

TextMatch – Dokumentacja projektu

1. Wprowadzenie

TextMatch to pełnostackowa aplikacja webowa służąca do porównywania plików tekstowych i zarządzania historią takich porównań.

Projekt składa się z:

- Backend: Python 3.x + Flask
- Frontend: React 18 + Vite
- Stylowanie: Tailwind CSS, shadcn/ui, lucide-react

2. Stos technologiczny

Warstwa	Język / Framework	Kluczowe biblioteki
Backend	Python 3.x + Flask	flask-cors, flask-mysqldb, PyJWT, APScheduler
Baza DB	MySQL 8	–
Frontend	React 18 + Vite	TailwindCSS, shadcn/ui, lucide-react, diff2html
Dev Tools	ESLint, Prettier, Vite, npm	–

3. Struktura katalogów (high level)

```
├── backend
|   ├── app.py      # Główny plik aplikacji Flask
|   ├── certs/      # Klucze i certyfikaty TLS
|   ├── uploads/    # Przesłane pliki użytkowników
|   └── app.log      # Logi serwera (poziom WARNING+)
└── frontend
    └── file-comparator-fe/
        ├── public/
        └── src/
```

```
| |— assets/
| |— components/
| | |— lib/utils.js
| | |— ui/... (Button, Card, Input, Label, Sheet).jsx
| |— App.jsx # router + logika auth
| |— api.js # REST client
| |— main.jsx
|— tailwind.config.js
|— vite.config.js
|— ...inne pliki konfiguracyjne
```

4. Konfiguracja środowiska

4.1 Wymagania wstępne

- Python ≥ 3.10
- Node ≥ 18
- MySQL 8 (lub kompatybilny)

4.2 Zmienne środowiskowe

Zmienna	Opis
MYSQL_HOST	Adres serwera MySQL
MYSQL_PORT	Port MySQL (domyślnie 3306)
MYSQL_USER	Użytkownik bazy
MYSQL_PASSWORD	Hasło do bazy
MYSQL_DB	Nazwa bazy
SECRET_KEY	Klucz JWT (Flask app.config['SECRET_KEY'])
UPLOAD_FOLDER	Katalog na pliki (domyślnie uploads/)
DB_BACKUP_DIR	Katalog na backupy (domyślnie C:/.../db_backup/)

Zmienna	Opis
FLASK_SSL_CERT	Ścieżka do certyfikatu TLS
FLASK_SSL_KEY	Ścieżka do klucza TLS

4.3 Instalacja zależności

Backend

```
cd backend
```

```
python -m venv .venv && source .venv/bin/activate
```

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

Frontend

```
cd ../frontend/file-comparator-fe
```

```
npm install
```

5. Uruchomienie w trybie developerskim

5.1 Backend (HTTPS + autoreload)

```
cd backend
```

```
python app.py
```

Serwer dostępny domyślnie pod https://localhost:5000

5.2 Frontend (Vite dev server)

```
cd ../frontend/file-comparator-fe
```

```
npm run dev
```

Domyślnie http://localhost:5173

6. Backend — API

6.1 Uwierzytelnienie

- Rejestracja POST /api/register
 - JSON: { username, email, password }

- Haszowanie hasła: `werkzeug.security.generate_password_hash`
 - Błędy: 400 (brak danych), 400 (duplikat)
- Logowanie POST `/api/login`
 - JSON: `{ username, password }`
 - Generowanie JWT (HS256), ważnego 1 h:

```
jwt.encode({
    'user_id': user['id'],
    'exp': datetime.utcnow() + timedelta(hours=1)
}, SECRET_KEY, algorithm='HS256')
```

- Błędy: 400 (brak danych), 401 (niepoprawne dane)
- Dekorator `@token_required`
 - Wymaga nagłówka `Authorization: <token>`
 - Dekoduje JWT, ustawia `g.user_id`
 - Obsługuje `ExpiredSignatureError (401)`, `InvalidTokenError (401)`

6.2 Operacje na plikach

- Dozwolone rozszerzenia: `.txt`, `.py`, `.md`
- Czyszczenie nazwy (usuwa prefiksy numeryczne):

```
_RE_PREFIX = re.compile(r'^[0-9]+_[0-9]+\.(?:[0-9]+)?_')
```

- POST `/api/compare`
 - Wymaga dwóch plików `file1`, `file2` (multipart)
 - Czyta jako UTF-8, tworzy unified diff przy pomocy `difflib.unified_diff`
 - Zwraca JSON `{ filename1, filename2, diff }`
 - Kody: 200, 400 (brak plików albo błąd odczytu), 415 (nieobsługiwane rozszerzenie)

- POST /api/save-comparison
 - JSON { filename1, filename2, diff }
 - Zapisuje do tabeli file_comparisons (user_id, created_at = NOW())
 - Kody: 201 (sukces), 400 (brak danych)
- DELETE /api/comparison/<id>
 - Weryfikuje właściciela porównania
 - Kody: 200 (usunięto), 404 (nie znaleziono / brak dostępu)
- GET /api/my-comparisons
 - Zwraca listę porównań zalogowanego użytkownika (id, filename1, filename2, diff, created_at)
 - Posortowane malejąco wg created_at

6.3 Diagnostyka i dostęp

- GET /api/test-db
 - Kwerenda SELECT 1 → 200 lub 500
- Serwowanie plików GET /uploads/<filename>
 - send_from_directory(app.config['UPLOAD_FOLDER'], filename)

6.4 Kopie zapasowe bazy

- Funkcja backup_database() wywołuje mysqldump, tworzy plik backup_YYYY-MM-DD_HH-MM-SS.sql w DB_BACKUP_DIR.
- Harmonogram: APScheduler, zadanie interval(hours=24).

7. Frontend — React + Vite

7.1 Podstawowe skrypty

Komenda	Opis
npm run dev	Dev server z hot reload
npm run build	Produkcyjny build do dist/
npm run preview	Podgląd build-u

7.2 Główne pliki

- `src/App.jsx` – główny routing + logika uwierzytelniania
- `src/api.js` – centralny klient HTTP (Axios/fetch)
- `src/components/ui/` – komponenty UI oparte na `shadcn/ui` i `lucide-react`
- `diff2html` – do renderowania różnic po stronie klienta

7.3 Stylowanie

- Tailwind CSS – system siatki, util-classy, klasa `max-w-5xl`
- `shadcn/ui` – zestaw gotowych komponentów (Card, Button, Input)