Problem Statement

 # Test task: Web application for analyzing web pages

Objective

The objective is to build a web application that does an analysis of a web-page/URL.

The application should show a form with a text field in which users can type in the URL of the web page to be analyzed. Additionally, to the form, it should contain a button to send a request to the server.

After processing, the results should be shown to the user.

Results should contain next information:

- What HTML version has the document?
- What is the page title?
- How many headings of what level are in the document?
- How many internal and external links are in the document? Are there any inaccessible links and how many?

- Does the page contain a login form?

In case the URL given by the user is not reachable an error message should be presented to a user. The message should contain the HTTP status code and a useful error description.

Restrictions

1. The application should be written in Golang
2. The application must be put under git control
3. You can use whatever libraries/tools you want.

Submission

Please provide the result as a git repo bundled with:

- A short text document that lists the main steps of building/deploying your solution as well as all assumptions/decisions you made in case of unclear requirements or missing information
- Suggestions on possible improvements of the application

Solution : Web-Page-Analyzer

Before You Read

- This is an MVP version of app for the problem statement
- Uses GIN framework, with Golang 1.20.3 (this is due to my system limitation)
- Unit tests are added for the service layer and core engine only (due to time limitations)
- No DI framework been used (just everything in plain go for now)
- Api versioning is not done

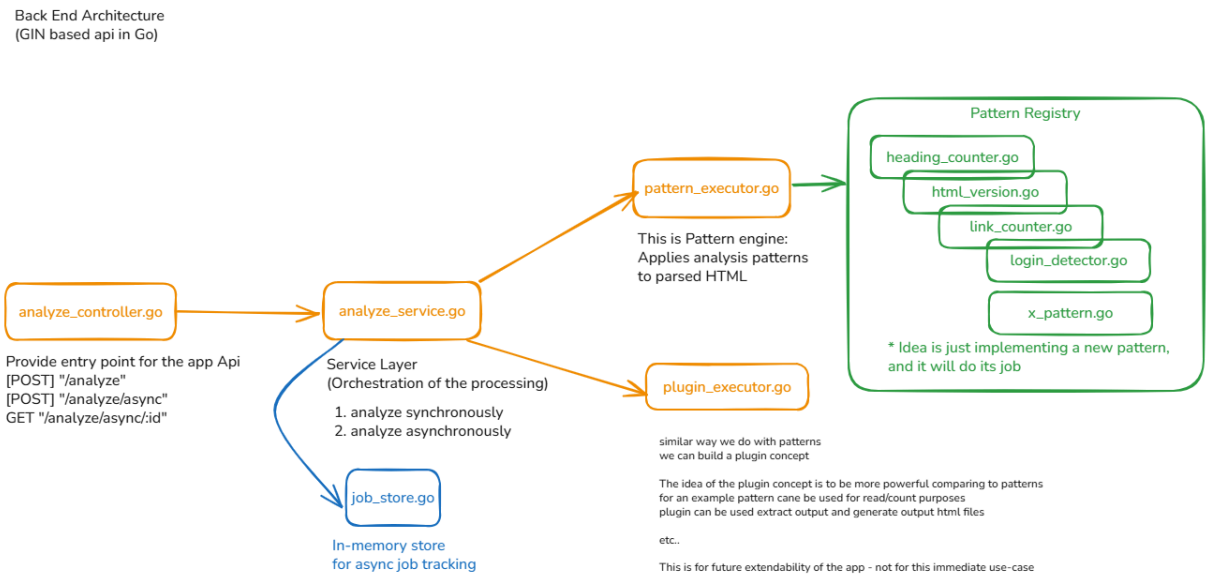
(It's not because I don't know about above but due to time constraints having limited time to evaluate everything)

Repo

https://github.com/Erandauh/web-page-analyzer

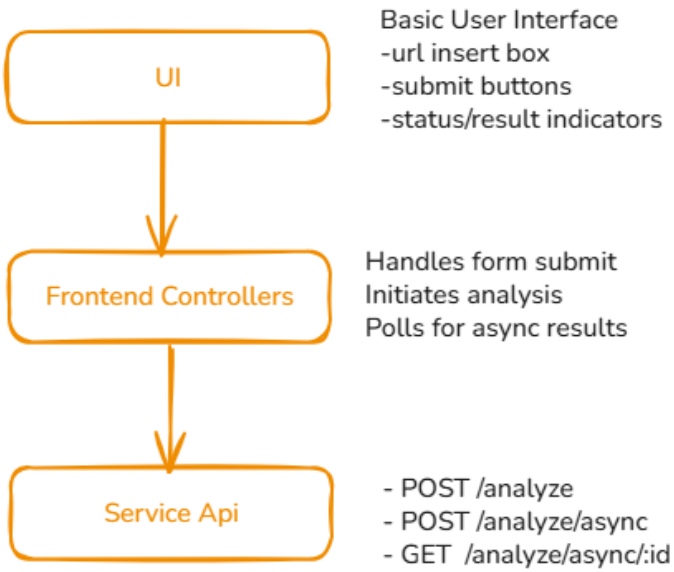
(It's public for now, please let me know once this evaluation is done, so I'll make it private)

Backend Architecture



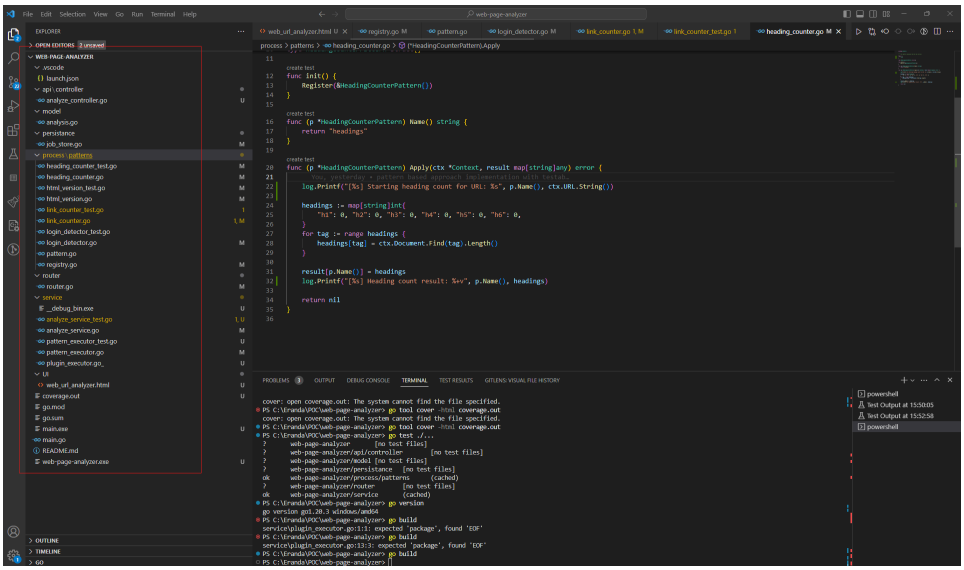
Frontend Architecture

(HTML5 Basic UI)



Development Environment

Windows Based PC
VSCode as IDE
Golong 1.20.3 (due to my system limitation)

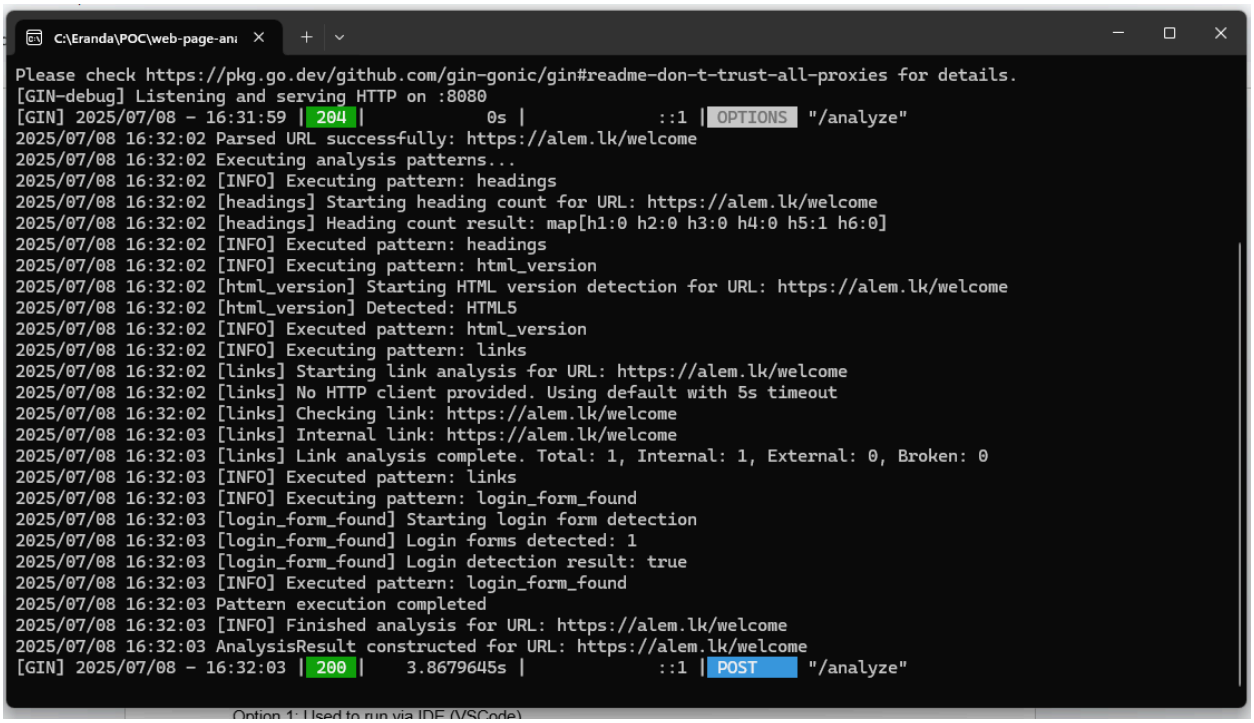


How to Run

Running the BE Server

Option 1: Used to run via IDE (VSCode)

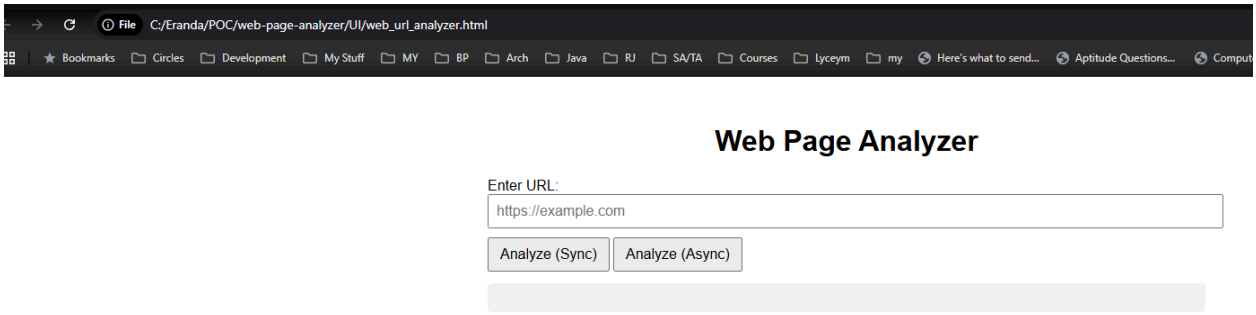
Option 2: Executable exe “web-page-analyzer.exe” is also in the repo itself (if you are using a windows based PC)



Running the FE

Go to Dir
\web-page-analyzer\UI

Just double click and open the “web_url_analyzer.html” in any browser
(its basic JS and HTML, so it should work on any browser, out-of-the-box)

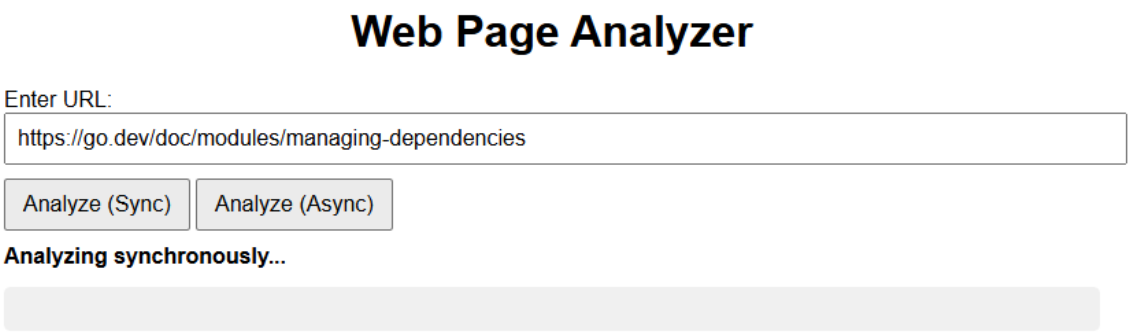


Working App

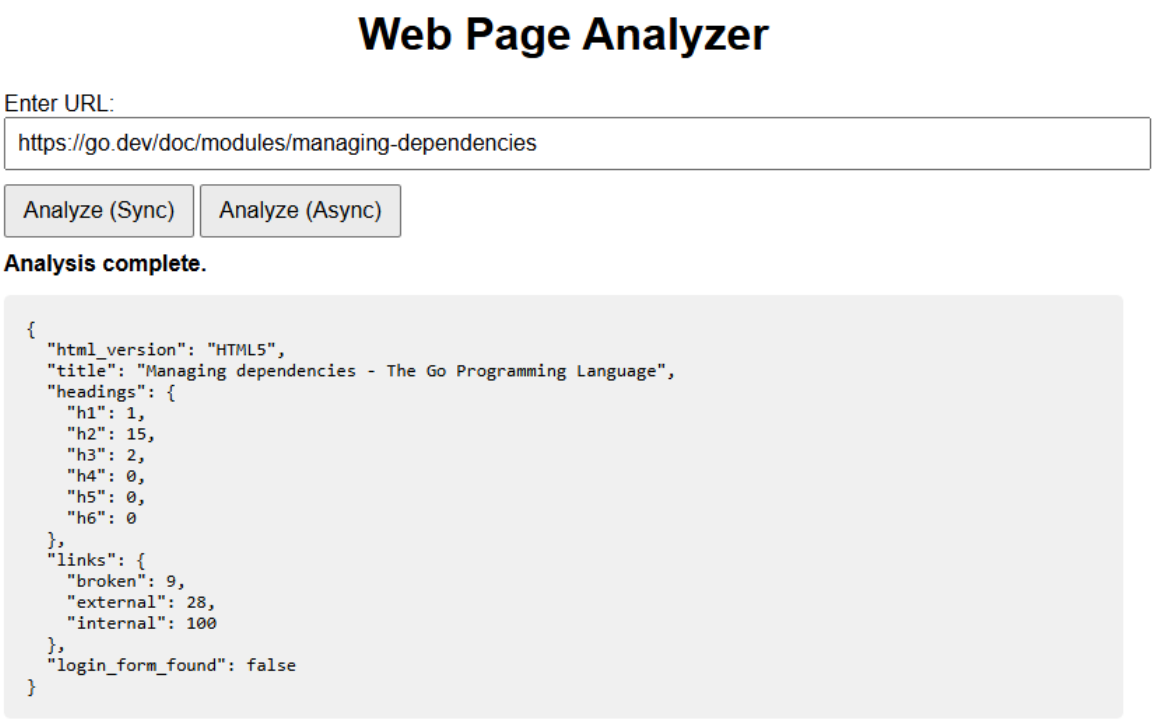
Sync Analysis

Analyze synchronously (this will take a bit of a time, depending on the webpage complexity)

Start:



Results display:

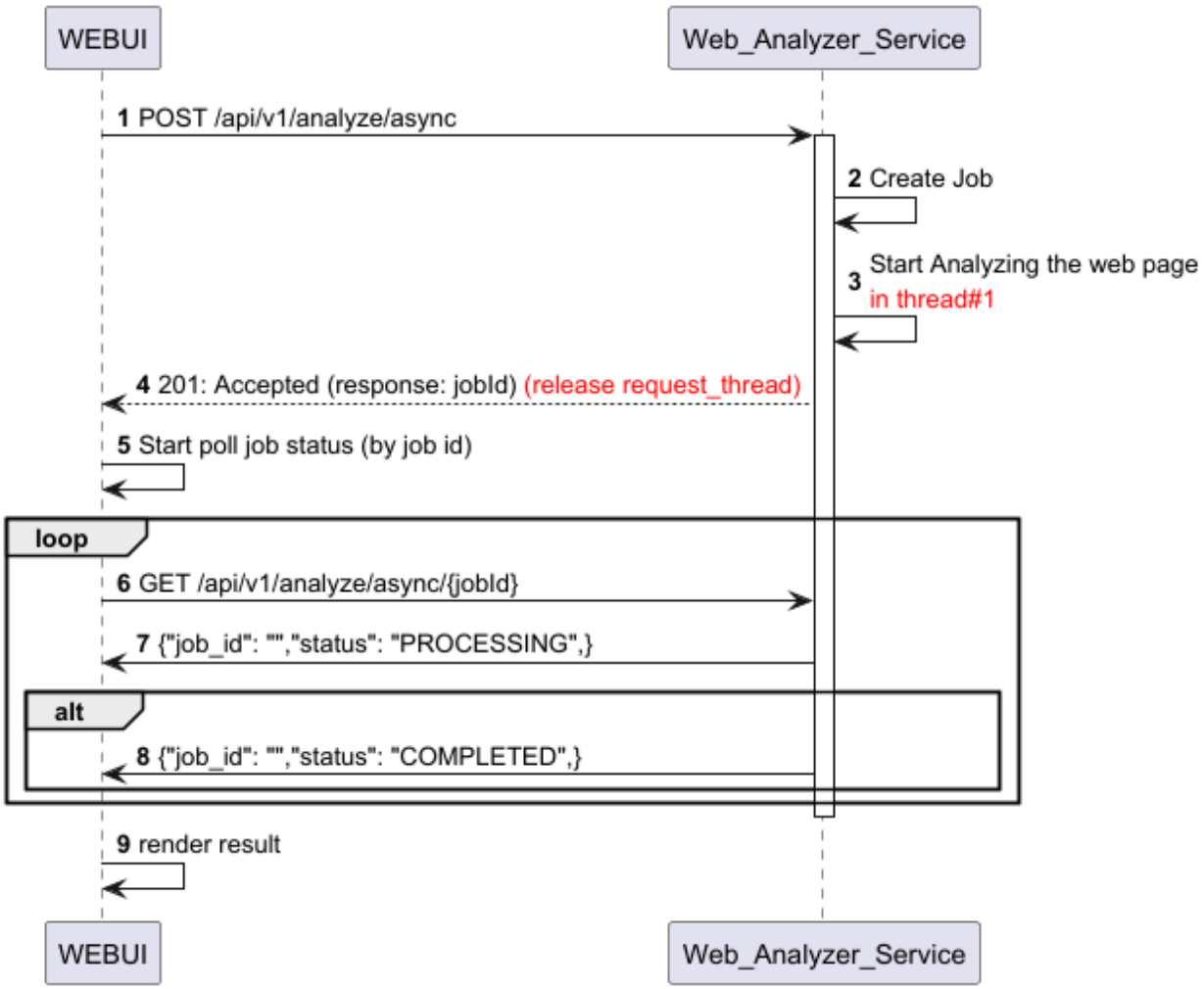


TO DO: display time for analysis, so this becomes handy!

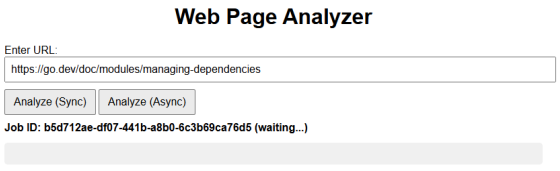
Async Analysis

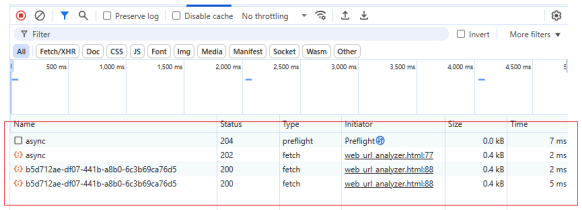
In real world scenario, we should use async way of analyzing web pages as this HTML parse and evaluate is a time-consuming process

See the sequence:

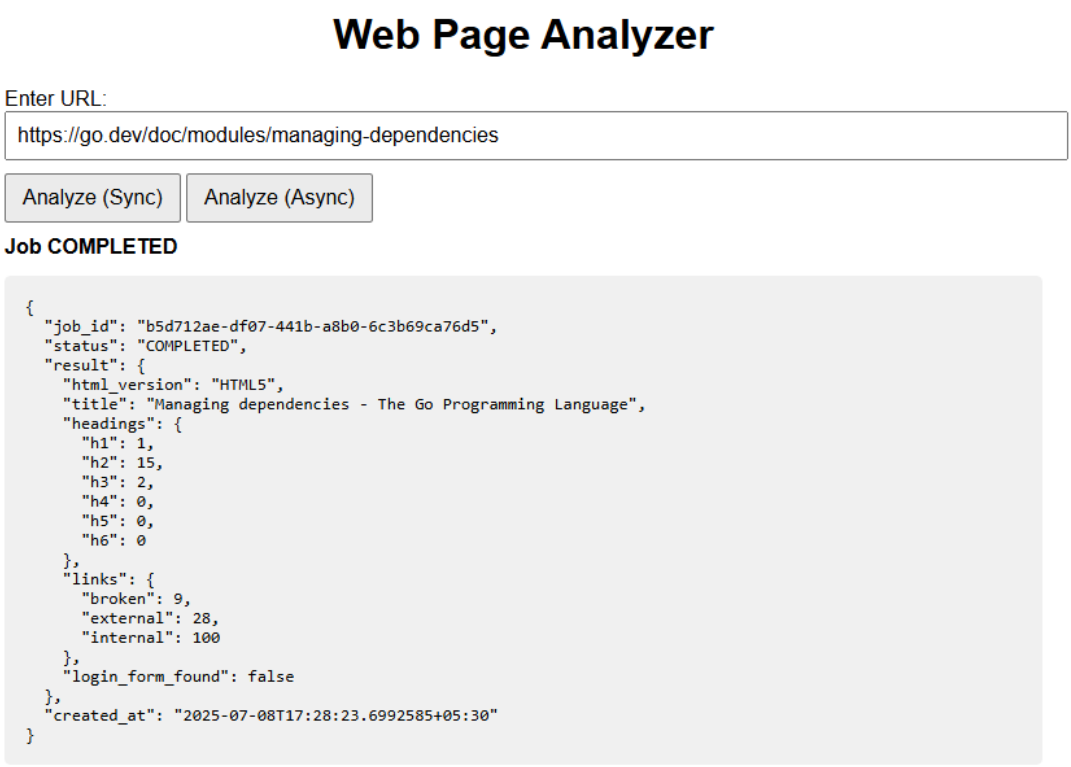


Start: Returns the ‘JobId’ and continue processing



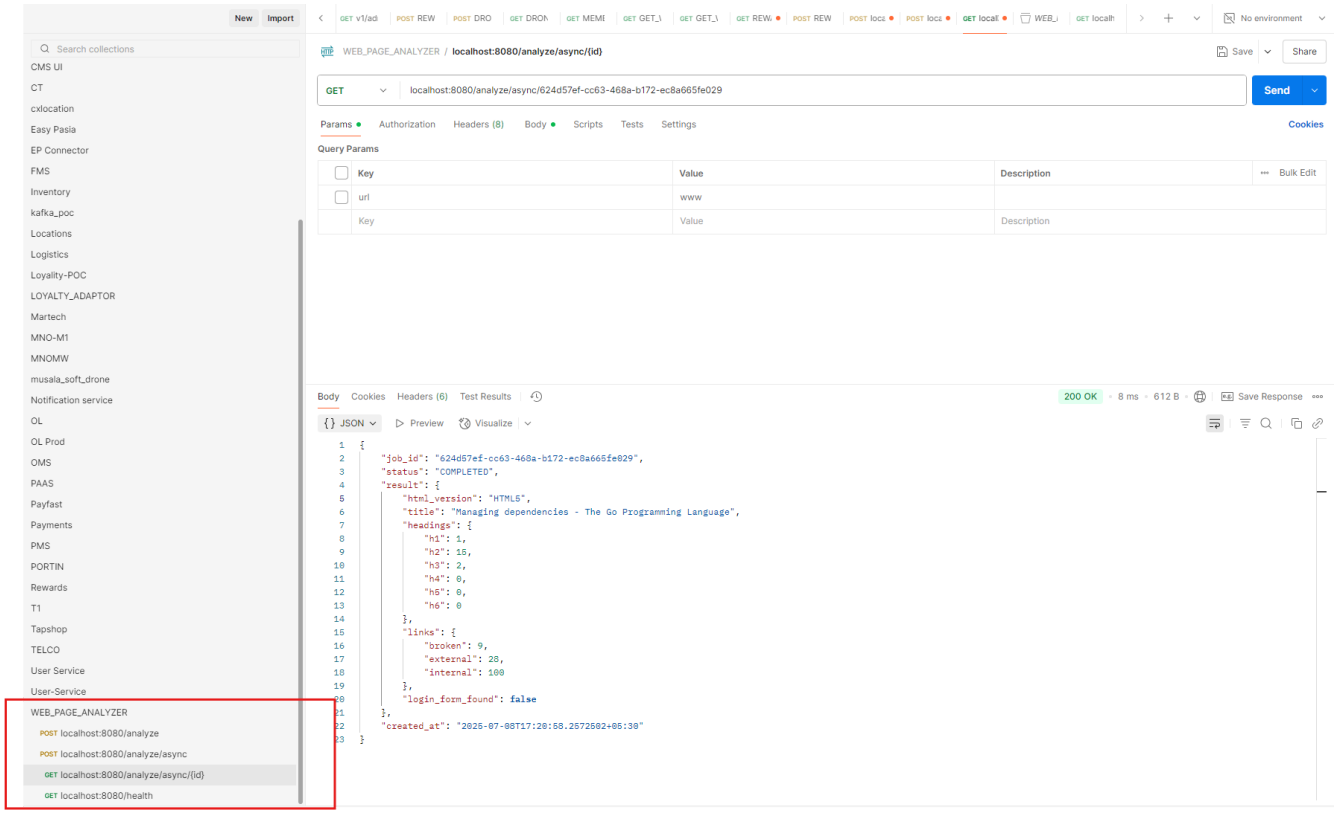


Results display:



Back End Api Collection

There are three main endpoints (excluding the health)



[POST] /analyze

Perform **synchronous analysis** of a provided web page URL

Behavior**:**

- Parses the HTML document.
- Applies multiple analysis patterns (HTML version, headings, links, login form).
- Returns the complete result immediately.

Use Case: Suitable for fast analysis of user interaction in UI.

Request	Response
{ "url": "https://go.dev/doc/modules/managing-dependencies" }	{ "html_version": "HTML5", "title": "Managing dependencies - The Go Programming Language", "headings": { "h1": 1, "h2": 15, "h3": 2, "h4": 0, "h5": 0,

	"h6": 0
	},
	"links": {
	"broken": 9,
	"external": 28,
	"internal": 100
	},
	"login_form_found": false
	}

[POST] /analyze/async
Initiates **asynchronous analysis** of a given URL
Behavior:

- Creates a job with a unique job ID.
- Starts analysis in a background goroutine.
- Immediately returns a job ID to the client.

Use Case: For long-running analysis or UI polling scenarios.

Request	Response

{
"url": "https://go.dev/doc/modules/managing-dependencies"
}
	{
	"job_id": "624d57ef-cc63-468a-b172-ec8a665fe029",
	"status": "PROCESSING",
	"created_at": "2025-07-08T17:20:58.2572502+05:30"
	}

[GET] /analyze/async/:id
Fetch the **status or result** of an async analysis job
Behavior:

- Checks if the job exists and its current status.
- Returns job details with result if available.

Use Case: For long-running analysis or UI polling scenarios.

Request	Response

../analyze/async/624d57ef-cc63-468a-b172-ec8a665fe029	{
	"job_id": "624d57ef-cc63-468a-b172-ec8a665fe029",
	"status": "PROCESSING",
	"created_at": "2025-07-08T17:20:58.2572502+05:30"
	}
	
	OR
	
	{
	"job_id": "624d57ef-cc63-468a-b172-ec8a665fe029",
	"status": "COMPLETED",
	"result": {
	"html_version": "HTML5",
	"title": "Managing dependencies - The Go Programming Language",
	"headings": {
	"h1": 1,
	"h2": 15,
	"h3": 2,
	"h4": 0,
	"h5": 0,
	"h6": 0
	},
	"links": {
	"broken": 9,
	"external": 28,
	"internal": 100
	},
	"login_form_found": false
	},
	"created_at": "2025-07-08T17:20:58.2572502+05:30"
	}
	
	OR
	{
	"job_id":

	"624d57ef-cc63-468a-b172-ec8a665fe029", "status": "FAILED", "created_at": "2025-07-08T17:20:58.2572502+05:30", “error”: “Html parse error!” }

[GET] /health

Health check endpoint to verify that the backend server is running.
Use case: Used by monitoring tools, load balancers, or during deployments.

Request	Response

../health	{
	"status": "ok"
	}

Curl Commands

/analyze	curl --location 'localhost:8080/analyze' \ --header 'Content-Type: application/json' \ --data '{ "url": "https://go.dev/doc/modules/managing-dependencies" }'
/analyze/async	curl --location 'localhost:8080/analyze/async' \ --header 'Content-Type: application/json' \ --data '{ "url": "https://go.dev/doc/modules/managing-dependencies" }'
/analyze/async/id	curl --location 'localhost:8080/analyze/async/624d57ef-cc63-468a-b172-ec8a665fe029' \

	--data ''
/health	curl --location 'localhost:8080/health'

Postman Collection here:
https://github.com/Erandauh/web-page-analyzer/blob/main/WEB_PAGE_ANALYZER.postman_collection.json