# **Day 5 – Testing, Validation & Error Handling**

## **Overview**

On Day 5, the focus was on testing the existing modules, validating user input, and adding error-handling mechanisms. The objective was to make the system more robust by ensuring it can gracefully handle invalid or missing data during file uploads or API requests.

## **Tasks Completed**

1. Validated input fields in uploaded files (e.g., student ID, score, and percentage).  
2. Added custom exception handling in the controllers using a global exception handler.  
3. Implemented standardized error responses in JSON format.  
4. Tested all endpoints for various edge cases like missing fields, incorrect formats, and invalid data.  
5. Enhanced response messages for better readability.

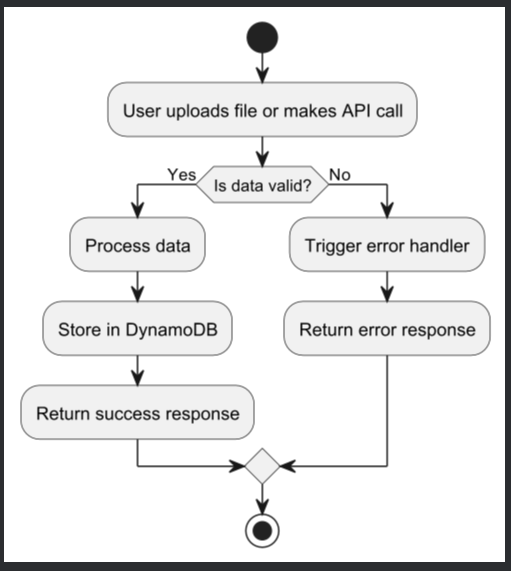
## **Error Handling Mechanism**

A centralized exception handler (`ExceptionHandlerController.java`) was created to manage runtime errors gracefully. This handler ensures that meaningful error messages are sent back to the client, improving debugging and API usability.

## **System Flow (Activity Diagram)**

## **day5.uml**

@startuml  
start  
:User uploads file or makes API call;  
if (Is data valid?) then (Yes)  
 :Process data;  
 :Store in DynamoDB;  
 :Return success response;  
else (No)  
 :Trigger error handler;  
 :Return error response;  
endif  
stop  
@enduml



## **Outcome of Day 5**

The application became more reliable with improved validation and error handling. All existing API endpoints were tested against real-world scenarios to ensure system stability.

## **Conclusion**

Day 5 strengthened the project’s backend by ensuring robustness and dependability. The validation and error-handling mechanisms made the system production-ready, laying a strong foundation for future integration with automated testing and evaluation workflows.