Configuration Manual – Smishlock Holmes

1. Python Dependencies

The following packages are required for correct operation of the framework:

Package	Purpose
fastapi	REST API framework for the `/analyze` and `/geval` endpoints
uvicorn	ASGI server for running the FastAPI application
httpx	Asynchronous HTTP client for API requests
pydantic	Data validation and parsing using Python type hints
langchain	LLM orchestration and prompt management
langchain-core	Core components for LangChain operations
python-dotenv	Loads environment variables from `.env` files
pytest	Unit testing framework

Installation:

```
pip install -r requirements.txt
```

2. Environment Configuration

2.1 Environment File Setup

```
cp .env.template .env
```

2.2 Edit `.env` with:

- OPENAI_API_KEY – API key for OpenAI models (if applicable)
- MODEL_NAME – e.g., deepseek-v2-instruct or gpt-4.1
- MODEL_BASE_URL – Ollama or OpenAI endpoint URL
- LOG_DIR – directory for logs (default: logs)

3. Application Components

3.1 Main Application (app/)

main.py – The entry point of the FastAPI service, defining the `/analyze` and `/geval` routes, configuring middleware, and initiating dependency injection for model execution.
prompt_engine.py – Defines the JSON-embedded Chain-of-Thought (CoT) reasoning prompt.
llm_executor.py – Contains the asynchronous logic for sending requests to the configured LLM backend.
schemas.py – Houses all Pydantic models that validate incoming requests and outgoing responses.
geval.py – Implements the G-Eval framework for output scoring.

3.2 Logging (logs/)

results.jsonl – Sequential log of `/analyze` outputs, containing the original message, model reasoning, and classification results.
geval_scores.jsonl – Log of `/geval` evaluation scores for each analyzed message, including

correctness, completeness, and clarity metrics.
All logs are stored in JSONL format for easy parsing, reproducibility, and performance auditing.

3.3 Supporting Files

batch_test.py – Reads a list of test messages (e.g., `likelybenign.txt`, `suspicious.txt`) and sends them sequentially to `/analyze`.
run_geval.py – Processes previously logged `/analyze` outputs and submits them to `/geval`.
likelybenign.txt – Contains SMS examples expected to be classified as non-malicious.
suspicious.txt – Contains SMS examples suspected of being smishing attempts.

4. Running the Application

4.1 Development Server

```
uvicorn app.main:app --host 0.0.0.0 --port 8000
```

4.2 Endpoints

***/*analyze** – Accepts an SMS message and returns classification, explanation, and detected cues.
***/*geval** – Scores `/analyze` outputs for correctness, completeness, and clarity.