## **Project Day 3 – Java Implementation**

**Date:** 13-09-2025

**Name**: **Abdul Samad**

**Title: E-Learning Platform with Auto Evaluation.**

### **1. Objective**

Convert the Day 2 design into functioning Java code and create the Upload API that allows students to submit assignments for automatic evaluation.

**2. Key Deliverables**

* Implement core Java classes (Student, Assignment, EvaluationEngine) as POJOs.
* Set up a REST-style Upload API to accept file submissions and store metadata.
* Provide a lightweight server alternative to Spring Boot for simpler deployment.

**3. Technology Stack**

| **Layer** | **Technology** | **Reason** |
| --- | --- | --- |
| Language | **Java 17** | Modern features (records, var) with long-term support. |
| Web/API Server | **Javalin** (or Spark Java as another option) | Minimal, fast, single-jar deployment without heavy configuration. |
| File Storage | Local filesystem (initial) | Simple prototype; can later move to S3 or similar. |
| Database | MySQL or PostgreSQL | To persist assignment metadata and evaluation results. |

### **4. Class Implementations**

**Student.java**

public class Student {

private String studentId;

private String name;

private String email;

// constructors, getters, setters

public void submitAssignment(Assignment a) {

// logic to associate assignment with this student

}

}

**Assignment.Java**

import java.time.LocalDateTime;

public class Assignment {

private String assignmentId;

private String studentId;

private String title;

private String filePath;

private LocalDateTime submissionTime;

// constructors, getters, setters

public boolean isLate(LocalDateTime deadline) {

return submissionTime.isAfter(deadline);

}

}

**EvaluationEngine.Java**

public class EvaluationEngine {

public EvaluationResult evaluate(Assignment assignment) {

// placeholder for auto-grading logic

// e.g., run tests, check formatting, plagiarism scan

return new EvaluationResult(assignment.getAssignmentId(), 85, "Good job");

}

}

**EvaluationResult.Java**

public class EvaluationResult {

private final String assignmentId;

private final int score;

private final String remarks;

public EvaluationResult(String assignmentId, int score, String remarks) {

this.assignmentId = assignmentId;

this.score = score;

this.remarks = remarks;

}

// getters only

}

### **5. Upload API Using Javalin**

Server Code (Upload endpoint)

import io.javalin.Javalin;

import io.javalin.http.UploadedFile;

import java.nio.file.\*;

public class UploadServer {

public static void main(String[] args) {

Javalin app = Javalin.create().start(7000);

app.post("/api/assignments/upload", ctx -> {

String studentId = ctx.formParam("studentId");

UploadedFile file = ctx.uploadedFile("file");

if (file == null) {

ctx.status(400).result("File missing");

return;

}

Path destination = Paths.get("uploads/" + file.filename());

Files.copy(file.content(), destination, StandardCopyOption.REPLACE\_EXISTING);

Assignment assignment = new Assignment(

java.util.UUID.randomUUID().toString(),

studentId,

file.filename(),

destination.toString(),

java.time.LocalDateTime.now()

);

// Persist assignment info to DB (DAO code omitted)

// Trigger evaluation asynchronously

EvaluationEngine engine = new EvaluationEngine();

EvaluationResult result = engine.evaluate(assignment);

ctx.json(result);

});

}

}

Start server with java -jar target/yourapp.jar and test with Postman.

**6. Testing Activities**

* Verified server starts in under a second on a basic JVM.
* Uploaded .pdf and .zip files successfully.
* Checked that files are saved to /uploads directory with correct metadata.
* Ran EvaluationEngine mock scoring for each upload and returned JSON response.

**7. Next Steps**

1. **Database Integration** *Use JDBC/Hibernate or a lightweight library like JOOQ for persistence.*
2. **Real Evaluation Logic** *Integrate code compilation, unit testing, or plagiarism detection as needed.*
3. **Frontend UI** *Develop a simple HTML/JavaScript client or React app for students to view scores.*
4. **Authentication** *Add login tokens to secure the upload endpoint.*