**ITECH 7201**

**Assignment 2**

**Part B - Group Task**

**Human Resource Management System**

**Group Members:**

**Tanvir Iqbal – 30437681**

**Syeda Tamanna Sheme – 30432670**

**Hansi Nipunika Panwillaarachchi - 30423990**

**Table of Contents**

[1. Introduction 4](#_Toc22238)

2. [Functionality 4](#_Toc22239)

[2.1. Employee Management System Functionality 5](#_Toc22240)

[2.2. Attendance Management System Functionality 5](#_Toc22240)

[2.3. Leave Management System Functionality 5](#_Toc22240)

[2.4. Performance Management System Functionality 5](#_Toc22240)

[2.5. Test Case Functionality 5](#_Toc22240)

[3. Singleton pattern 7](#_Toc22242)

[3.1. Implementation of Singleton pattern 10](#_Toc22245)

[3.2. Explanation of where and why this pattern is used 14](#_Toc22246)

[4. Sate pattern 7](#_Toc22243)

[4.1. Implementation of Sate pattern 10](#_Toc22245)

[4.2. Explanation of where and why this pattern is used 14](#_Toc22246)

[5. Project Presentation Video 10](#_Toc22244)

[6. Conclusion 16](#_Toc22247)

# **Introduction**

The assignment involves developing a Human Resource Management System (HRMS) to centralize and automate HR functions such as employee management, leave management, and attendance management. In Part A, we focused on the Employee Management System, defining the system scenario and purpose, detailing its scope, and developing the domain model. This included creating classes like Employee, Supervisor, and HRManager, along with their relationships and functionalities. We also implemented code for key operations such as creating, updating, and viewing employee details. This work sets the foundation for integrating other subsystems into a comprehensive HRMS.

In this assignment, we will implement the system design created in the first assignment, focusing on our designated subsystems. Each team member will translate use case diagrams, class diagrams, and sequence diagrams into actual Java code for their subsystem. We will incorporate interface and inheritance implementation, as well as design patterns like singleton and state, to enhance the system's structure and efficiency. Additionally, we will develop comprehensive unit tests to validate the functionality of our subsystems and participate in a peer code review process to ensure code quality and adherence to standards. The implementation will be managed using GitHub, ensuring proper version control and collaboration within the team.

## **2. Functionality**

The HRMS (Human Resource Management System) Java project is organized following the Model-View-Controller (MVC) design pattern. It comprises several packages: hrms, hrms.boundary, hrms.control, hrms.entity, tests. The hrms package includes HardCodedData.java for static data. The hrms.boundary package defines interfaces for client interactions related to attendance, employee management, HRMS data, leave requests, and performance reviews. The hrms.control package contains the application logic, with classes like AttendanceControl.java, EmployeeControl.java, HRMSMain.java, LeaveRequestControl.java, and PerformanceReviewControl.java. The hrms.entity package consists of core data structures such as Attendance.java, Employee.java, LeaveRequest.java, LeaveType.java, PerformanceReview.java, and User.java. The tests package includes unit tests, maintaining a similar structure to the main application packages. Additionally, the project integrates JUnit 5 for testing, and a README.md file for documentation. There is also an ITECH7201 project/module in the workspace, containing source files for further coursework or exercises. This organized structure ensures a clear separation of concerns, facilitating maintainability and scalability.

The code in HRMSMain.java is part of a Human Resource Management System (HRMS) and is responsible for handling user login functionality. It includes importing necessary data, initializing lists for employees, attendance, and performance reviews, and setting up the main method to

prompt the user for a username and password. The login method checks the entered credentials against hardcoded user data, printing a success message and displaying the main menu if the login is successful, or an error message if the login fails.

The code in HRMSMain.java successfully implements the login functionality for the Human Resource Management System (HRMS). Upon execution, the program initializes necessary data, including users, employees, attendance records, and performance reviews. The user is prompted to enter a username and password, which are then validated against hardcoded user data. If the credentials are correct, the system grants access and displays the main menu with options to view employee details, attendance, leave, performance reviews, and an option to log out. If the credentials are incorrect, an error message is displayed, prompting the user to try again. The code ensures that the login process is handled securely and efficiently, providing appropriate feedback to the user based on the outcome.

In the first screenshot, the console output shows a failed login attempt with the username lm1 and password bb, resulting in the message "Invalid username or password. Please try again." In the second screenshot, a successful login attempt is made with the username lm1 and password x, resulting in the message "Login successful." This is followed by the display of the main menu options, allowing the user to choose from various HRMS functionalities.

A computer screen shot of a program

Description automatically generated



**2.1. Employee Management System Functionality**

The provided screenshots display the functionality of an Employee Details module within a Human Resource Management System (HRMS). This module allows users to access detailed information about employees. The main menu offers several options, including viewing Employee Details, Attendance, Leave, Performance, and Logout. When the "Employee Details" option is selected, a sub-menu is presented, enabling the user to view details for "Self," "Subordinate (All)," "Subordinate (Employee)," or go back to the main menu. This hierarchical structure ensures easy navigation and focused access to specific employee information based on the user's role and permissions.

A computer screen shot of a program

Description automatically generated

**Viewing Self and Subordinate Details**

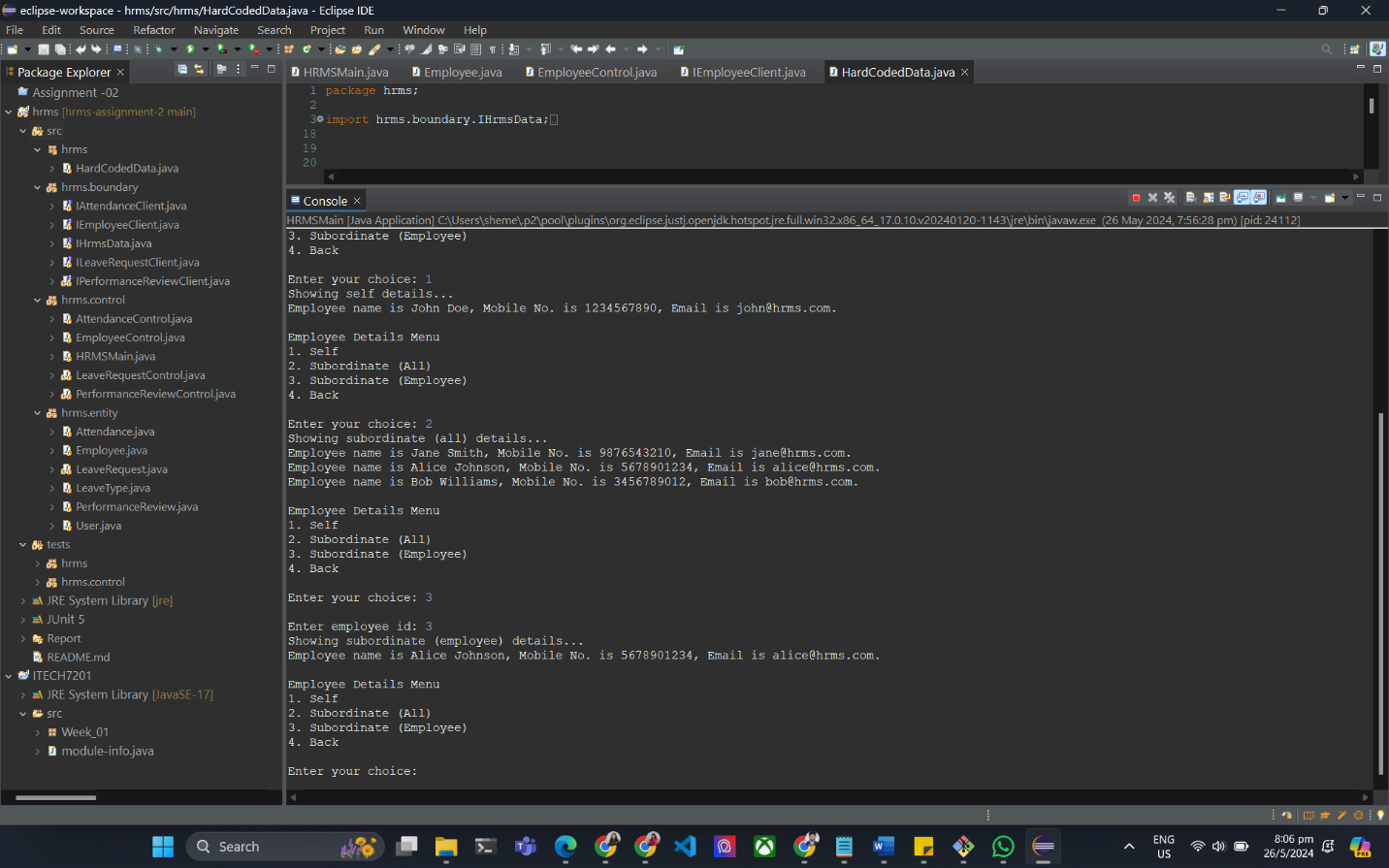
When the user opts to view "Self" under the Employee Details menu, the system displays their personal information, such as name, mobile number, and email. In the screenshots, the system shows details for an employee named John Doe, including his mobile number (1234567890) and email (john@hrms.com). If the user selects "Subordinate (All)," the system lists details for all subordinates under the user’s supervision. For instance, it shows details for Jane Smith, Alice Johnson, and Bob Williams, including their respective contact information. This functionality aids supervisors and managers in keeping track of their team members' information efficiently.

A computer screen shot of a black screen

Description automatically generated

**Viewing Specific Employee Details**

The "Subordinate (Employee)" option allows the user to view details of a specific subordinate by entering the employee's ID. For example, entering the ID for Alice Johnson displays her details, including her mobile number (5678901234) and email (alice@hrms.com). This targeted approach helps in quickly accessing individual employee information without sifting through all subordinates. The final part of the Employee Details menu provides an option to go back to the main menu, ensuring smooth navigation. This comprehensive approach to managing employee information demonstrates the system's capability to centralize and streamline HR-related data, facilitating better management and accessibility.



**2.2. Attendance Management System Functionality**

The screenshots illustrate the Attendance module within the Human Resource Management System (HRMS). Upon logging in, the user is presented with a main menu offering various options, including Employee Details, Attendance, Leave, Performance, and Logout. Selecting the Attendance option leads to a submenu where users can view their own attendance details, the attendance details of all subordinates, or the attendance of a specific subordinate. This structured navigation ensures that users can access detailed attendance records efficiently.

A computer screen with many colorful lines

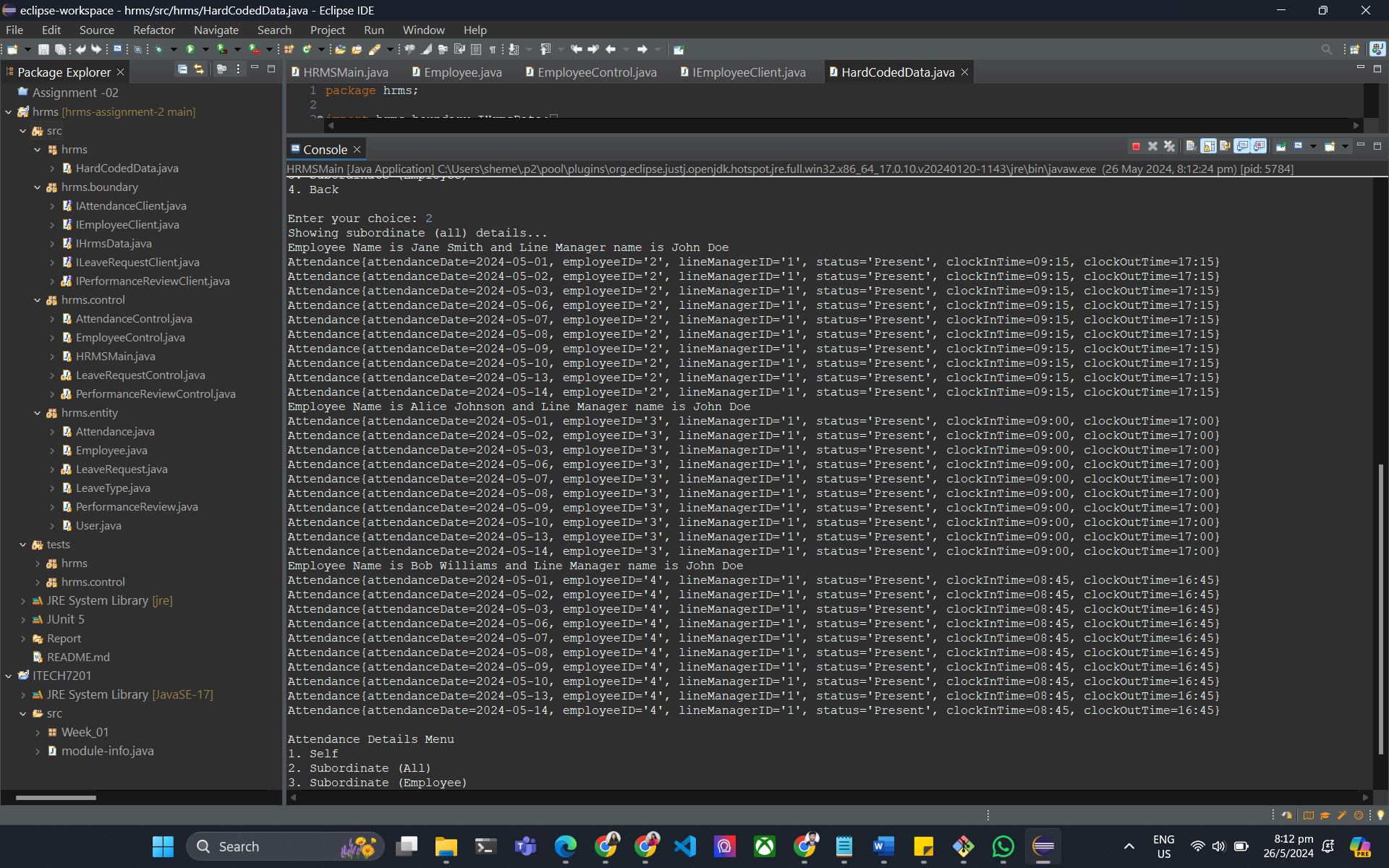
Description automatically generated

A computer screen shot of a program

Description automatically generated

**Viewing Self and Subordinate Attendance Details**

When the user selects "Self" in the Attendance submenu, the system displays the user's attendance records. For example, the screenshots show attendance details for John Doe, with each entry listing the attendance date, employee ID, line manager ID, status (e.g., 'Present'), clock-in time, and clock-out time. This detailed breakdown helps employees keep track of their daily attendance. Similarly, when the user opts to view "Subordinate (All)," the system provides attendance records for all subordinates under the user's supervision. This view includes details for multiple employees, such as Jane Smith, Alice Johnson, and Bob Williams, facilitating comprehensive oversight of team attendance.



**Viewing Specific Subordinate Attendance**

The "Subordinate (Employee)" option allows the user to view the attendance details of a specific subordinate by entering the employee's ID. For instance, entering the ID for Alice Johnson displays her attendance records, including attendance dates, status, clock-in times, and clock-out times. This feature enables managers to monitor the attendance patterns of individual employees closely, identifying any irregularities or patterns that might require attention. The final option in the Attendance submenu is to go back to the main menu, ensuring that users can easily navigate back and forth between different modules within the HRMS. This functionality showcases the system's capability to provide detailed and accessible attendance tracking, essential for effective human resource management.

A computer screen shot of a computer screen

Description automatically generated

A computer screen shot of a black screen

Description automatically generated

**2.3. Leave Management System Functionality**

The screenshots provide a detailed view of the Leave module within the Human Resource Management System (HRMS). Upon logging in, the user is greeted with a main menu offering options such as Employee Details, Attendance, Leave, Performance, and Logout. When the Leave option is selected, the user can access a submenu that includes options for viewing leave details for "Self," "Subordinate (All)," "Subordinate (Employee)," or returning to the main menu. This setup allows for organized navigation, making it straightforward to access specific leave information based on user roles and permissions.

A computer screen with many colorful text

Description automatically generated

**Viewing Self and Subordinate Leave Details**

Selecting the "Self" option under the Leave submenu displays the leave records for the logged-in user. For example, the screenshots show leave details for John Doe, including the start date, end date, reason (e.g., 'Annual Leave'), total days, and status. This information helps users keep track of their own leave history and status. Similarly, the "Subordinate (All)" option provides a comprehensive view of leave records for all subordinates under the user's supervision. This view includes leave details for various employees, such as Bob Williams, Emma Brown, and Michael Davis, enabling supervisors to oversee and manage the leave status of their entire team effectively.

A computer screen shot of a black screen

Description automatically generated

A computer screen shot of a black screen

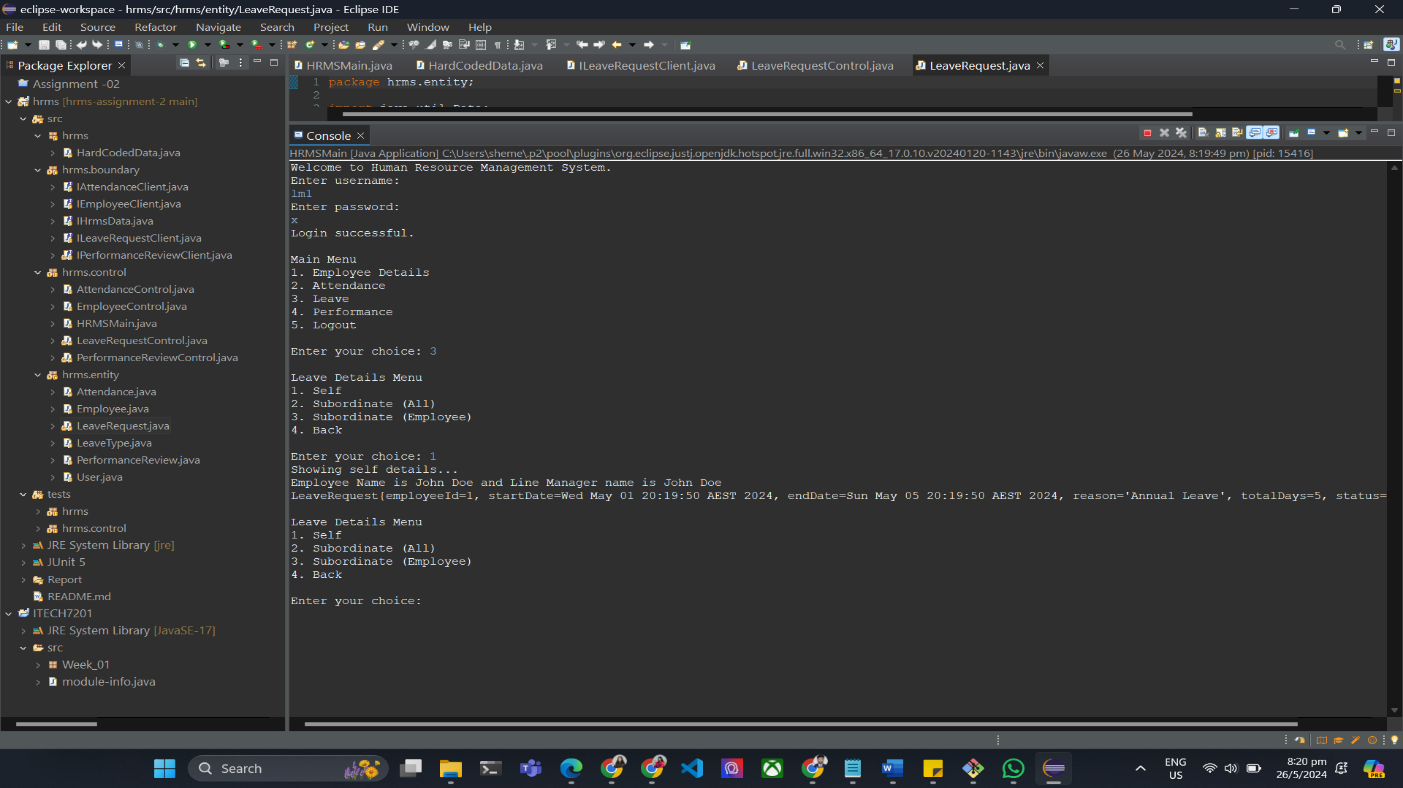
Description automatically generated

**Viewing Specific Subordinate Leave**

The "Subordinate (Employee)" option allows the user to view the leave details of a specific subordinate by entering the employee's ID. For instance, entering the ID for a subordinate provides details such as start date, end date, reason for leave, total days, and status. This functionality is useful for managers who need to monitor and approve individual leave requests, ensuring proper staffing levels and managing workload distribution. The system also indicates if there are no leave records found for a specific employee and line manager combination, as seen in one of the screenshots. This feature ensures that managers have complete and accurate information to make informed decisions regarding employee leave requests.

A computer screen shot of a program

Description automatically generated



**2.4. Performance Management System Functionality**

The screenshots illustrate the Performance Review module within the Human Resource Management System (HRMS). Upon successful login, the user is presented with a main menu offering several options, including Employee Details, Attendance, Leave, Performance, and Logout. Selecting the Performance option directs the user to a submenu where they can view performance reviews for "Self," "Subordinate (All)," "Subordinate (Employee)," or return to the main menu. This hierarchical structure allows users to efficiently navigate through the system to access relevant performance review data.

A computer screen shot of a program

Description automatically generated

**Viewing Self and Subordinate Performance Reviews**

When the user selects "Self" under the Performance Review submenu, the system displays their own performance review details. For example, the screenshots show performance review information for John Doe, including employee ID, line manager ID, review date, feedback, rating, and goals. This information is essential for employees to understand their performance evaluations and areas for improvement. Similarly, selecting "Subordinate (All)" provides a comprehensive view of performance reviews for all subordinates under the user's supervision. This view includes detailed performance reviews for various employees such as Jane Smith, Alice Johnson, and Bob Williams, enabling supervisors to monitor and manage their team's performance effectively.

A computer screen with text on it

Description automatically generated

A computer screen with a black screen

Description automatically generated

**Viewing Specific Subordinate Performance Reviews**

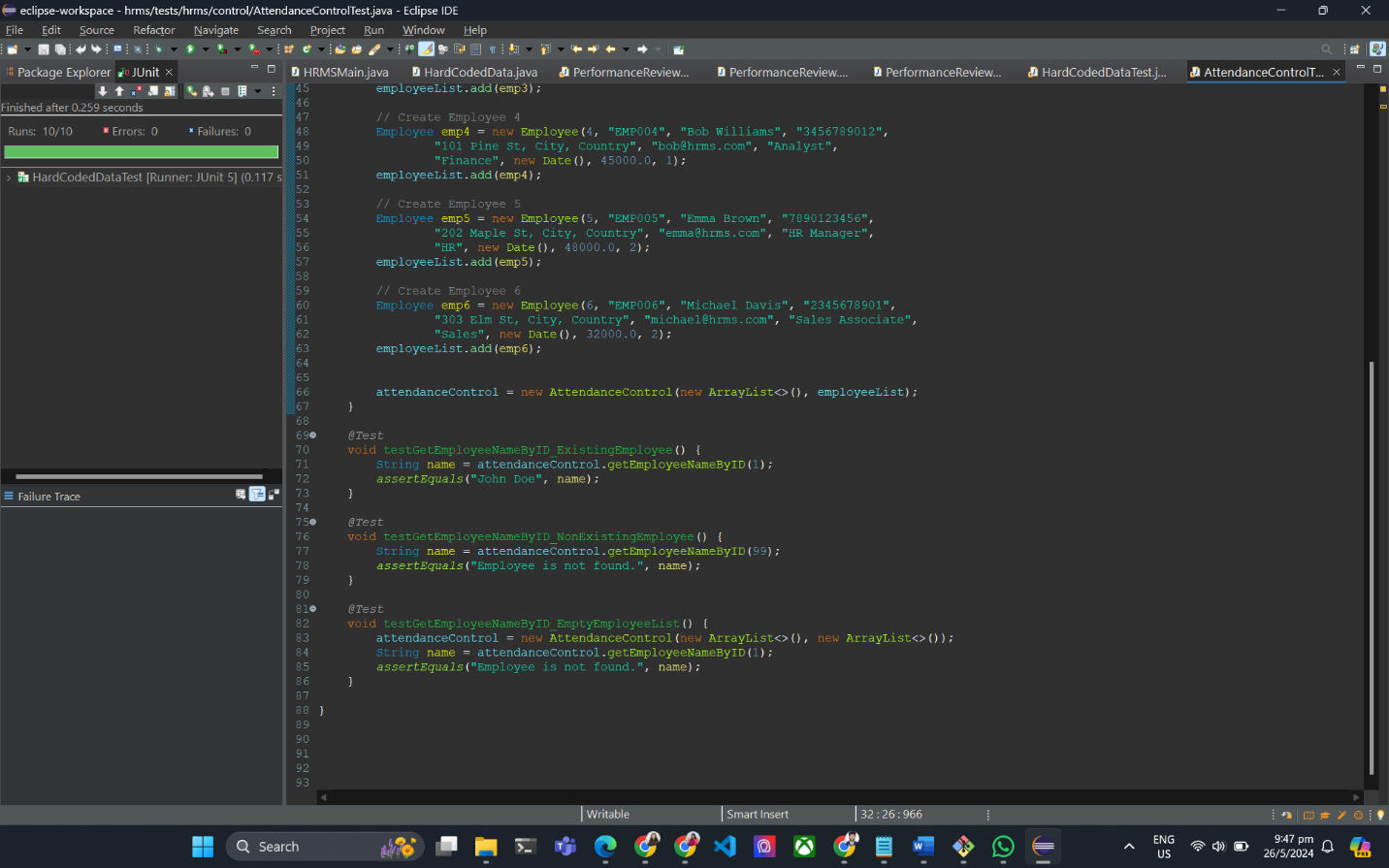
The "Subordinate (Employee)" option allows the user to view the performance review details of a specific subordinate by entering the employee's ID. For instance, entering the ID for Alice Johnson displays her performance review, including review date, feedback, rating, and goals. This targeted approach helps managers provide personalized feedback and set specific goals for individual employees. The system also indicates if there are no performance review records found for a specific employee and line manager combination, as seen in one of the screenshots. This functionality ensures that managers have detailed and accurate information to guide their team's development and performance improvement efforts.

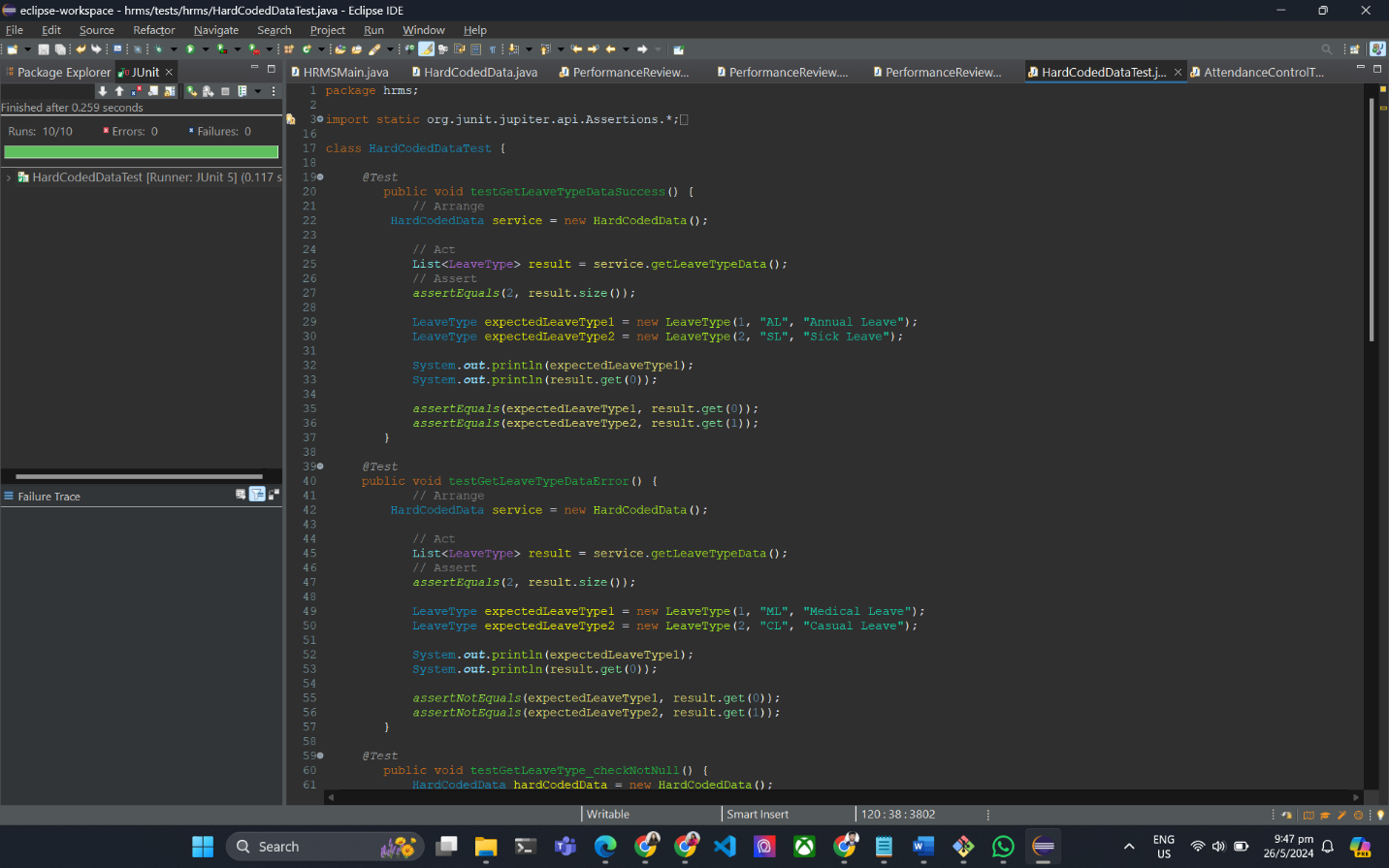
A computer screen with a black screen

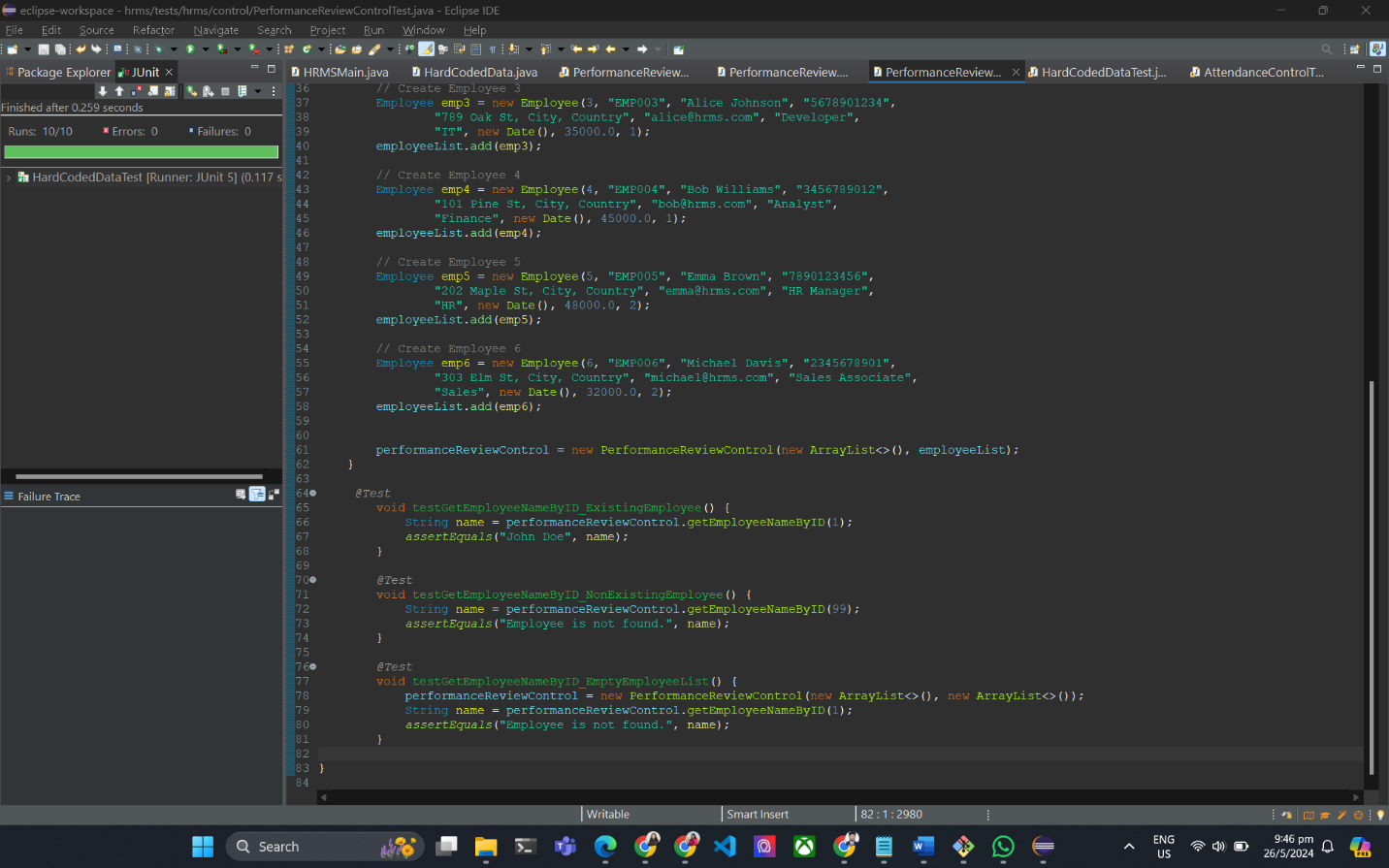
Description automatically generated

[**2.5. Test Case Functionality**](#_Toc22240)

The provided screenshots and file depict test cases for various functionalities within the HRMS system, focusing on employee, leave, attendance, and performance review modules. These test cases are implemented using the JUnit framework to ensure that methods work correctly and produce the expected outcomes. For employees, tests validate the getEmployeeNameByID method, checking responses for existing, non-existing, and empty employee lists, ensuring correct data retrieval and error handling. For leave types, tests verify the getLeaveTypeData method, confirming successful data retrieval, handling data discrepancies, and performing null checks. Attendance tests also focus on the getEmployeeNameByID method, validating the correct responses for different employee scenarios. Performance review tests check the getEmployeeNameByID method, ensuring accurate performance review data retrieval and handling of various employee states. These comprehensive test cases ensure the robustness and reliability of the HRMS system across multiple modules and scenarios.







**3. Singleton pattern**

**3.1. Implementation of Singleton Pattern**

**i) Implementation of Singleton Pattern in the HRMSMain Class**

To implement the Singleton pattern in the HRMSMain class, follow these steps to ensure that there is only one instance of the class and provide a global point of access to it:

* **Private Constructor:** Make the constructor private to prevent instantiation from outside the class.
* **Static Instance:** Create a private static instance of the class.
* **Public Static Method:** Provide a public static method to get the instance.

Here is the implementation based on these steps:

**In this implementation:**

1. The constructor of HRMSMain is private to prevent creating instances from outside the class.  
   A close up of a sign

   Description automatically generated
2. A private static variable instance holds the single instance of HRMSMain.  
   
3. The public static method getInstance() returns the single instance, creating it if it doesn't already exist.  
   A screen shot of a computer code

   Description automatically generated

Now, the HRMSMain class is a singleton and can only be instantiated once, ensuring that only one instance exists throughout the application.

A black and white text

Description automatically generated

In this implementation, the constructor of HRMSMain is private to prevent creating instances from outside the class. A private static variable instance holds the single instance of HRMSMain. The public static method getInstance() returns the single instance, creating it if it doesn't already exist. This ensures that HRMSMain can only be instantiated once, maintaining a single instance throughout the application.

* 1. **Explanation of Where and Why the Singleton Pattern is Used**

The Singleton pattern is used in the HRMSMain class to ensure that only one instance of this class exists across the entire application. This is crucial for the following reasons:

* **Centralized Control:** Having a single instance allows centralized control over the application's main operations, ensuring consistency and coordination.
* **Resource Management:** It helps in managing resources efficiently, as only one instance of the class handles operations, reducing the overhead of multiple instances.
* **Global Access Point:** It provides a global point of access to the instance, simplifying the interaction with the main class from different parts of the application.

Using the Singleton pattern in HRMSMain ensures that the application operates smoothly with a single point of management, enhancing maintainability and reliability.

**4. Sate pattern**

**4.1. Implementation of State Pattern**

**i) Implementation of State Pattern in the HRMSMain Class**

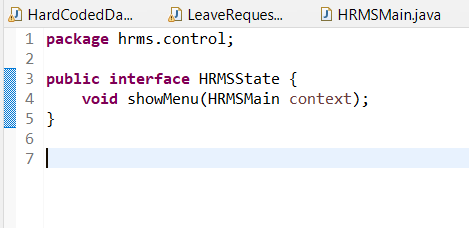
The State pattern is useful when an object needs to change its behavior based on its internal state. In the HRMSMain class, it can be applied to manage the user authentication states (logged in and logged out).

Here is a brief outline of how to implement the State pattern:

* **State Interface:** Define an interface that represents the state of the application.
* **Concrete States:** Implement concrete state classes for the logged-in and logged-out states.
* **Context Class:** Modify the HRMSMain class to manage the current state and delegate behavior to the current state object.

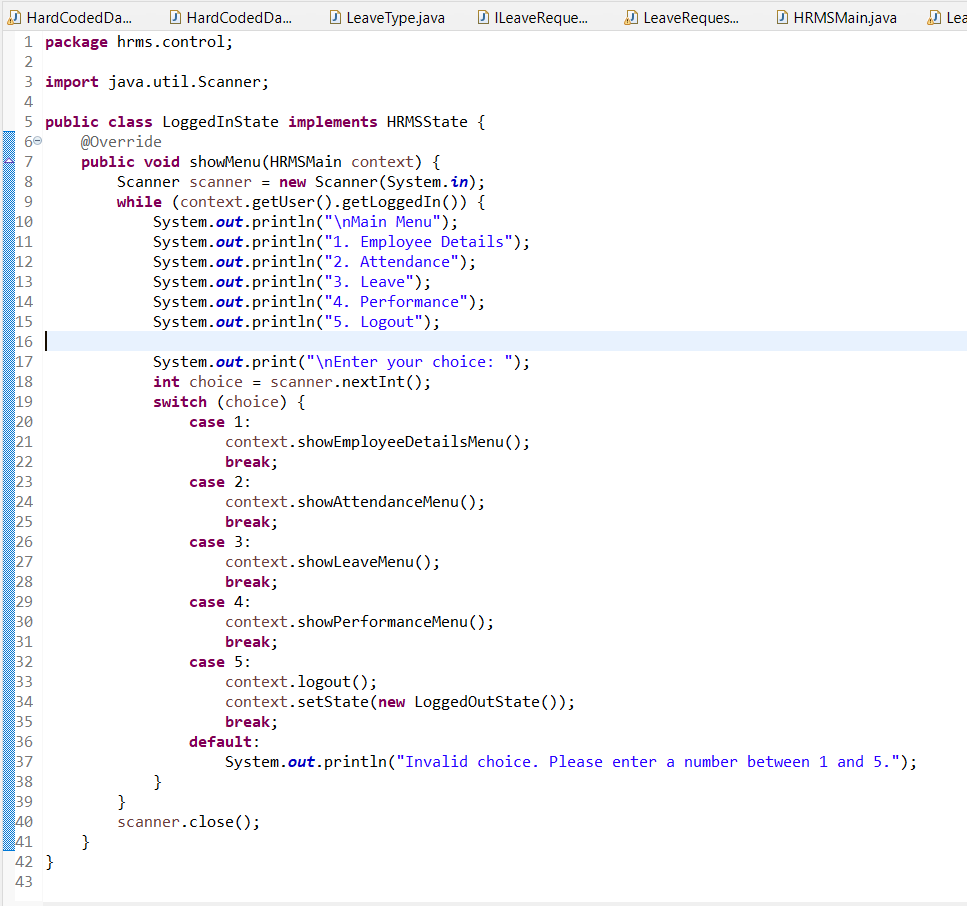
Step 1: Define the State Interface

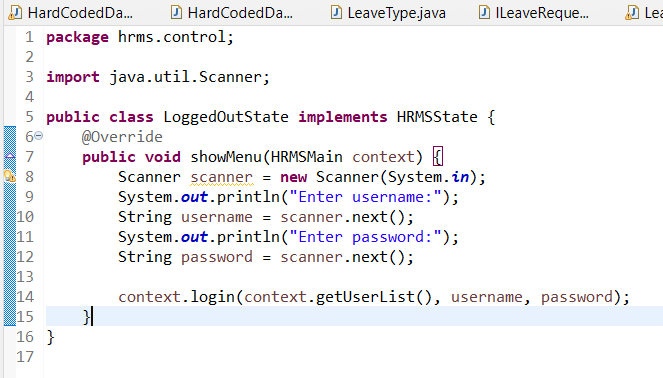
Define an interface HRMSState that declares the methods which will be implemented by the concrete states.



**Step 2: Implement Concrete State Classes**

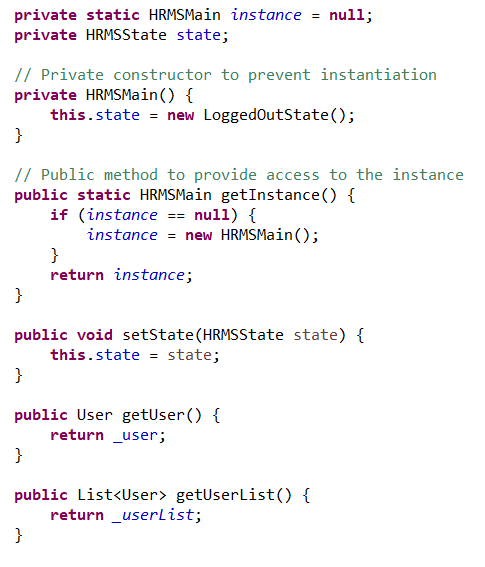
Implement the concrete state classes for logged-in and logged-out states.

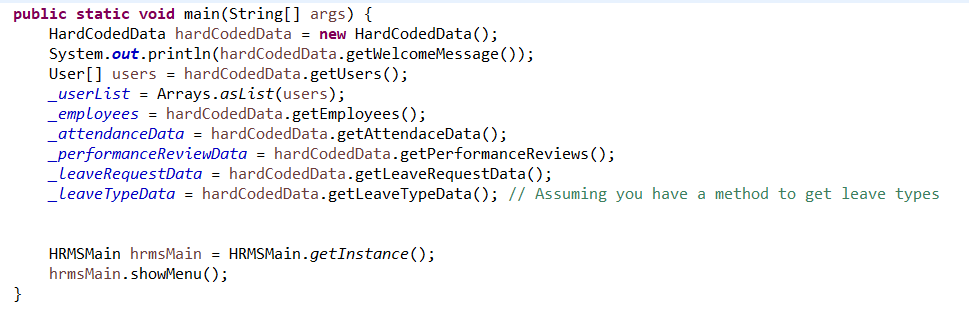


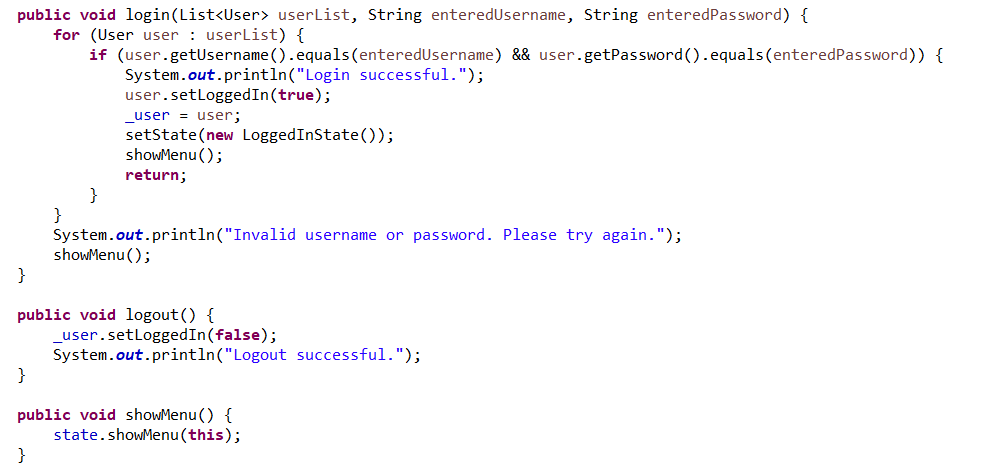


**Step 3: Modify the HRMSMain Class**

Modify the HRMSMain class to use the State pattern.







This implementation ensures that the behavior of the HRMSMain class changes based on the user's authentication state, improving code organization and readability.

**4.2 Explanation of Where and Why the State Pattern is Used**

The State pattern is used in the HRMSMain class to manage the different states of user authentication, specifically the logged-in and logged-out states. This is beneficial for the following reasons:

* **Behavioral Flexibility: It** allows the HRMSMain class to change its behavior dynamically based on the user's authentication state.
* **Improved Readability:** Separating the state-specific behavior into different classes improves the readability and maintainability of the code.
* **Scalability:** The pattern makes it easy to add new states or modify existing ones without changing the context class, enhancing scalability.

Using the State pattern in managing user authentication states helps in organizing the code better and allows for easier modifications and extensions in the future.

[**5. Project Presentation Video**](#_Toc22244)

[**6. Conclusion**](#_Toc22247)

In conclusion, our development of the Human Resource Management System (HRMS) has effectively centralized and automated key HR functions, such as employee management, leave management, attendance management, and performance reviews. Through the implementation of the Singleton and State design patterns, we ensured efficient resource management and behavioral flexibility, enhancing the system's maintainability and scalability. The Singleton pattern was crucial in maintaining a single instance of the HRMSMain class, providing centralized control and reducing the overhead of multiple instances. Meanwhile, the State pattern allowed for dynamic behavior changes based on user authentication states, improving code organization and readability. Our comprehensive unit tests, validated using the JUnit framework, confirmed the reliability and robustness of the system across various modules and scenarios. By integrating these design patterns and rigorous testing, we have created a robust and scalable HRMS that streamlines HR processes and improves overall efficiency.

### 