📦 MLOps Course Final Project - Assessment Criteria

Course Period: January 5, 2026 - March 15, 2026

Final Deadline: March 15, 2026, 24:00 (UTC+1)

Submission Format: GitHub Repository

Group Size: 4 students per group

🎯 Project Overview

The final project aims to assess students' ability to **design, implement, and manage an end-to-end MLOps pipeline** using the tools and practices introduced in the course.

Each group will:

- Independently define a **machine learning task**, dataset, and evaluation strategy
- Implement the project following **MLOps best practices**
- Track progress and collaboration through **GitHub**
- Deliver a **final project report** inside the repository

🧱 Core Technologies Covered

Your project should meaningfully incorporate most of the following:

- **UV** (Python environment & dependency management)
- **Git & GitHub** (collaboration, commits, PRs)
- **Pre-commit & Unit Testing**
- **FastAPI** (model inference service)
- **Docker**

- **MLflow** (experiment tracking & model registry)
- **Monitoring** (basic metrics, logging, health checks)
- **CI/CD**

🗓️ Project Timeline & Checkpoints

The project is divided into **4 major checkpoints** in Feburary and March.

✅ Checkpoint 1 — Project Setup & Foundations

📅 **Due: February 1, 2026, 24:00**

Required Deliverables

- GitHub repository created and shared
- All team members have **write access**
- Python environment managed with **UV**
- Initial project structure (clear, modular, readable)
- Basic data loading and preprocessing logic
- A runnable training script (even a simple baseline model)
- Dependencies correctly tracked (`pyproject.toml`, `uv.lock`)
- Clear README with:
  - Project description
  - Task definition
  - Dataset source
  - Team member roles

Assessed Skills

- Git fundamentals
- Environment reproducibility

- Code organisation
- Early project planning

✅ Checkpoint 2 — Code Quality & Experiment Tracking

📅 **Due: February 15, 2026, 24:00**

Required Deliverables

- Pre-commit hooks configured
- Unit tests cover 60%
- Tests runnable locally
- MLflow integrated with log parameters, metrics, model artifacts
- Clear experiment naming and comparison
- Meaningful Git commit history
- README.md updated

Assessed Skills

- Software engineering discipline
- Testing mindset
- Reproducible ML experiments
- Experiment tracking with MLflow

✅ Checkpoint 3 — Model Serving & Containerisation

📅 **Due: March 1, 2026, 24:00**

Required Deliverables

- FastAPI application for model inference
- Clear API schema (request & response)

- Dockerfile for training and inference
- Application runnable via Docker
- Basic API test
- Inference model loaded from MLflow or saved artifacts
- README.md updated

Assessed Skills

- Model deployment concepts
- API-based ML services
- Containerisation
- Practical usability of the system

✅ Checkpoint 4 — Monitoring, Polish & Final Report

📆 **Due: March 15, 2026, 24:00**

Required Deliverables

- Basic monitoring strategy
- Discussion of potential future work
- Final project report (see below)
- Clean, well-documented GitHub repository

📝 Final Project Report in form of README.md

Your report should include:

1. **Project Overview**
2. **Problem Definition & Data**
3. **System Architecture**
4. **MLOps Practices**

5. **Monitoring & Reliability**
6. **Team Collaboration**
7. **Limitations & Future Work**

> 🕛 Only commits made before March 15, 24:00 will be graded.

🎥 Build, Deployment & App Demo (Video)

Each group should submit a **demo video (≤ 10 minutes)** showing:

- **Automatic build & deployment triggered by GitHub**
  - CI pipeline, Docker build, app startup
- **How to use the application**
  - FastAPI endpoint(s)
  - Example request and response

Submission:

- Provide a **public or unlisted video link**
- Include the link in `reports/README.md`

📊 Grading Rubric (100 Points)

Category	Points
Project Setup & Git Usage	15
Code Quality & Testing	15
MLflow & Experiment Tracking	15
Model Serving (FastAPI + Docker)	15
Monitoring & Reliability	10
Final Report Quality	15
Bonus	10
Total	**100**

👥 Teamwork & Individual Contribution

- Git commit history will be reviewed
- Each Pull Request should be reviewed and validated by **at least** one group member.
- All team members should understand **all parts** of the project

> 🚨 Unequal contribution may lead to individual score adjustments.

💡 Evaluation Philosophy

This project emphasises:

- **Engineering thinking over model performance**
- **Reproducibility over novelty**
- **Clarity over complexity**

A simple model with excellent MLOps practices can score higher than a complex model with poor engineering.

🚀 Good Luck!

If you have questions during the process, **ask us early and often** 😊

yue.li@ext.devinci.fr

linghao.kong@ext.devinci.fr

🗄️ Good Example

https://github.com/nielstiben/MLOPS-Project.git