Request Engine - Developer Documentation

# Overview

Request Engine is a modular, asynchronous Python-based tool for sending HTTPS requests with:

* TLS fingerprint spoofing
* Proxy rotation
* Structured logging
* Response classification
* Prometheus-compatible metrics

It is designed for extensibility and observability, with minimal dependencies.

# Project Structure

Request\_Engine/

├── core/ # Core engine components

│ ├── config\_loader.py # Loads config.yaml

│ ├── proxy\_manager.py # Handles proxy rotation

│ ├── tls\_fingerprint.py # Applies fake TLS profile headers

│ ├── https\_engine.py # Makes actual HTTPS requests

│ ├── response\_classifier.py # Classifies responses (Success, Blocked, CAPTCHA)

│ ├── structured\_logger.py # Logs events in structured JSONL format

│ ├── metrics\_collector.py # Exposes Prometheus counters via HTTP

│ └── request\_manager.py # Orchestrates one or more request cycles

├── config/

│ └── config.yaml # User-defined targets, proxies, TLS profiles

├── logs/

│ └── logs.jsonl # Structured logs output

├── main.py # Entry point using RequestManager

├── make\_request.py # Testing script for individual requests

├── requirements.txt # Python dependencies

└── README.md

# Setup & Install

1. Clone the repo
2. Create a virtual environment
3. Install requirements:

pip install -r requirements.txt

1. Run the engine:

python main.py

1. Access Prometheus metrics:

http://localhost:8000/metrics

# Component Details

### **1. ConfigLoader**

* Loads YAML config from config/config.yaml
* Returns dictionary used by all modules

### **2. ProxyManager**

* Rotates list of proxies in round-robin fashion
* Optional validator function (to be added)

### **3. TLSFingerprintModule**

* Mimics real browsers via User-Agent injection
* Profiles supported: Chrome\_108, Firefox\_102, etc.

### **4. HTTPSRequestEngine**

* Async HTTP client using httpx.AsyncClient
* Accepts headers, TLS profile, and proxy
* Handles timeout and request exceptions

### **5. ResponseClassifier**

* Detects:
  + SUCCESS (200)
  + BLOCKED (403/429)
  + CAPTCHA (via keyword search)
  + UNKNOWN (everything else)

### **6. StructuredLogger**

* Uses loguru to log structured events
* Output: logs/logs.jsonl
* Event types: request\_sent, response\_received

### **7. MetricsCollector**

* Prometheus counters for success/failure
* Exposed on http://localhost:8000/metrics
* Built with prometheus\_client

### **8. RequestManager**

* High-level orchestrator
* Loads config, cycles through URLs
* Calls TLS + proxy + engine + logger + metrics

# Configuration: config/config.yaml

targets:

- https://httpbin.org/get

- https://example.com

proxies:

- http://127.0.0.1:8080

- http://127.0.0.2:8080

tls\_profiles:

- Chrome\_108

- Firefox\_102

# Extending the Engine

* **New classifier**: Add method to ResponseClassifier
* **New export format**: Add method to StructuredLogger
* **Scheduled runs**: Add loop in main.py using asyncio.sleep()
* **Concurrency**: Use asyncio.gather() inside RequestManager
* **Web GUI**: Can be added with Gradio or Streamlit

# Dependencies

* [httpx==0.23.3](https://www.python-httpx.org/)
* [loguru](https://github.com/Delgan/loguru)
* [prometheus\_client](https://github.com/prometheus/client_python)
* [PyYAML](https://pyyaml.org/)

# Testing Tips

* Use make\_request.py to isolate the HTTPS engine
* For dry-run: disable proxies and test with simple GET endpoints
* Use curl http://localhost:8000/metrics to verify metrics

# Known Limitations

* Does not support real uTLS fingerprinting (header only for now)
* No retry/backoff logic yet
* Proxy validation is stubbed (always returns True)
* CAPTCHA detection is keyword-based only

# Handover Notes

* All modules are stateless and testable
* Orchestration is driven by RequestManager
* For CI/CD, use main.py as the entry point
* Prometheus is embedded and runs automatically on port 8000
* Logs are written to JSONL for ELK/Grafana compatibility

# Contact / Support

This engine was built by Harsh Nair. For further questions or escalations, refer to project documentation or extend modules under core/.