ui.py


```
Function for processing file
Processes the file (File's path)
and generates c_count number of cards
:param file: String representing file path
:param c_count: Input number of anki cards
```

when testing use searchquery[:10 or less]. Still working on better threading to get faster results

selecting random customised number of flash cards

```
Function for processing url
Processes the url (web url)
and generates c_count number of cards
:param url: String representing URL path
:param c_count: Input number of anki cards
```

```python
import os
from PIL import ImageTk
from user_cli import *
from tkinter import filedialog
from tkinter import *
from tkinter import filedialog, Tk, ttk, Label, Button, StringVar, OptionMenu, messagebox, Text
from docx2pdf import convert
import sys
import threading
import gpt_prompting as gp
import gpt4 as gp4
from tkinter.ttk import Progressbar
import json
import random

from docx2pdf import convert

sys.path.append(
    '/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages')









def process_(file, c_count):




    try:
        update_status("Processing file...")
        lect_name = file.split("/")[-1].split(".")[0]

        if file.split("/")[-1].split(".")[1] == "pdf":
            pass
        elif file.split("/")[-1].split(".")[1] == "docx":
            convert(file,os.path.join("uploads",lect_name+'.pdf'))
            file = file[:-5] + ".pdf"
        elif file.split("/")[-1].split(".")[1] == "pptx":
            template = f"soffice --headless --convert-to pdf {file}"
            os.system(template)
            file = file[:-5] + ".pdf"

        raw_data = extract_words(file)
        raw_data = text_to_groupings(raw_data)
        keyword_data = wp.extract_noun_chunks(raw_data)
        keyword_data = wp.merge_slide_with_same_headers(keyword_data)

        keyword_data = wp.duplicate_word_removal(keyword_data)
        search_query = wp.construct_search_query(
            keyword_data)
        if source_choice.get() == "Google":
            with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:


                results = executor.map(
                    get_people_also_ask_links, search_query[:c_count])
        elif source_choice.get() == "GPT":
            with concurrent.futures.ThreadPoolExecutor(max_workers=10) as executor:
                results = executor.map(
                    gp.get_gpt_answers, search_query[:c_count])


        results_new = [qapair for result in results for qapair in result]
        results_new = random.sample(results_new, int(c_count))
        auto_anki_model = get_model()
        deck = get_deck(deck_name=lect_name)
        for qapair in results_new:
            question = qapair["Question"]
            answer = qapair["Answer"]
            qa = add_question(
                question=f'{question}', answer=f'{answer}', curr_model=auto_anki_model)
            deck.add_note(qa)
        add_package(deck, lect_name)
        messagebox.showinfo(
            "Hurray!", "The Anki deck has been created successfully.")
        update_status("File processed successfully.")
    except Exception as e:
        messagebox.showerror("Error", str(e))


def process_url(url, c_count):




    try:
        update_status("Processing URL...")
        results = gp4.get_gpt_link_answers(url, c_count)
        results_json = results.replace("'", '"')
        results_list = json.loads(results_json)
        auto_anki_model = get_model()
        lect_name = url.split("/")[-1]
        deck = get_deck(deck_name=lect_name)
        for result in results_list:
            question = result["Question"]
            answer = result["Answer"]
            qa = add_question(
                question=f'{question}', answer=f'{answer}', curr_model=auto_anki_model)
            deck.add_note(qa)
        add_package(deck, lect_name)
```

Function to call process_url

You might want to add some validation for the URL here

Function to show finish message

progress['value'] = 0

Function for opening the file explorer window

file = filedialog.askopenfilename(initialdir="/",title="Select a File",filetypes=
(("Text files",".*txt*"),("all files",".")))

Function to update sattus in TkInter

Create the root window

window.minsize(width=450, height=450)

Set window title

Set window size

Configure the grid to be responsive

set logo

Set the theme to the default theme

Status

Add a text field for URL input

```
            update_status("File processed successfully.")
        except Exception as e:
            messagebox.showerror("Error", str(e))


    def process_link():
        url = url_input.get()
        c_count = cards_count.get()
        if url:


            process_url(url, c_count)
        else:
            messagebox.showerror("Error", "Please enter a valid URL")


    def on_finish():


        messagebox.showinfo(
            "Success", "The Anki deck has been created successfully.")


    def browseFiles():


        file = filedialog.askopenfilename(parent=window, title="Choose a file", filetypes=[
                                          ("Doc file", "*.docx"), ("Pdf file", "*.pdf"), ("PowerPoint file", "*.pptx")])

        if file:
            text_box = Text(window, height=10, width=50, padx=15, pady=15)
            text_box.insert(1.0, file)
            text_box.tag_configure("center", justify="center")
            text_box.tag_add("center", 1.0, "end")
            text_box.grid(column=0, row=3)
            c_count = int(cards_count.get())
            process_(file, c_count)


    def update_status(message):
        status_label.config(text=message)
        window.update_idletasks()


    window = Tk()


    window.config(background="#515A5A")


    window.title('Auto-Anki')


    window.geometry("500x550")

    canvas = Canvas(window, bg='#515A5A')


    number_of_rows = 7
    number_of_columns = 3

    for i in range(number_of_rows):
        window.grid_rowconfigure(i, weight=1)

    for j in range(number_of_columns):
        window.grid_columnconfigure(j, weight=1)

    canvas.grid(row=0, column=0, rowspan=number_of_rows,
                columnspan=number_of_columns, sticky='nsew')


    logo = ImageTk.PhotoImage(file='code/Auto_Anki_Logo.jpg')
    logo_label = Label(image=logo)
    logo_label.image = logo
    logo_label.grid(column=0, row=0, columnspan=3)

    instructions = Label(
        window, text="    Select file to upload: ", font="Raleway")
    instructions.grid(column=0, row=2, sticky='w')

    button_explore = Button(window,
                            text="Browse Files",
                            command=browseFiles)

    source_choice = StringVar(window)  # Variable to hold the choice
    sources = ["Google", "GPT"]  # List of choices
    source_choice.set(sources[0])  # Set default value to Google

    source_dropdown = OptionMenu(window, source_choice, *sources)
    source_dropdown_label = Label(
        window, text="    Choose an API source:", font="Raleway")

    source_dropdown_label.grid(column=0, row=1, sticky='w')
    source_dropdown.grid(column=1, row=1)


    style = ttk.Style(window)


    style.theme_use('default')
    style.configure('TProgressbar', thickness=10)


    status_label = Label(window, text="Ready", bd=1,
                         relief="sunken")  # , anchor="w")
    status_label.grid(column=0, row=5, columnspan=3,
                      sticky="we", padx=(10, 10), pady=(10, 10))
    status_label.config(justify="center")

    button_exit = Button(window,
                         text="Exit",
                         command=exit)

    button_explore.grid(column=1, row=2)
    button_exit.grid(column=0, row=6, columnspan=3)


    url_input = Entry(window, width=30)
    url_input.grid(column=1, row=3)
```

Add a text field for no.of flash cards input

```
instructions2 = Label(
    window, text="    Number of flash cards: ", font="Raleway")
instructions2.grid(column=0, row=4, sticky='w')
cards_count = Entry(window, width=5)
cards_count.grid(column=1, row=4)


instructions1 = Label(
    window, text="    Process URL: ", font="Raleway")
instructions1.grid(column=0, row=3, sticky='w')
```

Add a button to process the URL

```
button_process_url = Button(window,
                            text="->",
                            command=process_link)
button_process_url.grid(column=2, row=3)
```

Let the window wait for any events

```
window.mainloop()
```