DocGen Amendments - Streamlit

The docgen feature is set up to render a document generator interface, which is served using both a React application and a Streamlit app. The integration involves using page.tsx to display the Streamlit interface within the React component using an iframe.

1) **Initial Integration of DocGen Feature:**

   Goal: Set up a new **DocGen** Feature for generating **GPLC Documents** and **integrate** it into **Danswer** but with additional **features**.

   Implementations Taken:

   Created a new **route** for **'docgen'** under the **danswer/web/src/app** directory:

   docgen/page.tsx: This file utilizes Next.js and manages the React-based component that acts as the **frontend** for the **docgen** feature.

   Purpose: 1) Its core purpose is to embed the **streamlit** application within an **iframe** to display its content at http://3.24.137.188:8502/. The React components (DocgenPage) returns a div that fills the viewport (100vh) and contains an iframe.

   2) The iframe loads the content from the streamlit server URL (http://3.24.137.188:8502/), providing a seamless **integration** of the **Streamlit** app into the existing **React frontend**.

   docgen/page_backup.tsx: This file is a backup version of the **page.tsx** file, potentially used for a different **IP method** of integrating the Streamlit app.

   Status: Currenlty **commented** out and not in use.

   docgen/start-react.sh: This script is used to start the **React development server** using **PM2** (process manager) to run we should be in the **root** directory which is **danswer/web** and run **pm2 start src/app/docgen/start-react.sh --name react-app**.

   Status: **Working** and to see the status run **pm2 status** before looking at status its a must to run **pm2 save**.

   Purpose: Changes the directory to the **Root** of the React app **(danswer/web)**, Runs **npm run dev - - -p 3005(port forwarding to 3005)**, which starts the React app on port **3005**.

2) **Streamlit Directory:** The **streamlit directory** contains all the necessary files to run a **Streamlit** app, which is integrated in our **Danswer** application for generating documents.

Key File Changes:

streamlit_app.py: The **main** file that **Streamlit** runs to display the document generation interface.

main.py: Contains the core **logic** for **generating documents** and interacting with external tools (e.g. **ChromaDB**).

start-streamlit.sh: A script to start the Streamlit server on specified port (**8502**) first navigate to the Streamlit directory **danswer/streamlit** run **pm2 start start-streamlit.sh –name streamlit-app.**

Status: Working and to see the status run **pm2 status** before looking at status its a must to run **pm2 save**.

Changes: Use of **st.session_state** to manage the state of the document generation process and to **Keep track** of whether the generation is in progress (**is_generating**), allowing the user to **start** and **stop** the generation dynamically.

Added two buttons, **"Init Document Generation"** and **"Stop Generation"** it allows the user to start and stop the document generation process.

Real Time Display o**f the Results** in order to display the generated output **incrementally** its purpose is to update the **UI** with new results as they are generated instead of **waiting** for the entire progress to **complete.**

Uses a for **loop** to iterate over the results **yielded** by **main_func** in **main.py**. The **loop** allows the app to update the UI incrementally with new **paragraphs** and **clauses** as they are generated. Created an **st.empty()** function for **written clauses (editable)** and **clauses generated (editable)** to dynamically update the content and added **set()** to remove any **duplicates**.

Displays **Written clauses** and **Clauses generated** in **text areas**, allowing user to edit them. Uses **Streamlit's spinner** to show a **loading** animation while the content is being generated.

3) **Modifications in main.py:**

Changes: Use of **yield** in **main_func** to return **intermediate** results rather than waiting for the entire process to complete. Instead of returning all the **results** at **once**, the function uses **yield** to provide **partial results** (**output_container**, **titles**) after processing each title. This allows **Streamlit** app to display results incrementally.

Processes each title iteratively **for title in titles: result = run_inference(prompt) clean_result = clean_output(result) yield output_container, titles** and yield the results after each iteration.

The **output_container** and **summary_container** are updated **incrementally** and **yielded** to provide immediate feedback to the user

clean_output, **get_titles**, **get_summary** from **utils** are the **Helper functions** for processing and managing generated content otherwise it looks **clustered**.

4) **Modifications in FunctionalWrapper.tsx:** The **primary** purpose of this file is to **redirect** the user to a specific **URL** or path when this component is rendered just as **Search** and **Chat** our objective now is to be Redirecting to New IP Address for **docgen** tab Feature

   Path: danswer/web/src/app/chat/shared_chat_search/FunctionalWrapper.tsx

   Changes: Updated the code to redirect the user to the **docgen** feature hosted on the specified address (http://3.24.137.188:3005/docgen) otherwise if you simply select docgen tab the **toggle** immediately shifts to **docgen** and makes a **direct request** to the server at IP (http://3.24.137.188/docgen) on port **3005** specifically asking for **/docgen** path. It ensures then when the **'Docgen'** tab is clicked, the user is redirected to the correct location where the docgen feature is **hosted**.

   It uses **useEffect Hook** to execute the **redirection logic** immediately after the component is **mounted window.location.href** sets the browser's location to the specified **URL**.

   The **handleKeyDown** function checks if either **Ctrl** Key (**event.ctrlKey**) or the **Meta key** is **pressed** and also inside the **switch** statement it checks the specific key that was pressed in case of chat, search and docgen its **case "d", case "s", case "a"** as **Ctrl "a"** typically selects **all text** in a **text area event.preventDefault()** makes sure to **prevent** it.

   Since the component is **solely** responsible for redirection, there is no need to **render** any content

   Note: if the server's SSH path changes, the application will need to be updated accordingly to reflect the new address (if running on localhost).

5) **Modification in Package.json:** as our **sole** purpose was to render the application not only on **localhost** but also **successfully** running the application **open source** on **external network** as well these are the following changes:

   Changes: Updated the scripts to make the **Next.js** application accessible on all **network interfaces (0.0.0.0)** and a **specific port (3000)** to ensure it can be accessed **publicly**. **dev script** was changed to run the **development server** on all **network interfaces (-H 0.0.0.0) and port (-p 3000)** Ensures that the Next.js app can be accessed **locally** and **publicly** over the network using the **server's IP** address during development.

6) **Modifications in docker-compose.dev.yml:**

   Added a **ports configuration** for the **web_server** service to expose the **Next.js** on port **3000**. This exposes port 3000 for the **web_server service**, allowing the **Next.js** app to be accessed externally via port **3000**.

   Note: Adding the ports in both the package.json and docker-compose.dev.yml files was necessary to ensure that the Next.js application and other services can be accessed externally, both during development and when using Docker.

7) **Summary of All the Changes:**

   1) **Public Accessibility:** The application is accessible both locally and publicly over the network.
   2) **Dynamic Content Updates:** Streamlit app updates content dynamically, providing a better user experience.
   3) **Automation:** Scripts automate the starting of necessary services, reducing manual steps.
   4) **Flexible Redirection:** Redirection logic ensures the correct navigation paths for users.
   5) **Access the Site:** To access the site please click on the given URL (http://3.24.137.188/chat) or run the SSH Server **("sudo ssh -i "/mnt/c/Users/ibrah/Downloads/techpeek_1.pem" ubuntu@ec2-3-24-137-188.ap-southeast-2.compute.amazonaws.com -L localhost:3000:localhost:3000")** with the mentioned port forwarding it to 3000 and access it via (http://localhost:3000/search).
   6) **Relocated Docgen Amendments** into the **Enterprise Site** here's the link for it: (http://54.206.187.178/chat) to directly view **Docgen** here it is (http://54.206.187.178/docgen).

Note: All the original codes have been backed up, so there's no need to worry.

- **Ibrahim Sultan**