A Project Report on

“STUDENT RESULT MANAGEMENT SYSTEM”

Submitted By

DIMPAL K L	**AF0459972**
SNEHA CS	**AF0460057**

Under the Guidance of

Vanisha Nagesh

2024-25

STUDENT RESULT MANAGEMENT SYSTEM

INTRODUCTION

The Student Result Management System is a Java-based application that lets users manage their bank accounts efficiently. Built with Java Hibernate and MySQL, Student Result Management System is a software solution that helps manage the academic records of students, including their results, marks, subjects, and other related data. It allows for efficient tracking, analysis, and reporting of student performance

Key Features:

Student– Represents a learner enrolled in the academic institution.

Result – the summary of the student's academic performance in each subject, typically represented by grades or marks..

Marks – The numerical score or grade assigned to a student in a specific subject.

Subject – The academic discipline or course for which the student receives evaluation

Admin – he user role responsible for managing the overall operation of the Student Result Management System.

Technology Stack:

Backend: Java (JDBC)

Database: MySQL

IDE: VS Code / Eclipse

OBJECTIVE

Student Result Management System (SRMS) project is to streamline and automate the process of managing student academic records, including marks, results, and subject management. It aims to provide a user-friendly interface for both administrators and students while ensuring accuracy, security, and efficiency in handling educational data. Here are the key objectives of this project

Key Objectives:

- **Manage Student Information:** Store and organize all student details in one place.
- **Automate Result Calculation:** Automatically calculate and generate student results to avoid errors.

- **Provide Instant Access to Results:** Allow students to easily view their results at any time**.**
- **Manage Subjects and Marks:** Let admins manage subjects and enter student marks efficiently.
- **Generate Reports**: Create reports on student performance, class averages, and trends.
- **Easy to Use**: Design the system to be simple and user-friendly for students, teachers, and admins.
- **Secure Data**: Keep student data safe and make sure only authorized people can access it.
- **Reduce Admin Workload**: Automate tasks like registration, result entry, and subject management to save time.
- **Improve Communication:** Help students and admins communicate easily about academic results.
- **Scalable and Flexible:** Build a system that can grow and adapt to future needs.
- **Ensure Accuracy:** Make sure all student data is accurate and reliable**.**

DATABASE DESIGN / ENTITY-RELATIONSHIP DIAGRAM

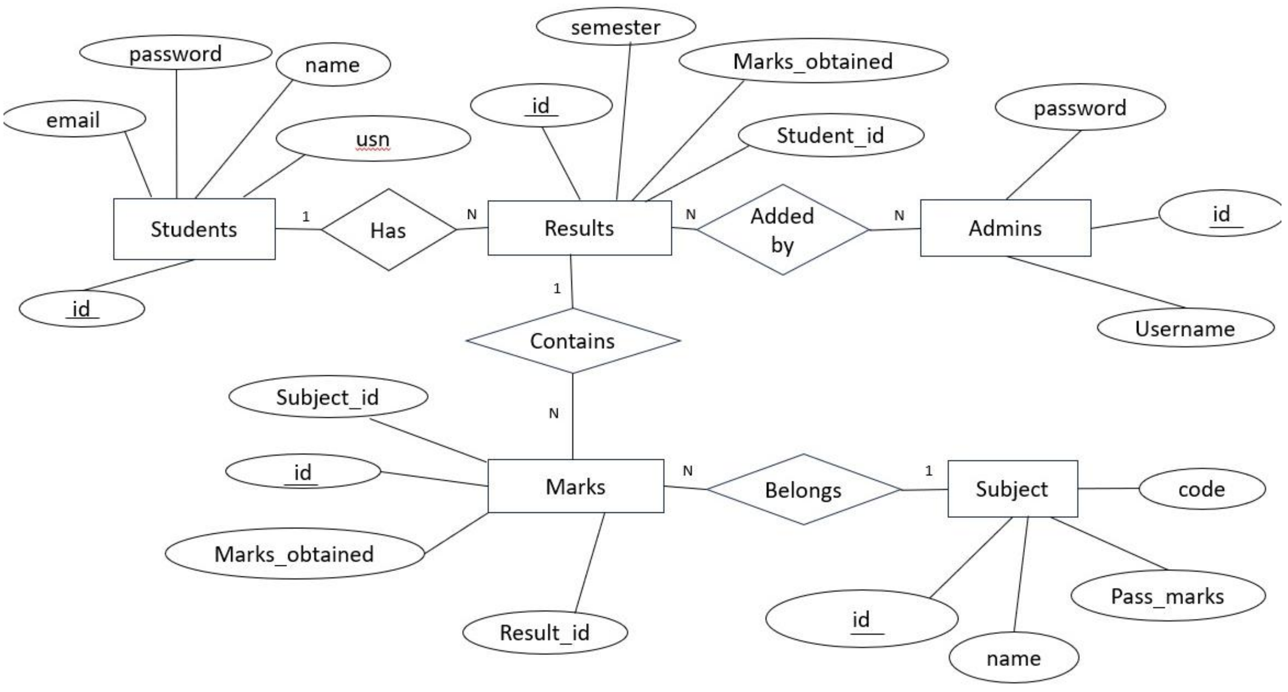
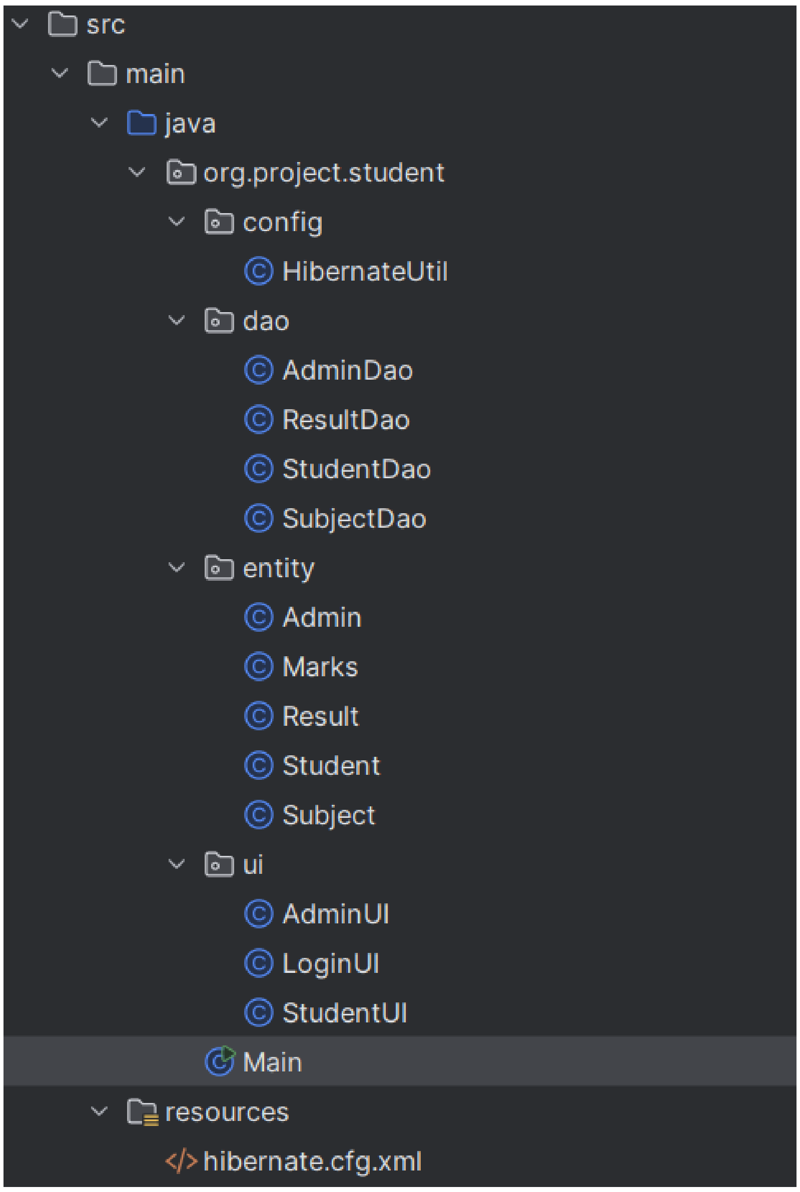


Fig.1.1:E-R Diagram of Student Result Management System

IMPLEMENTATION



```
src
  main
    java
      org.project.student
        config
          HibernateUtil
        dao
          AdminDao
          ResultDao
          StudentDao
          SubjectDao
        entity
          Admin
          Marks
          Result
          Student
          Subject
        ui
          AdminUI
          LoginUI
          StudentUI
        Main
    resources
      hibernate.cfg.xml
```

DAO Package (Data Access Layer)

1. AdminDao.java

Handles all operations related to the admins table.

Methods:

- **findByUsername(String username)** – Retrieves an admin by username from the database.
- **validateLogin(String username, String password**) – Authenticates an admin by checking the provided credentials.
- **saveAdmin(Admin admin)** – Saves or updates an admin in the database.
- **initialize()** – Checks if any admin exists; if not, creates a default admin (admin/admin123).
- **countAdmins() –** Returns the total number of admins in the database.

2. ResultDao.java

Handles all operations related to student results.

Methods:

- **findByStudentAndSemester(Student student, int semester) – Retrieves a result for a specific student and semester.**
- **saveResult(Result result) – Saves or updates a result after calculating pass/fail status.**
- **findByStudent(Student student) – Fetches all results for a particular student.**
- **createResult(Student student, int semester, List<Subject> subjects, List<Integer> marksObtained) – Creates or updates a result for a student, assigning marks to subjects and computing the final status.**

3. StudentDao.java

Handles all operations related to the students table.

Methods:

- **findByUsn(String usn)** – Retrieves a student based on their USN.
- **validateLogin(String usn, String password)** – Authenticates a student using their USN and password.
- **saveStudent(Student student)** – Saves or updates a student in the database.
- **getAllStudents()** – Fetches a list of all students from the database.
- **initialize()** – Creates a sample student (1MS17CS001/password) if no students exist.
- **countStudents()** – Returns the total number of students in the database.

4. SubjectDao.java

Handles all operations related to the subjects table.

Methods:

- **findByCode(String code)** – Retrieves a subject based on its code.
- **saveSubject(Subject subject)** – Saves or updates a subject in the database.
- **getAllSubjects()** – Fetches a list of all subjects from the database.
- **initialize()** – Creates default subjects (CS101, MA101, PH101) if no subjects exist.
- **countSubjects()** – Returns the total number of subjects in the database.

Model Package (Data Models)

1. Admin.java

Represents the admins table in the database.

Attributes:

- **id *(Long)*** – Unique identifier for each admin.
- **username *(*String, unique, non-null)** – Admin's login username.
- **password (String, non-null)** – Admin's password for authentication.

2. Marks.java (Entity Class)

This entity represents the **marks** obtained by a student in a particular **subject** within a **result** entry. It links a student's result to specific subjects and tracks whether they have passed or failed.

Attributes:

- **id (Long, Primary Key, Auto-Generated)** – Unique identifier for each marks entry.
- **result (Result, Many-to-One relation*)*** – Links marks to a specific student's result.
- **subject (Subject, Many-to-One relation)** – Links marks to a specific subject.
- **marksObtained *(*int, Non-null)** – The score a student achieved in the subject.
- **status (String, Non-null, "PASS" or "FAIL")** – Indicates whether the student passed or failed based on the subject's **pass marks**.

Result.java (Entity Class)

This entity represents a **student's academic result** for a particular **semester**. It stores the student's **total marks**, **pass/fail status**, and the list of **marks** obtained in various subjects.

Attributes:

1. **id (Long, Primary Key, Auto-Generated***)* – Unique identifier for each result.
2. **student (Student, Many-to-One relation)** – The student to whom this result belongs.
3. **semester (int, Non-null***)* – Semester in which the student appeared.
4. **totalMarks (int)** – Total marks obtained in all subjects.
5. **status (String, "PASS" or "FAIL")** – Indicates whether the student passed or failed.
6. **marksList (List<Marks>, One-to-Many relation, Cascade.ALL, Orphan Removal, FetchType.EAGER)**
    - Stores the **marks** obtained in each subject.
    - **Cascade.ALL** ensures that if a result is deleted, its marks are deleted too.
    - **Orphan Removal** removes marks entries if they are removed from the list.
    - **FetchType.EAGER** ensures marks are loaded immediately when a result is retrieved.

Subject.java (Entity Class)

The Subject entity represents a **subject** in the **Student Result Management System**. It defines the subject's **code, name, maximum marks, and passing marks**.

Attributes:

1. **id (Long, Primary Key, Auto-Generated)** – Unique identifier for each subject.
2. **code (String, Unique, Non-null, Max Length: 20***)* – A **unique subject code** (e.g., "CS101").
3. **name (String, Non-null, Max Length: 100) –** The **name** of the subject (e.g., "Computer Science Fundamentals").
4. **maxMarks (int, Default: 100, Non-null)** – The **maximum marks** a student can obtain in this subject.

5. **passMarks (int, Default: 35, Non-null)** – The **minimum passing marks** required to pass the subject.

UI Package (User Interface Layer)

Contains console-based user interface classes.

AdminUI

1. Menu & Input Handling

- **showAdminMenu(Admin admin)** → Displays the admin dashboard and processes user choices.
- **displayAdminMenu()** → Prints the admin menu options on the console.
- **getStringInput(String prompt**) → Reads and returns a trimmed string input from the user.
- **getIntInput(String prompt)** → Reads and returns a valid integer input from the user.

2. Student management

- **addStudent**() → Adds a new student after checking for duplicate USN.

- **viewAllStudents()** → Retrieves and displays all students from the database.

3. Subject Management

- **addSubject()** → Adds a new subject with unique subject code and predefined max/pass marks.
- **viewAllSubjects()** → Retrieves and displays all subjects from the database.

4. Result Management

- **addResult()** → Adds a result for a student, selecting up to 3 subjects and entering marks

LoginUI

1. Initialization & Menu Handling

- **initializeSampleData()** → Initializes sample admin and student data.
- **start()** → Displays the main menu and processes user choices.
- **displayMainMenu()** → Prints the main login menu options

2. Authentication

- **adminLogin()** → Handles admin login by validating username and password.
- → Handles student login by validating USN and password.

3. Input Handling studentLogin()

- **getStringInput(String prompt)** → Reads and returns a trimmed string input from the user.
- **getIntInput(String prompt)** → Reads and returns a valid integer input from the user.

StudentUI

1. Menu Handling

- **showStudentMenu(Student student**) → Displays the student dashboard and processes menu options.
- **displayStudentMenu()** → Prints the student dashboard menu options.

2. Viewing Results

- **viewAllResults(Student student**) → Fetches and displays all results for the logged-in student.
- **viewResultBySemester(Student student)** → Fetches and displays the result for a specific semester.
- **displayResult(Result result**) → Displays the details of a single result, including subject-wise marks and overall status.

3.Input Handling

- **getIntInput(String prompt)** → Reads and returns a valid integer input from the user.

Main App

1. main(String[] args)

Entry point of the application; initializes the system, starts the login UI, and handles exceptions.

HibernateUtil

Methods:

buildSessionFactory() → Creates and initializes the
Hibernate SessionFactory from **hibernate.cfg.xml.**

1. **getSessionFactory()** → Returns the singleton instance of SessionFactory for database operations.
2. **shutdown()** → Closes the SessionFactory to release database resources.

hibernate.cfg.xml

Database Connection Properties

1. **connection.driver_class** → Specifies the JDBC driver (com.mysql.cj.jdbc.Driver for MySQL).
2. **connection.url** → Defines the database URL, including database name (student_db) and additional parameters.
3. **connection.username** → Database username (root).
4. **connection.password** → Database password (Sanjay@123).

Connection Pool & SQL Configuration

5. **connection.pool_size** → Sets the connection pool size (10).

6. **dialect** → Defines the SQL dialect (MySQL8Dialect for MySQL 8+).
7. **show_sql** → Enables logging of executed SQL queries (true).
8. **format_sql** → Formats SQL output for better readability (true).
9. **hbm2ddl.auto** → Specifies schema generation strategy (update to modify schema without data loss).

Entity Mappings

10. <mapping class="org.project.student.entity.Admin"/> → Maps the Admin entity to the database.
11. <mapping class="org.project.student.entity.Student"/> → Maps the Student entity to the database.
12. <mapping class="org.project.student.entity.Subject"/> → Maps the Subject entity to the database.
13. <mapping class="org.project.student.entity.Result"/> → Maps the Result entity to the database.
14. <mapping class="org.project.student.entity.Marks"/> → Maps the Marks entity to the database.

CONCLUSION

The **Student Result Management System** streamlines the process of managing student grades and academic records. By automating the grading and reporting process, it reduces human errors and administrative workload, ensuring that student data is handled in a secure and efficient manner.