Gyalpozhing College of Information Technology

Royal University of Bhutan

Kabjisa, Chamjekha, Thimphu

**AI and Data Science**

**CSF304 Design Patterns**

**Final Report On Student Attendance Management System**

Group members:

Pema Chozom, 12210024

Parjeet Mongar, 12210022

Jigme Tenzin, 12210011

Tenzin Namgyal Yeshey, 12210099

**Tutor :** Mrs. Pema Wangmo

[**1. Project Title:**](#_43392l5btnq) **3**

[**2. Brief Project Description:**](#_la0q044ikbpc) **3**

[**4. Source Code:**](#_cqui8wwndjf1) **5**

[**5. User Interface: (How to use it):**](#_u9bhp791sc42) **5**

[**6. All Class Diagrams:**](#_swi1y8xe29vb) **12**

[**7. Justification for all the design pattern used:**](#_rs3zbegl3uat) **13**

[**8. How the framework is converted into jar files**](#_lk8sp47r5831) **14**

[**9. Challenges:**](#_i8k97zuncg1k) **17**

[**10. Conclusion:**](#_oc9y7kpj84eq) **17**

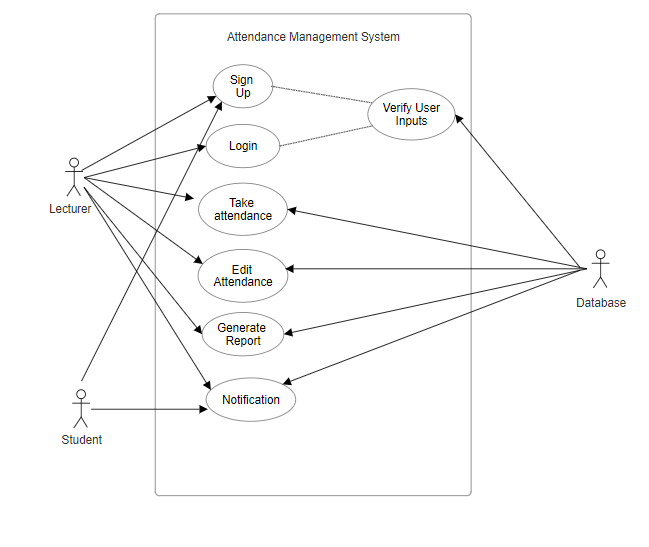
# **Project Title:**

Student Attendance Management

# **Brief Project Description:**

The Student Attendance Management System is a project designed to automate and streamline the process of recording and managing student attendance. The system will provide functionalities for students and lecturers to track attendance on the particular date for each module (Students), generate reports and take attendance (lecturer).

1. **Use Cases(Functionality):**



1. Signup/Registration:

Actors: Lecturers

Description: New users can create an account by providing their personal and authentication details.

1. Login/Authentication:

Actors: Students, Teachers

Description: Users can log in using their credentials to access the system.

1. Mark Attendance:

Actors: Lecturers

Description: Lecturers who are registered in a particular module can mark attendance for their classes.

1. View Attendance:

Actors: Students

Description: Users can view attendance records for specific dates for all the modules they are enrolled in.

1. Generate Reports:

Actors: Students

Description: The system can generate detailed attendance reports for individual students for weekly and monthly basis.

1. Edit Attendance

Actors: Lecturers

Description: Lecturers can edit attendance records for their classes. They can update the status of students (present or absent). Lecturers select a date to mark attendance. Interface allows marking students as present or absent.

1. Notification

Actors: Student

Description: Message on students page is shown if the student's attendance is less than 90 percent.

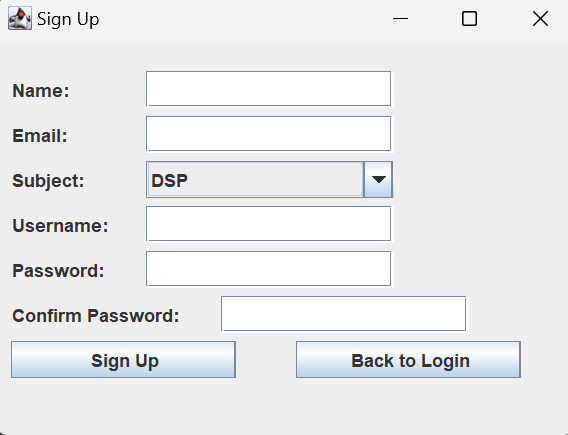
# Source Code:

<https://gitlab.com/g34710822/attendancesystem>

# User Interface: (How to use it):

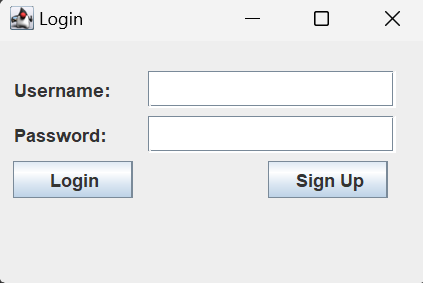
1. Signup Page:

Lecturers enter their personal information (name, email, module they are teaching(in our case we have 4 modules - PRJ, DSP, ADL & UIUX), username) and set a password.



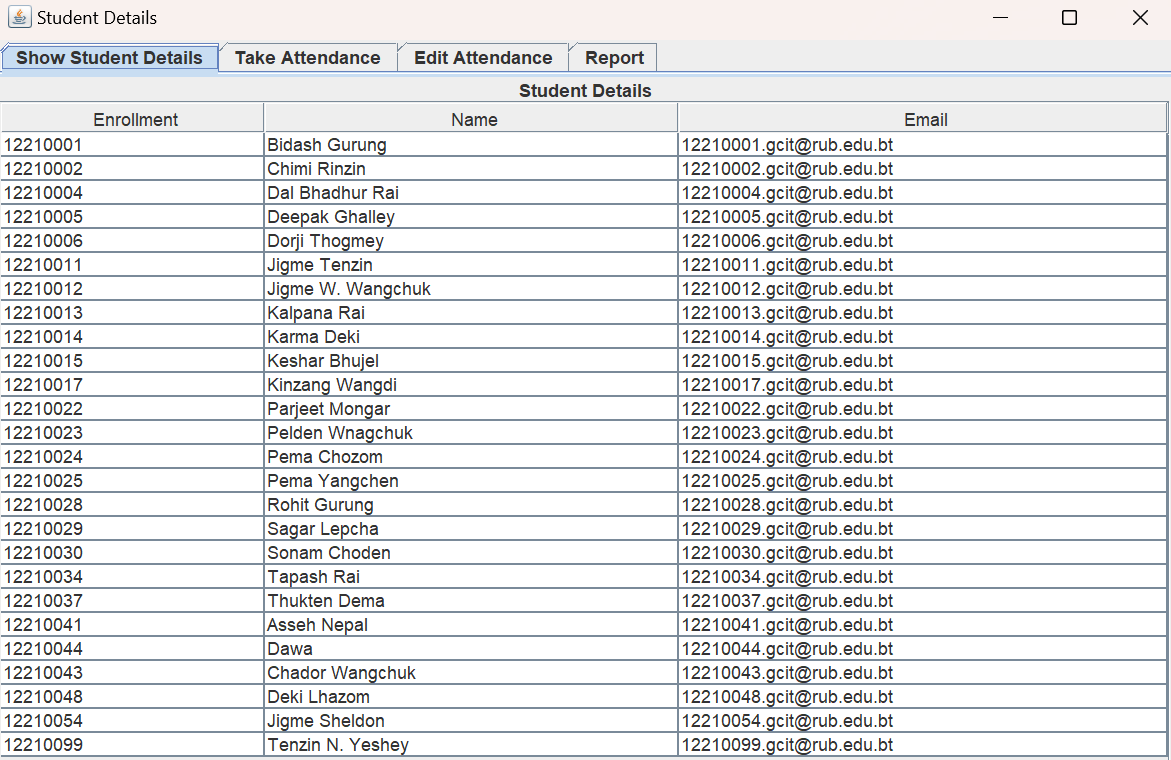
1. Login Page:

Users (Student or Lecturers) enter their username and password to log in.



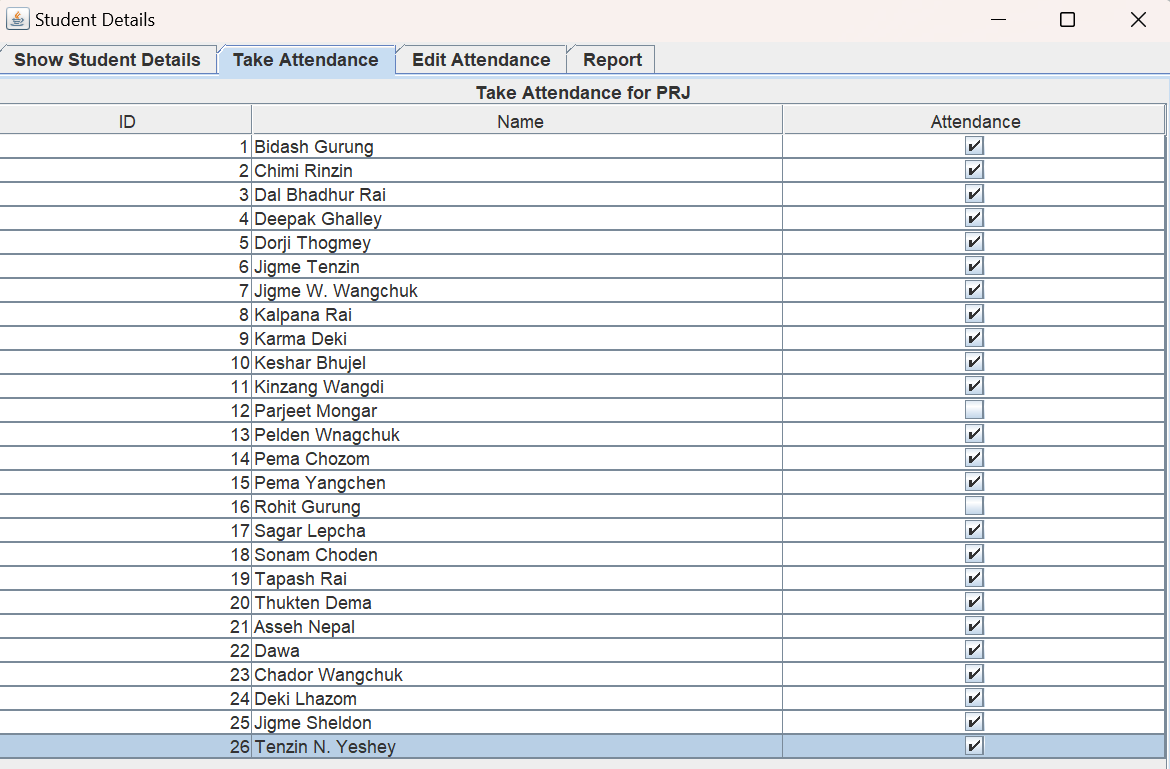
1. View Student details

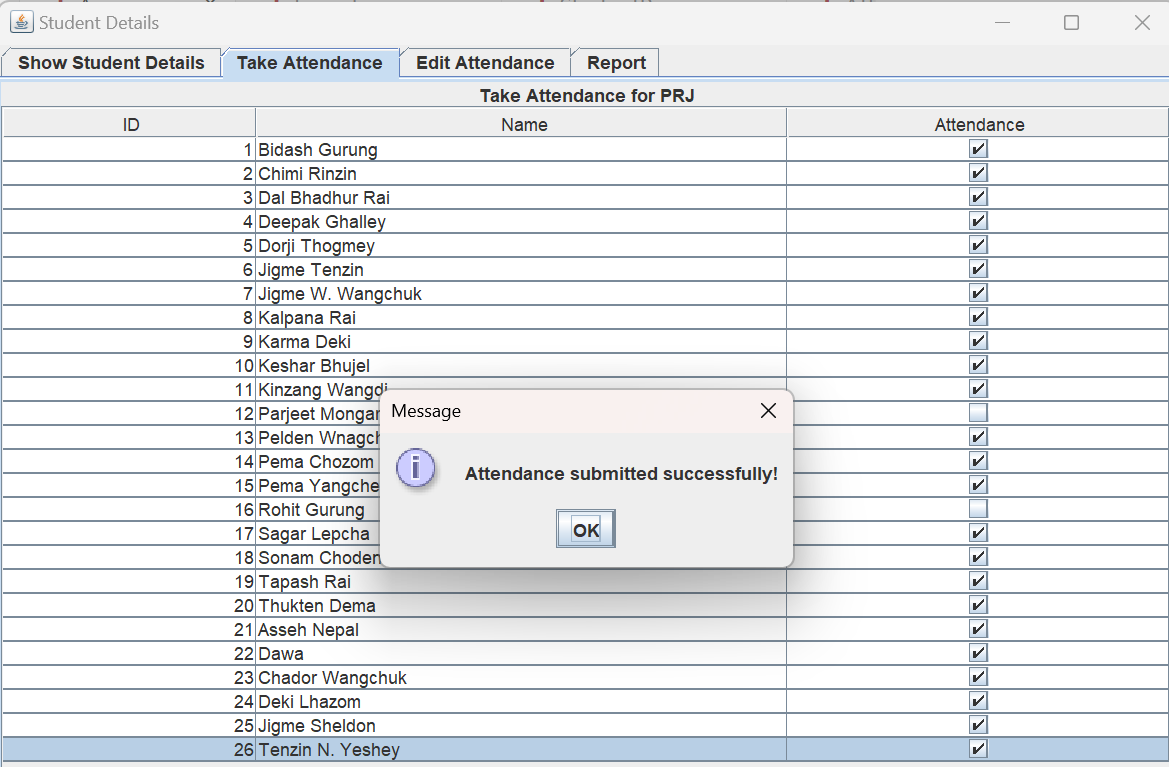
If the user is a lecturer, they can view student’s detail where the students enrolled in a particular module are shown with their enrollment number, name and email address.



1. Take Attendance

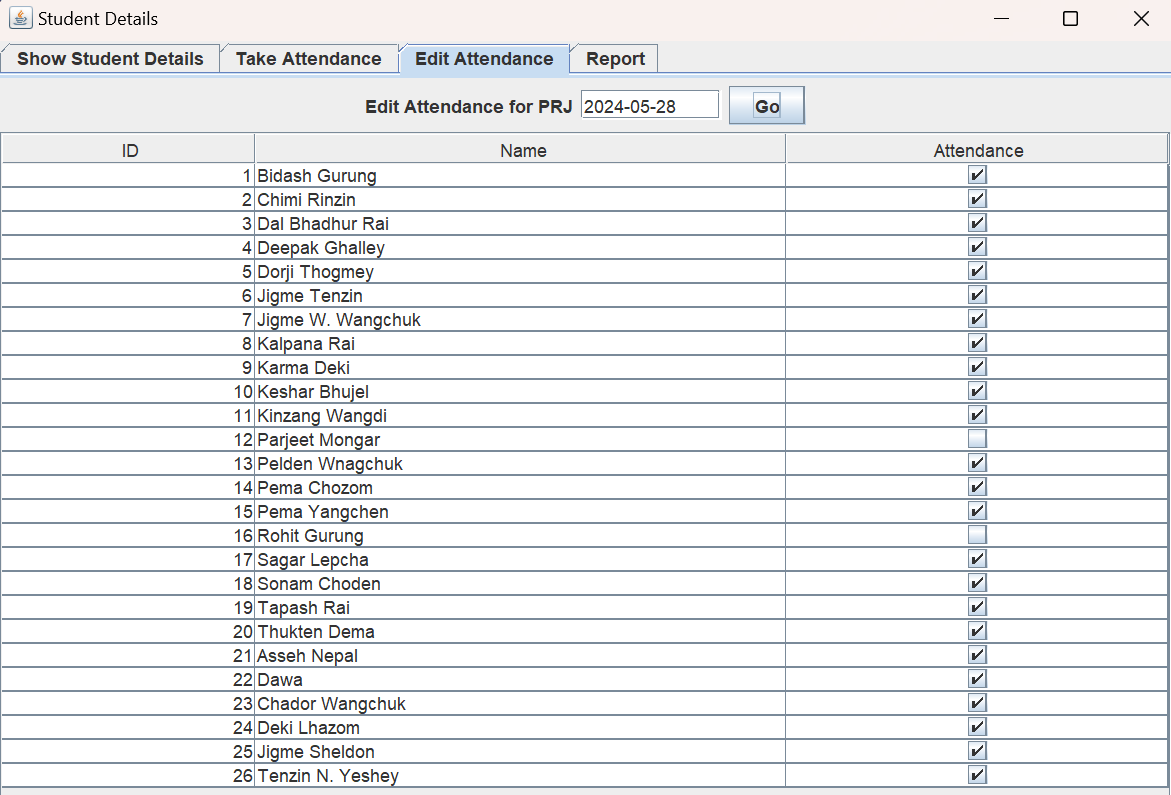
As a lecturer, they can take attendance for the students who are absent or present and save it in the database.

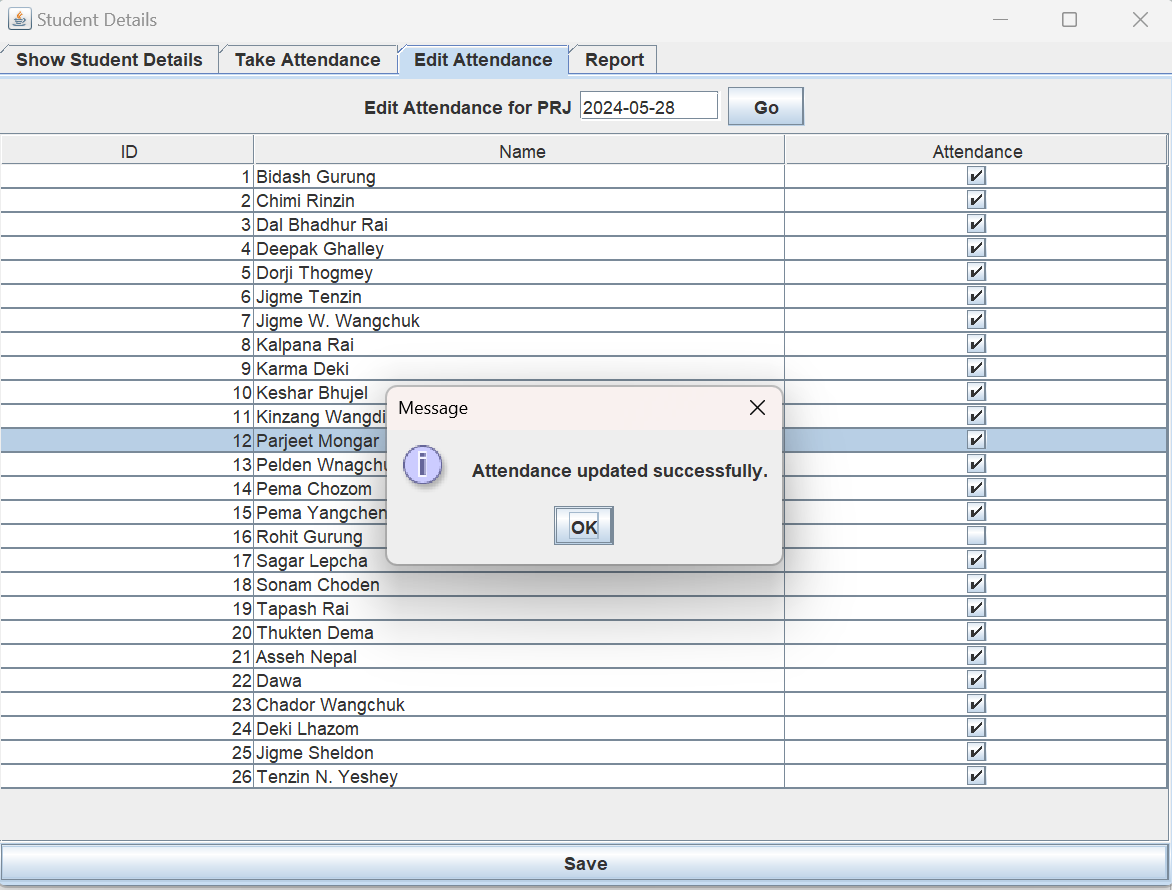




1. Edit Attendance

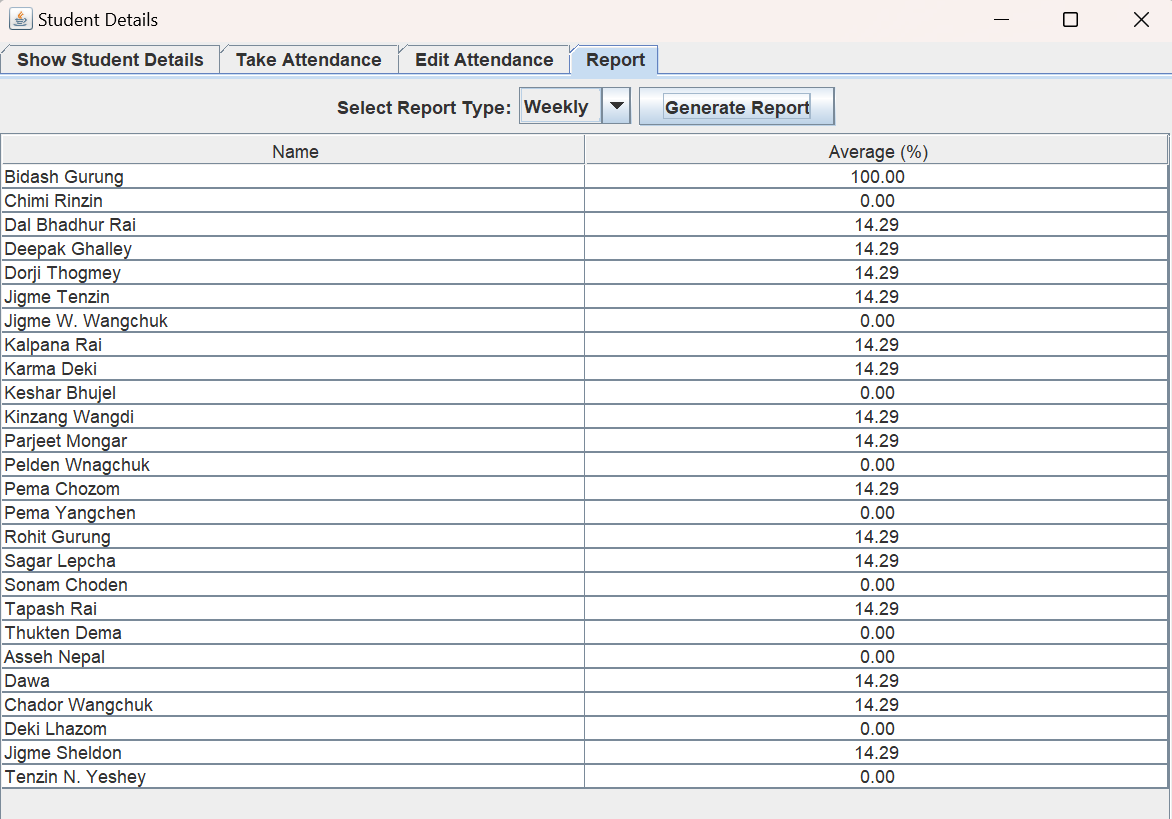
The attendance records for their classes are updated by lecturers. They are able to update their student's attendance or absence status. Instructors designate a day for recording attendance and edit. Students can be marked as present or absent using the interface.

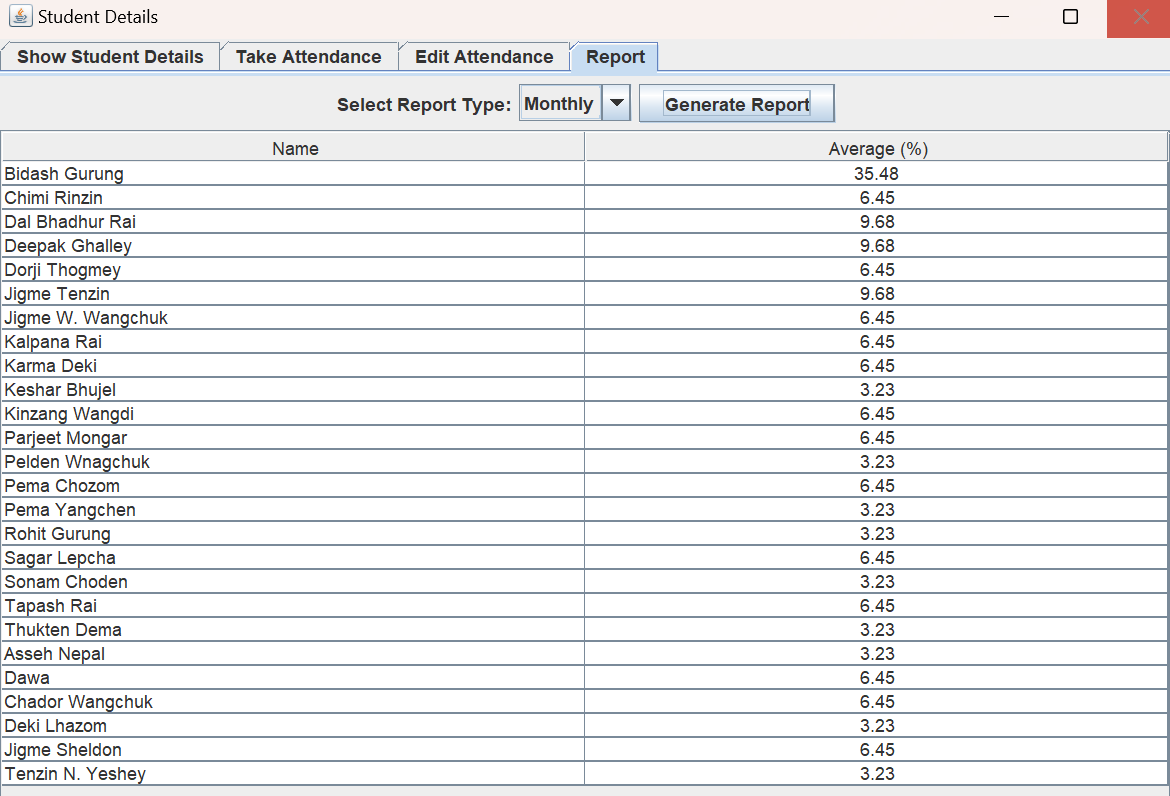




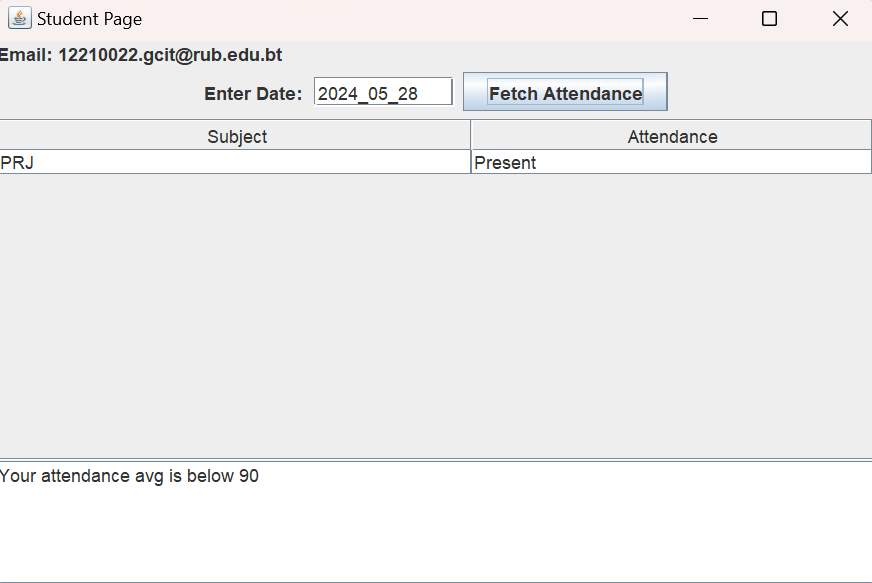
1. Generate Attendance Report

Every student can receive comprehensive weekly and monthly attendance reports from the system.





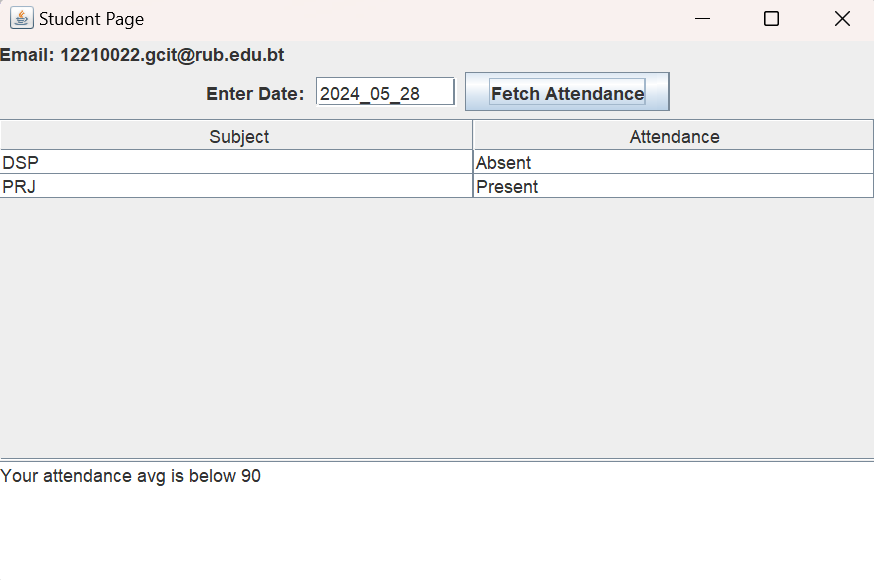
1. Notification

If a student's attendance is less than 90%, a message is displayed on their student page. 

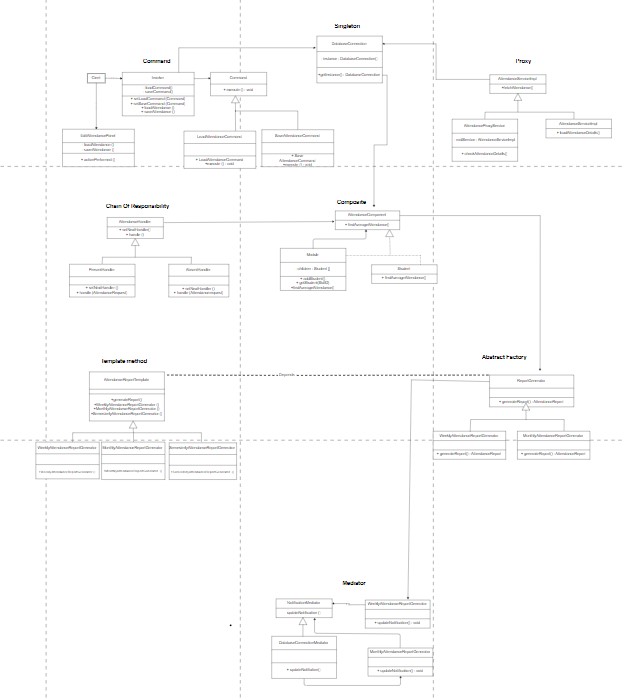


1. View Attendance on particular date for all modules

Users can view attendance records for specific dates for all the modules they are enrolled in.



# All Class Diagrams:



Class diagram link:

<https://app.diagrams.net/#G1jJ73FrrSCWiCYSZL4cO_M1nOxEQMRsmM#%7B%22pageId%22%3A%22C5RBs43oDa-KdzZeNtuy%22%7D>

# Justification for all the design pattern used:

Singleton: Managing multiple connections to the database can be resource-intensive and lead to inconsistencies. The Singleton pattern ensures that only one instance of the database connection is used throughout the application. This centralizes access to the database and avoids the overhead associated with establishing multiple connections.

Abstract factory: Different types of attendance reports (e.g., weekly, monthly) require different calculations and data handling. The Abstract Factory pattern provides a way to encapsulate a group of individual factories that have a common theme. This allows the creation of related objects (attendance report generators).

Proxy: Direct access to sensitive parts of the database by students could lead to security breaches and unauthorized modifications. Proxy acts as a control point, ensuring that any request to load attendance details is logged before being passed to the actual implementation, providing an additional layer of functionality and control.

CoR: Different handlers (e.g., marking attendance as present or absent, handling login) require distinct processing steps that should be processed in a sequence. The Chain of Responsibility pattern allows a request to be passed along a chain of handlers, where each handler decides either to process the request or to pass it to the next handler in the chain.

Command: Command pattern encapsulates editing operations for attendance records as discrete objects, enabling flexible, maintainable, and extendable handling of various edit actions.

Mediator: Mediator pattern manages notifications for students with attendance below 90% by centralizing interaction logic, promoting loose coupling, enhancing maintainability, and simplifying complex communication between components.\

# Framework

The framework remains constant and cannot be changed, whereas the implementation is flexible and can be adjusted to meet the client's requirements or the application developer's preferences. The framework offers a foundational structure and guidelines, ensuring uniformity and dependability across various applications.

Chain of responsibility for login

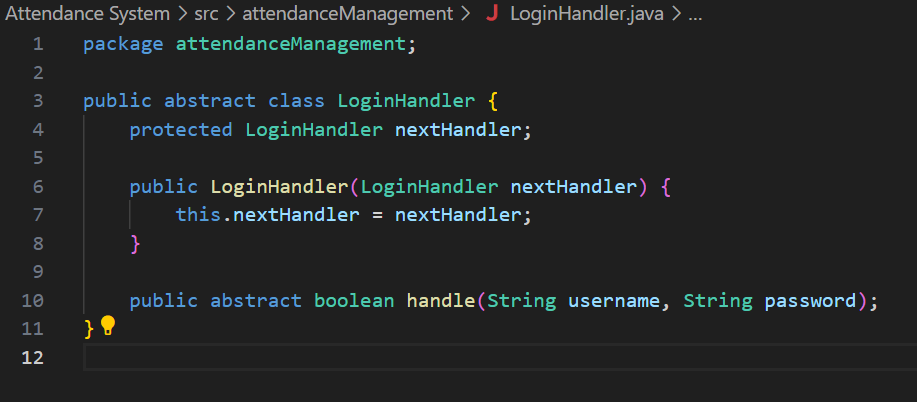
The provided code snippet uses the Chain of Responsibility design pattern. This pattern allows a request to be passed along a chain of handlers, where each handler can either process the request or pass it to the next handler in the chain. In this code:

LoginHandler is an abstract class that defines an interface for handling requests (handle method) and maintains a reference to the next handler in the chain (nextHandler).

Concrete subclasses of LoginHandler will implement the handle method to process login requests (username and password).

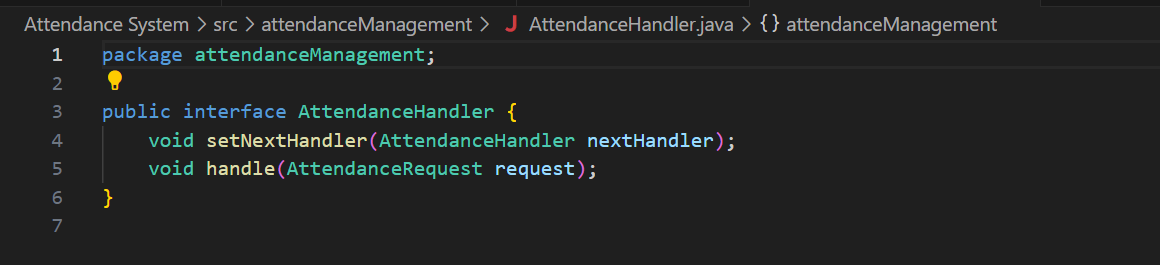
If a handler cannot process the request, it passes the request to the next handler in the chain.

This pattern decouples the request sender from its receivers, provides flexibility in assigning responsibilities, and supports scalability by allowing new handlers to be added easily.



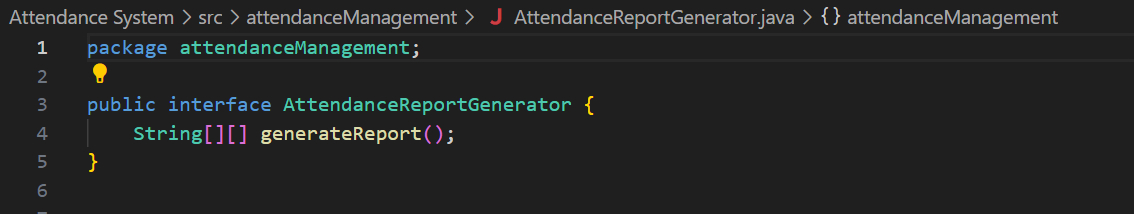
Chain of responsibility for marking absent or present

The Chain of Responsibility (CoR) pattern is used to process a request through a sequence of handlers, where each handler can either handle the request or pass it to the next handler in the chain.



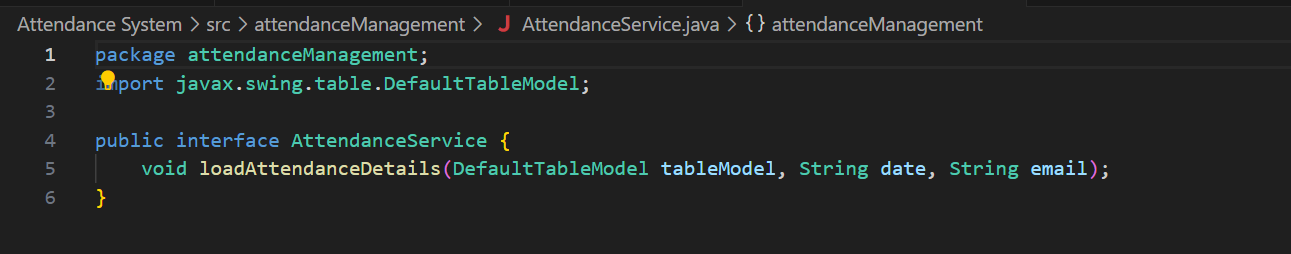
Factory Method for generating report

The AttendanceReportGenerator interface defines a method for generating attendance reports, returning a 2D array of strings. To incorporate the Abstract Factory design pattern, we created an abstract factory that provides different types of AttendanceReportGenerator implementations.

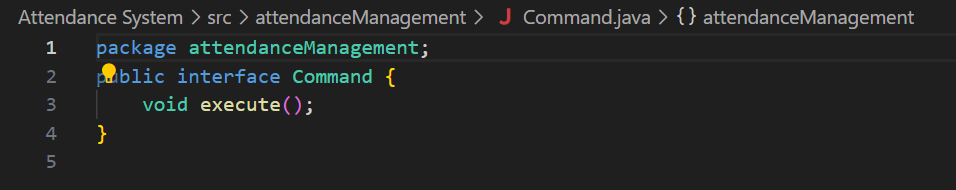


Proxy for loading Attendance

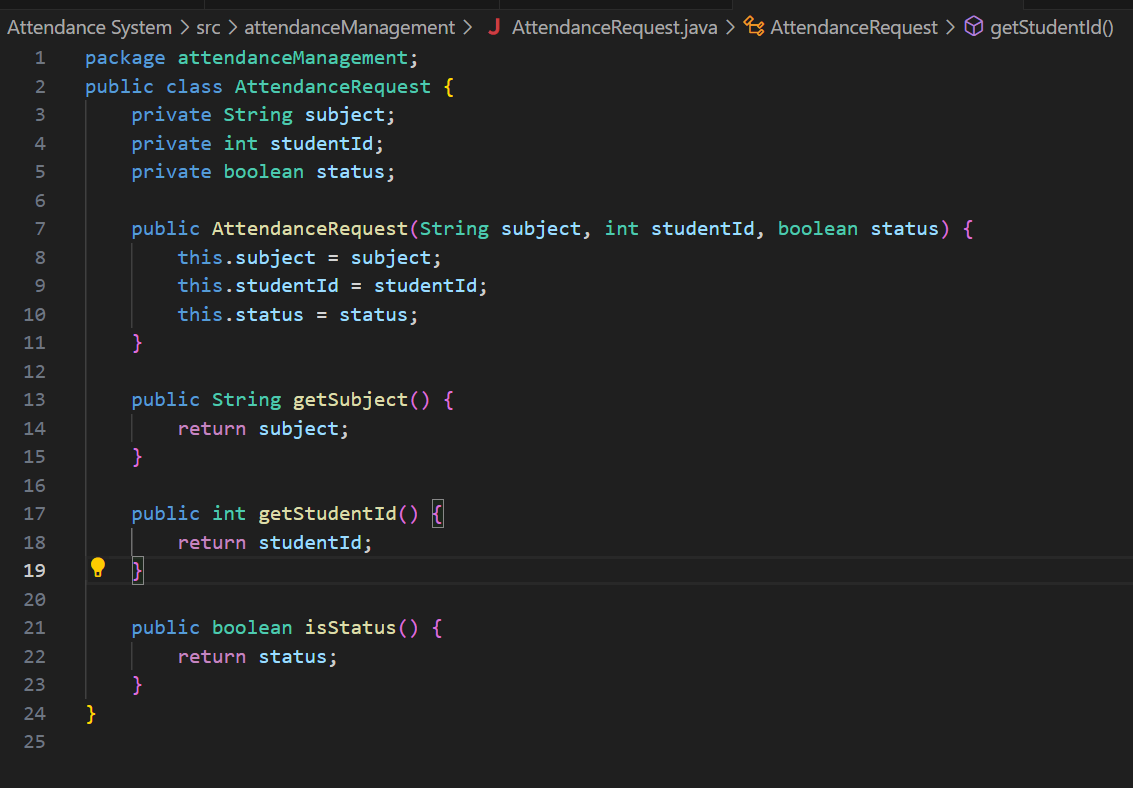
The proxy controls access to the real attendance service, adding additional layers such as access control or logging without changing the client code.



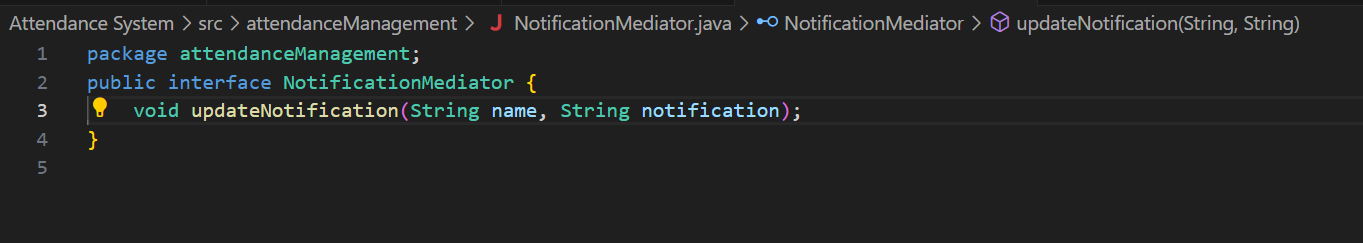
Command for loading,saving and updating attendance



Check the attendance request based on subject, student id and status

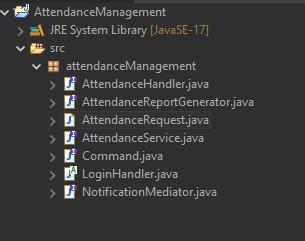


Mediator for sending notification to the student who have attendance below 90%.

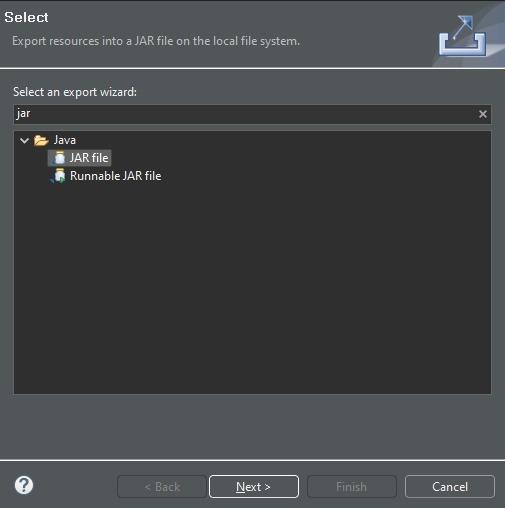


# How the framework is converted into jar files

I. We ensured our project structure was organized. Our source files were in the src folder, and all resources were placed in their respective folders. In the Package Explorer, we right-clicked on the project we wanted to export i.e, atttendanceMangement.



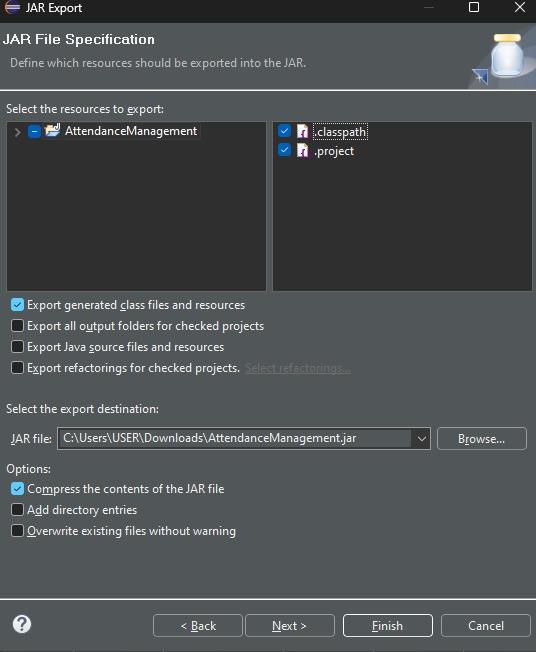
II. From the context menu, we selected Export. In the Export dialog, we expanded the Java folder and selected the JAR file. Then, we clicked Next.



III. We selected ‘export generated class files and resources’ to ensure our compiled .class files were included.

We saved the JAR file as AttendanceManagement.jar in our desired location.

Clicked Finish to create the JAR file.



IV. And in our downloads a new JAR file was created.



# Challenges:

Patterns: Implementing design patterns in the project was challenging and took a lot of effort.

Scalability: Designing the system to handle a large number of users and records efficiently.

Security: Protecting sensitive user information and preventing unauthorized access.

User Interface: Creating a user-friendly interface that is easy for all user types to navigate.

# Conclusion:

The Student Attendance Management System effectively addresses the need for a streamlined and automated process of managing student attendance. By utilizing appropriate design patterns and providing a robust set of features, the system enhances the efficiency and accuracy of attendance tracking while offering valuable insights through report generation and notifications. Despite challenges in implementing design patterns, scalability, security, and user interface design, the system provides a comprehensive solution to attendance management in educational institutions.