Private & Confidential: **WASSERSTOFF INNOVATION & LEARNING LABS PRIVATE LIMITED**

Position: Ai Engineer Intern **DEADLINE: 4 Days(96 hrs)**

Submission Guidelines:

1. **Deadline**:
   - **Submission Date**: 4 days(96 hrs) from the assignment date.
   - **Time**: By 11:59 PM on the fourth day.
2. **Submission Method**:
   - Submit the project as a GitHub repository link or a zipped folder via email.
3. **Project Requirements**:
   - Complete the pipeline as per the task sheet.
   - Implement a Streamlit UI for testing the pipeline.
   - Ensure all deliverables are met.
4. **Presentation**:
   - Include a README.md file with an overview of the project, setup instructions, and usage guidelines.
   - Prepare a short presentation (3-5 slides) summarizing the approach, implementation, and results.

Task Sheet: Building an AI Pipeline for Image Segmentation and Object Analysis

Objective:

Develop a pipeline using transformers or deep learning models that processes an input image to segment, identify, and analyze objects within the image, and outputs a summary table with mapped data for each object.

Steps and Deliverables:

Step 1: Image Segmentation

Task: Segment all objects within an input image.

- **Deliverables**:
  - Implement a model or use a pre-trained model (e.g., Mask R-CNN, DETR) for image segmentation.
  - Code to input an image and output segmented regions for each object.
  - Visual output showing segmented objects within the image.

Suggested Tools/Resources: PyTorch, TensorFlow, pre-trained segmentation models.
Decide for yourself

Step 2: Object Extraction and Storage

Task: Extract each segmented object from the image and store separately with unique IDs.

- **Deliverables**:
  - Code to extract each segmented object and save them as separate images.
  - Assign a unique ID for each object and a master ID for the original image.
  - Save the object images and their metadata (unique ID, master ID) in a file system or database.

Tools/Resources: OpenCV, PIL, SQLite or any preferred database. Decide for yourself

Step 3: Object Identification

Task: Identify each object and describe what they are in the real world.

- **Deliverables**:
  - Implement a model or use a pre-trained model (e.g., YOLO, Faster R-CNN, CLIP) to identify and describe objects.
  - Code to generate a description for each object image.
  - Document containing the identified objects and their descriptions.

Suggested Tools/Resources: Pre-trained object detection models, CLIP. ***Decide for yourself***

Step 4: Text/Data Extraction from Objects

Task: Extract text or data from each object image.

- **Deliverables**:
  - Implement or use a pre-trained model (e.g., Tesseract OCR, EasyOCR) for text extraction.
  - Code to extract and store text/data from each object image.
  - Document containing extracted text/data for each object.

Suggested Tools/Resources: OCR tools, PyTorch, TensorFlow. ***Decide for yourself***

Step 5: Summarize Object Attributes

Task: Summarize the nature and attributes of each object.

- **Deliverables**:
  - Code to generate a summary of the nature and attributes of each object.
  - Document containing summarized attributes for each object.

Suggested Tools/Resources: NLP models, summarization algorithms. ***Decide for yourself***

Step 6: Data Mapping

Task: Map all extracted data and attributes to each object and the master input image.

- **Deliverables**:
    - Code to map unique IDs, descriptions, extracted text/data, and summaries to each object.
    - Data structure (e.g., JSON, database schema) representing the mapping.

Suggested Tools/Resources: JSON, SQL, any preferred database. ***Decide for yourself***

Step 7: Output Generation

Task: Output the original image along with a table containing all mapped data for each object in the master image.

- **Deliverables**:
    - Code to generate the final output image with annotations.
    - Table summarizing all data mapped to each object and the master image.
    - Final visual output showing the original image with segmented objects and an accompanying table.

Suggested Tools/Resources: Matplotlib, pandas, any visualization library. ***Decide for yourself***

General Requirements

- **Documentation**: Document all steps, methodologies, and code used.
- **Code Quality**: Ensure the code is well-commented, modular, and follows best practices.
- **Testing**: Provide test cases to verify the functionality of each step in the pipeline.
- **Presentation**: Prepare a presentation summarizing the approach, implementation, results, and any challenges faced.

Submission Guidelines:

5. **Deadline**:
    - **Submission Date**: 4 days(96 hrs) from the assignment date.
    - **Time**: By 11:59 PM on the fourth day.
6. **Submission Method**:
    - Submit the project as a GitHub repository link or a zipped folder via email.
7. **Project Requirements**:
    - Complete the pipeline as per the task sheet.
    - Implement a Streamlit UI for testing the pipeline.
    - Ensure all deliverables are met.
8. **Presentation**:
    - Include a README.md file with an overview of the project, setup instructions, and usage guidelines.

- Prepare a short presentation (3-5 slides) summarizing the approach, implementation, and results.

Code and Folder Structure Guidelines:

Folder Structure:

```
project_root/
│
├── data/
│   ├── input_images/               # Directory for input images
│   ├── segmented_objects/          # Directory to save segmented object
images
│   └── output/                     # Directory for output images and
tables
│
├── models/
│   ├── segmentation_model.py       # Script for segmentation model
│   ├── identification_model.py     # Script for object identification
model
│   ├── text_extraction_model.py    # Script for text/data extraction model
│   └── summarization_model.py      # Script for summarization model
│
├── utils/
│   ├── preprocessing.py            # Script for preprocessing functions
│   ├── postprocessing.py           # Script for postprocessing functions
│   ├── data_mapping.py             # Script for data mapping functions
│   └── visualization.py            # Script for visualization functions
│
```

```
├── streamlit_app/
│   ├── app.py                  # Main Streamlit application script
│   └── components/             # Directory for Streamlit components
│
├── tests/
│   ├── test_segmentation.py        # Tests for segmentation
│   ├── test_identification.py      # Tests for identification
│   ├── test_text_extraction.py     # Tests for text extraction
│   └── test_summarization.py       # Tests for summarization
│
├── README.md                   # Project overview and setup
instructions
├── requirements.txt            # Required Python packages
└── presentation.pptx           # Presentation slides summarizing the
project
```

Detailed Code Guidelines:

1. **Data Directory**:
   - `input_images/`: Store all input images here.
   - `segmented_objects/`: Save all segmented object images with unique IDs.
   - `output/`: Save the final output images and tables here.
2. **Models Directory**:
   - `segmentation_model.py`: Implement or integrate the image segmentation model.
   - `identification_model.py`: Implement or integrate the object identification model.
   - `text_extraction_model.py`: Implement or integrate the text/data extraction model.
   - `summarization_model.py`: Implement or integrate the summarization model.
3. **Utils Directory**:
   - `preprocessing.py`: Functions for preprocessing images before model input.
   - `postprocessing.py`: Functions for postprocessing model outputs.
   - `data_mapping.py`: Functions to map data to objects and master images.
   - `visualization.py`: Functions for visualizing segmented images and generating final output.
4. **Streamlit App Directory**:
   - `app.py`: Main script to launch the Streamlit UI for testing.
   - `components/`: Additional components or utilities for the Streamlit app.
5. **Tests Directory**:
   - `test_segmentation.py`: Unit tests for the segmentation functionality.
   - `test_identification.py`: Unit tests for the identification functionality.
   - `test_text_extraction.py`: Unit tests for the text extraction functionality.
   - `test_summarization.py`: Unit tests for the summarization functionality.
6. **Root Directory**:
   - `README.md`: Detailed documentation on the project setup, usage, and overview.
   - `requirements.txt`: List of required Python packages for the project.
   - `presentation.pptx`: Presentation summarizing the approach, implementation, results, and challenges.

Streamlit UI Requirements:

1. **File Upload**:
   - Allow users to upload an input image.
2. **Segmentation Display**:
   - Display the segmented objects on the original image.
3. **Object Details**:
   - Show extracted object images with unique IDs.
   - Display descriptions, extracted text/data, and summarized attributes for each object.
4. **Final Output**:
   - Display the final output image with annotations.
   - Present a table containing all mapped data for each object in the master image.
5. **User Interaction**:
   - Allow users to interact with and review each step of the pipeline.

Evaluation Criteria:

- **Accuracy**: Precision of segmentation, identification, and data extraction.
- **Efficiency**: Performance and speed of the pipeline.
- **Robustness**: Ability to handle diverse and complex images.
- **Clarity**: Quality of documentation and presentation.