CS 203: Software Tools & Techniques for AI
IIT Gandhinagar
Sem II - 2024-25

LAB 01

Total marks: 10
Submission deadline: Sunday, 12/01/2025 11:59:59 PM
Submission guidelines:

1. Code should be added to a GitHub repository, and the repository details should be shared.
2. Late submissions will be penalized 20% per day.
3. Prebuilt Flask app link: https://github.com/EshwarDhande/CS203_Lab_01.git
4. Plagarism will not be tolerated. Marks will be penalized for the same.
5. You have to submit Github repo link. Some of you may be called to show demo for the assignment.

Note: By submitting this assignment solution you confirm to follow the IITGN's honor code. We shall strictly penalize the submissions containing plagiarized text/code.

Objective

This lab will explore distributed tracing and telemetry in a pre-built Flask-based Course Information Portal. Students will focus on adding OpenTelemetry instrumentation, analyzing traces, and generating telemetry data.

Provided Setup
A basic Flask application will be provided, which includes the following:

1. **Course Catalog:** A page/route to list all courses.
2. **Browse a Course:** A page to view details of a selected course.

The application includes two pre-added courses. Students must analyze the codebase and manually implement the ability to add more courses by updating the application.

Features to Implement

1. Add Courses to the Catalog

- The portal should have an **"Add a New Course"** button on the catalog page. When clicked, it should lead the user to a form where they can manually enter details for a new course (e.g., course name, instructor, semester).
- Upon successful form submission, the course should be added to the portal's course catalog, and an appropriate log message should be generated.
- If required fields are missing (e.g., course name or instructor), the application should log an error and notify the user with an appropriate message.

2. OpenTelemetry Tracing

- **Add OpenTelemetry instrumentation** to trace user requests across the following routes:
    - Course catalog page.
    - Adding a new course (newly created route).
    - Browsing course details.
- Create **meaningful spans** for key operations (e.g., rendering the course catalog, handling form submissions, etc.).
- Add **trace attributes** (e.g., user IP, request methods, metadata about courses).

3. Exporting Telemetry Data to Jaeger

- **Set up Jaeger** as the tracing backend and ensure that telemetry data is exported to Jaeger.
- **Export telemetry data**, including:
    - Total requests to each route (e.g., how many times the catalog page is accessed, how many courses are added).
    - Total processing time for each operation.
    - Error counts (e.g., when fields are missing in the course addition form, or database connection issues occur).
- Ensure structured logging output (e.g., **JSON format**) and use appropriate logging levels (**INFO**, **WARNING**, **ERROR**).
- Logs should capture key events such as:
    - A user adding a course.
    - Errors during form submissions.
    - Successful rendering of pages.

4. Code Quality

- Maintain **clean, modular**, and readable code.

- Follow best practices for **logging**, **instrumentation**, and **Flask development**.
- Provide comments where necessary to explain OpenTelemetry spans and attributes.