**📑 Interim Submission Report – Improve Your Previous Week’s Projects**

**1. Project Overview and Understanding**

The project focuses on **improving and extending the work from the previous week**, where initial implementations of FastAPI services, Telegram client integration, and YOLO-based image enrichment were built. The overall objective is to create a **robust, production-ready system** capable of:

* Running APIs inside Docker for portability and scalability.
* Seamlessly integrating with **Telegram** using Telethon to fetch and process messages or media.
* Performing **object detection using YOLOv8** on images and storing results in a structured **PostgreSQL database** for further analytics.

This improvement cycle has been centered around **refactoring, error handling, and optimization** to ensure better reliability, cleaner codebase, and readiness for deployment.

**2. Methodology – Process, Progress So Far, and Results**

**Process Followed**

* **Code Review & Refactoring:** Carefully reviewed the initial implementation for inefficiencies, lack of error handling, and missing features.
* **Environment & Secrets Management:** Improved .env handling to ensure no sensitive data is hardcoded.
* **Async Integration:** Updated Telegram client methods to use proper async handling for scalability.
* **YOLO Enrichment Pipeline:** Automated the loop for image scanning, YOLO predictions, and database insertion with added safety checks.
* **Database Integration:** Ensured PostgreSQL queries are parameterized to prevent SQL injection and included transaction handling.

**Progress So Far**

* FastAPI service was enhanced with better structure, health-check endpoint, and Docker readiness.
* Telegram client was refactored for secure initialization, session handling, and tested for reliable connections.
* YOLO enrichment script now performs batch processing of images, with **confidence thresholds**, **logging**, and **graceful error recovery**.
* Detection results are successfully inserted into fct\_image\_detections table for structured analytics.

**Results**

* Verified FastAPI runs smoothly inside Docker and responds with API endpoints.
* Successfully authenticated and connected to Telegram, tested by fetching account details.
* YOLO pipeline processed test images, extracted detected objects, and stored metadata in the database without errors.

## 3. Challenges & Solutions

* **Challenge 1**: Managing environment variables securely.
  + ✅ **Solution**: Centralized .env usage with python-dotenv and default fallbacks.
* **Challenge 2**: Unclear message ID extraction from filenames.
  + ✅ **Solution**: Wrote a helper function with error handling to standardize extraction.
* **Challenge 3**: Potential database connection leaks.
  + ✅ **Solution**: Implemented context managers (with conn.cursor() as cur) and exception handling.
* **Challenge 4**: Model inference speed vs. accuracy tradeoff.
  + ✅ **Solution**: Chose yolov8n.pt for testing, with flexibility to upgrade to yolov8m.pt later.

**4. Future Plan – What’s Left and How I Plan to Finish**

* **Docker-Compose Integration:** Run FastAPI, PostgreSQL, and YOLO service inside Docker containers for full portability.
* **Logging & Monitoring:** Implement structured logging and monitoring (possibly with Prometheus + Grafana).
* **Error Notifications:** Add Telegram bot notifications for failed image enrichments.
* **Testing & QA:** Add unit tests and integration tests for API endpoints and enrichment pipeline.
* **Performance Optimization:** Optimize YOLO by experimenting with different model sizes (yolov8s.pt, yolov8m.pt) depending on accuracy vs. speed tradeoff.
* **Documentation:** Write detailed developer setup guide and usage instructions.

**5. Conclusion – Summary of Progress and Confidence**

The past week has been focused on **stabilizing and enhancing** the core project components. The FastAPI, Telegram client, and YOLO enrichment modules are now more **robust, secure, and production-ready**. With the foundation in place, the upcoming tasks will focus on **deployment, monitoring, and testing**.

I am confident that with these improvements and the planned future work, the project will not only meet the submission requirements but also serve as a **scalable real-world application** that integrates APIs, messaging platforms, machine learning, and databases seamlessly.