Implementation Flowchart

This document describes the flow of the GenAIEmailClassification project.

# 1. Main Execution Flow

* Start
* ↓
* Check for test files (.eml)
* ↓
* No test files? → Log error and exit
* Yes → Load previous emails from previous\_emails.json
* ↓
* For each test file:

1. • Extract text from file
2. - Use file type–specific extraction (PDF, DOCX, EML, TXT)
3. - Fallback to OCR if extraction fails
4. • If EML file: Extract metadata and attachments
5. • Split email into multiple requests (if applicable)

* ↓
* For each email part:

1. • Call classify\_email()
2. - Try local fine-tuned model (classify\_with\_fine\_tuned\_model)
3. - If low confidence and DeepSeek enabled, use LLM (classify\_with\_deepseek)
4. - Optionally train local model with LLM output (train\_local\_model)
5. • Detect duplicates via detect\_duplicates()
6. - Compute sentence embeddings
7. - Analyze changes if duplicate is found (analyze\_changes\_with\_llm)
8. • Extract entities with extract\_entities() (using spaCy and regex)
9. • Determine email priority (determine\_priority)
10. • Assign to team (assign\_to\_team)

* ↓
* Append processed email text and timestamp
* ↓
* Store comprehensive results in email\_results.json
* Update previous\_emails.json

# 2. Classification Flow

* Function: classify\_email()

1. • Use local fine-tuned model (classify\_with\_fine\_tuned\_model)
2. - If confidence ≥ 0.95, return local result
3. • Otherwise, if DEEPSEEK is enabled, call classify\_with\_deepseek()
4. - Upon LLM success, train local model via train\_local\_model()
5. • If all steps fail, return a default response

# 3. File Extraction & Entity Processing

* Function: extract\_text\_from\_file()

1. • Determine file type: PDF, DOCX, EML, TXT
2. • Apply OCR fallback when needed

* Function: extract\_entities()

1. • Use spaCy to extract entities
2. • Use regex to extract amounts, dates, and IDs

* Function: detect\_duplicates()

1. • Compute cosine similarity between embeddings
2. • Trigger change analysis if similar email found

# 4. Maintenance Flow (maintenance.py)

* Scheduled cleanup of previous\_emails.json

1. • Load previous emails
2. • Remove entries older than a given threshold (e.g., 30 days) using cleanup\_previous\_emails()
3. • Save cleaned data back to previous\_emails.json
4. • Schedule task to run daily at 2:00 AM using the schedule library

# 5. Overview Summary

* Main Execution → Process Each File → Extract/Analyze/Classification → Duplicate Check → Finalize & Save Results