Google News RSS Ingestion Pipeline Approach Document

Overview

This project creates an offline-friendly automated pipeline for fetching, processing, and storing Google News RSS feeds. The system is designed to run on a RHEL VM and communicate externally only through a single configurable port.

Architecture

Core Components:

1. **Config Manager**: Handles loading and validation of configuration files
2. **RSS Fetcher**: Manages HTTP requests to Google News RSS feeds
3. **RSS Parser**: Processes XML responses and extracts relevant fields
4. **Storage Manager**: Handles file I/O operations and deduplication
5. **Scheduler**: Coordinates execution timing and retries
6. **Proxy Configurator**: Ensures all traffic routes through the specified port
7. **Logger**: Provides comprehensive logging of all operations

Data Flow:

1. Config Manager loads keywords from `feeds.json`
2. Scheduler triggers execution at specified times
3. For each keyword/group:
   - RSS Fetcher retrieves feed through configured proxy
   - RSS Parser extracts and normalizes data
   - Storage Manager deduplicates and persists to JSON
   - Logger records operation details
4. Process repeats with configured timing

Implementation Details

Configuration Management

- Store configurations in JSON format in `/config/` directory
- `feeds.json`: Keywords/groups for RSS feeds
- `settings.json`: System settings (proxy port, schedule, etc.)

Networking and Proxy

- Use environment variables for proxy configuration (http_proxy/https_proxy)
- Implement fallback to system-level proxy config if needed
- Set proper timeouts and retry logic

Data Processing

- Parse RSS XML with feedparser library
- Normalize and structure data in LLM-friendly JSON format
- Implement hash-based deduplication with multiple fallback methods

Storage Strategy

- Daily JSON files in `/feeds/` directory
- UTF-8 encoding with consistent formatting
- Optional statistics files for analytics

Scheduling and Resilience

- Use APScheduler for job scheduling
- Implement retry logic with exponential backoff
- Add defensive programming for robustness

Logging

- Hierarchical logging to files in `/logs/` directory
- Structured log format for easy parsing
- Configurable verbosity levels

Technical Considerations

Offline Operation

- All dependencies will be included in the requirements.txt
- Minimize external dependencies to essential libraries
- Ensure all operations except Google News requests can function offline

Security

- No hardcoded credentials or sensitive information
- Use environment variables for sensitive configuration
- Implement proper error handling to prevent information disclosure

Scalability

- Modular design to allow for future extensions
- Configuration-driven processing pipeline
- Separation of concerns across components

Testing Strategy

1. Unit tests for individual components
2. Integration tests for the complete pipeline
3. Mock responses for offline testing
4. Error injection to verify resilience

Deployment

- Simple directory structure with clear organization
- Requirements.txt for dependency management
- Optional Dockerfile for containerized deployment