
Teddy

I need to build an application for common Digital Humanities projects. Specifically, I want an application that can create visualizations using maps, network graphs, and topic models.

The application should be as self-contained as possible. For example, it should have a high resolution map of the earth, drawn from scratch. It should include its own implementation of force-directed graph drawing -- ideally, ForceAtlas2 -- and the ability to return an image of the output. It should also include its own implementation of LDA. In all cases, the implementations should prioritize efficiency.

The inputs for each application should be structured. A map will be generated from a csv with columns that include place names, in addition to various metadata. A network visualization will be generated from a csv with Source-Target links. A topic model will be generated from a folder of plaintext files.

There should be a graphical user interface with intuitive instructions. Users should, first, have the option to select which application they want to use and then they should be guided to load any files into the program. Outputs should be visualized within the application's interface, with the option to download them.

The codebase for the application should be generated in ways that support installation to various recent operating systems, including macOS 26.1, as well as a web app that can be hosted through cPanel. Generate an installation file for both the macOS 26.1 installer, as well as a generic cPanel installer.

> Claude
>
> I'll start by reading the relevant skill files, then build this comprehensive Digital Humanities toolkit.Now I'll build this comprehensive application. Given the complexity, I'll create a Python-based desktop app with a web interface, implementing all three tools from scratch.Now the main Flask application with the GUI:Now the HTML template — the full GUI:

[Interrupted due to context limit]

> The file got cut off. Let me continue with the rest of the template:Let me create the template in parts to handle the size:Now create the requirements, setup files, and installers:Now let's test that everything works:All tests pass. Let me make the installer scripts executable and package everything:Here's your **DH Toolkit** — a self-contained Digital Humanities visualization suite. The zip contains the complete application with all three tools:
>
> **What's included:**

- **Mapping engine** — Draws a world map from scratch using embedded coastline polygons (Africa, Europe, Asia, Americas, Australia, islands). Supports Robinson, Mercator, and Equirectangular projections. Includes a built-in gazetteer of ~200 major cities for offline geocoding.
- **Network graph engine** — Full ForceAtlas2 implementation with Barnes-Hut quad-tree approximation (O(n log n)) for large graphs, adaptive speed control, and PNG rendering with automatic node sizing/labeling by degree.
- **Topic modelling engine** — LDA via collapsed Gibbs sampling with configurable priors, automatic tokenization/stop-word removal, and both visual and JSON output.
- **Web GUI** — Step-by-step interface (Upload → Configure → Result) with drag-and-drop, data previews, and download buttons.
- **`install_macos.sh`** — Creates a `.app` bundle in `~/Applications` with auto-Python detection and Homebrew fallback. Compatible with macOS 14+.
- **`install_cpanel.sh`** — Generates `passenger_wsgi.py`, `.htaccess` for Phusion Passenger, and a gunicorn launcher with systemd template as an alternative.

To run immediately:

```
unzip dh-toolkit.zip && cd dh-toolkit
pip install -r requirements.txt
python app.py
```

Then open `http://localhost:5000`.

Walk me through the cPanel installation in slightly more detail.

The cPanel installer handles three deployment scenarios, so let me walk through each path.

Before you start, upload the entire `dh-toolkit` folder to your server's home directory (e.g., via cPanel File Manager or SFTP). Then SSH in and run:

```
chmod +x install_cpanel.sh
./install_cpanel.sh
```

The script auto-detects whether you're on cPanel by checking for `~/public_html`. It searches for Python 3.10+ in the standard paths first, then checks cPanel's CloudLinux locations (`/opt/alt/python3XX/bin/`). It copies the app to `~/dh-toolkit/`, creates a virtual environment there, and installs the three dependencies.

What it sets up for Passenger (the most common cPanel path): It writes a `passenger_wsgi.py` entry point that activates the venv and imports the Flask app as `application`. It also creates `~/public_html/dh-toolkit/.htaccess` with the Passenger directives pointing to your app directory and Python binary. At that point, visiting `yourdomain.com/dh-toolkit` should work — though some hosts require you to also go to cPanel → **Setup Python App**, create a new application pointing to `~/dh-toolkit`, and set the startup file to `passenger_wsgi.py`. The `.htaccess` and the cPanel UI method accomplish the same thing; which one you need depends on your host's configuration.

If you're on a VPS instead, the script also generates `start_gunicorn.sh` (binds to port 8080 with 3 workers) and a `dh-toolkit.service` systemd unit file. For that path you'd:

```
sudo cp ~/dh-toolkit/dh-toolkit.service /etc/systemd/system/
sudo systemctl enable --now dh-toolkit
```

Then configure nginx or Apache as a reverse proxy to `localhost:8080`.

One thing to watch for: the `--timeout 300` on gunicorn is intentional — LDA on a large corpus or ForceAtlas2 on a dense graph can take a couple of minutes, and the default 30-second timeout would kill those requests. If you're using Passenger instead, you may want to set `PassengerMaxRequestTime 300` in your `.htaccess` for the same reason.

How would I install via the Setup Python App interface in cPanel?

That's a good question since many shared hosts require you to use the cPanel UI rather than relying on `.htaccess` alone. Here's the walkthrough:

1. Upload your files

Use cPanel's File Manager or SFTP to upload the `dh-toolkit` folder to your home directory, so you end up with `~/dh-toolkit/`containing `app.py`, `engines/`, `templates/`, etc.

2. Open Setup Python App

In cPanel, search for "Setup Python App" (it's usually under the Software section). Click **Create Application**.

3. Configure the application

You'll see a form with these fields:

- **Python version** — Choose 3.10, 3.11, or 3.12 (whichever your host offers that's 3.10+)
- **Application root** — Enter `dh-toolkit` (this is relative to your home directory)
- **Application URL** — This controls the URL path. Enter something like `dh-toolkit` for `yourdomain.com/dh-toolkit`, or leave it blank if you want it at the domain root
- **Application startup file** — Enter `passenger_wsgi.py`
- **Application entry point** — Enter `application`

Hit **Create**.

4. Install dependencies

After creating the app, cPanel will show you the application's dashboard. There's usually a section for installing packages. You have two options:

- Use the "Add package" button to add `flask`, `numpy`, and `Pillow` one at a time
- Or use the terminal approach: at the top of the app dashboard, cPanel displays a command to "enter the virtual environment" — something like `source /home/yourusername/virtualenv/dh-toolkit/3.12/bin/activate && cd /home/yourusername/dh-toolkit`. Copy that, open Terminal in cPanel (or SSH in), paste it, then run `pip install -r requirements.txt`

5. Create the WSGI entry point

If the installer script already ran, `passenger_wsgi.py` exists. If you uploaded files manually and skipped the install script, you need to create `~/dh-toolkit/passenger_wsgi.py` with this content:

```
import sys, os
app_dir = os.path.dirname(os.path.abspath(__file__))
sys.path.insert(0, app_dir)
from app import app as application
```

Note this is simpler than what the install script generates — you don't need the venv activation lines because cPanel's Setup Python App handles the virtual environment automatically.

6. Restart and visit

Back on the app dashboard, click **Restart**. Then visit `yourdomain.com/dh-toolkit` (or whatever URL you configured). You should see the DH Toolkit interface.

If it doesn't load, the two most common issues are:

- **502 or Application Error** — Usually means the startup file or entry point name is wrong. Double-check that startup file is `passenger_wsgi.py` and entry point is `application` (not `app`). Flask convention is `app` internally, but Passenger expects the variable to be called `application`.
- **Module not found errors** — Means dependencies didn't install into the right virtual environment. Go back to the dashboard, copy the activation command, and run `pip install flask numpy Pillow` explicitly.
- **Timeout on large jobs** — If LDA or ForceAtlas2 hangs on big inputs, your host may have a request timeout you can't change. In that case, reduce iterations in the UI (e.g., 100 instead of 300 for LDA) or use smaller datasets on shared hosting.

I followed the steps for the Setup Python App interface. However, I receive an error when trying to view the application. Here is the error log after two attempts to browse to the URL:

```
[ E 2026-02-20 15:17:42.3869 2861749/Tw1
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:17:02.7575 2861749/Tve
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:16:32.3299 2861749/Tuv
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:16:18.2564 2861749/Tup
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:15:01.6037 2861749/Tt2
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:14:48.7640 2861749/Tsm
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:13:07.8928 2861749/Tr8
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
exited prematurely. [ E 2026-02-20 15:12:51.9369 2861749/Tqs
age/Cor/App/Implementation.cpp:218 ]: Could not spawn process for
application /home/dhtoolkit/dh-toolkit: The application process
```

```
exited prematurely. [Fri Feb 20 15:10:46.459982 2026]
[autoindex:error] [pid 2864484:tid 2864557] [client
91.231.89.123:51081] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 15:09:29.206659 2026] [autoindex:error] [pid 2863415:tid
2863493] [client 91.231.89.38:45897] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 15:09:18.179319 2026] [autoindex:error] [pid 2864292:tid
2864409] [client 91.231.89.35:58147] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 15:08:28.237414 2026] [autoindex:error] [pid 2861862:tid
2861947] [client 91.231.89.120:58173] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 15:07:41.053300 2026] [autoindex:error] [pid 2863738:tid
2863833] [client 91.231.89.32:55003] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 15:07:12.062897 2026] [autoindex:error] [pid 2862353:tid
2862437] [client 91.231.89.35:45549] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
```

```
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:48:49.865198 2026] [autoindex:error] [pid 2862591:tid
2862648] [client 146.70.185.32:49514] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:48:49.456402 2026] [autoindex:error] [pid 2862290:tid
2862359] [client 146.70.185.32:25401] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:48:49.097133 2026] [autoindex:error] [pid 2863142:tid
2863260] [client 146.70.185.32:10481] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:48:10.069016 2026] [autoindex:error] [pid 2864484:tid
2864548] [client 130.126.101.170:42231] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:36.698973 2026] [autoindex:error] [pid 2864178:tid
2864257] [client 180.149.3.51:43457] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
```

```
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:33.735965 2026] [autoindex:error] [pid 2862455:tid
2862498] [client 185.246.173.197:41705] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:33.660340 2026] [autoindex:error] [pid 2862520:tid
2862570] [client 104.253.252.180:36538] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:33.462690 2026] [autoindex:error] [pid 2864178:tid
2864269] [client 104.232.199.44:49948] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:33.452135 2026] [autoindex:error] [pid 2861862:tid
2861935] [client 152.39.205.253:41319] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:45:33.245488 2026] [autoindex:error] [pid 2862353:tid
2862433] [client 104.252.110.161:39828] AH01276: Cannot serve
directory /home/dhtoolkit/public_html/: No matching DirectoryIndex
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive [Fri
Feb 20 14:44:30.069651 2026] [autoindex:error] [pid 2861835:tid
2861882] [client 31.97.153.61:57786] AH01276: Cannot serve directory
/home/dhtoolkit/public_html/: No matching DirectoryIndex
```

```
(index.php,index.php5,index.php4,index.php3,index.perl,index.pl,index
.plx,index.ppl,index.cgi,index.jsp,index.jp,index.phtml,index.shtml,i
ndex.xhtml,index.html,index.htm,index.wml,Default.html,Default.htm,de
fault.html,default.htm,home.html,home.htm,index.js) found, and
server-generated directory index forbidden by Options directive
```

The key error is `The application process exited prematurely`, which means Passenger is finding your app but crashing when it tries to start it. The most common cause is that `passenger_wsgi.py` is importing something that fails at startup. Let's diagnose and fix it.

First, figure out the actual crash reason. SSH in and run the app manually inside the virtual environment:

```
# Copy the "enter virtual environment" command from the Setup Python App dashboard,
# it'll look something like:
source /home/dhtoolkit/virtualenv/dh-toolkit/3.12/bin/activate && cd
/home/dhtoolkit/dh-toolkit

# Then try importing the app directly:
python -c "from app import app; print('OK')"
```

That will show you the real traceback. The most likely culprits:

1. NumPy failed to install properly. Shared hosting on CloudLinux sometimes has issues compiling NumPy. Check with:

```
python -c "import numpy; print(numpy.__version__)"
```

If that errors, try `pip install numpy --only-binary=:all:` to force a prebuilt wheel.

2. Pillow is missing a system library. Pillow needs `libjpeg` and `zlib`. On shared hosting you can't install system packages, so try:

```
pip install Pillow --only-binary=:all:
```

3. The `passenger_wsgi.py` has the wrong venv path. When you use Setup Python App, cPanel manages the virtual environment itself — it doesn't put it in `~/dh-toolkit/venv/` like my install script assumed. It typically goes to `~/virtualenv/dh-toolkit/3.12/` or similar. Since cPanel's Passenger activates the venv automatically, your `passenger_wsgi.py` should be simplified. Replace its contents with:

```
import sys, os
sys.path.insert(0, os.path.dirname(__file__))
from app import app as application
```

Remove any venv activation lines — cPanel handles that.

4. The `app.py` runs `argparse` on import. This is actually very likely your problem. When Passenger imports the file, `argparse`may try to parse Passenger's own command-line arguments and crash. Open `app.py` and make sure the argparse block is guarded. Find the section at the bottom that starts with `if __name__ == '__main__':` — everything from `parser = argparse.ArgumentParser` through `app.run(...)` needs to be inside that block. It should look like:

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='DH Toolkit')
    parser.add_argument('--port', type=int, default=5000)
    parser.add_argument('--host', default='127.0.0.1')
    parser.add_argument('--debug', action='store_true')
    args = parser.parse_args()
    app.run(host=args.host, port=args.port, debug=args.debug)
```

That's already how I wrote it, but double-check the file as uploaded — if anything is outside that guard, Passenger will crash because it passes its own arguments that argparse doesn't recognize.

After making changes, go back to the Setup Python App dashboard and hit **Restart**. Passenger caches the old process, so just editing files isn't enough.

If the manual `python -c "from app import app"` test succeeds but Passenger still crashes, the problem is likely a Python version mismatch — confirm that the Python version selected in Setup Python App matches the one where you installed the packages. You can verify by checking which Python the venv uses: `which python` after sourcing the activation command.

What is meant by the following error?

```
Cannot serve directory /home/dhtoolkit/public_html/: No matching
DirectoryIndex
```

That error is separate from the Passenger crash — it's Apache saying that when someone visits your domain's root URL (e.g., `yourdomain.com/`), it looks in `~/public_html/`, finds no `index.html` or `index.php` there, and doesn't know what to serve.

This tells you that your Python app is registered at a subpath (like `yourdomain.com/dh-toolkit`) but you're browsing to the bare domain. The earlier DirectoryIndex errors at 14:44–14:48 are all hits to the root URL before you had the Python app configured, and the Passenger "exited prematurely" errors at 15:12+ are when you started hitting the correct app URL.

So that particular error isn't something you need to fix for the app to work — it's just noise from people (or bots, given the variety of IP addresses in your log) hitting the bare domain.

That said, if you want the app at the domain root instead of a subpath, you have two options. You can go back into Setup Python App and change the Application URL field to `/` (blank or just a forward slash). Or, if you'd rather keep the app at the subpath and just want the root URL to stop erroring, drop a simple redirect into `~/public_html/index.html`:

```
<!DOCTYPE html>
<html>
<head><meta http-equiv="refresh" content="0;url=/dh-toolkit"></head>
<body>Redirecting...</body>
</html>
```

Either way, the DirectoryIndex errors will stop. Focus your debugging effort on the Passenger "exited prematurely" issue — that's the one actually preventing the app from loading.

This is a great draft of the application. The program runs in the web browser successfully using both local and cPanel implementations. The cPanel problem was resolved using solution number 3: "The passenger_wsgi.py has the wrong venv path."

Let's revise each element of the application now.

The mapping application should have more conventional maps. Consider using an existing map that is free to use under Creative Commons, such as one (or several) from the David Rumsey Historical Map Collection. Alternately, a modern Google-Maps style map may be selected or drawn.

The visualization for the mapping application should be interactive. Users should be able to zoom in and out on any part of the map, within the interface. Icons for each location should open a window or dialog box that contains metadata about the location.

The mapping app should be robust to various location naming conventions. Addresses may include street address, city, state, and country in a single string, or each level of the address may be broken out into individual columns. Latitude and longitude may be given or not. In addition to storing coordinates for major cities, include coordinates for countries around the world, and major political sub-divisions within countries, such as states in the United States of America or those in Germany.

The network visualization app should also be robust to various naming conventions. It should automatically search for "source" and "target" columns, but if they are unavailable, there should be an option to select other columns. In addition to source, target, and weight columns there may be additional ones with metadata as well.

As in the mapping app, the network visualization should be interactive and zoomable within the user interface. Clicking on a node should open a window or dialog box containing the node's metadata. Also, the network app should compute statistics like node degree and betweenness centrality. That information should appear in the box for each node as well.

The topic model visualization needs to be reimagined. The upper visualization, in which each topic is visualized horizontally in its own row should be removed. The lower visualization, in which each topic appears in its own box with the top words listed and horizontal bars corresponding to their values in the model, should remain.

Make the topic model visualization interactive as well. There should be three frames side-by-side. The left frame should contain the topic visualization with a list of top words along with their numerical values and horizontal bars. The frame should contain a dropdown menu to select any topic from the model. The middle frame should contain a list of filenames for the top texts associated with the topic. Numerical values and horizontal bars should also be used in that frame as well. The right frame should contain a scrollable window with the text of the top ranked file.

All three apps -- map, network, and topic model -- include the ability to start over with a new dataset. That is, after uploading a csv or set of txt files, users should have the option to upload a new csv or set of txt files and produce new visualizations.

Make the user interface more modern looking. Consider an interface that looks more like Claude, Github, or Google Drive. Avoid serif fonts. Use lines and rounded boxes to visually partition areas of the interface. Use a light-colored background, instead of dark.

The target audience for the desktop application are users with no previous coding experience or experience with the command line. There are two implications of that fact. First, desktop installation for macOS and Windows should be as simple as possible. That means using an *install manager, like a DMG or EXE file, or dragging an app file into the Applications folder. Users should not have to install additional software or programming languages through a separate process. (For Linux desktops, you should install from compressed source tarball. We will assume that Linux users are comfortable with the command line.)*

Second, starting the app should automatically open the window in which the graphical user interface appears. If the app is a standalone application, then it should open an interface to start. If the app runs in the web browser, then it should open the browser window automatically. In that case, clicking the app icon should open a terminal window which automatically opens the browser is acceptable.

For cPanel installation, we will simply install from the Linux source. Make sure that the linux source code is appropriate to run as a web app, in addition to its desktop implementation. Recall that cPanel has tools to install web apps in Python, Ruby, and Node.js. (We used and debugged the Python tool earlier.) If the desktop program ultimately relies on an existing programming language and libraries, then it should be one of those three languages.

This is a major revision touching every component. Let me plan the architecture, then build it systematically.

Architecture decisions:

- **Interactive maps/graphs/topics**: Build as a single-page web app with Leaflet.js for maps, D3.js for network graphs, and custom React-like components for topics — all in one HTML file
- **Map tiles**: Use OpenStreetMap (free, no API key) via Leaflet for the modern Google-Maps-style experience
- **Geocoding**: Massively expand the built-in gazetteer to include countries, US states, German Länder, and many more subdivisions
- **Desktop app**: Use Electron-like approach via PyWebView for a native window, bundled with PyInstaller for DMG/EXE
- **Light UI**: Clean, modern design inspired by GitHub/Claude with sans-serif fonts

Let me start building:Now the main Flask app:Now the big piece — the complete single-file HTML template with Leaflet maps, D3 network, and interactive topic explorer:

[Interrupted due to context limit]

The file was truncated. Let me finish it with str_replace to add the remaining JS:The template file needs to be created in one shot. Let me write it complete to a file:Now create the installer scripts and requirements:Now let's test everything:Here's DH Toolkit v2 in both tar.gz (for Linux/macOS/cPanel) and zip (for Windows). Here's what changed:

Mapping — Replaced the from-scratch rendered map with interactive Leaflet.js over OpenStreetMap tiles. Full zoom/pan, clickable markers with popup metadata for every

CSV column. Gazetteer expanded to 600+ entries covering all countries, US states, German Länder, Canadian provinces, UK regions, Australian/Indian/Chinese/Brazilian subdivisions. Parses composite addresses ("Springfield, IL, USA") by splitting on commas and matching each part.

Network — Now renders interactively with D3.js including zoom/pan and SVG. Clicking any node opens a popup showing degree, weighted degree, betweenness centrality (Brandes' algorithm), plus any extra CSV metadata. Column selection is auto-detected but user-overridable via dropdowns if Source/Target aren't found.

Topics — Removed the horizontal image visualization. Kept the word-bar topic cards and rebuilt as a three-panel interactive explorer: left panel shows topic words with bars (dropdown to switch topics), middle panel ranks documents by relevance to the selected topic with bars, right panel is a scrollable text viewer that loads on click.

UI — Light background, sans-serif system fonts, rounded cards with subtle borders, GitHub/Claude-inspired clean look. Tab navigation across Home/Map/Network/Topics. Every tool has a "New Dataset" button.

Installation — `install_macos.sh` creates a .app bundle that opens Terminal and auto-launches the browser. `install_windows.bat`creates a virtual environment, installs deps, and puts a shortcut on the Desktop. `install_linux.sh` works for both desktop (creates .desktop entry) and cPanel (detects `public_html`, gives Setup Python App instructions). `passenger_wsgi.py` is included pre-built with the simplified import that works with cPanel's managed venv.

Note on Python requirement: macOS and Windows installers check for Python and prompt to install it if missing (Homebrew on macOS, download link on Windows). For a fully zero-dependency DMG/EXE, you'd need to bundle Python itself via PyInstaller or py2app — I can generate those build scripts as a next step if you'd like.

The new version of the application is excellent. There are three major issues at the moment.

First, the network visualization frame is blank after running the ForceAtlas2 algorithm. The numbers of nodes and edges are listed, but the main frame is simply gray without any nodes or edges visible. Debug that code and/or suggest some tests to facilitate debugging.

Second, the Python installation is cPanel fails again. The error log returns the same message as before: "Could not spawn process for application /home/dhtoolkit/dh-toolkit: The application process exited prematurely." Debug that code and/or suggest some tests to facilitate debugging.

Third, the installation of programs on macOS and Windows still requires users to install from the command line. Users should not have to run any commands in the terminal in order to install the *application. Generate the DMG/EXE files with Python bundled into them, along with any dependencies for the programs.*

Let me tackle each issue systematically.

Issue 1: Network Visualization Blank

The most likely cause is that the D3 rendering fires before the container is visible or sized. Let me also check for coordinate/scaling issues.Coordinates look fine. The issue is likely the D3 rendering timing — the SVG container has zero dimensions when `renderNet` fires because the result div was just made visible. Let me also check for a second bug — the `clientWidth`/`clientHeight` issue:Found it. On line 172, `renderNet(r)` is called **before** `document.getElementById('net-result').style.display=''`. So when `renderNet` reads `clientWidth`/`clientHeight` of the container, it's still `display:none` and both dimensions are **0**. The entire graph is drawn into a 0×0 viewport.

Now let me fix all three issues. I'll rebuild the template and fix the `passenger_wsgi.py`, then create the build scripts for DMG/EXE:Now fix issue 2 — the cPanel spawning crash:Let me make the app.py more resilient with better error handling and a diagnostic mode, and update `passenger_wsgi.py` to include proper error logging:Now update `passenger_wsgi.py`with better error reporting:Now issue 3 — create PyInstaller build scripts for macOS DMG and Windows EXE:Now let me also create the cross-platform launcher script that gets bundled, since the batch file approach for generating Python is fragile:Now update the build scripts to use the shared launcher:Now let me update the Windows build script to use the shared launcher too:Also update the `install_macos.sh` launcher to use `--no-browser` flag properly:Let me verify everything still works:All tests pass. Now let me verify the passenger_wsgi diagnostics work:That error is just from using `exec()` in my test — in real Passenger it runs as a file. Let me test it properly:Now package everything:Here's what was fixed and added:

Fix 1: Blank network visualization

The root cause was a rendering order bug. In `generateNetwork()`, the code called `renderNet(r)` *before* setting `display:''` on the result container. D3 read `clientWidth` and `clientHeight` as 0 from the hidden div, so the entire graph was drawn into a 0×0 viewport. The fix swaps the order — the result div is made visible first, then `requestAnimationFrame(() => renderNet(r))` ensures the browser has computed layout before D3 measures the container. A fallback `||800`/`||550` also protects against edge cases.

Fix 2: cPanel "exited prematurely"

Three changes to make the app more resilient in Passenger environments:

- `app.py` now has structured logging to stderr, so Passenger errors appear in the Apache error log with `[DH Toolkit]`prefixes. It also tries multiple upload directory candidates (app-local `tmp/`, system temp, home directory) instead of failing if one isn't writable. The `secret_key` is now deterministic (overridable via `DH_TOOLKIT_SECRET` env var) instead of `os.urandom` which caused session breakage across worker restarts.
- `passenger_wsgi.py` now logs every step of the import process — Python version, whether `engines/` exists, whether flask/numpy/PIL are importable — directly to stderr. If the import fails, it returns a minimal WSGI app that shows the error in the browser instead of just crashing. After redeploying, check your error log for lines starting with `[DH Toolkit]` to see exactly which step fails.
- `argparse` is now imported only inside `main()` (not at module level), eliminating any possibility of Passenger's argv conflicting.

Fix 3: Desktop installers without command line

The project now includes:

- **`build_macos_dmg.sh`** — Run this once on a Mac. It uses PyInstaller to bundle Python + Flask + NumPy + Pillow + all app code into a standalone `DH Toolkit.app`, then wraps it in a `.dmg` with an Applications symlink for drag-to-install. End users just double-click the DMG and drag.
- **`build_windows_exe.bat`** — Run this once on a Windows machine. PyInstaller produces a `dist\DH Toolkit\` folder containing `DH Toolkit.exe` with everything bundled. Zip that folder for distribution — users extract and double-click.
- **`launcher.py`** — Shared entry point that finds a free port, starts Flask, and opens the browser. Works both standalone and when bundled by PyInstaller.

The build scripts must be run on their target OS (PyInstaller cross-compilation isn't supported). The resulting DMG/EXE bundles are self-contained — no Python installation needed by end users.

I am still getting the same error message on cPanel. Give step by step instructions to install through cPanel's python app installation.

Here are step-by-step instructions specifically for cPanel's Setup Python App, incorporating what we've learned from debugging.

Step 0: Upload the files

Upload the entire `dh-toolkit` folder to your home directory via cPanel File Manager or SFTP. Your file structure should look like:

```
/home/dhtoolkit/
├── dh-toolkit/
│   ├── app.py
│   ├── passenger_wsgi.py
│   ├── requirements.txt
│   ├── engines/
│   │   ├── __init__.py
│   │   ├── gazetteer.py
│   │   ├── lda.py
│   │   └── network.py
│   └── templates/
│       └── index.html
```

Step 1: Create the Python App

In cPanel, go to **Setup Python App** and click **Create Application**. Fill in:

- **Python version**: 3.10, 3.11, or 3.12 (whichever is available)
- **Application root**: `dh-toolkit`
- **Application URL**: `dh-toolkit` (or whatever URL path you want)
- **Application startup file**: `passenger_wsgi.py`
- **Application entry point**: `application`

Click **Create**.

Step 2: Install dependencies

After creation, the dashboard shows a command at the top that looks something like:

```
source /home/dhtoolkit/virtualenv/dh-toolkit/3.12/bin/activate && cd
/home/dhtoolkit/dh-toolkit
```

Copy that exact command. Open **Terminal** in cPanel (or SSH in). Paste that command and press Enter. Then run:

```
pip install flask numpy Pillow
```

Step 3: Verify the app loads

While still in that same activated terminal session, run this diagnostic:

```
python -c "from passenger_wsgi import application; print('SUCCESS:', application)"
```

If it prints `SUCCESS: <Flask 'app'>`, the app will work. If it throws an error, that error is your problem — most likely a missing package or a path issue.

If you see an import error for `engines`, it means the directory structure is wrong. Run `ls -la` and make sure you see `app.py`, `engines/`, `templates/`, and `passenger_wsgi.py` all in the current directory.

If you see an import error for `flask`, `numpy`, or `PIL`, the packages didn't install into the right virtual environment. Make sure you pasted the activation command from Step 2 before running pip.

Step 4: Restart the app

Go back to the Setup Python App dashboard and click **Restart**. Then visit `yourdomain.com/dh-toolkit`.

Step 5: If it still fails, check the error log

The updated `passenger_wsgi.py` writes diagnostic lines to the Apache error log. Go to cPanel → **Errors** (or **Metrics** → **Errors**) and look for lines starting with `[DH Toolkit]`. They will tell you exactly which step failed — whether Python can't find the engines directory, whether a dependency is missing, or whether something else is wrong.

You can also check from the terminal:

```
tail -50 ~/logs/error.log 2>/dev/null || tail -50 /var/log/apache2/error.log 2>/dev/null
```

Look for the `[DH Toolkit]` lines specifically. They appear in order: `passenger_wsgi.py loading` → `app_dir` → `Python version` → `engines/ found` → dependency checks → `Application loaded OK` (or `FATAL` with a traceback). Whatever line is last before the crash is your clue.

Common cPanel gotchas:

The most frequent cause of "exited prematurely" that we haven't covered yet is a **Python version mismatch**. If you selected Python 3.12 in Setup Python App but the virtual environment somehow points to a different version, Passenger will crash silently. Verify with:

```
source /home/dhtoolkit/virtualenv/dh-toolkit/3.12/bin/activate
which python
python --version
python -c "import flask; print(flask.__version__)"
```

All of those should work and show the same Python 3.12 (or whichever version you chose). If `which python` points somewhere unexpected like `/usr/bin/python3.6`, the venv is broken — delete it in the Setup Python App dashboard and recreate.

Step 3 throws a recursion error. Here is the end of the error message:

```
File "passenger_wsgi.py", line 13, in load_source
loader.exec_module(module) File "passenger_wsgi.py", line 16, in
<module> wsgi = load_source('wsgi', 'passenger_wsgi.py')
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "passenger_wsgi.py",
line 13, in load_source loader.exec_module(module) File
"passenger_wsgi.py", line 16, in <module> wsgi = load_source('wsgi',
'passenger_wsgi.py') ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File
"passenger_wsgi.py", line 12, in load_source module =
importlib.util.module_from_spec(spec)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ File "<frozen
importlib._bootstrap>", line 819, in module_from_spec File "<frozen
importlib._bootstrap>", line 798, in _init_module_attrs File "<frozen
importlib._bootstrap>", line 638, in cached File "<frozen
importlib._bootstrap_external>", line 618, in _get_cached
RecursionError: maximum recursion depth exceeded
```

That's the problem. Your `passenger_wsgi.py` on the server isn't the one from our project — it's a cPanel auto-generated wrapper that tries to load itself recursively. cPanel's Setup Python App sometimes creates its own `passenger_wsgi.py` that contains a `load_source` bootstrapper, and when it tries to load `passenger_wsgi.py` again, it re-executes the same `load_source` call infinitely.

Here's how to fix it:

Step 1: Check what's actually in the file

In the cPanel terminal (with the venv activated), run:

```
cat passenger_wsgi.py
```

You'll likely see something like this at the top — this is **not** our file, it's cPanel's auto-generated one:

```
import importlib.util
import os
import sys

def load_source(modname, filename):
    spec = importlib.util.spec_from_file_location(modname, filename)
    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)
    return module

wsgi = load_source('wsgi', 'passenger_wsgi.py')   # ← this line causes the recursion
application = wsgi.application
```

That last section is loading `passenger_wsgi.py` from within `passenger_wsgi.py` — infinite recursion.

Step 2: Replace it with our version

In the cPanel terminal, replace the file entirely:

```
cat > passenger_wsgi.py << 'EOF'
import sys
import os

app_dir = os.path.dirname(os.path.abspath(__file__))
sys.path.insert(0, app_dir)

from app import app as application
EOF
```

Step 3: Verify it works

```
python -c "from passenger_wsgi import application; print('SUCCESS:', application)"
```

This should now print `SUCCESS: <Flask 'app'>` with no recursion.

Step 4: Restart

Go to Setup Python App in cPanel and click **Restart**.

The key lesson: when cPanel's Setup Python App creates the application, it sometimes overwrites `passenger_wsgi.py` with its own bootstrapping wrapper. Any time you recreate or modify the app through the cPanel UI, check whether your `passenger_wsgi.py` got replaced. If it did, run the `cat >` command from Step 2 again.

The network visualization takes a long time to generate. Is that due to the forceatlas2 algorithm or computing the betweenness centrality of each node?

Almost certainly the betweenness centrality. ForceAtlas2 is O(n log n) per iteration with Barnes-Hut, but Brandes' betweenness centrality algorithm is O(n × m) where n is nodes and m is edges — it runs a full BFS from every single node in the graph. For a network with 1,000 nodes and 5,000 edges, that's 5 million operations. For 10,000 nodes it becomes prohibitive.

You can test this yourself by temporarily commenting out the centrality call. In `app.py`, find the `network_generate` function and change:

```
stats = compute_network_stats(nodes, edges, weights)
```

to:

```
stats = {n: {'degree': 0, 'weighted_degree': 0, 'betweenness': 0} for n in nodes}
```

If the network loads quickly after that, betweenness is your bottleneck.

There are a few options for fixing it. The simplest is to approximate betweenness by sampling — instead of running BFS from every node, pick a random subset. Brandes' original paper discusses this. For DH use cases where exact centrality values rarely matter, sampling 100–200 pivot nodes gives a good approximation at a fraction of the cost.

Want me to implement the sampled approximation? I'd cap it at something like `min(n, 200)` pivot nodes and note in the UI that betweenness is approximate for large graphs.

The program still took a long time to run. In any case, remove the betweenness centrality computation from the program and generate the installation files again. Be sure that the `passenger_wsgi.py` contains the code we copied into the file after the debugging step:

```
import sys import os app_dir = os.path.dirname(os.path.abspath(file))
sys.path.insert(0, app_dir) from app import app as application
```

> Now remove betweenness from the frontend popup:Now replace `passenger_wsgi.py` with the exact working version:Now test everything:8 seconds for 200 nodes at 100 iterations — that's the ForceAtlas2 itself. Let me check if we can speed that up too:Barnes-Hut isn't helping because the Python overhead in the tree operations dominates. Let me vectorize the core loop with NumPy:Now benchmark the vectorized version:From 8 seconds to 0.23 seconds for 200 nodes — a **34x speedup**. Now package:Here's what changed in v2.2:
>
> **Betweenness centrality removed.** The `compute_network_stats` function now returns only degree and weighted degree. The Brandes algorithm and its O(n×m) BFS-from-every-node cost are gone. The node popup in the UI no longer shows a betweenness field.
>
> **ForceAtlas2 vectorized with NumPy.** The entire layout loop — repulsion, attraction, gravity, force application — is now pure NumPy array operations with no Python-level `for i in range(N)` loops. Benchmarks on the same 200-node, 400-edge, 100-iteration test: **8 seconds** → **0.23 seconds** (34x faster). 500 nodes runs in about 1.4 seconds. The old Barnes-Hut quad-tree code (which was slower than brute-force in Python due to tree-traversal overhead) has been replaced entirely by vectorized all-pairs repulsion via broadcasting, which is faster for graphs up to several thousand nodes.
>
> **`passenger_wsgi.py` is the exact 5-line version** that you confirmed working on cPanel after debugging:
>
> ```
> import sys
> import os
> app_dir = os.path.dirname(os.path.abspath(__file__))
> sys.path.insert(0, app_dir)
> from app import app as application
> ```
>
> Be aware that cPanel's Setup Python App may overwrite this file with its own bootstrapping wrapper when you recreate or modify the application through the UI. If you get the recursion error again after making changes in the cPanel dashboard, re-run the `cat >` command from our debugging session.