### **Documentation**

### **Alert System**

### 

### **System Overview**

System consists of several components:

1. **Main Script (main.py)**: Orchestrates fetching data, processing it, and sending emails.
2. **Data Fetching (fetch\_data.py)**: Handles database connections and data retrieval.
3. **Email Sending (send\_email.py)**: Sends the CSV report via email.
4. **Utilities (utils.py)**: Contains helper functions for file path handling.
5. **API Server (app.py)**: Provides endpoints to generate reports and manage user access.

### **Detailed Code Flow**

#### **1. Main Script (main.py)**

**Purpose**: Orchestrate the fetching of data, processing it, and sending emails.

**Flow**:

1. **Load Configuration**:
   * Reads configuration from config.json to get database settings, report parameters, and email settings.
2. **Fetch Data for Each Country**:
   * Uses ThreadPoolExecutor to handle concurrent data fetching for each country listed in the configuration.
   * Calls process\_country() for each country to fetch data and generate CSV reports.
3. **Send Emails**:
   * After fetching the data, calls send\_email() to send the generated CSV reports via email.

**Key Parts**:

* ThreadPoolExecutor manages concurrent execution of tasks.
* process\_country() function calls fetch\_data() to fetch and save the data.
* send\_email() is executed to send emails with the fetched CSV files.

#### **2. Data Fetching (fetch\_data.py)**

**Purpose**: Connect to the database, execute queries, and save results to a CSV file.

**Flow**:

1. **Establish Database Connection**:
   * Uses the MySQL Connector to connect to the database using configuration details.
2. **Execute Query**:
   * Runs a query to fetch data based on start\_ID and end\_ID.
   * Query is dynamically generated based on the configuration.
3. **Write to CSV**:
   * Writes the fetched data to a CSV file.
   * Ensures the directory for the CSV file exists before writing.

**Key Parts**:

* Handles database connection and query execution.
* Writes the results to a CSV file and handles any potential errors.

#### **3. Email Sending (send\_email.py)**

**Purpose**: Send emails with the generated CSV files as attachments.

**Flow**:

1. **Prepare Email Details**:
   * Reads email configuration from config.json and prepares email details.
2. **Send Email**:
   * Uses Mailgun API to send the email with the CSV attachment.

**Key Parts**:

* Uses environment variables to get Mailgun API credentials.
* Sends an email using a POST request to Mailgun's API.

#### **4. Utilities (utils.py)**

**Purpose**: Provide utility functions, such as getting absolute file paths.

**Flow**:

1. **Get CSV Output Path**:
   * Converts a relative path to an absolute path for the CSV file.

**Key Parts**:

* Provides a helper function to construct absolute paths for files.

#### **5. API Server (app.py)**

**Purpose**: Provide an API to generate reports and manage user access.

**Flow**:

1. **Authentication**:
   * Uses basic authentication to protect API endpoints.
2. **Generate Report Endpoint**:
   * Executes a query to generate a report and saves it to a CSV file.
   * Responds with the path to the generated report or an error message.
3. **Revoke Access Endpoint**:
   * Placeholder endpoint for revoking user access.

**Key Parts**:

* Uses Flask for routing and handling HTTP requests.
* Provides endpoints to generate reports and manage access.