March 26, 2025

TEAM: GUARDINAS of the galaxy

[Company name]

[Company address]

**Smart Email Orchestrator**

AI-Driven Efficiency for Every Inbox

Contents

[**1. Introduction** 2](#_Toc193840958)

[**2. System Overview** 2](#_Toc193840959)

[**Objectives:** 2](#_Toc193840960)

[**Key Features:** 2](#_Toc193840961)

[**3. Architecture Diagram** 3](#_Toc193840962)

[**4. System Components** 4](#_Toc193840963)

[**4.1 Web Interface (Frontend)** 4](#_Toc193840964)

[**4.2 Web Server (Backend)** 4](#_Toc193840965)

[**4.3 Email Processor** 4](#_Toc193840966)

[**4.4 AI Classifier** 4](#_Toc193840967)

[**4.5 Data Handler** 5](#_Toc193840968)

[**5. Data Flow** 5](#_Toc193840969)

[**6. Deployment Considerations** 5](#_Toc193840970)

[**7. Conclusion** 6](#_Toc193840971)

**1. Introduction**

The **Smart Email Orchestrator** is a web-based system that automates the classification of emails using AI techniques. It extracts email content, performs text classification, and applies OCR for image attachments. This document outlines the architecture, data flow, and design considerations.

The system also handles complex email scenarios, including:

* **Multi-request Emails with Primary Intent Detection**: Identifies and processes emails containing multiple requests while determining the sender's main intent.
* **Duplicate Email Detection**: Recognizes duplicate emails from multiple replies or forwards within the same thread to reduce redundancy.
* **Priority-based Extraction**: Prioritizes extraction rules to ensure critical information is processed first, such as extracting numerical fields from attachments. **Smart Email Orchestrator** is a web-based system that automates the classification of emails using AI techniques. It extracts email content, performs text classification, and applies OCR for image attachments. This document outlines the architecture, data flow, and design considerations.

**2. System Overview**

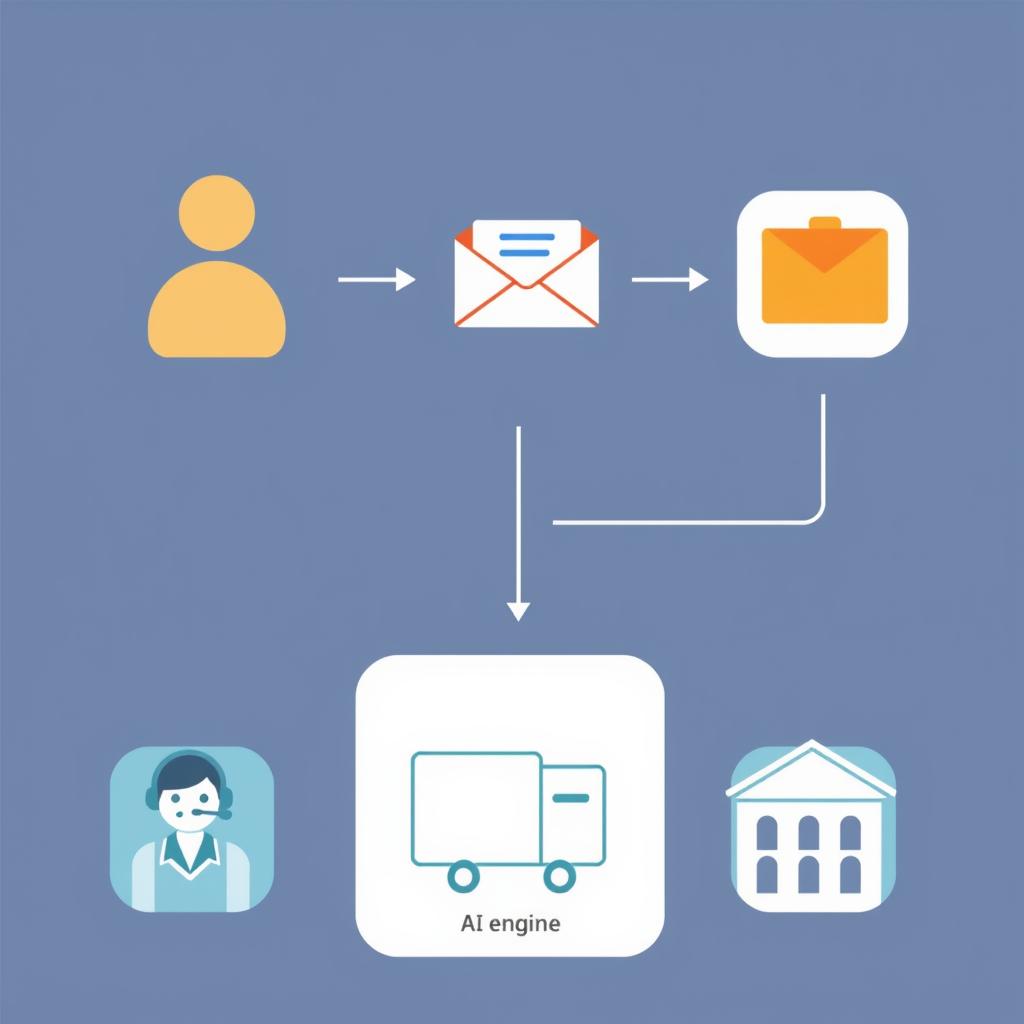
**Objectives:**

* Automate the extraction and classification of email content.
* Use a zero-shot AI model for flexible and accurate categorization.
* Extract text from image attachments via OCR.
* Provide a user-friendly web interface for uploading, classifying, and downloading results.

**Key Features:**

* **Email Extraction**: Parse .eml files to extract subject, body, and attachments.
* **AI-based Classification**: Classify emails into predefined categories using the facebook/bart-large-mnli model.
* **Optical Character Recognition (OCR)**: Extract text from images in email attachments.
* **User Interface**: A web interface for email upload, result viewing, and data download.

**3. Architecture Diagram**



**4. System Components**

**4.1 Web Interface (Frontend)**

* **Technologies**: HTML, CSS (Bootstrap), JavaScript (DataTables)
* **Functions**:
  + Upload .eml files.
  + Display classification results.
  + Provide options to download structured output (JSON).

**4.2 Web Server (Backend)**

* **Framework**: Flask (Python)
* **Functions**:
  + Handle HTTP requests (upload, process, display results).
  + Route endpoints for data operations.
  + Manage application state and configurations.

**4.3 Email Processor**

* **Components**:
  + email\_extraction.py: Parses .eml content (subject, body, attachments).
  + ocr.py: Extracts text from image attachments.

**4.4 AI Classifier**

* **Model**: facebook/bart-large-mnli (Zero-shot classification via Hugging Face Transformers)
* **Function**:
  + Classifies extracted email content.
  + Uses config.json for categories and subcategories.
  + Handles multi-intent emails by detecting and prioritizing the primary intent for accurate routing.
  + Identifies duplicate emails within a conversation thread to prevent redundant processing.

Example Initialization:

from transformers import pipeline

classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

**Multi-Intent Handling Strategy**:

* Extracts and analyzes multiple intents within a single email.
* Applies a scoring mechanism to prioritize and route based on the primary intent.
* Logs secondary intents for future reference or manual review.
* **Model**: facebook/bart-large-mnli (Zero-shot classification via Hugging Face Transformers)
* **Function**:
  + Classifies extracted email content.
  + Uses config.json for categories and subcategories.

Example Initialization:

from transformers import pipeline

classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

**4.5 Data Handler**

* **Libraries**: pandas, numpy
* **Function**:
  + Analyze, structure, and store classified data.
  + Output results as output.json.

**5. Data Flow**

1. **User Interaction**
   * Users upload an email through the web interface.
2. **Email Extraction**
   * Extracts email content and image attachments.
3. **OCR Processing**
   * Extracts text from images if present.
4. **Classification**
   * Runs zero-shot classification on email content.
5. **Output Handling**
   * Structured data is stored in output.json.
6. **Result Presentation**
   * Results are displayed in the web interface.

**6. Deployment Considerations**

* **Environment**: Python 3.x, Flask
* **Dependencies**: Listed in requirements.txt

Deployment Steps:

1. Clone the repository:
2. git clone https://github.com/yourusername/smart-email-orchestrator.git
3. cd smart-email-orchestrator
4. Install dependencies:
5. pip install -r requirements.txt
6. Run the application:
7. python app/main.py
8. Access the app: <http://localhost:5000>

**7. Conclusion**

The **Smart Email Orchestrator** is an AI-driven system that automates email classification, streamlining workflows and improving efficiency.