EduCompiler User Manual

Introduction

EduCompiler is an interactive compiler designed to process educational functions related to course management, grading, and study planning. This guide provides a comprehensive overview of its functionalities, interface, and available functions.

The functions which user can interact with are:

1. Student Performance Functions

- calculateGPA(grades): Calculates the Grade Point Average (GPA) based on grades.
- calculateCGPA(grades): Calculates the Cumulative Grade Point Average (CGPA) from all semesters.
- calculateFinalGrade(midTermGrade, finalExamGrade): Calculates the final grade based on midterm and final exam grades.
- calculatePercentage(marksObtained, totalMarks): Calculates the percentage of marks obtained in an exam.
- checkPassFail(grade): Checks whether the grade is passing or failing.
- calculateGrade(percentage): Converts the percentage to a letter grade (A, B, C, etc.).
- calculateSubjectAverage(subjectGrades): Calculates the average grade in a particular subject.
- calculateClassAverage(grades): Calculates the average grade of the entire class.
- trackStudentProgress(studentID): Tracks a student's performance over multiple exams or subjects.
- isStudentEligibleForScholarship(gpa, extracurriculars): Checks if a student is eligible for a scholarship based on GPA and extracurricular activities.

2. Study and Learning Plan Functions

- createStudyPlan(subject, studyHours, daysAvailable): Creates a study plan based on available study hours and number of days.
- adjustStudyPlanForDifficulty(subject, difficultyLevel): Adjusts the study plan based on the difficulty of the subject.

- suggestStudyMaterial(subject): Suggests study material (books, videos) for a particular subject.
- calculateStudyTime(subjectDifficulty, goals): Calculates the study time needed based on subject difficulty and learning goals.
- createStudySchedule(studyHoursPerDay, totalSubjects): Generates a study schedule by dividing study hours between subjects.
- calculateTimeToMasterSubject(totalHours, practiceRate): Estimates the time required to master a subject based on the total study hours and practice rate.
- checkStudyBreaks(studyTime): Checks if the student needs a break based on their study time.
- calculateRevisionTime(subject, revisionSessions): Calculates how much time is needed for revision based on the number of revision sessions.
- trackStudySessions(date, subject, hours): Tracks study sessions for each subject, including date and hours spent.
- suggestLearningTechniques(subject): Suggests effective learning techniques based on the subject (e.g., spaced repetition, mind mapping).

3. Exam and Test Preparation Functions

- generateMockTest(subject, difficulty): Generates a mock test with questions based on subject and difficulty.
- calculateTestScore(correctAnswers, totalQuestions): Calculates the score on a test based on the number of correct answers.
- calculateExamDuration(totalQuestions, timePerQuestion): Calculates the total exam duration based on the number of questions and time per question.
- trackTestPerformance(previousScores): Tracks a student's performance across multiple tests to detect improvements or weaknesses.
- generateTestResult(studentAnswers, correctAnswers): Compares student answers to correct answers and generates a result.
- suggestTestPreparationSchedule(subject, examDate): Suggests a preparation schedule based on the subject and the exam date.
- calculateConfidenceLevel(previousScores, currentPerformance): Estimates the student’s confidence level before an exam.
- calculateExpectedScore(studyHours, practiceTestScores): Estimates the expected score on the exam based on study hours and practice test performance.

- trackStudySessionsForTest(date, subject, focusAreas): Tracks the study sessions, focusing on the areas of the subject to be studied.
- generatePastExamQuestions(subject): Generates past exam questions for practice in a specific subject.

4. Academic Resource and Material Functions

- recommendBooks(subject): Suggests books for studying a particular subject.
- suggestOnlineCourses(subject): Suggests online courses for further learning in a subject.
- findResearchPapers(subject): Finds and suggests relevant research papers for a given subject.
- generateFlashcards(subject, topic): Creates flashcards to aid in memorizing key concepts in a subject.
- searchForTutorialVideos(String subject):
- generateQuizzes(String subject):
- suggestPodcasts(String subject):
- recommendStudyGroups(String subject):
- generateMindMap(String subject, String topic):
- createSummaryNotes(String subject, String topic):

5. Course Management and Academic Tracking Functions

- createCourseSchedule(String classes, String semester):
- calculateCourseCompletionPercentage(int totalCredits, int completedCredits):
- checkPrerequisite(String course, String[] studentCourses):
- generateStudentReportCard(int studentID):
- generateCourseFeedback(int courseID):
- trackStudentAttendance(int studentID, String classDate):
- generateClassRoster(int courseID):
- calculateCourseGrade(int courseID, int studentID):
- assignHomework(int studentID, String homeworkDetails):
- createCourseOutline(int courseID, String[] topics):

Interface Overview

The EduCompiler interface consists of the following components:

1. Function Selection Dropdown: Allows users to select a specific function.
2. Input Area: Users enter data corresponding to the selected function.
3. Run Button: Executes the selected function with the provided input.
4. Output Area: Displays the results of the executed function.
5. Clear Button: Clears the input and output areas.
6. Tabs:
    - Tokenization Code: Displays tokenization code of the functions.
    - Three Address Code: Shows the intermediate representation of the compiled code.

How to Use EduCompiler

1. Select a function from the dropdown list.
2. Enter the required input in the input area.
3. Click the Run button to execute the function.
4. View the output in the Output Area.
5. To reset, click the Clear button.