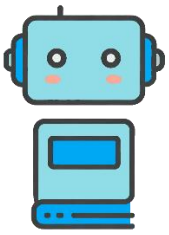

CAIN

Phung Minh Chien | Nguyen Danh Huy | Ma Khoa Hoc

link to prototype

Target audience

CAIN is designed for researchers who usually work and find information from documents. We determined our target audience after hearing from multiple interviewees about how they lacked a specialized chatbot for working multiple documents at a time without reuploading them in each conversation like ChatGPT.

Design tools

Figma was used as our main design and wireframing tool for our medium-fidelity prototype. Our solution emulates a desktop web application but can still scale elements to different sizes. We also used Iconify for the initial sketching of our icons and logos.

Operating instructions

In this section, we detail screens associated with each task flow and describe screen interactions.

Clicking on any space within the interface will show bounding boxes (“hotspots”) which indicate valid interactions with the app. All interactions are tap interactions. Users can interact with predetermined buttons that lead them through three task flows, and also for general navigation of the prototype.

Task flows:

Search notebook by name using search bar

- Use the search bar
    - Tap on the search bar with “Search” placeholder at the center of it
    - The first tap will show how the UI display with results
    - The next tap will show how the UI display with no result

Chat to ask about chosen files

- Go to notebook management page
    - Click on your notebook field to open notebook management page for that notebook (only My notebook 1 available for this med-fi)
- Interaction with sidebar
    - Tap on 3-line button on the top left to expand the sidebar
    - Tap on “pmc.pdf” checkbox to choose that file for your conversation
    - You can also click on “All” checkbox to check all files in notebook
- Interaction with chat box
    - After choosing the files, tap on chat box to generate the chat message
    - Tap send button which has arrow up icon
    - After 1 second, a bot message will appear
- Back to notebook page
    - Tap on the button has right arrow icon to navigate back to notebook page

Upload File to Notebook

- Go to notebook management page
    - Click on your notebook field to open notebook management page for that notebook (only My notebook 1 available for this med-fi)
- Main action for uploading file
    - Tap on 3-line button on the top left to expand the sidebar
    - Tap on “+ Add document” button to open dialog
    - Tap on dashed zone to open system dialog
    - Tap on “PhungMinhChien_20213565.pdf” to choose it to upload or “Cancel” to close the system dialog
    - If you tap on “PhungMinhChien_20213565.pdf”, a pop up appear on the top of UI and the file is recently added to the sidebar
- Back to notebook page
    - Tap on the button has right arrow icon to navigate back to notebook page

Limitations

In this section, we detail three major limitations of our prototype.

We have limited the number of clickable options on a screen to one so that only the first option can be used to complete a task flow. This applies to notebook item in the

notebook page, uploaded file item in notebook management pages and the file to choose in system dialog.

There is no functionality for the search bar on the notebook page because this requires implementation of a search algorithm and a backend data system to store information.

Users are not able to type message in the prototype because the prototype cannot ingest typed user input and perform the necessary logical calls to change the displayed information.

Wizard of oz techniques

In this section, we detail what Wizard-of-Oz techniques we used to simulate automation and our rationale behind using these in our prototype.

The user magically receives a message after sending the message from the chat box, which has been hard-coded.

- **Why?** This is because it would be difficult to simulate for the chatbot that requires NLP frameworks

The searching for notebook by name function in the first task and their corresponding results are hard-coded.

- **Why?** This is because we do not have a backend system to dynamically search and display the data of the notebook list (currently nonexistent) or have the necessary logic to apply the query for notebook's name with the prototype

Hard-coded items

In this section, we go into more detail about fixed data built into the prototype design and our rationale behind hard-coding.

Many of the wizard of oz components listed above are hard-coded to make this prototype function. The user's notebooks, files and messages are pre-set to display on the notebook and notebook management page, which are also hard-coded.

- **Why?** We did this because the prototype cannot load customize data of user from a backend server in this prototype