RAJALAKSHMI ENGINEERING COLLEGE

THANDALAM – 602 105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ACADEMIC YEAR 2024-2025

CS23432 SOFTWARE CONSTRUCTION
Lab Manual 2024-2025

Name : VISHWA J

Year/Branch/Section : II /CSE/ D

Register No. : 230701384

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.

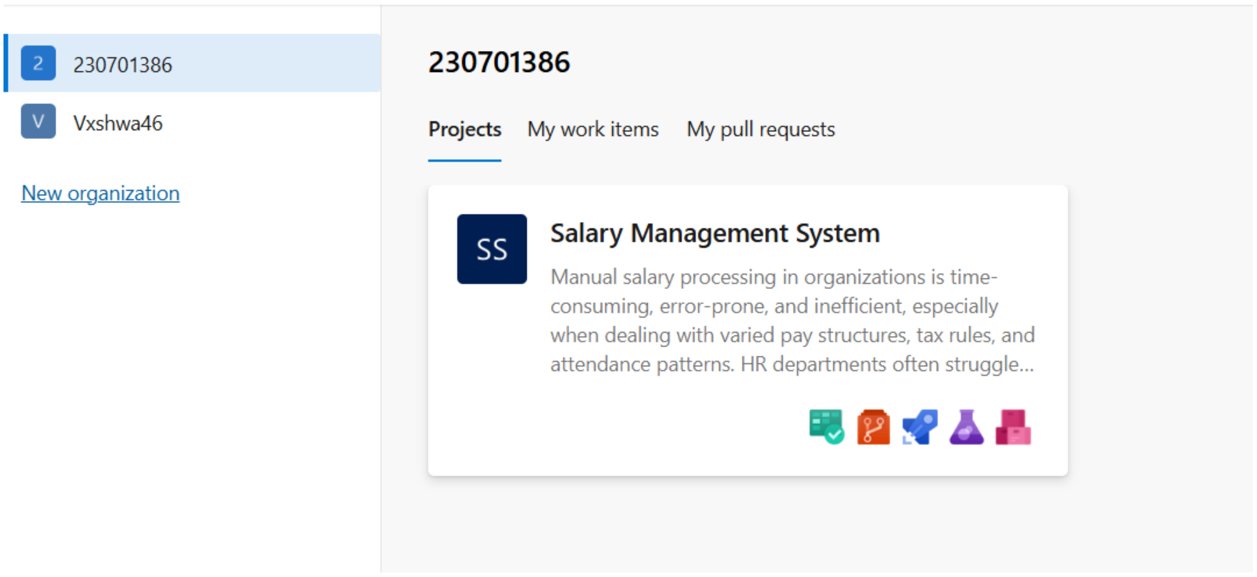
6. Create the First Project in Your Organization. After the organization is set up, you’ll need to create your first project. This is where you'll begin to manage code, pipelines, work items, and more.

(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.

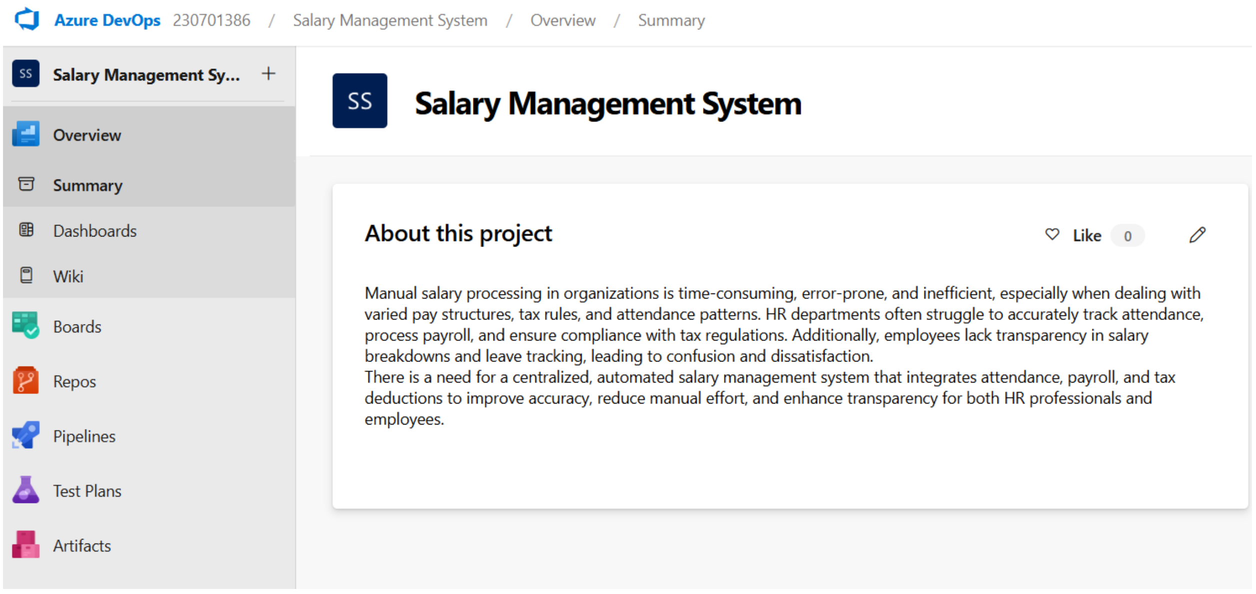


Salary Management System

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...

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.


Azure DevOps 230701386 / Salary Management System / Overview / Summary
SS Salary Management Sy... +
Overview
Summary
Dashboards
Wiki
Boards
Repos
Pipelines
Test Plans
Artifacts
SS
Like 0

Salary Management System

About this project

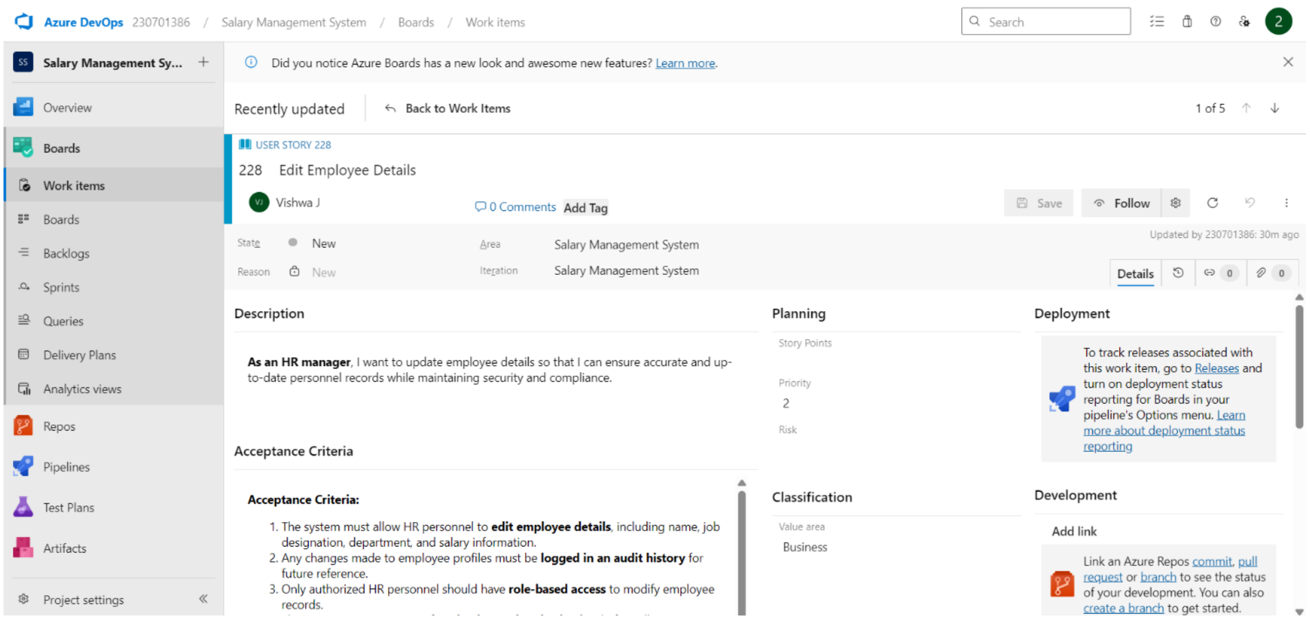
Manual salary processing in organizations is time-consuming, error-prone, and inefficient, especially when dealing with varied pay structures, tax rules, and attendance patterns. HR departments often struggle to accurately track attendance, process payroll, and ensure compliance with tax regulations. 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.

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.



Azure DevOps 230701386 / Salary Management System / Boards / Work items

Did you notice Azure Boards has a new look and awesome new features? Learn more.

Recently updated | Back to Work Items | 1 of 5

USER STORY 228

228 Edit Employee Details

Vishwa J | 0 Comments | Add Tag | Save | Follow

Updated by 230701386: 30m ago

State	New	Area	Salary Management System
Reason	New	Iteration	Salary Management System

Description

As an HR manager, I want to update employee details so that I can ensure accurate and up-to-date personnel records while maintaining security and compliance.

Acceptance Criteria

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.

Planning

Story Points

Priority: 2

Risk

Classification

Value area: Business

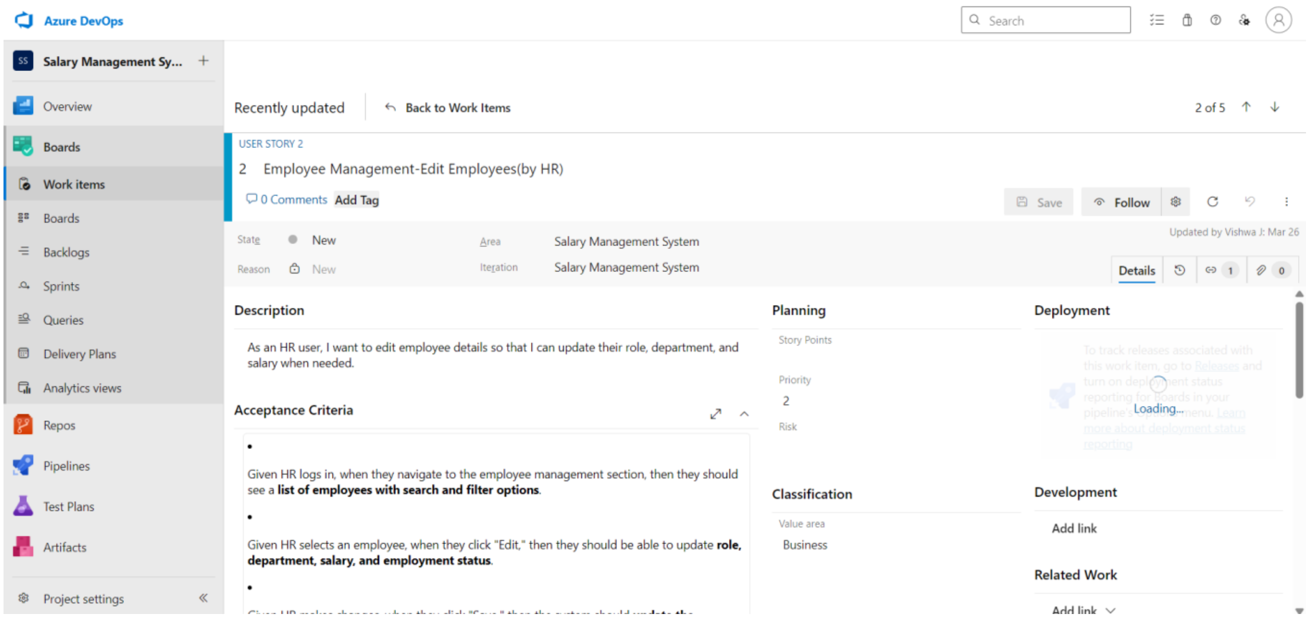
Deployment

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.



Azure DevOps

Recently updated | Back to Work Items | 2 of 5

USER STORY 2

2 Employee Management-Edit Employees(by HR)

0 Comments | Add Tag | Save | Follow

Updated by Vishwa J: Mar 26

State	New	Area	Salary Management System
Reason	New	Iteration	Salary Management System

Description

As an HR user, I want to edit employee details so that I can update their role, department, and salary when needed.

Acceptance Criteria

- Given HR logs in, when they navigate to the employee management section, then they should see a **list of employees with search and filter options**.
- Given HR selects an employee, when they click "Edit," then they should be able to update **role, department, salary, and employment status**.

Planning

Story Points

Priority: 2

Risk

Classification

Value area: Business

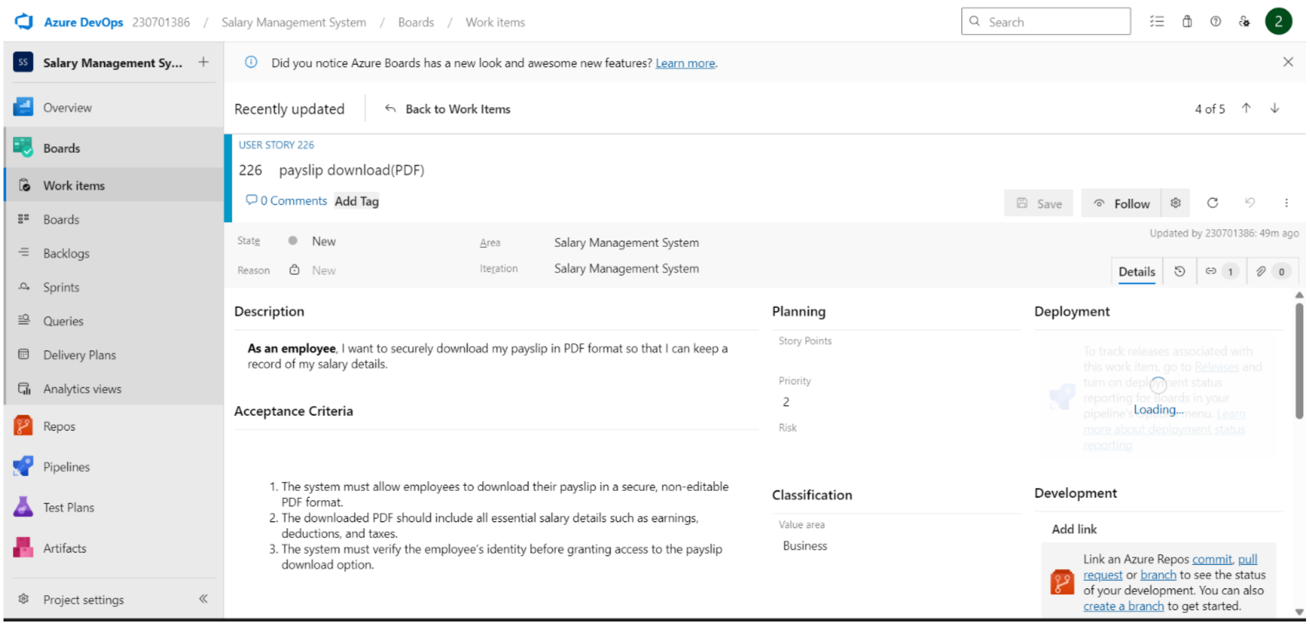
Deployment

Loading...

Development

Add link

Related Work



Did you notice Azure Boards has a new look and awesome new features? Learn more.

USER STORY 226

226 payslip download(PDF)

0 Comments Add Tag

Updated by 230701386: 49m ago

State	New	Area	Salary Management System
Reason	New	Iteration	Salary Management System

Description

As an employee, I want to securely download my payslip in PDF format so that I can keep a record of my salary details.

Acceptance Criteria

1. The system must allow employees to download their payslip in a secure, non-editable PDF format.
2. The downloaded PDF should include all essential salary details such as earnings, deductions, and taxes.
3. The system must verify the employee's identity before granting access to the payslip download option.

Planning

Story Points

Priority
2

Risk

Classification

Value area
Business

Deployment

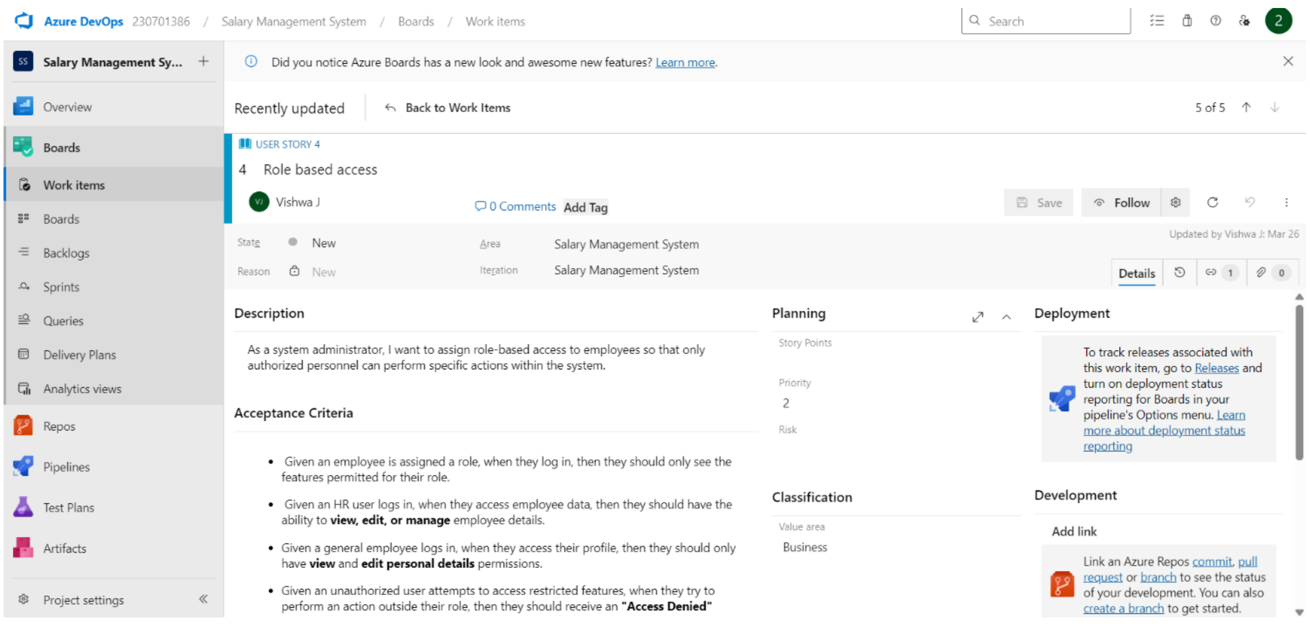
To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Loading...

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.



Did you notice Azure Boards has a new look and awesome new features? Learn more.

USER STORY 4

4 Role based access

Vishwa J

0 Comments Add Tag

Updated by Vishwa J: Mar 26

State	New	Area	Salary Management System
Reason	New	Iteration	Salary Management System

Description

As a system administrator, I want to assign role-based access to employees so that only authorized personnel can perform specific actions within the system.

Acceptance Criteria

- Given an employee is assigned a role, when they log in, then they should only see the features permitted for their role.
- Given an HR user logs in, when they access employee data, then they should have the ability to **view, edit, or manage** employee details.
- Given a general employee logs in, when they access their profile, then they should only have **view** and **edit personal details** permissions.
- Given an unauthorized user attempts to access restricted features, when they try to perform an action outside their role, then they should receive an **"Access Denied"**

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.

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: Edit Employee Details (by HR)

As an HR manager, I want to edit employee records so that I can update relevant information such as job designation, department, and salary while maintaining security and compliance.

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. The system should allow HR to attach and modify relevant documents
7. Salary modifications should trigger an automatic recalculation of payroll before the next payment cycle
8. Historical profile changes should be trackable, allowing HR to revert to previous data if necessary.
9. The system must prevent duplicate records, ensuring accuracy in employee data management.
10. HR must have the option to temporarily lock employee profiles if modifications are being reviewed.
11. Profile updates must comply with company policies and labour laws regarding employment records.
12. Employees must be able to view, but not modify, certain sensitive fields in their profile.
13. Any profile edits should sync across all related modules, ensuring consistency in payroll, attendance, and tax calculations.

USER STORY 2: Payslip Download

As an employee, I want to securely download my payslip so that I can keep a record of my salary details and financial transactions

Acceptance Criteria:

1. The system must allow employees to download their payslip in PDF format.
2. Payslips must include earnings, deductions, tax details, and net salary breakdown.
3. Employees must be able to access and download past payslips for at least 12 months.
4. Payslips should be password-protected to maintain security.
5. Employees should receive an automatic notification when a new payslip is available.
6. Payslips must comply with local payroll and tax regulations.
7. Employees should be able to print their payslips without formatting issues.
8. The system should allow bulk-download functionality for HR managers handling multiple employees.
9. Payslips should be accessible on both web and mobile platforms.
10. Employees should have the option to export payslips in other formats (PDF).
11. Payslip download must be instant and error-free, without delays or system failures.
12. The payslip design should be structured and easy to read, ensuring clarity for employees.
13. The system must ensure confidentiality and restrict unauthorized access to employee payslips.

USER STORY 3: Employee Management

As an HR manager, I want to manage employee profiles, roles, and payroll-related details so that I can ensure seamless personnel administration.

Acceptance Criteria:

1. The system must allow HR personnel to add, edit, and remove employee records.
2. Employee profiles should contain personal details, department, designation, and salary information.
3. The system must track employment status, including active, probationary, and terminated employees.
4. HR personnel should be able to assign roles and permissions to employees.
5. The system must integrate with attendance tracking for payroll calculations.
6. Employees should be able to update limited personal details (address, phone number, emergency contacts).
7. The system should allow uploading important documents (offer letter, contract, identification proof).
8. HR should be able to assign and update salary structures for employees.
9. The system must support different pay grades and bands.
10. Employees should receive automated alerts for document updates (renewal of contracts, policy updates).

11. The system should provide an overview dashboard of employee distribution by department and role.
12. Employees should have a self-service portal to access their personal information and payroll details.
13. The system must support multi-country employee management with localization options.
14. HR personnel should be able to generate custom employee reports based on department, tenure, and salary.
15. The system must ensure data confidentiality and security for employee records.

USER STORY 4: Role-Based Access (HR & Employee)

As an HR manager, I want role-based access control so that I can ensure employees have appropriate permissions to view and modify information securely.

Acceptance Criteria:

1. The system must restrict access based on user roles, ensuring HR personnel can edit employee records while employees can only view their own details.
2. HR should have the ability to assign, modify, and revoke roles for employees.
3. Employees should only have access to their own payroll, attendance, and profile data.
4. The system should support role hierarchy, allowing senior HR members to have greater access levels.
5. Certain sensitive fields (e.g., salary structure, tax details) must be restricted from non-HR personnel.
6. Employees should receive a notification when their role permissions are updated.
7. HR should be able to grant temporary access for specific actions when needed.
8. The system must log all access changes for security auditing.
9. Users must undergo authentication before making role-based modifications.
10. Access permissions should be revoked immediately when an employee is terminated.
11. The system must prevent unauthorized users from bypassing role restrictions.
12. Employees should be able to view their assigned role within the system.
13. The system should provide error messages when a user attempts an action outside their role permissions.
14. HR must be able to generate reports on role-based access history.

15. The system must comply with data protection and privacy regulations when managing access rights.

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

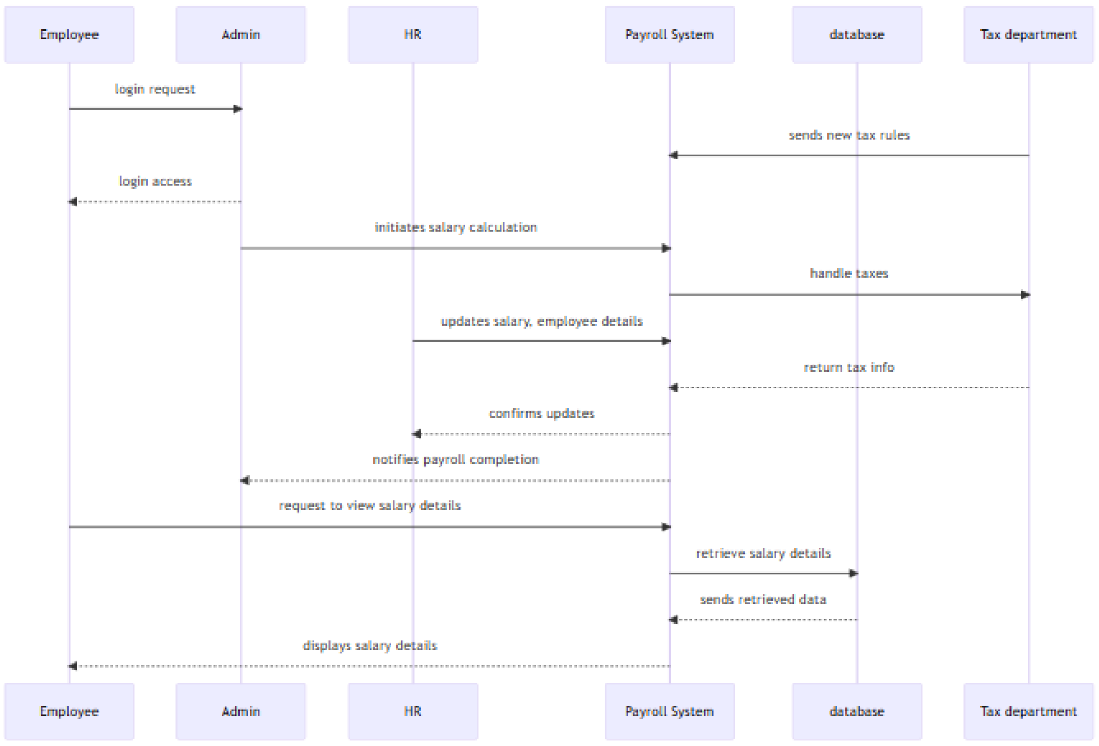
Employee ->> Payroll System : request to view salary details

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.

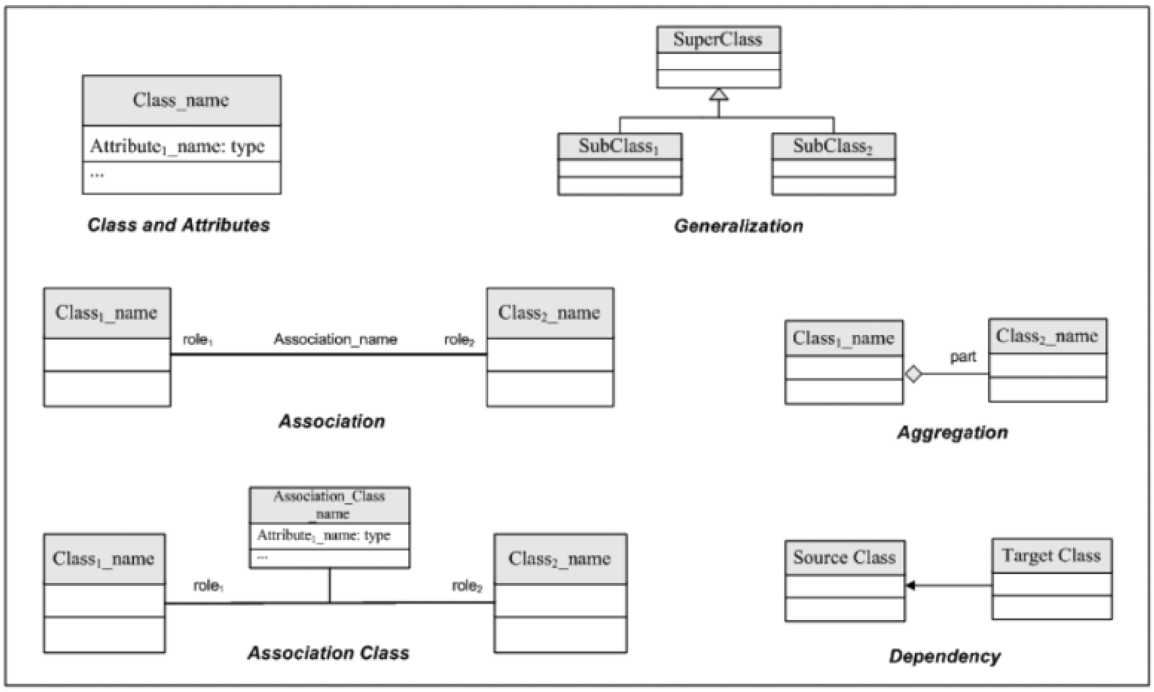
EX NO: 6 CLASS DIAGRAM

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.

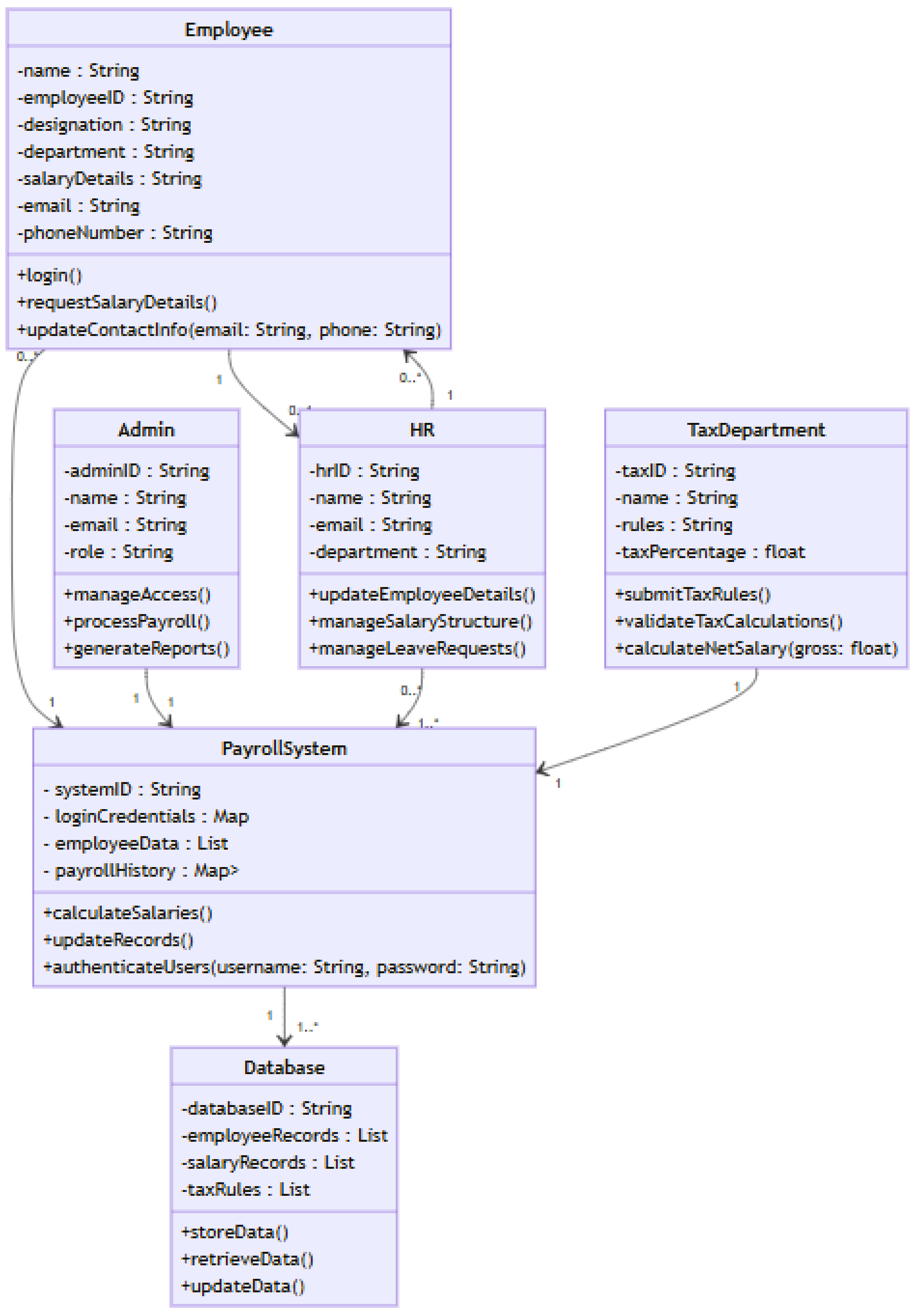
2. To design select wiki from menu.

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

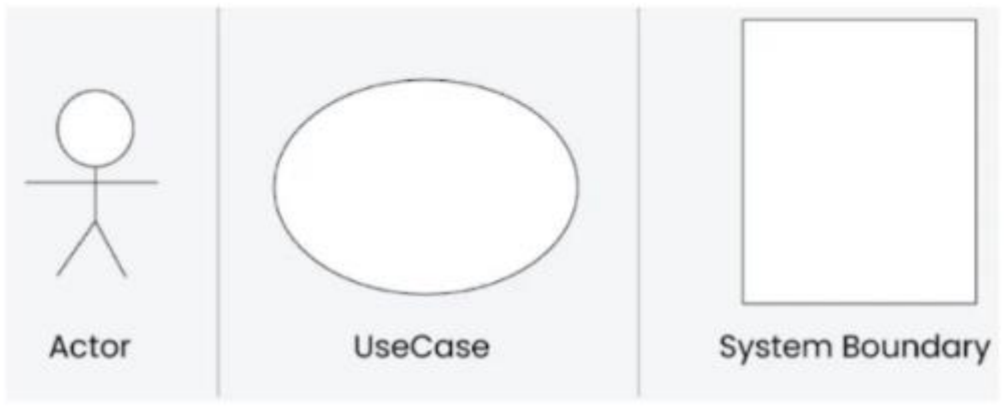
AIM:

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.

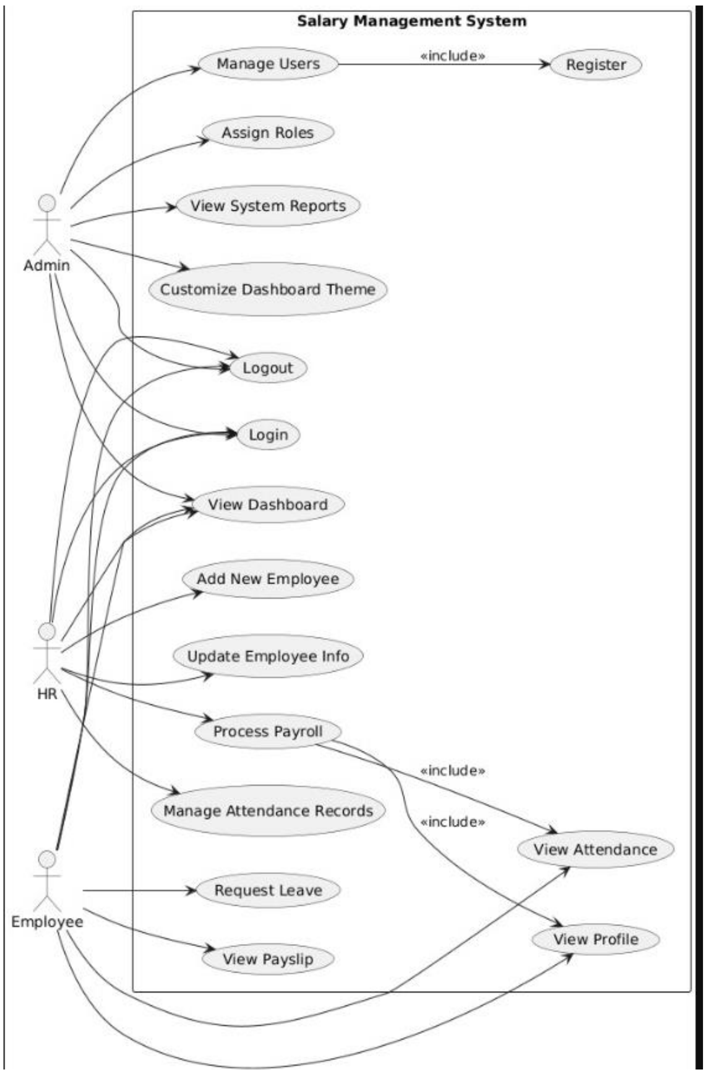
Step 2: Upload the Diagram to Azure DevOps

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.

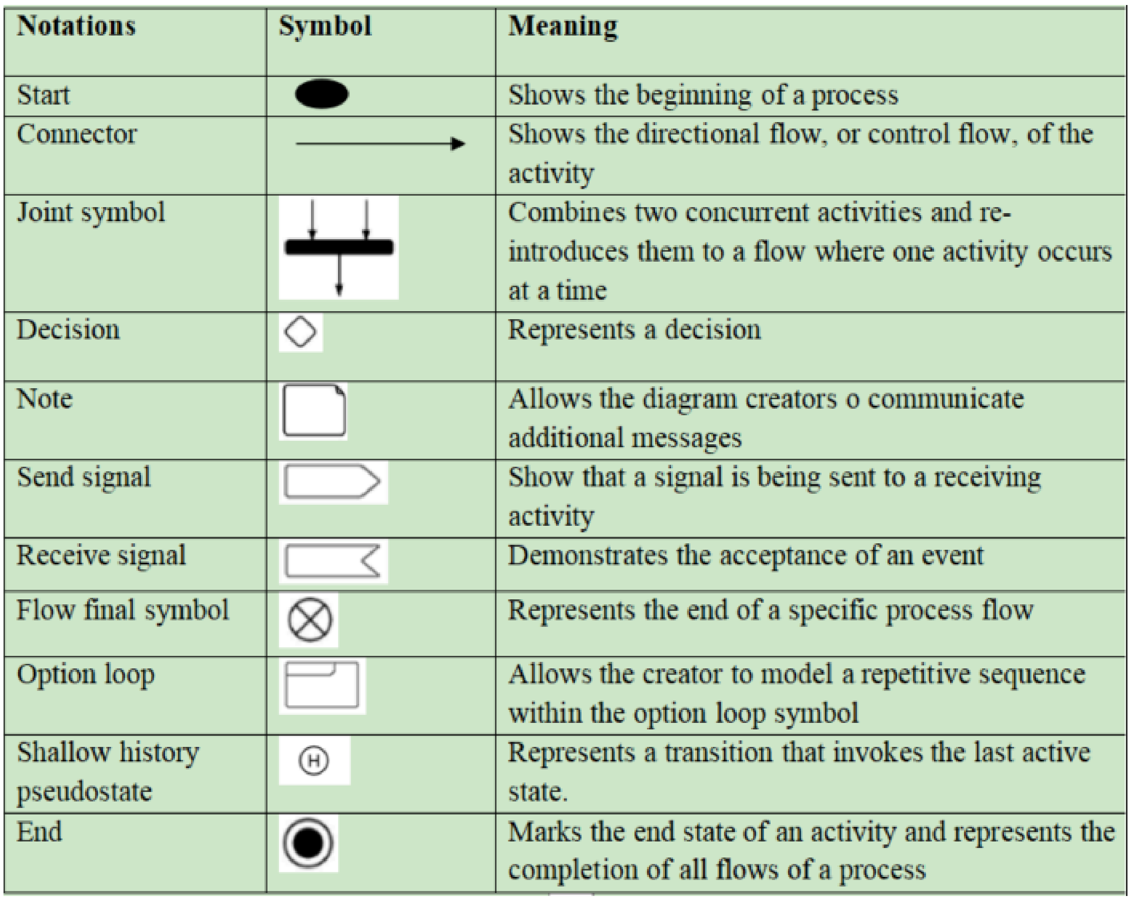
EX NO: 8 ACTIVITY DIAGRAM

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.

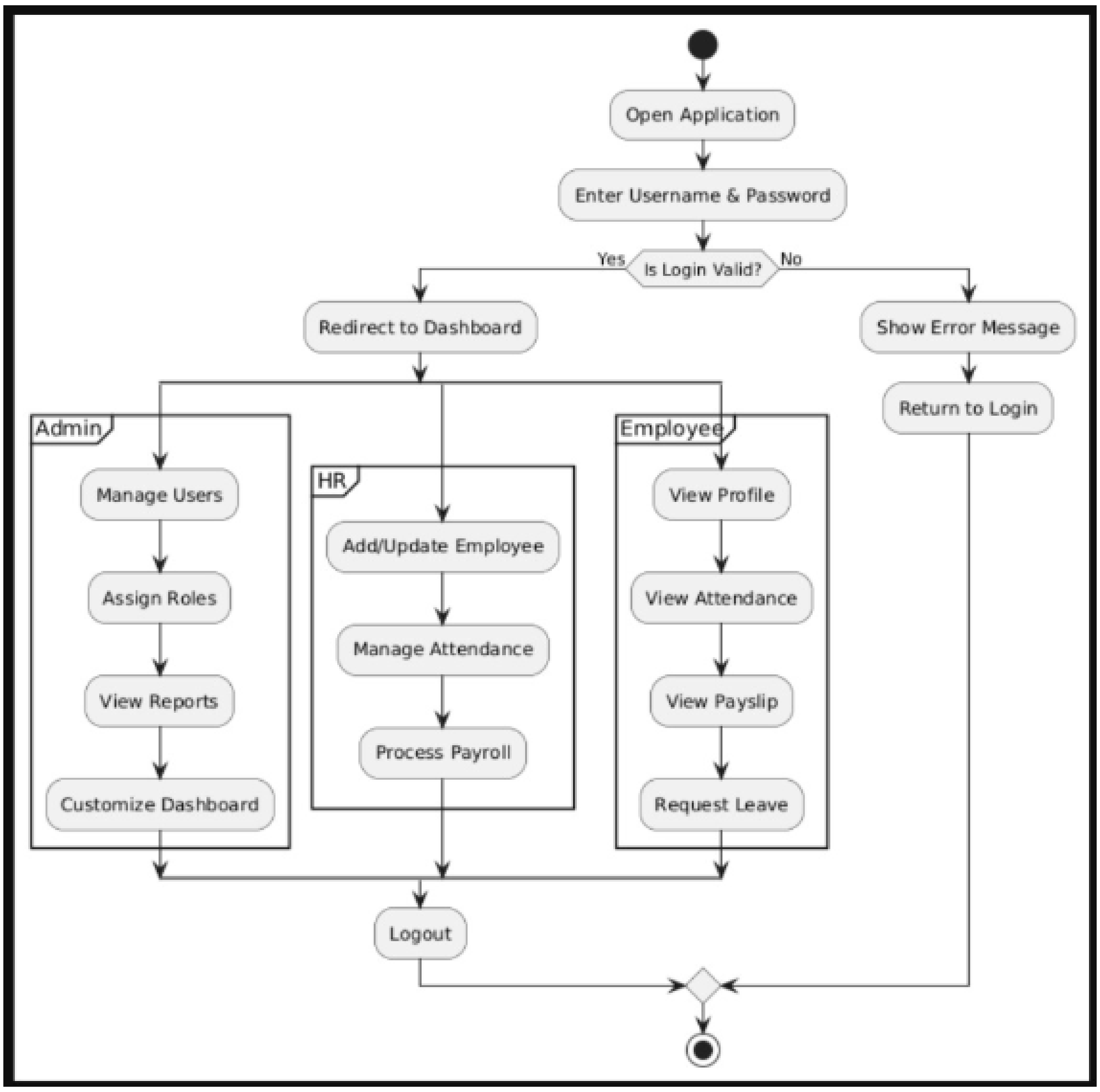


Notations	Symbol	Meaning
Start	●	Shows the beginning of a process
Connector	→	Shows the directional flow, or control flow, of the activity
Joint symbol	⊥	Combines two concurrent activities and re-introduces them to a flow where one activity occurs at a time
Decision	◇	Represents a decision
Note	▭	Allows 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.

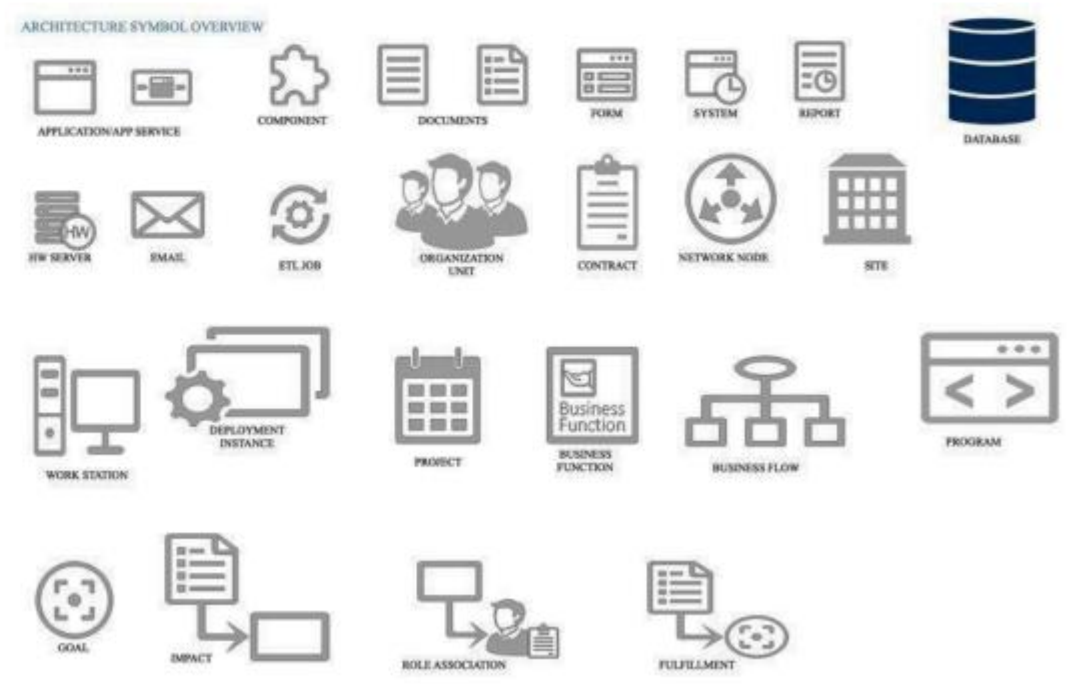
EX NO: 9 ARCHITECTURE DIAGRAM

AIM:

To draw the Architecture Diagram using draw.io.

THEORY:

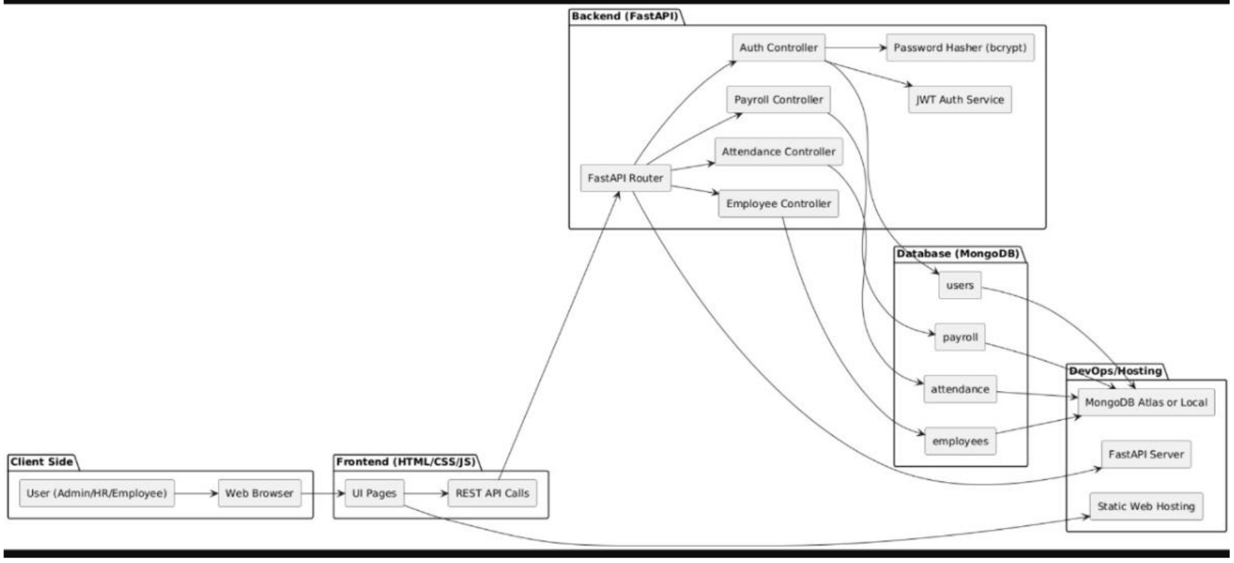
An architectural diagram is a visual representation that maps out the physical implementation for components of a software system. It shows the general structure of the software system and the associations, limitations, and boundaries between each element.



PROCEDURE:

Step 1. Draw diagram in draw.io

Step 2. Upload the diagram in Azure DevOps wiki.



RESULT:

The architecture diagram was designed successfully

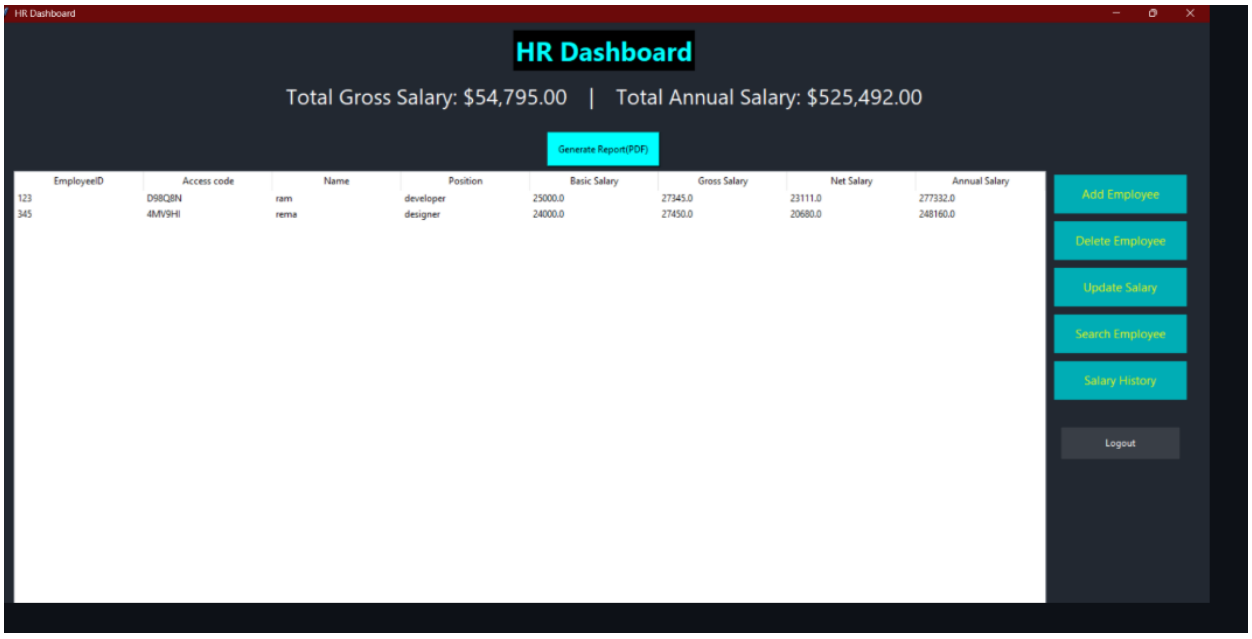
EX NO: 10 USER INTERFACE

AIM:

Design User Interface for the Salary Management System.

UI DESIGNS OF SALARY MANAGEMENT APPLICATION:

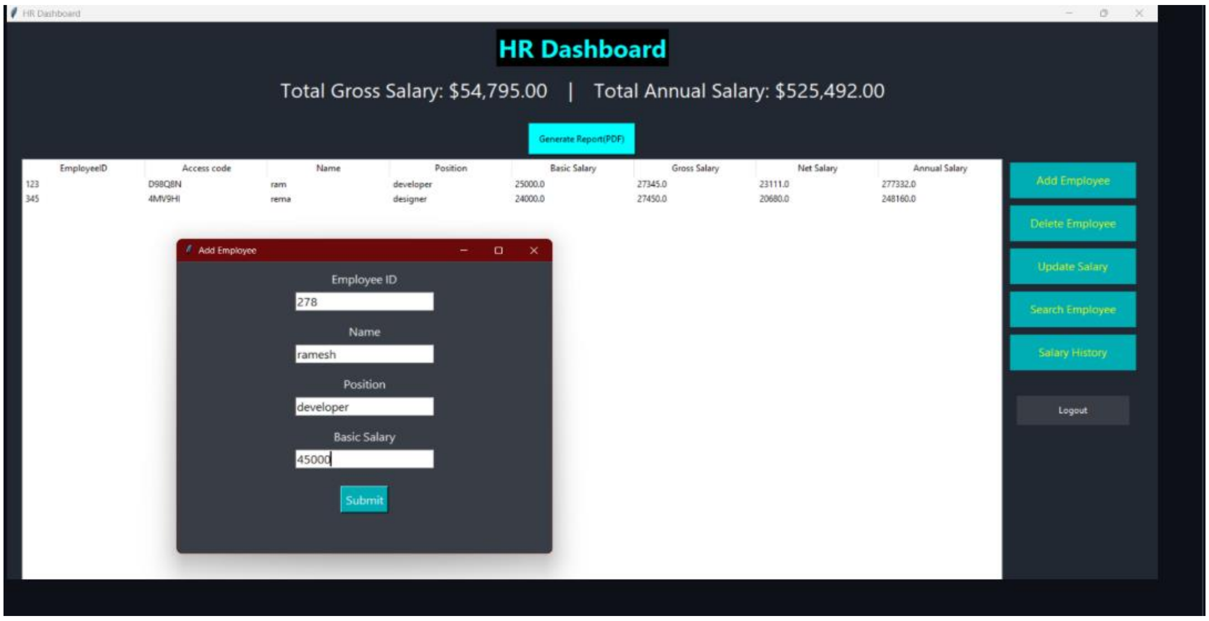
1. HR Dashboard



Total Gross Salary: $54,795.00 | Total Annual Salary: $525,492.00

EmployeeID	Access code	Name	Position	Basic Salary	Gross Salary	Net Salary	Annual Salary
123	D98Q8N	ram	developer	25000.0	27345.0	23111.0	277332.0
345	4MV9HI	rema	designer	24000.0	27450.0	20680.0	248160.0

2. Add Employee Feature

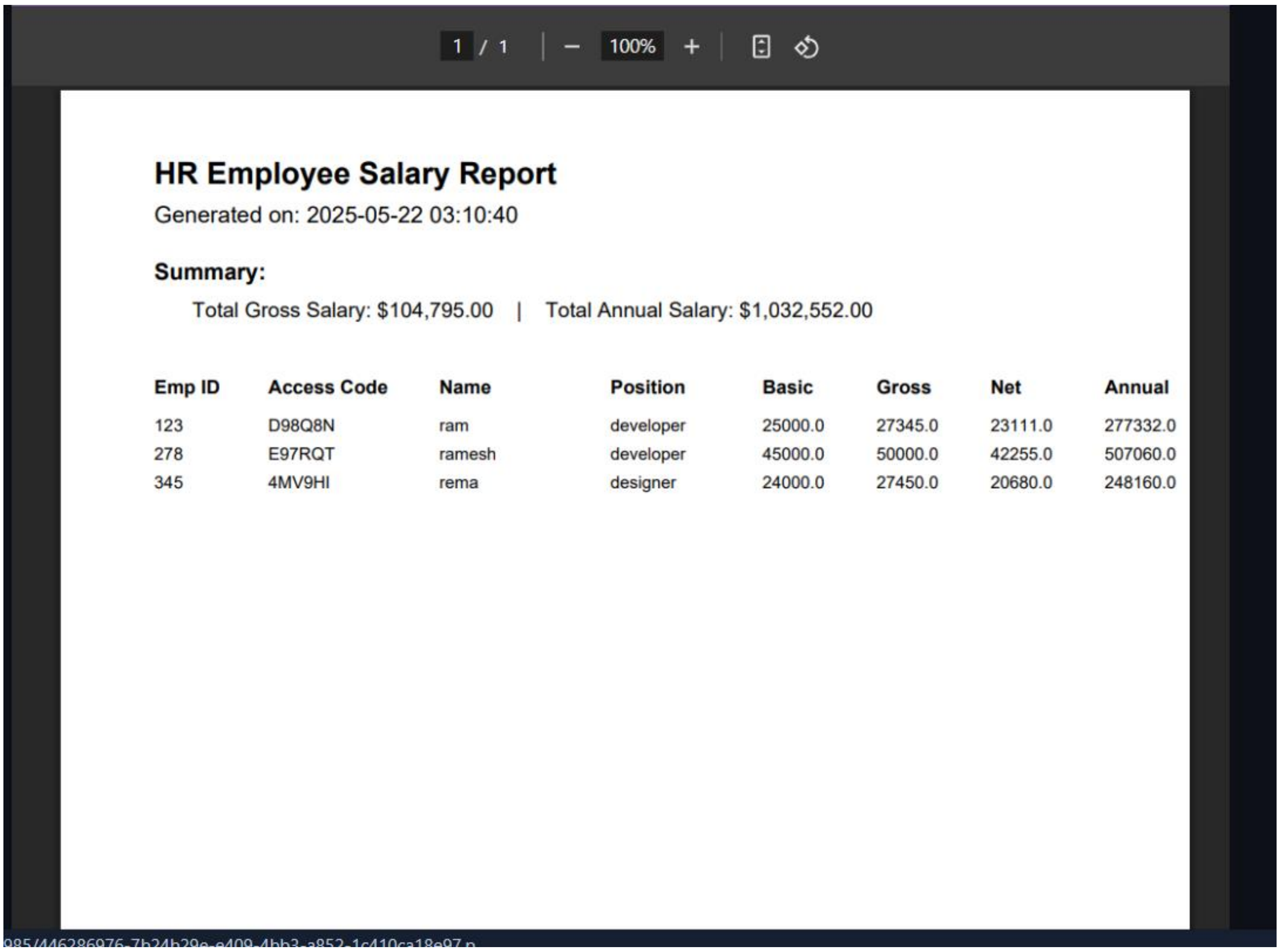


Total Gross Salary: $54,795.00 | Total Annual Salary: $525,492.00

EmployeeID	Access code	Name	Position	Basic Salary	Gross Salary	Net Salary	Annual Salary
123	D98Q8N	ram	developer	25000.0	27345.0	23111.0	277332.0
345	4MV9HI	rema	designer	24000.0	27450.0	20680.0	248160.0

Employee ID: 278
Name: ramesh
Position: developer
Basic Salary: 45000

3. Employee Details report generated as PDF



HR Employee Salary Report

Generated on: 2025-05-22 03:10:40

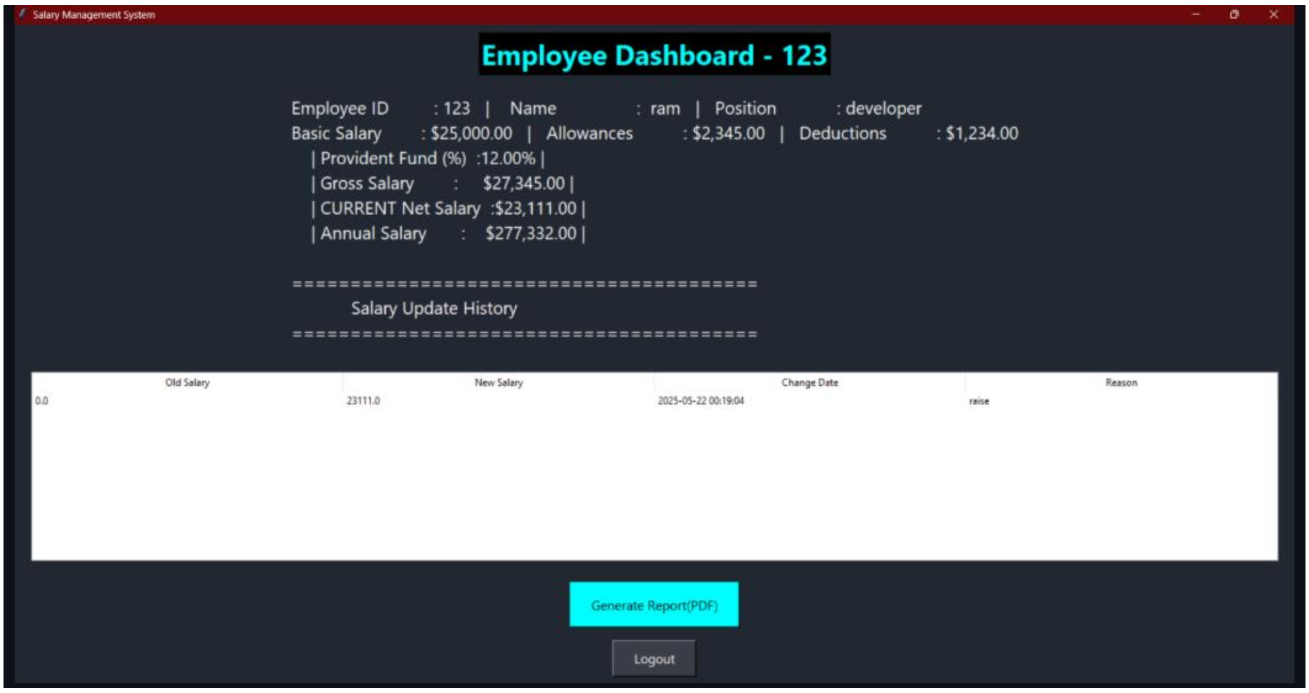
Summary:

Total Gross Salary: $104,795.00 | Total Annual Salary: $1,032,552.00

Emp ID	Access Code	Name	Position	Basic	Gross	Net	Annual
123	D98Q8N	ram	developer	25000.0	27345.0	23111.0	277332.0
278	E97RQT	ramesh	developer	45000.0	50000.0	42255.0	507060.0
345	4MV9HI	rema	designer	24000.0	27450.0	20680.0	248160.0

985/446286976-7b24b29e-e409-4bb3-a852-1c410ca18e97.p

4. Employee Dashboard



Employee Dashboard - 123

```
Employee ID      : 123  |  Name          : ram  |  Position        : developer
Basic Salary     : $25,000.00  |  Allowances      : $2,345.00  |  Deductions       : $1,234.00
  | Provident Fund (%)  :12.00% |
  | Gross Salary      :     $27,345.00 |
  | CURRENT Net Salary  :$23,111.00 |
  | Annual Salary      :    $277,332.00 |

=========================================
        Salary Update History
=========================================
```

Old Salary	New Salary	Change Date	Reason
0.0	23111.0	2025-05-22 00:19:04	raise

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

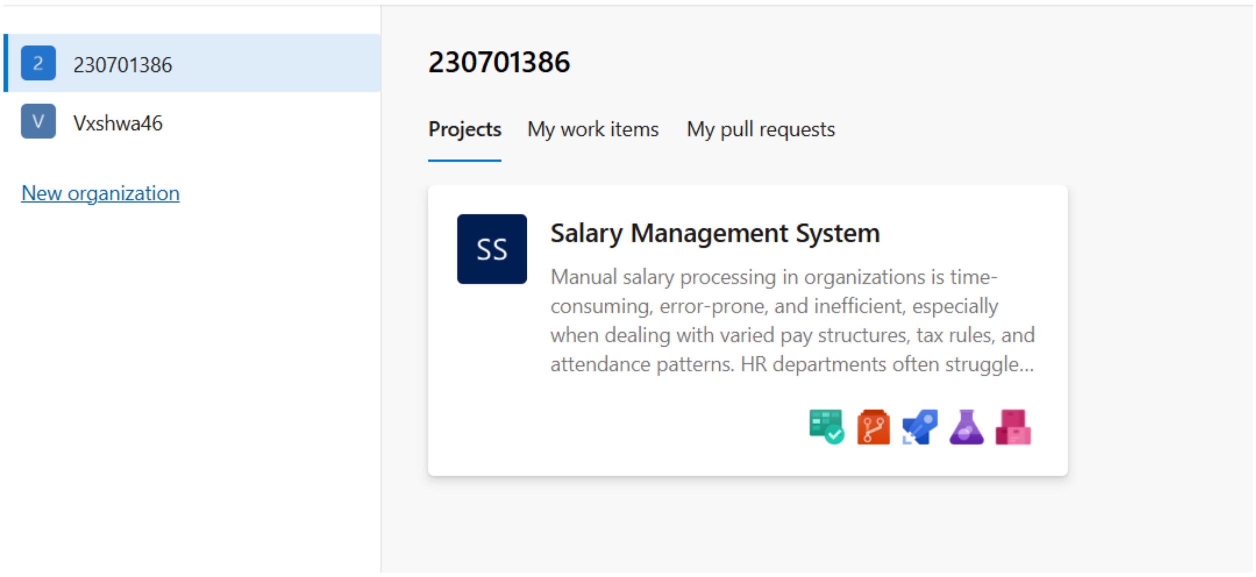
- Navigate to Repos → Click "Clone" to get the Git URL.
- Open Visual Studio Code / Terminal and run:

```
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
```

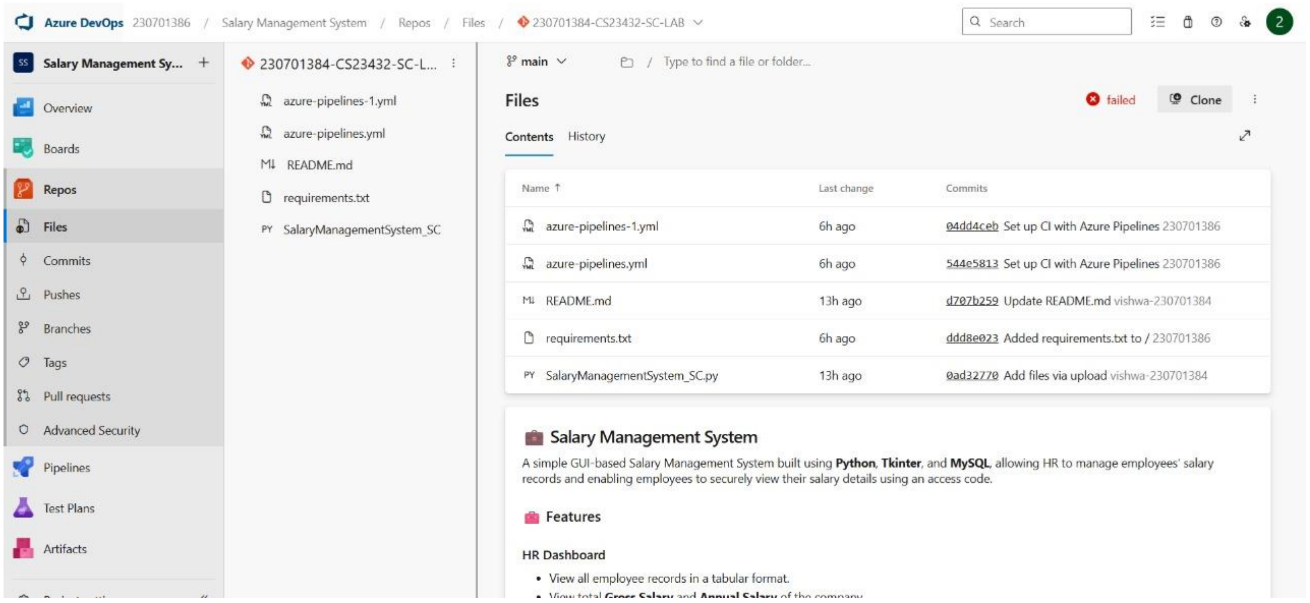
Step 3: Set Up Build Pipeline (CI/CD - Continuous Integration)



230701386

Salary Management System

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...



Files

Name ↑	Last change	Commits
azure-pipelines-1.yml	6h ago	04dd4ceb Set up CI with Azure Pipelines 230701386
azure-pipelines.yml	6h ago	544e5813 Set up CI with Azure Pipelines 230701386
README.md	13h ago	d707b259 Update README.md vishwa-230701384
requirements.txt	6h ago	ddd8e023 Added requirements.txt to / 230701386
SalaryManagementSystem_SC.py	13h ago	0ad32770 Add files via upload vishwa-230701384

Salary Management System

A simple GUI-based Salary Management System built using **Python**, **Tkinter**, and **MySQL**, allowing HR to manage employees' salary records and enabling employees to securely view their salary details using an access code.

Features

HR Dashboard

- View all employee records in a tabular format.

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:

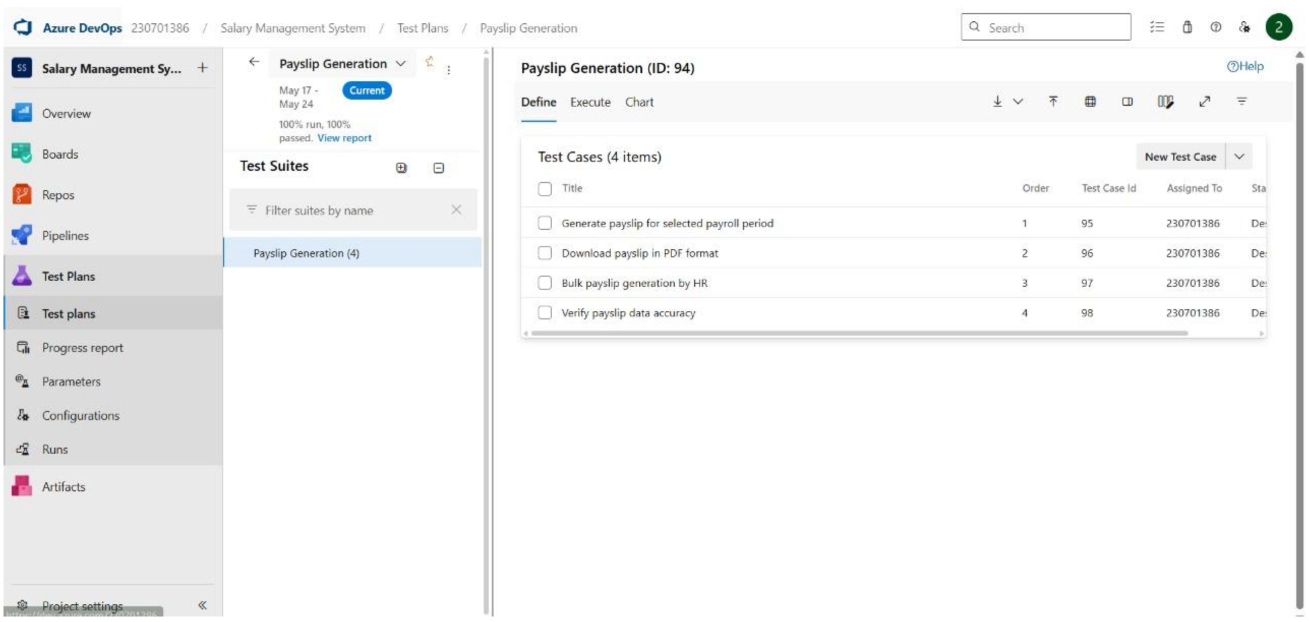
  * Number of test cases passed/failed
  * Percentage of coverage for related user stories
  * Linked bugs or issues discovered during testing

Example Outcome (Optional in Record):

* Test Case TC004: *PASS*

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

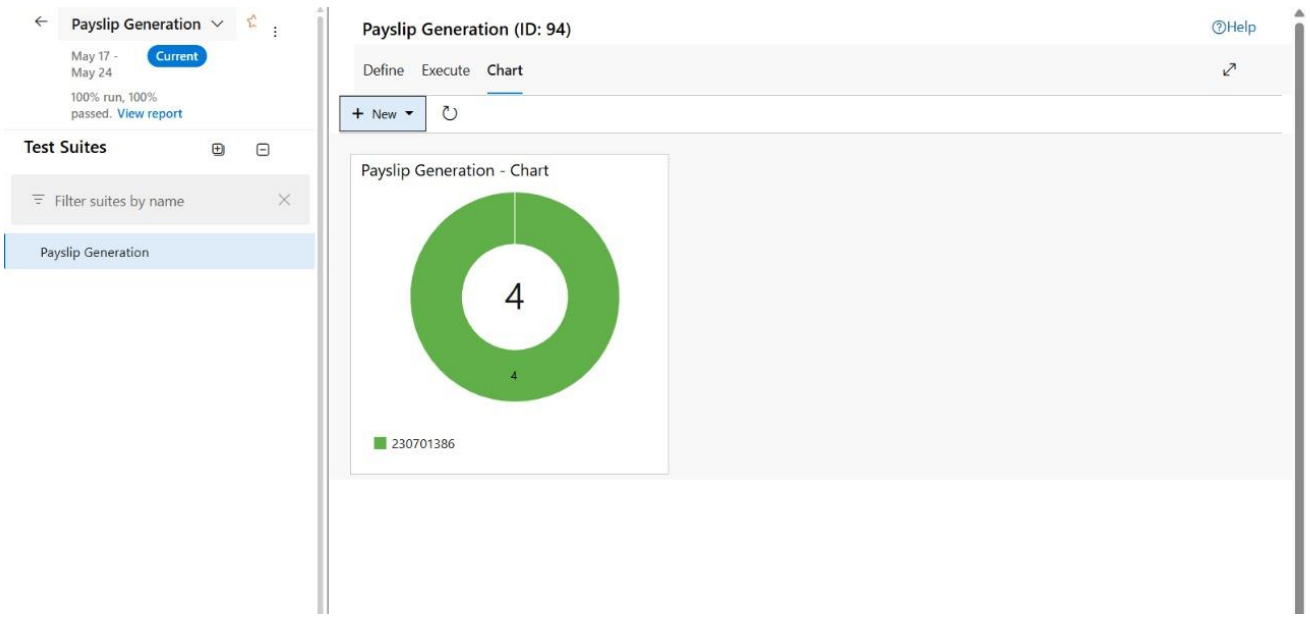
* Remarks: Layout and formatting are acceptable. No issues found.



Payslip Generation (ID: 94)

Test Cases (4 items)

Title	Order	Test Case Id	Assigned To	Sta
Generate payslip for selected payroll period	1	95	230701386	De
Download payslip in PDF format	2	96	230701386	De
Bulk payslip generation by HR	3	97	230701386	De
Verify payslip data accuracy	4	98	230701386	De



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)

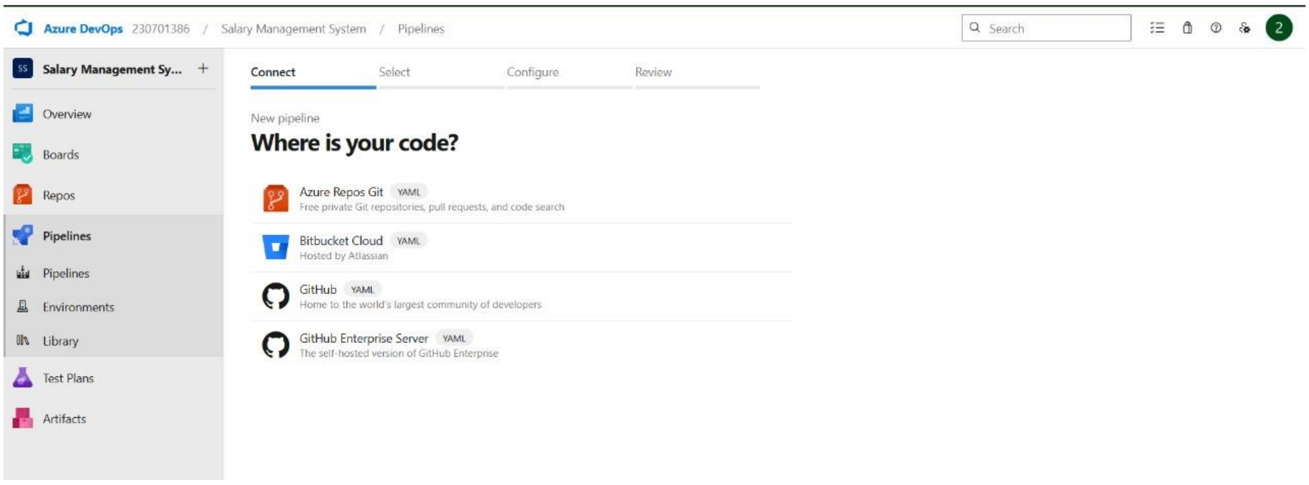
- Add pre-deployment approvals to each stage for review.
- Add gates like API health checks or test validations.

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

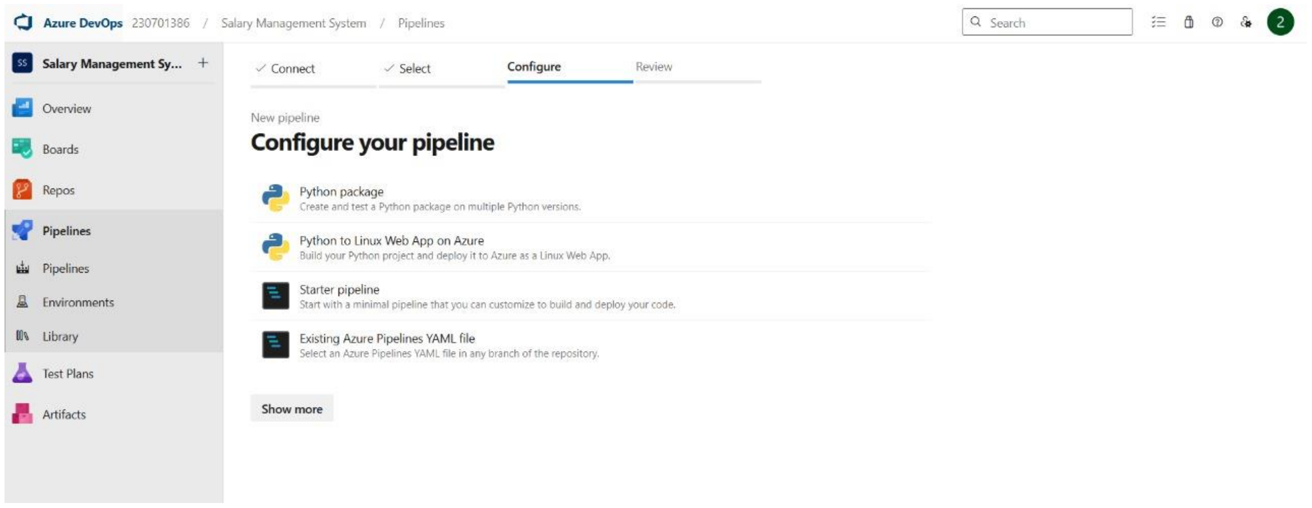
- Track pipeline status under Pipelines → Runs.
- Debug failures and download logs if necessary.
- Use Azure Boards to link builds with user stories and bugs.



New pipeline

Where is your code?

- Azure Repos Git YAML — Free private Git repositories, pull requests, and code search
- Bitbucket Cloud YAML — Hosted by Atlassian
- GitHub YAML — Home to the world's largest community of developers
- GitHub Enterprise Server YAML — The self-hosted version of GitHub Enterprise



New pipeline

Configure your pipeline

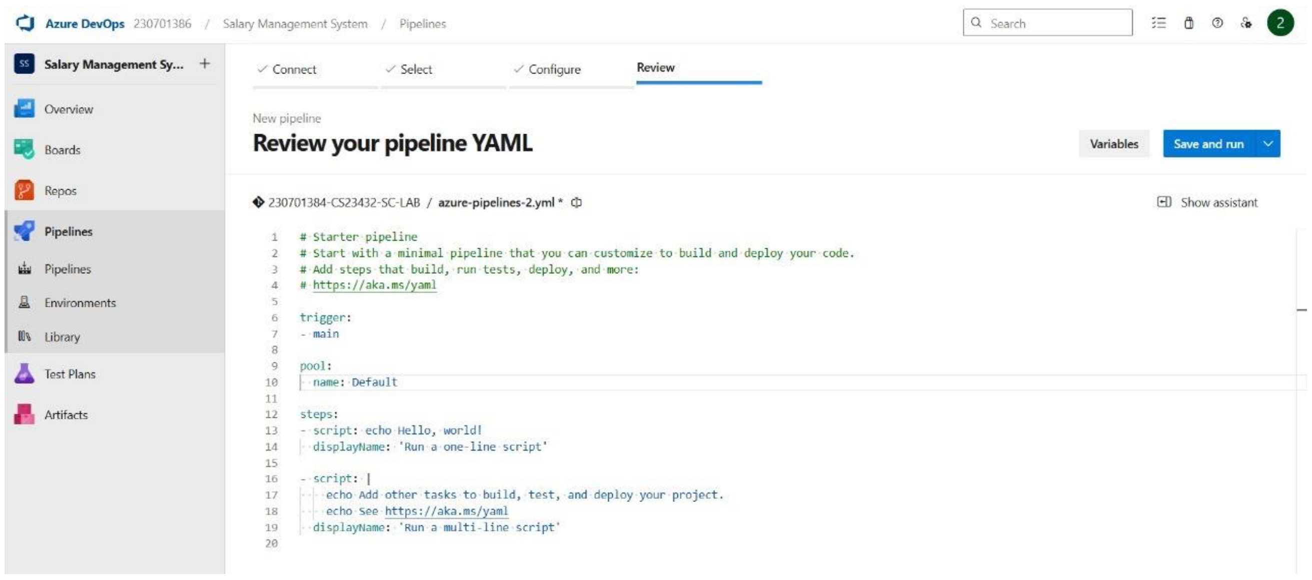
Python package
Create and test a Python package on multiple Python versions.

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

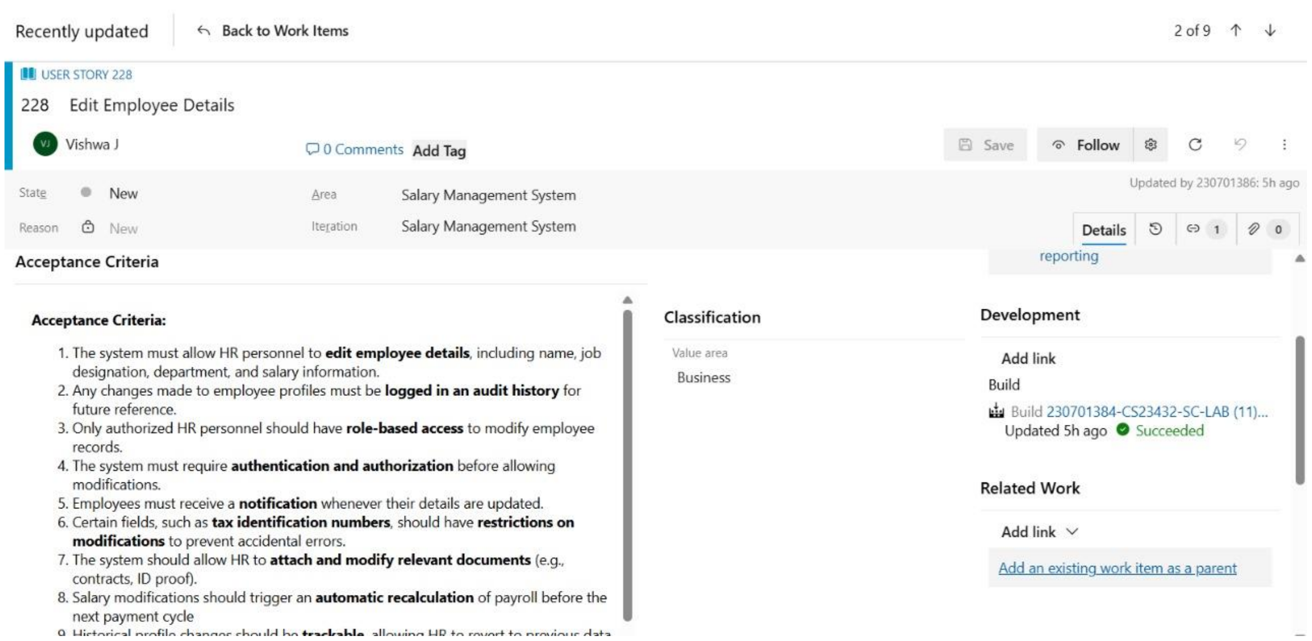


New pipeline

Review your pipeline YAML

230701384-CS23432-SC-LAB / azure-pipelines-2.yml *

```yaml
# 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'
```



USER STORY 228

228 Edit Employee Details

Vishwa J 0 Comments Add Tag

Updated by 230701386: 5h ago

State	New	Area	Salary Management System
Reason	New	Iteration	Salary Management System

reporting

Acceptance Criteria

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

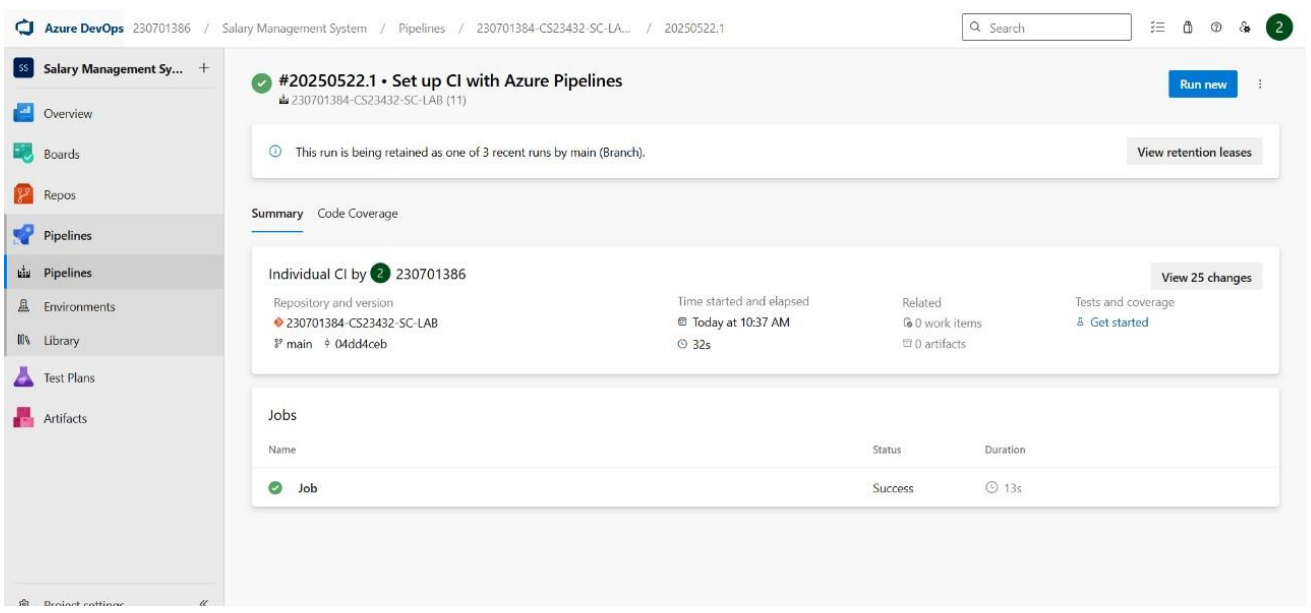
Value area
Business

Development

Build
Build 230701384-CS23432-SC-LAB (11)...
Updated 5h ago ✓ Succeeded

Related Work

Add an existing work item as a parent


Azure DevOps 230701386 / Salary Management System / Pipelines / 230701384-CS23432-SC-LA... / 20250522.1

#20250522.1 • Set up CI with Azure Pipelines
230701384-CS23432-SC-LAB (11)

This run is being retained as one of 3 recent runs by main (Branch).

Individual CI by 230701386

Repository and version: 230701384-CS23432-SC-LAB, main, 04dd4ceb
Time started and elapsed: Today at 10:37 AM, 32s
Related: 0 work items, 0 artifacts
Tests and coverage: Get started

Jobs

Name	Status	Duration
Job	Success	13s

RESULT:

Thus the CI/CD pipeline has been successfully implemented.