Fair Online Judge

Звіт №4 "Project architecture"

Розробники:
Пилипець Гліб
Геворгян Артем

Зміст

1. Вступ

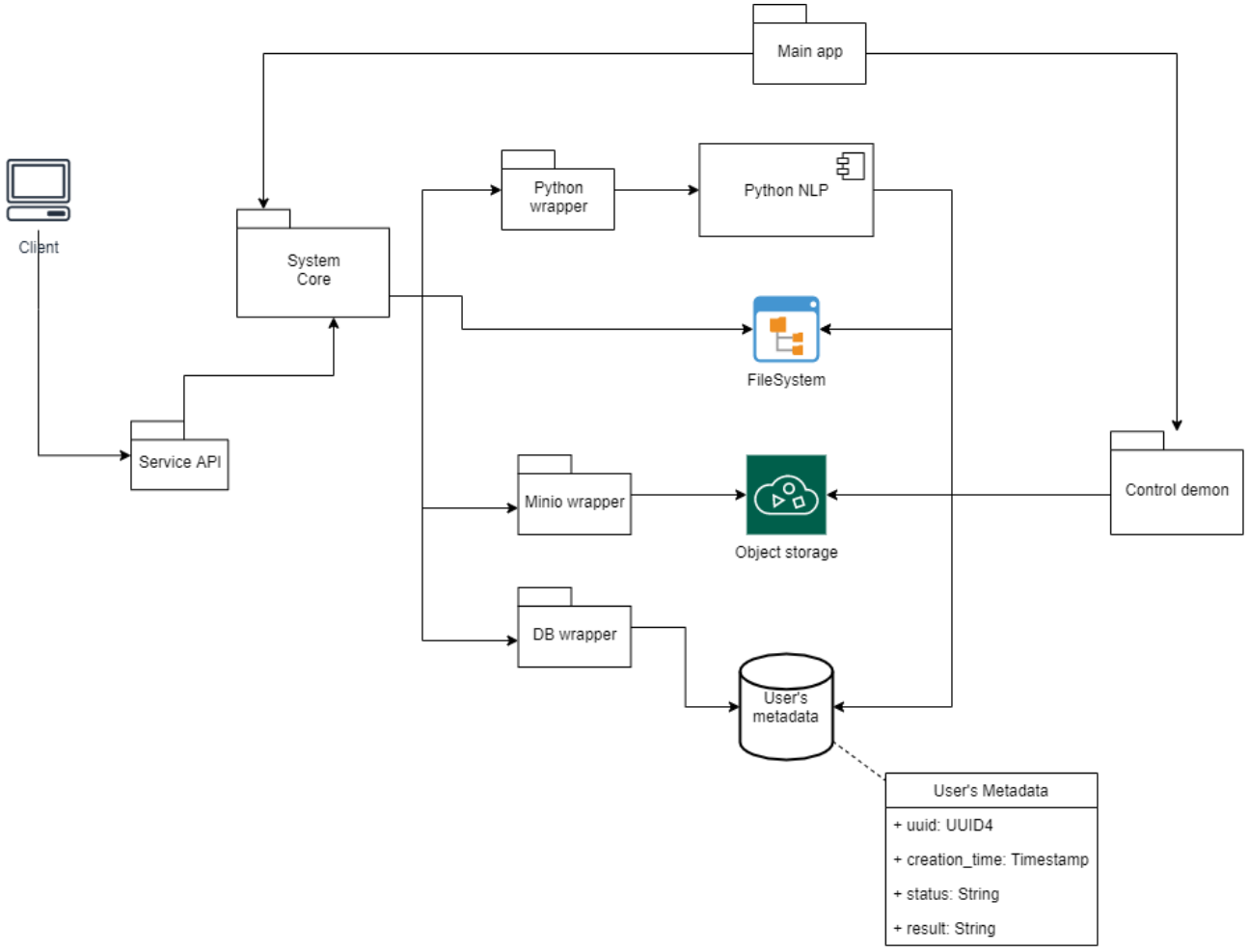
Цей звіт є описом архітектури системи Smart Online Judge та основних потоків даних в ній. В ньому описані модульна декомпозиція системи, комунікація інтерфейсів, data-flow діаграми, специфікація API.

Для розробки була вибрана багаторівнева архітектура, що дозволяє зробити роботу над компонентами та рівнями незалежною, спираючись на визначені інтерфейси з ефективним використанням інкрементної моделі розробки.

Умовно систему можна поділити на наступні компоненти:

- Веб-сервіс, що написаний на Golang.
- NLP частина для роботи з текстами, що написана на Python.
- Object storage, що є окремим мікро-сервісом та працює по S3 протоколу.
- FileSystem, Database.

2. Модульна декомпозиція

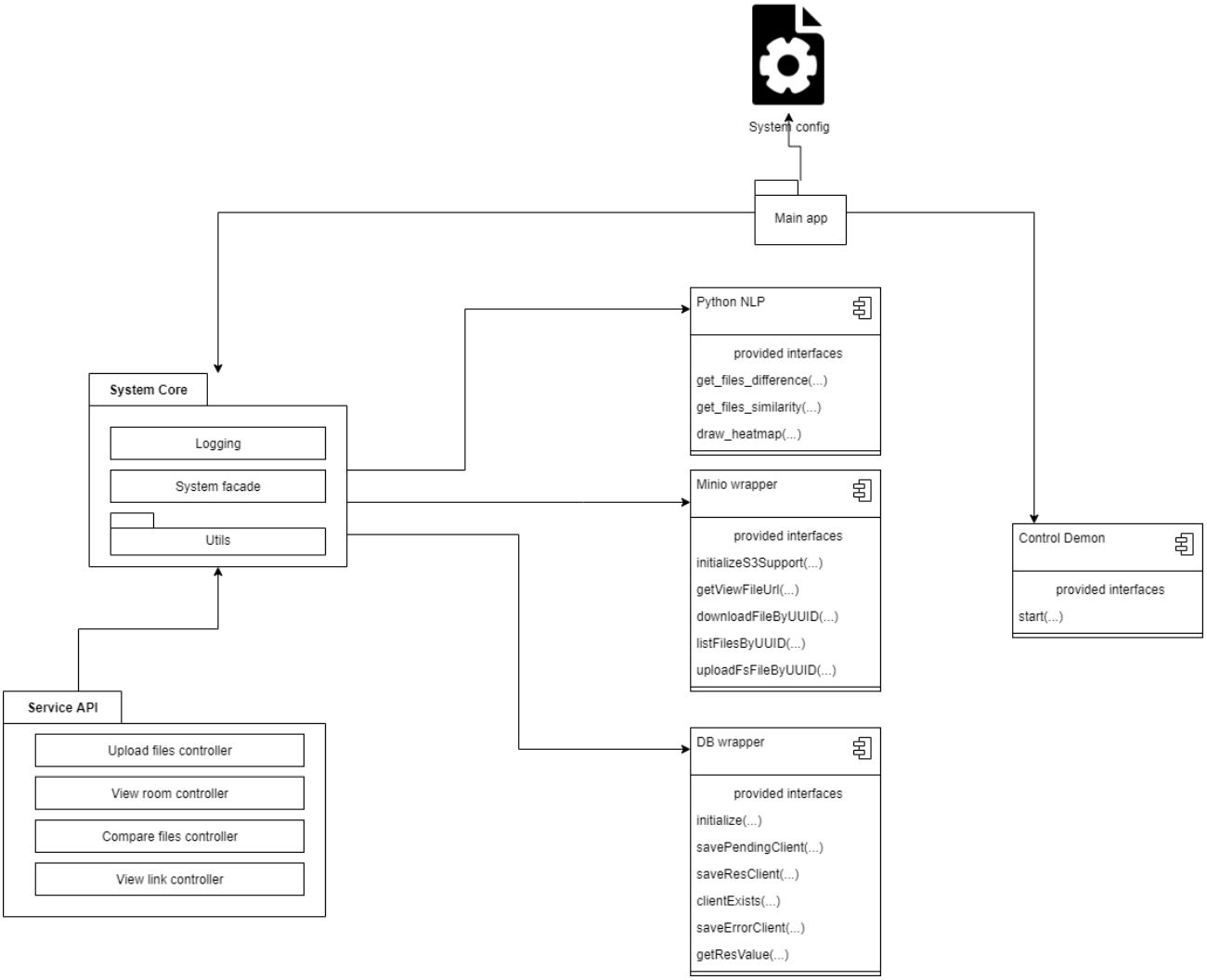


Особливості деяких модулів:

1. Minio – для збереження файлів користувача використовується сервер хмарного збереження, що є сумісним з Amazon S3.
2. SQLite DB – для зберігання результату подібності текстових документів та метаданих користувача.

3. FileSystem – для тимчасовового збереження файлів користувача, що використовується в перед та пост обробках.
4. Python NLP – компонент, що використовує Python код та надає інтерфейси по порівнянню двох, подібності групи текстових документів, відображення подібності у вигляді heatmap.
5. Control demon – демон, що працює як фоновий процес - використовуючи SQLite database, видаляє файли користувачів на Minio, FileSystem, SQLite, що знаходяться на сервері довше встановленого часового проміжку.

3. Комунікація інтерфейсів



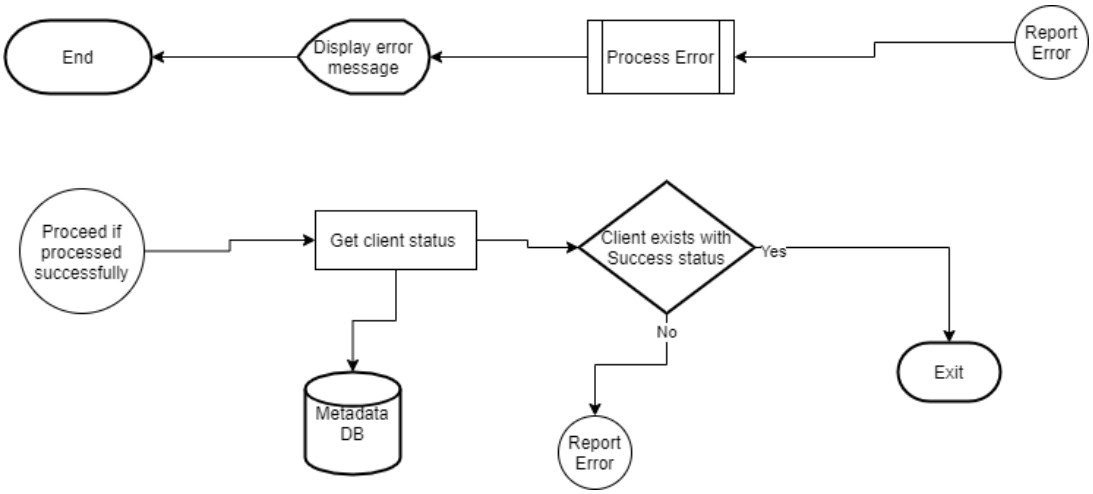
Особливості деяких інтерфейсів:

1. Service API – надає клієнту API по збереженню(upload_files), порівнянню(cmp_files), перегляду результатів подібності(view), завантаженню/перегляду(link) файлів – Golang.
2. Python NLP – компонент, що відповідає за обробку чи відображення текстових документів на мові програмування Python, використовує файлову систему для обробку групи текстових документів та OS unnamed pipe для порівняння текстових документів:
    - Синтаксична подібність групи текстових документів алгоритмом cosine similarity з використанням Python пакету nltk;
    - Семантична подібність групи текстових документів алгоритмом soft-cosine similarity з використанням GloVe моделі для NLP та Python пакету gensim;

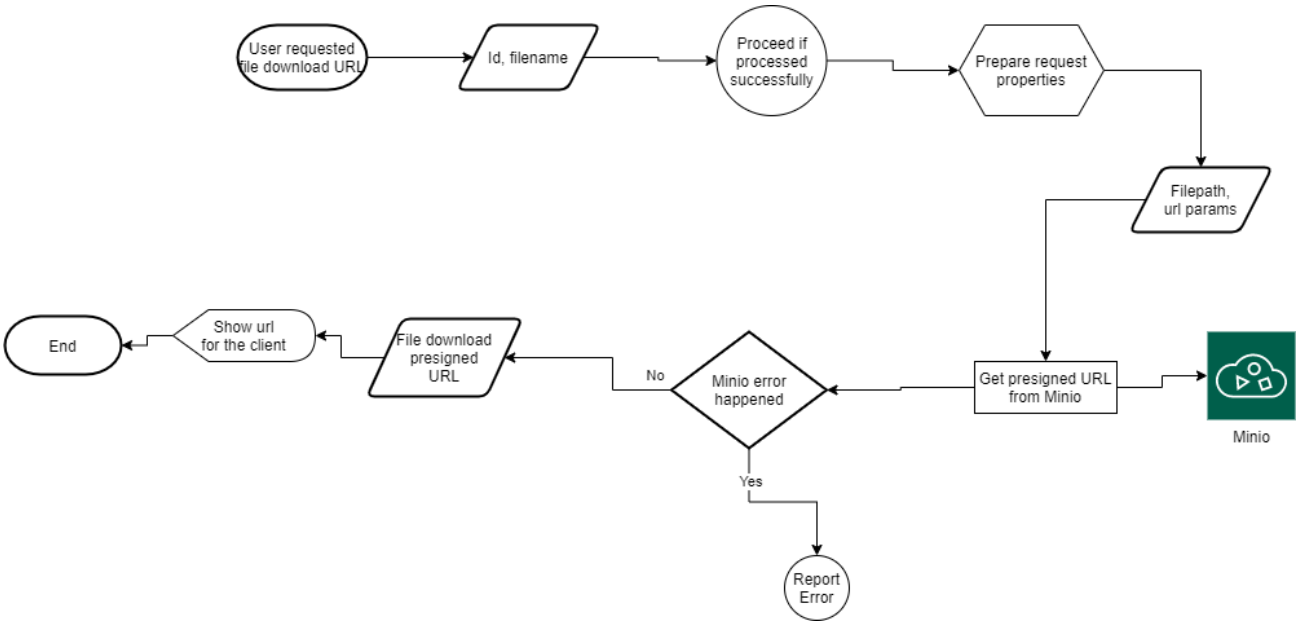
- Порівняння двох текстових документів алгоритмом Маєрса з використанням Python пакету Google diff-match-patch (https://github.com/google/diff-match-patch).
- Відображення подібності групи текстових документів на 2Д зображення у формі heatmap з використанням Python matplotlib.pyplot, seaborn пакетів;

4. Minio wrapper – компонент, що містить connection до minio server та зберігає файли користувача у випадку успішного підрахунку подібності (https://docs.min.io/docs/golang-client-api-reference.html)

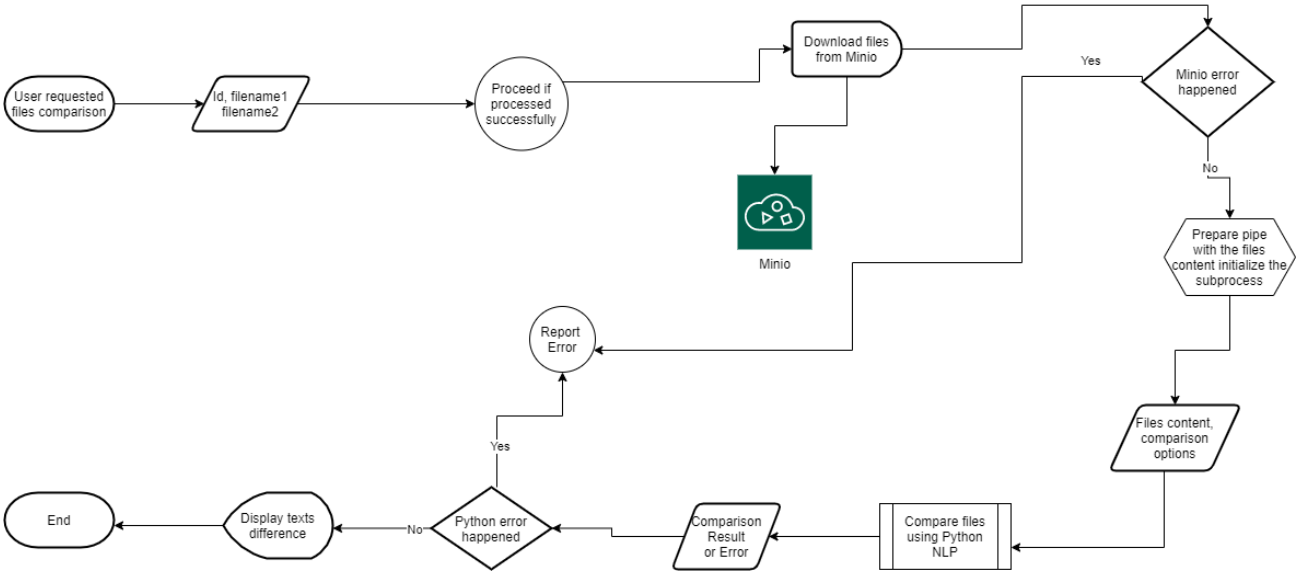
5. Flowchart діаграми



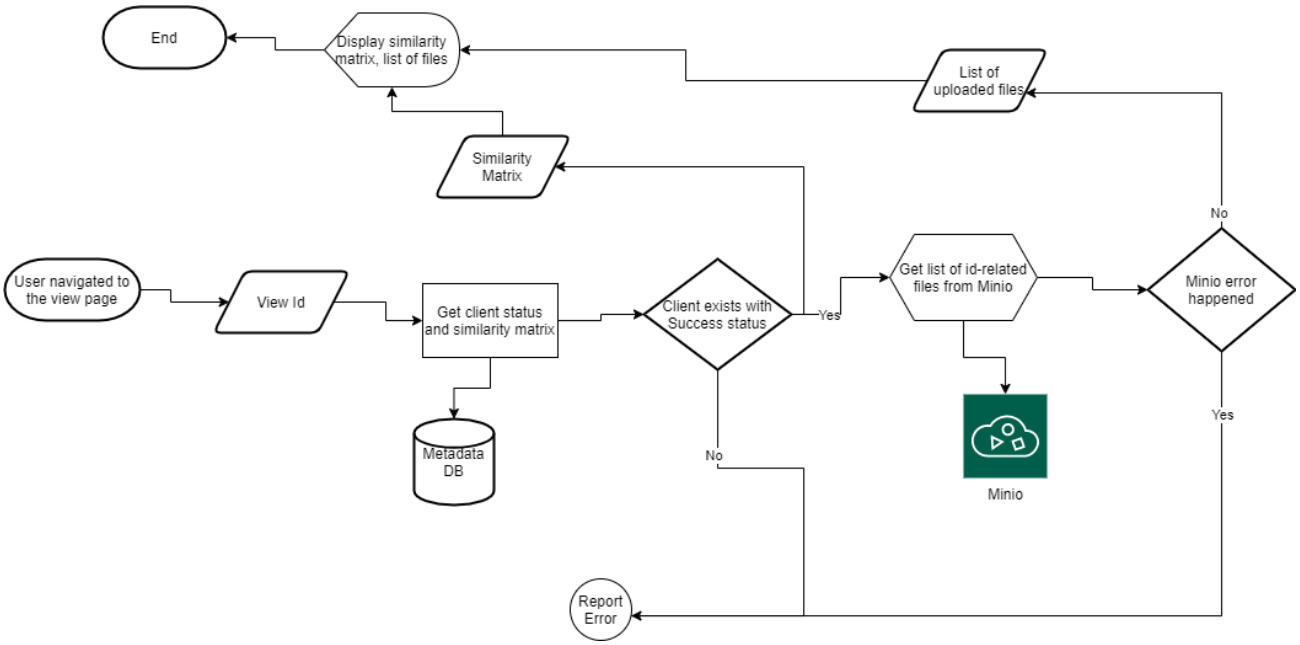
Запит на посилання для завантаження вибраного файлу:



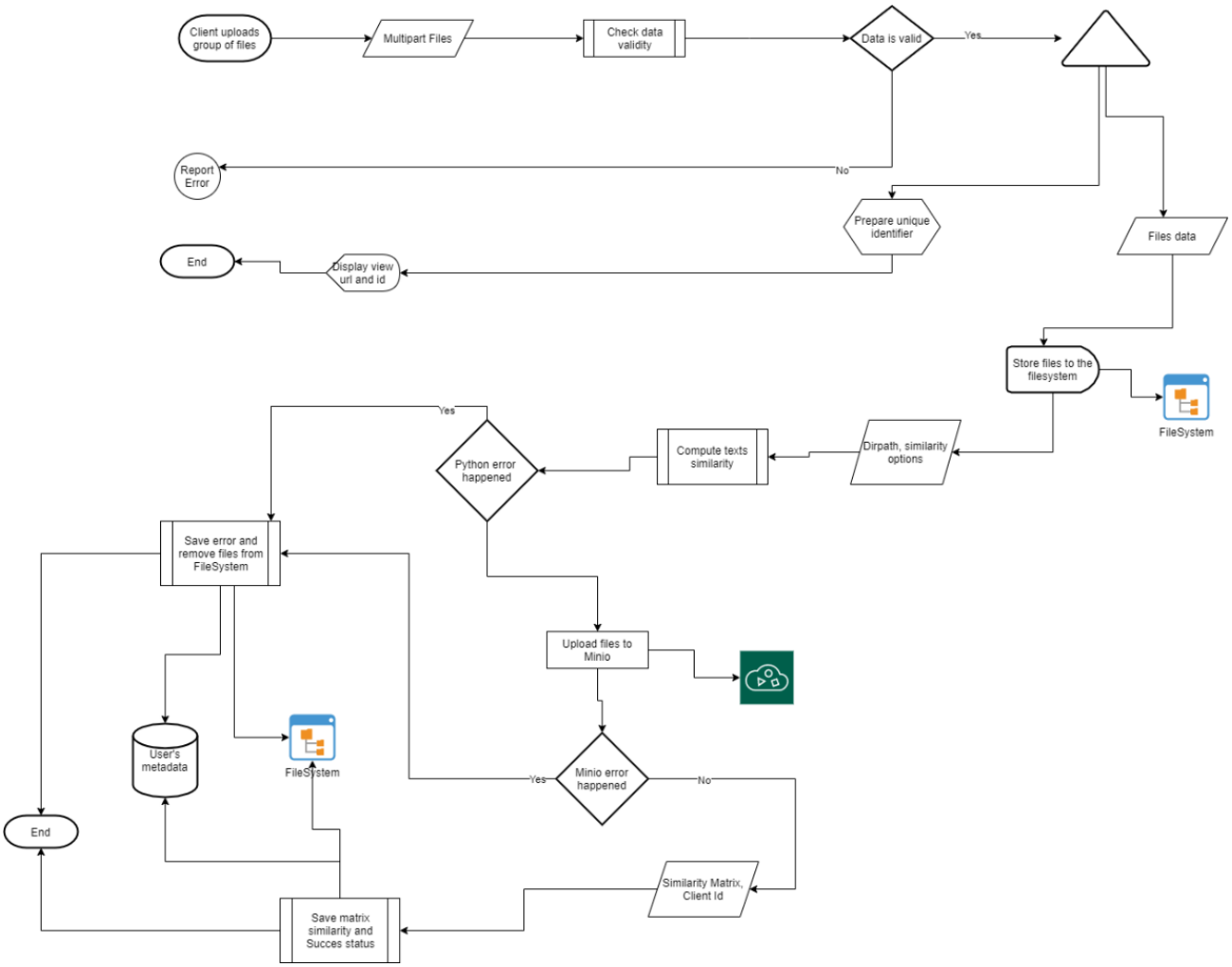
Запит на порівняння двох файлів:



Запит на перегляд подібності групи файлів:



Запит на завантаження групи файлів на обробку:



6. Специфікація API

№	HTTP method type	API endpoint	Params Body	Response JSON у випадку успіху
1	POST	/api/upload_files	Body: Multipart.file, text/plain	Унікальний id користувача для виконання операцій 2-4.
2	GET	/api/view/{id}		Матриця подібності, імена файлів, завантажених користувачем.
3	GET	/api/link?	id=UUID4&name=str	Посилання на завантаження файлу для id та імені файлу з аргументів URL.
4	GET	/api/cmp_files?	id=UUID4&f1=str&f2=string [&editcost=int&timeout=int &option=str&html=bool]	Результат порівняння двох файлів у форматі HTML для переданого id з іменами f1, f2. Додатковими параметрами є максимальний час обробки різниці timeout, ступінь групування editcost, html чи raw дані – відповідно флаг hml.