University of Exeter

COURSEWORK SPECIFICATION
ECM3408 — Enterprise Computing
Module Leader: David Wakeling
Academic Year: 2025/26

Title: **Coursework Assignment**

Submission deadline: **17th February, 2026, Midday**

This assessment contributes **40%** of the total module mark and assesses the following **Intended Learning Outcomes**:

- Demonstrate recognition of the problems that can arise in the development of large-scale distributed information systems;
- construct concurrent and distributed computing systems using an enterprise-level model (e.g. EJB, .Net);
- understand and use protocol specifications;
- design and implement heterogeneous systems.

This is *an individual* assessment and you are reminded of the University's Regulations on Collaboration and Plagiarism. You must avoid plagiarism, collusion or any other academic misconduct. Further details about Academic Honesty and Plagiarism can be found at `https://ele.exeter.ac.uk/course/view.php?id=1957`.

Background

The widespread deployment of Large Language Models (LLMs), and their widespread abuse, has necessitated the development of safety measures called *guardrails* to control their inputs and outputs. A guardrail may prevent the generation of harmful content, the leaking of sensitive information or the production of biased responses. A simple system has:

- *input guardrails* that may modify the inputs of an LLM;
- *output guardrails* that may modify the outputs of an LLM.

This assessment is to implement an LLM-with-guardrails MVP as a system using three microservices.

Part I: LLM (20%)

Show an LLM microservice provides an LLM and listens on port 3000.

The LLM microservice takes a prompt string and returns an output string. See Figure 1. Here, the `POST` method takes a JSON object that has a `prompt` property whose value is a

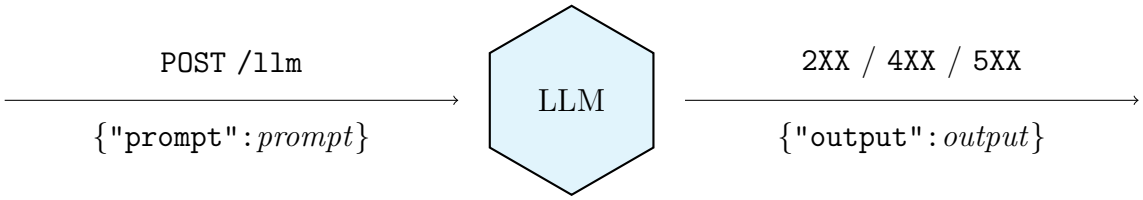

Figure 1: The LLM microservice.

string representing a prompt. The resulting JSON object has an `output` property whose value is a string representing the response of the LLM to the prompt.

This LLM microservice must be implemented using Mistral. Please use `MISTRAL_API_KEY` for your API key.

This part will be assessed by ten unit tests, and the score obtained will be based on the number of tests that are passed (zero or two marks for each of ten unit tests). A small number of unit tests will be made available for development.

Part II: Guardrails (60%)

Show a Guardrails microservice that stores guardrails and listens on port 3001.

Creating Guardrails

The Guardrails microservice allows guardrails to be created. See Figure 2. Here, the `PUT`

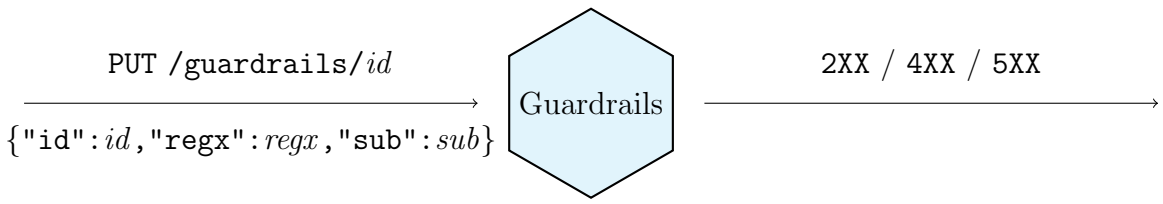

Figure 2: Creating guardrails.

method takes a JSON object that has a `id` property, whose value is a guardrail identifier, a `regx` property, whose value is a guardrail (Python) regular expression, and a `sub` property whose value is a guardrail substitution string.

Reading Guardrails

The Guardrails microservice allows guardrails to be read. See Figure 3. Here, the `GET`

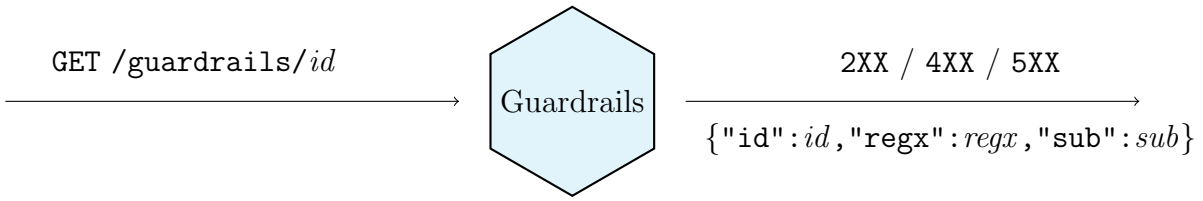

Figure 3: Reading guardrails.

method returns a JSON object that has an `id` property whose value is a guardrail identifier, a `regx` property whose value is a guardrail regular expression, and a `sub` property whose value is a guardrail substitution.

Deleting Guardrails

The Guardrails microservice allows guardrails to be deleted. See Figure 4.

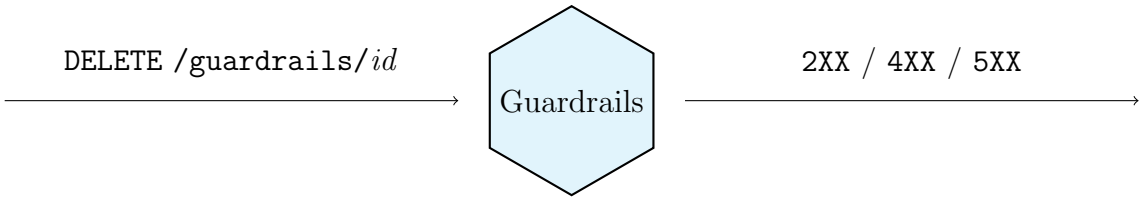

Figure 4: Deleting guardrails.

Listing Guardrails

The Guardrails microservice allows guardrails to be listed. See Figure 5. Here, the get method returns a JSON object that is a list of guardrail identifiers.

This Guardrails microservice must be implemented using a Google Firebase Realtime database. There are three possible locations for one of these databases: `United States`

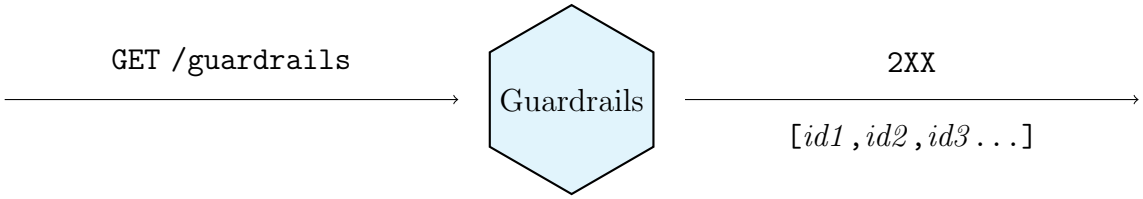

Figure 5: Listing guardrails.

`(us-central1)`, `Belgium (europe-west1)` and `Singapore (asia-southeast1)`. Please choose `Belgium (europe-west1)`. The database can run in *locked mode* or *test mode*. Please choose *test mode*. Your database name will look like `guardrails-3d1c1`, which gets incorporated into a URL like

```
https://guardrails-3d1c1-default-rtdb.europe-west1.firebasedatabase.app
```

Please use `FIREBASE_DB` for the name of your database.

This part will be assessed by 30 unit tests, and the score obtained will be based on the number of tests that are passed (zero or two marks for each of ten unit tests). A small number of unit tests will be made available for development.

Part III: Auberge (20%)

Show an Auberge microservice that provides a guarded LLM and listens on port 3002.

The Auberge microservice uses the guardrails provided by the Guardrails microservice to sanitise *both* the input and output of the LLM provided by the LLM microservice. The sanitisation is done by replacing any string matching the regular expression with the substitution string. See Figure 6. Here, the `POST` method takes a JSON object that has

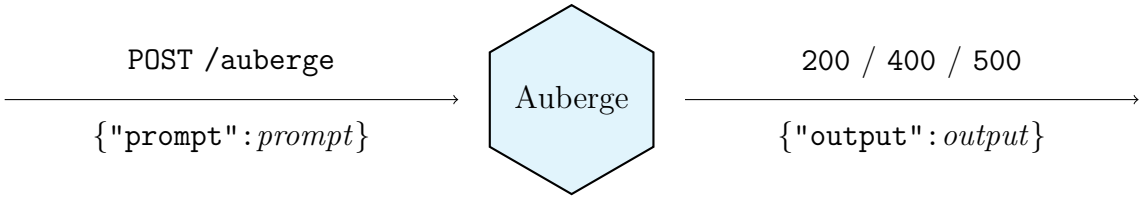

Figure 6: The Auberge microservice.

a `prompt` property whose value is a string representing a prompt. The resulting JSON object has an `output` property whose value is a string representing the sanitised response of the LLM to the sanitised prompt.

This part will be assessed by ten unit tests, and the score obtained will be based on the number of tests that are passed (zero or two marks for each of ten unit tests). A small number of unit tests will be made available for development.

AI-assisted

This module is *AI-assisted*. That is, it is one

> *where you may use GenAI tools ethically and responsibly to assist in the development of an assessment. In AI-assisted assessments, you must include prompts and, where possible, hyperlinks to their output as part of your list of references if you use GenAI tools. The assessment brief will include a checklist of the uses of AI allowed. If anything is not clear, please ask the lecturer running your module for clarification.*

Please use comments to indicate any code that is entirely produced using GenAI tools.

Submission

You should submit a single ".zip" file [1] containing the Python code of your three separate microservices, `auberge.py`, `guardrails.py` and `llm.py`. Other Python source code files may be supplied as needed. Your Python source code must run on the Anaconda implementation in the University's Lovelace laboratory. This implementation must be used without modification; that is, without the installation of modules using `pip`. Should any question arise as to whether code runs or not, it will be answered on the unmodified Anaconda implementation in the Lovelace laboratory. After uploading your ".zip" file, you should immediately download it, to check that is contains the files you expect.

[1] Other archive formats, including (but not limited to) `tar`, `xz`, `7zip` and `WinRAR` are all unacceptable.