AI SAMADHAN

PROBLEM - 1

"AI SAMADHAN" is more than just coding competitions; this is **catalysts for innovation, learning, and collaboration**. It empower individuals and organizations to harness the power of AI to solve pressing challenges and drive technological progress. By participating in AI **SAMADHAN**, you contribute to the growth of the AI ecosystem and make a meaningful impact on society.

Objective:

To design and implement an automated system that generates **lesson plans**, **course objectives**, **program objectives**, **CO-PO mapping**, **question banks**, and **course materials** based on a given syllabus. The system will also generate comprehensive reports to evaluate the effectiveness of the course and its alignment with program outcomes.

Scope:

The system will cater to the needs of educational institutions by automating the creation of structured lesson plans, course materials, and assessments. It will ensure alignment with course objectives (COs), program objectives (POs), and syllabus requirements. The system will also generate reports to analyze the effectiveness of the course and its outcomes.

Key Requirements:

1. Input Data:

- **Syllabus:** Detailed syllabus including topics, subtopics, and learning outcomes.
- **Course Objectives (COs):** Specific goals that the course aims to achieve.
- **Program Objectives (POs):** Broader goals of the program that the course contributes to.
- **Bloom's Taxonomy Levels:** Mapping of topics to cognitive levels (e.g., Remember, Understand, Apply, Analyze, Evaluate, and Create).
- **Resources:** Textbooks, reference materials, and online resources.
- **Assessment Criteria:** Guidelines for creating question banks and evaluation methods.

2. Output:

- **Lesson Plans:** Structured lesson plans with timelines, topics, and teaching methods.
- **Course Objectives (COs):** Clearly defined objectives for the course.
- **Program Objectives (POs):** Mapping of course objectives to program objectives.
- **CO-PO Mapping:** Detailed mapping of course outcomes to program outcomes.
- **Question Bank:** A repository of questions categorized by topic, difficulty level, and cognitive level.
- **Course Material:** Comprehensive course material including slides, notes, and reference links.

- **Reports:**
  - CO-PO Mapping Report.
  - Lesson Plan Effectiveness Report.
  - Question Bank Analysis Report.
  - Course Material Utilization Report.

3. Constraints:

- **Alignment:** Course objectives must align with program objectives and syllabus requirements.
- **Bloom's Taxonomy:** Questions and assessments must cover all cognitive levels as per Bloom's Taxonomy.
- **Time Management:** Lesson plans must fit within the allocated time for the course.
- **Resource Availability:** Course materials must be based on available resources (textbooks, online materials, etc.).

4. Functional Requirements:

- Automatically generate lesson plans based on syllabus and course objectives.
- Create course objectives and map them to program objectives.
- Generate CO-PO mapping reports.
- Develop a question bank with questions categorized by topic, difficulty, and cognitive level.
- Compile course materials including slides, notes, and reference links.
- Generate reports to evaluate the effectiveness of the course and its alignment with program outcomes.

5. Non-Functional Requirements:

- **Scalability:** The system should handle multiple courses and programs.
- **Usability:** Intuitive interface for educators to input data and view outputs.
- **Efficiency:** The system should generate outputs within a reasonable time frame.
- **Accuracy:** Ensure accurate alignment of COs, POs, and syllabus requirements.

Stakeholders:

1. **Educators:** Input syllabus, course objectives, and program objectives; use the system to generate lesson plans and course materials.
2. **Students:** Access course materials and question banks for study and practice.
3. **Administrators:** Monitor the effectiveness of courses and their alignment with program objectives.
4. **Management:** Ensure the quality and consistency of course delivery and outcomes.

Challenges:

1. **Complexity:** Managing the alignment of course objectives, program objectives, and syllabus requirements.
2. **Resource Constraints:** Ensuring course materials are based on available resources.
3. **Time Management:** Creating lesson plans that fit within the allocated time for the course.
4. **Assessment Quality:** Developing a question bank that covers all cognitive levels and difficulty levels.

Deliverables:

1. A software application or tool for generating lesson plans, course materials, and assessments.
2. User manuals for educators, students, and administrators.
3. Testing and validation reports to ensure the system meets all requirements.

Success Criteria:

1. The system generates accurate and comprehensive lesson plans, course materials, and assessments.
2. Course objectives are effectively aligned with program objectives.
3. The question bank covers all cognitive levels and difficulty levels.
4. Reports provide valuable insights into the effectiveness of the course and its alignment with program outcomes.

Problem Definition Related Evaluation Criteria (Manual Evaluation):

1. **Alignment of COs and POs:**
    - Percentage of course objectives aligned with program objectives.
    - Accuracy of CO-PO mapping.
2. **Lesson Plan Effectiveness:**
    - Adherence to syllabus requirements.
    - Coverage of all topics within the allocated time.
3. **Question Bank Quality:**
    - Coverage of all cognitive levels (Bloom's Taxonomy).
    - Distribution of questions by difficulty level.
4. **Course Material Quality:**
    - Relevance and comprehensiveness of course materials.
    - Utilization of available resources.

5. **Report Accuracy:**
    - Accuracy of CO-PO mapping reports.
    - Effectiveness of lesson plan and question bank analysis reports..
6. **Efficiency:**
    - Time taken to generate lesson plans, course materials, and assessments.
    - Time taken to generate reports.

Programming Evaluation Criteria (Score generated by AI):

1. Test Cases implemented in code. (higher the number higher the score)
2. Minimum required number of Files.
3. Repeat Lines Density (higher the number lower the score)
4. Code Complexity (higher the number lower the score)
5. Security Rating (higher the number lower the score)
6. Scale Rating (higher the number lower the score)
7. Reliability Rating (higher the number higher the score)
8. Bugs per File (higher the number lower the score)
9. Code Smells per File (higher the number lower the score)
10. Vulnerabilities per File (higher the number lower the score)
11. How much concepts of AI/ML used in your code. (higher the number higher the score)
12. How much Code written through AI (higher the number higher the score)

Test Cases:

Test Case 1: Lesson Plan Generation

- **Input:** Syllabus with topics and subtopics, course duration (e.g., 12 weeks).
- **Expected Output:** A structured lesson plan with topics and subtopics distributed across weeks.
- **Evaluation:** Verify that all topics are covered within the allocated time.

Test Case 2: CO-PO Mapping

- **Input:** Course objectives and program objectives.
- **Expected Output:** A mapping of course objectives to program objectives.
- **Evaluation:** Verify that all course objectives are correctly mapped to program objectives.

Test Case 3: Question Bank Generation

- **Input:** Syllabus topics, cognitive levels, and difficulty levels.
- **Expected Output:** A question bank with questions categorized by topic, cognitive level, and difficulty level.
- **Evaluation:** Verify that questions cover all topics and cognitive levels.

Test Case 4: Course Material Compilation

- **Input:** Syllabus topics and available resources (textbooks, online materials).
- **Expected Output:** Comprehensive course materials including slides, notes, and reference links.
- **Evaluation:** Verify that course materials are relevant and comprehensive.

Test Case 5: Report Generation

- **Input:** Lesson plan, CO-PO mapping, question bank, and course materials.
- **Expected Output:** Reports on CO-PO mapping, lesson plan effectiveness, question bank analysis, and course material utilization.
- **Evaluation:** Verify that reports are accurate and provide valuable insights.

Test Case 6: Error Handling

- **Input:** Invalid syllabus data (e.g., missing topics or subtopics).
- **Expected Output:** Appropriate error messages and handling of invalid data.
- **Evaluation:** Verify that the system handles errors gracefully and provides meaningful feedback.

Example Reports:

1. **CO-PO Mapping Report:**
    - Detailed mapping of course objectives to program objectives.
    - Analysis of alignment and gaps.
2. **Lesson Plan Effectiveness Report:**
    - Coverage of syllabus topics.
    - Adherence to time allocation.
3. **Question Bank Analysis Report:**
    - Distribution of questions by cognitive level and difficulty level.
    - Coverage of all topics.
4. **Course Material Utilization Report:**
    - Utilization of textbooks, reference materials, and online resources.
    - Relevance and comprehensiveness of course materials.

AI SAMADHAN

PROBLEM - 2

"AI SAMADHAN" is more than just coding competitions; this is **catalysts for innovation, learning, and collaboration**. It empower individuals and organizations to harness the power of AI to solve pressing challenges and drive technological progress. By participating in AI **SAMADHAN**, you contribute to the growth of the AI ecosystem and make a meaningful impact on society.

Objective:

To design and implement an automated timetable generation system for an educational institute that efficiently schedules classes, faculty, and resources while adhering to various constraints and preferences. The system will also generate **complex reports** to analyze resource utilization, faculty workload, and student schedules.

Scope:

The system will cater to the scheduling needs of an institute offering multiple courses, subjects, and programs. It will allocate time slots for lectures, labs, tutorials, and other activities while ensuring optimal utilization of resources (classrooms, labs, and faculty) and minimizing conflicts. Additionally, the system will generate **complex reports** to provide insights into the timetable's effectiveness and resource allocation.

Key Requirements:

1. Input Data:

- **Courses and Subjects:** List of all courses, subjects, and their respective credits or hours.
- **Faculty Information:** Availability, preferred time slots, and subjects they teach.
- **Classrooms and Labs:** Availability and capacity of rooms and labs.
- **Student Groups:** Division of students into batches or sections.
- **Constraints:**
  - Maximum hours per day for a faculty member.
  - Minimum gaps between classes for students.
  - Room capacity and equipment requirements.
- **Preferences:**
  - Faculty preferences for specific time slots.
  - Student preferences for balanced schedules.

2. Output:

- A weekly timetable for each student group, faculty member, and room.
- Conflict-free schedules that meet all constraints.
- Visual representation of the timetable (e.g., grid view, export to PDF/Excel).

- **Complex Reports:**
  - Faculty Workload Report.
  - Classroom Utilization Report.
  - Student Schedule Report.
  - Conflict and Constraint Violation Report.

3. Constraints:

- **Hard Constraints (Must be satisfied):**
  - No overlapping classes for the same student group or faculty.
  - Room availability and capacity must be respected.
  - Faculty cannot be assigned more than their maximum allowed hours per day.
- **Soft Constraints (Preferred but not mandatory):**
  - Balanced workload for faculty and students.
  - Minimize gaps between classes for students.
  - Accommodate faculty preferences for specific time slots.

4. Functional Requirements:

- Automatically generate timetables based on input data.
- Allow manual adjustments to the generated timetable.
- Handle last-minute changes (e.g., faculty absence, room unavailability).
- Support multiple semesters or academic terms.
- Generate and export complex reports in various formats (PDF, Excel).

5. Non-Functional Requirements:

- **Scalability:** The system should handle a large number of courses, faculty, and rooms.
- **Usability:** Intuitive interface for administrators to input data and view timetables and reports.
- **Efficiency:** Timetable generation should be completed within a reasonable time frame (e.g., a few minutes).

Stakeholders:

1. **Administrators:** Input data, generate timetables, manage changes, and view reports.
2. **Faculty:** View their schedules and provide preferences.
3. **Students:** Access their class schedules.
4. **Management:** Ensure efficient resource utilization and adherence to academic goals.

Challenges:

1. **Complexity:** Managing a large number of variables (courses, faculty, rooms, etc.) and constraints.
2. **Conflicts:** Rescheduling in case of conflicts or last-minute changes.
3. **Optimization:** Balancing multiple objectives (e.g., faculty preferences, student convenience, and resource utilization).
4. **Reporting:** Generating accurate and insightful reports for decision-making.

Deliverables:

1. A software application or tool for timetable generation and reporting.
2. User manuals for administrators, faculty, and students.
3. Testing and validation reports to ensure the system meets all requirements.

Success Criteria:

1. The system generates conflict-free timetables that meet all hard constraints.
2. Faculty and students report satisfaction with the schedules.
3. The system reduces the time and effort required for timetable creation compared to manual methods.
4. The generated reports provide valuable insights into resource utilization and faculty workload.

Evaluation Criteria:

1. **Constraint Satisfaction:**
    - Percentage of hard constraints satisfied (e.g., no overlapping classes, room capacity respected).
    - Percentage of soft constraints satisfied (e.g., faculty preferences, balanced workload).
2. **User Satisfaction:**
    - Feedback from faculty and students on the generated timetables.
    - Satisfaction scores based on surveys or interviews.
3. **Resource Utilization:**
    - Classroom utilization rate (percentage of time classrooms are used).
    - Faculty workload distribution (variance in hours assigned to faculty).
4. **Efficiency:**
    - Time taken to generate the timetable.
    - Time taken to generate reports.
5. **Report Accuracy:**
    - Accuracy of faculty workload reports.

    - Accuracy of classroom utilization reports.
    - Accuracy of student schedule reports.
6. **Scalability:**
    - Performance of the system with increasing numbers of courses, faculty, and rooms.
    - Ability to handle multiple semesters or academic terms.
7. **Usability:**
    - Ease of use for administrators, faculty, and students.
    - Intuitiveness of the interface for inputting data and viewing timetables and reports.

Programming Evaluation Criteria (Score generated by AI):

1. Test Cases implemented in code. (higher the number higher the score)
2. Minimum required number of Files.
3. Repeat Lines Density (higher the number lower the score)
4. Code Complexity (higher the number lower the score)
5. Security Rating (higher the number lower the score)
6. Scale Rating (higher the number lower the score)
7. Reliability Rating (higher the number higher the score)
8. Bugs per File (higher the number lower the score)
9. Code Smells per File (higher the number lower the score)
10. Vulnerabilities per File (higher the number lower the score)
11. How much concepts of AI/ML used in your code. (higher the number higher the score)
12. How much Code written through AI (higher the number higher the score)

Test Cases:

Test Case 1: Timetable Generation

- **Input:** Courses, faculty, classrooms, and student groups.
- **Expected Output:** A conflict-free timetable for all student groups, faculty, and classrooms.
- **Evaluation:** Verify that all hard constraints are satisfied (e.g., no overlapping classes, room capacity respected).

Test Case 2: Faculty Workload Report

- **Input:** Generated timetable and faculty workload constraints.
- **Expected Output:** A report showing the workload distribution for each faculty member.
- **Evaluation:** Verify that no faculty member exceeds their maximum allowed hours per day.

Test Case 3: Classroom Utilization Report

- **Input:** Generated timetable and classroom capacities.
- **Expected Output:** A report showing the utilization rate for each classroom.
- **Evaluation:** Verify that no classroom is over utilized or underutilized.

Test Case 4: Student Schedule Report

- **Input:** Generated timetable and student groups.
- **Expected Output:** A report showing the schedule for each student group.
- **Evaluation:** Verify that all student groups have balanced schedules with minimal gaps between classes.

Test Case 5: Conflict and Constraint Violation Report

- **Input:** Generated timetable and constraints.
- **Expected Output:** A report listing any conflicts or constraint violations.
- **Evaluation:** Verify that the report accurately identifies and lists all conflicts and violations.

Test Case 6: Manual Adjustments

- **Input:** Manual adjustments to the generated timetable (e.g., changing a faculty assignment).
- **Expected Output:** Updated timetable and reports reflecting the changes.
- **Evaluation:** Verify that the system handles manual adjustments correctly and updates the reports accordingly.

Test Case 7: Error Handling

- **Input:** Invalid input data (e.g., missing faculty availability, incorrect room capacity).
- **Expected Output:** Appropriate error messages and handling of invalid data.
- **Evaluation:** Verify that the system handles errors gracefully and provides meaningful feedback.

Example Reports:

1. **Faculty Workload Report:**
    - Total hours assigned to each faculty member.
    - Comparison of assigned hours vs. maximum allowed hours.
    - Faculty satisfaction with their schedules.
2. **Classroom Utilization Report:**
    - Percentage of time each classroom is utilized.
    - List of underutilized or over utilized classrooms.

- Suggestions for optimizing classroom usage.

3. **Student Schedule Report:**
   - List of subjects assigned to each student group.
   - Gaps between classes for each student group.
   - Student satisfaction with their schedules.
4. **Conflict and Constraint Violation Report:**
   - List of conflicts (e.g., overlapping classes, room overcapacity).
   - List of constraint violations (e.g., faculty over-assigned hours).
   - Suggestions for resolving conflicts and violations.

In Case of any query contact Alok Manke (9426360744)

AI SAMADHAN

PROBLEM - 3

"AI SAMADHAN" is more than just coding competitions; this is **catalysts for innovation, learning, and collaboration**. It empower individuals and organizations to harness the power of AI to solve pressing challenges and drive technological progress. By participating in AI **SAMADHAN**, you contribute to the growth of the AI ecosystem and make a meaningful impact on society.

Objective:

To design and implement an AI-based system that evaluates student assignments submitted in **PDF format** by comparing them with a **model answer**. The system will provide marks out of 10 based on the **summary similarity** between the assignment and the model answer. Additionally, the system will generate **complex reports** to analyze assignment quality, student performance, and evaluation accuracy.

Scope:

The system will cater to the needs of educational institutions by automating the evaluation of assignments. It will:

1. Accept assignments in PDF format.
2. Extract text from the PDF.
3. Compare the assignment text with a model answer using AI-based **text summarization and similarity analysis**.
4. Assign marks out of 10 based on the similarity score.
5. Generate **complex reports** to provide insights into assignment quality and student performance.

Key Requirements:

1. Input Data:

- **Model Answer:** A reference answer provided by the instructor in text format.
- **Student Assignments:** Assignments submitted by students in PDF format.
- **Evaluation Criteria:**
    - Summary similarity between the assignment and the model answer.
    - Coverage of key points from the model answer.
    - Grammar and readability (optional).

2. Output:

- **Marks:** A score out of 10 for each assignment based on summary similarity.
- **Complex Reports:**

- Assignment Evaluation Report: Marks, similarity score, and feedback for each student.
- Class Performance Report: Overall performance of the class, including average marks and distribution.
- Key Point Coverage Report: Analysis of how well students covered key points from the model answer.
- Grammar and Readability Report (optional): Analysis of grammar and readability of assignments.

3. Constraints:

- **Accuracy:** The system must accurately evaluate assignments based on the model answer.
- **Scalability:** The system should handle a large number of assignments efficiently.
- **File Format:** Assignments must be submitted in PDF format.
- **Fairness:** The evaluation should be consistent and unbiased.

4. Functional Requirements:

- Extract text from PDF assignments.
- Summarize the assignment text and the model answer.
- Compare the summaries using AI-based similarity analysis.
- Assign marks out of 10 based on the similarity score.
- Generate complex reports for analysis and feedback.

5. Non-Functional Requirements:

- **Usability:** Intuitive interface for instructors to upload model answers and assignments.
- **Efficiency:** The system should evaluate assignments and generate reports within a reasonable time frame.
- **Security:** Ensure the privacy and security of student data.

Stakeholders:

1. **Instructors:** Upload model answers, evaluate assignments, and view reports.
2. **Students:** Submit assignments in PDF format and receive marks and feedback.
3. **Administrators:** Monitor the evaluation process and ensure system accuracy.
4. **Management:** Analyze class performance and make data-driven decisions.

Challenges:

1. **Text Extraction:** Accurately extracting text from PDF files, especially if they contain images or complex formatting.

2. **Summary Similarity:** Ensuring the AI model accurately summarizes and compares the assignment and model answer.
3. **Fairness:** Maintaining consistency and fairness in evaluation across all assignments.
4. **Scalability:** Handling a large number of assignments efficiently.

Deliverables:

1. A software application or tool for AI-based assignment evaluation.
2. User manuals for instructors, students, and administrators.
3. Testing and validation reports to ensure the system meets all requirements.

Success Criteria:

1. The system accurately evaluates assignments based on the model answer.
2. Students receive fair and consistent marks out of 10.
3. The system generates comprehensive and insightful reports.
4. Instructors and administrators report high satisfaction with the system.

Problem Definition related Evaluation Criteria(Manual Evaluation):

1. **Accuracy of Evaluation:**
    - Percentage of assignments evaluated correctly based on the model answer.
    - Consistency in marks assigned across similar assignments.
2. **User Satisfaction:**
    - Feedback from instructors and students on the evaluation process and results.
    - Satisfaction scores based on surveys or interviews.
3. **Report Quality:**
    - Accuracy and comprehensiveness of complex reports.
    - Usefulness of reports for analyzing class performance and assignment quality.
4. **Efficiency:**
    - Time taken to evaluate assignments and generate reports.
    - System performance with a large number of assignments.
5. **Scalability:**
    - Ability to handle increasing numbers of assignments and users.
    - Performance under high load.

Programming Evaluation Criteria (Score generated by AI):

1. Test Cases implemented in code. (higher the number higher the score)
2. Minimum required number of Files.

3. Repeat Lines Density (higher the number lower the score)
4. Code Complexity (higher the number lower the score)
5. Security Rating (higher the number lower the score)
6. Scale Rating (higher the number lower the score)
7. Reliability Rating (higher the number higher the score)
8. Bugs per File (higher the number lower the score)
9. Code Smells per File (higher the number lower the score)
10. Vulnerabilities per File (higher the number lower the score)
11. How much concepts of AI/ML used in your code. (higher the number higher the score)
12. How much Code written through AI (higher the number higher the score)

Test Cases:

Test Case 1: Text Extraction from PDF

- **Input:** A PDF file containing text and images.
- **Expected Output:** Accurate extraction of text from the PDF.
- **Evaluation:** Verify that the extracted text matches the content of the PDF.

Test Case 2: Summary Similarity Analysis

- **Input:** Extracted text from a student assignment and a model answer.
- **Expected Output:** A similarity score between the assignment and the model answer.
- **Evaluation:** Verify that the similarity score accurately reflects the content similarity.

Test Case 3: Marks Assignment

- **Input:** Similarity score between the assignment and the model answer.
- **Expected Output:** A mark out of 10 based on the similarity score.
- **Evaluation:** Verify that the mark is consistent with the similarity score.

Test Case 4: Assignment Evaluation Report

- **Input:** Marks and similarity scores for multiple assignments.
- **Expected Output:** A report showing marks, similarity scores, and feedback for each assignment.
- **Evaluation:** Verify that the report accurately reflects the evaluation results.

Test Case 5: Class Performance Report

- **Input:** Marks and similarity scores for all assignments in a class.

- **Expected Output:** A report showing overall class performance, including average marks and distribution.
- **Evaluation:** Verify that the report provides accurate and useful insights into class performance.

Test Case 6: Key Point Coverage Report

- **Input:** Extracted text from student assignments and the model answer.
- **Expected Output:** A report showing how well students covered key points from the model answer.
- **Evaluation:** Verify that the report accurately identifies key point coverage.

Test Case 7: Error Handling

- **Input:** Invalid PDF files (e.g., corrupted files, non-PDF formats).
- **Expected Output:** Appropriate error messages and handling of invalid files.
- **Evaluation:** Verify that the system handles errors gracefully and provides meaningful feedback.

Example Reports:

1. **Assignment Evaluation Report:**
    - Student Name: John Doe
    - Assignment: Assignment 1
    - Similarity Score: 0.85
    - Marks: 8.5/10
    - Feedback: "Good coverage of key points, but some areas need more detail."
2. **Class Performance Report:**
    - Class: Introduction to AI
    - Average Marks: 7.8/10
    - Marks Distribution: 20% (9-10), 50% (7-8), 30% (5-6)
3. **Key Point Coverage Report:**
    - Key Point 1: Covered by 90% of students.
    - Key Point 2: Covered by 70% of students.
    - Key Point 3: Covered by 50% of students.
4. **Grammar and Readability Report (optional):**
    - Grammar Score: 8/10
    - Readability Score: 7.5/10
    - Feedback: "Good grammar, but sentences could be more concise."

AI SAMADHAN

PROBLEM - 4

"AI SAMADHAN" is more than just coding competitions; this is **catalysts for innovation, learning, and collaboration**. It empower individuals and organizations to harness the power of AI to solve pressing challenges and drive technological progress. By participating in AI **SAMADHAN**, you contribute to the growth of the AI ecosystem and make a meaningful impact on society.

Objective:

To design and implement an automated system that optimizes the scheduling of **theory and practical exams** for both **regular and remedial** students in a university. The system will ensure that:

1. Exams are scheduled within a maximum duration of **3 hours**.
2. The **minimum number of days** is used for the exam schedule.
3. There are **no conflicts** between students, classrooms, and exams.
4. The system generates **complex reports** to analyze the schedule's efficiency and resource utilization.

Scope:

The system will cater to the needs of a university by automating the scheduling of exams. It will:

1. Schedule **theory and practical exams** for regular and remedial students.
2. Ensure that exams are scheduled within a maximum duration of 2 hours.
3. Minimize the number of days required for the exam schedule.
4. Avoid conflicts between students, classrooms, and exams.
5. Generate **complex reports** to provide insights into the schedule's efficiency and resource utilization.

Key Requirements:

1. Input Data:

- **Courses and Exams:** List of courses, theory exams, and practical exams.
- **Students:** List of students enrolled in regular and remedial courses.
- **Classrooms:** List of available classrooms with their capacities.
- **Constraints:**
    - Maximum exam duration: 2 hours.
    - No overlapping exams for the same student.
    - Classroom capacity must not be exceeded.
    - Minimum number of days for the exam schedule.
    - Maximum number of shifts: 3

- **Preferences:**
  - Preferred time slots for exams (e.g., morning, afternoon).

2. Output:

- **Exam Schedule:** A conflict-free schedule for theory and practical exams.
- **Complex Reports:**
  - Exam Schedule Report: Detailed schedule for each exam, student, and classroom.
  - Resource Utilization Report: Analysis of classroom utilization and exam distribution.
  - Conflict and Constraint Violation Report: List of conflicts and violations (if any).
  - Student Exam Schedule Report: Personalized schedule for each student.

3. Constraints:

- **Hard Constraints (Must be satisfied):**
  - No overlapping exams for the same student.
  - Classroom capacity must not be exceeded.
  - Exams must be scheduled within 2 hours.
- **Soft Constraints (Preferred but not mandatory):**
  - Minimize the number of days required for the exam schedule.
  - Accommodate preferred time slots for exams.

4. Functional Requirements:

- Automatically generate an optimized exam schedule based on input data.
- Allow manual adjustments to the generated schedule.
- Handle last-minute changes (e.g., student absence, classroom unavailability).
- Generate and export complex reports in various formats (PDF, Excel).

5. Non-Functional Requirements:

- **Scalability:** The system should handle a large number of students, courses, and classrooms.
- **Usability:** Intuitive interface for administrators to input data and view schedules and reports.
- **Efficiency:** The system should generate the schedule and reports within a reasonable time frame.

Stakeholders:

1. **Administrators:** Input data, generate schedules, manage changes, and view reports.
2. **Students:** Access their personalized exam schedules.
3. **Faculty:** View exam schedules and ensure no conflicts with their availability.

4. **Management:** Ensure efficient resource utilization and adherence to academic goals.

Challenges:

1. **Complexity:** Managing a large number of variables (students, courses, classrooms) and constraints.
2. **Conflicts:** Resolving conflicts between students, classrooms, and exams.
3. **Optimization:** Minimizing the number of days required for the exam schedule.
4. **Reporting:** Generating accurate and insightful reports for decision-making.

Deliverables:

1. A software application or tool for optimized exam scheduling.
2. User manuals for administrators, students, and faculty.
3. Testing and validation reports to ensure the system meets all requirements.

Success Criteria:

1. The system generates a conflict-free exam schedule that meets all hard constraints.
2. The schedule uses the minimum number of days required.
3. Students and faculty report satisfaction with the schedule.
4. The system reduces the time and effort required for exam scheduling compared to manual methods.
5. The generated reports provide valuable insights into resource utilization and schedule efficiency.

Problem Definition Related Evaluation Criteria (Manual Checking):

1. **Constraint Satisfaction:**
    - Percentage of hard constraints satisfied (e.g., no overlapping exams, classroom capacity respected).
    - Percentage of soft constraints satisfied (e.g., preferred time slots, minimum number of days).
2. **User Satisfaction:**
    - Feedback from students and faculty on the exam schedule.
    - Satisfaction scores based on surveys or interviews.
3. **Resource Utilization:**
    - Classroom utilization rate (percentage of time classrooms are used).

    - Exam distribution across days and time slots.
4. **Efficiency:**
    - Time taken to generate the exam schedule.
    - Time taken to generate reports.
5. **Report Accuracy:**
    - Accuracy of exam schedule reports.
    - Accuracy of resource utilization reports.
    - Accuracy of conflict and constraint violation reports.
6. **Scalability:**
    - Performance of the system with increasing numbers of students, courses, and classrooms.
    - Ability to handle multiple exam sessions (e.g., regular and remedial).
7. **Usability:**
    - Ease of use for administrators, students, and faculty.
    - Intuitiveness of the interface for inputting data and viewing schedules and reports.

Programming Evaluation Criteria (Score generated by AI):

1. Test Cases implemented in code. (higher the number higher the score)
2. Minimum required number of Files.
3. Repeat Lines Density (higher the number lower the score)
4. Code Complexity (higher the number lower the score)
5. Security Rating (higher the number lower the score)
6. Scale Rating (higher the number lower the score)
7. Reliability Rating (higher the number higher the score)
8. Bugs per File (higher the number lower the score)
9. Code Smells per File (higher the number lower the score)
10. Vulnerabilities per File (higher the number lower the score)
11. How much concepts of AI/ML used in your code. (higher the number higher the score)
12. How much Code written through AI (higher the number higher the score)

Test Cases:

Test Case 1: Exam Schedule Generation

- **Input:** Courses, students, classrooms, and constraints.
- **Expected Output:** A conflict-free exam schedule for theory and practical exams.
- **Evaluation:** Verify that all hard constraints are satisfied (e.g., no overlapping exams, classroom capacity respected).

Test Case 2: Resource Utilization Report

- **Input:** Generated exam schedule and classroom capacities.
- **Expected Output:** A report showing the utilization rate for each classroom.
- **Evaluation:** Verify that no classroom is overutilized or underutilized.

Test Case 3: Conflict and Constraint Violation Report

- **Input:** Generated exam schedule and constraints.
- **Expected Output:** A report listing any conflicts or constraint violations.
- **Evaluation:** Verify that the report accurately identifies and lists all conflicts and violations.

Test Case 4: Student Exam Schedule Report

- **Input:** Generated exam schedule and student enrollments.
- **Expected Output:** A personalized exam schedule for each student.
- **Evaluation:** Verify that the schedule accurately reflects the exams for each student.

Test Case 5: Manual Adjustments

- **Input:** Manual adjustments to the generated schedule (e.g., changing an exam time).
- **Expected Output:** Updated schedule and reports reflecting the changes.
- **Evaluation:** Verify that the system handles manual adjustments correctly and updates the reports accordingly.

Test Case 6: Error Handling

- **Input:** Invalid input data (e.g., missing student enrollments, incorrect classroom capacities).
- **Expected Output:** Appropriate error messages and handling of invalid data.
- **Evaluation:** Verify that the system handles errors gracefully and provides meaningful feedback.

Example Reports:

1. **Exam Schedule Report:**
    - Exam: Introduction to AI (Theory)
    - Date: 2023-10-15
    - Time: 10:00 AM - 12:00 PM
    - Classroom: Room 101
    - Students: 50
2. **Resource Utilization Report:**
    - Classroom: Room 101
    - Utilization Rate: 90%
    - Exams Scheduled: 5

3. **Conflict and Constraint Violation Report:**
    - Conflict: Overlapping exams for Student John Doe.
    - Constraint Violation: Classroom Room 101 over capacity for Exam Introduction to AI (Theory).
4. **Student Exam Schedule Report:**
    - Student: John Doe
    - Exam: Introduction to AI (Theory)
    - Date: 2023-10-15
    - Time: 10:00 AM - 12:00 PM
    - Classroom: Room 101

AI SAMADHAN

PROBLEM - 5

AI-Powered Smart Mark Distribution

This system automates the allocation of students' overall attendance bonus marks along with Hod bonus marks and Extra bonus marks to ensure students pass all subjects, maximize grades based on subject credits, and generate final graded reports in Excel.

Input Data :

1. Total number of subjects offered in exam
2. Total subjects with only theory out of total subjects and its theory credits.

   (100 marks for subject in theory component)
3. Total subjects with only practical out of total subjects and its practical credits

   (100 marks for subject in practical component)
4. Total subjects with theory and practical both and its theory credit and practical credit (50 marks for theory and 50 marks for practical component)
5. Student's score in each subject

   (theory and practical wise according to its offering in subject)
6. Student's overall attendance bonus marks of whole semester

   (including all subjects attendance)
7. Grades and grade points according to final marks after distribution of all bonus marks (attendance bonus marks,Hod bonus marks and extra bonus marks) for f

Marks Range	Grade	Grade Points
0–34	F	0
35–39	E	4
40–44	D	4.5
45–49	C	5
50–54	B	5.5
55–59	B+	6
60–64	B++	6.5
65–69	A	7
70–74	A+	7.5
75–79	A++	8
80–84	O	8.5
85–89	O+	9
90–94	O++	9.5
95–100	O+++	10

Table A

Constraints:

1. **Maximum Overall attendance marks Rule:**

   Out of obtained overall attendance bonus marks,

- Maximum 7 marks from overall attendance bonus marks can be allocated in each subject with theory only.
- Maximum 7 marks from overall attendance bonus marks can be allocated in each subject with practical only.
- Maximum 14 marks from overall attendance bonus marks can be allocated in each subject with theory and practical both.(Maximum 7 marks in each theory and Practical Component)

2. **Passing surety** : First priority is students passing all subjects.
   Passing criteria is,
   Only Theory subject : 35/100 in theory
   Only Practical subject : 35/100 in practical
   Combined subject : 17.5/50 in theory and 17.5/50 in practical

3. **HoD Bonus Rule** : If student is fail in exactly 1 subject by 1 or 2 marks after distribution of overall attendance bonus marks then HoD bonus of 1 or 2 (can not be given in fraction) will be given to pass that subject

4. **Extra Bonus rule** : If a student passes in all subjects after allocation of overall attendance bonus without HoD bonus marks then student will be awarded 2 marks as an extra bonus mark individually in all subjects

   Note: 2 Extra bonus marks are given subject wise , not componentwise.
   i.e.Don't allocate 2 extra bonus marks in theory and 2 extra bonus marks in practical individually.
   So, Total marks obtained = actually scored marks by student + overall bonus marks + HoD bonus marks + Extra bonus marks

5. **Priority to Credits**: Second Priority is to allocate the remaining overall attendance bonus marks (if any overall attendance bonus mark is remaining to distribute) in total marks obtained to maximize final SPI

6. **Maximize SPI :** After smart allocation of overall attendance bonus, Hod bonus and Extra Bonus marks for each student in subject S1,S2,...,Sn with credits C1,C2,..,Cn and grade points g1,g2,..,gn as in *Table A* such that final SPI is maximum, SPI is calculated by following formula

$$SPI = (g1 * C1 + g2 * C2 + .. + gn * Cn) / (C1 + C2 + .. + Cn)$$

7. **Excel Integration:** Inputs read from Excel and outputs written back. (Sample Excel File is given)

Tasks to perform:

Task 1: AI-Driven Attendance Bonus Allocation

Definition:

1. Take input from Excel File : input.xlsx

2. Distribute attendance bonus marks to: Ensure students pass all subjects (theory/practical: ≥35/100; combined: ≥17.5 per theory and practical component).

3. Allocate the HoD Bonus and Extra Bonus if applicable.

Task 2: AI -Driven SPI Maximization and reporting

Definition:

1. Distribute remaining overall attendance bonus marks in each subject such that final SPI is maximum.

2. Generate Report in Excel sheet output.xlsx

Programming Evaluation Criteria (Score generated by AI):

1. Test Cases implemented in code. (higher the number higher the score)
2. Minimum required number of Files.
3. Repeat Lines Density (higher the number lower the score)
4. Code Complexity (higher the number lower the score)
5. Security Rating (higher the number lower the score)
6. Scale Rating (higher the number lower the score)
7. Reliability Rating (higher the number higher the score)
8. Bugs per File (higher the number lower the score)
9. Code Smells per File (higher the number lower the score)
10. Vulnerabilities per File (higher the number lower the score)
11. How much concepts of AI/ML used in your code. (higher the number higher the score)
12. How much Code written through AI (higher the number higher the score)

In case of any query Contact : Jayson Shukla (9099701954)