

<b>Problem Statement</b>	<b>1</b>
<b>Solution : Web-Page-Analyzer</b>	<b>3</b>
Before You Read	3
Repo	3
Backend Architecture	3
Frontend Architecture	4
Development Environment	4
How to Run	5
Working App	6
Swagger Link	6
Sync Analysis	6
Async Analysis	9
Application Logs	11
Back End Api Collection	12
Curl Commands	16
Docker Image Generate and Run	17
Future Improvements	19
Functional Improvements	19
Codebase Improvements	19

## 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)
  - DI framework : Google wire
  - Logging : logrus
- UI is in Basic HTML5 with CSS
- Docker engine (Tested only in this)
  - Docker Desktop v 23.0.5 running on Window 11 PC
- Unit tests are added for the service layer and core engine only (due to time limitations)

Refer : [Future Improvements](#)

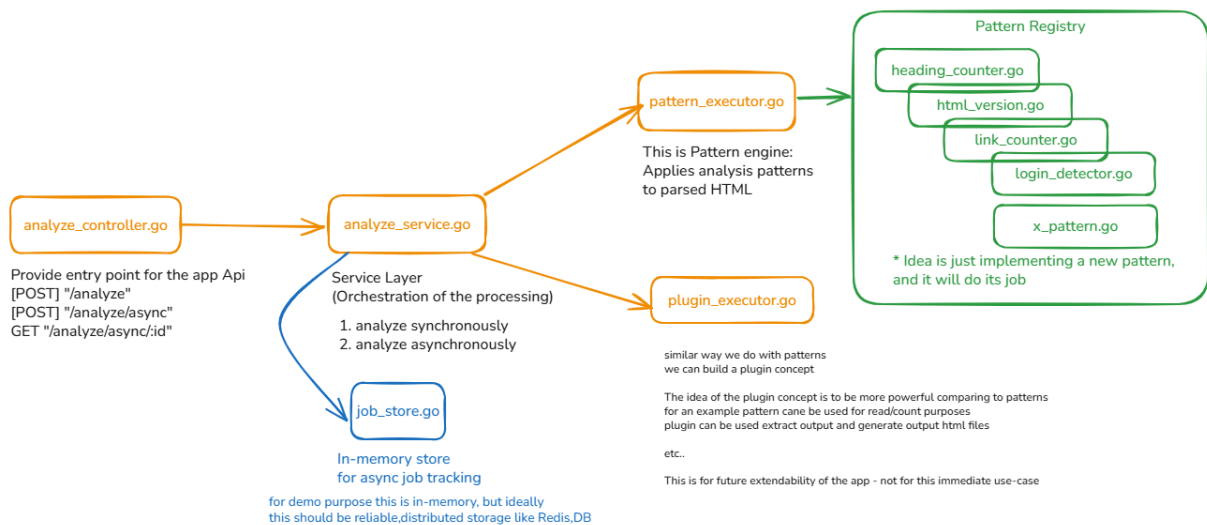
## Repo

<https://github.com/Erandaauh/web-page-analyzer>

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

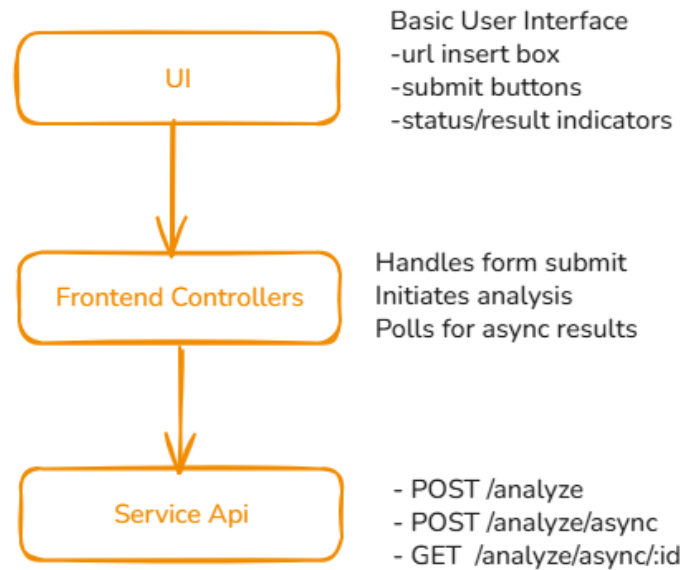
## Backend Architecture

Back End Architecture  
(GIN based api in Go)



# Frontend Architecture

(HTML5 Basic UI)

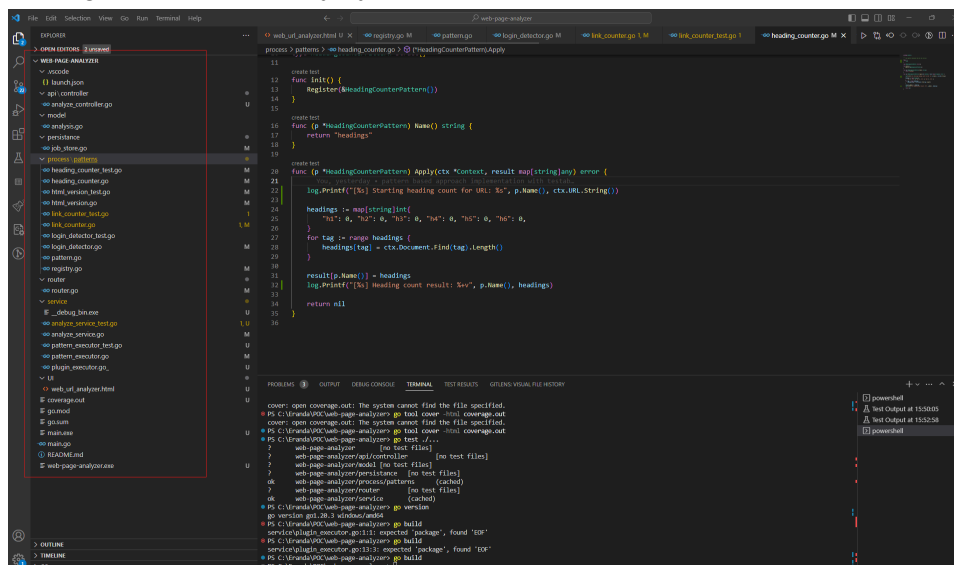


## Development Environment

Windows Based PC

VSCode as IDE

Golang 1.20.3 (due to my system limitation)

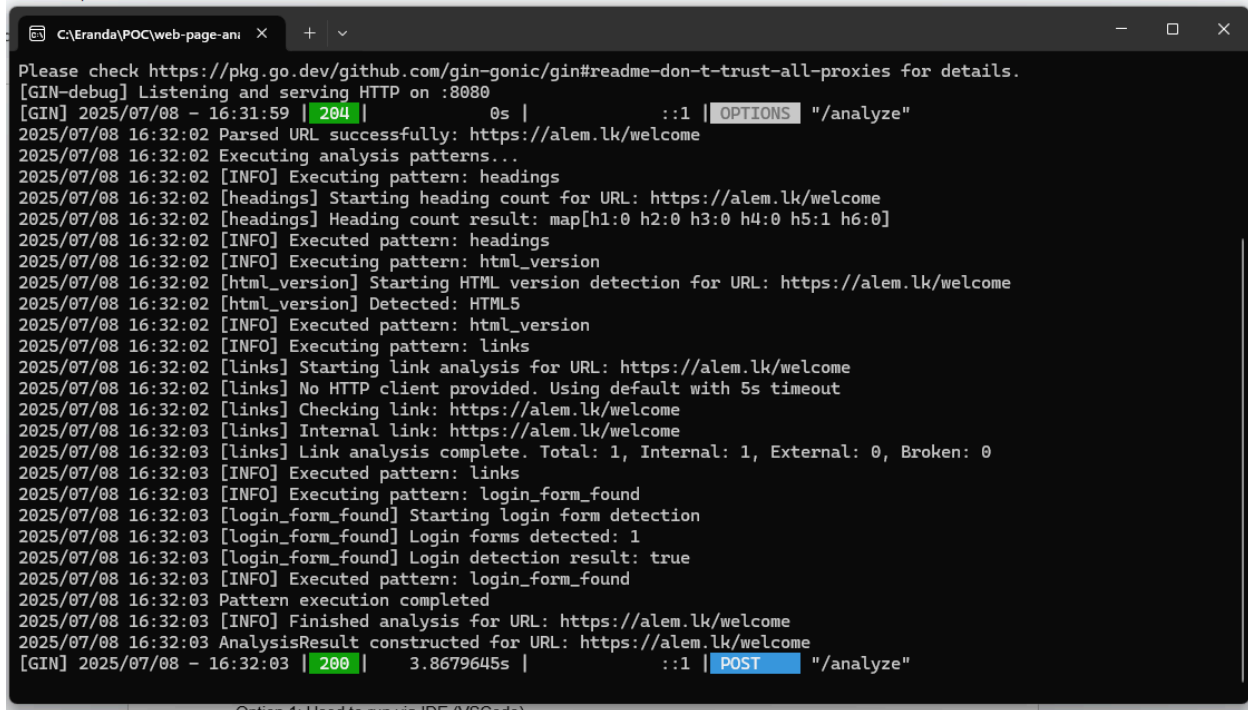


## How to Run

### 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)



```
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Listening and serving HTTP on :8080
[GIN] 2025/07/08 - 16:31:59 | 204 | 0s | ::1 | OPTIONS | "/analyze"
2025/07/08 16:32:02 Parsed URL successfully: https://alem.lk/welcome
2025/07/08 16:32:02 Executing analysis patterns...
2025/07/08 16:32:02 [INFO] Executing pattern: headings
2025/07/08 16:32:02 [headings] Starting heading count for URL: https://alem.lk/welcome
2025/07/08 16:32:02 [headings] Heading count result: map[h1:0 h2:0 h3:0 h4:0 h5:1 h6:0]
2025/07/08 16:32:02 [INFO] Executed pattern: headings
2025/07/08 16:32:02 [INFO] Executing pattern: html_version
2025/07/08 16:32:02 [html_version] Starting HTML version detection for URL: https://alem.lk/welcome
2025/07/08 16:32:02 [html_version] Detected: HTML5
2025/07/08 16:32:02 [INFO] Executed pattern: html_version
2025/07/08 16:32:02 [INFO] Executing pattern: links
2025/07/08 16:32:02 [links] Starting link analysis for URL: https://alem.lk/welcome
2025/07/08 16:32:02 [links] No HTTP client provided. Using default with 5s timeout
2025/07/08 16:32:02 [links] Checking link: https://alem.lk/welcome
2025/07/08 16:32:03 [links] Internal link: https://alem.lk/welcome
2025/07/08 16:32:03 [links] Link analysis complete. Total: 1, Internal: 1, External: 0, Broken: 0
2025/07/08 16:32:03 [INFO] Executed pattern: links
2025/07/08 16:32:03 [INFO] Executing pattern: login_form_found
2025/07/08 16:32:03 [login_form_found] Starting login form detection
2025/07/08 16:32:03 [login_form_found] Login forms detected: 1
2025/07/08 16:32:03 [login_form_found] Login detection result: true
2025/07/08 16:32:03 [INFO] Executed pattern: login_form_found
2025/07/08 16:32:03 Pattern execution completed
2025/07/08 16:32:03 [INFO] Finished analysis for URL: https://alem.lk/welcome
2025/07/08 16:32:03 AnalysisResult constructed for URL: https://alem.lk/welcome
[GIN] 2025/07/08 - 16:32:03 | 200 | 3.8679645s | ::1 | POST | "/analyze"
```

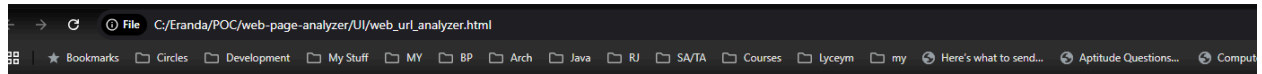
Option 1: Used to run via IDE (VSCode)

### 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)



## Web Page Analyzer

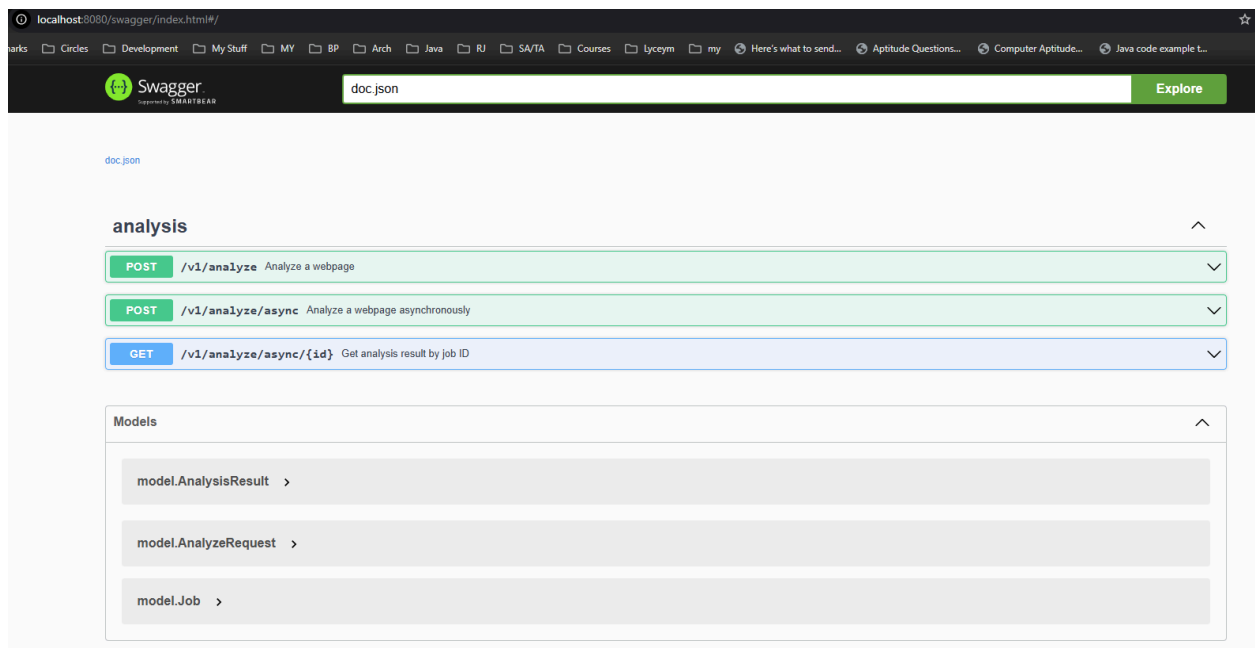
Enter URL:

Working App

[localhost:8080/health](http://localhost:8080/health)

**Swagger Link**

<http://localhost:8080/swagger/index.html#/>



**Sync Analysis**

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

**Start:**


# Web Page Analyzer

Enter URL:

https://go.dev/doc/modules/managing-dependencies

Analyze (Sync)

Analyze (Async)

 Analyzing synchronously...

## Results:

# Web Page Analyzer

Enter URL:

https://go.dev/doc/modules/managing-dependencies

Analyze (Sync)

Analyze (Async)

✅ Analysis complete.

```
{
  "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
}
```

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

## Error:

# Web Page Analyzer

Enter URL:

https://www.tesla.com/

Analyze (Sync)

Analyze (Async)

✖ Failed to analyze!

Show Error Details

# Web Page Analyzer

Enter URL:

https://www.tesla.com/

Analyze (Sync)

Analyze (Async)

✖ Failed to analyze!

Hide Error Details

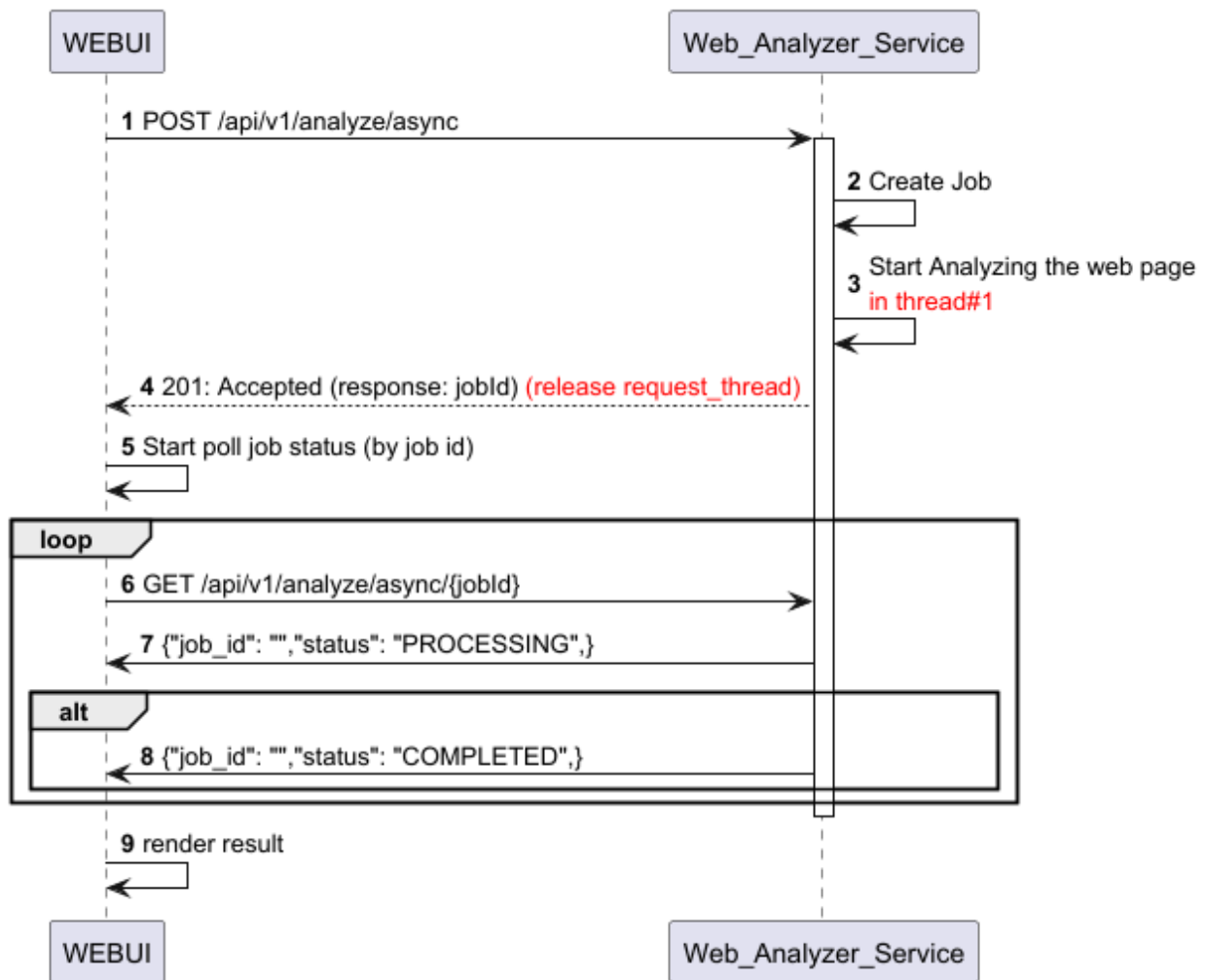
```
{
  "error": "Get \"https://www.tesla.com/\": stream error: stream ID 1; INTERNAL_ERROR; received from peer"
}
```



## 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

File C:/Eranda/POC/web-page-analyzer/UI/web\_url\_analyzer.html

Web Page Analyzer

Enter URL:  
https://go.dev/doc/modules/managing-dependencies

Analyze (Sync) Analyze (Async)

Job ID: 98a4f2d5-639c-422c-b6ed-38131cd44083 (waiting...)

Network Performance Memory Application Privacy and security

Name	Status	Type	Initiator	Size	Time
web_url_analyzer.html	200	document	Other	5.0 kB	4 ms
async	204	preflight	Preflight	0.0 kB	3 ms
async	202	fetch	web_url_analyzer.html:110	0.4 kB	5 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	6 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	2 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	5 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	3 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	6 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	4 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	4 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	3 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	2 ms
98a4f2d5-639c-422c-b6ed-38131cd44083	200	fetch	web_url_analyzer.html:122	0.4 kB	7 ms

## Results:

# Web Page Analyzer

Enter URL:

https://go.dev/doc/modules/managing-dependencies

Analyze (Sync)

Analyze (Async)

✓ Job completed successfully.

```
{
  "job_id": "98a4f2d5-639c-422c-b6ed-38131cd44083",
  "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": "2025-07-10T02:53:39.213115+05:30"
}
```

## Error:

# Web Page Analyzer

Enter URL:

https://www.tesla.com/

Analyze (Sync)

Analyze (Async)

✖ Job failed!

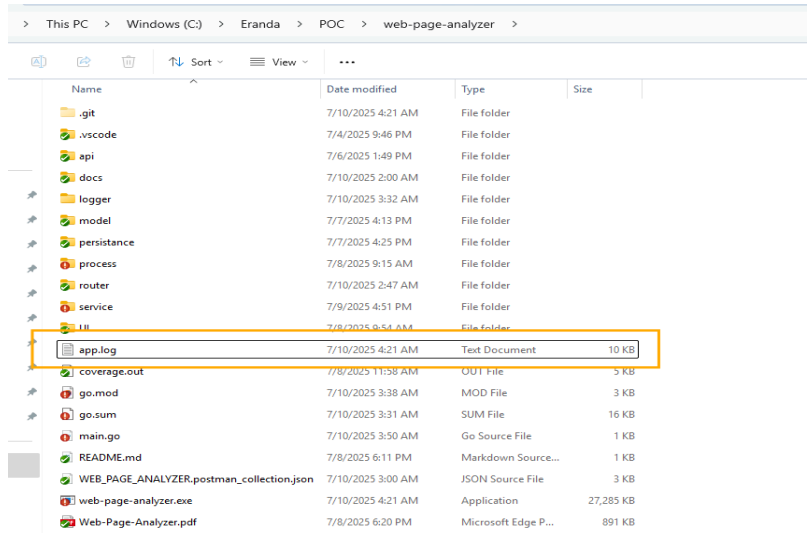
Hide Error Details

Get "https://www.tesla.com/": stream error: stream ID 1; INTERNAL\_ERROR; received from peer

```
{
  "job_id": "24255cc0-b783-486b-96aa-c3ecb3c412a2",
  "status": "FAILED",
  "error": "Get \"https://www.tesla.com/\": stream error: stream ID 1; INTERNAL_ERROR; received from peer",
  "created_at": "2025-07-10T03:12:51.5734907+05:30"
}
```

## Application Logs

Log file is generated in the app folder with name `{app.log}`. [Logrus](#) (pluggable logging framework) is used for logging



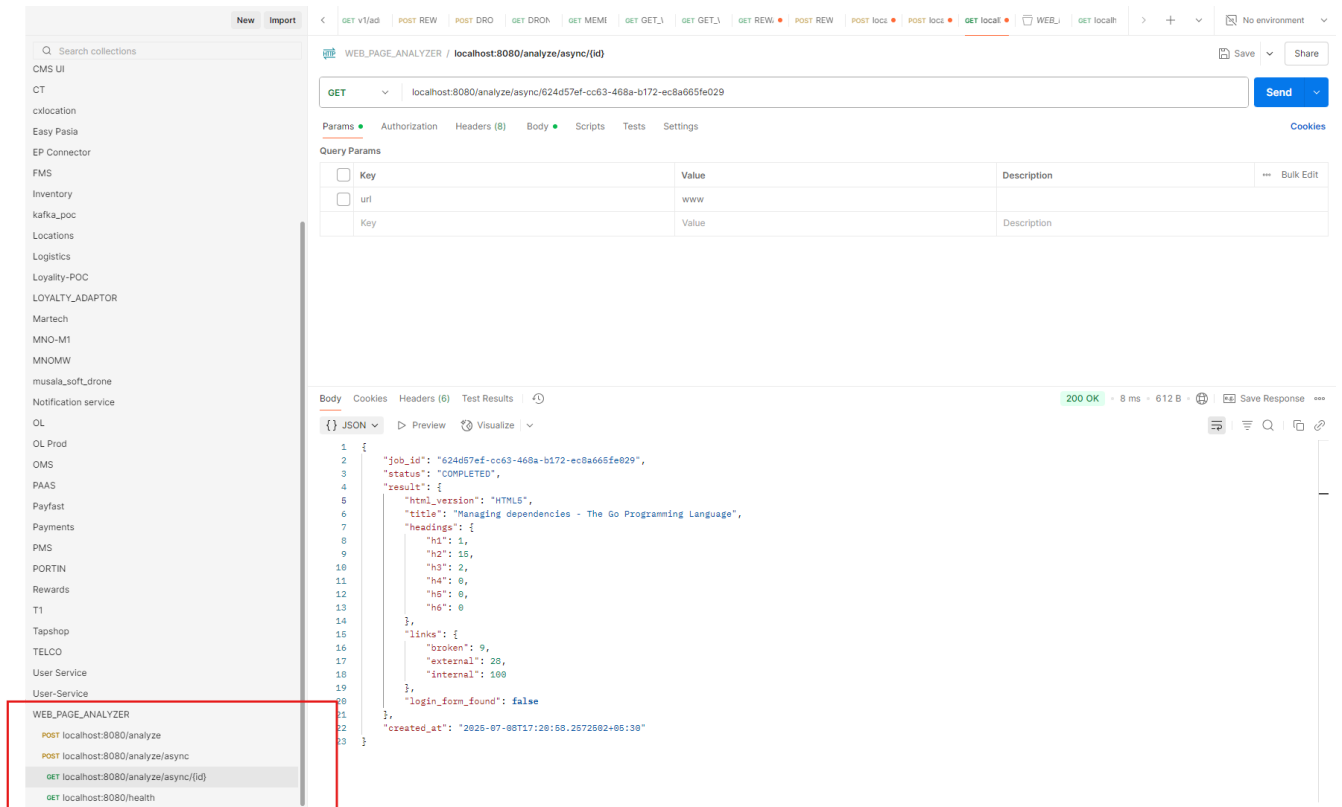
```
telco_sq.txt * OIR.txt * iframe.html * iframe - Copy.htm * catalina.2024-12-0 * ssaa.txt * id_ed25519.pub * id_rsa.pub * Untitled (1) * Coupons Import ( * config * index * packed-refs * Untitled * app.log X
```

```
File Edit View
```

```
time="2025-07-10T03:52:47+05:30" level=info msg="Starting server..."
time="2025-07-10T03:53:18+05:30" level=info msg="Starting heading count" pattern=headings url="https://alem.lk/welcome"
time="2025-07-10T03:53:18+05:30" level=info msg="Heading count completed" pattern=headings result="map[h1:0 h2:0 h3:0 h4:0 h5:1 h6:0]"
time="2025-07-10T04:20:35+05:30" level=info msg="Starting server..."
time="2025-07-10T04:20:48+05:30" level=info msg="Starting async analysis" url="https://alem.lk/welcome"
time="2025-07-10T04:20:48+05:30" level=info msg="Job created" job_id=946e9389-746c-4b56-a695-a85488017093 url="https://alem.lk/welcome"
time="2025-07-10T04:20:48+05:30" level=info msg="Running analysis" job_id=946e9389-746c-4b56-a695-a85488017093 url="https://alem.lk/welcome"
time="2025-07-10T04:20:50+05:30" level=info msg="Parsed URL successfully" parsed_url="https://alem.lk/welcome" url="https://alem.lk/welcome"
time="2025-07-10T04:20:50+05:30" level=info msg="Executing analysis patterns..." url="https://alem.lk/welcome"
time="2025-07-10T04:20:50+05:30" level=info msg="Executing pattern" pattern=headings
time="2025-07-10T04:20:50+05:30" level=info msg="Starting heading count" pattern=headings url="https://alem.lk/welcome"
time="2025-07-10T04:20:50+05:30" level=info msg="Heading count completed" pattern=headings result="map[h1:0 h2:0 h3:0 h4:0 h5:1 h6:0]"
time="2025-07-10T04:20:50+05:30" level=info msg="Pattern executed successfully" elapsed=0s pattern=headings
time="2025-07-10T04:20:50+05:30" level=info msg="Executing pattern" pattern=html_version
time="2025-07-10T04:20:50+05:30" level=info msg="Starting HTML version detection" pattern=html_version url="https://alem.lk/welcome"
time="2025-07-10T04:20:50+05:30" level=info msg="Detected HTML5" pattern=html_version
time="2025-07-10T04:20:50+05:30" level=info msg="Pattern executed successfully" elapsed="773us" pattern=html version
```

## Back End Api Collection

There are three main endpoints (excluding the health)



## [POST] /v1/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
<pre> {   "url":   "https://go.dev/doc/modules/managing-depe ndencies" } </pre>	<pre> {   "html_version": "HTML5",   "title": "Managing dependencies - The Go Programming Language",   "headings": {     "h1": 1,     "h2": 15,     "h3": 2,     "h4": 0,     "h5": 0, </pre>

	<pre>         "h6": 0       },       "links": {         "broken": 9,         "external": 28,         "internal": 100       },       "login_form_found": false     } </pre>
--	--

### [POST] /v1/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
<pre> {   "url":   "https://go.dev/doc/modules/managing-depe ndencies" } </pre>	<pre> {   "job_id":   "624d57ef-cc63-468a-b172-ec8a665fe029",   "status": "PROCESSING",   "created_at":   "2025-07-08T17:20:58.2572502+05:30" } </pre>

### [GET] /v1/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
<code>../analyze/async/624d57ef-cc63-468a-b172-ec8a665fe029</code>	<pre>{   "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":</pre>

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

**[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	<pre>{   "status": "ok" }</pre>

Curl Commands

v1/analyze	<pre>curl --location 'localhost:8080/v1/analyze' \ --header 'Content-Type: application/json' \ --data '{   "url": "https://go.dev/doc/modules/managing-dependencies" }'</pre>
v1/analyze/async	<pre>curl --location 'localhost:8080/v1/analyze/async' \ --header 'Content-Type: application/json' \ --data '{   "url": "https://go.dev/doc/modules/managing-dependencies" }'</pre>
v1/analyze/async/id	<pre>curl --location</pre>



	'localhost:8080/v1/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](https://github.com/Erandauh/web-page-analyzer/blob/main/WEB_PAGE_ANALYZER.postman_collection.json)

## Docker Image Generate and Run

You can use the Dockerfile to generate the docker image

The screenshot shows a VS Code editor with a Dockerfile open in the editor and its execution output in the terminal. The Dockerfile contains the following instructions:

```

0 COPY go.mod go.sum ./
7 RUN go mod download
8
9 COPY . .
10
11 RUN go install github.com/google/wire/cmd/wire@latest
12 RUN wire ./ # generate wire_gen.go
13
14 # Build with docker
15 # Go binary was built against a newer version than runtime container, so need additional flags
16 RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -a -installsuffix cgo -o web-page-analyzer
17
18 # Runtime
19 FROM gcr.io/distroless/base-debian11
20
21 WORKDIR /app
22
23 COPY --from-builder /app/web-page-analyzer .
24
25 ENV TZ=Asia/Colombo
26 EXPOSE 8080
27
28 ENTRYPOINT ["./web-page-analyzer"]

```

The terminal output shows the execution of the Dockerfile, including the building of the image, the transfer of context, and the final naming of the image. The output is as follows:

```

[*] Building 30.4s (17/17) FINISHED
[+] Building 30.4s (17/17) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 588B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for gcr.io/distroless/base-debian11:latest
=> [internal] load metadata for docker.io/library/golang:1.21
=> [stage-1 1/3] FROM gcr.io/distroless/base-debian11@sha256:ac69aa622e5dcbca0883ca877d47069f51bd4282d5c96977e0390d7d256455
=> [builder 1/8] FROM docker.io/library/golang:1.21@sha256:4746d26432a9117a5f58e95c0f954ddfd0e128e9d5816886514199316e4a2fb
=> [internal] load build context
=> => transferring context: 23.04kB
=> CACHED [builder 2/8] WORKDIR /app
=> CACHED [builder 3/8] COPY go.mod go.sum ./
=> CACHED [builder 4/8] RUN go mod download
=> [builder 5/8] COPY . .
=> [builder 6/8] RUN go install github.com/google/wire/cmd/wire@latest
=> [builder 7/8] RUN wire ./ # generate wire_gen.go
=> [builder 8/8] RUN CGO_ENABLED=0 GOOS=linux GOARCH=amd64 go build -a -installsuffix cgo -o web-page-analyzer
=> CACHED [stage-1 2/3] WORKDIR /app
=> [stage-1 3/3] COPY --from-builder /app/web-page-analyzer .
=> exporting to image
=> => exporting layers
=> => writing image sha256:2da408f5a2e5839b89230bf28de739c58730e34289fd9f6485c591816bdcadea6
=> naming to docker.io/library/webpageanalyzer:latest
Terminal will be reused by tasks, press any key to close it.

```

I have Docker Desktop running in my Windows Laptop, If you have similar setup you should be able to just run it (Said that not tested in other systems like Ubuntu, Mac: It should work out of the box though)

The screenshot shows the Docker Desktop interface with the 'Images' tab selected. The left sidebar contains navigation options: Containers, Images, Volumes, Dev Environments (BETA), Docker Scout (EARLY ACCESS), and Learning Center. Below this is an 'Extensions' section with 'RabbitMQ' and 'Add Extensions'. The main area displays a list of Docker images. A red box highlights the first image, 'webpageanalyzer' (b2f724f826c), which is in an 'Unused' state. Other images include 'redis', 'testcontainers/ryuk', 'testcontainers/ssh', 'redis', 'rabbitmq', 'rabbitmq', 'redis', 'redis/redis-stack-server', 'yogendra08sharma/rabbitmq-docker-extension', 'redis', and 'redis'. The bottom status bar shows 'RAM 5.03 GB', 'CPU 0.35%', and 'Not connected to Hub'.

Name	Tan	Status	Created	Size	Actions
webpageanalyzer b2f724f826c	latest	Unused	less than a mi	48.33 MB	▶   ⋮   🗑
redis 590b81f2ea1	latest	In use	12 months ago	116.95 MB	▶   ⋮   🗑
testcontainers/ryuk e1c72b043805	0.8.1	Unused	12 months ago	15.41 MB	▶   ⋮   🗑
testcontainers/ssh 7c0468e27300	1.2.0	Unused	about 1 year a	13.92 MB	▶   ⋮   🗑
redis d3d0b9fa4be	6.0	In use	over 1 year ag	126.37 MB	▶   ⋮   🗑
rabbitmq f4a13390c450	3-management-alpine	In use	about 2 years	183.4 MB	▶   ⋮   🗑
rabbitmq 70ee31043d3c	3-management	In use	about 2 years	255.24 MB	▶   ⋮   🗑
redis 0ec8ab59a35f	<none>	In use (dangling)	about 2 years	117.11 MB	▶   ⋮   🗑
redis/redis-stack-server 0021013ed76	latest	In use	about 2 years	328.82 MB	▶   ⋮   🗑
yogendra08sharma/rabbitmq-docker-extension 875c2fa2cc42	0.0.1	Unused	over 2 years a	7.46 MB	▶   ⋮   🗑
redis 3c3da61c4be0	6.2.6	Unused	about 3 years	112.61 MB	▶   ⋮   🗑
redis 3d2a373f46ae	5.0.3-alpine	Unused	over 6 years a	50.83 MB	▶   ⋮   🗑

The screenshot shows the Docker Desktop interface with the 'Logs' tab selected for the 'objective\_carver' container. The left sidebar is the same as the previous screenshot. The main area displays the container's logs. The logs show the container starting up, creating an engine instance, running in debug mode, and then listening on port 8080. The logs also show the container's internal API endpoints and the Swagger UI. The bottom status bar shows 'RAM 5.03 GB', 'CPU 0.35%', and 'Not connected to Hub'.

```
2025-07-10 14:13:47 [GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.
2025-07-10 14:13:47 [GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
2025-07-10 14:13:47 - using env: export GIN_MODE=release
2025-07-10 14:13:47 - using code: gin.SetMode(gin.ReleaseMode)
2025-07-10 14:13:47 [GIN-debug] GET /health --> web-page-analyzer/router.SetupRouter.func1 (6 handlers)
2025-07-10 14:13:47 [GIN-debug] POST /v1/analyze --> web-page-analyzer/internal/apl.(*AnalyzerController).Analyze-fn (6 handlers)
2025-07-10 14:13:47 [GIN-debug] POST /v1/analyze/async --> web-page-analyzer/internal/apl.(*AnalyzerController).AnalyzeAsync-fn (6 handlers)
2025-07-10 14:13:47 [GIN-debug] GET /v1/analyze/async/:id --> web-page-analyzer/internal/apl.(*AnalyzerController).GetAsyncAnalysisById-fn (6 handlers)
2025-07-10 14:13:47 [GIN-debug] GET /swagger/*any --> github.com/swaggo/gin-swagger.CustomWrapHandler.func1 (6 handlers)
2025-07-10 14:13:47 [GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
2025-07-10 14:13:47 Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
2025-07-10 14:13:47 [GIN-debug] Listening and serving HTTP on :8080
2025-07-10 14:14:10 [GIN] 2025/07/10 - 14:14:10 | 204 | 162.33µs | 172.17.0.1 | OPTIONS "/v1/analyze/async"
2025-07-10 14:14:10 [GIN] 2025/07/10 - 14:14:10 | 202 | 437.88µs | 172.17.0.1 | POST "/v1/analyze/async"
2025-07-10 14:14:12 [GIN] 2025/07/10 - 14:14:12 | 200 | 97.87µs | 172.17.0.1 | GET "/v1/analyze/async/10880d58-81bf-4e3a-b7cf-339ab4031fbb"
2025-07-10 14:14:14 [GIN] 2025/07/10 - 14:14:14 | 200 | 91.85µs | 172.17.0.1 | GET "/v1/analyze/async/10880d58-81bf-4e3a-b7cf-339ab4031fbb"
```

## Future Improvements

### Functional Improvements

1. **Include Caching support for Frequent URLs**  
Cache analyzed results with TTL (e.g., with Redis) to reduce redundant processing overhead.
2. **Support Batch URL Analysis**  
Accept a list of URLs for batch analysis in one request.
3. **Persist Async Jobs (Support Multipods)**  
Replace in-memory store with persistent and centralized job store (e.g., PostgreSQL, Redis).
4. **Introduce Cleanup process/Job**
5. **Add elapsed time as a response attribute** (currently it's in application logs only)

### Codebase Improvements

(These are not addressed due to time constraints)

1. Core BL and Engine is fully covered with the unit tests. However, unit test coverage can be increased and extended to other modules too.
2. Use rolling log files