Day 8: BDD Auto Evaluation Scenario

## **Objective (one-liner for doc)**

* Creating a human-readable feature describing: Upload → Evaluate → Persist → Cache.
* Implementing Cucumber step definitions which call your existing EvaluationEngine logic.
* To allow running tests locally without AWS, provide a InMemoryScoreRepository and InMemoryLogRepository stub. Tests will use the stubs by default. (If you have AWS credentials and want to test with DynamoDB, I’ll show that option too.)
* Run the Cucumber feature with JUnit; confirm that a ScoreRecord is saved and the ScoreCache contains it.
* Save test results and include scenario output in your doc.

1. **PlantUML**

**ClassDiagram5.puml**

@startuml

package "Core" {

class Student

class Assignment

class CandidateSolution

class TestCase

class TestResult

class ScoreRecord

}

package "Evaluation" {

class TestCaseRunner

class EvaluationEngine {

+ evaluateAndPersist(student,assignment,solution) : int

+ getScores(studentId) : List<ScoreRecord>

}

class ScoreCache

}

package "Persistence" {

class DynamoDbScoreRepository

class DynamoDbLogRepository

class InMemoryScoreRepository

class InMemoryLogRepository

}

Student --> Assignment

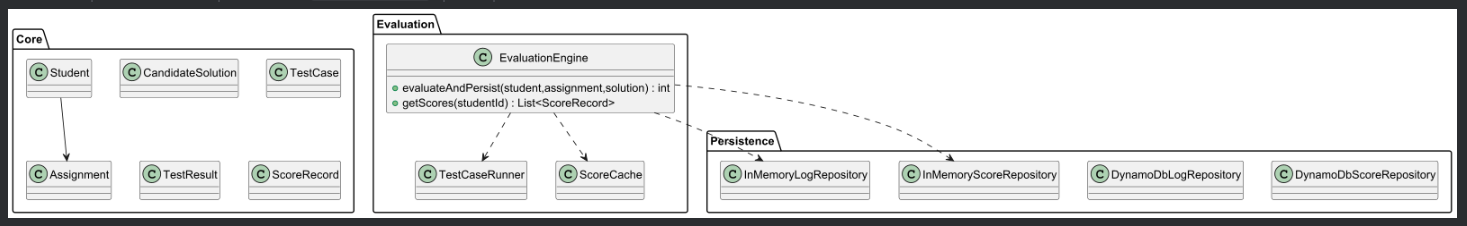
EvaluationEngine ..> TestCaseRunner

EvaluationEngine ..> ScoreCache

EvaluationEngine ..> InMemoryScoreRepository

EvaluationEngine ..> InMemoryLogRepository

@enduml



**SequenceDiagram6.puml**

@startuml

actor Tester

participant "Feature Runner" as FR

participant "EvaluationEngine" as Engine

participant "ScoreCache" as Cache

participant "Repository" as Repo

participant "LogRepo" as Log

Tester -> FR : Given student S001 and assignment A001

Tester -> FR : When the student submits a demo solution

FR -> Engine : evaluateAndPersist(student, assignment, solution)

Engine -> Repo : save(scoreRecord)

Engine -> Log : saveLog(start/result)

Engine -> Cache : put(scoreRecord)

FR -> Engine : getScores(S001)

Engine -> Cache : getByStudentId(S001)

alt cache hit

Cache --> FR : return cached result

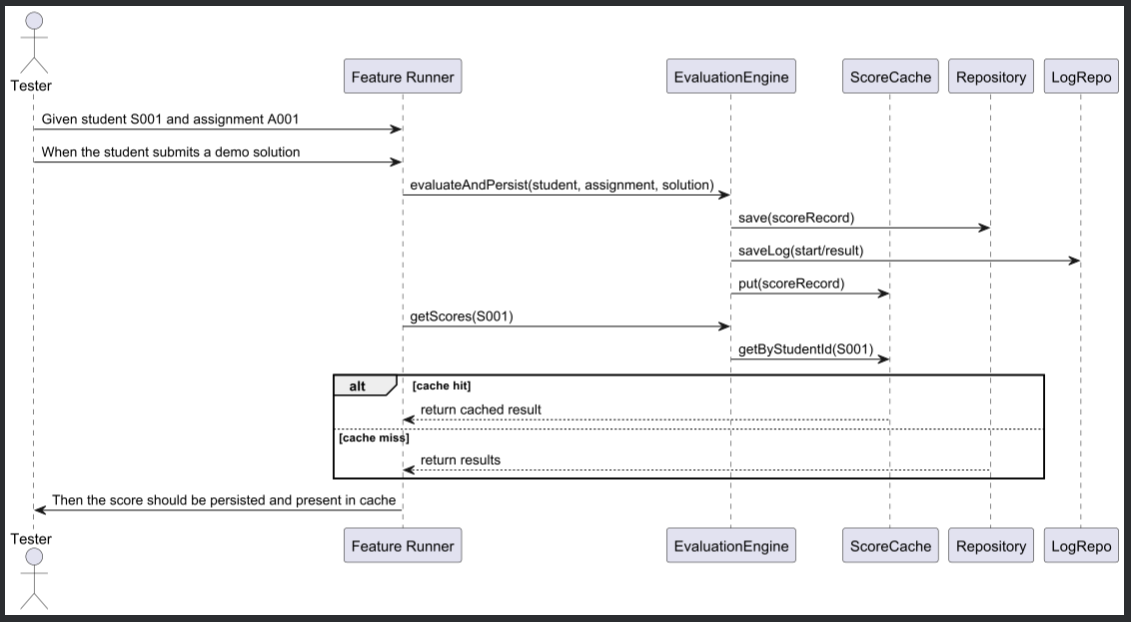
else cache miss

Repo --> FR : return results

end

FR -> Tester : Then the score should be persisted and present in cache

@enduml



1. **Adding Files**

1) Feature file (human-readable)

2) In-memory repo stubs (recommended for local testing)

3) Step definitions (wired to in-memory stubs)

4) Minimal adapter wrappers

* DynamoDbScoreRepositoryAdapter.java
* DynamoDbLogRepositoryAdapter.java

5) Cucumber runner (JUnit Platform)

1. **pom.xml additions (test dependencies)**

<!-- Cucumber JUnit Platform Engine -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit-platform-engine</artifactId>

<version>8.14.1</version>

<scope>test</scope>

</dependency>

<!-- Cucumber Java -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>8.14.1</version>

<scope>test</scope>

</dependency>

<!-- JUnit Jupiter (if not already present for tests) -->

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

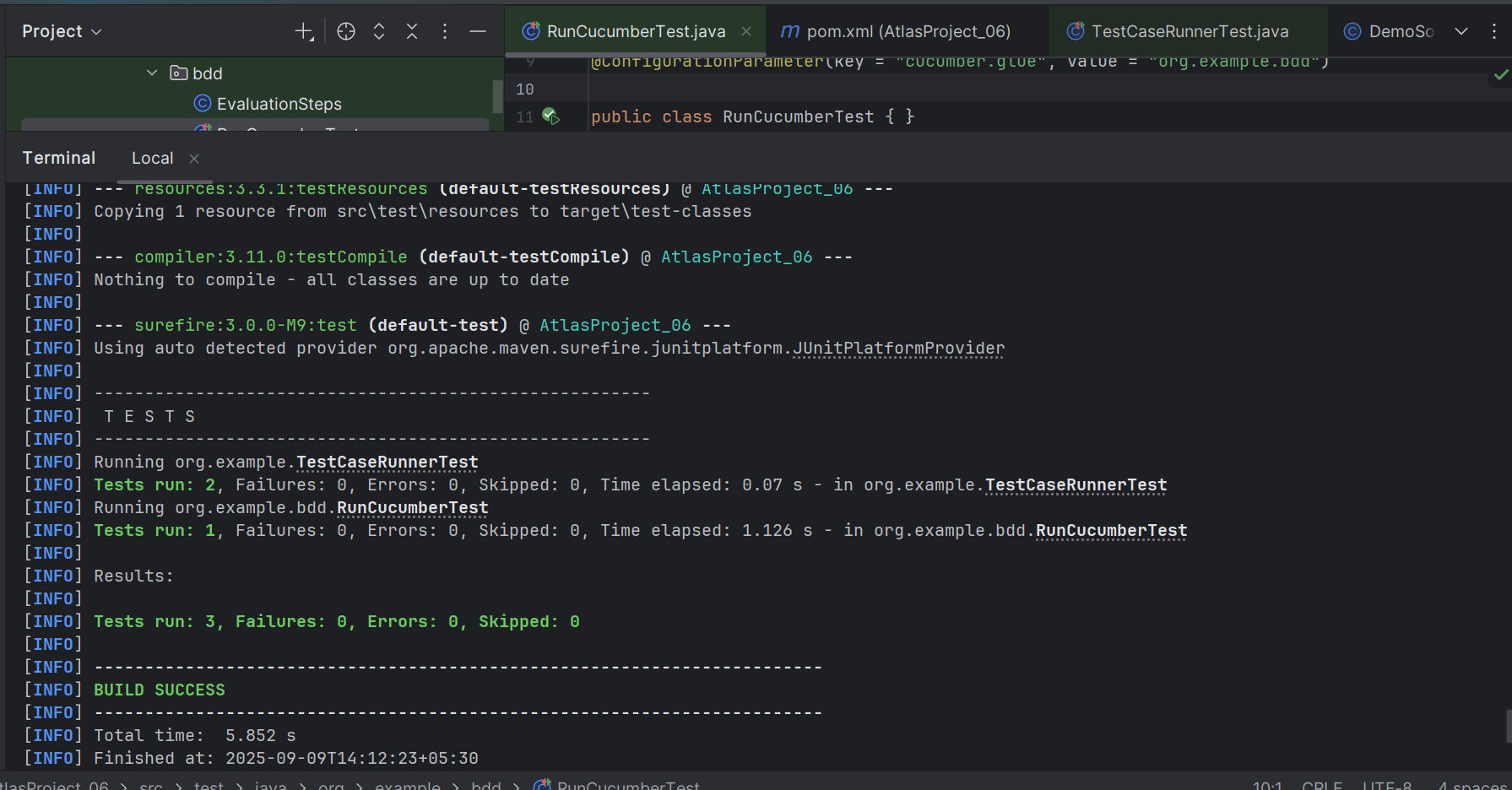
<version>5.9.3</version>

<scope>test</scope>

</dependency>

**Run**

* mvn test
* mvn -Dtest=org.example.bdd.RunCucumberTest test



**Day 8 – Conclusion:**

* Implemented BDD tests for the auto-evaluation workflow using Cucumber and JUnit.
* Added a human-readable feature that describes upload → evaluate → persist → cache.
* Implemented step definitions that invoke the real evaluation pipeline; tests use in-memory adapters by default to avoid AWS dependency during development.
* The test verifies that evaluation persists a ScoreRecord and that the in-memory ScoreCache contains the entry.
* These scenarios can be executed locally and included in CI to validate the core flow automatically.