Attendance Management System – Report

**AttendanceManagementSystem**, appears to be a **Java-based web application** designed to manage student attendance. The structure suggests it follows **Maven** conventions and uses **Java EE (Jakarta EE) technologies** along with **JSP (JavaServer Pages)** for the frontend. Used **JSTL(JavaServer Pages Standard Tag Library)** which provides tags to control JSP page functionality.

This project allows Faculty to maintain student attendance.

* Enter new student on today’s date
* Generate report of that
* Search on list by any keyword
* Give attendance
* Attendance can be re-considered
* Can show all the students entered by faculty
* Generate report of that
* Search on list by any keyword
* Give attendance
* Attendance can be re-considered
* Make student absent or present
* Can edit or delete student record
* Can change faculty’s password
* Can update faculty’s profile

**Technology used:** JAVA, Spring, MySQL Database, Servlet, JSP, JDBC

### 1. Project Structure overview

**1.1. Java Code (Backend - Business Logic)**

Located in **com.example.attendancemanagementsystem**, your backend consists of multiple servlets responsible for handling HTTP requests.

**Servlets and Their Responsibilities**

| **Servlet Name** | **Function** |
| --- | --- |
| **AbsentServlet** | Handles marking students as absent. |
| **AddNewStudentServlet** | Adds a new student to the system. |
| **AllMyStudentServlet** | Retrieves a list of all students. |
| **ChangePasswordServlet** | Allows faculty to change their password. |
| **DashboardServlet** | Loads students record which is added inn today’s date. |
| **DeleteStudentServlet** | Deletes a student from the system. |
| **ForgetPasswordServlet** | Handles the password reset process of faculty |
| **GenerateReportOfAllMyStudentServlet** | Generates reports for all students. |
| **GenerateReportOfDashboardStudentServlet** | Generates a report for dashboard students. |
| **InsertTodayDateServlet** | Inserts the current date (For attendance records). |
| **LoginServlet** | Manages user login authentication. |
| **LogoutServlet** | Handles user logout functionality. |
| **PresentServlet** | Marks students as present. |
| **ProfileServlet** | Retrieves and updates user profile information. |
| **RegistrationServlet** | Handles new user registration. |
| **SearchAllMyStudentServlet** | Searches for students with any criteria. |
| **SearchDashboardStudentServlet** | Searches students within the dashboard view. |
| **UpdateAttendanceServlet** | Reconsider any attendance records. |
| **UpdateStudentServlet** | Updates student details. |

**1.2. Database Connectivity (database.db)**

* **Contains database connection logic (JDBC-based).**
* **Ensures interaction with MySQL/PostgreSQL/any other DB.**
* **Handles SQL queries for CRUD operations (Create, Read, Update, Delete).**

**Example Connection Code**: (This is for Revision)

public Connection getConnection() {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

return DriverManager.getConnection(DB\_URL, USER, PASSWORD);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

**1.3. Model Classes (model Package)**

Contains JavaBeans (POJOs) representing database entities.

| **Model Class** | **Description** |
| --- | --- |
| **Login** | Stores user login details (id, username, password, todaydate). |
| **Registration** | Represents user registration details (id, username, password, fullname, mobile, dob, qualification). |
| **Student** | Represents student details (id, username, todaydate, sname, rollno, course, semester, branch, attendance). |

**Example Student.java Model:** (For revision later on)

public class Student {

private int id;

private String name;

private String email;

// Constructors

public Student() {}

public Student(int id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

// Getters and Setters

}

**1.4. Resources (resources Folder)**

* Contains META-INF files for database persistence (persistence.xml).
* persistence.xml configures JPA settings (though currently empty).
* Likely used for Hibernate or EntityManagerFactory.

**2. Configuration Files**

**2.1. web.xml (Deployment Descriptor)**

* Defines servlet mappings, session timeouts, and filters (if configured).
* Currently empty, meaning the project relies on annotations (@WebServlet) instead.

**2.2. persistence.xml (JPA Configuration)**

* Ensures database entity management via JPA/Hibernate.
* Defines a persistence unit (default).

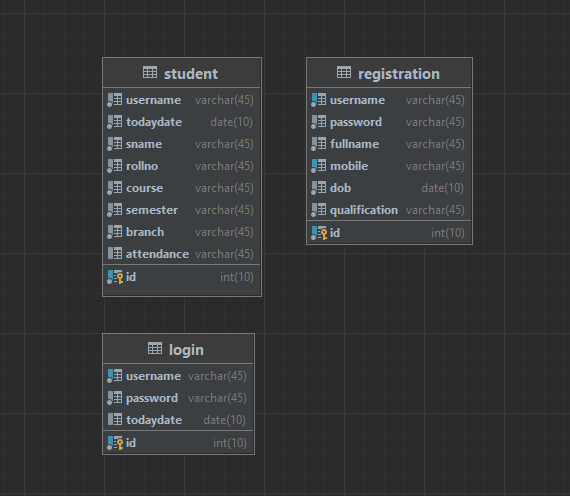
**3. Frontend (webapp Folder)**

* Contains JSP files that serve as the UI (View Layer).
* Handles user interactions (login, registration, attendance marking, etc.).
* Uses JSTL (JavaServer Pages Standard Tag Library) and EL (Expression Language) for dynamic data rendering.

## Project Structure Analysis

**1. Source Code (src/main)**

* **java/com.example.attendancemanagementsystem**
  + Contains different **packages**, for organizing the backend logic:
    - database: “db” class include **DAO (Data Access Object)**for database interactions. It contains Database connection, queries, many functions to retrieve user data, perform crud as per that using prepared statement.

Created Database like this using mySQL workbench

* + - model: holds **POJOs (Plain Old Java Objects)** representing entities like students and faculty. 3 Pojos in total – Login, Registration and Student. These have variables are shown in DataBase image. Means same name of variable as in database table’s entity.

**2. Resources (src/main/resources)**

* **META-INF Folder**
  + beans.xml: used for dependency injection (**Jakarta EE**).
  + persistence.xml: Defines database configuration for **JPA (Java Persistence API)**, specifying **database connection settings**. As this class is empty for now , my JPA is perfectly handling with db() class. That contains getConnection();

**3. Web Module (src/main/webapp)**

* **Contains JSP files**, which serve as the **frontend UI** for different user actions:
  + **Authentication & User Management**
    - login.jsp: Login page.
    - registration.jsp: User registration.
    - forget-password.jsp: Allows user to change password.
    - pre-forget-password-validation.jsp: Wants user to enter username to go to forget-password.jsp.
    - changemypass.jsp: Change password functionality.
  + **Student Management**
    - addnew-student.jsp: Add a new student by a certain faculty.
    - delete-student.jsp: Remove a student.
    - update-student.jsp: Modify student details.
    - allmy-student.jsp: Display all students entered by logged in faculty in previous.
  + **Faculty Profile**
    - faculty-profile.jsp: Likely shows faculty information.
  + **Dashboard & Attendance Management**
    - dashboard.jsp: The main dashboard page.
    - index.jsp: Likely the homepage or entry point.
    - insert-todaydate.jsp: Might be for inserting today's attendance records.
* **WEB-INF/web.xml**
  + Configuration file for the **Servlet container** (Tomcat/Jetty). But not using this.
  + Using **annotations** (@WebServlet, @WebFilter) **instead of web.xml**.

**4. Other Files**

* **.gitignore**: Specifies files and folders to be ignored in version control (Git).
* **pom.xml**: The Maven configuration file, which manages dependencies and build settings.
* **AttendanceManagementSystem.iml**: An IntelliJ IDEA module file.

**General Observations**

* **Well-structured project** following Java EE conventions.
* **JSP-based frontend**, which suggests a **Servlet-JSP-MVC architecture**.
* **Database interaction** is likely handled using **JPA (Hibernate)** based on the persistence.xml file.
* **Uses Maven**, making dependency management and build automation easier.

**Drawbacks as per nowadays:**

* Consider using **Spring Boot** instead of Java EE for better maintainability.
* **Servlets should be well-structured** to avoid placing too much business logic in JSP pages.
* Implement **security measures** such as **session management and validation** in login and registration pages.

##Signout is not working properly