STUDENT GRADEBOOK

A PROJECT REPORT



SUBMITTED BY- ISHIKA AVASTHI (23CSU131)

SAIYAM TANEJA (23CSU273)

SUBMITTED TO-

DR. ANURADHA DHULL

INDEX

1. INTRODUCTION
2. PROJECT OUTLINE
3. FLOWCHART
4. CODE
5. OUTPUT

INTRODUCTION

The **Digital Gradebook Management System** is a robust standalone Java application designed to streamline academic management for educational institutions. Built with a role-based structure, it caters to Admins, Teachers, and Students, ensuring personalized functionality for each user. Admins can manage students, subjects, and assignments, while Teachers can record grades and track performance. Students have access to their grades and progress reports.

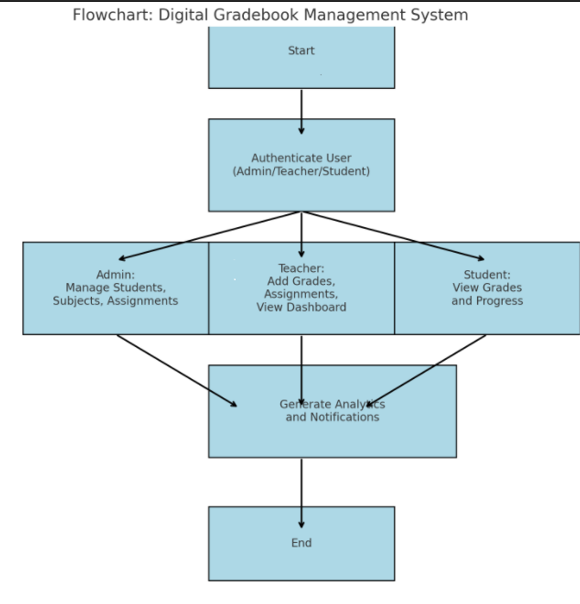
The system supports grading schemes like weighted averages and letter grades, along with features such as attendance tracking, assignment management with deadlines, and performance analytics. A user-friendly dashboard provides insights into total students, subjects, and assignments, helping educators identify top performers and students needing improvement. Notifications simulate reminders for deadlines or grade updates, enhancing the overall user experience.

This project showcases key concepts of Java programming, including data structures, modular design, and object-oriented principles, while also integrating advanced features like role-based permissions and analytics. Designed as an efficient and scalable solution, this gradebook system demonstrates real-world applicability and offers an impressive showcase of technical and problem-solving skills.

PROJECT OUTLINE

The **Digital Gradebook Management System** is a Java-based standalone application designed to manage academic data efficiently. It supports role-based access control for Admins, Teachers, and Students, enabling features like student and subject management, assignment tracking, and grade recording. Admins can oversee system operations, Teachers can manage grades and assignments, and Students can view their grades and progress reports. The system includes grading schemes, attendance tracking, notifications for deadlines, and an analytics dashboard displaying performance insights. Built using object-oriented principles and advanced Java concepts, this project demonstrates scalability, efficiency, and real-world applicability, making it ideal for educational institutions

FLOWCHART



CODE

THE FOLLOWING CLASSES ARE USED-

1. User
2. Student
3. Subject
4. Assignment
5. Gradebook

**1. USER CLASS**

public class User {

private String username;

private String role;

public User(String username, String role) {

this.username = username;

this.role = role;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

**2. STUDENT CLASS**

import java.util.HashMap;

import java.util.Map;

public class Student {

private int id;

private String name;

private String email;

private String gradeLevel;

private Map<String, Double> grades = new HashMap<>();

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

this.id = id;

this.name = name;

this.email = email;

this.gradeLevel = gradeLevel;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getGradeLevel() {

return gradeLevel;

}

public void setGradeLevel(String gradeLevel) {

this.gradeLevel = gradeLevel;

}

public Map<String, Double> getGrades() {

return grades;

}

public void setGrades(Map<String, Double> grades) {

this.grades = grades;

}

}

**3. SUBJECT CLASS**

public class Subject {

private String name;

private String teacher;

public Subject(String name, String teacher) {

this.name = name;

this.teacher = teacher;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getTeacher() {

return teacher;

}

public void setTeacher(String teacher) {

this.teacher = teacher;

}

}

**4. ASSIGNMENT CLASS**

import java.util.Date;

public class Assignment {

private String title;

private String subject;

private double weightage;

private Date dueDate;

private boolean submitted;

public Assignment(String title, String subject, double weightage, Date dueDate) {

this.title = title;

this.subject = subject;

this.weightage = weightage;

this.dueDate = dueDate;

this.submitted = false;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getSubject() {

return subject;

}

public void setSubject(String subject) {

this.subject = subject;

}

public double getWeightage() {

return weightage;

}

public void setWeightage(double weightage) {

this.weightage = weightage;

}

public Date getDueDate() {

return dueDate;

}

public void setDueDate(Date dueDate) {

this.dueDate = dueDate;

}

public boolean isSubmitted() {

return submitted;

}

public void setSubmitted(boolean submitted) {

this.submitted = submitted;

}

}

**5. GRADEBOOK CLASS**

import java.util.\*;

public class GradebookApp {

private Map<Integer, Student> students = new HashMap<>();

private Map<String, Subject> subjects = new HashMap<>();

private List<Assignment> assignments = new ArrayList<>();

private User currentUser;

public void authenticate(Scanner scanner) {

System.out.print("Enter username: ");

String username = scanner.nextLine();

System.out.print("Enter role (Admin/Teacher/Student): ");

String role = scanner.nextLine();

currentUser = new User(username, role);

System.out.println("Welcome, " + username + "! Role: " + role);

}

public boolean hasPermission(String requiredRole) {

return currentUser != null && currentUser.getRole().equalsIgnoreCase(requiredRole);

}

public void addStudent(Scanner scanner) {

if (!hasPermission("Admin")) {

System.out.println("Permission Denied: Only Admins can add students.");

return;

}

System.out.print("Enter student ID: ");

int id = scanner.nextInt();

scanner.nextLine();

System.out.print("Enter student name: ");

String name = scanner.nextLine();

System.out.print("Enter email: ");

String email = scanner.nextLine();

System.out.print("Enter grade level: ");

String gradeLevel = scanner.nextLine();

students.put(id, new Student(id, name, email, gradeLevel));

System.out.println("Student added successfully!");

}

public void addSubject(Scanner scanner) {

if (!hasPermission("Admin")) {

System.out.println("Permission Denied: Only Admins can add subjects.");

return;

}

System.out.print("Enter subject name: ");

String name = scanner.nextLine();

System.out.print("Enter teacher name: ");

String teacher = scanner.nextLine();

subjects.put(name, new Subject(name, teacher));

System.out.println("Subject added successfully!");

}

public void addAssignment(Scanner scanner) {

if (!hasPermission("Teacher")) {

System.out.println("Permission Denied: Only Teachers can add assignments.");

return;

}

System.out.print("Enter assignment title: ");

String title = scanner.nextLine();

System.out.print("Enter subject name: ");

String subject = scanner.nextLine();

if (!subjects.containsKey(subject)) {

System.out.println("Subject not found!");

return;

}

System.out.print("Enter weightage: ");

double weightage = scanner.nextDouble();

scanner.nextLine();

System.out.print("Enter due date (yyyy-mm-dd): ");

String dueDateStr = scanner.nextLine();

Date dueDate;

try {

dueDate = new Date(dueDateStr);

} catch (Exception e) {

System.out.println("Invalid date format!");

return;

}

assignments.add(new Assignment(title, subject, weightage, dueDate));

System.out.println("Assignment added successfully!");

}

public void recordGrade(Scanner scanner) {

if (!hasPermission("Teacher")) {

System.out.println("Permission Denied: Only Teachers can record grades.");

return;

}

System.out.print("Enter student ID: ");

int studentId = scanner.nextInt();

scanner.nextLine();

Student student = students.get(studentId);

if (student == null) {

System.out.println("Student not found!");

return;

}

System.out.print("Enter subject: ");

String subject = scanner.nextLine();

if (!subjects.containsKey(subject)) {

System.out.println("Subject not found!");

return;

}

System.out.print("Enter grade: ");

double grade = scanner.nextDouble();

scanner.nextLine();

student.getGrades().put(subject, grade);

System.out.println("Grade recorded successfully!");

}

public void viewDashboard() {

System.out.println("\n--- Dashboard ---");

System.out.println("Total Students: " + students.size());

System.out.println("Total Subjects: " + subjects.size());

System.out.println("Total Assignments: " + assignments.size());

}

public void generateGradeReport(Scanner scanner) {

if (!hasPermission("Teacher") && !hasPermission("Student")) {

System.out.println("Permission Denied: Only Teachers and Students can view grade reports.");

return;

}

System.out.print("Enter student ID: ");

int studentId = scanner.nextInt();

scanner.nextLine();

Student student = students.get(studentId);

if (student == null) {

System.out.println("Student not found!");

return;

}

System.out.println("\nGrade Report for " + student.getName() + ":");

for (Map.Entry<String, Double> entry : student.getGrades().entrySet()) {

System.out.println("Subject: " + entry.getKey() + ", Grade: " + entry.getValue());

}

}

public void menu() {

Scanner scanner = new Scanner(System.in);

authenticate(scanner);

while (true) {

System.out.println("\n--- Gradebook Menu ---");

System.out.println("1. Add Student");

System.out.println("2. Add Subject");

System.out.println("3. Add Assignment");

System.out.println("4. Record Grade");

System.out.println("5. View Dashboard");

System.out.println("6. Generate Grade Report");

System.out.println("7. Exit");

System.out.print("Choose an option: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1 -> addStudent(scanner);

case 2 -> addSubject(scanner);

case 3 -> addAssignment(scanner);

case 4 -> recordGrade(scanner);

case 5 -> viewDashboard();

case 6 -> generateGradeReport(scanner);

case 7 -> {

System.out.println("Exiting...");

scanner.close();

return;

}

default -> System.out.println("Invalid choice! Please try again.");

}

}

}

public static void main(String[] args) {

GradebookApp app = new GradebookApp();

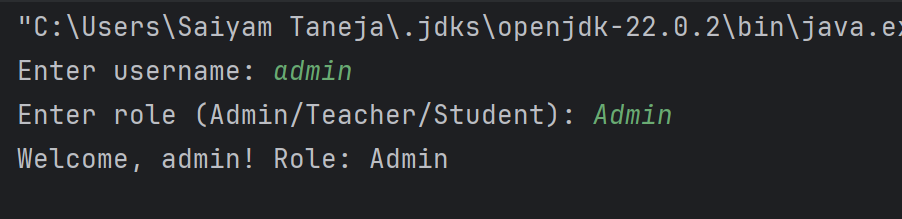
app.menu();

}

}

OUTPUT

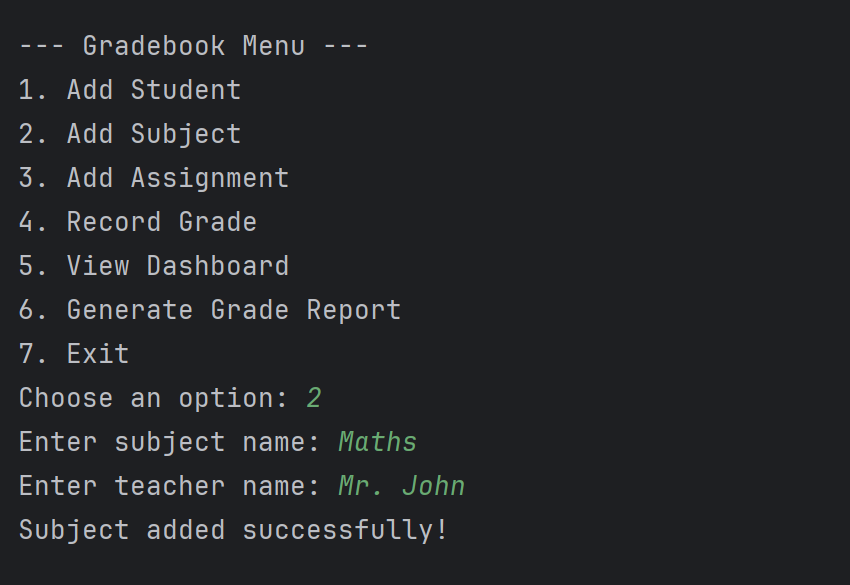
USER CLASS: -



STUDENT CLASS: -



STUDENT CLASS: -

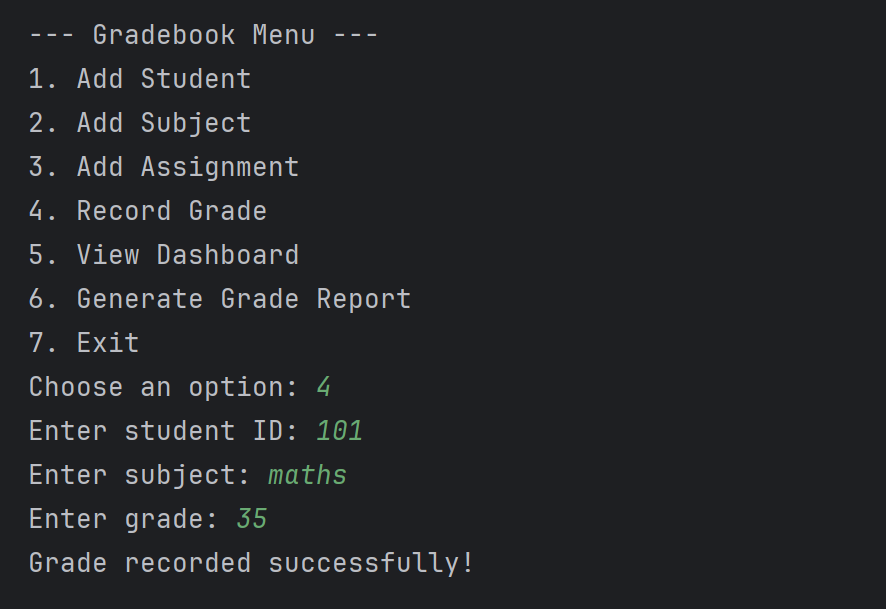


ASSIGNMENT CLASS: -

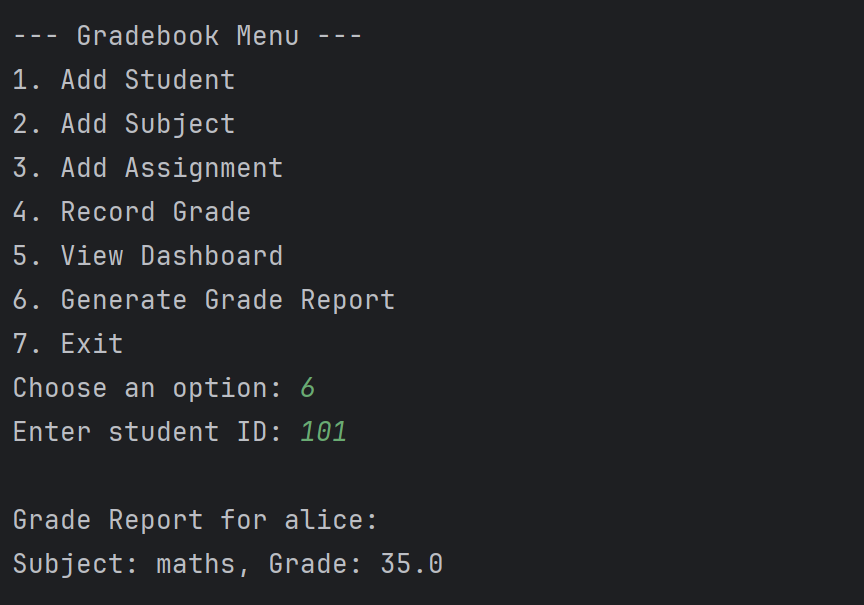
A screenshot of a computer

Description automatically generated

GRADBOOK CLASS:-



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

APPLICATIONS

**1.Educational Institutions**:

* + Used by schools, colleges, and universities to manage student performance, track grades, and monitor academic progress efficiently

1. **Performance Analytics**:
   * Teachers and administrators can use the gradebook to analyze student performance, identify areas where students are excelling or struggling, and provide targeted feedback to improve learning outcomes.
2. **Parent-Teacher Communication**:
   * Parents can access their child's gradebook to monitor academic performance and be informed of any upcoming assignments, tests, or performance concerns, fostering better communication between parents and teachers.
3. **Automated Reporting**:
   * Generates detailed grade reports for students, including insights on grades for assignments, tests, and overall performance. These reports can be easily shared with stakeholders such as students, teachers, and parents.
4. **Streamlining Administrative Tasks**:
   * Simplifies administrative tasks such as grade calculation, assignment tracking, and report generation, reducing manual effort and the likelihood of errors.

ADVANTAGES

1. **Centralized Data Management**:
   * Organizes and stores all student-related information (grades, assignments, attendance) in one place, making it easy to access and manage.
2. **Time-Efficient**:
   * Automates the process of grade calculation, report generation, and assignment tracking, reducing manual effort and time spent on administrative tasks.
3. **Real-Time Updates**:
   * Teachers can record and update grades instantly, providing students with real-time feedback on their academic performance.
4. **Customizable Grading**:
   * Supports different grading schemes (e.g., weighted averages, letter grades) to accommodate diverse grading policies in educational institutions.
5. **Improved Communication**:
   * Facilitates easy communication between teachers, students, and parents by providing access to grade reports, assignment status, and attendance updates.

FUTURE SCOPE AND CONCLUSION

The future scope of the Digital Gradebook System includes cloud integration, mobile app development, AI-based analytics, multi-language support, and LMS integration. It can also include automated notifications, advanced reporting, blockchain for grade security, gamification features, and external assessment tool integration to enhance user experience and functionality.

In conclusion, the Digital Gradebook Management System offers an efficient and organized way to manage student data, grades, and assignments. It streamlines administrative tasks, improves communication between teachers, students, and parents, and provides valuable insights into student performance. With its role-based access control, customizable grading, and real-time updates, the system enhances the overall learning experience. The future potential for cloud integration, AI analytics, and mobile access further demonstrates its scalability and adaptability to modern educational needs.

REFERENCES

* [www.w3schools.com](http://www.w3schools.com)
* [www.javatpoint.com](http://www.javatpoint.com)
* [stackoverflow.com](http://stackoverflow.com)
* [www.javascript.info](http://www.javascript.info)