**MODULE - 7**

**Assignment 14: Implement Logging and Error Handling**

**Objective**

Enhance the backend of the IELTS Speaking Test platform by implementing middleware for request validation and structured logging. Capture API usage details and errors in logs for monitoring and debugging.

To tackle the assignment efficiently, let's break down the problem and outline the step-by-step approach for implementing each requirement. This method ensures that the solution is organized, optimal, and adheres to good programming practices.

**Step-by-Step Approach**

1. Request Validation Middleware:

Objective: Validate incoming requests for required fields and data formats, and reject invalid requests with appropriate error messages and status codes.

**Steps:**

1.Define Required Fields and Data Formats: Identify the fields and their formats that each API endpoint requires.

2.Create a Middleware Function:

This function will intercept the incoming requests.

Use a schema or a validation library (e.g., pydantic or marshmallow) to check the presence and format of required fields.

3.Validate Requests:

If a request is valid, pass it to the next middleware or endpoint handler.

If a request is invalid, return an appropriate error message and status code (e.g., 400 Bad Request).

2. Structured Logging:

Objective: Use Python's logging module to capture structured logs including metadata like timestamps, request paths, status codes, and error details.

**Steps:**

1.Set Up the Logging Configuration:

Configure the logging module to capture the required metadata.

Use log formats to include details such as timestamps, request paths, HTTP methods, status codes, and error messages.

2.Integrate Logging in Middleware:

Log each incoming request with its metadata before processing it.

Log the outcome of the request (status code) after it is processed.

3. Error Handling Middleware:

Objective: Create custom error handlers for common HTTP errors, log all unhandled exceptions, and return user-friendly error messages.

**Steps:**

1.Define Custom Error Handlers:

Create functions to handle common HTTP errors like 400 (Bad Request), 404 (Not Found), and 500 (Internal Server Error).

Each error handler should return a user-friendly error message and log the error details.

2.Capture Unhandled Exceptions:

Implement a global error handler that will catch all unhandled exceptions.

Log the exception details and return a generic error message (e.g., 500 Internal Server Error).

4. Testing:

Objective: Simulate valid and invalid API requests to ensure the middleware and logging are working as expected.

**Steps:**

1.Write Test Cases:

Create test cases for valid and invalid requests for each endpoint.

Include tests that trigger different error handlers.

2.Simulate Requests:

Use a tool like pytest or Postman to simulate API requests.

Verify that the middleware correctly validates requests and the logging captures all necessary details.

3.Check Logs:

Inspect the generated logs to ensure all metadata is captured and errors are logged as expected.

**Submission Guidelines:**

Ensure that you:

1.Submit Updated Backend Files:

Include middleware implementations and logging configurations.

2.Provide Sample Log Outputs:

Attach logs demonstrating validation and error handling.

3.Include Test Cases and Results:

Provide the test cases used for simulation and their outcomes.

**Final Checklist:**

● Middleware for request validation is implemented and tested.

● Logging configuration captures all necessary metadata and is well-structured.

● Custom error handlers are in place and log unhandled exceptions.

● All relevant files, log samples, and test evidence are included in the submission.

By following these steps methodically, you will create an optimized and robust solution that satisfies all the assignment requirements.