Team Ginkgo

Release 1.0.0

Team Ginkgo

Feb 27, 2025

CONTENTS:

CHAPTER

ONE

INSTALLATION GUIDE

1.1 Prerequisites

Before installing the project, ensure you have the following prerequisites:

- Python 3.8 or higher
- pip (Python package installer)
- Git

1.2 Installation Steps

1. Clone the repository:

```
git clone https://github.com/iftucl/ift_coursework_2024.git
cd ift_coursework_2024/team_Ginkgo
```

2. Create a virtual environment (recommended):

```
python -m venv venv
source venv/bin/activate  # On Windows, use: venv\Scripts\activate
```

3. Install required packages:

```
pip install -r requirements.txt
```

1.3 Configuration

1. Set up environment variables:

   Create a *.env* file in the project root with the following variables:

```
MINIO_ENDPOINT=your_minio_endpoint
MINIO_ACCESS_KEY=your_access_key
MINIO_SECRET_KEY=your_secret_key
MINIO_BUCKET_NAME=your_bucket_name
```

2. Configure the database:

   Update the database configuration in *config.py* if needed.

1.4 Verification

To verify the installation:

1. Run the tests:

   ```
   python -m pytest tests/
   ```

2. Start the application:

   ```
   python modules/main.py
   ```

If you encounter any issues during installation, please refer to the troubleshooting section or raise an issue on the GitHub repository.

CHAPTER TWO

USAGE GUIDE

This guide explains how to use the main features of the Team Ginkgo project.

2.1 Basic Usage

The project consists of several modules that work together to scrape, process, and store financial data.

2.2 Running the Scraper

The scraper module can be used to collect financial data:

```python
from modules.scraper import FinancialDataScraper

# Initialize the scraper
scraper = FinancialDataScraper()

# Start scraping data
scraper.start_scraping()
```

2.3 Using the MinIO Client

Store and retrieve data using the MinIO client:

```python
from modules.minio_client import MinioClient

# Initialize the client
minio_client = MinioClient()

# Upload data
minio_client.upload_file('local_file.csv', 'remote_file.csv')

# Download data
minio_client.download_file('remote_file.csv', 'local_file.csv')
```

2.4 Database Operations

Interact with the database:

```python
from modules.database import Database

# Initialize database connection
db = Database()

# Store data
db.store_data(data_dict)

# Query data
results = db.query_data(query_params)
```

2.5 Scheduling Tasks

Use the scheduler to automate tasks:

```python
from modules.scheduler import Scheduler

# Initialize scheduler
scheduler = Scheduler()

# Add a task
scheduler.add_task(task_function, "0 */1 * * *")  # Run every hour

# Start the scheduler
scheduler.start()
```

2.6 Configuration

Modify the configuration in *config.py*:

```python
from modules.config import Config

# Load configuration
config = Config()

# Access configuration values
endpoint = config.get('MINIO_ENDPOINT')
```

2.7 Error Handling

The project includes comprehensive error handling. Here's how to handle common errors:

```python
try:
    # Your code here
    scraper.start_scraping()
except ConnectionError as e:
    print(f"Connection error: {e}")
except Exception as e:
    print(f"Unexpected error: {e}")
```

2.8 Best Practices

1. Always use environment variables for sensitive information
2. Regularly backup your data
3. Monitor the logs for any issues
4. Follow the rate limiting guidelines when scraping
5. Use appropriate error handling

For more detailed examples and advanced usage, refer to the API Reference section.

CHAPTER

THREE

ARCHITECTURE OVERVIEW

This document provides a comprehensive overview of the Team Ginkgo project architecture.

3.1 System Components

The project consists of several key components:

1. Data Collection Layer - Web Scraper (`scraper.py`) - Scheduler (`scheduler.py`)
2. Storage Layer - MinIO Client (`minio_client.py`) - Database Interface (`database.py`)
3. Configuration - Configuration Management (`config.py`)
4. Main Application - Application Entry Point (`main.py`)

3.2 Component Interactions

```
+-------------+     +-----------+     +-------------+
|   Scraper   |---->| Scheduler |---->| MinIO Client|
+-------------+     +-----------+     +-------------+
      |                                      |
      v                                      v
+-------------+                        +-----------+
|  Database   |<-----------------------| Main App  |
+-------------+                        +-----------+
```

3.3 Data Flow

1. Data Collection - The scraper collects financial data from various sources - Scheduled tasks are managed by the scheduler - Data is validated and preprocessed
2. Data Storage - Raw data is stored in MinIO object storage - Processed data is stored in the database - Data versioning is maintained
3. Data Access - API endpoints for data retrieval - Query interface for database access - File operations for MinIO storage

3.4 Technology Stack

- **Programming Language**: Python 3.8+
- **Object Storage**: MinIO

- **Database**: PostgreSQL
- **Task Scheduling**: APScheduler
- **Web Scraping**: BeautifulSoup4, Requests
- **Configuration**: Python-dotenv

3.5 Security Considerations

1. Authentication - MinIO access credentials - Database connection security - API authentication
2. Data Protection - Encrypted storage - Secure data transmission - Access control
3. Error Handling - Graceful failure recovery - Error logging and monitoring - Data consistency checks

3.6 Scalability

The architecture is designed to be scalable:

1. Horizontal Scaling - Multiple scraper instances - Distributed storage - Load balancing
2. Vertical Scaling - Resource allocation - Performance optimization - Cache management

3.7 Monitoring and Logging

- Comprehensive logging system
- Performance monitoring
- Error tracking and reporting
- Resource usage monitoring

3.8 Future Considerations

1. Planned Improvements - Enhanced data processing - Additional data sources - Advanced analytics
2. Potential Extensions - Real-time processing - Machine learning integration - API expansion

CHAPTER FOUR

API REFERENCE

This section provides detailed documentation for all modules and their components.

4.1 Scraper Module

4.2 MinIO Client Module

4.3 Database Module

4.4 Scheduler Module

4.5 Configuration Module

This file contains the configuration for the project. The main configuration settings include: - Google API Key - Google Custom Search Engine ID - Database Configuration - MINIO Configuration These settings are used by the different modules of the project to interact with external services and resources. For example, the Google API Key is used to access Google Custom Search API, the Database Configuration is used to connect to the SQL database, and the MINIO Configuration is used to store and retrieve files from the MINIO object storage service.

4.6 Main Module

CHAPTER

FIVE

TEAM GINKGO PROJECT

This is the documentation for Team Ginkgo's Big Data in Quantitative Finance project. This documentation provides comprehensive information about the project's setup, usage, and API reference.

CHAPTER

SIX

QUICK LINKS

- *Installation Guide*
- *Usage Guide*
- *Architecture Overview*
- *API Reference*

CHAPTER SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX