1. **Image Classification / Object Detection Task.**
    - **Task** : Create a **Car Retrieval System** that includes Object Detection Model and **Car Classification/Retrieval Model**.
    - **Objective** : The Car Retrieval System should be able to detect multiple car instances then classify/retrieve what is the type (eg. MPV, Sedan, Hatchback, etc) of the detected car.
    - **Note :**
        - In the detection task it's optional to use which dataset, in classification you should use Indonesian Car as the baseline.
        - The classifier should be developed separately from the object detection model, you are not allowed to only use one model.
        - The classifier should be developed using the Tensorflow/PyTorch framework.
        - You should demo your Model using this video : https://intip.in/QNpw
    - **Instruction :**
        - You should Develop the **Object Detection Model** and **Car Classification/Retrieval** from **the scratch**.
        - You can use any base model you like. For example you decide to use SSD for object detection, VGG16 for image classification or ResNet50 for image backbone in Siamese Neural Network.
        - Pre-trained models for transfer learning are allowed such as ImageNet, etc.
        - It is encouraged to use Multiple libraries that support your development such as OpenCV, Scikit-Learn, and especially **Deep Learning Frameworks like Tensorflow/PyTorch (Mandatory)**.
        - Feel free to use a public dataset or gather your own personalized dataset to improve your model.
    - **What You need to Submit :**
        - Create a Report that Explain the entire process of your Project.
        - Make sure, the report is well-structured.
        - It's Mandatory to add a **Figure/Chart/Graph** on **how your Application runs**.
        - It's Mandatory to add a **Figure/Chart/Graph** about your **training & inference pipeline.**
        - It's Mandatory to add **figures or tables that evaluate the performance of your model.**
        - Make sure to **documentate the training log**, submit the **best model weight** along with **the training log.**
        - Inside the Report, you need to **explain each of the figures** you include and give a **conclusion why your approach is better.**
        - The **more detail you explain each step you gave**, the better points you could obtain.
        - Create a **GitHub that stores all of your code inside**, you can separate the repository depending on your needs. Include the Readme with all of

the supported Python versions and necessary library versions with the installation step.
    - Make sure you also include a list of the car/vehicle you include for your model training. The minimum different type of car you need to classify at least 8 different cars.
    - It's a plus point if you can describe and reason your way to tune your model from a not so good model into a robust model.
- **Assessment Criteria** :
    - Code Running Flawlessly (30%)
    - Deep Analytical Thinking and AI knowledge (30%)
    - Model Performance (20%)
    - Clean Code (10%)
    - Creativity (10%)

2. Large Language Model Case (Bonus and Encouraged).

- **Task :** Create **Retrieval Augment Generation (RAG)** that integrates with LLM included with **Previous Conversations**. You may also consider Fine-Tune the LLM if possible.
- **Objective :** The LLM response should be improved after giving RAG information and being able to memorize previous conversations.
- **Topic :** Cybersecurity, Pentesting, and Hacking.
- **What You need to Submit (Instruction)** :
    - Create a GitHub repo that includes all of the code that includes the pipeline of Preprocessing the RAG, RAG, previous conversation, and Inference.
    - The code should be written in a way using multiple small separated modules, so you need to customize a function or class and not include it into one script/notebook (clean-code).
    - In readme, you need to provide supported dependency, version, and installation steps.
    - Explain how your pipeline works and why you choose the LLM to run the RAG for cybersecurity topics and why you choose a certain text, document or dataset for your RAG.
    - You may use any data you can find on the Internet and convert it into your own RAG.
- **Assessment Criteria** :
    - Code Running Flawlessly (30%)
    - Design of the Pipeline (30%)
    - Clean Code (20%)
    - Creativity on Pipeline or Services (20%)

3. **Evaluation Large Language Model (Bonus).**
    - **Task :** Evaluate the RAG Pipeline you just created from the previous case.
    - **Objective :** Compare the response from your pipeline with the given dataset. You must obtain Quantitative and Qualitative results during the evaluation.
    - **Dataset URL :** **https://huggingface.co/datasets/preemware/pentesting-eval**
    - **What You need to Submit (Instruction)** :
        - You can sample 10-20% of the data from the given datasets and then use it as evaluation.
        - You can choose any evaluation technique to evaluate how robust your pipeline response is compared to the ground-truth given from the dataset.
        - The code that you use to evaluate must be uploaded to the GitHub repository from task number 2.
        - Create a short report that includes a sample of the response compared with the ground-truth response from the dataset.
        - Make sure to also include Qualitative results by either table or graph.

- **How to Submit**
    - Deliver all of the tasks into a PowerPoint, including all of the necessary details.
    - For Demo, please make a 5-10 minute demo and explain how your code works and the result.
    - Include your GitHub Repository that holds each task you already done inside powerpoint.
    - Submit all of the files and any supporting document into : **agung@islab.re.kr**