AI 381 - Final Project Report: MLOps System for Sentiment Analysis

Team Members:

- Farah Alqudah (164312)
- Mahaba Alkhawaldeh (161271)

1. Introduction

This report presents a comprehensive MLOps (Machine Learning Operations) system designed for real-time sentiment analysis. The project fulfills the requirements of the AI 381 final project by demonstrating a complete software development lifecycle for an AI/ML application, from initial design and implementation to deployment and automation. The system is built upon a modern, cloud-native architecture, incorporating industry-standard practices in DevOps, security, and software engineering.

1.1. Problem Statement

In today's data-driven world, organizations are inundated with vast amounts of unstructured text data from various sources such as social media, customer reviews, and support tickets. The ability to automatically analyze this data to gauge public opinion, identify customer sentiment, and detect emerging trends is crucial for making informed business decisions. Manual analysis is impractical at scale, creating a need for automated, reliable, and scalable solutions for sentiment analysis.

This project addresses this need by developing a production-ready MLOps system that provides real-time sentiment classification as a service. The system is designed to be scalable, secure, and maintainable, enabling organizations to integrate sentiment analysis capabilities into their existing workflows and applications with ease.

1.2. Project Goals and Objectives

The primary goal of this project is to design, implement, and deploy a robust MLOps system for sentiment analysis. The key objectives are as follows:

- **Develop a Minimum Viable Product (MVP):** Create a functional sentiment analysis service that can classify text into positive, negative, or neutral categories.
- **Implement a RESTful Web API:** Build a secure and well-documented API for easy integration with other applications.
- **Integrate Identity Management:** Implement a secure authentication and authorization system with role-based access control (RBAC).
- **Containerize the Application:** Package the application and its dependencies into a Docker container for portability and reproducibility.
- **Automate CI/CD:** Create a continuous integration and continuous delivery (CI/CD) pipeline to automate testing, building, and deployment.
- **Ensure High Quality:** Implement comprehensive testing and documentation to ensure the system is reliable and maintainable.
- **Demonstrate Cloud-Native Principles:** Utilize cloud-native technologies and design patterns to build a scalable and resilient system.

2. System Architecture and Design

The system is designed with a layered, cloud-native architecture that promotes separation of concerns, scalability, and maintainability. The architecture consists of several key layers, each with distinct responsibilities.

2.1. Architectural Overview

The following diagram illustrates the high-level architecture of the system:

```
┌─────────────────────────────────────────────────────────────────┐
│                        Client Applications                       │
│                    (Web, Mobile, CLI Tools)                      │
└────────────────────────────┬────────────────────────────────────┘
                             │
                    ┌────────▼────────┐
                    │   Load Balancer │
                    │   (Optional)    │
                    └────────┬────────┘
                             │
        ┌────────────────────┼────────────────────┐
        │                    │                    │
   ┌────▼────┐         ┌────▼────┐         ┌────▼────┐
   │ Instance │         │ Instance │         │ Instance │
   │    1     │         │    2     │         │    N     │
   └────┬────┘         └────┬────┘         └────┬────┘
        │                    │                    │
        └────────────────────┼────────────────────┘
                             │
        ┌────────────────────▼────────────────────┐
        │      API Gateway / Reverse Proxy        │
        │         (Nginx / Azure Gateway)         │
        └────────────────────┬────────────────────┘
                             │
        ┌────────────────────▼────────────────────┐
        │     Flask Web Application (REST API)    │
        └────────────────────┬────────────────────┘
                             │
        ┌────────────────────┼────────────────────┐
        │                    │                    │
   ┌────▼────┐         ┌────▼────┐         ┌────▼────┐
   │  Cache  │         │ Database │         │  Logs   │
   │ (Redis) │         │ (Optional)│        │ (ELK)   │
   └─────────┘         └──────────┘         └─────────┘
```

2.2. Technology Stack

The project utilizes a modern and robust technology stack, chosen for its performance, scalability, and developer productivity.

Component	Technology
Backend Framework	Flask
ML Model	scikit-learn
Authentication	JWT + Azure AD
Containerization	Docker
CI/CD	GitHub Actions
Testing	pytest

3. Core Components

This section details the implementation of the core components of the MLOps system, aligned with the project's evaluation categories.

3.1. Problem Definition & Requirements (10%)

The problem of automated sentiment analysis was clearly defined, and a set of functional and non-functional requirements was established. The MVP scope was defined to include core sentiment analysis functionality, a secure API, and a demonstration of the complete MLOps lifecycle.

3.2. Web API + Identity Management (10%)

A secure RESTful Web API was implemented using Flask. The API provides endpoints for sentiment analysis, user authentication, and system administration. Identity management is handled through a robust authentication service that supports JWT-based authentication and role-based access control (RBAC). The system defines three roles: `admin`, `student`, and `viewer`, each with a specific set of permissions.

3.3. Cloud-Native Function + Containerization (20%)

The application is containerized using Docker, which encapsulates the application and its dependencies into a portable and reproducible image. A multi-stage Dockerfile is used to create an optimized and secure production image. The system is designed to be deployed as a cloud-native application, with support for deployment to Azure Container Instances, Azure App Service, and Kubernetes.

3.4. CI/CD + GitHub Repository (15%)

A comprehensive CI/CD pipeline was created using GitHub Actions. The pipeline automates the following stages:

- **Testing:** Automatically runs unit and integration tests on every push and pull request.
- **Linting:** Enforces code quality and style standards.
- **Building:** Builds the Docker image and pushes it to a container registry.
- **Security Scanning:** Scans the application and its dependencies for vulnerabilities.

The GitHub repository is well-organized, with a clear directory structure, comprehensive documentation, and a detailed commit history.

3.5. Testing & Documentation (15%)

The project includes a comprehensive suite of tests, including:

- **Unit Tests:** For the sentiment analysis model and other core components.
- **Integration Tests:** For the API endpoints.

Code coverage is measured to ensure a high level of test quality. The project is also extensively documented, with a detailed `README.md` file, API documentation, a deployment guide, and an architecture overview.

3.6. Project Management (10%)

Project management was conducted using GitHub Projects, which provided a Kanban board for tracking tasks, milestones, and progress. The project was divided into sprints, with clear deliverables for each sprint.

3.7. Presentation & Innovation (10%)

The project demonstrates innovation through its adoption of modern MLOps practices and its focus on building a production-ready system. The live demonstration will showcase the end-to-end functionality of the system, from making an API request to viewing the automated CI/CD pipeline in action.

4. Conclusion

This project successfully demonstrates the design, implementation, and deployment of a comprehensive MLOps system for sentiment analysis. By integrating a machine learning model with a robust web API, a secure authentication system, and a fully automated CI/CD pipeline, the project provides a blueprint for building and operating production-ready AI/ML applications. The system is scalable, maintainable, and secure, making it a valuable asset for any organization looking to leverage the power of sentiment analysis.