RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM – 602 105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025

RAJALAKSHMI
ENGINEERING COLLEGE

CS23432

SOFTWARE CONSTRUCTION

Lab Manual

2024-2025

Name : VIKNESHKUMAR MN

Year/Branch/Section : II /CSE/ D

Register No. :230701382

Semester : IV

Academic Year: 2024-25

INDEX

EX NO : 1 STUDY OF AZURE DEVOPS

AIM:

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

STUDY:

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:

Manual salary processing in organizations is time-consuming, error-prone, and inefficient, especially when dealing with varied pay structures, tax rules, and attendance patterns. HR departments often struggle to accurately track attendance, process payroll, and ensure compliance with tax regulations. Additionally, employees lack transparency in salary breakdowns and leave tracking, leading to confusion and dissatisfaction.
There is a need for a centralized, automated salary management system that integrates attendance, payroll, and tax deductions to improve accuracy, reduce manual effort, and enhance transparency for both HR professionals and employees.

RESULT:

The Problem statement is 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 STORY

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.

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. Go to 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.

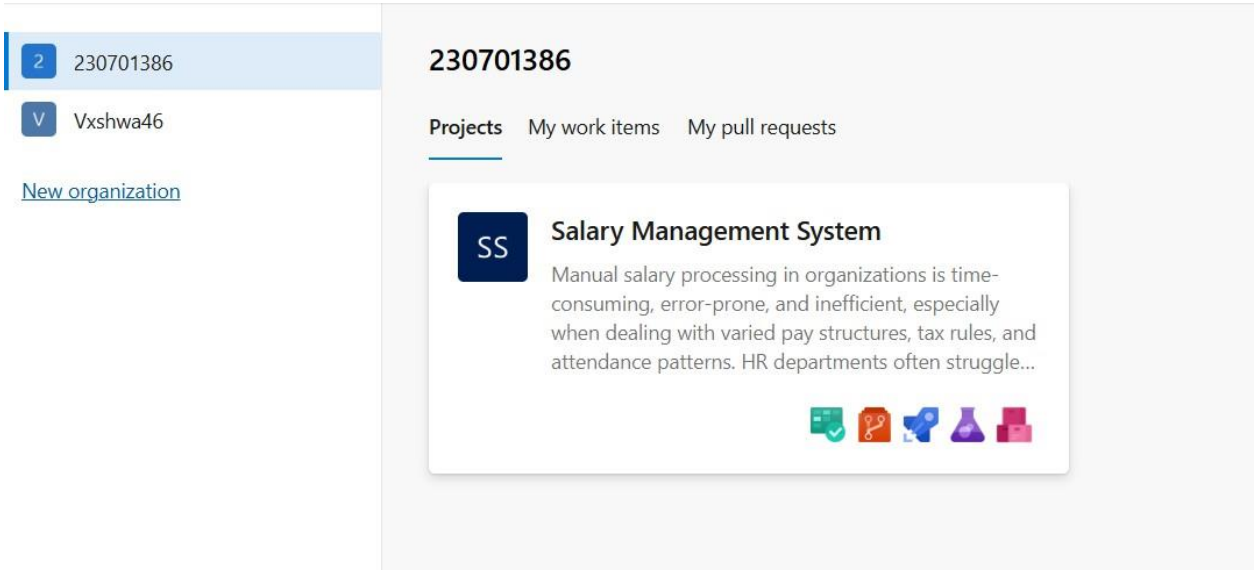
6. Create the First Project in Your Organization. After the organization is set up, you’ll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.

(i) On the organization’s Home page, click on the New Project button.

(ii) Enter the project name, description, and visibility options:

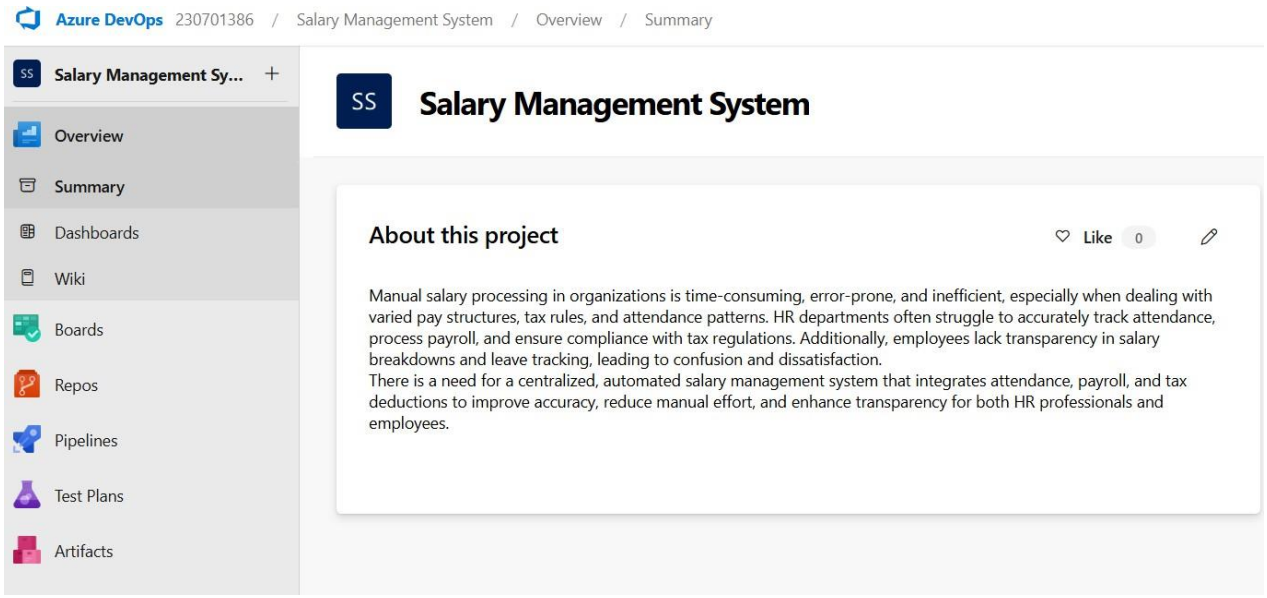
- Name: Choose a name for the project (e.g., LMS).
- Description: Optionally, add a description to provide more context about the project.
- Visibility: Choose whether you want the project to be Private
- (accessible only to those invited) or Public (accessible to anyone).

(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.

8. Open project's dashboard.

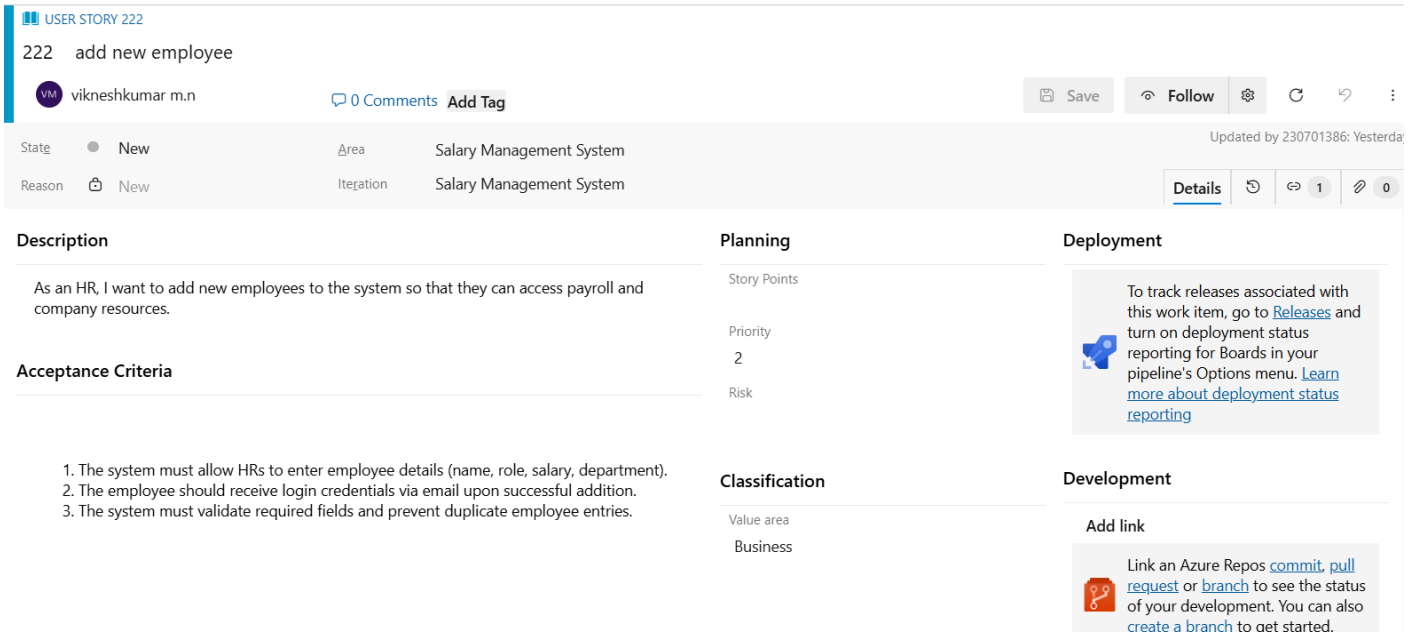


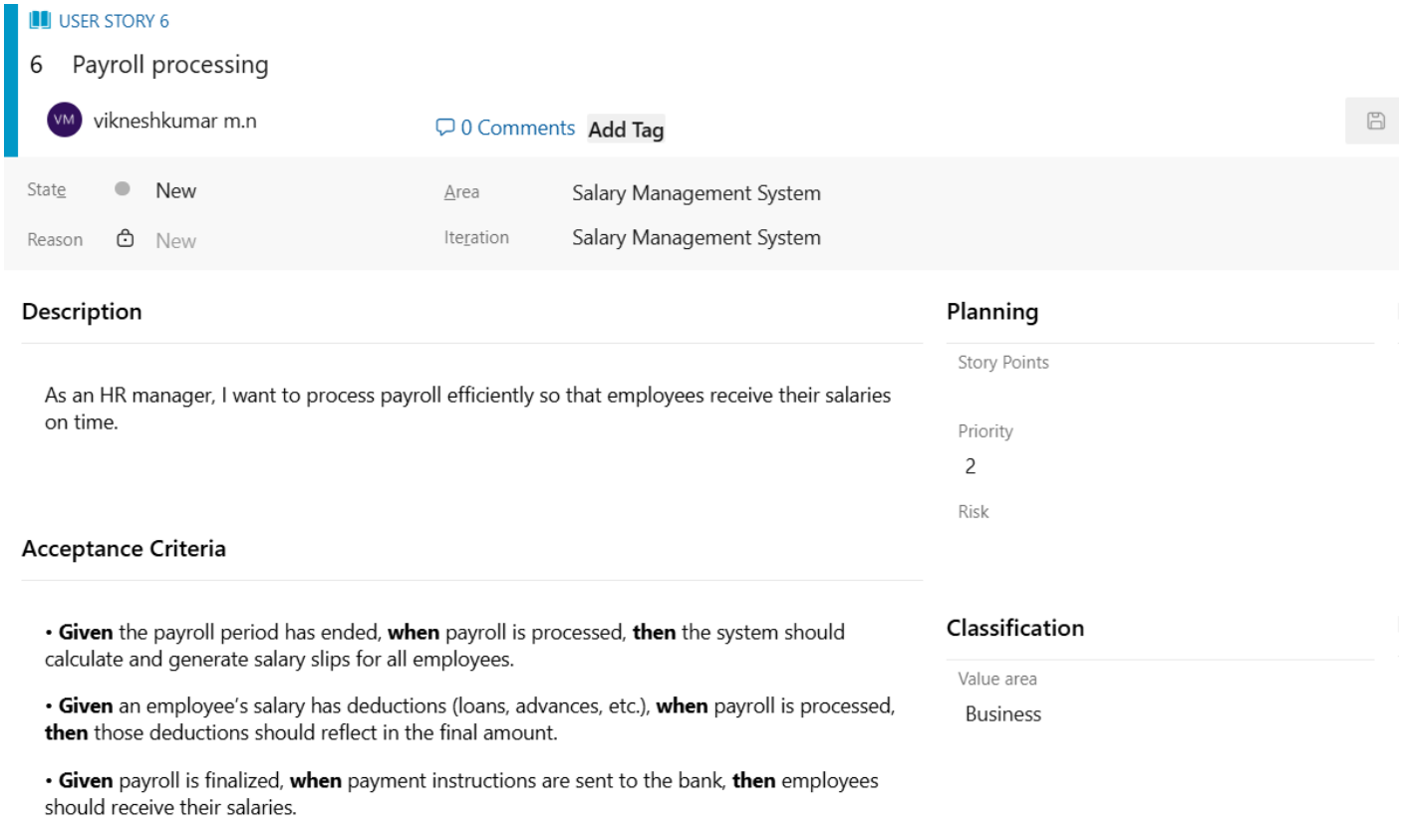
9. To manage user stories:

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 the User Story.





USER STORY 223

223 remove employee

vikneshkumar m.n 0 Comments Add Tag Save Follow

State New Area Salary Management System

Reason New Iteration Salary Management System

Updated by 230701386: 12h

Details

Description

As an HR, I want to remove employees from the system so that I can maintain an accurate employee database.

Acceptance Criteria

1. HRs must be able to deactivate or permanently remove an employee from the system.
2. The system should prompt HRs to confirm the removal before proceeding.
3. Payroll records for removed employees should remain accessible for auditing purposes.

Planning

Story Points

Priority

2

Risk

Classification

Value area

Business

Deployment

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Development

Add link

Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started

Discussion

USER STORY 221

221 salary calculation

vikneshkumar m.n 0 Comments Add Tag

State New Area Salary Management System

Reason New Iteration Salary Management System

Description

As an employee, I want to view my salary breakdown so that I can understand my earnings, deductions, and net pay.

Acceptance Criteria

1. The system must display the employee's gross salary, deductions (tax, benefits), and net salary.
2. The salary breakdown must be updated dynamically based on any changes in tax rates or deductions.

Planning

Story Points

5

Priority

2

Risk

Classification

Value area

Business

Epic: Employee Management

The **Employee Management** epic focuses on maintaining accurate employee records, streamlining HR operations, and ensuring efficient personnel administration. It enables HR teams to manage employee details, track changes, and maintain compliance with company policies.

This epic consists of two major components:

1. Edit employee details – store personal details, job designation, edit details of the employees
2. Role-Based Access (HR & Employee)– assign and modify permissions, access based on user roles(employee/HR)
3. Employee self-service portal - Enables employees to view salary details, tax deductions, and payment history

Key Features & Their Functionalities

1. Edit Employee Details

- Allows HR personnel to update employee profiles, including job designation, department, and salary.
- Ensures audit tracking for all modifications to maintain transparency.
- Supports role-based access, restricting unauthorized changes.
- Sends notifications to employees when their details are updated.
- Integrates with payroll systems to reflect salary adjustments.

2. Add Employees

- Enables HR to create new employee profiles with essential details.
- Supports document uploads for contracts, identification, and tax records.
- Assigns unique employee IDs for tracking.
- Automatically syncs new employees with payroll and attendance systems.
- Sends welcome notifications to onboarded employees.

3. Delete Employees

- Allows HR to remove employee records when necessary.
- Ensures data retention policies for compliance with labor laws.
- Prevents accidental deletions by requiring confirmation steps.
- Automatically updates payroll and attendance records upon deletion.
- Restricts access to former employees while maintaining historical records.

4. **Employee Self-Service Portal**

- Enables employees to view salary details, tax deductions, and payment history.
- Allows employees to update details and tax preferences.
- -Provides access to downloadable payslips and tax documents.

5. **Payslip download**

- Securely access and retrieve salary slips
- transparency in payroll processing
- provides employees with a record of their earnings, deductions

USER STORY 1: Add New Employee

***As an HR, I want to add new employees to the system so that they can access payroll and company resources*.**

Acceptance Criteria:

1. The system shall allow HR to input mandatory employee details including name, email, and date of joining.
2. The system shall validate that the employee email address is in the correct format.
3. The system shall check for duplicate employee records based on email or employee ID.
4. The system shall assign a unique employee ID upon successful addition.
5. The system shall allow HR to select the employee's department and role.
6. The system shall trigger an automated welcome email to the new employee.
7. The system shall provide an option to upload employee documents during onboarding.
8. The system shall assign default access rights to payroll and company resources upon creation.
9. The system shall log the addition of a new employee in the audit trail.
10. The system shall allow HR to review all entered details before submission.
11. The system shall confirm successful employee addition with a confirmation message.
12. The system shall notify the IT department to create system credentials for the new employee.
13. The system shall allow HR to view and edit employee details after creation.

USER STORY 2: Payroll processing

As an HR manager, I want to process payroll efficiently so that employees receive their salaries on time.

Acceptance Criteria:

1. The system shall allow HR to initiate payroll processing for a selected pay period.
2. The system shall automatically calculate gross and net salary based on salary structure.
3. The system shall factor in deductions such as taxes, insurance, and loan repayments.
4. The system shall include attendance and leave data in salary calculations.
5. The system shall validate that all employee data is complete before processing payroll.
6. The system shall generate payslips for each employee upon payroll completion.
7. The system shall support exporting payroll reports in PDF and Excel formats.
8. The system shall notify HR of any discrepancies or errors during payroll processing.
9. The system shall ensure salary credits are scheduled for the specified payment date.
10. The system shall provide a summary dashboard of total payroll costs and liabilities.
11. The system shall maintain an audit log of all payroll processing activities.
12. The system shall allow HR to reprocess payroll if corrections are needed.
13. The system shall provide role-based access to ensure payroll data confidentiality.

USER STORY 3: Remove employee

As an HR, I want to remove employees from the system so that I can maintain an accurate employee database.

Acceptance Criteria:

1. The system shall allow HR to search and select an employee to be removed.

2. The system shall require confirmation before permanently removing an employee.

3. The system shall display a warning about the consequences of deletion (e.g., data loss).

4. The system shall verify that the employee is no longer active or on payroll before removal.

5. The system shall automatically revoke access to all company systems and resources.

6. The system shall allow HR to archive employee data before removal.
7. The system shall update the employee status to "Inactive" before permanent deletion.
8. The system shall prevent deletion if final payroll settlement is pending.
9. The system shall log the removal action in the audit trail with timestamp and HR credentials.
10. The system shall notify relevant departments (e.g., IT, Finance) upon employee removal.
11. The system shall ensure that removed employees do not appear in active employee lists.
12. The system shall retain historical payroll and performance data for removed employees.
13. The system shall allow HR to generate a report of all removed employees with reasons.

USER STORY 4: Salary calculation

As an employee, I want to view my salary breakdown so that I can understand my earnings, deductions, and net pay.

Acceptance Criteria:

1. The system shall allow employees to securely log in to access their salary details.
2. The system shall display a clear breakdown of basic salary, allowances, and bonuses.
3. The system shall itemize all deductions, including tax, insurance, and loans.
4. The system shall calculate and display the net pay for the selected month.
5. The system shall allow employees to select and view salary breakdowns by month or year.
6. The system shall provide downloadable payslips in PDF format.
7. The system shall display salary breakdowns in a user-friendly and readable layout.
8. The system shall highlight any changes in salary compared to the previous month.
9. The system shall show tax-saving investments or exemptions applied (if any).
10. The system shall provide tooltips or explanations for each component of the salary.

11. The system shall maintain historical salary data for at least the last 3 years.
12. The system shall ensure that only the logged-in employee can view their own data.
13. The system shall be accessible from both desktop and mobile devices.

RESULT:

The assigned user story for my project has been 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 Employee

participant Admin

participant HR

participant Payroll System

participant database

participant Tax department

Employee ->> Admin : login request

Tax department ->> Payroll System : sends new tax rules

Admin -->> Employee : login access

Admin ->> Payroll System : initiates salary calculation

Payroll System ->> Tax department : handle taxes

HR ->> Payroll System : updates salary, employee details

Tax department -->> Payroll System : return tax info

Payroll System -->> HR : confirms updates

Payroll System -->> Admin : notifies payroll completion

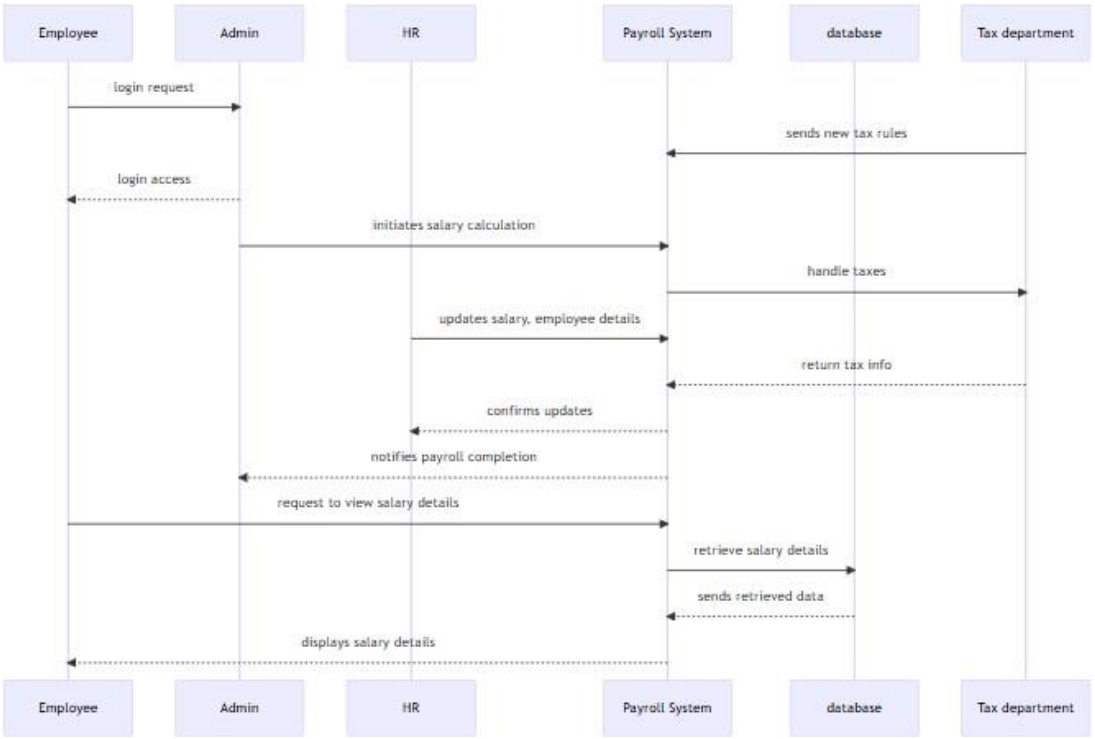
Employee ->> Payroll System : request to view salary details

Payroll System ->> database : retrieve salary details

database -->> Payroll System : sends retrieved data

Payroll System -->> Employee : displays salary details

4. Click wiki menu and select the page.



RESULT:

The sequence diagram is drawn successfully.

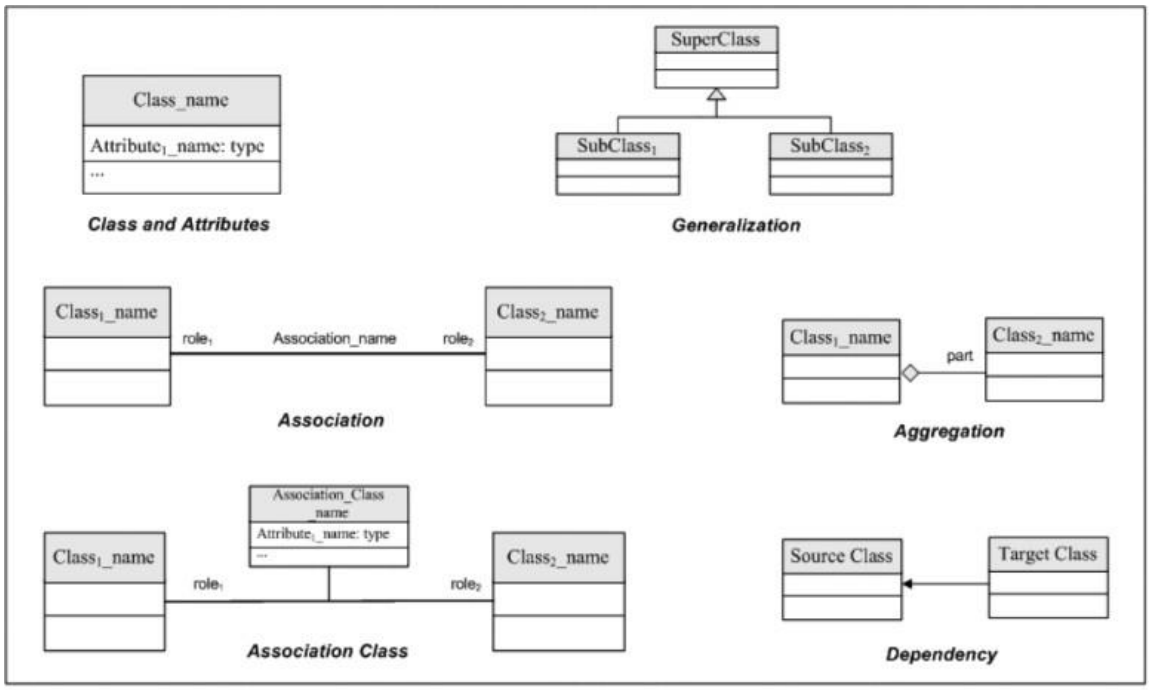
EX NO: 6 CLASS DIAGRAM

AIM:

To draw a simple class diagram.

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.

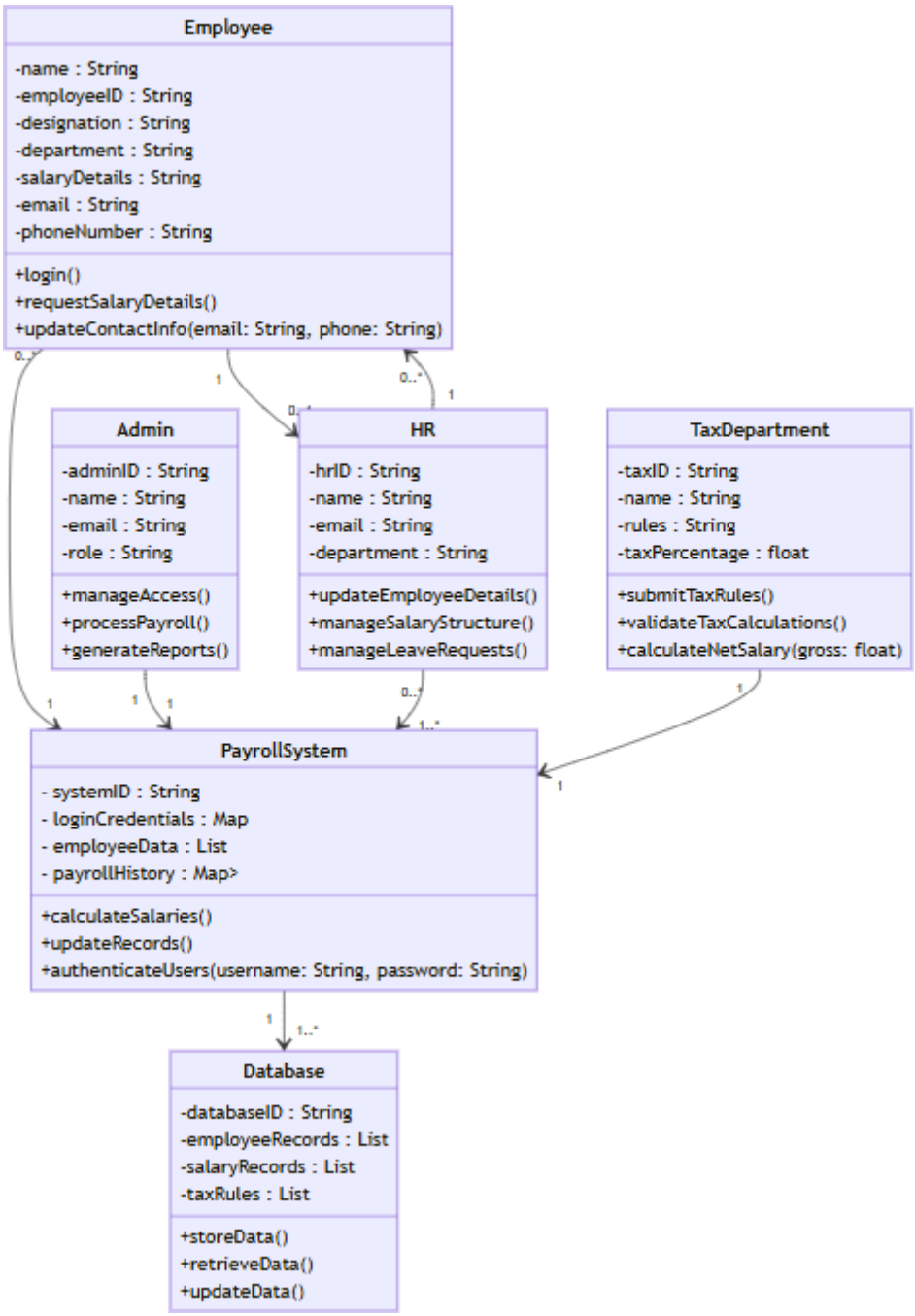
2. To design select wiki from menu.

3. Write the code for drawing Class Diagram and save the code.

```
:::mermaid
classDiagram
    class Employee {
       -name : String
       -employeeID : String
       -designation : String
       -department : String
```

```
    -salaryDetails : String
    -email : String
    -phoneNumber : String
    +login()
    +requestSalaryDetails()
    +updateContactInfo(email: String, phone: String)
}

class Admin {
    -adminID : String
    -name : String
    -email : String
    -role : String
    +manageAccess()
    +processPayroll()
    +generateReports()
}

class HR {
    -hrID : String
    -name : String
    -email : String
    -department : String
    +updateEmployeeDetails()
    +manageSalaryStructure()
    +manageLeaveRequests()
}

class TaxDepartment {
    -taxID : String
    -name : String
    -rules : String
    -taxPercentage : float
    +submitTaxRules()
    +validateTaxCalculations()
    +calculateNetSalary(gross: float)
}

class PayrollSystem {
- systemID : String
- loginCredentials : Map<String, String>
- employeeData : List<Employee>
- payrollHistory : Map<String, List<Salary>>
+ calculateSalaries()
+ updateRecords()
```

```
    + authenticateUsers(username: String, password: String)
}

    class Database {
        -databaseID : String
        -employeeRecords : List
        -salaryRecords : List
        -taxRules : List
        +storeData()
        +retrieveData()
        +updateData()
    }

     %% Each employee is associated with one HR (e.g., for leave/tax/salary queries)
    Employee "1" --> "0..1" HR

    %% Payroll system handles all employees
    Employee "0..*" --> "1" PayrollSystem

    %% Admin manages employees through the payroll system
    Admin "1" --> "1" PayrollSystem

    %% One HR handles many employees
    HR "1" --> "0..*" Employee

    %% HR interacts with PayrollSystem for multiple operations
    HR "0..*" --> "1..*" PayrollSystem


    PayrollSystem "1" --> "1..*" Database

    %% TaxDepartment supplies tax rules to the PayrollSystem
    TaxDepartment "1" --> "1" PayrollSystem
```



RESULT:

Thus the class diagram has been designed successfully.

EX NO: 7 USE CASE DIAGRAM

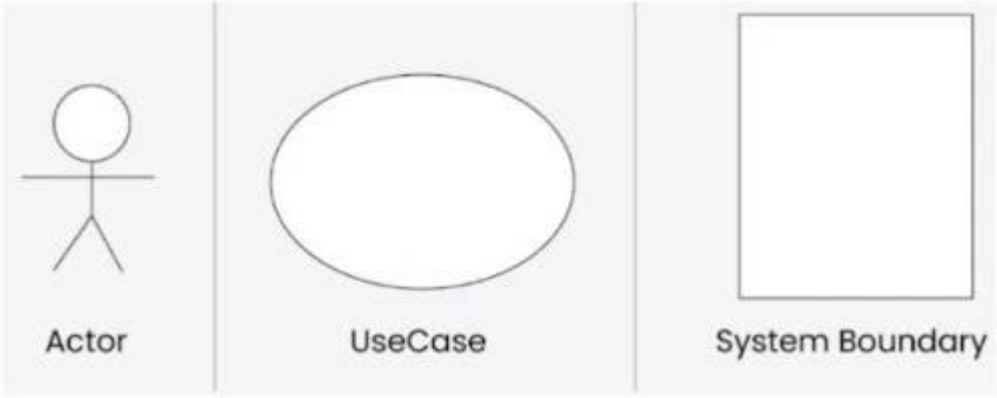
AIM:

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



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.

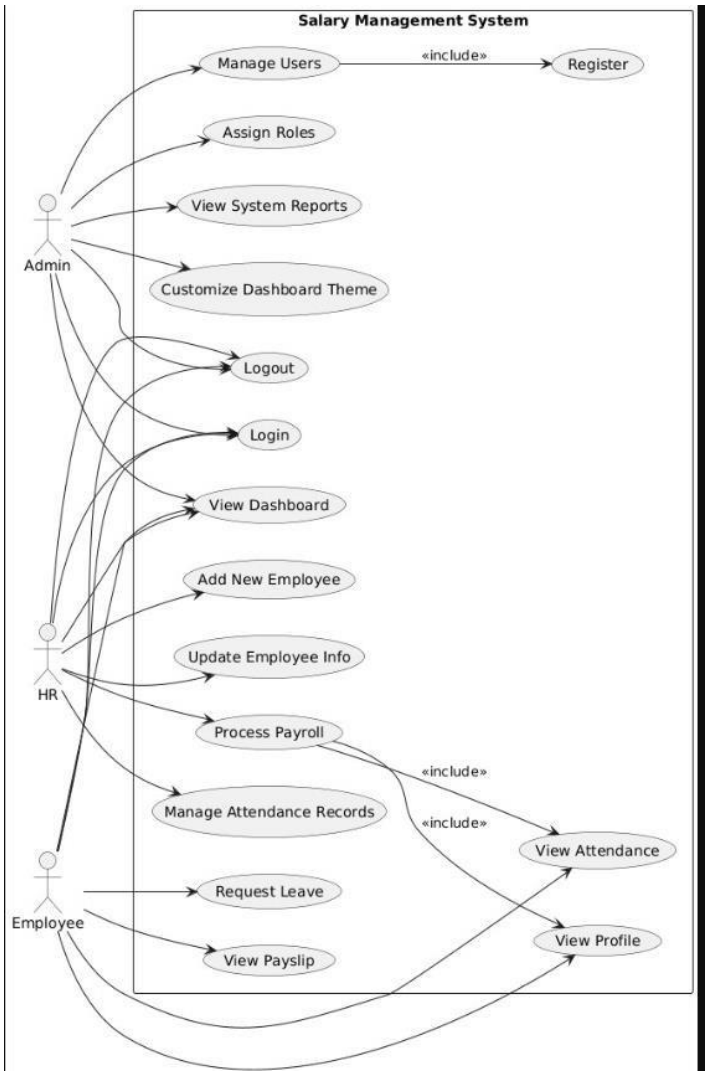
Step 2: Upload the Diagram to Azure DevOps

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.

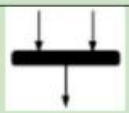
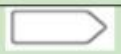
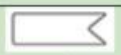
EX NO: 8 ACTIVITY DIAGRAM

AIM :

To draw a sample activity diagram for the Salary Management System.

THEORY:

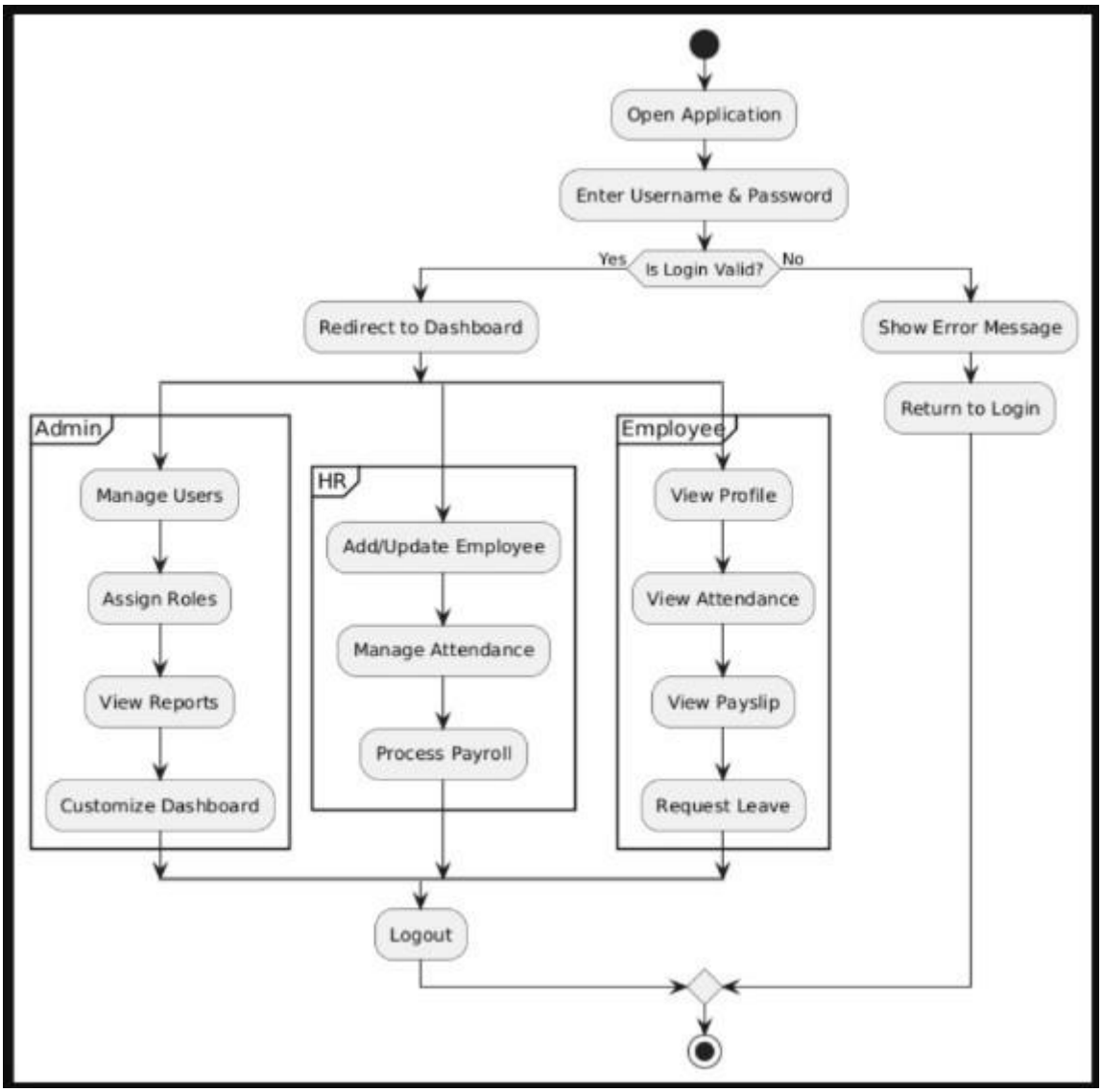
Activity diagrams are an essential part of the Unified Modelling Language (UML) that help visualize workflows, processes, or activities within a system. They depict how different actions are connected and how a system moves from one state to another.

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:

Step 1. Draw diagram in draw.io.

Step 2. Upload the diagram in Azure DevOps wiki.



RESULT:

The activity diagram was designed successfully.

EX NO: 9 ARCHITECTURE DIAGRAM

AIM:

To draw the Architecture Diagram using draw.io.

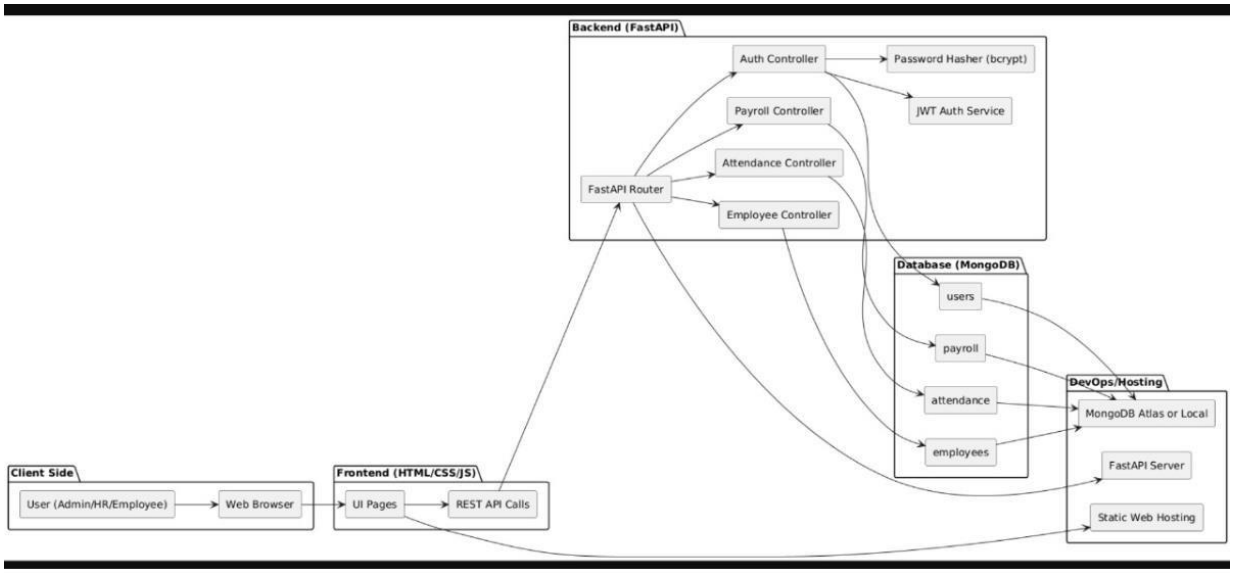
THEORY:

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:

Step 1. Draw diagram in draw.io

Step 2. Upload the diagram in Azure DevOps wiki.



RESULT:

The architecture diagram was designed successfully

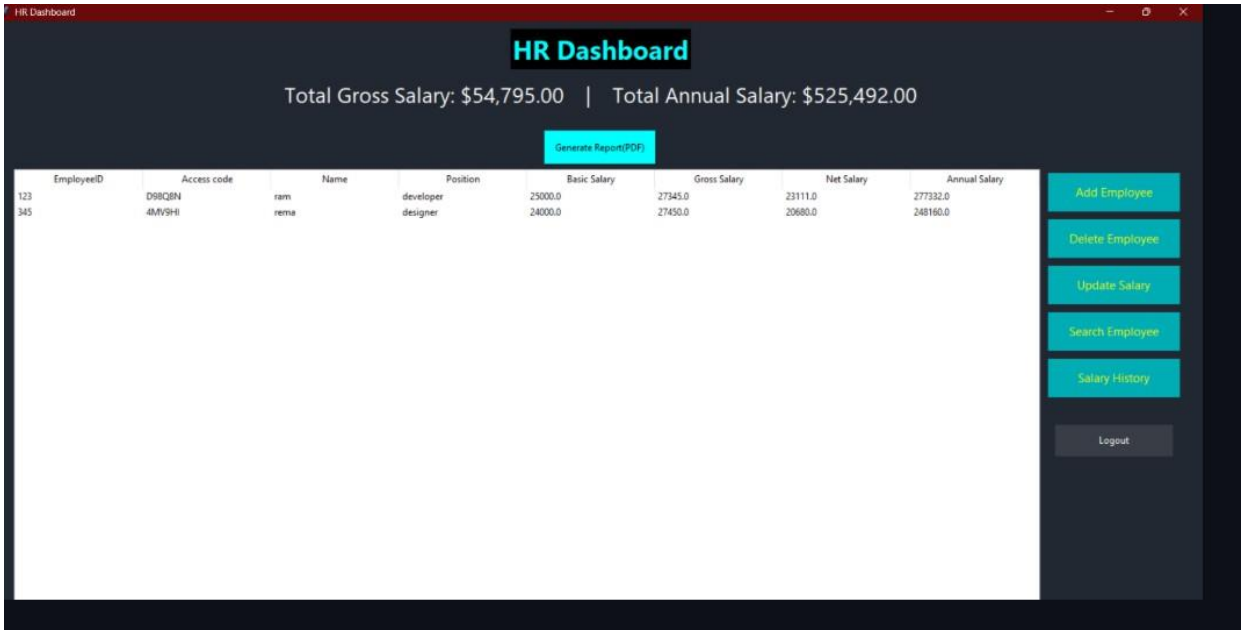
EX NO: 10 USER INTERFACE

AIM:

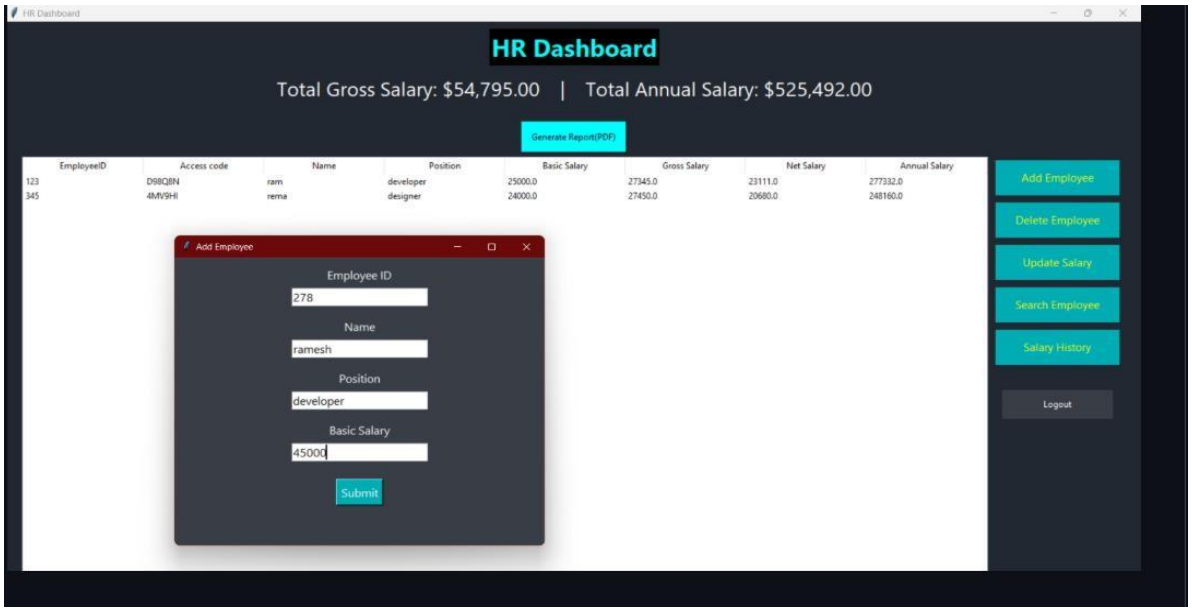
Design User Interface for the Salary Management System.

UI DESIGNS OF SALARY MANAGEMENT APPLICATION:

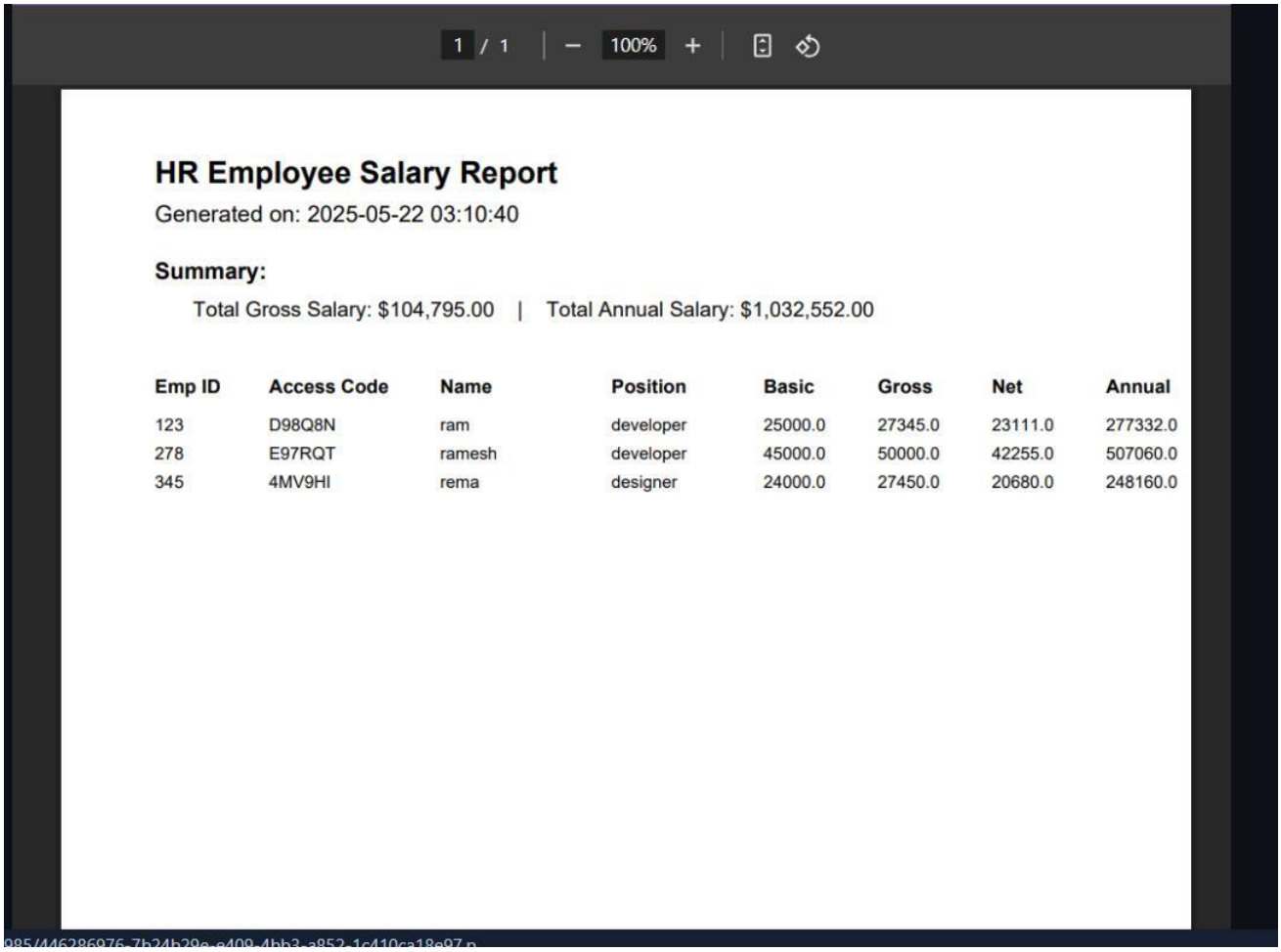
1. HR Dashboard



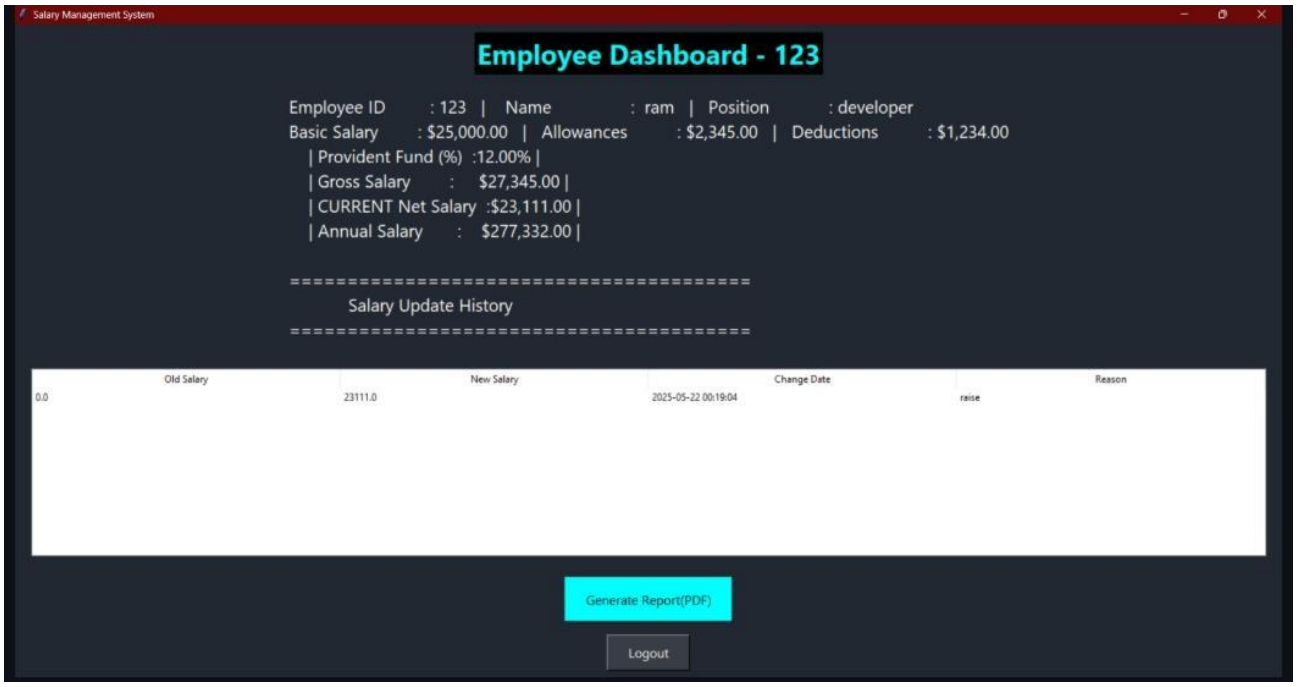
2. Add Employee Feature



3. Employee Details report generated as PDF



4. Employee Dashboard



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>

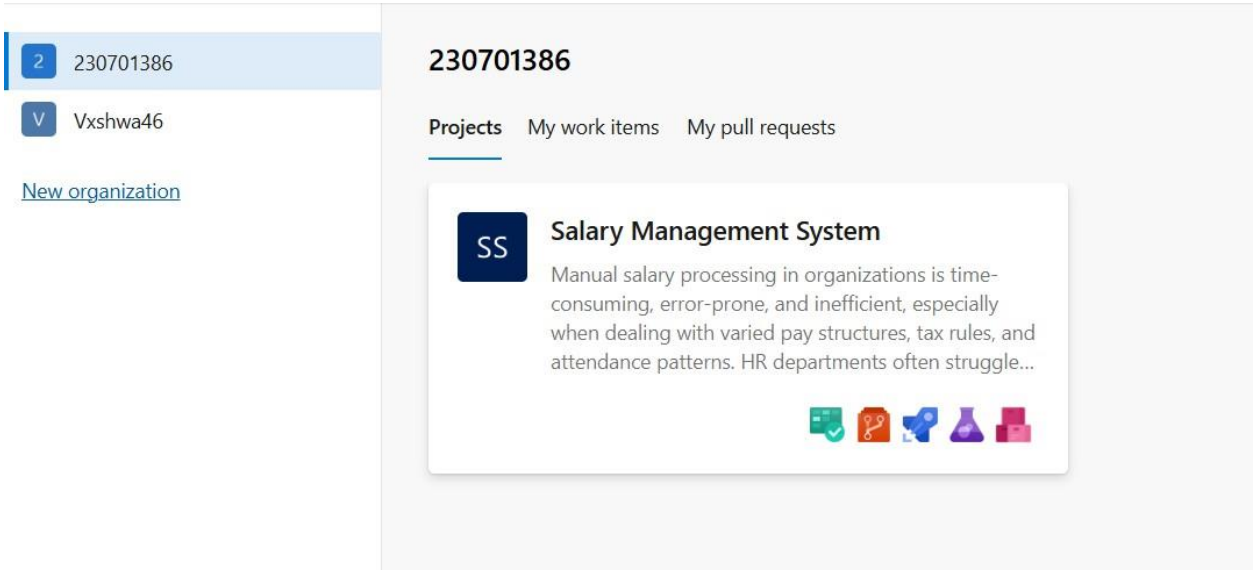
- Add web application code (HTML, CSS, JavaScript, React, Angular, or backend like Node.js, .NET, Python, etc.).
- Commit & push:

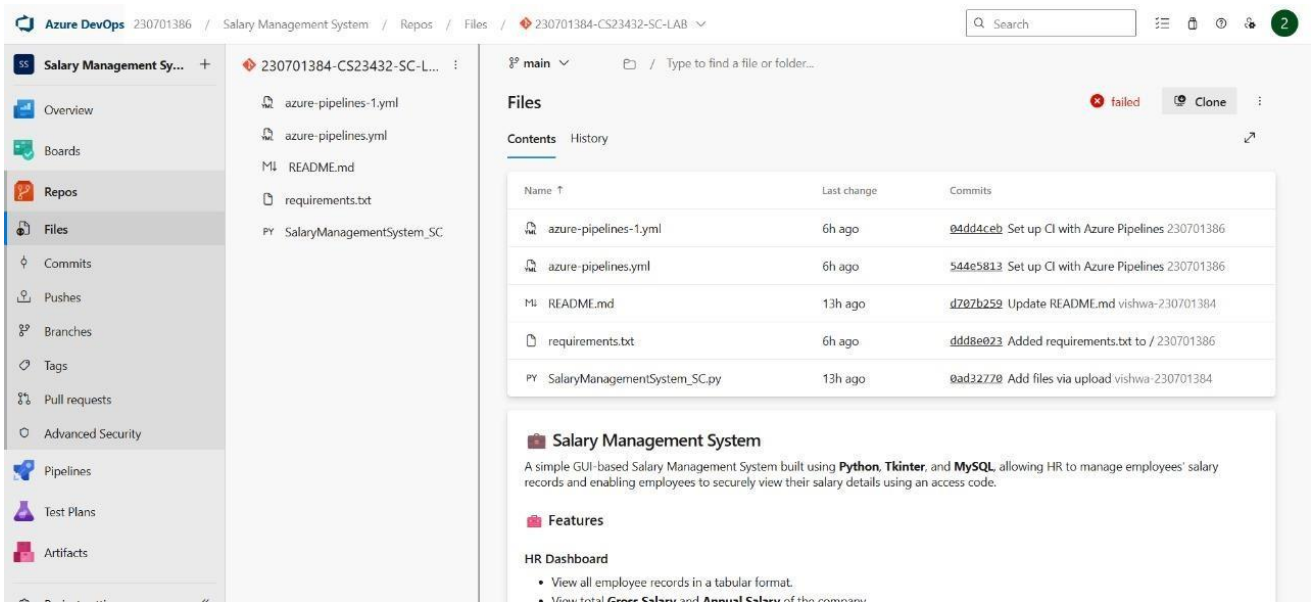
  git add .

  git commit -m "Initial commit"

  git push origin main

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





RESULT :

Thus the application was successfully implemented.

EX NO: 12 TESTING USING AZURE

AIM:

To perform testing of the Salary Management System project using Azure DevOps Test Plans, ensuring that all user stories meet their acceptance criteria as defined in Agile methodology.

PROCEDURE:

Step 1: Open Azure DevOps Project

- Log in to your Azure DevOps account.
- Open your project.

Step 2: Navigate to Test Plans

- In the left menu, click on Test Plans.
- Click on "New Test Plan" and give it a name (e.g., *Emergency Alerts*).

Step 3: Create Test Suites

Create a new suite for each feature/module being tested.

* *Suite 1*: Add Employee

* *Suite 2*: Update Salary

* *Suite 3*: Delete Employee

* *Suite 4: **Download Employee Details as PDF*

Step 4: Add Test Cases

Inside *Suite 4, click ***"+ New Test Case"* and fill in the details as follows:

Test Case ID: TC004

*Title: **Download Employee Details PDF with Header Info*

Scenario: Test if the application successfully generates a downloadable PDF containing all employee details along with total gross salary and total annual salary.

Steps:*

1. Launch the Salary Management App.

2. Log in as HR.

3. Navigate to the HR Dashboard.

4. Click the *"Download PDF"* button.

5. In the popup, select a valid folder location and filename.

6. Confirm the file is downloaded.

7. Open the downloaded PDF.

8. Verify the presence of:

* Employee table with correct data

* Total gross salary

* Total annual salary

Expected Result:

A PDF file is generated successfully, containing complete employee details and the correct totals, formatted properly and readable.

Step 5: Execute Test Cases

* Open each test case and click *“Run for web application.”*

* Perform the steps manually or with automation support.

* Mark the result as *Pass* or *Fail*.

* Provide *Actual Result* and *Remarks* if needed.

Step 6: Track and Report

* Navigate to *Test Plans → Charts* to view real-time progress.

* Use filters to track:

  * Number of test cases passed/failed

  * Percentage of coverage for related user stories

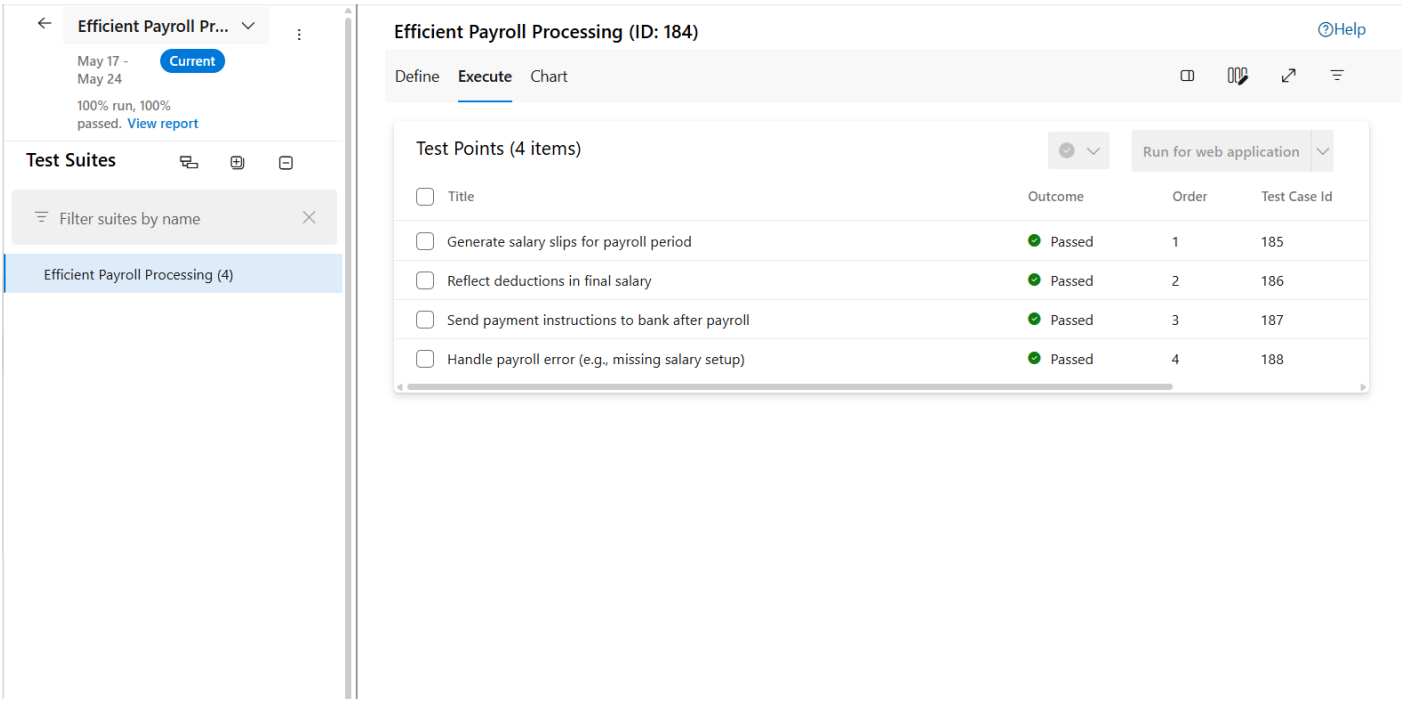
  * Linked bugs or issues discovered during testing

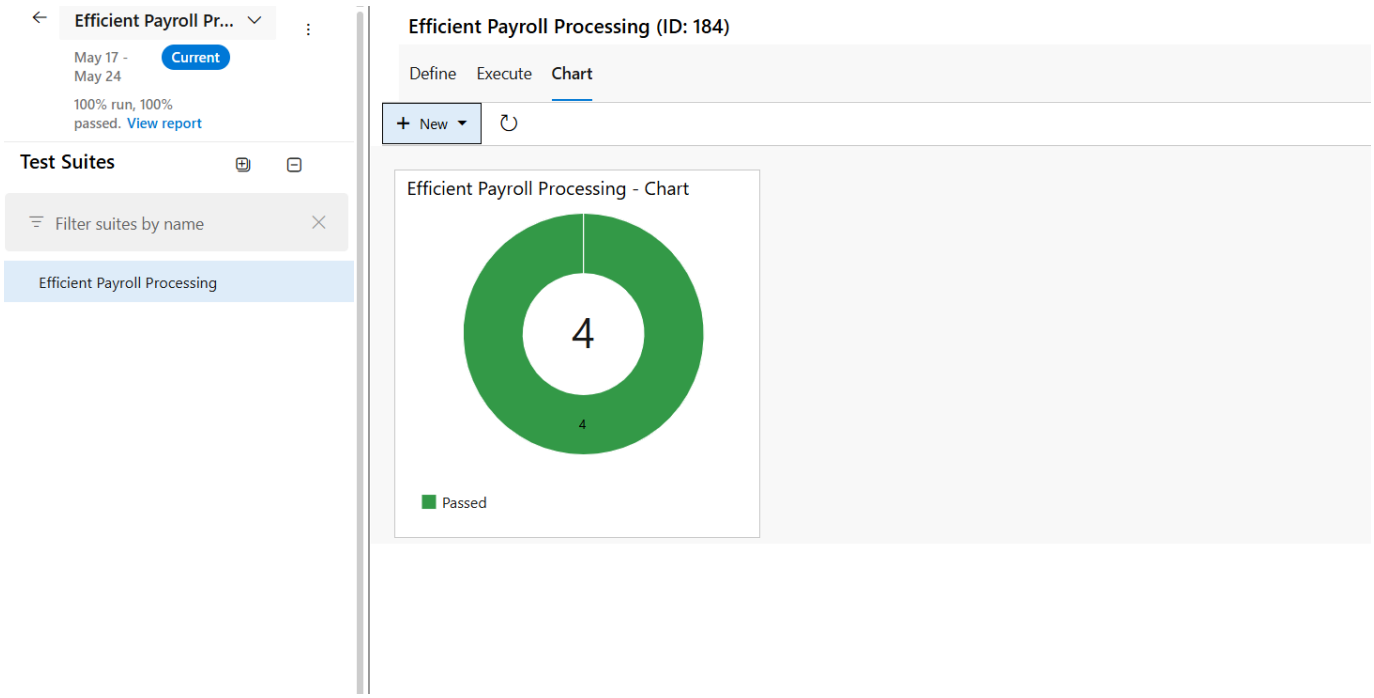
Example Outcome (Optional in Record):

* Test Case TC004: *PASS*

* Actual Result: PDF downloaded with 10 employee entries and correct total salaries shown.

* Remarks: Layout and formatting are acceptable. No issues found.





RESULT:

Thus the application was successfully tested in Azure.

EX NO: 13 CI/CD PIPELINE

AIM:

To implement a Continuous Integration and Continuous Deployment (CI/CD) pipeline for the Salary Management System using Azure DevOps, ensuring automated build, test, and deployment of the application.

PROCEDURE:

Step 1: Create a Build Pipeline (CI)

- Go to Pipelines → Create Pipeline.
- Select Azure Repos Git → Choose your repository.
- Choose Starter pipeline or YAML file.
- Add pipeline tasks like:

```
trigger:
  - main
pool:
  name: Default
steps:
  - task: UseNode@2
    inputs:
      version: '18.x'
  - script: npm install
    displayName: 'Install Dependencies'
  - script: npm run build
    displayName: 'Build Application'


  - script: npm run test
    displayName: 'Run Tests'
```

- Save and Run the pipeline to verify.

Step 4: Set Up Release Pipeline (CD)

- Navigate to Pipelines → Releases → New pipeline.
- Add an Artifact (your build pipeline output).
- Add Stages like:
  - Development
  - Production
- Configure Deploy tasks in each stage:
  - For web apps: Use Azure Web App Deploy task.
  - For mobile: Use relevant deployment tools.

Step 5: Add Approvals and Gates (Optional)

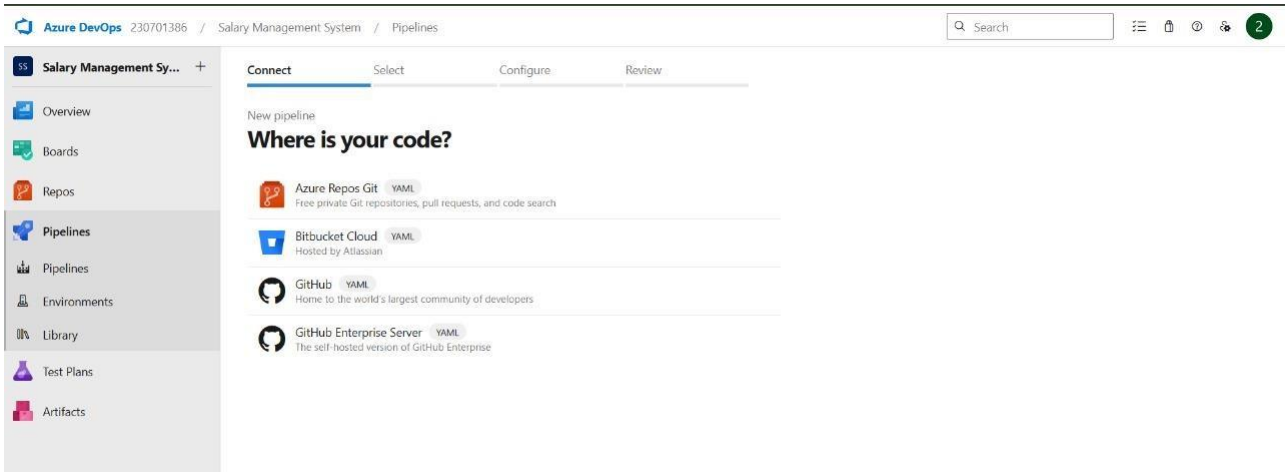
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

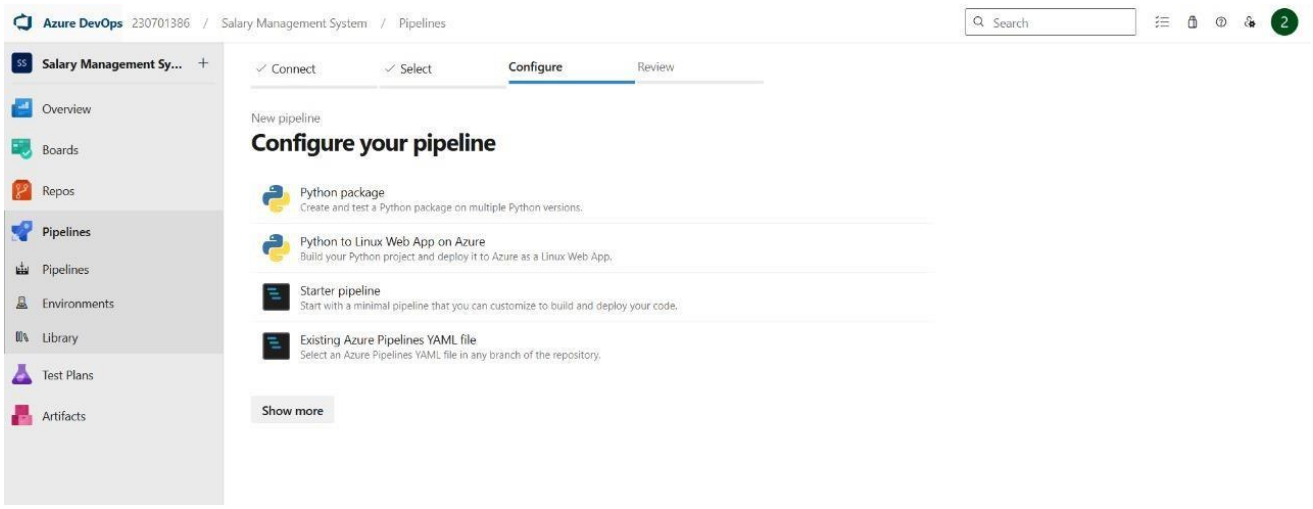
Step 6: Automate Triggering

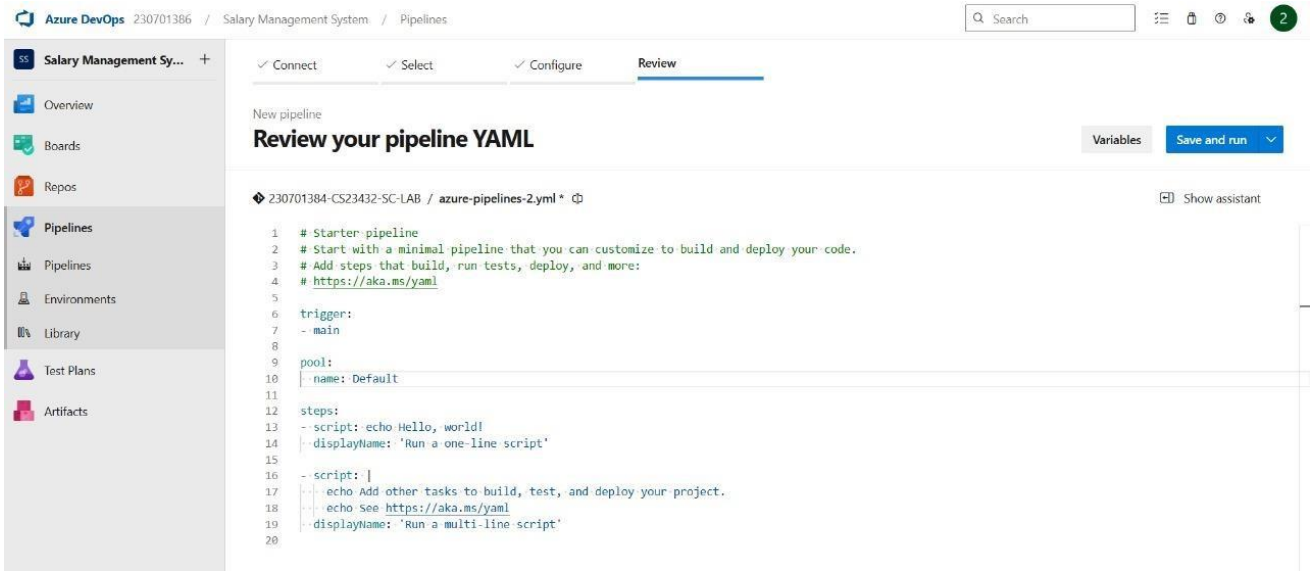
- Ensure the pipeline triggers:
  - On code push to main branch (CI)
  - After successful build for deployment (CD)

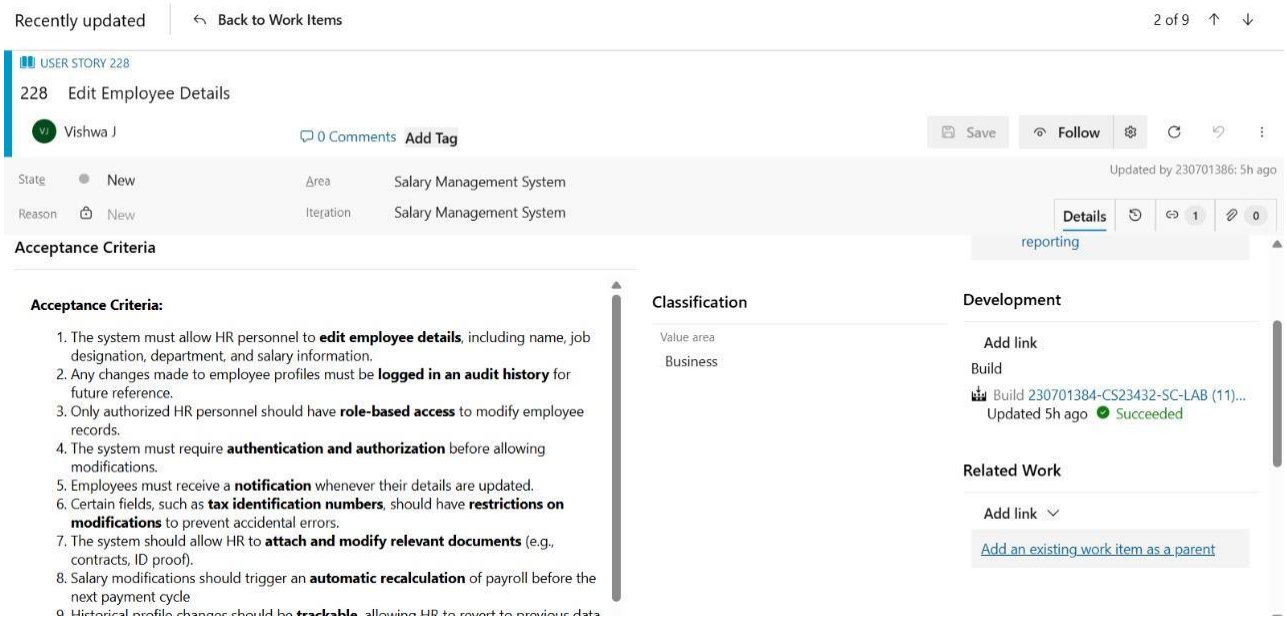
Step 7: Monitor Pipeline

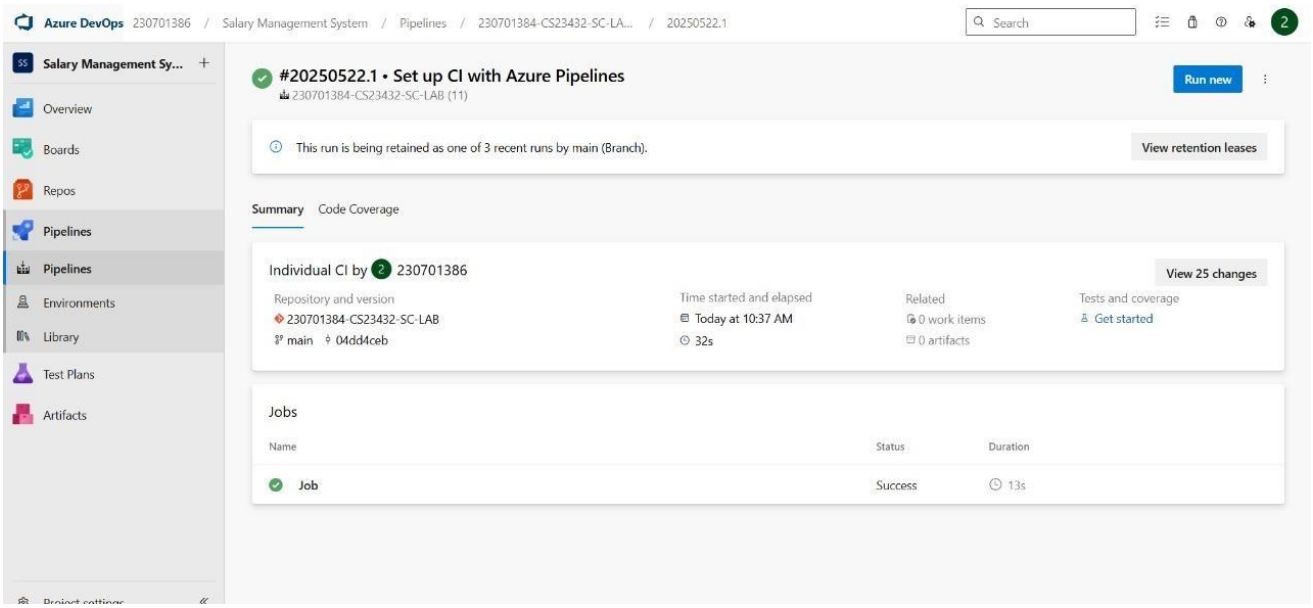
- Track pipeline status under Pipelines → Runs.
- Debug failures and download logs if necessary.
- Use Azure Boards to link builds with user stories and bugs.




Azure DevOps 230701386 / Salary Management System / Pipelines
Search
Salary Management Sy...
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Connect
Select
Configure
Review
New pipeline
Configure your pipeline
Python package
Create and test a Python package on multiple Python versions.
Python to Linux Web App on Azure
Build your Python project and deploy it to Azure as a Linux Web App.
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Existing Azure Pipelines YAML file
Select an Azure Pipelines YAML file in any branch of the repository.
Show more


Azure DevOps 230701386 / Salary Management System / Pipelines
Search
Salary Management Sy...
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Library
Test Plans
Artifacts
Connect
Select
Configure
Review
New pipeline
Review your pipeline YAML
Variables
Save and run
230701384-CS23432-SC-LAB / azure-pipelines-2.yml *
Show assistant
Starter pipeline
Start with a minimal pipeline that you can customize to build and deploy your code.
Add steps that build, run tests, deploy, and more:
https://aka.ms/yaml
trigger:
- main
pool:
name: Default
steps:
- script: echo Hello, world!
displayName: 'Run a one-line script'
- script: |
echo Add other tasks to build, test, and deploy your project.
echo See https://aka.ms/yaml
displayName: 'Run a multi-line script'


Recently updated
Back to Work Items
2 of 9
USER STORY 228
228 Edit Employee Details
Vishwa J
0 Comments
Add Tag
Save
Follow
Updated by 230701386: 5h ago
State
New
Area
Salary Management System
Reason
New
Iteration
Salary Management System
Details
Acceptance Criteria
reporting
Acceptance Criteria:
1. The system must allow HR personnel to edit employee details, including name, job designation, department, and salary information.
2. Any changes made to employee profiles must be logged in an audit history for future reference.
3. Only authorized HR personnel should have role-based access to modify employee records.
4. The system must require authentication and authorization before allowing modifications.
5. Employees must receive a notification whenever their details are updated.
6. Certain fields, such as tax identification numbers, should have restrictions on modifications to prevent accidental errors.
7. The system should allow HR to attach and modify relevant documents (e.g., contracts, ID proof).
8. Salary modifications should trigger an automatic recalculation of payroll before the next payment cycle
Classification
Value area
Business
Development
Add link
Build
Build 230701384-CS23432-SC-LAB (11)...
Updated 5h ago
Succeeded
Related Work
Add link
Add an existing work item as a parent



RESULT:

Thus the CI/CD pipeline has been successfully implemented.