

# Upload Menu

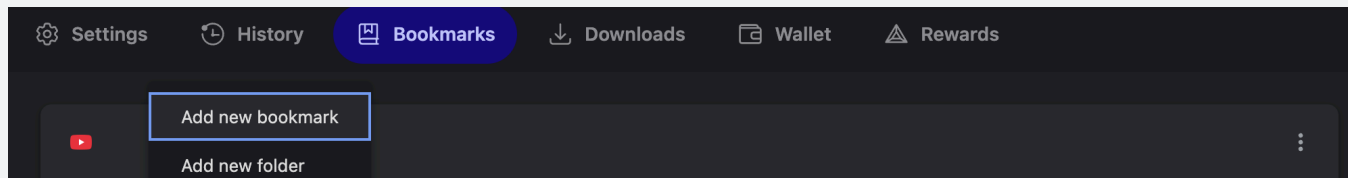
# Setting up your browser

## Overview

Here are instructions for the simple 2-step process of being able to open up the Webcrawling menu on your browser!

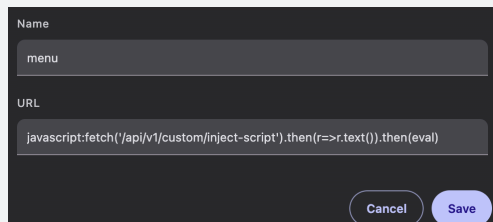
### STEP 1

→ In your browser's Bookmark menu, add a new one:



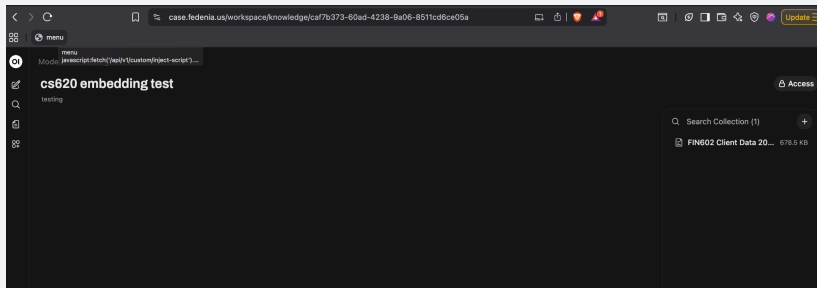
→ Name it whatever you like, and include the following command in the URL:

→ `javascript:fetch('/api/v1/custom/inject-script').then(r=>r.text()).then(eval)`

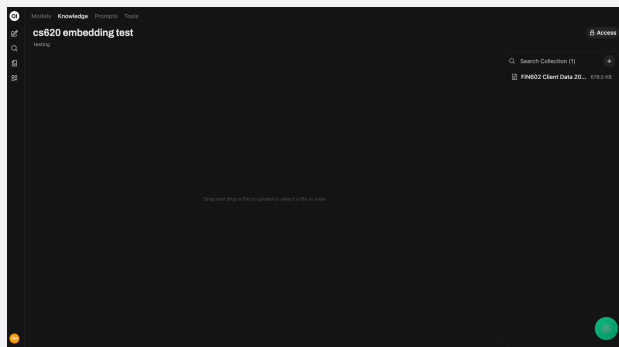
A screenshot of a browser's bookmark dialog box. It has two input fields: 'Name' and 'URL'. The 'Name' field contains the text 'menu'. The 'URL' field contains the command `javascript:fetch('/api/v1/custom/inject-script').then(r=>r.text()).then(eval)`. At the bottom right, there are two buttons: 'Cancel' and 'Save'.

## STEP 2

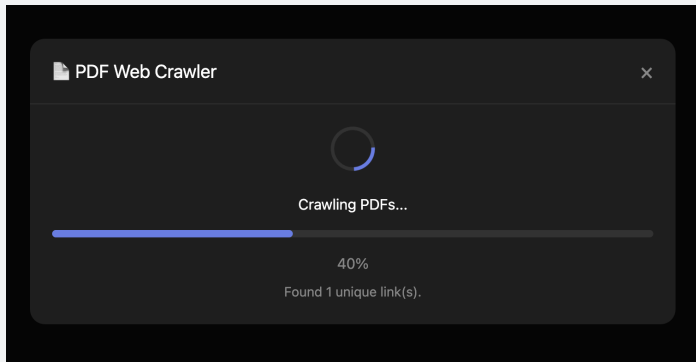
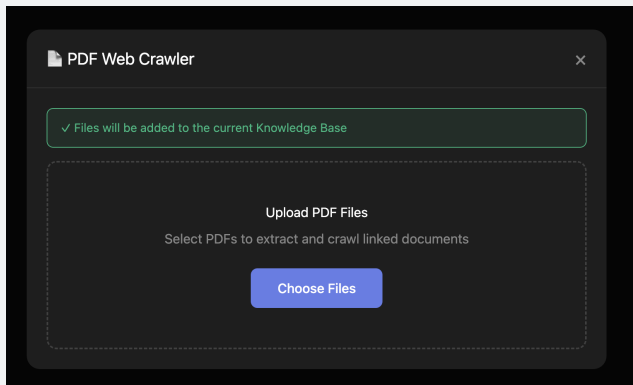
→ Once in your preferred Knowledge Base, select the bookmark

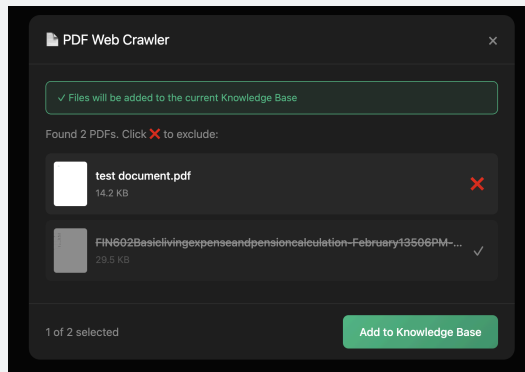
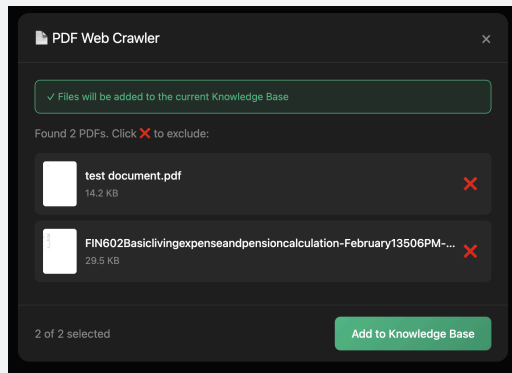


→ In the bottom-right corner of your screen, a new button should appear (That's the webcrawl upload button!)



→ Click it and feel free to upload & webcrawl as seen below!





- As seen above, after clicking the “Add to Knowledge Base” Button, your selected files will be added (following a brief page refresh)
- WARNING: files containing many links will likely take a longer period of time!

# Changes In The OpenWebUI Code

What changes were/are needed to make the menu work?

## [Main.py](#) (/app/backend/open\_webui/main.py)

→ First was importing our routing file, which allows for all our backend API calls:

```
70 from open_webui.routers import (
71     audio,
72     images,
73     ollama,
74     openai,
75     retrieval,
76     pipelines,
77     tasks,
78     auths,
79     channels,
80     chats,
81     notes,
82     folders,
83     configs,
84     groups,
85     files,
86     functions,
87     memories,
88     models,
89     knowledge,
90     prompts,
91     evaluations,
92     tools,
93     users,
94     utils,
95     scim,
96     custom_pdf_router,
97 )
98
```

→ Next was officially including the router in the app API:

```
1308 app.include_router(
1309     evaluations.router, prefix="/api/v1/evaluations", tags=["evaluations"]
1310 )
1311 app.include_router(utils.router, prefix="/api/v1/utils", tags=["utils"])
1312
1313 app.include_router(custom_pdf_router.router, prefix="/api/v1/custom", tags=["custom_pdf"])
1314
1315 # SCIM 2.0 API for identity management
1316 if SCIM_ENABLED:
1317     app.include_router(scim.router, prefix="/api/v1/scim/v2", tags=["scim"])
1318
1319
```

And that's all!

# What to expect for future versions

- For future versions, even though the layout of the specified [main.py](#) file may change, just having the router imported where the other routers seem to be and 'including' the router/API in the same way the others seem to be will likely be the only requirements!