Employee Management System - Project Roadmap

Project Description

The Employee Management System is a Java-based application designed to facilitate the management of employee records within an organization. The system allows HR personnel to add, edit, remove employees, and process employee payments efficiently. It also provides a graphical user interface (GUI) that enables users to interact seamlessly with employee data.

Intended Purpose & Audience

This system is intended for small to medium-sized businesses that require a simple, yet powerful, employee management tool. The primary users of this application are HR administrators, managers, and business owners who need an intuitive way to manage employee details, payroll, and salary records. The system will streamline the process of record-keeping, employee tracking, and salary distribution, reducing manual errors and improving efficiency.

Project Outline

Completed Features:

- Implemented Model-View-Controller (MVC) architecture
- Developed GUI using Swing
- Designed Singleton Pattern for the database connection
- Created Employee CRUD operations (Create, Read, Update, Delete)
- Added Strategy Pattern for handling multiple payment methods (Direct Deposit, Check)
- Implemented Factory Pattern for employee creation
- Integrated JUnit 5 tests for core functionalities
- Designed Observer Pattern for real-time UI updates
- Developed Table-based GUI for employee listing with Edit/Delete buttons

In-Progress Features:

- Improve GUI usability by adding validation for inputs
- Implement data persistence using an embedded database (e.g., SQLite)
- Allow employees to select different payment methods dynamically

Planned Features (End of Semester):

- Implement Role-based access control (RBAC) for user authentication
- Generate and export employee payroll reports (PDF/CSV)
- Enhance error handling and implement logging mechanisms
- Provide a search functionality for employee records

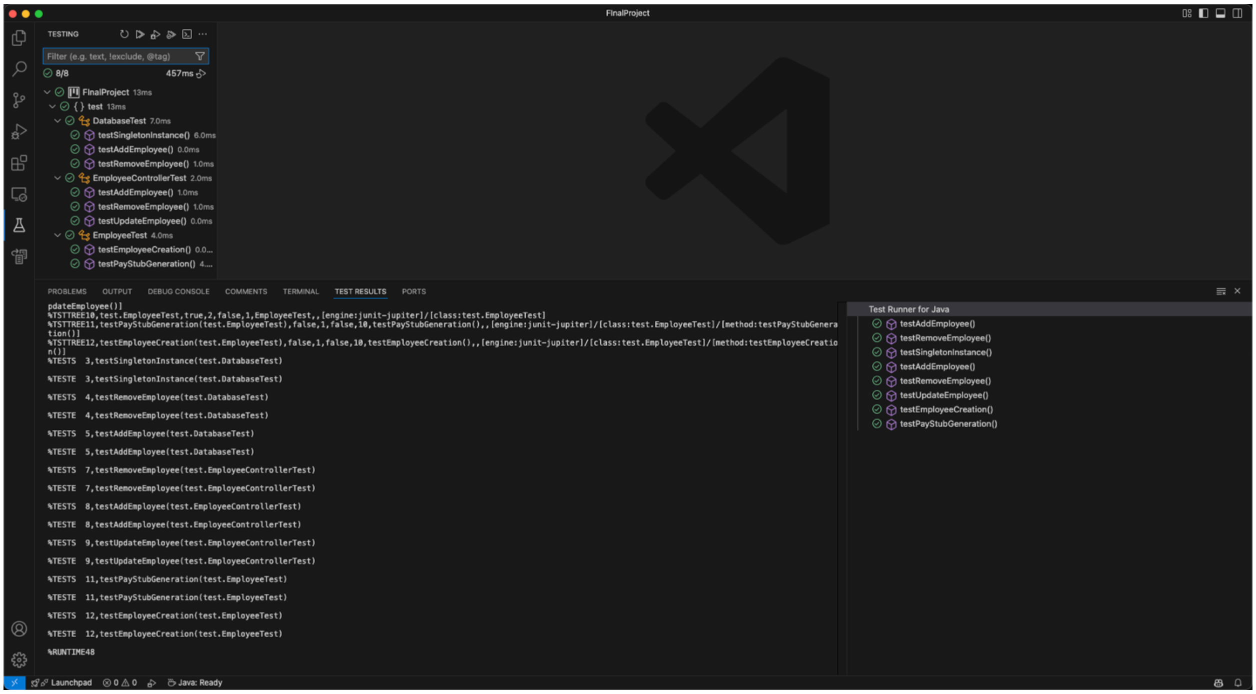
Design Patterns Used

- Singleton Pattern: Ensures a single instance of the database throughout the application.
- Factory Pattern: Used to create employee objects dynamically.
- Strategy Pattern: Allows employees to receive payments via different methods (Direct Deposit, Check).
- Observer Pattern: Enables real-time updates in the GUI when employee data changes.
- Mediator Pattern: Manages communication between the database and GUI through the controller.

Junit Testing

To ensure the reliability and robustness of the Employee Management System, **JUnit 5** has been used for unit testing. The following test cases have been implemented:

- EmployeeControllerTest
  - Verifies the correct addition, removal, and update of employee records.
  - Ensures that the `getAllEmployees()` method returns accurate data.
- DatabaseTest
  - Confirms that the Singleton pattern is correctly applied.
  - Tests the addition and removal of employees in the database.
- EmployeeTest
  - Ensures employee objects are created correctly with the expected attributes.
  - Validates that the paystub generation method includes the correct employee details.



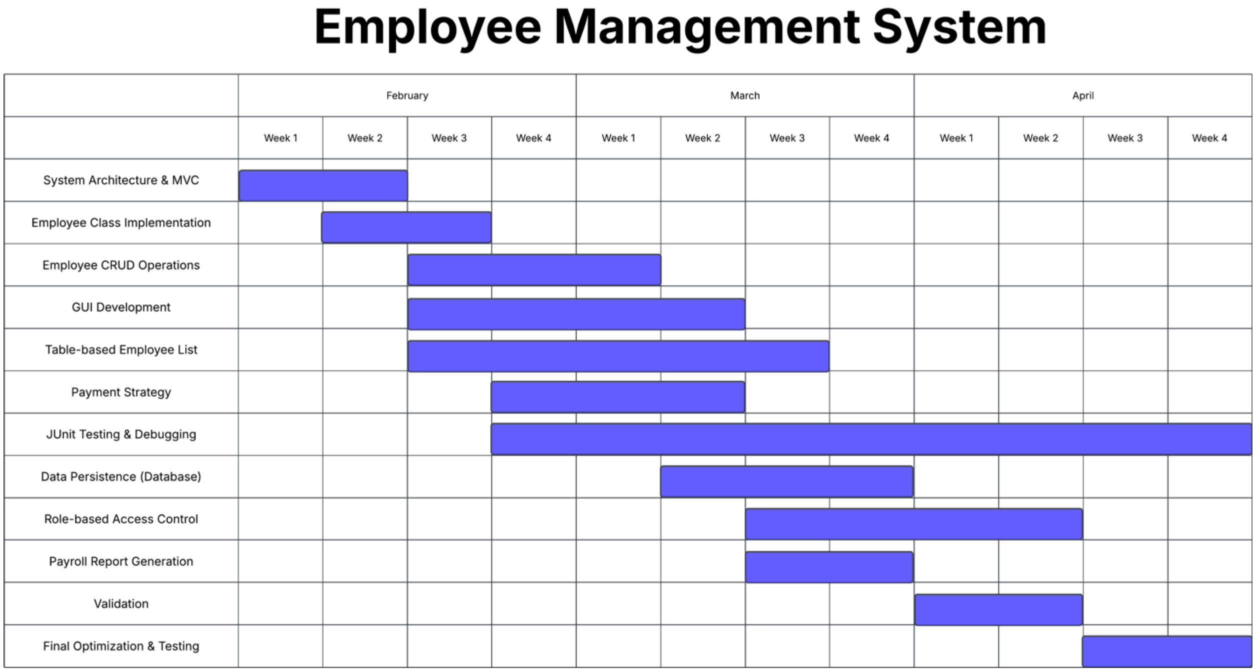
```
8/8                                457ms
FinalProject 13ms
  { } test 13ms
    DatabaseTest 7.0ms
      testSingletonInstance() 6.0ms
      testAddEmployee() 0.0ms
      testRemoveEmployee() 1.0ms
    EmployeeControllerTest 2.0ms
      testAddEmployee() 1.0ms
      testRemoveEmployee() 1.0ms
      testUpdateEmployee() 0.0ms
    EmployeeTest 4.0ms
      testEmployeeCreation() 0.0...
      testPayStubGeneration() 4....
```

```
pdateEmployee()]
%TSTTREE10,test.EmployeeTest,true,2,false,1,EmployeeTest,,[engine:junit-jupiter]/[class:test.EmployeeTest]
%TSTTREE11,testPayStubGeneration(test.EmployeeTest),false,1,false,10,testPayStubGeneration(),,[engine:junit-jupiter]/[class:test.EmployeeTest]/[method:testPayStubGenera
tion()]
%TSTTREE12,testEmployeeCreation(test.EmployeeTest),false,1,false,10,testEmployeeCreation(),,[engine:junit-jupiter]/[class:test.EmployeeTest]/[method:testEmployeeCreatio
n()]
%TESTS  3,testSingletonInstance(test.DatabaseTest)

%TESTE  3,testSingletonInstance(test.DatabaseTest)

%TESTS  4,testRemoveEmployee(test.DatabaseTest)

%TESTE  4,testRemoveEmployee(test.DatabaseTest)

%TESTS  5,testAddEmployee(test.DatabaseTest)

%TESTE  5,testAddEmployee(test.DatabaseTest)

%TESTS  7,testRemoveEmployee(test.EmployeeControllerTest)

%TESTE  7,testRemoveEmployee(test.EmployeeControllerTest)

%TESTS  8,testAddEmployee(test.EmployeeControllerTest)

%TESTE  8,testAddEmployee(test.EmployeeControllerTest)

%TESTS  9,testUpdateEmployee(test.EmployeeControllerTest)

%TESTE  9,testUpdateEmployee(test.EmployeeControllerTest)

%TESTS  11,testPayStubGeneration(test.EmployeeTest)

%TESTE  11,testPayStubGeneration(test.EmployeeTest)

%TESTS  12,testEmployeeCreation(test.EmployeeTest)

%TESTE  12,testEmployeeCreation(test.EmployeeTest)

%RUNTIME48
```

```
Test Runner for Java
  testAddEmployee()
  testRemoveEmployee()
  testSingletonInstance()
  testAddEmployee()
  testRemoveEmployee()
  testUpdateEmployee()
  testEmployeeCreation()
  testPayStubGeneration()
```

These tests help ensure that core business logic operates as expected, reducing bugs and improving maintainability.

Project Timeline



Employee Management System

	February				March				April			
	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
System Architecture & MVC	■	■										
Employee Class Implementation		■	■									
Employee CRUD Operations			■	■	■							
GUI Development			■	■	■	■						
Table-based Employee List			■	■	■	■	■					
Payment Strategy				■	■	■						
JUnit Testing & Debugging				■	■	■	■	■	■	■	■	■
Data Persistence (Database)						■	■	■				
Role-based Access Control							■	■	■	■		
Payroll Report Generation							■	■				
Validation									■	■		
Final Optimization & Testing											■	■

Conclusion

This Employee Management System is progressing towards a fully functional application that simplifies employee record management and payroll processing. The project has been structured to follow best design practices and software engineering principles, making it modular and scalable. By the end of the semester, the goal is to deliver a complete, fully tested, and user-friendly system that meets the needs of HR administrators and business owners.