CS203: Software Tools & Techniques for AI

Lab 01: Distributed Tracing and Telemetry (https://github.com/nishchaybhutoria/CS203_Lab_01-main)

IIT Gandhinagar, Sem II - 2024-25

Team Members

- Nishchay Bhutoria (23110222)
- Srivaths P (23110321)

Introduction

This lab submission demonstrates the use of distributed tracing and telemetry in a Flask-based Course Information Portal. Using **OpenTelemetry** and **Jaeger**, we add observability to the application, enabling efficient debugging, performance monitoring, and error tracking.

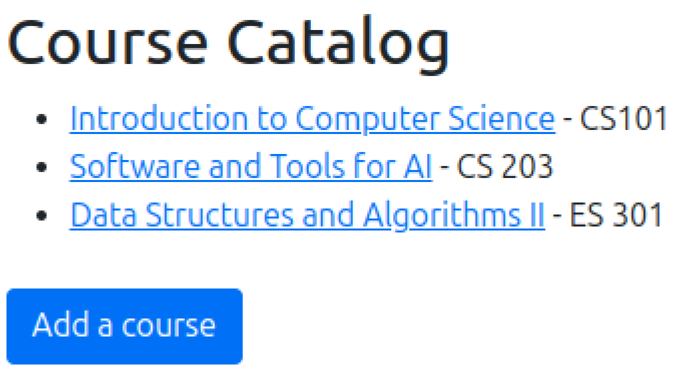
Technologies Used

- **Flask**: Web framework for building the portal.
- **OpenTelemetry**: Framework for generating and exporting telemetry data.
- **Jaeger**: Distributed tracing backend for storing and visualizing traces.
- **python-json-logger**: For structured JSON logging.

Features

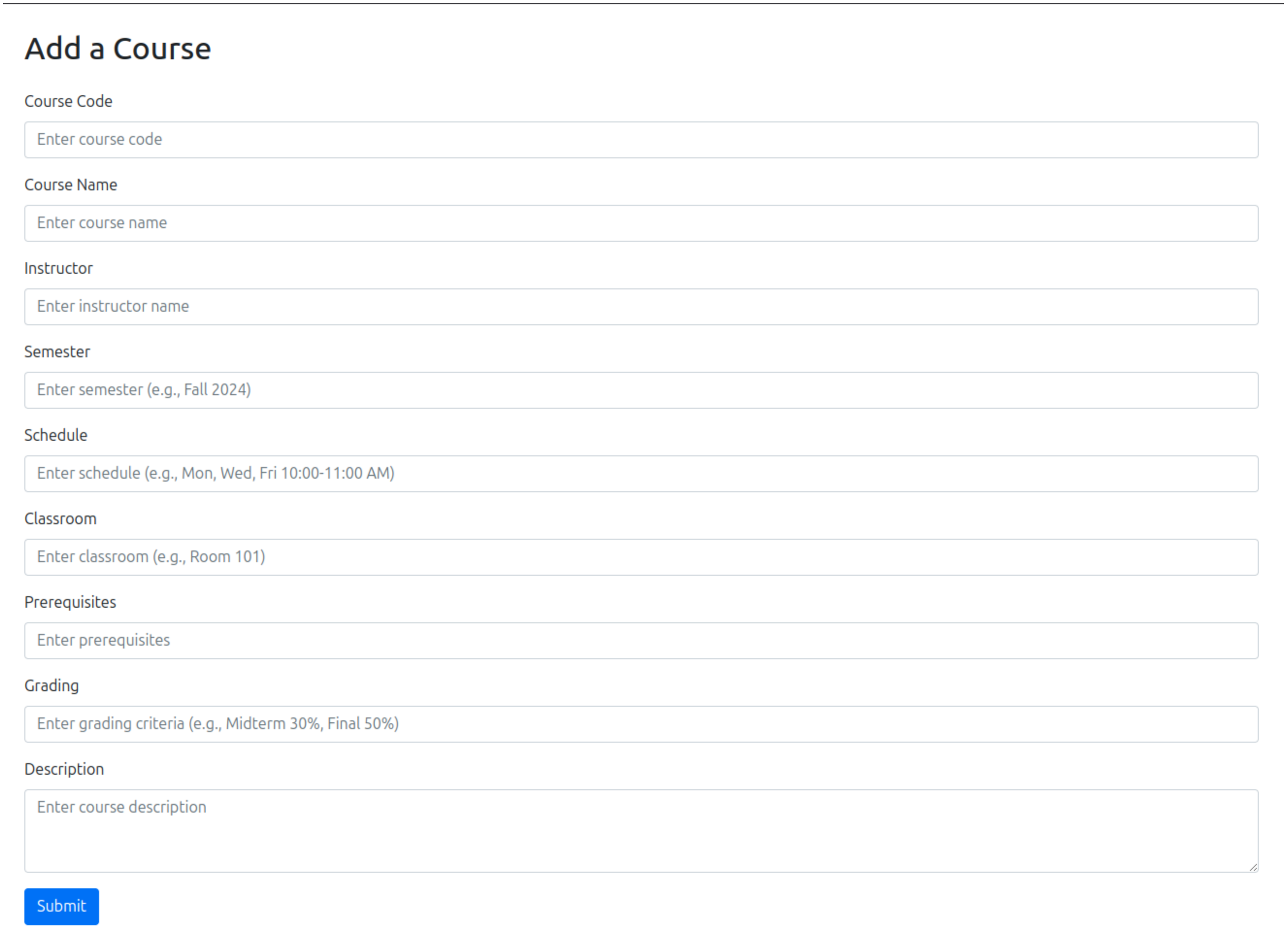
1. Add Courses to the Catalog

- An "Add a New Course" button on the catalog page navigates to a form for adding courses.



Course Catalog

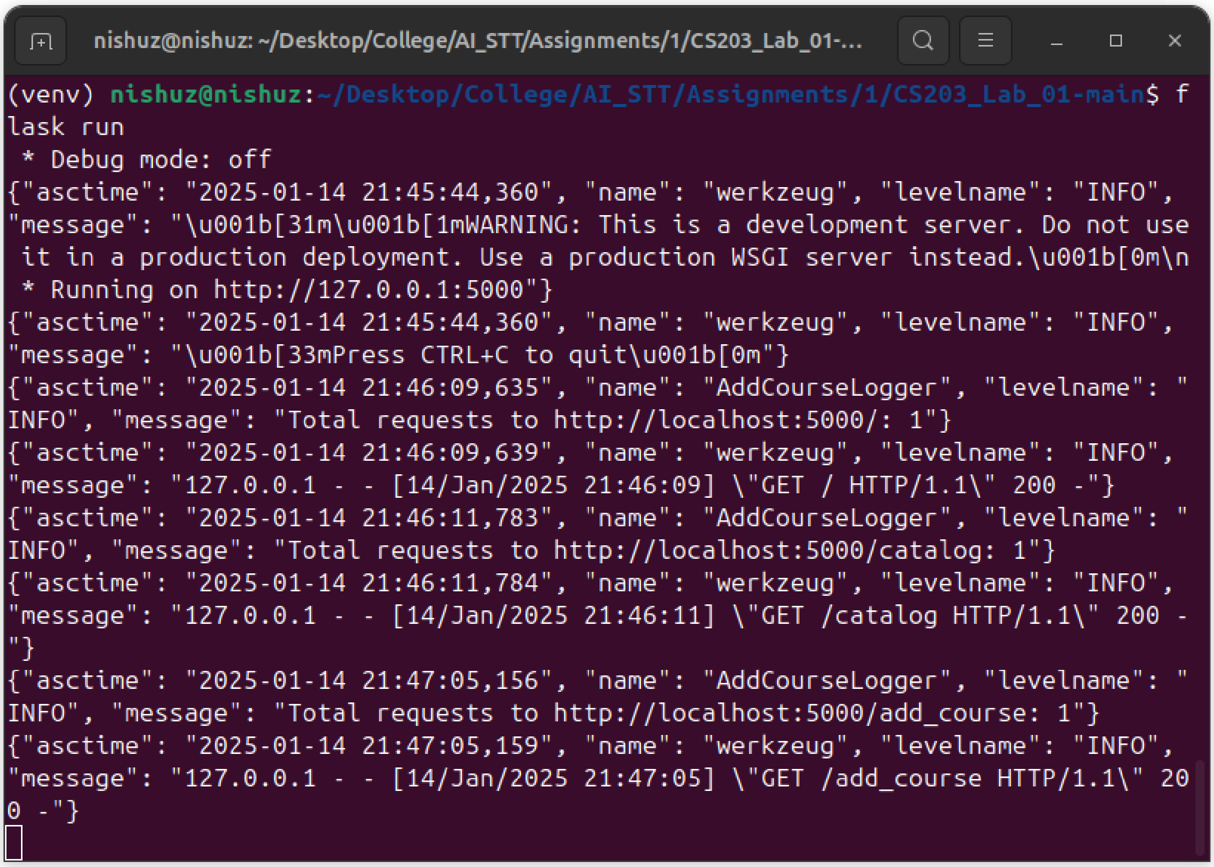
- Introduction to Computer Science - CS101
- Software and Tools for AI - CS 203
- Data Structures and Algorithms II - ES 301



- Logs are generated for:
    - **Successful course additions** (`INFO` level).
    - **Missing required fields** (`ERROR` level).
- Example log:

```
{
  "asctime": "2025-01-14 19:21:42,571",
  "name": "AddCourseLogger",
  "levelname": "INFO",
  "message": "Course CS 201 added successfully. All required fields are present."
}
```

- Console logs:


nishuz@nishuz: ~/Desktop/College/AI_STT/Assignments/1/CS203_Lab_01-...

```
(venv) nishuz@nishuz:~/Desktop/College/AI_STT/Assignments/1/CS203_Lab_01-main$ f
lask run
 * Debug mode: off
{"asctime": "2025-01-14 21:45:44,360", "name": "werkzeug", "levelname": "INFO",
"message": "\u001b[31m\u001b[1mWARNING: This is a development server. Do not use
 it in a production deployment. Use a production WSGI server instead.\u001b[0m\n
 * Running on http://127.0.0.1:5000"}
{"asctime": "2025-01-14 21:45:44,360", "name": "werkzeug", "levelname": "INFO",
"message": "\u001b[33mPress CTRL+C to quit\u001b[0m"}
{"asctime": "2025-01-14 21:46:09,635", "name": "AddCourseLogger", "levelname": "
INFO", "message": "Total requests to http://localhost:5000/: 1"}
{"asctime": "2025-01-14 21:46:09,639", "name": "werkzeug", "levelname": "INFO",
"message": "127.0.0.1 - - [14/Jan/2025 21:46:09] \"GET / HTTP/1.1\" 200 -"}
{"asctime": "2025-01-14 21:46:11,783", "name": "AddCourseLogger", "levelname": "
INFO", "message": "Total requests to http://localhost:5000/catalog: 1"}
{"asctime": "2025-01-14 21:46:11,784", "name": "werkzeug", "levelname": "INFO",
"message": "127.0.0.1 - - [14/Jan/2025 21:46:11] \"GET /catalog HTTP/1.1\" 200 -
"}
{"asctime": "2025-01-14 21:47:05,156", "name": "AddCourseLogger", "levelname": "
INFO", "message": "Total requests to http://localhost:5000/add_course: 1"}
{"asctime": "2025-01-14 21:47:05,159", "name": "werkzeug", "levelname": "INFO",
"message": "127.0.0.1 - - [14/Jan/2025 21:47:05] \"GET /add_course HTTP/1.1\" 20
0 -"}
```

---

2. OpenTelemetry Tracing

We instrumented the following routes to provide detailed traces for observability:

Route: `/catalog`

- **Span Name**: `render_catalog`
- **Attributes**:
  - `request_method`: HTTP method (e.g., `GET`).
  - `user_ip`: User's IP address.
  - `course_count`: Total number of courses loaded.

Route: `/add_course`

1. **Span Name**: `view_add_course_form` (GET request)
   - **Description**: Tracks the rendering of the course addition form.
   - **Attributes**:
     - `request_method`: HTTP method (e.g., `GET`).
     - `user_ip`: User's IP address.
2. **Span Name**: `submit_add_course_form` (POST request)
   - **Description**: Tracks the submission of the course addition form.

  - **Attributes**:
    - `request_method`: HTTP method (e.g., `POST`).
    - `user_ip`: User's IP address.
    - `course_code`: Code of the course being added.
    - `course_name`: Name of the course being added.

3. **Span Name**: `add_course_form_validation_error` (Child Span)

   - **Description**: Captures validation errors during form submission.
   - **Attributes**:
     - `missing_fields`: Fields that are missing in the form.
     - `error_count`: Number of errors (client-side or server-side).

4. **Span Name**: `count_errors` (Child Span)

   - **Description**: Tracks the total number of client-side and server-side errors.
   - **Attributes**:
     - `client_error_count`: Count of client-side errors.
     - `server_error_count`: Count of server-side errors.

Route: `/course/<code>`

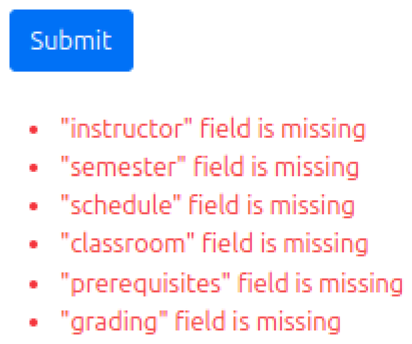
- **Span Name**: `view_course`
- **Attributes**:
  - `request_method`: HTTP method (e.g., `GET`).
  - `user_ip`: User's IP address.
  - `course_code`: Code of the course being viewed.
  - `error`: Set to `True` if the course is not found.
  - `error_message`: Error message if the course is not found.

---

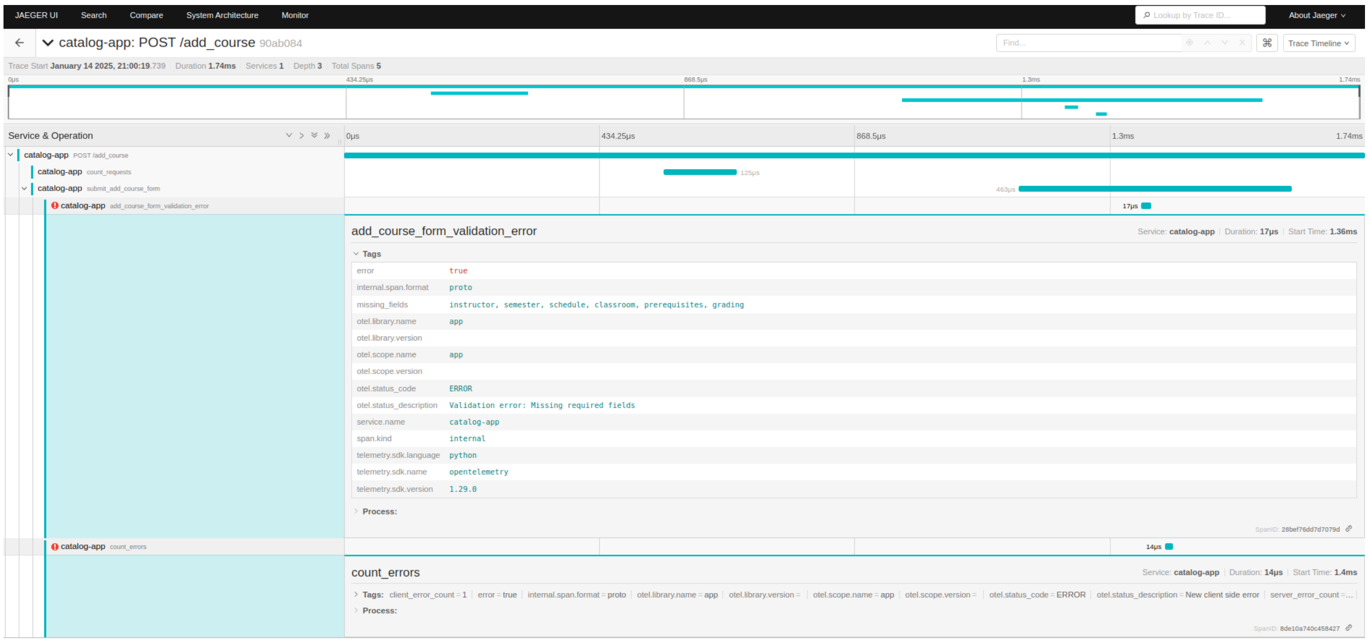
3. Exporting Telemetry Data to Jaeger

- Traces include:
  - Total requests to each route.
  - Processing time for operations.
  - Error counts for client-side and server-side errors.
- Screenshot of Jaeger traces:

Missing Fields:



- "instructor" field is missing
- "semester" field is missing
- "schedule" field is missing
- "classroom" field is missing
- "prerequisites" field is missing
- "grading" field is missing



catalog-app: POST /add_course 90ab084

Trace Start **January 14 2025, 21:00:19**.739 | Duration **1.74ms** | Services **1** | Depth **3** | Total Spans **5**

Service & Operation | 0μs | 434.25μs | 868.5μs | 1.3ms | 1.74ms

- catalog-app POST /add_course
  - catalog-app count_requests 125μs
  - catalog-app submit_add_course_form 463μs
    - catalog-app add_course_form_validation_error 17μs
    - catalog-app count_errors 14μs

add_course_form_validation_error

Service: **catalog-app** | Duration: **17μs** | Start Time: **1.36ms**

Tags

error	`true`
internal.span.format	`proto`
missing_fields	`instructor, semester, schedule, classroom, prerequisites, grading`
otel.library.name	`app`
otel.library.version	
otel.scope.name	`app`
otel.scope.version	
otel.status_code	`ERROR`
otel.status_description	`Validation error: Missing required fields`
service.name	`catalog-app`
span.kind	`internal`
telemetry.sdk.language	`python`
telemetry.sdk.name	`opentelemetry`
telemetry.sdk.version	`1.29.0`

Process:

SpanID: 28bef76dd7d7079d

count_errors

Service: **catalog-app** | Duration: **14μs** | Start Time: **1.4ms**

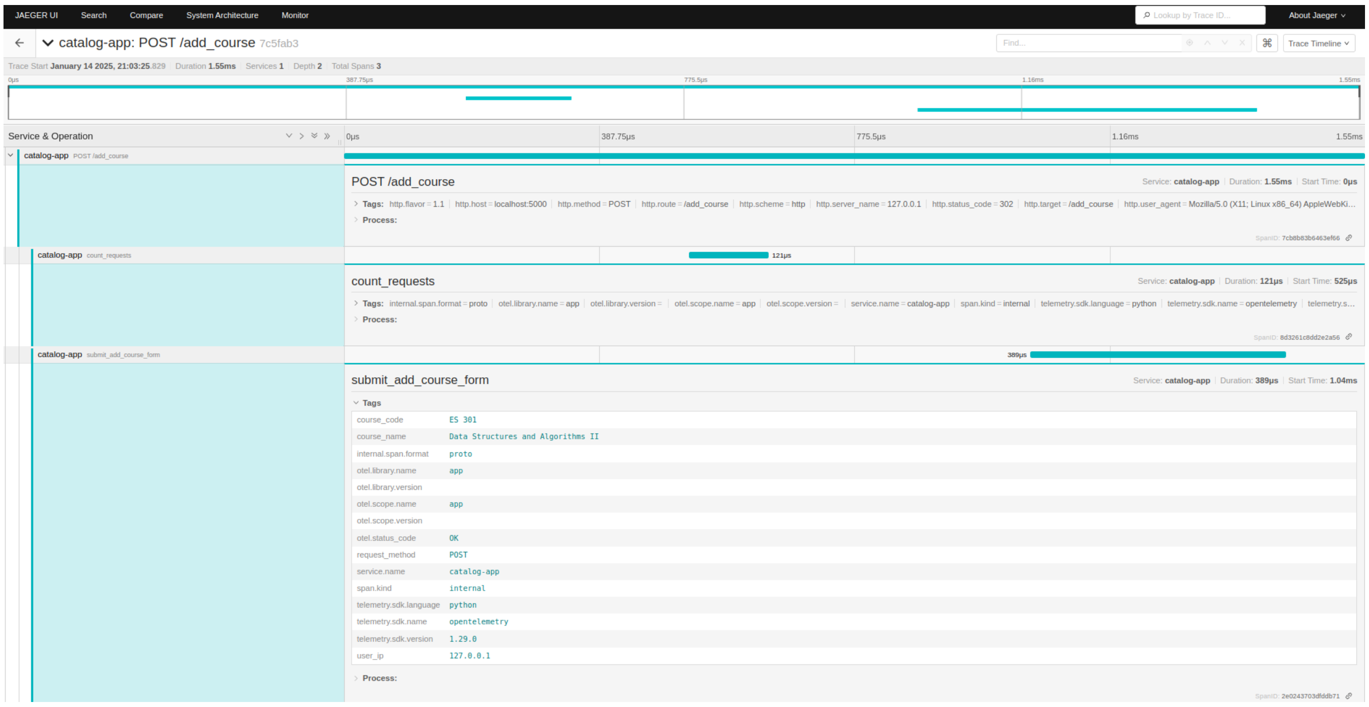
Tags: client_error_count = 1 | error = true | internal.span.format = proto | otel.library.name = app | otel.library.version = | otel.scope.name = app | otel.scope.version = | otel.status_code = ERROR | otel.status_description = New client side error | server_error_count =...

Process:

SpanID: 8de10a740c458427

The trace data contains exactly which fields are missing and also has an explicit error type.

Successful Course Addition:



catalog-app: POST /add_course 7c5fab3

Trace Start **January 14 2025, 21:03:25**.829 | Duration **1.55ms** | Services **1** | Depth **2** | Total Spans **3**

Service & Operation | 0μs | 387.75μs | 775.5μs | 1.16ms | 1.55ms

- catalog-app POST /add_course
  - catalog-app count_requests 121μs
  - catalog-app submit_add_course_form 389μs

POST /add_course

Service: **catalog-app** | Duration: **1.55ms** | Start Time: **0μs**

Tags: http.flavor = 1.1 | http.host = localhost:5000 | http.method = POST | http.route = /add_course | http.scheme = http | http.server_name = 127.0.0.1 | http.status_code = 302 | http.target = /add_course | http.user_agent = Mozilla/5.0 (X11; Linux x86_64) AppleWebKi...

Process:

SpanID: 7cb8b83b6463ef66

count_requests

Service: **catalog-app** | Duration: **121μs** | Start Time: **525μs**

Tags: internal.span.format = proto | otel.library.name = app | otel.library.version = | otel.scope.name = app | otel.scope.version = | service.name = catalog-app | span.kind = internal | telemetry.sdk.language = python | telemetry.sdk.name = opentelemetry | telemetry.s...

Process:

SpanID: 8d3261c8dd2e2a56

submit_add_course_form

Service: **catalog-app** | Duration: **389μs** | Start Time: **1.04ms**

Tags

course_code	`ES 301`
course_name	`Data Structures and Algorithms II`
internal.span.format	`proto`
otel.library.name	`app`
otel.library.version	
otel.scope.name	`app`
otel.scope.version	
otel.status_code	`OK`
request_method	`POST`
service.name	`catalog-app`
span.kind	`internal`
telemetry.sdk.language	`python`
telemetry.sdk.name	`opentelemetry`
telemetry.sdk.version	`1.29.0`
user_ip	`127.0.0.1`

Process:

SpanID: 2e0243703dfddb71

As visible from the above image, we are able to get metadata about the form submission, such as the course code, course name, etc. as well.

---

4. JSON Logging

- Structured logs for all events:
  - Example:

```
{
  "asctime": "2025-01-14 19:21:07,417",
  "name": "AddCourseLogger",
  "levelname": "ERROR",
  "message": "Failed to add course. Missing fields: semester,
schedule, classroom, prerequisites, grading"
}
```

Setup and Execution

1. **Clone the Repository**:

```
git clone https://github.com/nishchaybhutoria/CS203_Lab_01-main.git
cd CS203_Lab_01-main
```

2. **Set Up the Environment**:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r requirements.txt
```

3. **Run Jaeger**:

```
sudo docker run -d --name jaeger \
    -e COLLECTOR_ZIPKIN_HTTP_PORT=9411 \
    -p 5775:5775/udp \
    -p 6831:6831/udp \
    -p 6832:6832/udp \
    -p 5778:5778 \
    -p 16686:16686 \
    -p 14268:14268 \
    -p 14250:14250 \
    -p 9411:9411 \
    jaegertracing/all-in-one:1.41 --log-level=debug
```

4. **Run the Flask Application**:

   ```
   flask run
   ```

5. **Access the Application**:

   - Flask App: http://localhost:5000
   - Jaeger UI: http://localhost:16686