Automated LLM Refactoring Pipeline

Overview

This pipeline automatically detects design smells in the Apache Roller codebase, generates refactored code suggestions using the Groq LLM API (Llama models), and creates a Pull Request with the changes – all managed through GitHub Actions.

Key design decision: Code is processed at the **module level with size-based batching**. Each logical subsystem's files are packed into batches that fit within Groq's free-tier token-per-minute (TPM) limits, then sent sequentially with cooldown periods.

> **Note:** Refactored code is placed under `refactored_suggestions/` and is **never** applied to the actual source code automatically.

Pipeline Flowchart

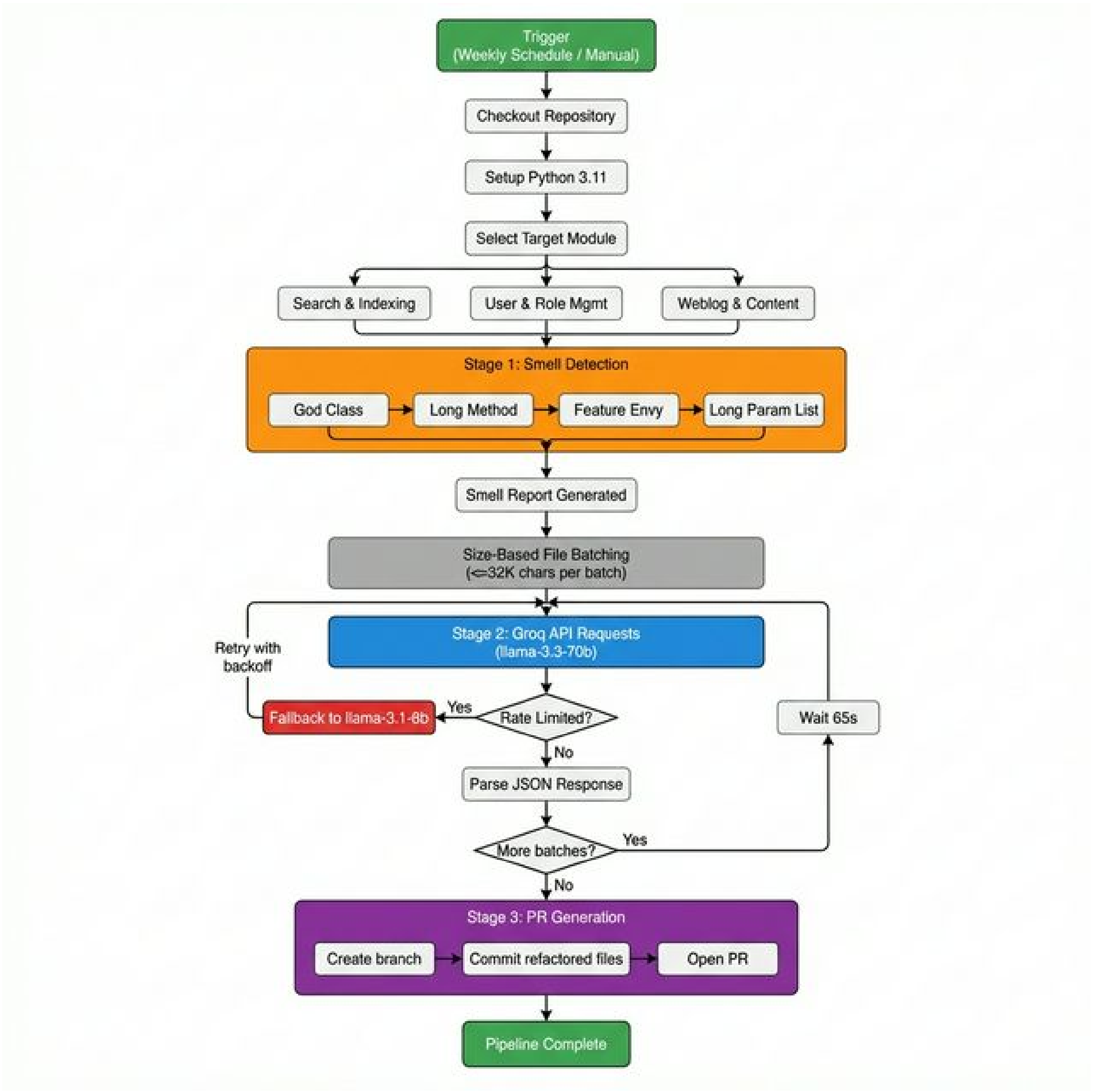

Figure 1: Pipeline Flowchart

Flow Summary:

1. **Trigger:** Weekly Schedule or Manual.
2. **Setup:** Checkout Repository and Setup Python 3.11.
3. **Module Selection:** Search & Indexing, User & Role Mgmt, or Weblog & Content.
4. **Stage 1: Smell Detection:** Identifies God Class, Long Method, Feature Envy, and Long Param List.
5. **Batching:** Size-based file batching (<= 32K chars per batch).
6. **Stage 2: Groq API Requests:**
   - Uses `llama-3.3-70b`.
   - Handles rate limits with a fallback to `llama-3.1-8b` or exponential backoff.
   - Wait 65s between batches to reset the TPM window.
7. **Stage 3: PR Generation:** Create branch, commit files, and open a Pull Request.

Batching & Rate Limit Strategy

The pipeline uses **size-based file batching** to work within Groq's free-tier limits:

Groq Free Tier Limits

Model	RPM	RPD	TPM
`llama-3.3-70b-versatile` (primary)	30	15,000	**12,000**
`llama-3.1-8b-instant` (fallback)	30	14,400	20,000

How it works:

1. **File reading:** All Java files in the selected module are read.
2. **Greedy packing:** Files are packed into batches of <=32K chars (~8K tokens), leaving ~4K tokens for LLM output within the 12K TPM limit.
3. **Sequential processing:** Each batch is sent as a separate API request.
4. **Inter-batch cooldown:** 65-second wait between batches to let the TPM window fully reset.
5. **Model fallback:** If `llama-3.3-70b-versatile` hits quota, automatically falls back to `llama-3.1-8b-instant`.
6. **Exponential backoff:** If still rate-limited, retries with 30s -> 60s -> 120s delays.

Batch sizes by module:

Module	Files	Lines	Batches	Est. Runtime
Search & Indexing	~25	~3,567	6	~6 min
User & Role Management	~16	~3,066	~5	~5 min
Weblog & Content	~79	~15,000+	~20+	~22 min

Design Smells Detected

Smell	Detection Rule	Severity
God Class (Size)	File > 300 lines	Medium/High
God Class (Methods)	> 15 public methods	Medium/High
Long Method	Method > 50 lines	Medium/High
Long Parameter List	Method with > 4 params	Medium
Feature Envy	Method with > 8 external class refs	Medium

Configuration

Repository Secrets Required

Secret	Description
`GEMINI_API_KEY`	Groq API key (stored under existing secret name)
`GITHUB_TOKEN`	Auto-provided by GitHub Actions

Running Locally

```
# Install dependencies
pip install -r scripts/requirements.txt

# Dry run (smell detection only -- no API key needed)
python scripts/refactor_pipeline.py --dry-run --module search

# Full run
export GEMINI_API_KEY="your-groq-api-key"
export GITHUB_TOKEN="your-token"
python scripts/refactor_pipeline.py --module search
```

GitHub Actions

The workflow runs **automatically every Monday** or can be triggered manually:

1. Go to **Actions** -> **LLM Refactoring Pipeline**.
2. Click **Run workflow**.
3. Select a module (`search`, `user`, or `weblog`).
4. The pipeline will create a PR with refactoring suggestions.

Files

File	Purpose
`scripts/refactor_pipeline.py`	Main pipeline script
`scripts/requirements.txt`	Python dependencies
`refactor-pipeline.yml`	GitHub Actions workflow
`refactoring-pipeline.md`	This documentation
`refactored_suggestions/`	Output directory