Day 2 : Design

### **Introduction:**

On Day2 we focus on designing the core structure of the E-Learning Platform. This involves creating the blueprint for the application using UML diagrams that define the relationships between entities and specify the data fields and methods needed for each class.

The main goal is to provide a clear structure that ensures maintainability, scalability, and proper separation of concerns in the implementation phase.

### **Classes to Implement**

1. **Student** – Represents the student’s profile and activities like submitting assignments and viewing results.
2. **Assignment** – Contains information about the assignment, including its title, description, submission file, and submission date.
3. **EvaluationEngine** – Handles the evaluation logic such as running test cases, calculating scores, and generating reports.

### **Class Responsibilities and Methods**

#### **Student**

1. Fields:  
   1. studentId: String – Unique identifier for the student.
   2. name: String – Name of the student.
   3. email: String – Email address.
   4. assignments: List<Assignment> – List of assignments submitted by the student.
2. Methods:  
   1. submitAssignment(assignment: Assignment): void – Allows the student to submit an assignment.
   2. viewResults(): void – Allows the student to view evaluation results.

#### **Assignment**

1. Fields:  
   1. assignmentId: String – Unique identifier for the assignment.
   2. title: String – Title of the assignment.
   3. description: String – Description of the assignment.
   4. submissionDate: Date – Date of submission.
   5. filePath: String – Path to the submitted file.
2. Methods:  
   1. getAssignmentDetails(): String – Returns assignment details.

#### **EvaluationEngine**

1. Methods:  
   1. runTestCases(assignment: Assignment): int – Runs test cases for the assignment.
   2. calculateScore(testResult: int): int – Calculates the score based on test results.
   3. generateReport(student: Student, assignment: Assignment, score: int): void – Generates and displays the final report.

### **Relationships Between Classes**

* A Student can submit multiple Assignments → One-to-many relationship.
* Each Assignment is evaluated by EvaluationEngine → One-to-one relationship during evaluation.

**UMLClassDiagram2:**

@startuml

class Student {

- studentId: String

- name: String

- email: String

- assignments: List<Assignment>

+ submitAssignment(assignment: Assignment): void

+ viewResults(): void

}

class Assignment {

- assignmentId: String

- title: String

- description: String

- submissionDate: Date

- filePath: String

+ getAssignmentDetails(): String

}

class EvaluationEngine {

+ runTestCases(assignment: Assignment): int

+ calculateScore(testResult: int): int

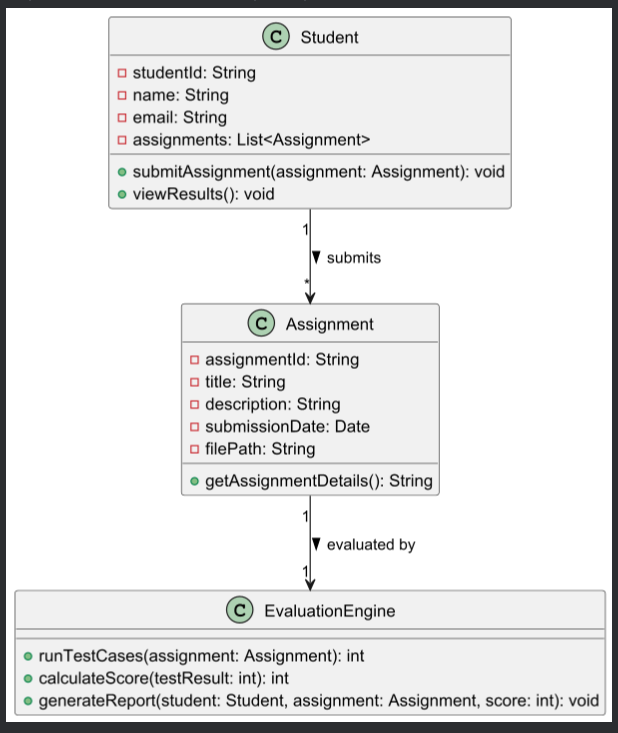
+ generateReport(student: Student, assignment: Assignment, score: int): void

}

Student "1" --> "\*" Assignment : submits >

Assignment "1" --> "1" EvaluationEngine : evaluated by >

@enduml



**UML Sequence Diagram:**

@startuml

actor Student

participant Assignment

participant EvaluationEngine

Student -> Assignment : submitAssignment()

Assignment -> EvaluationEngine : runTestCases()

EvaluationEngine --> Assignment : testResult

Assignment -> EvaluationEngine : calculateScore(testResult)

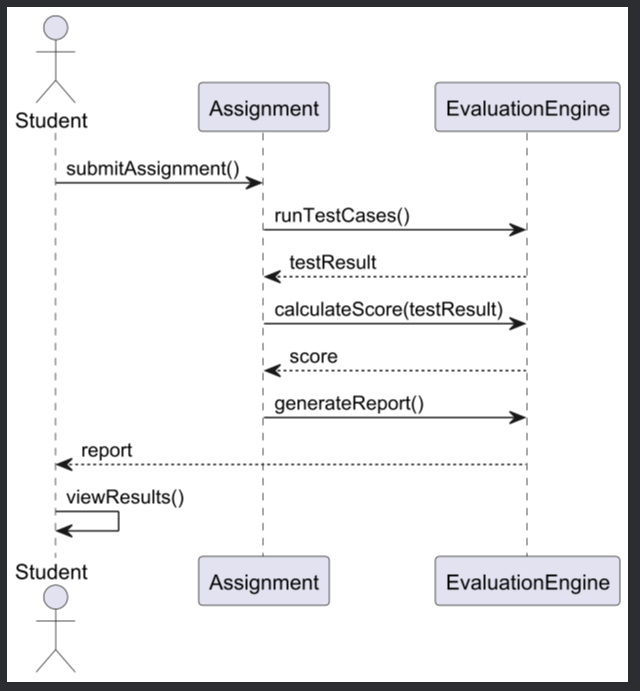
EvaluationEngine --> Assignment : score

Assignment -> EvaluationEngine : generateReport()

EvaluationEngine --> Student : report

Student -> Student : viewResults()

@enduml



**Day 2 – Conclusion:**

* Today, we successfully designed the structure of our E-Learning Platform by defining the main classes and their relationships.
* We created UML diagrams using PlantUML code for both the Class Diagram and Sequence Diagram.
* The relationships between Student, Assignment, and EvaluationEngine were clearly specified, laying a solid foundation for implementation.

This design will guide us through the coding process, ensuring proper architecture and functionality in the next phases of the project.