**WebScraper**

**Overview**

WebScraper is a **full-stack** project designed to analyze a user-supplied URL, extract details from the web page, and return structured data. The extracted details include:

* The count and total byte size of each type of image (e.g., .jpg, .png, etc.).
* All nested links categorized into **internal** and **external** links.
* The ability for users to request details for extracted links.

This project follows a **Clean Architecture** approach on the backend and a **modern Next.js frontend** ensuring modularity, maintainability, and separation of concerns.

**Technologies Used**

**Backend:**

* **Framework:** ASP.NET Core Web API
* **Architecture:** Clean Architecture
* **Language:** C#
* **Libraries:**
  + HtmlAgilityPack for parsing HTML.
  + Moq and xUnit for unit testing.
  + HttpClient for making HTTP requests.

**Frontend:**

* **Framework:** Next.js (App Router)
* **Styling:** Tailwind CSS
* **State Management:** React Hooks (useState)
* **Networking:** Fetch API

**Project Structure**

**Backend (ASP.NET Core - Clean Architecture)**

WebScraper/

│── src/

│ ├── WebScraper.API/ # API Layer - Handles HTTP requests/responses

│ │ ├── Controllers/ # Web API Controllers

│ │ │ ├── WebScraperController.cs

│ │ ├── Program.cs # Configures DI and services

│ │ ├── appsettings.json

│ │

│ ├── WebScraper.Application/ # Business Logic Layer

│ │ ├── Interfaces/ # Interfaces for services

│ │ │ ├── IWebScraperService.cs

│ │ ├── Services/ # Implementation of business logic

│ │ │ ├── WebScraperService.cs

│ │ ├── DTOs/ # Data Transfer Objects

│ │

│ ├── WebScraper.Domain/ # Core Entities & Business Rules

│ │ ├── Entities/

│ │ │ ├── ImageInfo.cs # Represents an image and its properties

│ │ │ ├── LinkData.cs # Represents extracted links

│ │

│ ├── WebScraper.Infrastructure/ # External dependencies (optional for DB/storage)

│

│── tests/ # Unit Tests

│ ├── WebScraper.Tests/

│ │ ├── WebScraperServiceTests.cs

│

│── WebScraper.sln # Solution File

**Frontend (Next.js - App Router & Tailwind)**

frontend/

│── src/

│ ├── app/ # Next.js App Router

│ │ ├── page.tsx # Main Scraping Page

│ │ ├── globals.css # Global Styles

│ │

│ ├── components/ # Reusable Components

│ │ ├── ScrapePage.tsx # UI for scraping and displaying results

│

│── package.json

│── tailwind.config.ts # Tailwind Configuration

│── README.md

**Features & Functionality**

**1. Extracting Images with Size**

* The API extracts all images from a given webpage.
* It retrieves **the file extension** (e.g., .jpg, .png) and **calculates the total file size**.
* Uses HttpClient to fetch image sizes asynchronously.

**2. Extracting Internal & External Links**

* Internal links: Links that belong to the same domain as the provided URL.
* External links: Links pointing to other domains.
* All extracted links are cleaned and standardized.

**3. Ability to Recurse into Extracted Links**

* Users can click on extracted links and submit them for further analysis.

**4. Responsive Frontend**

* Uses **Tailwind CSS** for a modern, mobile-friendly design.
* **Overflow handling** for long links with scroll support.

**How to Run the Project**

**Backend (ASP.NET Core API)**

1. **Install .NET SDK**
   * Ensure that you have .NET 8.0 installed on your system.
   * You can check your installation by running:
   * dotnet --version
2. **Navigate to the Project Directory**
3. cd WebScraper
4. **Restore Dependencies**
5. dotnet restore
6. **Run the API**
7. dotnet run --project src/WebScraper.API
   * The API will be available at: https://localhost:7215/swagger/index.html

**Frontend (Next.js App)**

1. **Navigate to the frontend directory**
2. cd frontend
3. **Install dependencies**
4. npm install
5. **Run the development server**
6. npm run dev
   * The frontend will be available at: http://localhost:3000/

**API Endpoints**

**Extract Data from a Web Page**

* **Endpoint:** POST /api/WebScraper/scrape
* **Request Body:**

{

"url": "https://example.com"

}

* **Response:**

{

"Url": "https://example.com",

"Images": [

{ "Url": "https://example.com/image1.jpg", "Extension": ".jpg", "Size": 12345 }

],

"Links": {

"Internal": ["https://example.com/about"],

"External": ["https://external.com"]

}

}

**Testing the Project**

**Backend Unit Tests**

This project includes **unit tests** using xUnit and Moq.

**Run Tests**

dotnet test

**Frontend Testing**

* Ensure UI elements render properly in **light/dark mode**.
* Test long URLs and link scrolling behavior.

**Why This Architecture?**

This project follows **Clean Architecture** to ensure: ✅ **Separation of concerns** (API, Business Logic, Domain, Infrastructure).  
✅ **Testability** - Business logic is decoupled from external dependencies.  
✅ **Scalability** - Easy to extend by adding databases, caching, or additional processing.  
✅ **Modern UI** - Next.js + Tailwind for a smooth frontend experience.

**Next Steps & Improvements**

* ✅ Add caching for repeated requests.
* ✅ Optimize performance for large webpages.
* ✅ Implement logging for better debugging.

**Author**

Created as part of a coding challenge. If you have any questions, feel free to reach out!