Big Data - Coursework 2

Release 1.0

Dogwood

Apr 30, 2025

CONTENTS:

1 Installation Guide 1

2 Config Modules 3

3 Source Modules 7

4 Indices and tables 11

Python Module Index 13

Index 15

CHAPTER

ONE

INSTALLATION GUIDE

To get started with the project, follow these steps:

1. **Clone the Repository**: If you haven't already, clone the repository using Git.

   ```
   git clone https://github.com/imanzaf/ift_coursework_2024.git
   ```

2. **Install Python and Poetry**: - Ensure you have Python installed. You can download it from https://python.org. - Install Poetry by following the official Poetry installation guide.
3. **Navigate to the Project Directory**: Move into the cloned repository directory.

   ```
   cd team_dogwood/coursework_two
   ```

4. **Install Dependencies**: Use Poetry to install the project dependencies.

   ```
   poetry install
   ```

5. **Set Up Environment Variables**: Create a *.env* file in the root directory and populate it with the required credentials.

   ```
   cp .env.template .env
   ```

6. **Run the Project**: Use Poetry to run the main script.

   ```
   poetry run python src/main.py
   ```

Services

The project uses the following services via Docker:

- MongoDB
- PostgreSQL
- MinIO

To start them:

```
docker compose up
```

To stop them:

```
docker compose down
```

Usage Options (CLI)

You can run the script in different modes:

- Run once:

  ```
  poetry run python src/main.py --run-once
  ```

- Run immediately then schedule (default: monthly):

  ```
  poetry run python src/main.py --run-now
  ```

- Set custom schedule:

  ```
  poetry run python src/main.py --schedule weekly
  ```

- Combine flags:

  ```
  poetry run python src/main.py --run-now --schedule weekly
  ```

Usage Options (.env)

You can configure scheduling and service credentials through the *.env* file:

1. Copy the template:

   ```
   cp .env.template .env
   ```

2. Edit the file with your preferences and credentials.
3. Run without CLI flags:

   ```
   poetry run python src/main.py
   ```

Running Unit Tests

To run unit tests:

```
poetry run pytest
```

Running Pre-commit Hooks

Install pre-commit hooks:

```
poetry run pre-commit install
```

Run all hooks manually:

```
poetry run pre-commit run --all-files
```

CHAPTER

TWO

CONFIG MODULES

class config.db.**DataBaseSettings**(**args: Any*, ***kwargs: Any*)

Bases: `BaseSettings`

Configuration for database settings, including PostgreSQL and MinIO.

This class defines the configuration settings required to connect to a PostgreSQL database and a MinIO storage service. The settings are loaded from environment variables or a *.env* file.

POSTGRES_DRIVER

The driver used to connect to the PostgreSQL database.

Type
str

POSTGRES_USERNAME

The username for the PostgreSQL database.

Type
str

POSTGRES_PASSWORD

The password for the PostgreSQL database.

Type
str

POSTGRES_PORT

The port on which the PostgreSQL database is running.

Type
str

POSTGRES_HOST

The host address of the PostgreSQL database.

Type
str

POSTGRES_DB_NAME

The name of the PostgreSQL database.

Type
str

MINIO_USERNAME

The username for the MinIO storage service.

Type
str

MINIO_PASSWORD
The password for the MinIO storage service.

Type
str

MINIO_PORT
The port on which the MinIO service is running.

Type
str

MINIO_BUCKET_NAME
The name of the bucket in MinIO where files are stored.

Type
str

MONGO_URI
The URI for connecting to a MongoDB database.

Type
str

MONGO_DB_NAME
The name of the MongoDB database.

Type
str

MONGO_COLLECTION_NAME
The name of the collection in the MongoDB database.

Type
str

Example

```
>>> database_settings = DataBaseSettings()
>>> print(database_settings.POSTGRES_HOST)
localhost
```

MINIO_BUCKET_NAME: str

MINIO_HOST: str

MINIO_PASSWORD: str

MINIO_PORT: str

MINIO_USERNAME: str

MONGO_COLLECTION_NAME: str

MONGO_DB_NAME: str

`MONGO_URI: str`

`POSTGRES_DB_NAME: str`

`POSTGRES_DRIVER: str`

`POSTGRES_HOST: str`

`POSTGRES_PASSWORD: str`

`POSTGRES_PORT: str`

`POSTGRES_USERNAME: str`

class `config.schedule.`**`ScheduleSettings`**(*_case_sensitive: bool | None = None, _nested_model_default_partial_update: bool | None = None, _env_prefix: str | None = None, _env_file: DotenvType | None = PosixPath('.'), _env_file_encoding: str | None = None, _env_ignore_empty: bool | None = None, _env_nested_delimiter: str | None = None, _env_nested_max_split: int | None = None, _env_parse_none_str: str | None = None, _env_parse_enums: bool | None = None, _cli_prog_name: str | None = None, _cli_parse_args: bool | list[str] | tuple[str, ...] | None = None, _cli_settings_source: CliSettingsSource[Any] | None = None, _cli_parse_none_str: str | None = None, _cli_hide_none_type: bool | None = None, _cli_avoid_json: bool | None = None, _cli_enforce_required: bool | None = None, _cli_use_class_docs_for_groups: bool | None = None, _cli_exit_on_error: bool | None = None, _cli_prefix: str | None = None, _cli_flag_prefix_char: str | None = None, _cli_implicit_flags: bool | None = None, _cli_ignore_unknown_args: bool | None = None, _cli_kebab_case: bool | None = None, _secrets_dir: PathType | None = None, *, FREQUENCY: str = 'monthly', RUN_NOW: bool = False, RUN_ONCE: bool = False*)

Bases: `BaseSettings`

Configuration for scheduling settings.

This class defines the configuration settings required for scheduling tasks. The settings are loaded from environment variables or a *.env* file.

`SCHEDULE`

The schedule for running tasks (e.g., “monthly”, “weekly”, etc.).

Type

str

Example

```
>>> schedule_settings = ScheduleSettings()
>>> print(schedule_settings.SCHEDULE)
monthly
```

FREQUENCY: str

RUN_NOW: bool

RUN_ONCE: bool

model_config: ClassVar[SettingsConfigDict] = {'arbitrary_types_allowed': True, 'case_sensitive': True, 'cli_avoid_json': False, 'cli_enforce_required': False, 'cli_exit_on_error': True, 'cli_flag_prefix_char': '-', 'cli_hide_none_type': False, 'cli_ignore_unknown_args': False, 'cli_implicit_flags': False, 'cli_kebab_case': False, 'cli_parse_args': None, 'cli_parse_none_str': None, 'cli_prefix': '', 'cli_prog_name': None, 'cli_use_class_docs_for_groups': False, 'enable_decoding': True, 'env_file': '.env', 'env_file_encoding': 'utf-8', 'env_ignore_empty': False, 'env_nested_delimiter': None, 'env_nested_max_split': None, 'env_parse_enums': None, 'env_parse_none_str': None, 'env_prefix': 'SCHEDULE_', 'extra': 'ignore', 'json_file': None, 'json_file_encoding': None, 'nested_model_default_partial_update': False, 'protected_namespaces': ('model_validate', 'model_dump', 'settings_customise_sources'), 'secrets_dir': None, 'toml_file': None, 'validate_default': True, 'yaml_file': None, 'yaml_file_encoding': None}

Configuration for the model, should be a dictionary conforming to [*ConfigDict*][pydantic.config.ConfigDict].

CHAPTER

THREE

SOURCE MODULES

`src.db_utils.helpers.`**`append_reports_to_companies`**`(`*companies: list[Company]*, *db:* PostgreSQLDB`)` → list[Company]

Append ESG reports to each company.

`src.db_utils.helpers.`**`get_all_companies`**`(`*db:* PostgreSQLDB`)` → list[Company]

Get all companies from the database.

`class src.db_utils.minio.`**`MinioFileSystem`**

Bases: *`MinioFileSystemRepo`*

Overwrite file read and file write methods in MinioFileSystemRepo to add functionality to process PDF files.

`bucket_name`

The name of the MinIO bucket.

Type

str

`user`

The username for MinIO.

Type

str

`password`

The password for MinIO.

Type

str

`endpoint_url`

The endpoint URL used to connect to MinIO, consisting of the MinIO host address and port.

Type

str

`create_bucket``(`*bucket_name: str*`)`

Ensures the bucket exists. Creates it if it doesn't exist.

Parameters

`bucket_name` (*`str`*) – The name of the MinIO bucket.

Example

```
>>> minio = MinioFileSystem()
>>> minio.create_bucket("my-bucket")
# Creates a bucket named "my-bucket" if it doesn't exist.
```

download_file(*file_name: str*, *dest_path: str*)

Downloads a file from MinIO to a local path.

Parameters

- **file_name** (*str*) – The name of the file in the bucket.
- **dest_path** (*str*) – The local path to save the file (e.g., "./downloaded.pdf").

get_pdf_bytes(*object_name: str*) → bytes

Fetches a PDF file from MinIO as bytes.

Parameters
object_name (*str*) – The MinIO path (e.g., "123/2024/report.pdf").

Returns
The PDF file content as bytes.

Return type
bytes

list_files_by_company(*company_id*)

Lists all files for a specific company by prefix 'company_id/'.

Parameters
company_id (*str or int*) – The company ID.

Returns
A list of object names belonging to that company's folder.

Return type
list

upload_pdf(*local_file_path: str*, *company_id: str*, *report_year: str*)

Uploads a PDF into a subfolder structure: company_id/year/filename.pdf.

Parameters

- **local_file_path** (*str*) – The path to the local PDF file.
- **company_id** (*str*) – The ID of the company for which the PDF is being uploaded.
- **report_year** (*str*) – The year of the CSR report.

Returns
The object name (MinIO path), e.g., "123/2024/report.pdf".

Return type
str

view_pdf(*object_name: str*, *expiry_hours: int = 1*)

Generates a presigned URL to view the PDF in a web browser.

Users can open the link in their browser without explicitly downloading.

Parameters

- **object_name** (*str*) – The MinIO path (e.g., "123/2024/report.pdf").
- **expiry_hours** (*int, optional*) – The expiry time for the presigned URL in hours. Defaults to 1.

Returns
A presigned URL string. Returns None if an error occurs.

Return type
str

write_pdf_bytes(*pdf_bytes: bytes*, *file_size: int*, *company_id: str*, *report_year: str*, *file_name: str*)

Uploads a PDF (as bytes) into a subfolder structure: company_id/year/filename.pdf.

Parameters

- **pdf_bytes** (*bytes*) – The PDF file as bytes.
- **company_id** (*str*) – The ID of the company for which the PDF is being uploaded.
- **report_year** (*str*) – The year of the CSR report.
- **file_name** (*str*) – The name of the file to be saved.

Returns
The object name (MinIO path), e.g., "123/2024/report.pdf".

Return type
str

class src.db_utils.minio.**MinioFileSystemRepo**

Bases: `object`

Dummy class to satisfy inheritance.

MongoDB collection class for interacting with the MongoDB database.

class src.db_utils.mongo.**MongCollection**

Bases: `object`

MongoDB collection class for interacting with the MongoDB database.

get_available_companies() → List[str]

List all unique company securities with parsed reports.

Returns
A list of strings like ['AAPL', 'MSFT', …].

get_available_years(*mongo_doc: dict*) → List[int]

Extract the report year from the report_metadata field in a parsed-report document.

Parameters
mongo_doc – Document from get_report_by_company().

Returns
List containing the year (e.g. [2023]), or empty list if not found.

get_report_by_company(*company: Company*) → List[Document]

Get a report document by company.

Parameters
company (*Company*) – The company to get the report for.

Returns
The report documents.

Return type
list[Document]

insert_report(*company: Company*, *report_metadata: ESGReport*, *report: List[Document]*) → None

Insert a report document into the MongoDB collection.

Parameters
report_dict (*dict*) – The report document to insert.

Methods for interacting with postgres database.

class src.db_utils.postgres.**PostgreSQLDB**

Bases: object

Methods for connecting to and interacting with the PostgreSQL database.

This class provides methods for connecting to a PostgreSQL database, executing SQL operations, and managing database sessions. It supports both read and upsert (update/insert) operations.

Parameters
BaseModel – Inherits from Pydantic's BaseModel for data validation and settings management.

Example

```
>>> db = PostgreSQLDB()
>>> with db:
...     db.execute("read", sql_statement="SELECT * FROM companies")
```

delete_csr_report(*report_id*)

Deletes a CSR report record from the database by report_id.

execute(*query*, *params=None*)

Executes a SQL statement (INSERT, UPDATE, DELETE) and returns an empty list.

fetch(*query*, *params=None*)

Fetches data (SELECT) and returns a single dictionary.

get_csr_report_by_id(*report_id*)

Fetch a single CSR report by its primary key (report_id). (Assumes you have a 'report_id' column in your table.)

get_csr_reports_by_company(*company_name*)

Retrieve all CSR reports for a specific company, ordered by year desc.

update_csr_report(*report_id*, *new_url=None*, *new_year=None*)

Updates a CSR report's URL and/or year based on report_id. Only updates fields that are provided.

upsert_metrics(*table: str*, *rows: List[Dict]*) → None

Bulk UPSERT a list of metrics into the specified Postgres table.

Parameters

- **db** – PostgreSQLDB instance (open transaction).
- **table** – Table name ('emissions', 'energy', or 'waste').
- **rows** – List of metric dicts containing matching columns.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

src.db_utils.helpers, 7
src.db_utils.minio, 7
src.db_utils.mongo, 9
src.db_utils.postgres, 10

INDEX

A

append_reports_to_companies() (*in module src.db_utils.helpers*), 7

B

bucket_name (*src.db_utils.minio.MinioFileSystem attribute*), 7

C

config.schedule
  module, 5

create_bucket() (*src.db_utils.minio.MinioFileSystem method*), 7

D

DataBaseSettings (*class in config.db*), 3

delete_csr_report() (*src.db_utils.postgres.PostgreSQLDB method*), 10

download_file() (*src.db_utils.minio.MinioFileSystem method*), 8

E

endpoint_url (*src.db_utils.minio.MinioFileSystem attribute*), 7

execute() (*src.db_utils.postgres.PostgreSQLDB method*), 10

F

fetch() (*src.db_utils.postgres.PostgreSQLDB method*), 10

FREQUENCY (*config.schedule.ScheduleSettings attribute*), 6

G

get_all_companies() (*in module src.db_utils.helpers*), 7

get_available_companies() (*src.db_utils.mongo.MongCollection method*), 9

get_available_years() (*src.db_utils.mongo.MongCollection method*), 9

get_csr_report_by_id() (*src.db_utils.postgres.PostgreSQLDB method*), 10

get_csr_reports_by_company() (*src.db_utils.postgres.PostgreSQLDB method*), 10

get_pdf_bytes() (*src.db_utils.minio.MinioFileSystem method*), 8

get_report_by_company() (*src.db_utils.mongo.MongCollection method*), 9

I

insert_report() (*src.db_utils.mongo.MongCollection method*), 10

L

list_files_by_company() (*src.db_utils.minio.MinioFileSystem method*), 8

M

MINIO_BUCKET_NAME (*config.db.DataBaseSettings attribute*), 4

MINIO_HOST (*config.db.DataBaseSettings attribute*), 4

MINIO_PASSWORD (*config.db.DataBaseSettings attribute*), 4

MINIO_PORT (*config.db.DataBaseSettings attribute*), 4

MINIO_USERNAME (*config.db.DataBaseSettings attribute*), 3, 4

MinioFileSystem (*class in src.db_utils.minio*), 7

MinioFileSystemRepo (*class in src.db_utils.minio*), 9

model_config (*config.schedule.ScheduleSettings attribute*), 6

module
  config.schedule, 5
  src.db_utils.helpers, 7
  src.db_utils.minio, 7
  src.db_utils.mongo, 9
  src.db_utils.postgres, 10

MongCollection (*class in src.db_utils.mongo*), 9
MONGO_COLLECTION_NAME (*config.db.DataBaseSettings attribute*), 4
MONGO_DB_NAME (*config.db.DataBaseSettings attribute*), 4
MONGO_URI (*config.db.DataBaseSettings attribute*), 4

P

password (*src.db_utils.minio.MinioFileSystem attribute*), 7
POSTGRES_DB_NAME (*config.db.DataBaseSettings attribute*), 3, 5
POSTGRES_DRIVER (*config.db.DataBaseSettings attribute*), 3, 5
POSTGRES_HOST (*config.db.DataBaseSettings attribute*), 3, 5
POSTGRES_PASSWORD (*config.db.DataBaseSettings attribute*), 3, 5
POSTGRES_PORT (*config.db.DataBaseSettings attribute*), 3, 5
POSTGRES_USERNAME (*config.db.DataBaseSettings attribute*), 3, 5
PostgreSQLDB (*class in src.db_utils.postgres*), 10

R

RUN_NOW (*config.schedule.ScheduleSettings attribute*), 6
RUN_ONCE (*config.schedule.ScheduleSettings attribute*), 6

S

SCHEDULE (*config.schedule.ScheduleSettings attribute*), 5
ScheduleSettings (*class in config.schedule*), 5
src.db_utils.helpers
    module, 7
src.db_utils.minio
    module, 7
src.db_utils.mongo
    module, 9
src.db_utils.postgres
    module, 10

U

update_csr_report() (*src.db_utils.postgres.PostgreSQLDB method*), 10
upload_pdf() (*src.db_utils.minio.MinioFileSystem method*), 8
upsert_metrics() (*src.db_utils.postgres.PostgreSQLDB method*), 10
user (*src.db_utils.minio.MinioFileSystem attribute*), 7

V

view_pdf() (*src.db_utils.minio.MinioFileSystem method*), 8

W

write_pdf_bytes() (*src.db_utils.minio.MinioFileSystem method*), 9