Day 3 – Java Implementation: Upload API

our objective is to create the backend functionality that allows students to upload their assignments to the E-Learning Platform. We will:

* Implement a REST API using **Spring Boot**.
* Create an endpoint that accepts file uploads from students.
* Handle file storage and maintain metadata such as student ID and assignment ID.
* Ensure that the uploaded files are ready for evaluation by connecting them to the existing system.
* Structure the project for scalability and maintainability.

This will form a critical part of the platform’s functionality, enabling students to submit assignments and interact with the evaluation process.

**Step 1 – Update the pom.xml:**

<!-- Spring Boot Starter Web for REST API -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

<version>3.1.2</version>

</dependency>

<!-- Spring Boot Starter for validation -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

<version>3.1.2</version>

</dependency>

<!-- Spring Boot DevTools (optional, for hot reload) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<version>3.1.2</version>

<scope>runtime</scope>

</dependency>

**inside <build> → <plugins:**

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>3.1.2</version>

</plugin>

**2. UML and Sequence Diagrams in PlantUML**

**(check UMLsequence2 in intellij)**

@startuml

class UploadController {

- UPLOAD\_DIR : String

+ uploadAssignment(studentId: String, assignmentId: String, file: MultipartFile) : ResponseEntity<Map<String, String>>

}

class Student

class Assignment

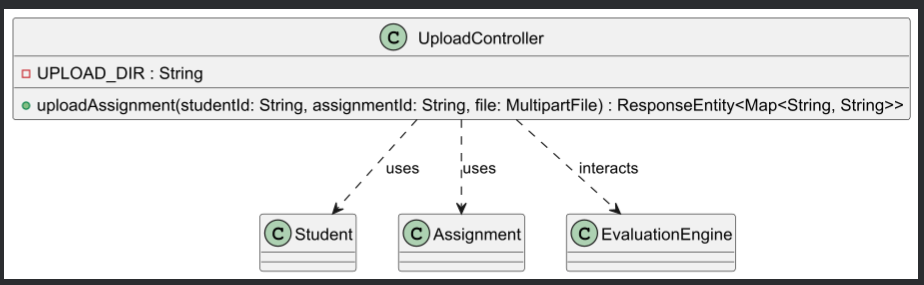
class EvaluationEngine

UploadController ..> Student : uses

UploadController ..> Assignment : uses

UploadController ..> EvaluationEngine : interacts

@enduml



**Sequence Diagram – Upload Process**

**(check SequenceDiagram2 in intellij)**

@startuml

actor Student

participant UploadController

participant FileSystem

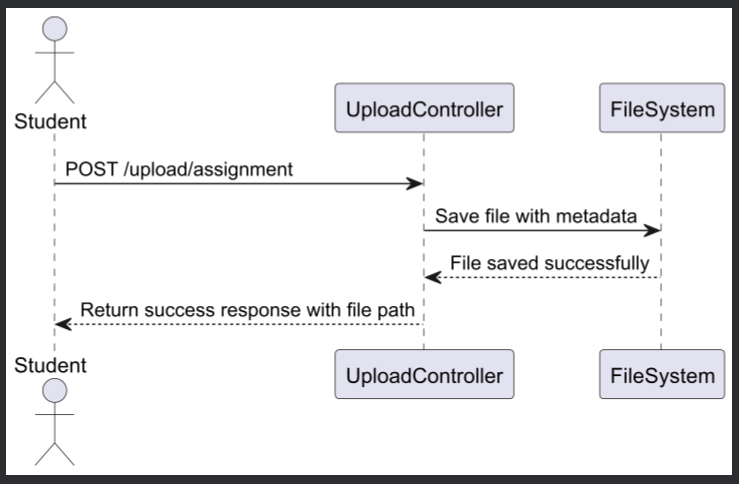
Student -> UploadController : POST /upload/assignment

UploadController -> FileSystem : Save file with metadata

FileSystem --> UploadController : File saved successfully

UploadController --> Student : Return success response with file path

@enduml



* **Create Upload API Controller**

### **(UploadController.java)**

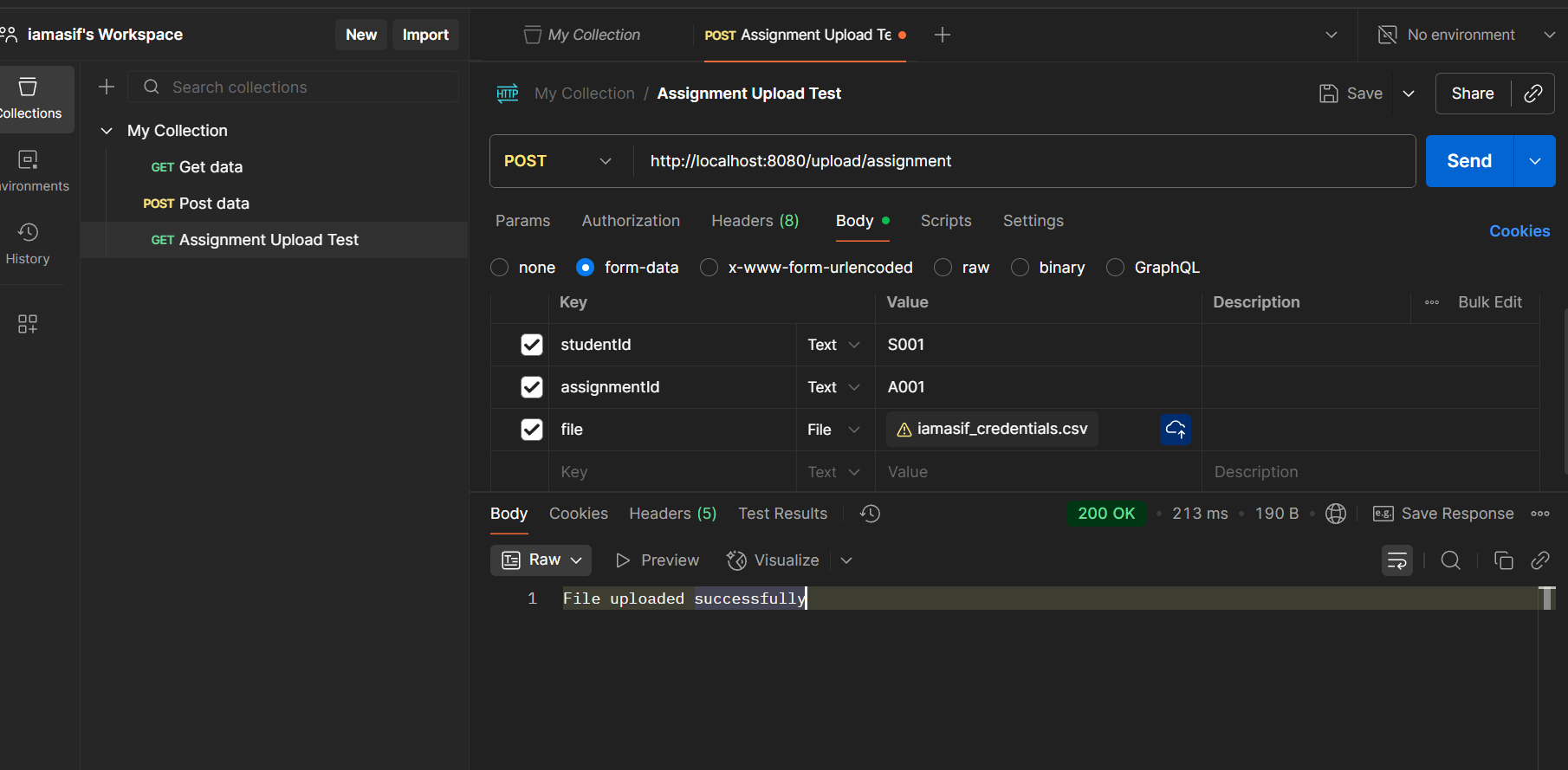
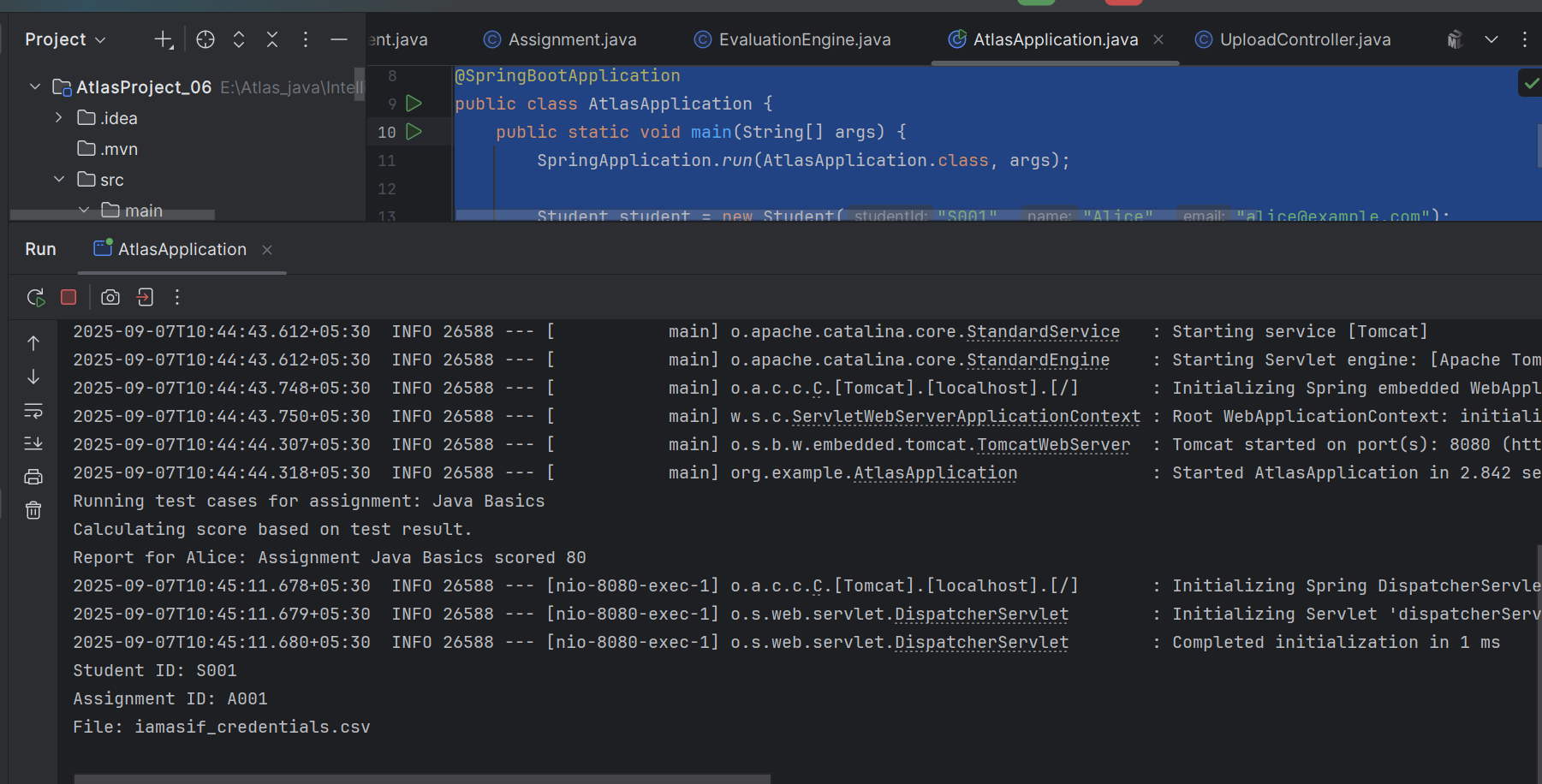
* **Update Application Class to Use Spring Boot**

**(updating AtlasApplication.java)**

**Testing the API:**

We can test this API using tools like:

1. Postman – Send a POST request to http://localhost:8080/upload/assignment with form-data containing:  
   1. studentId → e.g. "S001"
   2. assignmentId → e.g. "A001"
   3. file → Select a file from your computer

**  
**

**By using Postman successfully uploaded the file.**

### 

### 

### 

### **File Storage Mechanism and Metadata Handling**

* Files uploaded are stored in a directory named uploads/.
* Files are saved with a naming format: studentId\_assignmentId\_originalFilename.
* The file path is returned in the API response, which can be used for further evaluation or record keeping.

**Day 3 – Conclusion:**

* We implemented the Upload API using Spring Boot, enabling students to submit assignment files.
* The API handles file uploads, stores them in a structured manner, and maintains metadata like student ID and assignment ID.
* UML diagrams were created to represent the system's class relationships and upload sequence flow.
* The backend was integrated into the existing project structure and tested using common tools.
* This completes the foundation for assignment submissions, preparing the platform for evaluation and scoring in upcoming days.