API Data to Database

Grand Valley State University

CIS 611: Introduction to Software Engineering

Instructor: Dr. Byron DeVries

Applied Computer Science, M.S.

Submitted by

Vaishnavi Prakash Rasane,

1. Project Overview:

The project addresses the increasing need for efficient and reliable data processing in the rapidly evolving crypto currency market. Crypto currency data, being dynamic and vast, requires a streamlined approach to extraction, transformation, and loading for effective analysis and decision-making. This project aims to design and implement an ETL pipeline for extracting crypto currency data from the CoinCap API, transforming it, and loading it into a MariaDB database into a structured format for further analysis.

1.1 Project Scope:

The initial scope of the project includes the implementation of a basic ETL pipeline capable of handling data from the CoinCap API. The system will be designed to be extensible, allowing for future integration with additional data sources and expanded functionality. While the primary focus is on cryptocurrency data, the underlying ETL architecture can be adapted for various data sources and industries.

2. Project Requirements:

The requirements are initially given in the natural language and then they are transformed into a format such that each requirement has pre and post conditions.

2.1 Functional Requirements

2.1.1 Data Extraction: The system shall be able to fetch crypto currency data from the CoinCap API.
Pre-condition: The system shall have a stable internet connection.
Post-Condition: The API data is successfully fetched and ready for further processing.

2.1.2 Data Transformation: The system shall be able to process the extracted data by formatting and structuring it appropriately as per csv format.
Pre-condition: The system has successfully extracted the data from the CoinCap API.
Post-Condition: The data is successfully transformed into a structured format for storage.

2.1.3 Data Loading: The system shall be able to Store the transformed data into a MariaDB database.
Pre-condition: Functional requirements 2.1.1 and 2.1.2 are successfully completed.

Post-Condition: The transformed data is loaded into the MariaDB database and ready to can be viewed in the database.

2.2 Non-Functional Requirements

2.2.1 Performance: The ETL process should be optimized for efficiency and speed.
Pre-condition: The system shall have minimum 2GB RAM, I-3 processor, and 250GB storage available for ETL operations.
Post-condition: The ETL process should be completed within less than 1 minute.

2.2.2 Reliability: The pipeline should handle errors and provide appropriate logging.
Pre-Condition: The system should have expectation handling mechanism.
Post-Condition: If error occurs system shall be able to logs relevant information, and continues or halts processing as appropriate.

2.2.3 Scalability: The solution should be scalable to accommodate increasing data volumes.
Pre-Condition: The system shall be able to handle upto 5000 rows of data.
Post-condition: As the volume of crypto currency data increases, the system shall scales proportionally to maintain performance and reliability.

3. Technology Stack:

3.1 Programming Languages: Python

- The project is primarily implemented using Python, a versatile and widely-used programming language. Python was chosen for its simplicity, readability, and libraries that facilitate rapid development.

3.2 Libraries and Frameworks: Requests (for API communication), MySQL Connector

- The Requests library is imported for making HTTP requests to the CoinCap API. It simplifies the process of sending HTTP requests and handling responses, making API communication seamless.
- MySQL Connector ensures secure and reliable communication between the Python script and the MariaDB database, facilitating the storage of cryptocurrency data.

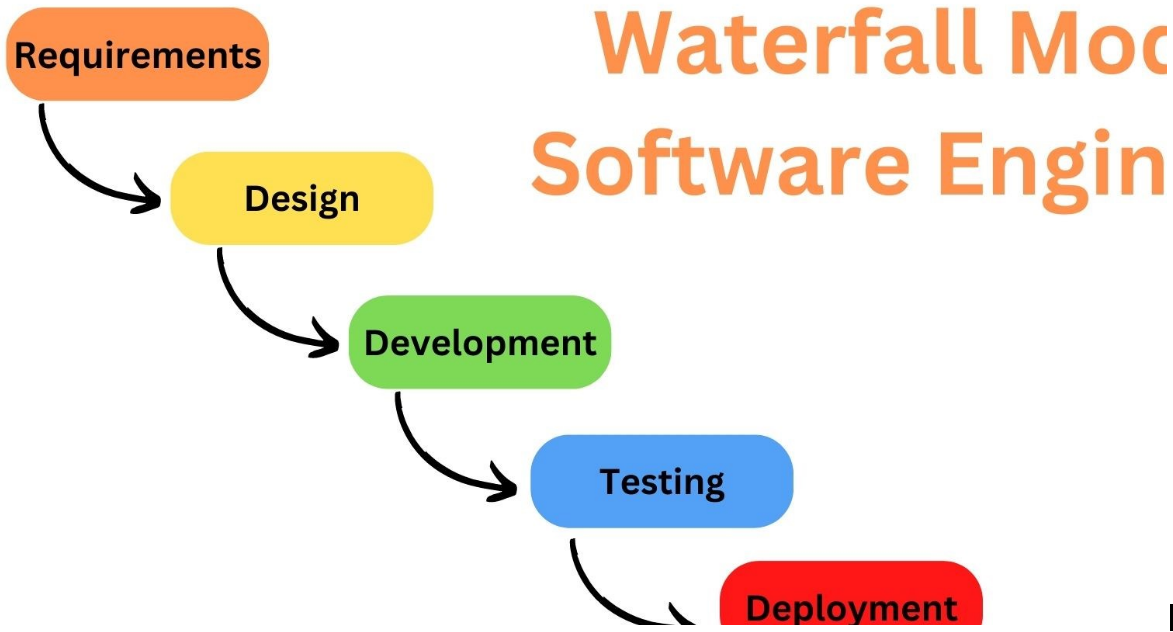
3.3 Database: MariaDB

- MariaDB, serves as the relational database management system for this project.

- MariaDB's robustness, scalability, and compatibility with MySQL make it an ideal choice for storing structured data. It provides a stable foundation for the ETL process.

4. Software Methodology:

4.1 Waterfall Model: This project follows the Waterfall model for software development. The Waterfall model provides a structured approach, moving through phases such as requirements analysis, design, implementation, testing, deployment, and maintenance in a linear fashion. This methodology is suitable for projects with well-defined and stable requirements, making it an appropriate choice for this ETL pipeline implementation.



[1]

4.1.1 Requirements Analysis: In this project, the functional and non-functional requirements are defined in the beginning, specifying what the system should do such as extract, transform, load data and the qualities it should have i.e. performance, reliability, scalability.

4.1.2 Design: It includes the design of a CryptoDataProcessor class that encapsulates the logic for fetching data from an API, transforming it, and storing it in a MariaDB database. The class design follows the principles of encapsulation, as it groups related functionalities together.

4.1.3 Implementation: The coding of the ETL pipeline involves implementing the designed solution. I have used libraries like requests for API communication, csv for CSV file operations, and mysql.connector for database interactions.

4.1.4 Error Handling: This code includes the error handling mechanisms, such as catching exceptions during database operations.

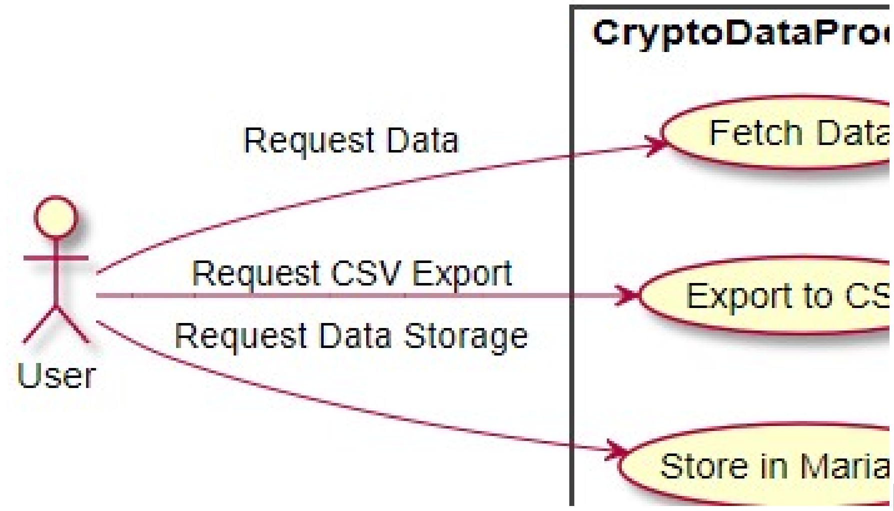
4.1.5 Testing: The unittest framework is implemented to ensure that individual components of the system work as expected.

4.1.6 Verification and Validation: It includes methods like check_csv_exists, check_mariadb_table_exists, check_data_in_mariadb for verifying that the data is correctly processed, stored, and can be checked against the API.

4.1.7 Maintenance: It includes maintenance of ETL pipeline by regularly tracking updates in the system.

5. UML Diagram:

5.1 Use-case Diagram:



[2]

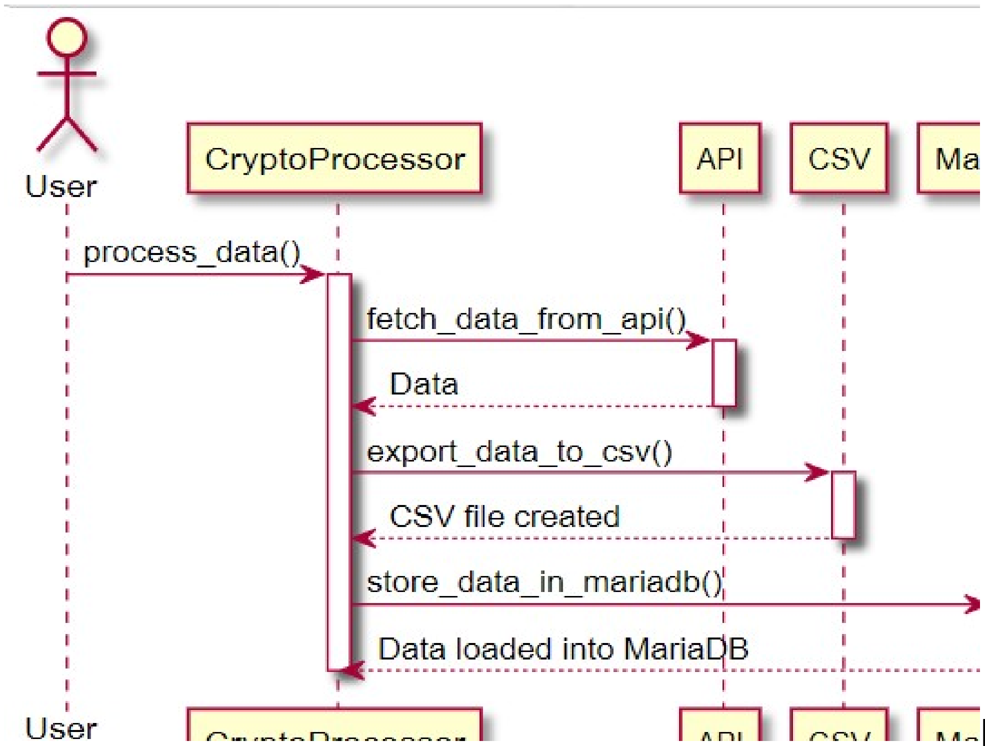
5.1.1 Use-Case Description:

5.1.1.1 Usecase: The user shall request the system to fetch crypto currency data from the CoinCap API.
Pre-condition: The system shall have a stable internet connection.
Post-Condition: The API data is successfully fetched and ready for further processing.

5.1.1.2 Usecase: The user shall request the system to process the extracted data by formatting and structuring it appropriately as per csv format.
Pre-condition: The system has successfully extracted the data from the CoinCap API.
Post-Condition: The data is successfully transformed into a structured format for storage.

5.1.1.3 Usecase: The user shall be able request the system to Store the transformed data into a MariaDB database.
Pre-condition: Usecase 5.1.1.1 and 5.1.1.2 are successfully completed.
Post-Condition: The transformed data is loaded into the MariaDB database and ready to can be viewed in the database.

5.2 Sequence Diagram:



[2]

6. Verification Method:

6.1 Unit Testing: Unit test is implemented to verify the functionalities of the **CryptoDataProcessor** class. There are 3 methods in the main class which are imported in unitTest framework.

6.2 Verification Criteria:

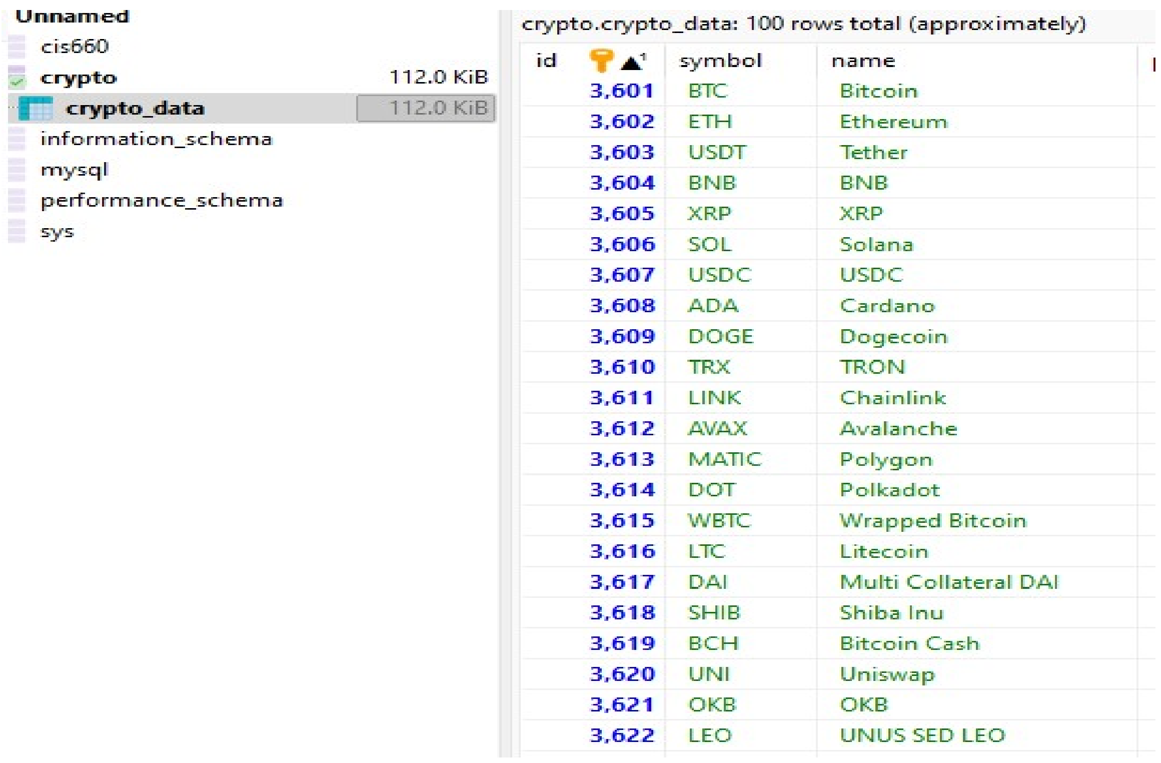
- All methods in the **CryptoDataProcessor** class must pass their respective unit tests.
- Mock objects should simulate different scenarios, including successful API responses, network errors, and database interactions.

6.3 Acceptance Criteria

- The ETL pipeline is considered successful if it meets the following acceptance criteria:
    - Successfully fetches cryptocurrency data from the API.
    - Transforms and formats the data correctly.
    - Loads the data into the MariaDB database without errors.
    - Meets performance requirements under varying data volumes.

7. **Results**:

7.1 Data in MariaDB: The CoinCap API data has been successfully fetched in the mariaDB database.



```
Unnamed
  cis660
  crypto                      112.0 KiB
    crypto_data               112.0 KiB
  information_schema
  mysql
  performance_schema
  sys
```

crypto.crypto_data: 100 rows total (approximately)

id	symbol	name
3,601	BTC	Bitcoin
3,602	ETH	Ethereum
3,603	USDT	Tether
3,604	BNB	BNB
3,605	XRP	XRP
3,606	SOL	Solana
3,607	USDC	USDC
3,608	ADA	Cardano
3,609	DOGE	Dogecoin
3,610	TRX	TRON
3,611	LINK	Chainlink
3,612	AVAX	Avalanche
3,613	MATIC	Polygon
3,614	DOT	Polkadot
3,615	WBTC	Wrapped Bitcoin
3,616	LTC	Litecoin
3,617	DAI	Multi Collateral DAI
3,618	SHIB	Shiba Inu
3,619	BCH	Bitcoin Cash
3,620	UNI	Uniswap
3,621	OKB	OKB
3,622	LEO	UNUS SED LEO

8. Conclusion:

In conclusion, this project has successfully designed and implemented the ETL pipeline for retrieving data from a crypto currency API, transforming the data, and loading it into both a CSV file and a MariaDB database.

References:

1. A *Quick Guide to Waterfall Model in Software Engineering - Board Infinity*, waterfall model image
2. *GitUML - UML visualization for Git repositories*
3. *Free Public APIs for Developers*