Day 6 : Store unstructured logs

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

Store unstructured evaluation logs in DynamoDB so we can keep detailed event traces (evaluation start/finish, errors, stack traces, debug info) per student and submission.

## **High-level design & table schema**

Table name: EvaluationLogs

Primary Key:

* Partition key: studentId (String) — allows querying logs for a student
* Sort key: timestamp (String, ISO8601) — sorts logs chronologically

Attributes:

* studentId (S) — partition key
* timestamp (S) — sort key (ISO8601 2025-09-08T20:45:34Z)
* logId (S) — UUID for uniqueness (optional)
* submissionId (S) — e.g. A001\_20250907T...
* level (S) — INFO / WARN / ERROR / DEBUG
* message (S) — short message
* payload (S) — optional JSON string with extra details (stacktrace, test result array, etc.)

Why this schema

* Query by studentId to get the student's logs ordered by time.
* Keep payload flexible for unstructured info.

1. **PlantUML diagrams**

* **ClassDiagram3.puml**

@startuml

package "Domain" {

class LogRecord {

- studentId : String

- timestamp : String

- logId : String

- submissionId : String

- level : String

- message : String

- payload : String

+ toString() : String

}

}

package "Persistence" {

class DynamoDbLogRepository {

- ddb : DynamoDbClient

- tableName : String

+ createTableIfNotExists() : void

+ saveLog(log:LogRecord) : void

+ queryByStudentId(studentId:String, limit:int) : List<LogRecord>

}

}

package "Evaluation" {

class EvaluationEngine {

- logRepo : DynamoDbLogRepository

+ evaluateAndPersist(...) : int

+ logEvent(level, message, payload) : void

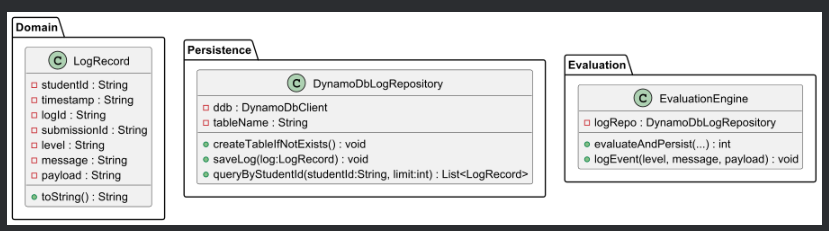
}

}

EvaluationEngine ..> DynamoDbLogRepository : logs to

DynamoDbLogRepository "maps to" LogRecord

@enduml

****

**SequenceDiagram4.puml**

@startuml

actor "Student" as S

participant "API / Controller" as API

participant "EvaluationEngine" as Engine

participant "DynamoDbLogRepository" as Repo

database "DynamoDB\nEvaluationLogs" as DynamoDB

S -> API : POST /upload/assignment (studentId, assignmentId, file)

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

Engine -> Repo : saveLog( studentId, timestamp, level=INFO, message="Evaluation started", payload={} )

Engine -> Repo : saveLog(... level=INFO, message="Test results", payload=[...])

Engine -> Repo : saveLog(... level=INFO, message="Score persisted", payload={score:85})

Repo -> DynamoDB : PutItem

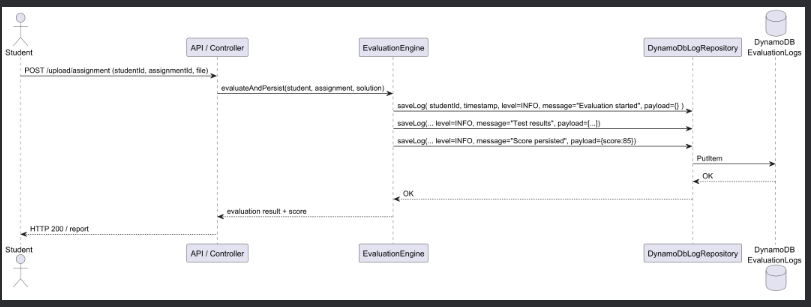
DynamoDB --> Repo : OK

Repo --> Engine : OK

Engine --> API : evaluation result + score

API --> S : HTTP 200 / report

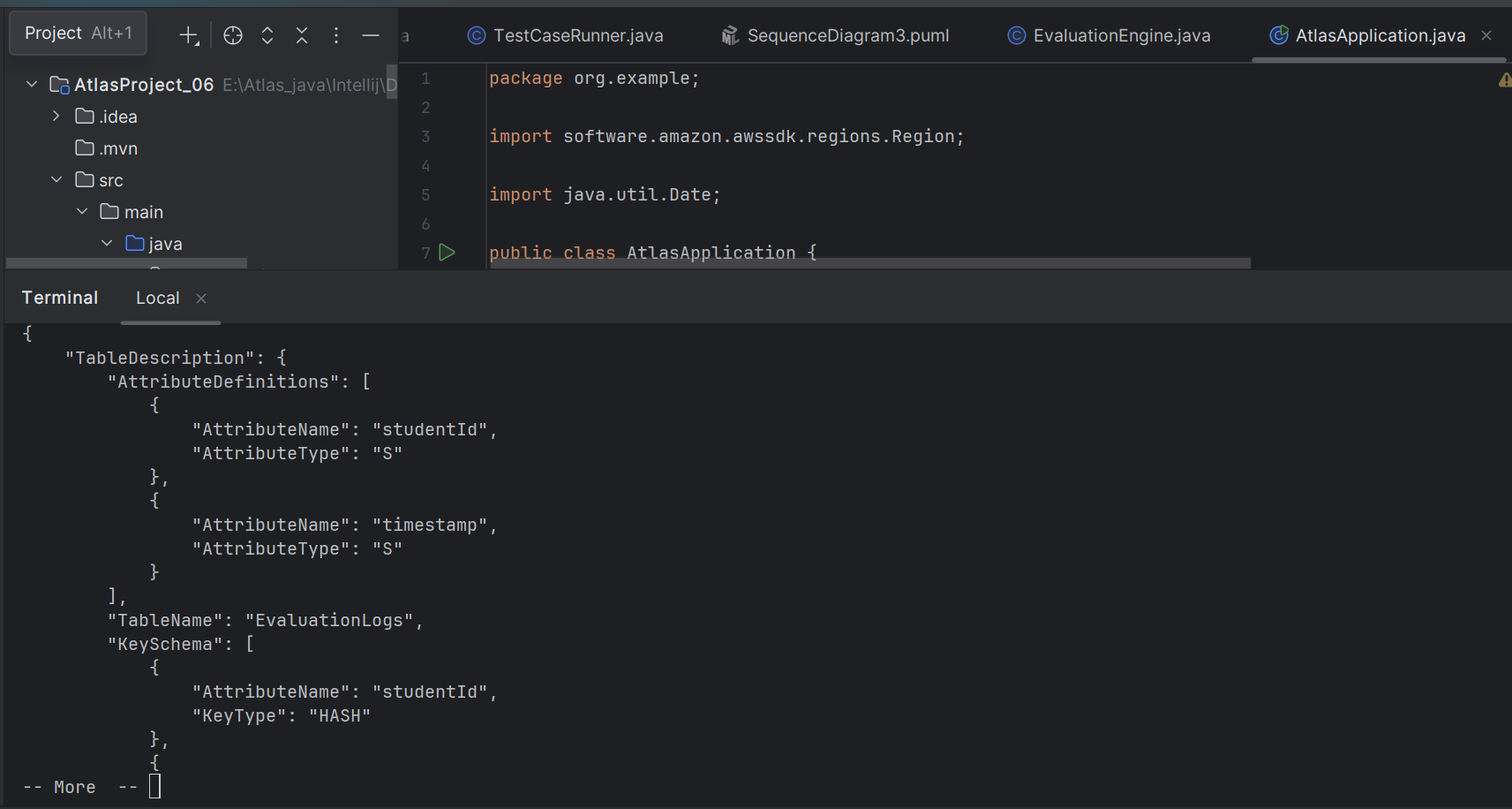
@enduml



1. **Java code**

* 4.1 LogRecord.java
* 4.2 DynamoDbLogRepository.java
* 4.3 — Integrate logging into EvaluationEngine
* 4.4 — Example wiring (AtlasApplication

1. **AWS CLI commands (create table manually)**



1. **IAM permissions needed**

AM principal must have DynamoDB permissions for the EvaluationLogs table: PutItem, Query, DescribeTable, CreateTable

{

"Effect":"Allow",

"Action":[

"dynamodb:PutItem",

"dynamodb:Query",

"dynamodb:DescribeTable",

"dynamodb:CreateTable"

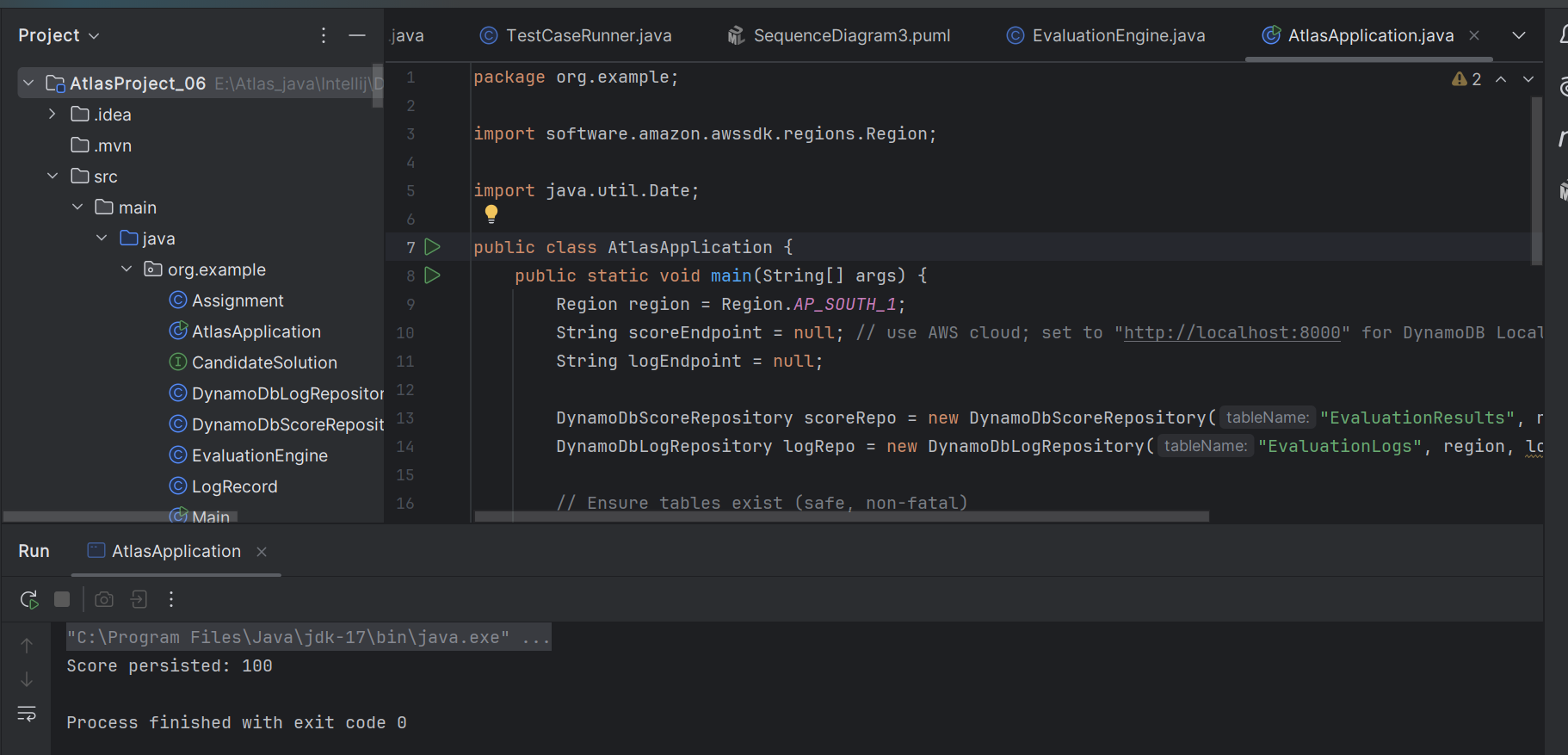
],

"Resource":"arn:aws:dynamodb:\*:\*:table/EvaluationLogs"

}

## **Testing (step-by-step)**

1. Ensure AWS credentials configured: aws sts get-caller-identity.
2. Create table via CLI or let createTableIfNotExists() create it.
3. Run AtlasApplication (IntelliJ run config or mvn spring-boot:run -Dspring-boot.run.main-class=org.example.AtlasApplication).
4. Watch console — you should see Log table is ACTIVE messages and Score persisted: X.



**Day 6 – Conclusion:**

* Implemented unstructured logging stored in DynamoDB (EvaluationLogs table) to capture evaluation events, errors, and debug payloads.
* Added LogRecord domain object and DynamoDbLogRepository with create-table, save, and query methods (with retries and wait-for-ACTIVE).
* Integrated logging calls into EvaluationEngine to record evaluation start, test results, and score persistence.
* Tested end-to-end and verified logs can be queried by studentId.