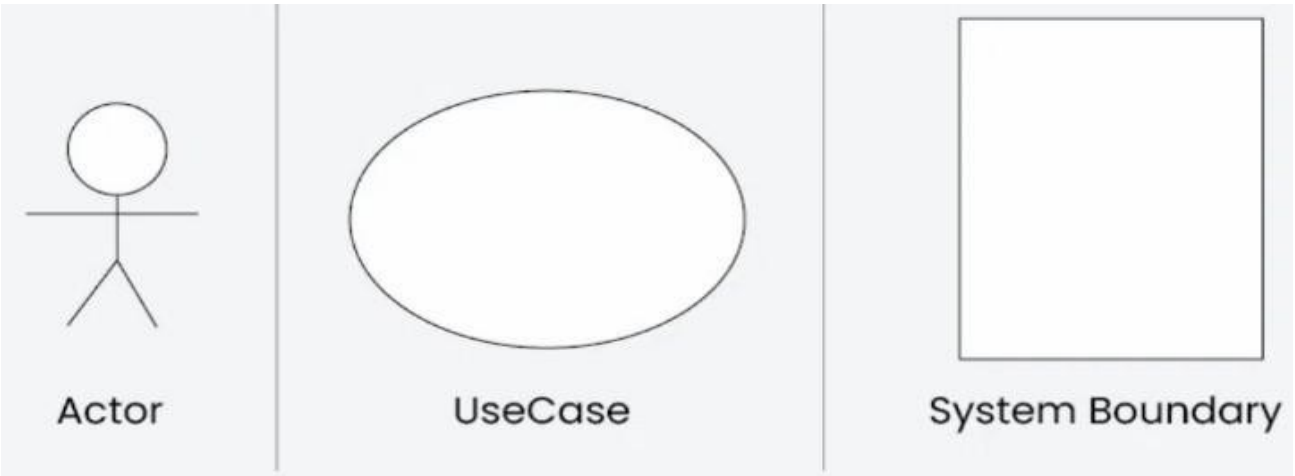
EX NO: 8 USECASE DIAGRAM

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 Diagram.

RESULT:

The use case diagram for the given problem statement was designed successfully.

EX NO: 9 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 o 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 the Azure Wiki

RESULT:

Thus, the Activity diagram for the above problem statement done successfully.

EX NO: 10 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

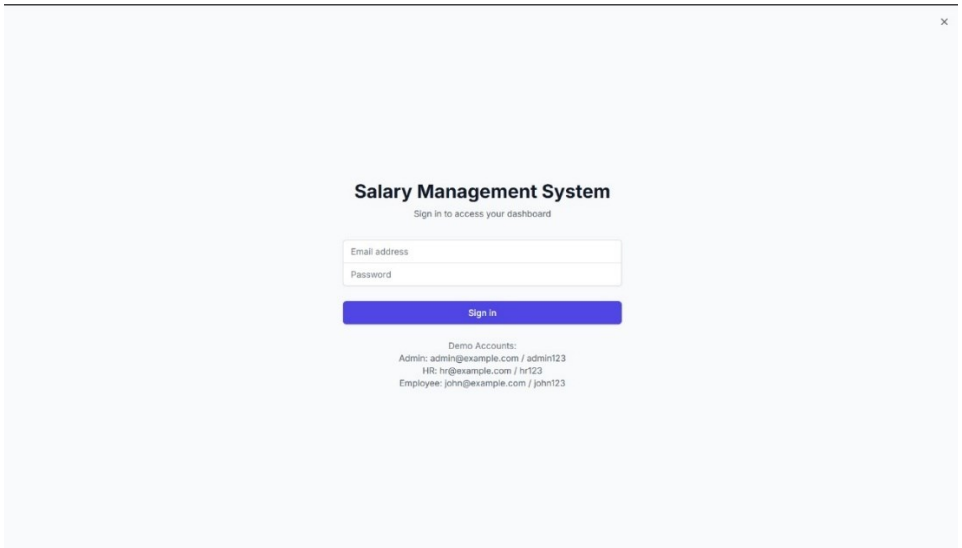
RESULT:

Thus, the architecture diagram for the given problem statement was designed successfully.

EX NO: 11 USER INTERFACE

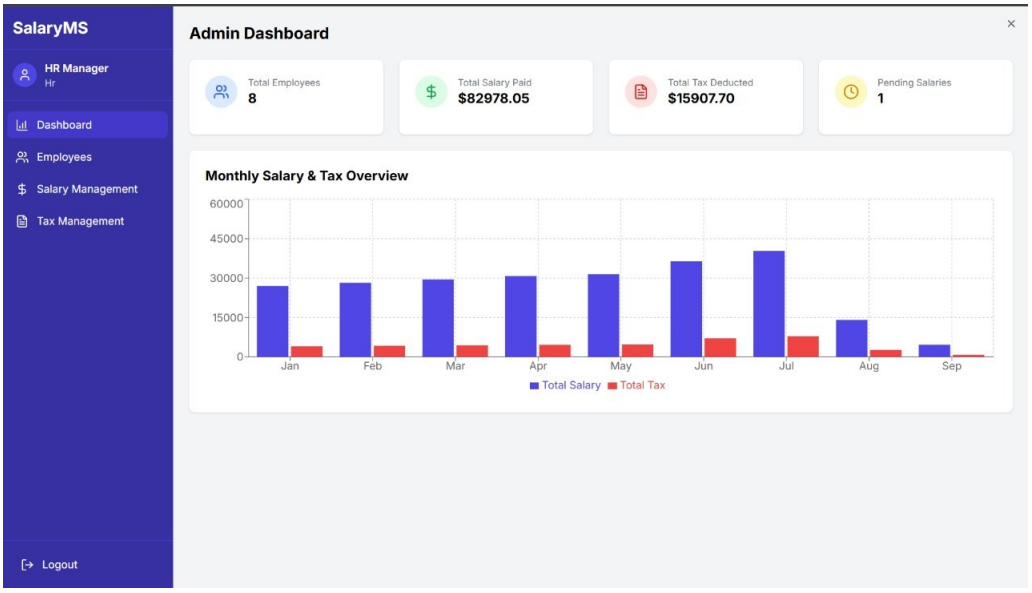
AIM:

Sign In :

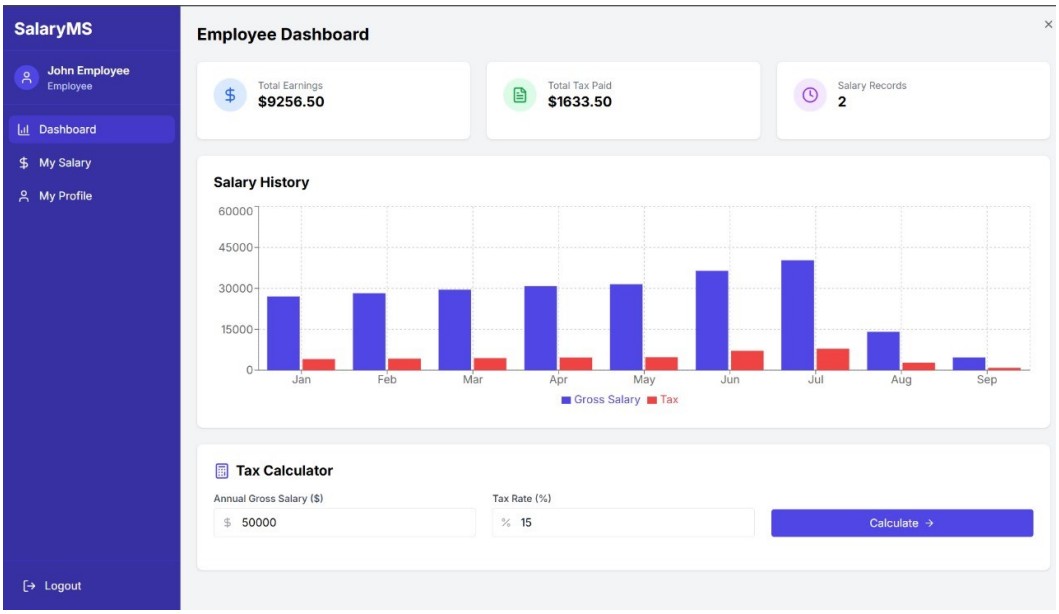


Dashbord:

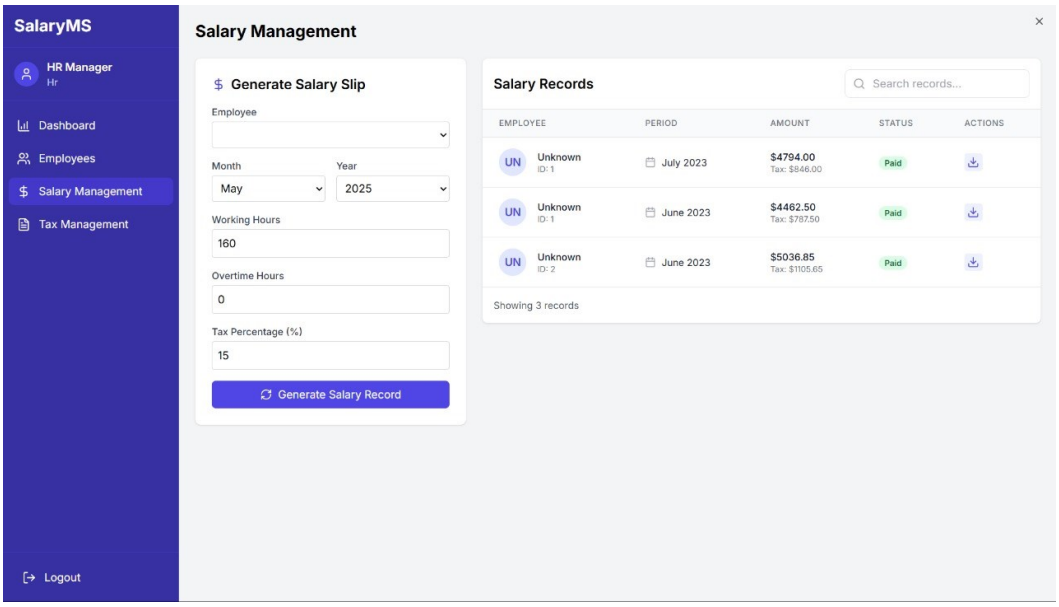
- **Admin**



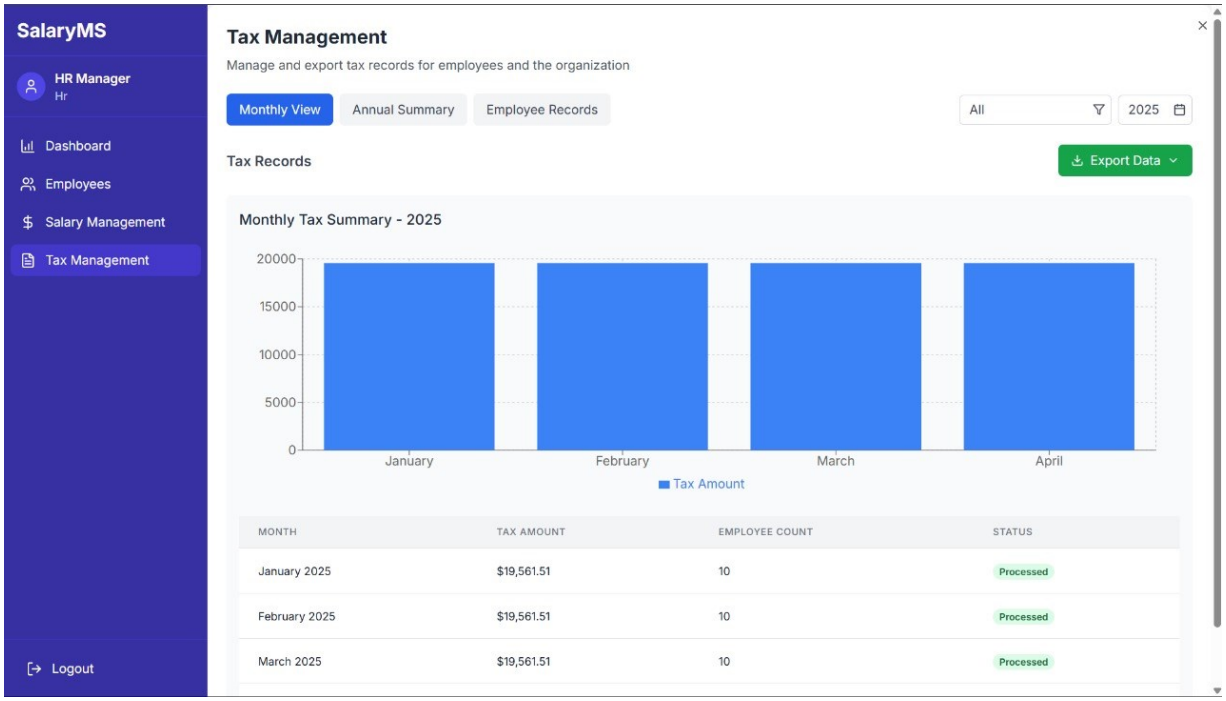
- **Employee**



Salary Management:



Tax Management:



Tax Report:

Tax Management Report

Generated on: 20/05/2025

Name	Tax ID	Salary	Tax Bracket	Annual Tax
Ava Garcia	TX78901	$82,000	22%	$18,039.96
Benjamin Lee	TX67890	$90,000	24%	$21,600
Daniel Kim	TX89012	$79,000	22%	$17,379.96
David Wilson	TX87654	$78,000	22%	$17,160
Emily Johnson	TX78901	$95,000	28%	$26,584.08
Emma Thompson	TX56789	$88,000	24%	$21,120
James Rodriguez	TX45678	$92,000	24%	$22,080
Michael Chen	TX65432	$105,000	32%	$33,584.04
Olivia Taylor	TX34567	$115,000	32%	$36,800.04
Sophia Martinez	TX23456	$85,000	24%	$20,390.04

Monthly Tax Breakdown

Month	Total Tax Amount	Status
January	$19,561.51	10 employees

Month	Total Tax Amount	Status
February	$19,561.51	10 employees

Month	Total Tax Amount	Status
March	$19,561.51	10 employees

Month	Total Tax Amount	Status
April	$19,561.51	10 employees

Salary Slip:

SMS

SalaryMS Inc.
123 Business Avenue, Corporate City
Phone: (555) 123-4567 | Email: payroll@salaryms.com

SALARY SLIP

Period: June 2023
Date of Issue: 20/05/2025

Reference: SAL-1
Status: PAID

Employee Details

Employee Name:	John Doe
Employee ID:	1
Department:	Engineering
Position:	Senior Developer
Bank Name:	ACME Bank
Account Number:	12345678901
IFSC Code:	ACME0001234
Join Date:	15/01/2022

Salary Details

Earnings	Amount
Basic Salary	$4800.00
Overtime Pay	$450.00
Total Earnings	$5250.00

Deductions	Amount
Tax (15%)	$787.50
Total Deductions	$787.50

Net Salary: **$4462.50**

This is a computer-generated document. No signature is required.

RESULT:

Thus, the UI for the given problem statement is completed successfully.

EX NO: 12 IMPLEMENTATIONS

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 cd
- 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 implementation of the given problem statement is done successfully.