# NewsLetter - Project Documentation

## OVERVIEW

The NewsLetter project is an automated system that connects to an Oracle database to fetch data, processes it using a local vLLM (Qwen/Qwen3-4B) to create AI-generated summaries, and exports the final output as a PDF document. It supports generating summaries from both a business and a risk/RA perspective.

## REPOSITORY LAYOUT

The project's main directory is located at /home/vision/app/NewsLetter/. Its structure is as follows:

* config.json: The main configuration file, which holds database connection details and LLM endpoint information.
* scripts/: This directory contains all the core logic and execution scripts.
  + db\_access/: Scripts for connecting to and querying the Oracle database.
  + llm\_client/: Scripts for interacting with the local vLLM server.
  + summary\_agents/: Custom agents that guide the LLM to generate specific types of summaries (e.g., business vs. risk).
  + main.py: The command-line interface (CLI) entry point for the application.
  + pipeline.py: The script that orchestrates the entire summary generation process.
  + run\_summary.sh: A shell script used to run the main application.
  + run\_vllm.sh: A shell script used to start the vLLM server.
* template/: This directory stores the Microsoft Word templates used to render the final PDFs.
  + BS\_AI\_SUMMARY\_TEMPLATE.docx: Template for business summaries.
  + RA\_AI\_SUMMARY\_TEMPLATE.docx: Template for risk/RA summaries.
* logs/: This directory stores all runtime logs.
  + ai\_summary\_\*.log: Logs generated by each run of the main application.
  + vllm\_8011.log: Logs from the vLLM server.
* outputs/: This directory is where the final generated PDF files are saved.
  + bs\_summary\_\*.pdf: Generated business summary PDFs.
  + ra\_ai\_summary\_\*.pdf: Generated risk/RA summary PDFs.

## RUNTIME ENVIRONMENT

The project requires two separate **Conda environments** for its operations.

* **crewai\_env**: This environment runs the core application logic.
  + **Activation Command**: source /opt/miniconda/etc/profile.d/conda.sh &&conda activate crewai\_env
  + **Key Python Dependencies**: pandas, sqlalchemy, oracledb.
* **vllm\_env**: This environment is dedicated to running the vLLM server.
  + **Activation Command**: source /opt/miniconda/etc/profile.d/conda.sh &&conda activate vllm\_env
  + **Key Python Dependencies**: vllm.

## CONFIGURATION

The main configuration file is config.json. It contains critical details for the project's operation, including:

* **Database Credentials**: Connection details for the Oracle database, such as host, service\_name, and username.
* **vLLM Server**: The base API endpoint (VLLM\_API\_BASE) and the name of the LLM model (Qwen/Qwen3-4B) being used.

**Note**: It is a best practice to move sensitive information like database credentials from config.json to environment variables to avoid storing them in plaintext.

## HOW TO RUN

The project is executed in two main steps: starting the vLLM server and then running the main application.

### **1. Start the vLLM Server**

Navigate to the project's root directory and execute the server script. The server will run in the background.

cd /home/vision/app/NewsLetter  
./scripts/run\_vllm.sh

### **2. Run the Main Application**

After the vLLM server is running, use the run\_summary.sh script to run the main application. You must provide specific arguments for a successful execution.

# General Usage  
./scripts/run\_summary.sh --country [COUNTRY] --le\_book [BOOK\_ID] --pipeline\_type [TYPE] [--business\_date [YYYY-MM-DD]]  
  
# Example  
./scripts/run\_summary.sh --country "USA" --le\_book "12345" --pipeline\_type "ra\_summary"

## OUTPUT AND LOGS

* **Output Files**: The generated PDF summaries are stored in the **outputs/** directory. The naming convention includes the pipeline type and a timestamp, for example: outputs/ra\_ai\_summary\_20240812\_144200.pdf.
* **Log Files**: All runtime logs are saved in the **logs/** directory. Application-specific logs are named ai\_summary\_\*.log, and the vLLM server logs are stored in vllm\_8011.log.