**1. Wprowadzenie**

Celem projektu jest stworzenie prostego serwera WWW w Node.js, który obsługuje pliki statyczne, umożliwia ich przesyłanie i udostępnia listę dostępnych plików w katalogu. Serwer jest w stanie obsłużyć wiele jednoczesnych połączeń dzięki wykorzystaniu socket.io. Klienci komunikują się z serwerem przez przeglądarkę internetową. Dodatkowo projekt zawiera prosty frontend z formularzem przesyłania plików i listą dostępnych zasobów.

**2. Funkcjonalności**

1. **Serwowanie plików statycznych**:
   * Użytkownik może przeglądać i pobierać pliki statyczne z katalogu serwera.
2. **Przesyłanie plików**:
   * Użytkownik może przesłać pliki z lokalnego urządzenia do serwera za pomocą formularza.
3. **Lista dostępnych plików**:
   * Serwer udostępnia dynamicznie generowaną listę dostępnych plików w katalogu publicznym.
4. **Obsługa błędów**:
   * Informowanie użytkownika w przypadku, gdy plik nie istnieje lub wystąpi problem z przesyłaniem.

**3. Struktura projektu**

- public/

- index.html // Plik frontendowy z interfejsem użytkownika

- client.js // Logika frontendowa

- styles.css // Stylizacja strony

- uploads/ // Folder na przesłane pliki

- server.js // Główny plik serwera

**4. Instalacja i uruchomienie**

**4.1. Wymagania**

* Node.js (zalecana wersja: 16+)
* npm (Node Package Manager)

**4.2. Kroki instalacji**

1. **Sklonowanie projektu**:

git clone https://github.com/lawerski/sieci\_projekt

cd sieci\_projekt

1. **Instalacja zależności**:

npm install

1. **Uruchomienie serwera**:

node server.js

1. **Dostęp do aplikacji**: Otwórz przeglądarkę i przejdź pod adres: http://localhost:3000.

**5. Plik serwera (server.js)**

**5.1. Importy modułów**

**Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie**

* **express**: Framework do obsługi żądań HTTP.
* **http**: Moduł do uruchomienia serwera HTTP.
* **socket.io**: Biblioteka do obsługi komunikacji w czasie rzeczywistym przez WebSocket.
* **fs**: Moduł do obsługi plików i systemu plików.
* **path**: Narzędzie do obsługi ścieżek plików.
* **multer**: Middleware do obsługi przesyłania plików.

**5.2. Konfiguracja Multer**Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

* destination: Określa folder, do którego zapisywane są przesłane pliki.
* filename: Zachowuje oryginalną nazwę przesyłanego pliku.

**5.3. Główne endpointy**

**5.3.1. Endpoint do przesyłania plików**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie**

* Obsługuje przesyłanie pojedynczych plików za pomocą multipart/form-data.
* Zwraca odpowiedź JSON z informacją o powodzeniu operacji.

**5.3.2. Endpoint do uzyskania listy plików**

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie**

* Rekurencyjnie przeszukuje katalog public i zwraca listę plików.
* Zwraca odpowiedź w formacie JSON zawierającą tablicę nazw plików.

**5.3.3. Obsługa połączenia WebSocket**

**Obraz zawierający tekst, zrzut ekranu, wyświetlacz, oprogramowanie

Opis wygenerowany automatycznie**

* Obsługuje zapytania o zawartość plików za pomocą socket.io.
* Wysyła odpowiedzi do klientów w czasie rzeczywistym.

**6. Frontend**

**6.1. Plik index.html**

Zawiera prosty interfejs z:

1. Formularzem do przesyłania plików.
2. Przycisk do załadowania listy dostępnych plików.
3. Listę wyświetlającą dostępne pliki.

**6.2. Plik client.js**

Obsługuje logikę:

* Przesyłania plików na serwer.
* Pobierania i wyświetlania listy plików.

**6.3. Plik styles.css**

Odpowiada za stylizację strony, zapewniając:

1. Proste i przejrzyste tło.
2. Wyraźne przyciski i pola formularza.
3. Estetyczną prezentację listy plików.

**7. Przykładowe scenariusze użycia**

1. Użytkownik otwiera przeglądarkę i przechodzi pod adres http://localhost:3000.
2. W sekcji „Przesyłanie plików” wybiera plik i klika „Wyślij plik”.
3. W sekcji „Dostępne pliki” klika przycisk „Załaduj dostępne pliki”, aby zobaczyć przesłane pliki.
4. Klikając na link „Pobierz”, pobiera plik na swoje urządzenie.

**8. Możliwe rozszerzenia**

1. Obsługa autoryzacji użytkowników.
2. Wsparcie dla większych plików (np. za pomocą streamingu).
3. Umożliwienie usuwania plików przez użytkowników.