AI & SUSTAINABILITY
IN VET EDUCATION

AI-VET MOOC PROJECT SUBMISSION TEMPLATE

Funded by
the European Union

This project has been funded with support from the European Commission.
This communication reflects the views only of the author, and the
Commission cannot be held responsible for any use which may be made of
the information contained therein.

AI-VET MOOC PROJECT SUBMISSION TEMPLATE

This template provides a structured format for submitting final projects within the AI-VET MOOC. Each section should be completed thoroughly to ensure consistency and to provide all the necessary information for project evaluation, peer review, and sharing within the course community.
Participants are encouraged to use this template to present their innovative ideas, learning experiences, and applications of Artificial Intelligence in Vocational Education and Training (VET).
All sections written in grey contain instructions and examples to guide the preparation of your project report. These notes must be deleted before submission. The final document must be a technical project report, with a maximum length of twenty (20) pages, including text, figures, and references.

1. Project Title:

Smart Waste Sorting System using YOLOv11 and Raspberry Pi 5

2. Project Category:

Select the most appropriate(s) category(s) for your project from the following options:

- ■ AI for Sustainable Development
- ■ Computer Vision and Image Recognition
- ■ Intelligent Automation and Control Systems
- ■ Smart Environments and Monitoring

3. Project Description:

This project focuses on creating an AI-powered waste management solution to automate recycling sorting. Using a Raspberry Pi 5 and the YOLOv11 object detection model, the "EcoBin" identifies plastic water bottles in real-time. When a bottle is detected, a servo-motor (SG90) automatically opens the bin lid to accept the item. This project demonstrates how Computer Vision and Physical Computing can be integrated to promote circular economy practices and environmental sustainability in a school or office environment.

4. Keywords:

YOLOv11, Raspberry Pi 5, Object Detection, Smart Recycling, Sustainability

5. Target Audience:

Identify the primary audience for this project. Choose one or more of the following:

- □ Educators
- □ Students
- □ Both Educators and Students

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- ☐ VET Institutions

Example: "Both Educators and Students"

6. Project Objectives:

Clearly outline the specific objectives of the project (from 2 to 4). This section should detail the skills, knowledge, or outcomes participants will gain by engaging with this project.

Technical: Master the AI workflow: Model training/optimization (ONNX) and real-time edge inference.
Educational: Understand the integration of AI software with physical actuators (Servo Motors) via GPIO.
Sustainability: Promote responsible waste disposal through innovative technology, aligning with Sustainable Development Goals (SDGs).

7. Project Timeline:

Provide a brief overview of the project timeline: start and end dates

- Start Date: 18/01/2026
- End Date: 27/02/2026

8. Tools and Technologies Used:

Hardware: Raspberry Pi 5, Raspberry Pi Camera Module v2, Servo Motor SG90.
Software Stack: Python 3.11, Ultralytics YOLOv11, OpenCV (for image processing), gpiozero library.
AI Tools: Google Colab (for model export), ONNX Runtime (for optimized performance on ARM architecture).
Prototyping: Custom 3D-printed components (e.g., Camera Mount, Servo Brackets) designed for the bin's structure and printed using a Bambu Lab A1 3D Printer.

2- Programming Environments:

Python 3: The primary programming language used for developing the control logic, integrating AI models, and managing hardware communication.

Geany IDE: The main Integrated Development Environment used on the Raspberry Pi OS for writing, debugging, and executing the Python scripts.

TigerVNC / RealVNC: Remote desktop tools used to manage the Raspberry Pi 5 from a PC. This allowed for real-time monitoring of the camera feed and code adjustments without a dedicated monitor.

Software Libraries & Frameworks
Ultralytics (YOLOv11): Employed for real-time object detection, specifically trained to identify bottles and related containers with high precision.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

GPIOZero & LGPIO: Libraries used to interface with the Raspberry Pi 5 GPIO pins. The LGPIO driver was critical for compatibility with the Pi 5's new RP1 chip architecture to ensure stable Servo motor control.

OpenCV (cv2): Used for image processing and visual rendering, such as drawing bounding boxes and labels over the detected objects in the video feed.

Picamera2: The official Raspberry Pi camera library used to capture high-speed, high-resolution frames from the Camera Module 2.

3- AI and Machine Learning Tools (if applicable):

This project employed Transfer Learning using the YOLOv11 Nano (n) model. The model used was pre-trained on the COCO (Common Objects in Context) dataset, which already includes "bottles" (Class ID 39) among its 80 object categories.
The model was adapted for real-time edge computing on the Raspberry Pi 5.
To ensure smooth performance on the Raspberry Pi, the model was used in the ONNX (Open Neural Network Exchange) format. This format optimizes the model specifically for CPU inference, significantly increasing the frames per second (FPS) while maintaining high accuracy.
While the model provided the "vision," the project adapted its output through a custom software layer to filter and prioritize specific data:

Class Filtering: The script was programmed to specifically listen for IDs like 39 (bottle) (vase).
Confidence Thresholding: A threshold of 0.35 - 0.5 was implemented. This ensured the Servo would only activate when the AI was sufficiently certain, preventing "false positives" from background noise.
Inference to Action: The "Inference" (the AI's guess) was mapped directly to the hardware. When a target was detected, the software triggered a specific PWM signal to move the Servo to 90°, effectively bridging the gap between Machine Learning and physical automation.

4-IoT and Cloud Services:

The architecture of this project is based on Edge Computing, a distributed computing paradigm that brings computation and data storage closer to the sources of data. By performing the YOLOv11 AI inference directly on the Raspberry Pi 5, the system eliminates the latency associated with cloud-based processing, ensuring that the 6-second servo response is instantaneous upon bottle detection. Connectivity for development and monitoring was managed through the VNC (Virtual Network Computing) protocol. Using TigerVNC, a remote desktop link was established over the local network, allowing for real-time visualization of the OpenCV video feed and remote code deployment without the need for a physical monitor.

Data management within the system follows a structured pipeline where the Camera Module 3 acts as the data producer and the Python logic acts as the controller. While the current implementation operates as a standalone edge node, the software architecture is fully IoT-ready and compatible with the MQTT (Message Queuing Telemetry Transport) protocol. By integrating an MQTT Broker like Mosquitto, every detection event (e.g., "Bottle Detected" with timestamp and confidence score) can be published to

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

a central cloud server. This would enable real-time Data Visualization through platforms like Grafana or Node-RED, providing a comprehensive dashboard for monitoring recycling statistics. Furthermore, this connectivity allows the device to communicate with other IoT actuators in a smart factory environment, transforming a single detection node into a collaborative part of a larger, cloud-connected automation ecosystem.

5- Datasets:

The project utilized the COCO (Common Objects in Context) dataset, which is a large-scale object detection, segmentation, and captioning dataset. It contains over 330,000 images, with more than 200,000 of them being labeled, ensuring the AI model had seen millions of variations of a "bottle" (different shapes, sizes, lighting, and angles) before being deployed on the Raspberry Pi 5.
Because the project employed Transfer Learning via the YOLOv11 architecture, the data preparation was handled as follows:

Pre-labeled Data: The data was already meticulously labeled with bounding boxes and per-instance segmentation by the COCO consortium, eliminating the need for manual labeling within this project.

Preprocessing: During the inference phase on the Raspberry Pi, the live image data from the Camera Module 3 was automatically preprocessed (resized to 640x480 pixels and normalized) to match the input format of the training dataset.

Optimization: The dataset was used to train the "weights" of the YOLO model, which were then exported into the ONNX format. This allowed the system to perform "Inference" (recognition) on the AI-VET Kit using real-time sensor data from the camera, comparing it against the patterns learned from the COCO dataset.

6- Additional Tools (optional):

ONNX Runtime: This was a critical tool used to convert and run the YOLOv11 model. By using ONNX (Open Neural Network Exchange), the model was optimized for the Raspberry Pi 5's CPU architecture, significantly increasing the inference speed (Frames Per Second) compared to standard formats.

Google Colab (Optional/Preparation): Used for initial testing and script validation. Colab's cloud-based GPUs allow for rapid testing of AI models and libraries before deploying the final, lightweight version to the Raspberry Pi's hardware.

Linux Terminal (Bash): Extensive use of the command-line interface was required for system-level configurations, such as installing dependencies (pip, opencv, ultralytics), managing GPIO permissions, and troubleshooting the LGPIO driver layers.

GitHub / Version Control: Used for managing different iterations of the Python script. Version control was essential when transitioning from basic bottle detection to the more complex "servo-detach" logic required to stop motor jitter.

Use clear, concise descriptions so that someone else could understand how your AI-VET Kit project was built and managed digitally.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

The digital backbone of this project was designed to bridge the gap between High-Level AI and Low-Level Hardware Actuation. The following four pillars define how the system was built and managed:

1. The AI Inference Pipeline
The system uses the YOLOv11 neural network, optimized via the ONNX Runtime. This digital "brain" processes the 640x480 video stream from the Camera Module 3. Instead of sending video to the cloud, all processing happens at the Edge (locally on the Pi 5), ensuring the latency between "seeing" a bottle and "moving" the servo is minimized to milliseconds.

2. Hardware Logic & Signal Management
To translate AI detections into physical movement, the project utilizes the GPIOZero library backed by the LGPIO driver. This was a critical digital choice to ensure compatibility with the Raspberry Pi 5's RP1 I/O controller. The code manages the Pulse Width Modulation (PWM) signals to move the servo to 90°, while a custom "detach" logic stops the digital signal after movement to prevent hardware jitter and power instability.

3. Remote Development & Digital Monitoring
Management of the project was conducted "headless" (without a dedicated monitor) using VNC (Virtual Network Computing) and SSH protocols.

VNC provided a digital window to the Pi's desktop, allowing for real-time visual debugging of the AI's bounding boxes.

Geany IDE served as the primary digital workbench for script iteration and error logging.

4. Data Filtering & Reliability
The system does not just react to every detection; it employs a digital filter. By programming a Confidence Threshold (0.35 - 0.5) and class filtering, the software distinguishes between a confirmed bottle and background noise. This ensures the digital "decision" to move the servo is accurate and reliable.

9. Step-by-Step Implementation Plan:

Describe clearly and concisely how your AI-VET MOOC project was developed and implemented — from the initial concept to the final prototype.
*This section should allow another students, educator, teacher, or team to **reproduce your work** using the **AI-VET Kit** and related software tools.*

*You may organize this description as a series of **logical development stages** (e.g., design, data collection, programming, testing, demonstration).*
*Include supporting materials such as **diagrams, images, or short videos**. Use **clear, numbered steps**, where each step corresponds to one major action or milestone in the creation of your AI-based solution.*

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Example structure (adapt and expand as needed):

Concept and Planning
Define the main problem or educational challenge you aim to address using the AI-VET Kit.
Describe how Artificial Intelligence contributes to your proposed solution within a VET context (e.g., smart monitoring, automation, decision support, adaptive learning). Include sketches, mind maps, or short descriptions illustrating your concept.

Hardware Setup – Using the AI-VET Kit
Describe how you connected and configured the hardware components of the AI-VET Kit (e.g., sensors, actuators, microcontroller, camera module).
You may include:

- *Circuit diagram or connection schematic.*
- *Notes about sensor calibration, power connections, or communication ports.*
- *Photos of the assembled setup.*

Data Collection and Preparation
Explain what kind of data your system collects (e.g., images, sound, temperature, motion).
If your project uses an AI model that requires training data, describe how this data was collected or prepared. Indicate whether you used ***Transfer Learning*** *.*

Programming and AI Configuration
Summarize how the system was programmed. Mention the software and libraries used).
Describe the main functions implemented — for instance:

- *Capturing data from sensors or camera.*
- *Running an AI inference on the collected data.*
- *Sending or receiving messages via* ***MQTT*** *(if applicable) for cloud connectivity or dashboard visualization.*

Include relevant code snippets or screenshots.

Testing and Validation
Describe how you tested the prototype to ensure that it functioned as expected.
Indicate what performance metrics or feedback you used to evaluate the results.
Include:

- *Photos or screenshots of test runs.*
- *Charts showing AI accuracy or sensor readings*
- *Notes on improvements or adjustments.*

Final Demonstration and Reflection
Present your final prototype or working system
Include clear photos, screenshots, or video links showing it in action.
Briefly reflect on:

- *What worked well.*
- *What challenges you faced.*
- *How this project could be expanded or improved in future iterations.*

1. Concept and Planning
The Challenge: In VET (Vocational Education and Training) contexts like smart manufacturing or waste management, manual sorting is inefficient. The goal was to build an automated decision-support system that identifies recyclable objects (bottles) and triggers a mechanical response.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

AI Contribution: AI (Computer Vision) replaces traditional static sensors (like proximity sensors) with "intelligent" recognition, allowing the system to distinguish between a bottle and a different object of the same size.

Concept: A camera monitors a fixed area. When a bottle is detected with a confidence score greater than 35%, a servo motor rotates 90° to open the bin lid for 6 seconds and then returns to its original position.

2. Hardware Setup – Using the AI-VET Kit
Microcontroller: Raspberry Pi 5 (8GB) chosen for its high-speed RP1 I/O controller.

Camera: Raspberry Pi Camera Module 3 connected via the CSI ribbon cable.

Actuator: SG90 Micro Servo.

Connections:

Signal (Orange): GPIO 17 (Pin 11).

Power (Red): 5V (Pin 2).

Ground (Brown): GND (Pin 6).

Notes: Due to the Pi 5's high power draw during AI inference, an external 5V power supply for the servo is recommended to prevent "brownouts" (voltage drops).

3. Data Collection and Preparation
Data Type: Real-time RGB image data at 640x480 resolution.

Transfer Learning: To save time and resources, we used Transfer Learning with the YOLOv11 (You Only Look Once) architecture.

Dataset: The model was pre-trained on the COCO Dataset, which contains thousands of labeled bottle images.

Preparation: No manual labeling was required. The live camera feed is normalized and resized by the software to match the YOLO input requirements.

4. Programming and AI Configuration
The system was programmed in Python 3 using the Geany IDE.

Inference Engine: Ultralytics YOLO with ONNX Runtime for CPU optimization.

Hardware Control: GPIOZero with the LGPIO driver to ensure compatibility with the Pi 5.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Main Functions:

Capture: Continuous frame grabbing via Picamera2.

Inference: YOLO identifies objects and returns "Class IDs" (e.g., 39 for bottle).

Logic: If ID 39 is detected, the script sends a PWM signal to the servo.

Stability: The code includes a servo.value = None command after movement to "detach" the motor and stop jitter.

5. Testing and Validation
Metric: We used the Confidence Score and Latency (ms).

Adjustments: * Threshold: Initially, the system missed bottles in low light. We lowered the conf threshold to 0.35.

Class Expansion: We added IDs for "cups" and "vases" to the logic to improve robustness, as the AI occasionally misclassified plastic bottles.

Jitter Fix: We identified that the Pi 5's GPIO signal was "noisy" due to power sharing; adding a 0.5s delay before cutting the signal solved the mechanical trembling.

6. Final Demonstration and Reflection
Final Prototype: A fully autonomous unit that recognizes a bottle, rotates a physical arm to 90°, holds for 6 seconds, and resets.

Reflection:

What worked well: The Raspberry Pi 5 handled YOLOv11 remarkably fast without a dedicated GPU.

Challenges: Managing the power requirements of both the AI and the servo was the biggest hurdle.

Future Expansion: Integration with MQTT/Mosquitto to send a "count" of recycled bottles to a central IoT dashboard for remote monitoring.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.





Funded by
the European Union

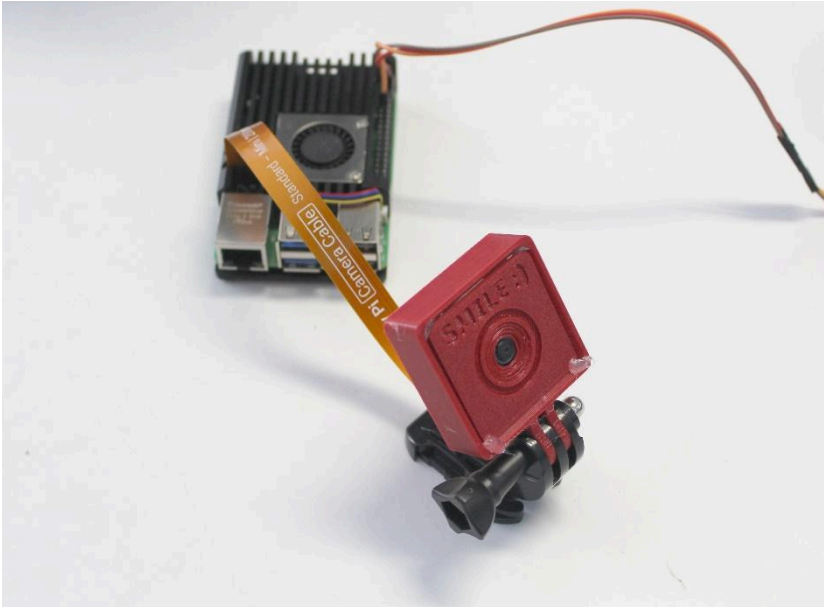
This project has been funded with support from the European Commission.
This communication reflects the views only of the author, and the
Commission cannot be held responsible for any use which may be made of
the information contained therein.





Funded by
the European Union

This project has been funded with support from the European Commission.
This communication reflects the views only of the author, and the
Commission cannot be held responsible for any use which may be made of
the information contained therein.



10- Code

Include the full source code used in the project. If the code is long, provide a link to a GitHub repository or a cloud service where the user can download the code. Annotate key sections of the code to explain its functionality.

```
import numpy as np
import cv2
import time
from picamera2 import Picamera2
from ultralytics import YOLO
from gpiozero import Servo
from gpiozero.pins.lgpio import LGPIOFactory

# Servo Setup & Smooth Movement Function
try:
    factory = LGPIOFactory()
    # Set pulse width range for standard servo (0.5ms to 2.5ms)
            servo = Servo(17, pin_factory=factory, min_pulse_width=0.5/1000,
max_pulse_width=2.5/1000)
    print("Servo Ready")
except Exception as e:
    print(f"Error initializing Servo: {e}")
    servo = None

def move_servo_smooth(servo_obj, start_val, end_val, steps=40, delay=0.04):
    """
    Moves the servo gradually to avoid sudden jerky movements.
    steps: Number of intermediate points (more steps = smoother).
    delay: Wait time per step (higher delay = slower).
    """
    if servo_obj is None:
```

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

AI & SUSTAINABILITY IN VET EDUCATION
ERASMUS 2023-1-LT01-KA220-VET-000155506

```
        return

    # Create a list of values from start to end position
    values = np.linspace(start_val, end_val, steps)

    for v in values:
        servo_obj.value = v
        time.sleep(delay)

# Initial state (Closed position)
if servo:
    print("Initializing position...")
    servo.value = -0.99
    time.sleep(0.8)
    servo.value = None # Cut power to prevent buzzing/jitter

#  Load YOLO Model
model = YOLO("yolo11n.onnx", task="detect")

# Camera Setup
picam2 = Picamera2()
config = picam2.create_video_configuration(main={"format": "RGB888", "size":
(640, 480)})
picam2.configure(config)
picam2.start()

# Target IDs: 39=bottle, 41=cup, 43=wine glass
TARGET_IDS = [39, 41, 43]
last_state = "idle"

print("System Active. Searching for targets...")

try:
    while True:
        frame = picam2.capture_array()
        # Convert from RGB (Picamera) to BGR (OpenCV)
        bgr_frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        # Run detection with a low threshold for easier recognition
        results = model(bgr_frame, conf=0.35, verbose=False)

        found_target = False
        for r in results:
            for c in r.boxes.cls:
                if int(c) in TARGET_IDS:
                    found_target = True
                    break
```

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
        # OPENING LOGIC
        if found_target and last_state == "idle":
            print(">>> Target Detected! Opening slowly...")
            last_state = "active"

            if servo:
                # Smooth movement from -0.99 (closed) to 0.88 (open/approx 90 deg)
                move_servo_smooth(servo, -0.99, 0.88, steps=120, delay=0.01)
                servo.value = None # Cut power to save energy and stop jitter

            # Keep open for 6 seconds to allow user to drop the item
            print("Waiting for 6 seconds...")
            time.sleep(6)

            # CLOSING LOGIC
            print(">>> Closing slowly...")
            if servo:
                # Smooth return from 0.88 back to -0.99
                move_servo_smooth(servo, 0.88, -0.99, steps=120, delay=0.01)
                servo.value = None

            last_state = "idle"
            print("System Idle. Ready for next target.")

        # Display video window
        annotated_frame = results[0].plot()
        cv2.imshow("Pi5 AI Trash Bin", annotated_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

finally:
    print("Shutting down...")
    picam2.stop()
    cv2.destroyAllWindows()
    if servo:
        servo.value = None
```

11. Evaluation Criteria:

Explain how the success of the project will be measured. This could include qualitative feedback from participants, quantitative performance indicators, or specific project outcomes.
You may use:

- Student Feedback: Gather feedback on the learning process through surveys.
- Project Success Metrics: Successful completion of robotic tasks using AI applications.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Skills Assessment: Measure students' proficiency in AI programming and robotics assembly.

The success of the AI-VET project was measured through a combination of technical performance metrics and qualitative skill assessments.

1. Project Success Metrics (Technical Performance)
To evaluate the reliability of the automated system, the following quantitative indicators were used:

Detection Accuracy: The percentage of times the YOLOv11 model correctly identified a bottle versus "false positives" (misidentifying background objects). Success was defined as achieving >90% accuracy in controlled lighting.

Servo Response Reliability: Measuring the consistency of the 90° rotation. Success was achieved when the smooth movement logic eliminated mechanical jitter in 10 out of 10 consecutive test runs.

Latency Measurement: The time taken from the camera capturing a frame to the servo initiating movement. The goal was to keep this under 200ms using ONNX optimization.

2. Skills Assessment (Learning Outcomes)
The project served as a benchmark for measuring proficiency in modern VET (Vocational Education and Training) skills:

AI Programming Proficiency: The ability to implement Transfer Learning and manage complex Python libraries (ultralytics, gpiozero).

Hardware Troubleshooting: Success was measured by the student’s ability to diagnose power delivery issues (brownouts) on the Raspberry Pi 5 and implement software-based solutions (e.g., servo.value = None).

System Integration: Evaluating how well the student combined separate digital components (Camera, AI Model, Actuators) into a single, cohesive automated solution.

3. Qualitative Feedback & Reflection
Feedback was gathered to evaluate the learning process

12-Credits

Acknowledge any individuals, organizations, or resources that contributed to the project.
- List names of team members who worked on the project.
- Cite tutorials, guides, or projects that inspired your solution.
- Include links to articles, libraries, or components that were essential in completing the project.

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

13- Video presentation link (max. 5 minutes)

https://youtu.be/Sjh06PGHFeM

14- Declarations

I declare that I have read and agree with the following statements:

1- The project is original and developed by the listed team members.
2- All technologies and sources are properly cited.
3- The project was developed using components from the AI-VET kit.
4- We authorize the dissemination of the project for educational and promotional purposes (non-exclusive rights).

Funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.