How Developers Can Auto-Create Claude Skills from Any Framework's Docs

Turn open-source documentation into Claude-ready Skills in minutes, just your terminal and a GitHub repo.

After publishing my last piece about building a Claude Skill from scratch, let's take things forward and generate a Skill automatically from existing documentation. Turn an entire framework's docs into a working Claude Skill. A new open-source project, will change how you think about automating skill creation.

Hi. Jannis here.

Learn how to transform any framework's documentation like React, Vue, Django and more into a Claude Skill using the open-source Skill Seekers project. This hands-on walkthrough shows the complete manual upload workflow, no API key required (Just a regular Claude subscription).

Why I Needed a Faster Way

My first manually coded Skill worked, but it took time: collecting examples, structuring YAML, writing descriptions. For small utilities that's fine, but when you want to support frameworks like React or Django, you're not going to copy paragraphs by hand. I wanted something that could handle that bulk automatically and still give me a clean upload.

That's what made Skill Seekers interesting. It treats documentation like a dataset: crawl, clean, convert. You tell it what docs to read, and it spits out a structured Skill folder Claude can understand.

Setting It Up

Here's how I got started.
I cloned the repository to my dev folder:

```
git clone https://github.com/yusufkaraaslan/Skill_Seekers.git
cd Skill_Seekers
```

Then I installed the dependencies:

```
pip install -r requirements.txt
```

It's pure Python — no fancy environment needed. Inside the `cli` folder there are two main scripts: `doc_scraper.py` for fetching docs and `package_skill.py` for zipping them into a Claude Skill package. The `configs` directory contains JSON templates for different frameworks. I opened `configs/react.json` to see how it worked. Each entry defines the base URL, depth, and cleaning rules. You can duplicate one of those files and point it to your own docs if you want to customize later.

Generating a Skill from React Docs

To test it, I ran:

```
python3 cli/doc_scraper.py --config configs/react.json --enhance-local
```

The `--enhance-local` flag tells the script to store the processed output locally instead of sending it anywhere. It crawled the React documentation and dumped everything into an `output/react/` directory. The process took a few minutes, and when it finished I had a folder full of Markdown files and metadata—each section of the docs neatly converted into Claude-readable snippets.

What surprised me most was how well it respected the site's structure. Each subtopic became its own content block: hooks, state management, components, and so on. It's the kind of organization Claude's Skills API loves, because it can reference each section separately instead of one giant text file.

Packaging and Uploading (Manual Mode)

I packaged the Skill:

```
python3 cli/package_skill.py output/react/
```

That command created a single zip file: `output/react.zip`. Inside Claude, I opened the **Skills** page, clicked **Upload Skill**, and selected the zip. That's it—no authentication, no environment variables, no token juggling.

The Skill appeared in my list immediately. I opened Claude and asked, "Explain useEffect in React with a short example." The response came back in seconds, accurate and contextually aware — clearly drawn from the React docs it had just consumed. That's when it clicked: I'd just built a doc-aware assistant without writing any integration code.

Customizing and Extending

Once it worked, I tried Vue. I copied `configs/vue.json`, tweaked the start URL to Vue's docs, and ran the same command. The scraper handled it flawlessly. You can adjust the `max_depth` value if the docs are nested, or change `ignore_patterns` to skip API references you don't need.

If you want to get fancy, you can merge multiple frameworks by running separate scrapes and packaging them into one Skill. I've also tested adding local Markdown files instead of URLs — Skill Seekers reads both.

What I Learned

The manual upload mode might sound basic, but it's perfect for developers who want full control. You can inspect every file, edit metadata, or even translate the docs before uploading. It's local-first in the best sense of the word.

Skill Seekers gives you automation without taking away transparency. And once you've built one Skill this way, you start to imagine a whole library of them — each generated from the frameworks you already use daily.

GitHub - yusufkaraaslan/Skill_Seekers: Single powerful tool to convert ANY documentation website…

Single powerful tool to convert ANY documentation website into a Claude skill - yusufkaraaslan/Skill_Seekers

github.com

If you found this article helpful, A few claps 👏, a highlight 🖍️, or a comment 💬 really helps.

If you hold that 👏 button down something magically will happen, Try it!

Don't forget to follow me to stay updated on my latest posts. Together, we can continue to explore fascinating topics and expand our knowledge.

Thank you for your time and engagement!