5CCSACCA Cloud Computing for Artificial Intelligence

Coursework Overview

The coursework accounts for 50% of the final assessment and is an individual project divided into two equally important components: the code and the report. The primary objective is to evaluate your development and design decisions for a specific machine learning problem you aim to solve.

Phase 1 – Description of the Problem

You are to transform a specific AI system into a SaaS application. You may choose any AI system, provided it can perform inference on 4 CPUs with a maximum of 16GB of RAM [1]. You need to identify tools that facilitate the development and scalability of the system. Specific procedures must be followed: GitFlow for development, Docker (and optionally Kubernetes) for deployment, and MLFlow for versioning models[2]. The SaaS should be a RESTful API using FastAPI, with comprehensive documentation for each service component. Begin by outlining the AI system you plan to use and detailing the entire development process.

Phase 2 – The implementation

You are required to use the GitHub Classroom repository provided via the link on KEATS for your implementation. This repository should include:

- All your code with at least Dockerfile for deplomen t.
- A README file with specific instructions for deploying the SaaS, including examples of expected input and output.
- A requirements.txt file listing the Python dependencies.
- Any MLProject files you implemented during the process.

Ensure the README instructions allow for setting up the SaaS on a fresh Linux machine to explore its capabilities, at least within the provided example. The final version of the code must be in the main branch, and the development process should adhere to the GitFlow approach.

Deployment must be performed in one command, if the architecture is complex, provide a script for deployment.

Phase 3 – The Report

You will submit a PDF report (maximum 3000 words) detailing your design. While you may include select screenshots and source code excerpts, be judicious about what you include, referencing the Git repository for extensive details. The report should include a title page with

[1] You can verify the specification by employing a virtual machine with the specific specifications.
[2] If you are employing a model, considering employing the versioning system for validating different parametrizations.

the project name, your name, and student ID. It must contain a detailed discussion covering the following points:

1) System Overview and design decisions: Begin by explaining the system and detailing the specific inputs and outputs. Describe how you envision the final SaaS application, including any necessary adaptations. If a database is used, explain its role in the system and specify the type of database.
2) Architecture: Describe the technologies and libraries used and how they fit into the system architecture. If technologies such as Kubernetes are used, explain the cluster setup and how the components communicate.
3) The Model: Detail the model, including its expected inputs and outputs, and describe its training process. If you are training the model, explain the training data selection. If you are fine-tuning the model, or selecting parameters, describe the process and how MLFlow is used to version and compare models during parameter tuning.
4) Development Process: Explain the development process, including the creation and purpose of different branches.
5) Continuous Integration/Continuous Deployment (CI/CD): Describe how you aim to automate the integration and deployment process.
6) Costs: Calculate the costs associated with system resources, considering scalability, and perform specific cost calculations.
7) Testing, Security, and Monitoring: Describe the testing procedures and how the system will be monitored. Outline any security measures implemented and their importance.
8) Limitations: Discuss the potential limitations of the system.
9) Sustainability: Address potential sustainability issues related to your system.
10) References: Provide references in Harvard style for any sources related to your decisions or discussions.

What and how to submit

- **Overview Plan Submission**: Submit an overview plan by the end of Week 5. This submission is ungraded and will provide formative feedback to support the development of the following phases. It should cover phase 1. The maximum length of the plan is 1 page.
- **Final Submission**: Submit a GitHub URL containing your implementation and the report by the coursework deadline.

The mark scheme

The table below provides guidance on the mark scheme for the different components:

Component	0-40	40-50	50-60	60-70	70+
GitHub (Implementation and Report)	No use of GitHub or GitFlow	Use of GitHub but incorrect use of GitFlow	Use of GitHub but poor use of GitFlow	Use of GitHub and reasonable use of GitFlow	Use of GitHub and complete use of GitFlow
Code (Implementation)	The implementation does not work or works poorly	The implementation works and performs the specific task but some parts do not work as expected	The implementation works with minor bugs and is properly documented	The implementation has no bugs and is properly documented	The implementation has no bugs, is properly documented, and has been clearly optimized
Containerization (Implementation and Report)	No containerization or trivial containerization for running the model	Simple containerization for the model	Reasonable containerization that fulfills the SaaS purpose	Reasonable containerization that fulfills the SaaS purpose and is optimized for microservices	Reasonable containerization that fulfills the SaaS purpose, is optimized for microservices, and scales using Kubernetes
SaaS (Implementation and Report)	No SaaS creation and no working RESTful APIs	Basic attempt to create a SaaS with a simple RESTful API connecting to a simple database	Reasonable RESTful API that employs the model and databases and handles users	Good RESTful API that employs the model and databases, handles user authentication, and employs microservices	Excellent RESTful API that employs the model and databases, handles user authentication, employs microservices through Ingress
Model (Implementation and Report)	Too simple model (basic regression or classification), basic training, no parameter effort	Reasonable model or reasonable complexity in training	Reasonable model handling complex data types	Reasonable model handling complex data types with basic versioning and parameterization	Advanced model solving an interesting problem, trained or fine-tuned for a specific case with deep versioning and parameterization

Testing and Monitoring (Implementation and Report)	No or very basic testing and monitoring	Basic strategies for testing and monitoring	Reasonable set of tests covering most functionalities with some monitoring strategies	Reasonable set of tests covering most functionalities with good monitoring strategies	Complete integration with comprehensive tests and outstanding monitoring strategies
Databases (Implementation and Report)	No use of databases or basic use	Reasonable use of CRUD strategies in a database	Good use of databases with different tables or collections	Good use of databases with different tables or collections and user authentication	Excellent use of databases with different tables or collections and protected user authentication
Design Decisions and Architecture Description (Report)	Missing, unclear, or extremely poor description	Reasonable description of design decisions and architecture	Mature description of design decisions with proper discussion and clear architecture description	Mature description of design decisions with proper discussion, comparison with alternatives, and mature architecture description	Outstanding description of design decisions with proper discussion, comparison with alternatives, and outstanding architecture description
Costs (Report)	No or poor calculation of costs	Reasonable calculation of costs	Precise calculation of costs for different numbers of users	Precise calculation of costs for different numbers of users based on resource consumption during testing	Precise calculation of costs for different numbers of users with specific formulas for scalability based on resource consumption during testing and for a complex architecture
CI/CD and Security (Implementation and Report)	No or poor consideration of CI/CD or security	Very basic considerations of CI/CD and security	Reasonable considerations and implementations of CI/CD and security	Good CI/CD and security implementations with specific problems handled	Complete CI/CD implemented and explained with strong security solutions

Limitations and Sustainability (Report)	No or poor description of limitations and sustainability	Basic description of limitations and sustainability	Reasonable description of limitations and sustainability	Good description of limitations and sustainability with solutions to some issues	Outstanding description of limitations and sustainability with solutions to main issues
References (Report)	None, wrongly formatted, or poor references	Basic set of references without many academic sources, correctly formatted	Reasonable set of references with good academic sources, correctly formatted	Good number of references including academic sources and books, correctly formatted	Outstanding selection of references with proper discussion about the state of the art relevant to the problem