Overview

Student Engagement Service (student-engage-svc) is Fast API – microservice that consolidates student engagement data from two different sources:

1. Canvas LMS API
2. Student Information System API

The system normalizes and merges data from these APIs, stores it in a PostgreSQL database ( Mock db is used for demo purposes), it exposes two endpoints for data ingestion and summary reporting.

Canvas LMS Mock API

* Method GET
* Description: Returns a list of student engagement logs from Canvas LMS.

Sample Response:

[

{

"student\_id": "S001",

"name": "Alice Johnson",

"email": "alice.johnson@example.com",

"course\_id": "CSE101",

"course\_name": "Intro to Computer Science",

"last\_activity": "2025-07-09T10:00:00Z",

"engagement\_time": 90

}

]

Student Info System Mock API

Method: GET

Description: Returns a structured set of student details and engagement from the internal student system

Sample Response:

[

{

"id": "S001",

"full\_name": "Alice Johnson",

"email\_address": "alice.johnson@example.com",

"course": {

"id": "CSE101",

"name": "Intro to Computer Science"

},

"activity\_log": {

"last\_seen": "2025-07-09T10:05:00Z",

"engagement": 100

}

}

]

**System Architecture Diagram**

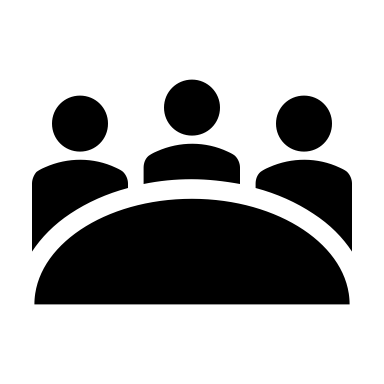
Canvas-lms -api

provides course information

Student-engage-svc

Fetches raw, Maps, Inserts, summarizes

advisor



Postgres -student db

Student-info-svc

provides student infotmation

**Simple Approach**

**Step-by-Step Flow:**

1. **Fetch**:  
   Two APIs are called from http\_connector.py
2. **Map**:  
   Raw data from each API is normalized to a common format using data\_mapper.py.
3. **Validate**:  
   Each record is validated (required fields, valid data) // As test data skipped validation
4. **Insert/Update**:  
   Records are inserted or updated in the database via repository.py.
5. **Summarize**:  
   The /summary endpoint aggregates and returns a report of engagement per course.

**Technologies Used**

| **Tool** | **Purpose** |
| --- | --- |
| **FastAPI** | REST API framework |
| **SQLAlchemy** | ORM for DB interaction |
| **PostgreSQL** | Primary database (or SQLite) |
| **Uvicorn** | ASGI server for FastAPI |
| **python-dotenv** | Securely load DB credentials |
| **requests** | Call external APIs |