# Machine Learning in Production Final Project

As a part of the curriculum of the Master 2 (M2) course entitled “Machine Learning in Production”, the students will complete a team project work. Each team is composed of 3 members, and the members will take care of dividing the tasks equally between them. The purpose of this project is to combine all the skills collected throughout the entire course, and to provide a solid example of real-life ML-powered application development in a DevOps environment.

The following sections provide necessary information about the description of the application to be created.

For any further detail, please contact the instructor: **Khodor Hammoud**

Project Summary

You are a part of a DevOps team who has been tasked with creating a web-based full stack application that serves a machine learning model that can predict whether an anime series is enjoyable or not. Thus, you must, with your teammates, apply DevOps principals to develop the application, and train and deploy the machine learning model that handles the predictions.

## 1. Requirements

* The application is a web interface.
* In the application, the user can input:
  + Anime Title
  + Anime Genre(s)
  + Anime Description
  + Anime Type
  + Anime Producer
  + Anime Studio

And based on these information, the application should inform whether it’s likely or not that the user would enjoy such anime; a ratings prediction.

* Every functional part of the application must be tested for proper functionality.
* The application must be able to handle at least 10 requests per minute.
* The application must be easily deployable (Dockerised).
* The application must be properly monitored after deployment, we want to be able to quickly find any issue that might cause performance problems or down time, and know how much resources is the application consuming.

2. Technical Description

**2.1 The ML Model**

Given the provided dataset, the task of the ML model to be developed is basically to predict the **“Rating”** column. This in itself is not a challenge, but most of the data that you have to process is textual. If you have not built a model that can handle text before, sing a model like Word2Vec (<https://www.tensorflow.org/tutorials/text/word2vec>) can be very helpful. It can convert a word to a vector of values, and then you can average out the values of every word in a paragraph to create a vector representation of an paragraph.

Keep in mind that not all the data is to be processed as textual, some are categorical.

**2.2 The Web Interface**

The students are free to choose whichever technology they know/like to create the web insterface. The end result should be a running application which the end user can access through a web browser, and start using immediately.

**2.3 The Application Backend**

The students are free to use whichever technology they know/like to create the application backend. The application backend covers the backend web server, as well as the machine learning model.

Instead of having the ML model packaged with the docker image, the model should be loaded from a separate volume.

Note that every entiry (Frontend, Backend) should be containerized separately.

**2.3 The Application Package**

The final format of the application ready for distribution should be a **set of Docker Images**, **posted to the DockerHub**, which administrators can simply run Containers from. The students should use **Docker Compose** to facilitate the communication between, and the deployment of, Docker containers. Students should provide a description file with their submitted application in which whey describe how to run their images (like providing on which port does the application run by default…), with sample commands on how to run the application.

3. Technical Requirements

The students are to use the following technologies and steps throughout their implementation:

**3.1 Task Management**

Each team is to use a project management tool of their choice (Trello, Asana, Jira…) to coordinate the tasks between the team members. Divide the user stories into tasks, list these tasks on the project manager, and track the progress of each task as it progresses. Each team is required to present their project management history during their presentation.

**3.2 Source Code Management**

Each team is required to create a github repository containing their project, and use it as their version control. **Each new task should have its own branch on the github repository**. At every task completion (from the project manager), the associated team member should merge their task’s branch to the dev branch. When a stable version is ready, you merge to the main branch. The github repository should contain all your files, including the docker file and any meta data files (like the python requirements.txt if it exists).

Students must use the CD version control branching scheme, where the version control repository will contain a main branch, a develop branch, a feature branch (for every added feaure) and a release branch (for every version release).

**3.3 Testing**

Each team should provide unit, end-to-end and stress tests to their final application.

* Unit tests are in the form of testing the functionality of each function of your program (when applicable).
* End-to-end testing would be testing the entire functionality of the system, from frontend to backend. Example: inserting the description of an Anime into the input form and clicking the submit button returns the predicted rating of that Anime.
* Stress testing will be writing a user simulation to prove that your application can handle 10 requests per minute.

**3.4 Automation**

The students are to use Jenkins for automating the building, testing, deployment and release (if applicable) of the application. At the end, each team is expected to have a Jenkins pipeline constructed which connects to the different github branches, and applies appropriate respective actions:

* build and run unit tests on feature branches.
* stress test and push to release on the develop branch
* wait for user acceptance on the release branch before pushing to main

# example code

pipeline {

agent any

stages {  
 stage("user confirms") {

input {

message "insert your question for the client here"

}

}  
 }

}

* on merging with main, push to Dockerhub.

**3.5 Monitoring**

The students are to use Prometheus to monitor:

* Hardware metrics: like CPU usage, memory usage, and disk space usage.
* Software metrics: integrate different software metrics inside your application to monitor information like response time, user requests count, exceptions,

Integrate Counters, Gauges, Histograms and Summaries as you see fit.

Add rules and alerts where you see fit, here are some examples:

* Alert before running out of memory
* Alert when cpu usage is very high
* Alert when your code raises an exception
* Alert when your system is down for more than a specific period of time
* …

Use Grafana as the monitoring dashboard.