ASC
A.I. Study Companion
User Manual

Team Members:

Seth Morgan

Cameron Calhoun

Jonathan Buckel

Gage Keslar

Professor:

Dr. Weifeng Chen

Course:

Senior Project II: Software Engineering

University:

Pennsylvania Western University, California Campus

Department of Computing and Engineering Technology

Instructors Notes:

Table of Contents:

Table of Contents

Project Overview

The AI Study Companion (ASC) is an online learning tool designed to support students in strengthening their academic understanding through personalized, AI-assisted study. By integrating adaptive question generation and intelligent progress tracking, ASC offers a dynamic alternative to traditional study methods. The platform emphasizes accessibility, flexibility, and learner autonomy, making it a valuable resource for students at various educational levels. ASC aims to enhance how students engage with academic material, offering a smarter and more efficient path to subject mastery.

Motivation

Effective study habits are essential for academic success, yet many students face significant barriers when it comes to studying efficiently. Traditional methods such as study groups or tutoring sessions, while beneficial, are often hindered by scheduling conflicts, limited availability, or financial constraints. These challenges can leave students without the support they need to reinforce and retain critical knowledge.

The AI Study Companion (ASC) seeks to address this gap by providing a flexible, accessible, and intelligent study tool. Our goal is to assist students in reinforcing subject-matter knowledge independently and efficiently through the integration of modern AI technologies.

Solution

Upon accessing ASC, users choose a subject of interest to begin practicing. To adapt to individual learning levels, ASC incorporates an Elo-based scoring system—widely used in competitive ranking scenarios—to track user performance. All users begin with a baseline score of 800, which increases with correct answers and decreases with incorrect ones. This score is used to automatically adjust the difficulty level of subsequent questions, ensuring an engaging and appropriately challenging experience as the user progresses.

To enhance long-term retention and support targeted review, ASC also offers a flashcard feature. Users can save challenging or incorrectly answered questions as digital flashcards, which can later be reviewed within the platform or exported as a formatted PDF for offline study. This feature encourages repetition and reinforcement of difficult material, aiding in knowledge retention and mastery over time.

In addition, ASC includes robust support for progress tracking and portability. Users can export their current progress—including Elo scores, question history, and saved flashcards—as a JSON file. This file can later be imported to resume studying seamlessly across different sessions or devices. This makes ASC especially useful for students with varying study schedules or those who wish to study across multiple platforms.

By combining adaptive AI-generated content, performance tracking, customizable study aids, and portable data management, ASC delivers a comprehensive and modern study solution that meets the evolving needs of today's learners.

Community Implications

Intended Audience

ASC is primarily intended for students who wish to enhance their understanding and retention of academic content across a variety of subjects. Whether preparing for exams, reviewing classroom material, or simply working to master a topic, ASC empowers learners with AI-generated questions, adaptive difficulty scaling through the Elo rating system, and the ability to save challenging questions as flashcards or export them in a PDF format.

By giving students control over their study experience and offering tools to reinforce difficult concepts, ASC supports more personalized and effective learning. It's intuitive interface and subject-based structure make it accessible for users of varying educational backgrounds, from middle school students to college-level learners.

Secondary Audience

In addition to students, ASC is designed with the broader educational community in mind. Educators, including teachers, tutors, academic advisors, and parents can leverage ASC to support student learning in structured or informal settings. For example, a teacher may recommend ASC to provide additional practice outside the classroom, while a tutor could use the exported JSON progress files to track and tailor instruction for individual learners.

Parents and guardians, especially in homeschool or hybrid learning environments, may also use ASC as a resource to support their child's academic development. With features like flashcard generation and performance tracking, ASC offers meaningful insight into student growth while promoting accountability and engagement.

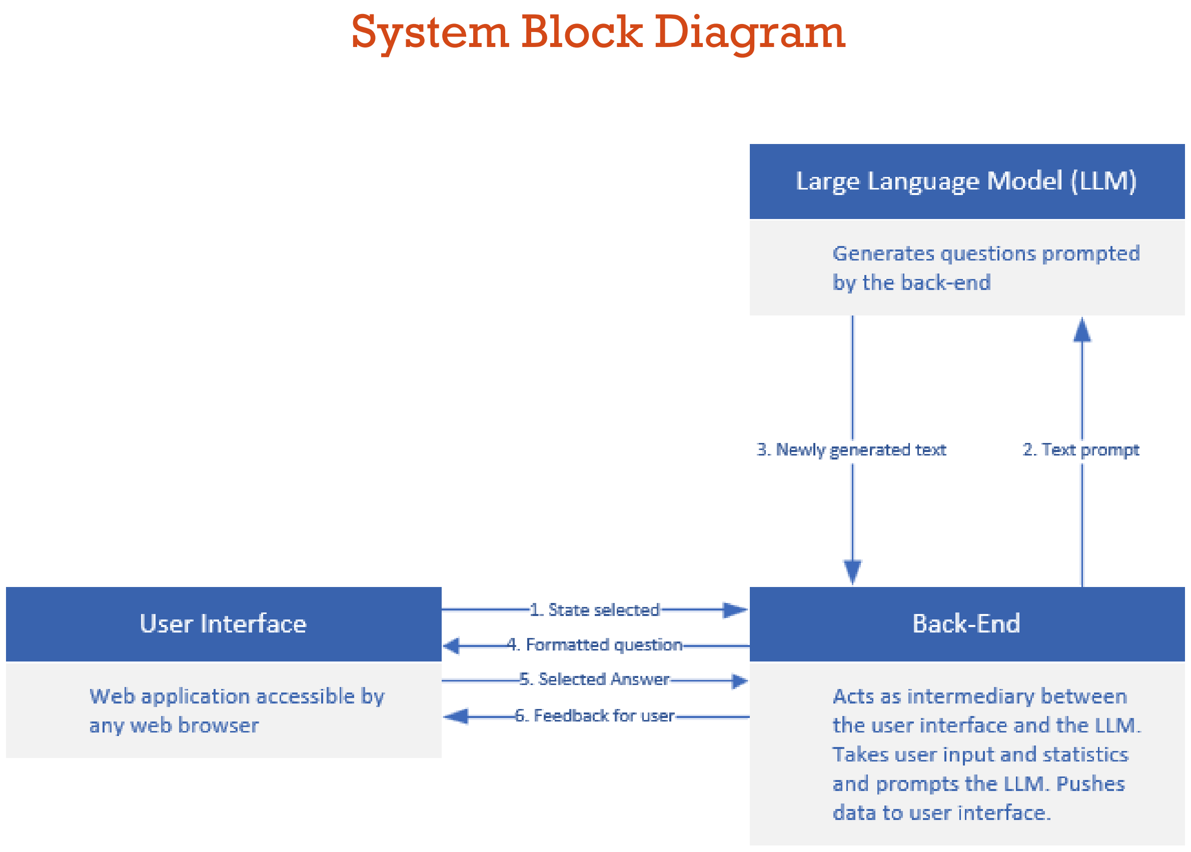

System Block Diagram
Large Language Model (LLM)
Generates questions prompted by the back-end
3. Newly generated text
2. Text prompt
User Interface
Web application accessible by any web browser
1. State selected
4. Formatted question
5. Selected Answer
6. Feedback for user
Back-End
Acts as intermediary between the user interface and the LLM. Takes user input and statistics and prompts the LLM. Pushes data to user interface.

Fig 1.1 System Block Diagram

The System Block diagram has remained unchanged from our design document. It contains three code bases, the user interface, backend, and LLM. Following along the diagram, the user will select the state, typically being flashcards or study questions. The backend sends a text prompt to the LLM and receives a message back. The backend formats the response depending on the current state. If the state selected is the study page, then the user will select answers and the backend will send feedback on the choice made by the user.

Project Implementation Details

Design Document Differences

Changes from our intended design were limited to removing the use of Docker. This was due to technical issues between team member devices, preventing installation on Linux specifically. Docker also was to add to our budget, considering we are a small team with limited resources, cutting Docker would prevent monetary strain on the team.

Implementation Challenges

The ASC team was faced with many challenges that had to be resolved in a head on approach. Some of these challenges included, Frontend/Backend hosting service and generating/parsing Math questions.

Front-end and Back-end hosting was always a necessity when building a webpage application. Our original plan was to containerize both front-end and back-end into a Docker image, but as stated previously we had to move away from Docker. Our solution was to use two separate services to hold the source code for the project. Our front-end was built using React and is now being contained by Vercel. The backend is created with FastAPI with endpoints exposed on Render. These services now work together to run ASC efficiently and prove to be easily deployable and scalable.

The ASC team was prepared for the Math subject class to require extra support, there were many systems put in place to ensure that Math questions were being generated, solved, and parsed correctly. Large Language Models were designed specifically for processing Language, while having the ability to solve math questions does not mean that it can do so accurately. What can be done is to make sure that prompts sent to OpenAI according to that will facilitate the needs for our solving function. We then validate the correctness of the LLM to ensure that the correct answer is available. If an issue arises such as no correct answer present or correct answer is present, but OpenAI designated as incorrect, then ASC will ensure that the correct answer is provided and will reward the user's choice. The Math subject is the only subject that will not provide an explanation to the answer, as the LLM is not reliable to produce an explanation.

Use of Software Engineering Principles

Agile Development

In the creation of ASC, it was of paramount importance that our work was structured in order to hit all target goals by deadlines. When looking at methodologies used in industry, the one that stands out above all others is the Agile methodology. Agile is characterized by adaptability, flexibility, collaboration, and responsiveness to change. Agile allows for frequent feedback loops, such that plans, priorities, and deliverable goals can be adjusted regularly based on feedback from previous sprints. We worked in weeklong sprints, where at the start of each sprint we would discuss what was worked on in the previous sprint. This gave immediate feedback on features implemented and would allow early refinement or removal of features that did not quite hit the mark. After the discussion of the previous sprint, a discussion will take place on goals for the following week. Tasks would be delegated, and the team would begin working on their action items for the week. This strategy allowed us to be flexible when implementations fail, and responsive to criticism towards our current designs. The final version of ASC is an iterative design with around fifteen weeks' worth of feedback directly influencing the product it is today.

User Manual

This segment is dedicated to the operation and handling of the ASC webpage. All functions and handling will be done through the internet browser and will ensure that the user can utilize ASC to its fullest potential.

Import/Exporting User Data

As stated previously, users have the option to transfer their data between sessions and devices. The user will be able to import data utilizing the "Import" button on either the home screen or header tab as seen below (Fig. 2.1).

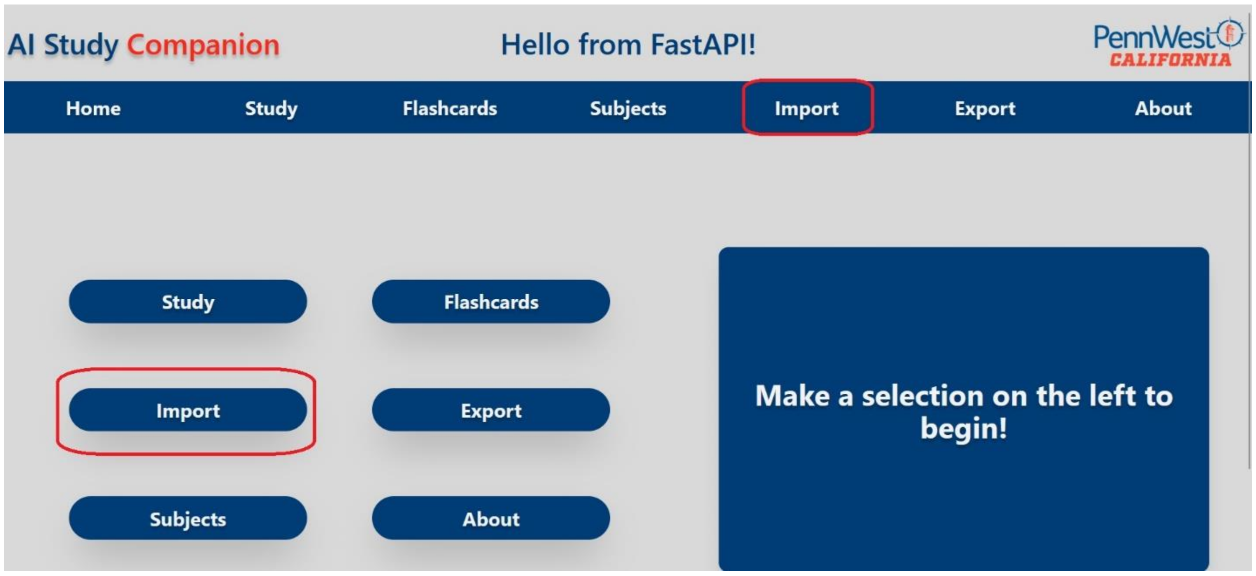
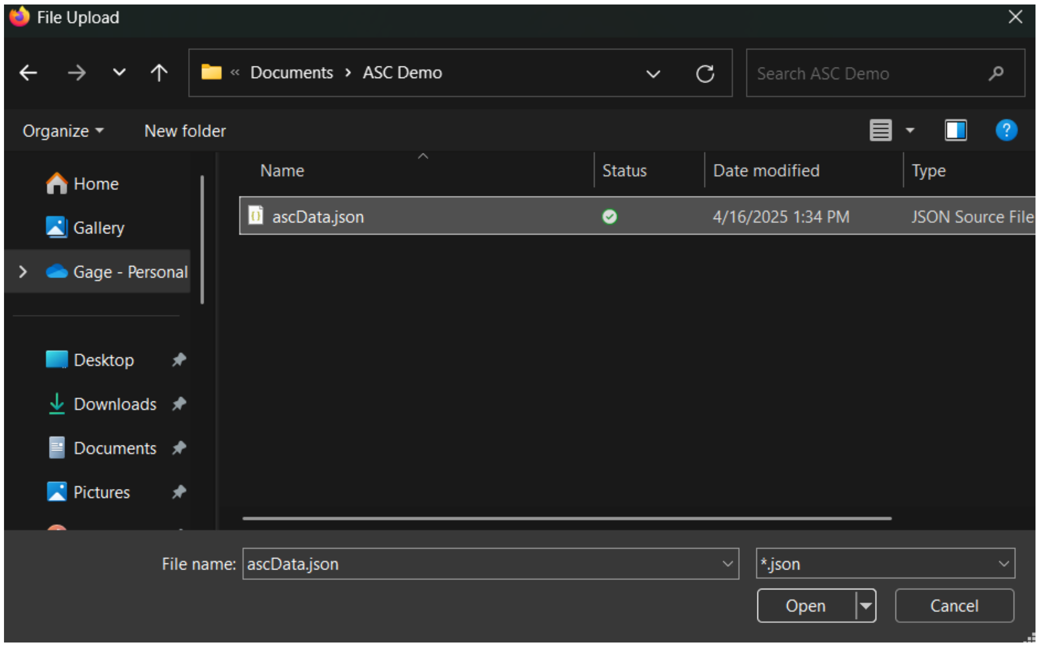


Fig. 2.1 Import Subject Selection

This function will open the users file explorer, which will allow the user to find the JSON file in which their data is stored. Upon selecting and uploading to the webpage, all data will be readily available for use. This can be confirmed by checking the subjects tab and seeing it populated with data and a toast pop up that confirms successful upload.

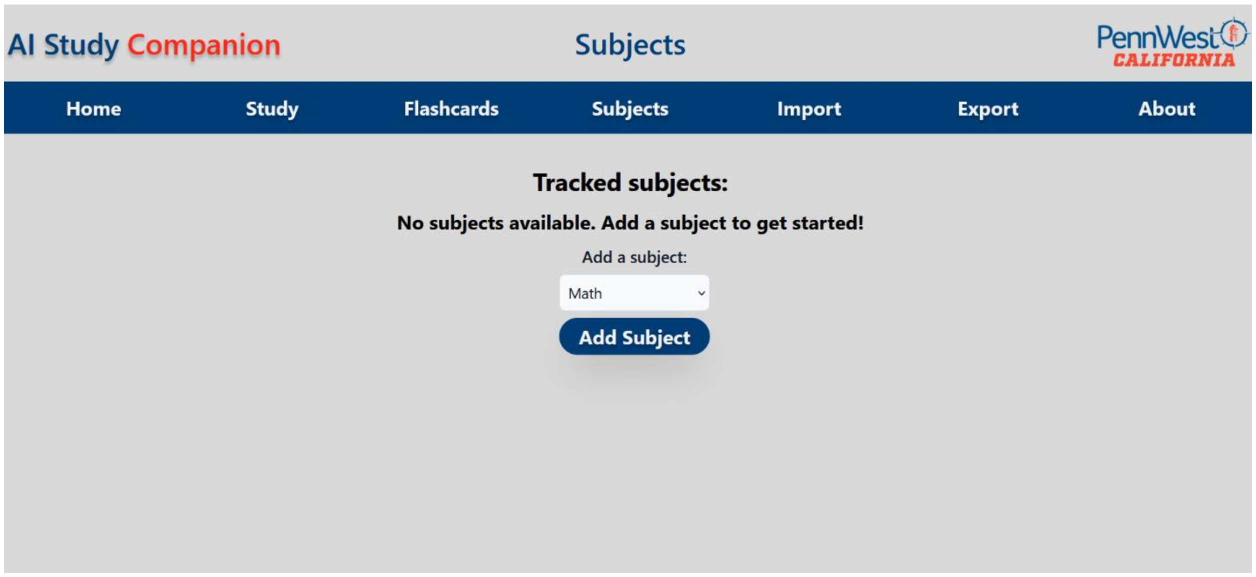


Name	Status	Date modified	Type
ascData.json	✓	4/16/2025 1:34 PM	JSON Source File

File name: ascData.json

*.json

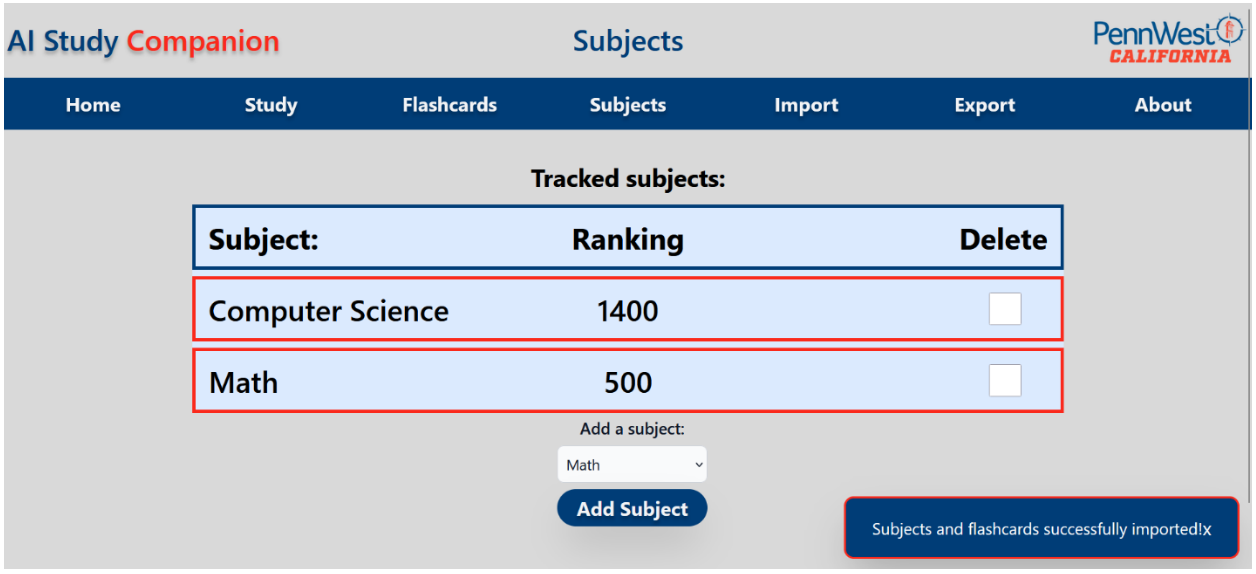
Fig. 2.2 File Upload



Tracked subjects:

No subjects available. Add a subject to get started!

Add a subject:



Tracked subjects:

Subject:	Ranking	Delete
Computer Science	1400	☐
Math	500	☐

Add a subject:

Subjects and flashcards successfully imported!x

Fig 2.3 Successful Import Before and After

Export works in a similar fashion. Upon clicking the Export tab on the home screen or header, all user data will be downloaded as a JSON file that can be used for the next session or to transfer between systems. If no user data is loaded, then a toast pop up will notify the user.

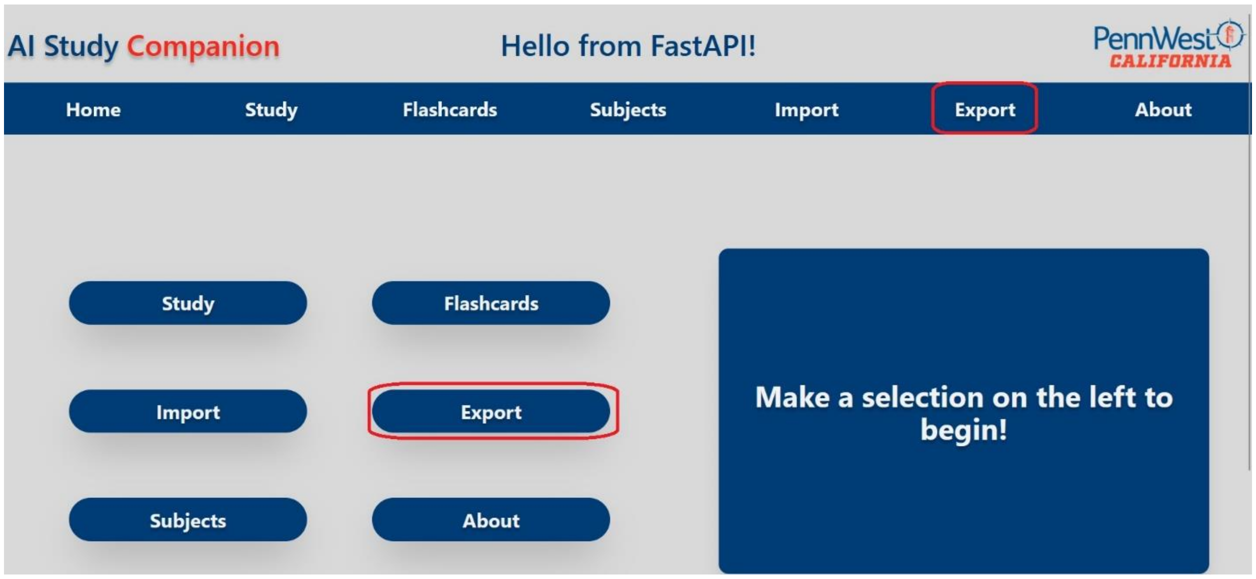


Fig #.4 Export Subject Selection

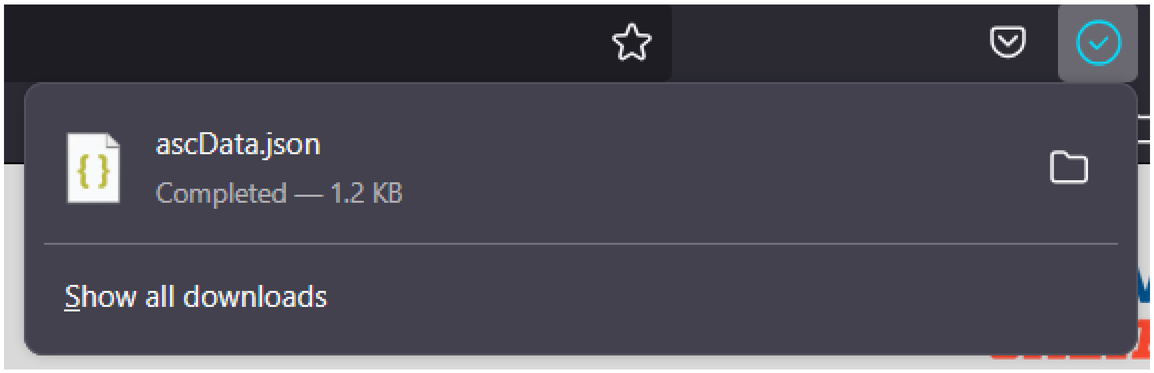
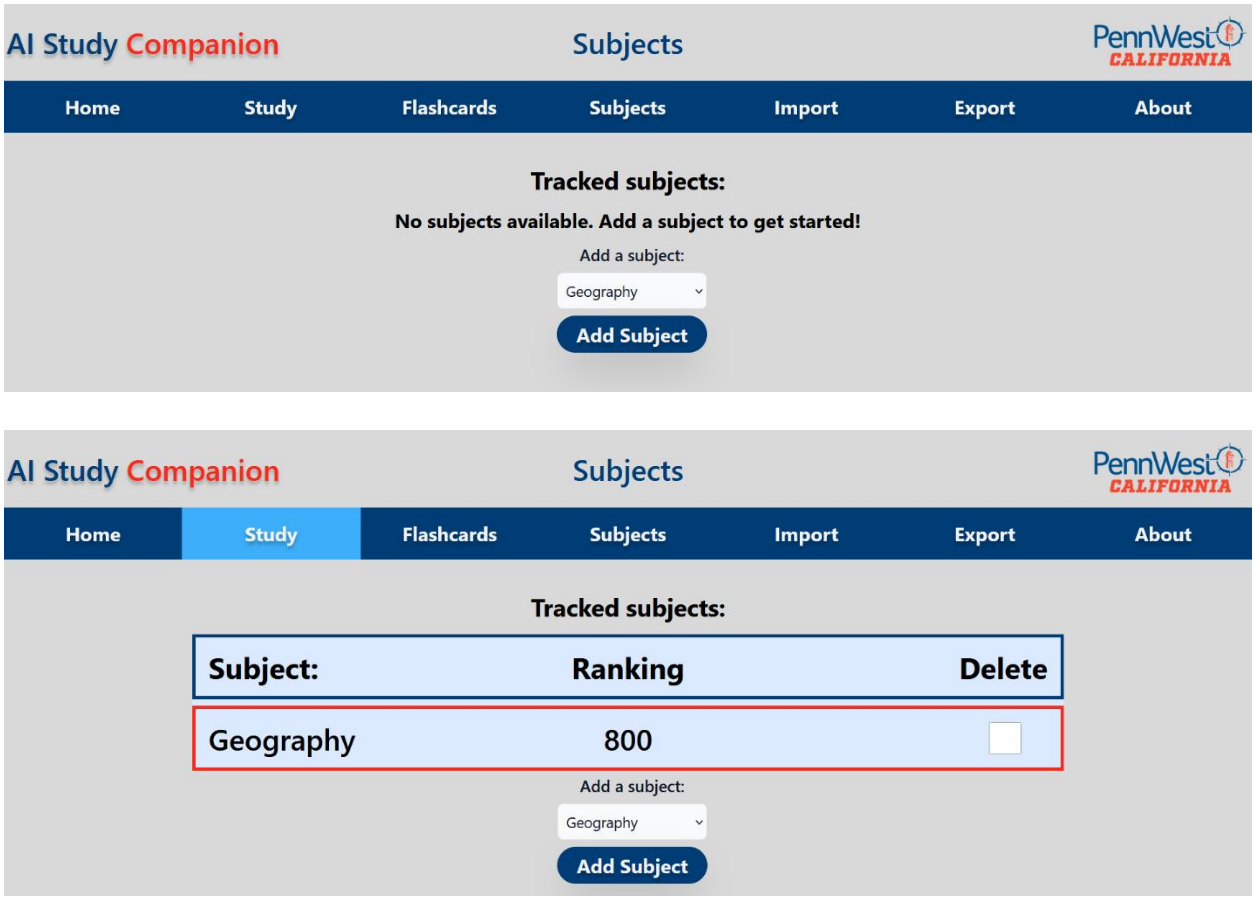


Fig 2.5 Successful Export of User Data

Adding / Removing Subjects

In the subject page, the user will be able to add subjects or delete them based on their current user data. A new user will have to add one of the five offered subject (Computer Science, English, Geography, History, and Math) before creating flashcards or study questions. This can be done by selecting the subject in the drop-down menu and then confirming by pressing the “Add Subject” button below. A newly created subject will be added to the user’s data with the default 800 ranking.



Tracked subjects:

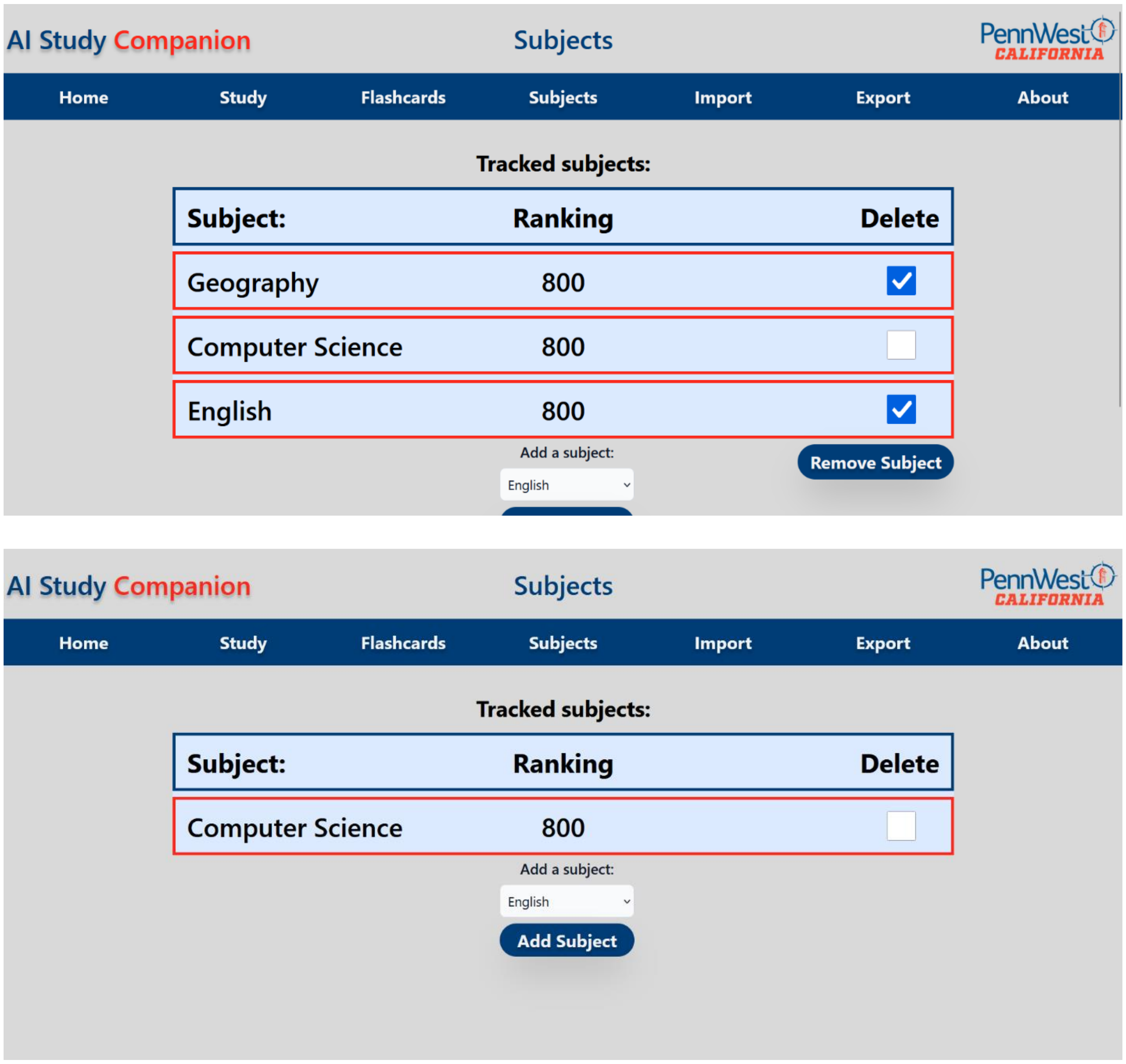
No subjects available. Add a subject to get started!

Tracked subjects:

Subject:	Ranking	Delete
Geography	800	☐

Fig 3.1 Successful Adding of Subject

In a similar vein, users can remove subjects they have had previously added by clicking the box of the subject the user wants to remove. Upon doing so, the “Remove Subject” button appears below the tracked subjects. Multiple subjects can be removed in one use of the remove subject button, depending on the subject's delete option that has been selected.



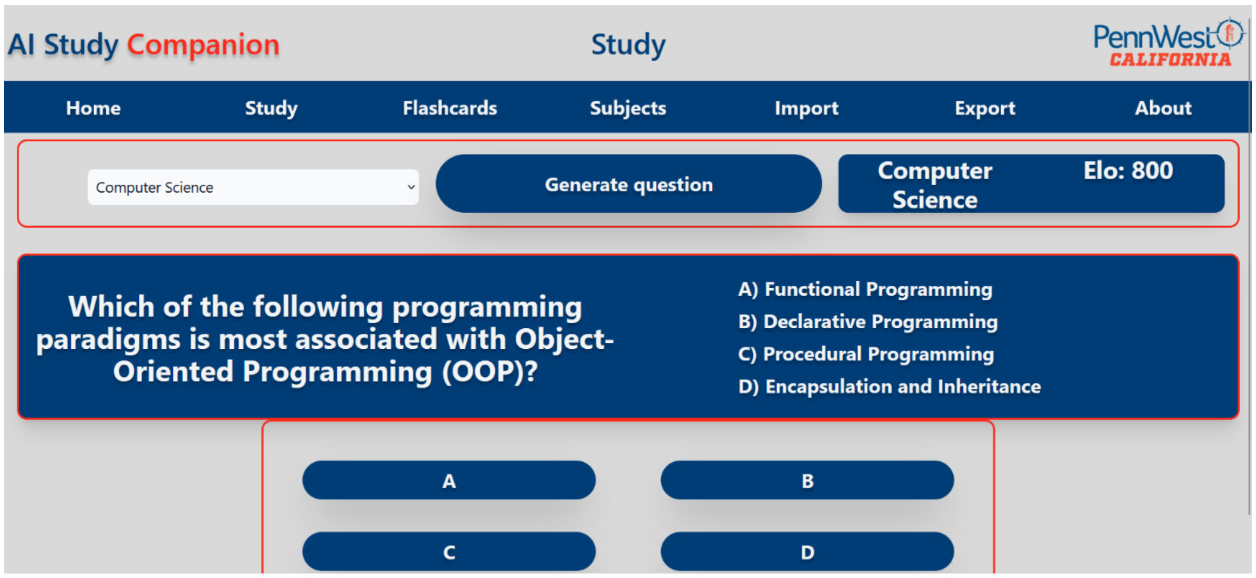
Subject:	Ranking	Delete
Geography	800	✓
Computer Science	800	☐
English	800	✓

Subject:	Ranking	Delete
Computer Science	800	☐

Fig 3.2 Successful Subject Removal

Study Section

Upon a successful import of user data or addition of subjects via the Subjects screen, users will navigate to the Study section. In this section, users will select one of their chosen subjects. That subject's skill level is then loaded to the screen and the user may select the generate question button. The user's skill level is then used to create a question for the selected subject using OpenAI. Users will have four options of answers to choose from with only one of them being the correct answer.

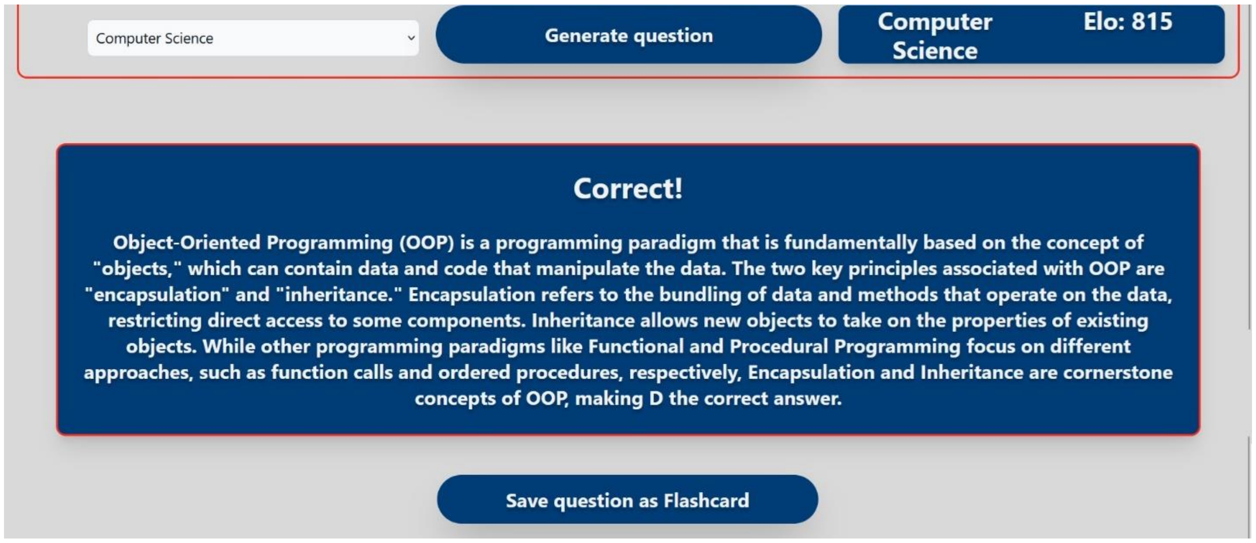


Which of the following programming paradigms is most associated with Object-Oriented Programming (OOP)?

A) Functional Programming
B) Declarative Programming
C) Procedural Programming
D) Encapsulation and Inheritance

Fig 4.1 Generated Question Example

After the user selects an answer, depending on if the correct answer is chosen or not, their subject Elo will change accordingly. For each correct answer the Elo will increase, and for each wrong answer the Elo will decrease. The Elo level will increase and decrease in an incrementing amount if the user has selected multiple correct answers or wrong answers in a row. Most subjects will give an explanation to why their answer corresponds to the option selected. Users also have the option to save the previous question as a flashcard. Users can then generate another question by doing the previous steps.



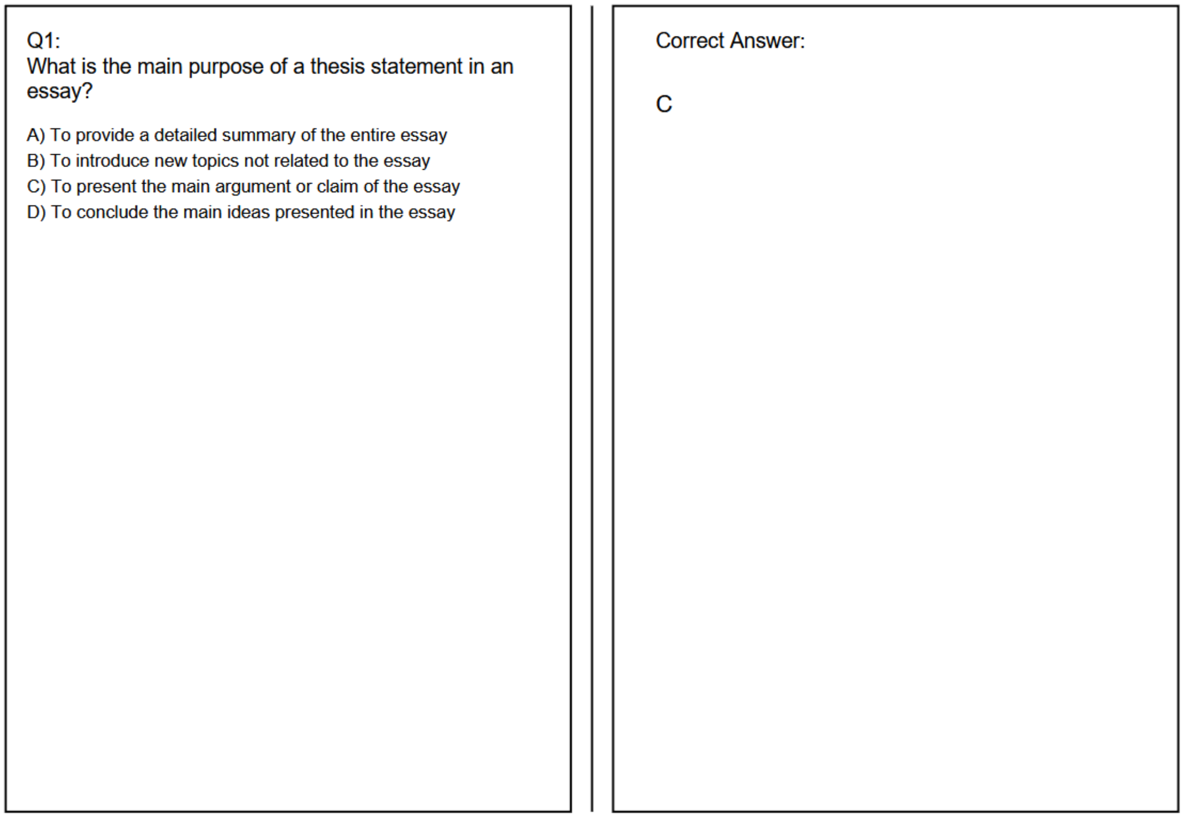
Correct!

Object-Oriented Programming (OOP) is a programming paradigm that is fundamentally based on the concept of "objects," which can contain data and code that manipulate the data. The two key principles associated with OOP are "encapsulation" and "inheritance." Encapsulation refers to the bundling of data and methods that operate on the data, restricting direct access to some components. Inheritance allows new objects to take on the properties of existing objects. While other programming paradigms like Functional and Procedural Programming focus on different approaches, such as function calls and ordered procedures, respectively, Encapsulation and Inheritance are cornerstone concepts of OOP, making D the correct answer.

Fig 4.2 Multiple Choice Question after input

Flashcard Section

The flashcard section allows the user to create flashcards based on their subjects and skill level in two ways. First, our "Legacy Flashcard" button can create 5 flashcards to be downloaded in PDF format. It works similarly to the study section as the user must first select their subject first, click generate, and then OpenAI will create five questions. The questions will be formatted with one half of the page pertaining to the questions and answer choices and the second half containing just the correct answer (See Fig 5.1).



Q1:
What is the main purpose of a thesis statement in an essay?

A) To provide a detailed summary of the entire essay
B) To introduce new topics not related to the essay
C) To present the main argument or claim of the essay
D) To conclude the main ideas presented in the essay

Correct Answer:

C

Fig 5.1 Flashcard PDF Format

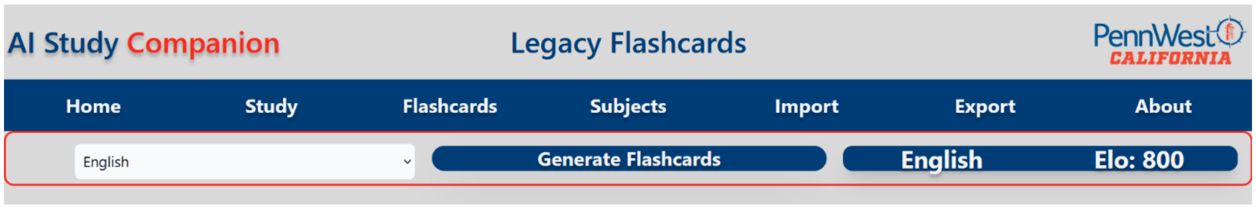
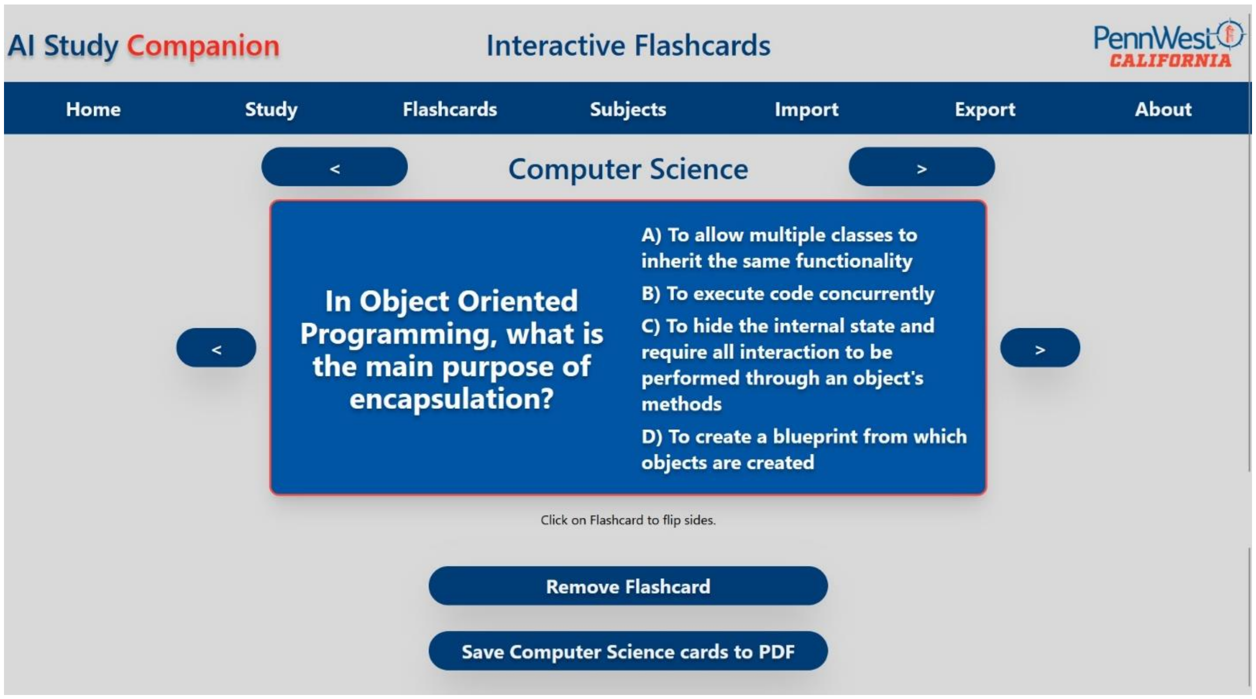


Fig 5.2 Legacy Flashcards

The second flashcard option is the "Interactive Flashcard" section. Questions saved from the Study section will appear here. The user can scroll between their subjects and saved flashcards using the buttons shown below (Fig 5.3). They will see the question and answers choices, clicking on the text box will then show the correct answer. These flashcards can then be deleted, downloaded in the format seen in Fig #.1 or left unchanged to go back to later.



In Object Oriented Programming, what is the main purpose of encapsulation?

A) To allow multiple classes to inherit the same functionality
B) To execute code concurrently
C) To hide the internal state and require all interaction to be performed through an object's methods
D) To create a blueprint from which objects are created

Fig 5.3 Interactive Flashcards

Appendix A: Team Details

The ASC Team shared responsibility for the work contained in this document. The creation, testing, and implementation of the ASC project was completed by

Jonathan Buckel, Cameron Calhoun, Gage Keslar and Seth Morgan. This document was created near the completion of said project. Notable contributions from each member includes:

Jonathan Buckel

- Math Class Handling
- Doxygen
- Class Demos / Gantt Chart

Cameron Calhoun

- Frontend
- Front/Backend Hosting & Communication
- Subject Classes
- ASC Progress Website
- Weekly Reports / Presentations / Gantt Chart

Gage Keslar

- Chatbot Class
- OpenAI API Reference and Handling
- Weekly Reports / Presentations / Gantt Chart / Class Demos

Seth Morgan

- Preliminary Math Class
- Math Class Handling

- User Manual

Appendix B: Writing Center Report

Dr. Chen,

At 9:30 this morning, I had an appointment via zoom with Gage Keslar to receive feedback for their user manual for their Senior Project. The student came in to receive feedback on their user manual for their Senior Project. Today we went over overall structure and ensuring the figures were formatted correctly. When going over the structure, we discussed the standard APA heading format. We also talked about the overall purpose and flow of the paper. Some of the later order concerns we discussed were making sure to include "a" before talking about PDF formats. We also discussed when it is appropriate to hyphenate words. Overall, the student made many of the corrections that we talked about in the fall semester.

If you have any other questions about our meeting, please do not hesitate to contact me at esp2609@pennwest.edu.

Sincerely,

Emily Esposito

Pennsylvania Western University California

Secondary Education Biology, *Junior*

Alpha Lambda Delta Honors Society, *Member*

Appendix C: Workflow Authentication

I, Jonathan Buckel, agree that I have performed all actions and contributed to the creation of the ASC Project as specified by the ASC User Manual.

Signature:__ Jonathan Buckel ____________Date: _4/20/2025

I, Cameron Calhoun, agree that I have performed all actions and contributed to the creation of the ASC Project as specified by the ASC User Manual.

Signature:___ Cameron Calhoun _______Date: _4/20/2025

I, Gage Keslar, agree that I have performed all actions and contributed to the creation to the ASC Project as specified by the ASC User Manual.

Signature:__ [illegible] ___________ _Date: 4/20/2025_

I, Seth Morgan, agree that I have performed all actions and contributed to the creation to the ASC Project as specified by the ASC User Manual.

Signature: Seth Morgan Date:

Appendix D: Code Listing

The following pages contain the source code for the services needed to make ASC Functional.

:AiStudyCompanion/ASC

Main.py

File: main.py

```
from fastapi import FastAPI, HTTPException, Request
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import JSONResponse

from SubjectClasses.english_subject import English
from SubjectClasses.geography_subject import Geography
from SubjectClasses.computerScience_subject import ComputerScience
from SubjectClasses.history_subject import History
from SubjectClasses.math_subject import Math

from user import User
from pydantic import BaseModel
from typing import List, Dict
from APITools import Chatbot
import copy

app = FastAPI()

user = User()

chatbot = Chatbot()
# user.addSubject(English())
# user.addSubject(Geography())
# user.addSubject(ComputerScience())
# user.addSubject(History())
#

origins = [
    "http://localhost:5173",
    'https://awc-staging.vercel.app',
    'https://ai-study-companion-pwc.vercel.app',
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

class AddSubjectPayload(BaseModel):
    subjects: List[Dict]
    newSubject: str

class QuestionPayload(BaseModel):
    subjects: List[Dict]
    curSubject: str
    newChat: bool
    delta: float
    streak: bool

class FlashcardsPayload(BaseModel):
    subjects: List[Dict]
    curSubject: str
    newChat: bool


@app.post("/api/add_subject")
async def add_subject(payload: AddSubjectPayload):
    subjects = payload.subjects
    newSubject = payload.newSubject

    user.importSubjects(subjects)
    match newSubject:
        case "English":
            user.addSubject(English())
        case "Geography":
            user.addSubject(Geography())
        case "Computer Science":
            user.addSubject(ComputerScience())
        case "History":
            user.addSubject(History())
        case "Math":
            user.addSubject(Math())
        case _:
            print("Subject not found")

    updated_subjects = user.exportSubjects()
    return {"subjects": updated_subjects}

@app.post("/api/generate_question")
async def generate_question(payload: QuestionPayload):
    # Assign values from payload
    global chatbot
    subjects = payload.subjects
    curSubject = payload.curSubject
    newChat = payload.newChat
    delta = payload.delta
    streak = payload.streak
    points: int

    basePoints = 15

    # if curSubject == math : set some bool to search for equations
    # Store user data in a User object
    user.importSubjects(subjects)
```

```

# Store current subject, elo, and set prompt
currentSubject = user.subjects[curSubject]
currentSubject.updatePrompt()

# if the user is streaking correct or incorrect answers, increase delta to a max of 2.0
# if they are not streaking, decrease by 10%, to a min of .6
# if no string was provided, that means:
#     question was unanswered,
#     or a newChat was started,
# so delta should remain unchanged
if streak == True:
    delta = min(2.0, delta + .1)
elif streak == False:
    delta = max(.6, delta * .9)


# If this is a new chat, create a new instance of chatbot
# removing context from old chat.
# Additionally, reset delta back to 1
if newChat:
    chatbot = Chatbot(curSubject)
    delta = 1

response = chatbot.generateQuestion(currentSubject.currentPrompt)

userIfWrong = copy.deepcopy(user)
ifWrongSubject = userIfWrong.subjects[curSubject]

points = int(round(basePoints * delta))
```

```

    ifWrongSubject.setSubjectElo(-points)
    currentSubject.setSubjectElo(points)

    updated_subjects_right = user.exportSubjects()
    updated_subjects_wrong = userIfWrong.exportSubjects()

    return {
        "ai_response": response,
        "subjects_right": updated_subjects_right,
        "subjects_wrong": updated_subjects_wrong,
        "delta": delta
    }

@app.post("/api/generate_flashcards")
async def generate_flashcards(payload: FlashcardsPayload):
    print("LOL")
    global chatbot
    subjects = payload.subjects
    curSubject = payload.curSubject
    newChat = payload.newChat
    flashcards = []

    user.importSubjects(subjects)
    currentSubject = user.subjects[curSubject]
    currentSubject.updatePrompt()

    if newChat:
        chatbot = Chatbot(curSubject)

    for _ in range(5):
        flashcards.append(chatbot.generateQuestion(currentSubject.currentPrompt))
    user.exportSubjects()

    return {
        "ai_response": flashcards
    }

@app.get("/api/hello")
async def hello():
    print("made it to api request")
    return {"message": "Hello from FastAPI!"}



```

requirements.txt

```
fastapi

uvicorn

pydantic

openai
```

```
python-dotenv

mangum

sympy
```

File: AiStudyCompanion\ASC\server\APITools.py

```

from openai import OpenAI
from dotenv import load_dotenv
import re

""" Design Doc

Class name: APITools

Class description: APITools is a utility class defined to house all utility functions for interacting with an LLM through various APIs.
	This will allow for clean reusable code, improve the readability of the program, and significantly help the writability of ASC.

Class data members: N/A

Class member functions: generateQuestion(), generateFlashcards(), solveMathEquation()
"""

load_dotenv()

class Chatbot:
    def __init__(self, subject=None):
        # Client between user and OpenAI
        self.client = OpenAI()
        # Messages to be used for creations
        self.context = [
            {"role": "system", "content": self.build_sys_prompt(subject)},
        ]
    def build_sys_prompt(self, subject=None):
        base_prompt = (
            "You are an educational assistant that provides multiple-choice questions "
            "to help users learn different subjects. Each question should have exactly four "
            "answer choices labeled A, B, C, and D. You may not provide any number of answer choices other than 4. Exactly one of these choices can be the correct answer. "
            "One choice MUST be the correct answer, and the other three MUST be incorrect."
            "NONE is not a valid answer. "
            "You must indicate the correct answer clearly "
            "and provide an explanation for why the answer is correct."
            "For this explanation, please surround the entire explanation in {}."
            "A format for your response will be described. Do not respond with anything other than the exact format."
            "The format of your response must be as follows:\n"
            "Question: <Insert question here>\n"
            "A) <Option A>\n"
            "B) <Option B>\n"
            "C) <Option C>\n"
            "D) <Option D>\n"
            "Answer: <Correct answer letter>\n"
            "Explanation: {<Detailed explanation of why the correct answer is correct and why the others are not>}"
        )
        if subject == "Math":
            return (
            f"{base_prompt}\n\n"
            "For Math questions, use the following additional rules:\n"
            "- Format every question as:\n"
            "  Question: Solve for $$<variable>$$\n"
            "  $$<expression>$$\n"
            "- Provide answer choices as single-line equations in the form:\n"
            "  A) $$<variable> = <value>$$\n"
            "  B) $$<variable> = <value>$$\n"
            "  C) $$<variable> = <value>$$\n"
            "  D) $$<variable> = <value>$$\n"
            "- Use only one of the following variables: a, b, c, x, y, z.\n"
            "- All expressions must be solvable with exactly one rational solution.\n"
            "- Do NOT use expressions with multiple valid solutions (e.g., sqrt(x^4)).\n"
            "- Do NOT use irrational or undefined solutions.\n"
            "- Ensure that the correct answer satisfies the given expression."        )
        else:
            return base_prompt

    # Send and receive messages to and from OpenAI
    def generateQuestion(self, message: str):
        self.context.append({"role": "user", "content": message})

        # Send the outgoing message, set model, store on OpenAI, and send messages in context
        response = self.client.chat.completions.create(
            model="gpt-4o",
            messages=self.context
        )
        # Store response
```

```
        response_content = response.choices[0].message.content

        self.context.append({"role": "assistant", "content": response_content})

        return response_content

    def get_user_answer(self, choices):
        valid_responses = {"A", "B", "C", "D"}
        while True:
            print("\nPlease select an answer:")
            user_answer = input("A/B/C/D").strip().upper()

            if user_answer in valid_responses:
                return user_answer
            else:
                print("Invalid choice. Please choose A, B, C, or D.")
```

File: AiStudyCompanion\ASC\server\user.py

```
import json
from typing import Dict
from pydantic import BaseModel
from subject import Subject
from SubjectClasses.english_subject import English
from SubjectClasses.geography_subject import Geography
from SubjectClasses.computerScience_subject import ComputerScience
from SubjectClasses.history_subject import History
from SubjectClasses.math_subject import Math
```

```

""" Design Doc

Class name: User

Class description: The User class will house a given users tracked subjects, as well as functionality pertaining to user information.

Class data members: trackedSubjects

Class member functions: addSubject(), removeSubject(), importData(), exportData()
"""

class User(BaseModel):
    # do we want to track userName?
    # we could use this in our front end to make it feel more interactive
    # "Hi userName, welcome back" etc
    # userName: str
    subjects: Dict[str, Subject] = {}

    def __init__(self, **kwargs):
        # Start with an empty subjects dictionary
        super().__init__(**kwargs)
        self.subjects = {}

    def addSubject(self, subject: Subject):
        subject_name = subject.subjectName
        if subject_name not in self.subjects:
            self.subjects[subject_name] = subject
        else:
            print(f"Subject '{subject.subjectName}' is already being tracked.")
```

```

def removeSubject(self, subjectName: str):
    if subjectName in self.subjects:
        del self.subjects[subjectName]
    else:
        print("Subject not found")

def exportSubjects(self):
    exportedData = [subject.model_dump() for subject in self.subjects.values()]
    self.subjects.clear()
    return exportedData

def importSubjects(self, data: list[dict]):
        for item in data:
            subject_name = item.get("subjectName")

            if subject_name == "English":
                subject = English(**item)
            elif subject_name == "Geography":
                subject = Geography(**item)
            elif subject_name == "Computer Science":
                subject = ComputerScience(**item)
            elif subject_name == "History":
                subject = History(**item)
            elif subject_name == "Math":
                subject = Math(**item)
            else:
                raise ValueError("Unknown Subject")

            self.addSubject(subject)
```

```

```

File: AiStudyCompanion\ASC\server\subject.py

```
from pydantic import BaseModel, Field
from typing import List, Dict

""" Design Doc
Class name: Subjects

Class description: The Subject class is a parent class that houses all generic methods and attributes applicable to all classes.
                This class will not be instantiated on its own but serves as a blueprint for every subject that inherits it.

Class data members: subjectName, subjectElo, subjectBreakpoints subjectPrompts

Class member functions: setSubjectName(), getSubjectName(), setSubjectElo(),
                        getSubjectElo(), setSubjectBreakpoints(), getSubjectBreakpoints(),
                        setSubjectPrompts(), getSubjectPrompts()
"""


class Subject(BaseModel):
    subjectName: str
    subjectElo: int = Field(800, ge=0, le=1600, description="ELO rating ranging from 0-1600")
    subjectBreakpoints: List[int] # Elo breakpoints
    subjectPrompts: Dict[int, str] # Prompts for each breakpoint
    currentPrompt: str = ""

    def setSubjectElo(self, amount: int):
        # Send a positive integer to increase, negative integer to decrease.
        self.subjectElo = max(0, min(self.subjectElo + amount, 1600))
        self.updatePrompt()

    def getSubjectElo(self):
        return self.subjectElo


    def setSubjectName(self, name: str):
        if not name:
            raise ValueError("Name cannot be empty")
        self.subjectName = name

    def getSubjectName(self):
        return self.subjectName
```

```


def setSubjectBreakpoints(self, breakpoints: List[int]):
    # Length of breakpoints must be > 0
    if not breakpoints:
        # All breakpoints must be in the bounds of 0 and 1600 inclusive
        for breakpoint in breakpoints:
            if not (0 <= breakpoint <= 1600):
                raise ValueError("Breakpoints must be between 0 and 1600 inclusive")
        self.subjectBreakpoints = breakpoints
    else:
        raise ValueError("Length of breakpoints cannot be zero")

def getSubjectBreakpoints(self):
    return self.subjectBreakpoints


def setSubjectPrompts(self, prompts: Dict[int, str]):
    for breakpoint, prompt in prompts.items():
        # All breakpoints must be in the bounds of 0 and 1600 inclusive
        if not (0 <= breakpoint <= 1600):
            raise ValueError("Breakpoints must be between 0 and 1600 inclusive")
        # All prompts must not be empty
        if not prompt:
            raise ValueError("Prompts cannot be empty")
    self.subjectPrompts = prompts

def getSubjectPrompts(self):
    return self.subjectPrompts


def updatePrompt(self):
    # Find highest available breakpoint, and set corresponding prompt
    validBreakpoints = [bp for bp in self.subjectBreakpoints if bp <= self.subjectElo]
    self.currentPrompt = self.subjectPrompts.get(max(validBreakpoints, default=0), "")



```

File: AiStudyCompanion\ASC\server\SubjectClasses\computerScience_subject.py

```
from subject import Subject
from typing import List, Dict

#@file computerScience_subject.py
#@brief This is the subject that manages everything for Computer Science.

class ComputerScience(Subject):
    #override subject variables
    subjectName : str = 'Computer Science'
    subjectElo : int = 800
    subjectBreakpoints: List[int] = [400, 600, 800, 1000, 1200]
    subjectPrompts: Dict[int, str] = {
        400: 'Basic computer literacy',
        600: 'High school level Computer Science and problem solving',
        800: 'Computer Science subjects. Namely: Intro Programming, Object Oriented Programming, System Design.',
        1000: 'Advanced Computer Science Subjects. Namely: Data Structures, Analysis of Algorithms, Computer Networking, programming paradigms, programming methodologies.',
        1200: 'Advanced Computer Science Subjects. Namely: Low level concepts, computer architecture, language theory, dynamic programming, distributed systems, artificial intelligence, scalable systems, language translation.',
    }
    currentPrompt : str = subjectPrompts[800]


```

File: AiStudyCompanion\ASC\server\SubjectClasses\english_subject.py

```
from subject import Subject
from typing import List, Dict


class English(Subject):
    #override subject variables
    subjectName : str = 'English'
    subjectElo : int = 800
    subjectBreakpoints: List[int] = [400, 600, 800, 1000]
    subjectPrompts: Dict[int, str] = {
        400: 'Elementary level English',
        600: 'Middle School level English',
        800: 'High school level English',
        1200: 'College level English'
    }
    currentPrompt : str = subjectPrompts[800]
```

```

```

File: AiStudyCompanion\ASC\server\SubjectClasses\geography_subject.py

```
from subject import Subject
from typing import List, Dict


class Geography(Subject):
    #override subject variables
    subjectName : str = 'Geography'
    subjectElo : int = 800
    subjectBreakpoints: List[int] = [400, 600, 800, 1200]
    subjectPrompts: Dict[int, str] = {
        400: 'Elementary level Geography',
        600: 'Middle school level Geography',
        800: 'High school level Geography',
        1200: 'College level Geography'
    }
    currentPrompt : str = subjectPrompts[800]
```

File: AiStudyCompanion\ASC\server\SubjectClasses\history_subject.py

```
from subject import Subject
from typing import List, Dict


class History(Subject):
    #override subject variables
    subjectName : str = 'History'
    subjectElo : int = 800
    subjectBreakpoints: List[int] = [400, 600, 800, 1200]
    subjectPrompts: Dict[int, str] = {
        400: 'Elementary level History',
        600: 'Middle school level History',
        800: 'High school level History',
        1200: 'College level History'
    }
    currentPrompt : str = subjectPrompts[800]
```

File: AiStudyCompanion\ASC\server\SubjectClasses\math_subject.py

```
from typing import List, Dict, ClassVar
from subject import Subject


class Math(Subject):
    #override subject variables
    subjectName : str = 'Math'
    subjectElo : int = 800
    subjectBreakpoints : List[int] = [400, 800, 1000]
    subjectPrompts : Dict[int, str] = {
        400: 'Easy algebra questions',
        800: 'Harder Algebra questions',
        1000: 'Advanced Algebra questions'
    }
    currentPrompt : str = subjectPrompts[800]

```

File: AiStudyCompanion\ASC\src\App.jsx

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import Subjects from "./pages/Subjects";
import Progress from "./pages/Progress"
import Study from "./pages/Study";
import Flashcards from "./pages/Flashcards"
import NewFlashcards from "./pages/NewFlashcards";
import LegacyFlashcards from "./pages/LegacyFlashcards";
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/subjects" element={<Subjects />} />
```

```
                <Route path="/progress" element={<Progress />} />
                <Route path="/study" element={<Study />} />
                <Route path="/flashcards" element={<Flashcards />} />
                <Route path="/interactive_flashcards" element={<NewFlashcards />} />
                <Route path="/legacy_flashcards" element={<LegacyFlashcards />} />
            </Routes>
        </Router>
    )
}

export default App;

```

File: AiStudyCompanion\ASC\src\main.jsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './styles.css'
import App from './App.jsx'
import { ToastProvider } from './contexts/ToastContext'

createRoot(document.getElementById('root')).render(
    <StrictMode>
        <ToastProvider>
            <App />
        </ToastProvider>
    </StrictMode>,
)
```

File: AiStudyCompanion\ASC\src\styles.css

```
@import "tailwindcss";

@theme{
	--color-pwblue: #003d77;
	--color-pwred: #FA291C;
	--color-bgwhite: #D9D9D9;
}

.katex-display {
  @apply inline-block mx-auto ;
}

.text-shadow {
	text-shadow: 0px 4px 4px rgba(0,0,0,.25);
}

.article-typography {
  @apply max-w-screen-lg mx-auto text-gray-900;
```

```
}

.article-typography h1 {
  @apply text-4xl font-extrabold mb-4;
}

.article-typography h2 {
  @apply text-2xl font-semibold mt-6 mb-3;
}

.article-typography h3 {
  @apply text-xl font-medium mt-4 mb-3;
}

.article-typography p {
  @apply text-lg leading-relaxed mb-4;
}

.article-typography ul {
  @apply list-disc pl-6 mb-4;
}

.article-typography iframe {
  @apply w-full h-80 max-w-screen-lg mx-auto rounded-lg shadow-lg;
}

.dot-flashing {
  position: relative;
  width: 20px;
  height: 20px;
  border-radius: 10px;
  background-color: #003d77;
  color: #003d77;
  animation: dot-flashing 1s infinite linear alternate;
  animation-delay: 0.5s;
}
.dot-flashing::before, .dot-flashing::after {
  content: "";
  display: inline-block;
  position: absolute;
  top: 0;
}
.dot-flashing::before {
  left: -30px;
  width: 20px;
  height: 20px;
  border-radius: 10px;
  background-color: #003d77;
  color: #003d77;
  animation: dot-flashing 1s infinite alternate;
  animation-delay: 0s;
}
.dot-flashing::after {
  left: 30px;
  width: 20px;
  height: 20px;
  border-radius: 10px;
  background-color: #003d77;
  color: #003d77;
  animation: dot-flashing 1s infinite alternate;
  animation-delay: 1s;
}

@keyframes dot-flashing {
  0% {
    background-color: #003d77;
  }
  50%, 100% {
    background-color: rgba(152, 128, 255, 0.2);
  }
}

```

File: AiStudyCompanion\ASC\src\components\Button.jsx

```
import '../styles.css'

function Button({ text, onClick, onMouseEnter, onMouseLeave, className }) {

    return (
        <div
            className={`w-full h-full ${className} py-2 bg-pwblue rounded-full shadow-2xl flex items-center whitespace-nowrap justify-center
                            hover:bg-[#3DAEF9]
                            active:scale-95 active:shadow-md transition-all duration-300`}
            onClick={onClick}
            onMouseEnter={onMouseEnter}
            onMouseLeave={onMouseLeave}
        >
            <h1 className="text-shadow text-2xl font-bold text-[#F3F4F6]">{text}</h1>
        </div>
    )
}

export default Button

```

File: AiStudyCompanion\ASC\src\components\Card.jsx

```
import '../styles.css'

function Card({ text, className }) {

    return (
        <div className="w-[85%] h-[71.77%] px-8 py-50 bg-pwblue rounded-xl shadow-xl flex text-center items-center justify-center ">
            <h1 className="text-shadow text-4xl font-bold text-[#F3F4F6]">{text}</h1>
        </div>
    )
}

export default Card

```

File: AiStudyCompanion\ASC\src\components\NavBar.jsx

```
import NavButton from "./NavButton"
import { useNavigate } from "react-router-dom"
import { handleExport, handleImport } from "../utils/fileHandlers"
import { useToast } from "../contexts/ToastContext"
function NavBar() {
    const navigate = useNavigate()
    const { addToast } = useToast()
    return (
        <div className="bg-pwblue h-16 flex justify-center">
            <div className="flex justify-center gap-1/2 w-full">
                <NavButton text="Home" onClick={() => navigate("/")} />
                <NavButton text="Study" onClick={() => navigate("/study")} />
                <NavButton text="Flashcards" onClick={() => navigate("/flashcards")} />
                <NavButton text="Subjects" onClick={() => navigate("/subjects")} />
                <NavButton text="Import" onClick={() => handleImport(addToast)} />
                <NavButton text="Export" onClick={() => handleExport(addToast)} />
                <NavButton text="About" onClick={() => navigate("/about")} />
            </div>
        </div>
    )
}
export default NavBar
```

File: AiStudyCompanion\ASC\src\components\NavButton.jsx

```
import '../styles.css'

function NavButton({ text, onClick }) {

    return (

        <div
            className="w-1/4 h-full bg-pwblue flex items-center whitespace-nowrap justify-center py-3 px-10
                        hover:bg-[#3DAEF9]
                        active:scale-x-95 active:shadow-md transition-all duration-300"
            onClick={onClick}
        >
            <h1 className="text-shadow text-2xl font-bold text-[#F3F4F6]">{text}</h1>

        </div>

    )
}

export default NavButton
```

File: AiStudyCompanion\ASC\src\components\Subject.jsx

```
import '../styles.css'

function Subject({ name, rank, onCheckboxChange }) {
    return (
        <div>
            <div className='border-4 border-pwred mb-2 font-semibold text-4xl flex justify-between
                            p-4'>
                <span className='w-1/3'>{name}</span>
                <span className='w-1/3 text-center'>{rank}</span>
                <input
                    type="checkbox"
                    className='my-auto ml-auto mr-8 w-10 h-10'
                    onChange={(e) => onCheckboxChange(name, e.target.checked)} />
            </div>

        </div>
    )
}

export default Subject

```

File: c:\Users\gagem\AiStudyCompanion\ASC\src\components\SubjectList.jsx

```
import React, { useEffect, useState } from 'react'
import Subject from './Subject'

function SubjectList({ onSubjectSelect }) {
    const [subjects, setSubjects] = useState([])



    useEffect(() => {
        const loadSubjects = () => {
            const storedData = sessionStorage.getItem("importedSubjects")
            setSubjects(storedData ? JSON.parse(storedData) : [])
        }

        loadSubjects()

        const handleUpdate = () => loadSubjects()

        window.addEventListener("subjectsUpdated", handleUpdate)
        return () => window.removeEventListener("subjectsUpdated", handleUpdate)
    }, [])

    return (
        <div>
            <h2 className="text-3xl text-center font-bold mb-4">Tracked subjects:</h2>
            <div className={`${subjects.length > 0 && `bg-blue-100`} w-full h-full`}>
                {subjects.length > 0 && (
                    <div className="border-4 border-pwblue font-bold text-4xl mb-2 flex justify-between p-4">
                        <span className="w-1/3">Subject:</span>
                        <span className="w-1/3 text-center">Ranking</span>
                        <span className="w-1/3 text-right">Delete</span>
                    </div>
                )}
                {subjects.length === 0 ? (
                    <p className="text-2xl text-center font-bold mb-4">No subjects available. Add a subject to get started!</p>
                ) : (
                    subjects.map((subject, index) => (
                        <Subject
                            key={index}
                            name={subject.subjectName}
                            rank={subject.subjectElo}
                            onCheckboxChange={onSubjectSelect}
                        />
                    ))
                )}
            </div>
        </div>
    )
}

export default SubjectList
```

File: AiStudyCompanion\ASC\src\components\TopBar.jsx

```
import NavBar from "./NavBar"
function TopBar({ title }) {
    return (
        <div className="sticky top-0 z-10">
            {/* White bar */}
            <div className="bg-bgwhite p-1 flex h-24 items-center justify-between text-4xl font-semibold">
                <h1 className="w-1/3">
                    <span className="text-shadow text-pwblue">AI Study</span>
                    <span className="text-shadow text-pwred"> Companion</span>
                </h1>
                <h1 className="text-pwblue w-1/3 text-center">{title}</h1>
                <div className="w-1/3">
                    <img src="pennwest-california.png" className="h-30 ml-auto" />
                </div>
            </div>

            {/* Blue bar */}
            <NavBar />
        </div>
    )
}

export default TopBar
```

File: AiStudyCompanion\ASC\src\config\api.js

```
const BASE_URL = import.meta.env.VITE_API_URL || "http://127.0.0.1:8000"

export const API_HELLO = `${BASE_URL}/api/hello`
export const API_ADD_SUBJECT = `${BASE_URL}/api/add_subject`
export const API_GENERATE_QUESTION = `${BASE_URL}/api/generate_question`
export const API_GENERATE_FLASHCARDS = `${BASE_URL}/api/generate_flashcards`
export const API_EXPORT = `${BASE_URL}/api/export`

```

File: AiStudyCompanion\ASC\src\contexts\ToastContext.jsx

```
import React, { createContext, useContext, useState, useCallback } from 'react'

const ToastContext = createContext()

export const ToastProvider = ({ children }) => {
    const [toasts, setToasts] = useState([])

    const addToast = useCallback((message, duration = 5000) => {
        const id = Date.now()
        setToasts((prev) => [...prev, { id, message }])
        setTimeout(() => removeToast(id), duration)
    }, [])

    const removeToast = useCallback((id) => {
        setToasts((prev) => prev.filter((t) => t.id !== id))
    }, [])

    return (
        <ToastContext.Provider value={{ addToast }}>
            {children}
            <div className='fixed bottom-4 right-4 flex flex-col-reverse gap-2 z-50'>
                {toasts.map(({ id, message }) => (
                    <div key={id} className="bg-pwblue border-3 border-pwred rounded-xl text-white px-8 py-5 rounded-xl shadow-2xl flex items-center justify-between min-w-[300px] animate-slideIn">
                <span className='text-xl'>{message}</span>
                <button className='text-2xl' onClick={() => removeToast(id)}>x</button>
              </div>
            ))}
          </div>
        </ToastContext.Provider>
    )

}

export const useToast = () => useContext(ToastContext)

```

File: AiStudyCompanion\ASC\src\pages\progressFiles\articles.js

```
const articles = {
  about: `
  <div class="article-typography">
  <h1>About ASC</h1>
  <h2>What is ASC?</h2>
    <p>AI Study Companion (ASC) is an in progress, AI powered study companion hosted on the web. ASC can track the users progress in selected subjects, and pushes them in the correct direction to facilitate learning.</p>
  <h2>Why choose ASC?</h2>
    <p>ASC differs from other study companions with its built in AI capabilities. ASC will not use AI to verify a students work, but to procedurally generate questions and study materials tailored to each individaul user. The more you interact with ASC, the better ASC will grow to understand your strengths and weaknesses regarding various subjects.</p>
  <h2>What technologies does ASC utilize?</h2>
    <p>ASC will be a full-stack web application with a React front-end and a Python back-end utilizing the FastAPI library.</p>
  <h2>Who created ASC?</h2>
    <p>ASC is a student created senior project for Pennsylvania Western University driven by the following members:</p>
    <ul>
      <li>Cameron Calhoun</li>
      <li>Gage Keslar</li>
      <li>Jonathan Buckel</li>
      <li>Seth Morgan</li>
    </ul>
    <p>All members are students of Pennsylvania Western University studying Computer Science.</p>
  <h2>Gantt Chart</h2>
```

```
   <h3>Updated 4/16/25</h3>

<iframe
  class="w-full h-[1000px] max-w-screen-lg mx-auto rounded-lg shadow-lg"
  src="Gantt Chart - Senior Project.pdf#toolbar=1&zoom=page-fit"
  title="Gantt Chart - Senior Project"
></iframe></div>
   `,
  week1: `
  <div class="article-typography">
  <h1>Week 1</h1>
  <h2>Summary:</h2>
   <p>Week 1 was entirely lectures, and thus no report was completed. Reports begin in week 2.</p>
</div>
  `,
  week2: `
  <div class="article-typography">
  <h1>Week 2</h1>
  <h2>Summary:</h2>
   <p>This first week was spent setting up our development environment, developing the project website, and making the Gantt chart. Overall, everything went according to plan. The only change made is that we will no longer be using Docker. Due to compatibility challenges we reevaluated our need for Docker and realized that we would be able to work fine without it.</p>
  <h2>What was Accomplished:</h2>
   <ul>
     <li>Gantt Chart</li>
     <li>Git Repository</li>
     <li>FastAPI and React communcation</li>
     <li>Website for weekly reports</li>
   </ul>
  <h2>Problems Encountered:</h2>
   <ul>
     <li>Docker compatability issues</li>
   </ul>
  <h2>Plans to Overcome Problems:</h2>
   <ul>
     <li>No longer using Docker. We determined that we wouldn't be losing annything by not using it.</li>
   </ul>
  <h2>Plans for next week:</h2>
   <ul>
     <li>Get API keys</li>
     <li>Begin backend development</li>
   </ul>
  <h2>Member Contributions:</h2>
   <ul>
     <li>Cameron Calhoun - Website</li>
     <li>Jonathan Buckel - Weekly Report</li>
     <li>Gage Keslar - Gantt Chart</li>
     <li>Seth Morgan - Presentation</li>
   </ul>
  </div>
  `,
  week3: `
  <div class="article-typography">
  <h1>Week 3</h1>
  <h2>Summary:</h2>
   <p>This second week of the project was focused primarily on OpenAI API Integration, and developing a working demo for this integration. We first obtained tokens for communicating with the AI model. These are spent with each message sent and received. Limits were set on the amount of there tokens used, as well as monetary monthly limits on OpenAI. To communicate with the AI model, we must make requests through an API. This meant we had to develop a way to store our API keys. This was done through the python-dotenv library, and making use of .env files. This hides the API key from the project itself, and makes the key secure. If we were to
```

```
expose our API key to the web, our tokens could be used by malicious, outside users, and our wallets drained. The rest of the time this week was spent developing our communication with the AI model, and developing a simple chat bot we can communicate with. This will allow easy integration of features in the program that require communicating with the AI model.</p>
<h2>What was Accomplished:</h2>
  <ul>
    <li>AI tokens acquired</li>
    <li>OpenAI API communication established</li>
    <li>Proper storage and hiding of API keys</li>
    <li>Simple chat bot created</li>
  </ul>
<h2>Problems Encountered:</h2>
  <ul>
    <li>Potential abuse of tokens</li>
    <li>Setting up API keys across all developers environments</li>
  </ul>
<h2>Plans to Overcome Problems:</h2>
  <ul>
    <li>No longer using Docker. We determined that we wouldn't be losing annything by not using it.</li>
  </ul>
<h2>Plans for next week:</h2>
  <ul>
    <li>Establish User class</li>
    <li>Establish Subject Classes</li>
  </ul>
<h2>Member Contributions:</h2>
  <ul>
    <li>Cameron Calhoun - Weekly Report</li>
    <li>Jonathan Buckel - AI communication, python-dotenv implementation</li>
    <li>Gage Keslar - AI communication, presentation, chat bot</li>
    <li>Seth Morgan - AI communication</li>
  </ul>
</div>
`,
week4: `
<div class="article-typography">

<h1>Week 4</h1>
<h2>Summary:</h2>
  <p>This week our focus was directed towards developing the User and Subject classes, and using them to interact with out chat bot. If you recall from our report last week, we were able to successfully establish communication with OpenAI’s GPT-4o model, and we could send and receive messages through our terminal. This week we wanted to focus these messages on the actual application of our program, and create some of the classes elaborated on in our specification document to facilitate this process. A Subject class was created which will be the parent class of all specific subjects, and contain all methods of said derived classes. This will allow for a very modular program, in which additional subjects can be easily added by deriving a new class. We created an English class to showcase the bots new functionality. In addition, we created a User class which stores a users data on which subjects they are tracking. In addition, this class contains methods to add and remove classes, as well as import and export user data using JSON.</p>
<h2>What was Accomplished:</h2>
  <ul>
    <li>Subject class created</li>
    <li>English class created</li>
    <li>User class created</li>
    <li>Chat bot subject interaction completed</li>
  </ul>
<h2>Problems Encountered:</h2>
  <ul>
    <li>Structure of subject class</li>
    <li>Other coursework</li>
  </ul>
<h2>Plans to Overcome Problems:</h2>
  <p>The only real issue found in this weeks work was discussing the data types for the subject class. At one point, we had debated using a list for the subjects prompts, but elected for a
```

```
dictionary in order to increase readability within the program. Other coursework kept us
occupied as well, and was something to juggle on top of this weeks work.</p>
<h2>Plans for this Week:</h2>
  <ul>
    <li>Create remaining subject classes</li>
    <li>Further work on back end</li>
  </ul>
<h2>Member Contributions:</h2>
  <ul>
    <li>Cameron Calhoun - Weekly Report, Subject class, User class</li>
    <li>Gage Keslar - Powerpoint, Chat bot updates, English class</li>
    <li>Seth Morgan - Communicated some ideas, began prototype Math class</li>
    <li>Jonathan Buckel - Communicated some ideas</li>
  </ul>
</div>
`,
week5: `
<div class="article-typography">
<h1>Week 5</h1>
<h2>Summary:</h2>
  <p>This week our focus was directed towards developing the remaining subject classes
and further developing the back end. Geography, History, and ComputerScience classes were
created and fleshed out to the same degree as English. Additionally, Math was completed. They
currently all only have 3 prompts and breakpoints, and are worded similarly to English. Over the
next week or two, we will be honing in on these prompts and breakpoints to ensure that a smooth
generation is created. We additionally spent time locking down the chatbot, such that users
cannot send nonsense to it. It now only accepts selections for questions. In incorporating multiple
subjects into the chatbot, some changes needed to be made to User and Subject. Namely,
changing the storage of classes in User from a list to a dictionary. This allows us to reference a
class from user using a key value pair, where the key is the name of the subject, and the value is
the class itself.</p>
<h2>What was Accomplished:</h2>
  <ul>
    <li>Geography class created</li>
```

```
      <li>History class created</li>
      <li>Computer Science class created</li>
      <li>Math class completed</li>
      <li>User class edited</li>
      <li>Chatbot locked down </li>
    </ul>
  <h2>Problems Encountered:</h2>
    <ul>
      <li>Poor indexing for User.subjects</li>
      <li>Prompt specific issues</li>
    </ul>
  <h2>Plans to Overcome Problems:</h2>
    <p>We redefined User.subjects into a dictionary, such that we could use the names of the subjects as keys in key value pairs. This way, we can directly address a subject instead of looping through User.subjects entirely to find a reference each time. Additionally, the prompts can sometimes generate vague answers, and create confusion. We will solve this in the coming weeks by engineering the prompts carefully.</p>
  <h2>Plans for this Week:</h2>
    <ul>
      <li>Complete APITools logic</li>
      <li>Engineer prompts that do not confuse the bot for each class</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
      <li>Cameron Calhoun - Weekly Report, Geography class, History class, ComputerScience Class, completed Math class, restructured User class, Powerpoint</li>
      <li>Gage Keslar - Communicated some ideas, helped reformat classes</li>
      <li>Seth Morgan - Reformatted to fit design specifications, restricted chatbot</li>
      <li>Jonathan Buckel - Communicated some ideas, helped reformat classes</li>
    </ul>
  </div>
  `,
  week6: `
  <div class="article-typography">
  <h1>Week 6</h1>
  <h2>Summary:</h2>
    <p>This week had focus on further backend improvements including, separating user input and AI, fixing our prompts to limit token usage / more concise questions and answer options, improving our math subject class, and beginning our front end design. The user will now no longer be able to put answers no longer pertaining to the question. Instead they are limited to the labeled options. The subject class no longer sends a prompt with formatting instructions, subject, and skill level. All the subject needs to do is send the subject and skill level and OpenAI will generate the question. Our math class only generates the questions and tools implemented will ensure a correct answer. Finally, the construction of our front end has started. The main menu and subject menu has been designed but lack complete functionality as of right now.</p>
  <h2>What was Accomplished:</h2>
    <ul>
      <li>Fixed amount of tokens being set to OpenAI</li>
      <li>Removing AI and user interaction</li>
      <li>Removing redundant code</li>
    </ul>
  <h2>Problems Encountered:</h2>
    <ul>
      <li>Math Subject Class</li>
    </ul>
  <h2>Plans to Overcome Problems:</h2>
    <p>We always planned to keep OpenAI from answering math questions. This was due to the fact that ChatGPT has always had unreliable answers to math questions. Using sympy, we can have ChatGPT make the questions and then create the answers using that tool.</p>
  <h2>Plans for this Week:</h2>
    <ul>
      <li>Continue work on Frontend</li>
      <li>Put Frontend online</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
```

```
      <li>Cameron Calhoun - Separated the logic and AI, Created frontend design, reorganized prompts, contributed to math subject class</li>
      <li>Gage Keslar - Weekly Report, PowerPoint presentation</li>
      <li>Seth Morgan - Contributed to math subject class</li>
      <li>Jonathan Buckel - Communicated some ideas, Sympy research</li>
    </ul>

</div>
`,
week7: `
<div class="article-typography">
<h1>Week 7</h1>
<h2>Summary:</h2>
    <p>This week had focus on further backend improvements including, separating user input and AI, fixing our prompts to limit token usage / more concise questions and answer options, improving our math subject class, and beginning our front end design. The user will now no longer be able to put answers no longer pertaining to the question. Instead they are limited to the labeled options. The subject class no longer sends a prompt with formatting instructions, subject, and skill level. All the subject needs to do is send the subject and skill level and OpenAI will generate the question. Our math class only generates the questions and tools implemented will ensure a correct answer. Finally, the construction of our front end has started. The main menu and subject menu has been designed but lack complete functionality as of right now.</p>
<h2>What was Accomplished:</h2>
    <ul>
      <li>Frontend Development</li>
      <li>Hosting Frontend and Backend</li>
      <li>Created Frontend / Backend interactivity</li>
      <li>Changes to Math Subject class</li>
      <li>Sympy</li>
    </ul>
<h2>Problems Encountered:</h2>
    <ul>
      <li>Hosting Frontend / Backend</li>
    </ul>
```

```
<h2>Plans to Overcome Problems:</h2>
  <p>We always planned to keep OpenAI from answering math questions. This was due to the fact that ChatGPT has always had unreliable answers to math questions. Using sympy, we can have ChatGPT make the questions and then create the answers using that tool.</p>
<h2>Plans for this Week:</h2>
  <ul>
    <li>Continued work on Frontend</li>
    <li>Creating APIs between front and backend</li>
  </ul>
<h2>Member Contributions:</h2>
  <ul>
    <li>Cameron Calhoun - Creating frontend, Backend/Frontend Interactivity, Hosting platforms, PowerPoint presentation.</li>
    <li>Gage Keslar - Communicated some ideas, weekly report</li>
    <li>Seth Morgan - Further backend development</li>
    <li>Jonathan Buckel - Further backend development, Sympy</li>
  </ul>

</div>
`,
week8: `
<div class="article-typography">
<h1>Week 8</h1>
<h2>Summary:</h2>
  <p>Week 8 was spring break, and thus there was no report. The work for week 8 will be included in week 9's report.</p>
</div>
`,
week9: `
<div class="article-typography">
<h1>Week 9</h1>
<h2>Summary:</h2>
```

```
    <p>The past two weeks were focused on fleshing out the front ends functionality. We began with redesigning the front end from our prototype, to a more finalized design that fits more modern user interface conventions, such as a smaller top bar, and a navigation bar for one click navigation to any page on the website. This design is substantially more user friendly, and easier to navigate when new to the website. A NavBar component was created and used in the program. Additionally, the Import and Export features were fully fleshed out in the program. This data is stored in a session, and is created in the back end using the User class defined earlier in the project. This will allow users to save their data for future uses of the program. Additionally, the Subjects page, Add Subject, and Remove Subject features have all been added and completed. This allows the users to visually track the progress of their subjects, add new subjects, and remove subjects from their tracking. These are obviously integral features, and having them up and running allows for very easy testing of the next phases of the program. Additionally, further work was done to the Math child class to smoothly implement into the program when it is complete</p>
  <h2>What was Accomplished:</h2>
    <ul>
      <li>Frontend redesigned</li>
      <li>Nav Bar created</li>
      <li>Subjects Page created</li>
      <li>Import feature added</li>
      <li>Export feature added</li>
      <li>Add subject feature added</li>
      <li>Remove subject feature added</li>
      <li>Math subject tidying</li>
    </ul>
  <h2>Problems Encountered:</h2>
    <ul>
      <li>Storage of User data</li>
      <li>Obtuse front end design</li>
    </ul>
  <h2>Plans to Overcome Problems:</h2>
    <p>We always planned to keep OpenAI from answering math questions. This was due to the fact that ChatGPT has always had unreliable answers to math questions. Using sympy, we can have ChatGPT make the questions and then create the answers using that tool.</p>
  <h2>Plans for this Week:</h2>
    <ul>
      <li>Begin Study section</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
      <li>Cameron Calhoun - Weekly Report, Presentation, NavBar, front end redesign,
  Import/Export, Subjects page, Add/Remove Subject</li>
      <li>Gage Keslar - Further tested Math class</li>
      <li>Seth Morgan - Further tested Math class</li>
      <li>Jonathan Buckel - Further tested Math class</li>
    </ul>

  </div>
  `,
  week10: `
  <div class="article-typography">
  <h1>Week 10</h1>
  <h2>Summary:</h2>
    <p>With this week, our focus was directed towards the study page. With this page being
  implemented, the question and answer communication demonstrated in our class demos from
  weeks prior is now online, and usable through the website. This is a massive chunk of our
  program, and the main deliverable which was promised in our design documents, and is thus
  something to be celebrated for coming online. Most of the work was done in creating the class
  demos in weeks prior, so all that really needed to be done was translating it to this client server
  call and response functionality. Additionally, some quality of life changes were added to the
  website. A loading page was added to the websites landing page such that when the client is
  connecting to the API, they cannot make requests to it until it has already connected. The
  subjects page was also cleaned up after feedback from last week to hopefully improve clarity of
  use for users.</p>
  <h2>What was Accomplished:</h2>
    <ul>
      <li>Study page created</li>
      <li>Study page functionality implemented</li>
      <li>Loading screen added</li>
      <li>Subjects page tidied up</li>
    </ul>
  <h2>Problems Encountered:</h2>
    <ul>
      <li>Testing of question generation, and overall website testing</li>
    </ul>
  <h2>Plans to Overcome Problems:</h2>
    <p>We always planned to keep OpenAI from answering math questions. This was due to the fact that ChatGPT has always had unreliable answers to math questions. Using sympy, we can have ChatGPT make the questions and then create the answers using that tool.</p>
  <h2>Plans for this Week:</h2>
    <ul>
      <li>Test website</li>
      <li>Play with breakpoints / prompts / points gained / points lost values</li>
      <li>Begin Flashcards</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
      <li>Cameron Calhoun - Weekly Report, Presentation, loading screen, study page, study page functionality, subject page redesign</li>
      <li>Gage Keslar - Further tested Math class</li>
      <li>Seth Morgan - Further tested Math class</li>
      <li>Jonathan Buckel - Further tested Math class</li>
    </ul>

  </div>
  `,
  week11: `
  <div class="article-typography">
  <h1>Week 11</h1>
  <h2>Summary:</h2>
    <p>This week, a few main goals were accomplished. To begin, part of the flashcards page was created. Users can now save questions generated in Study to a flashcard. These
```

```
flashcards are stored in their saved data, and can be accessed in Flashcards. These flashcards are grouped by subject, and allow the user to go back and review questions they specifically chose to save. This is not exactly fitting our design document, but we believe it to be a substantially more valuable addition to the program. The legacy flashcards feature will be added next week. Additionally, the subjects prompts were tweaked to allow for more engaging levels of difficulty. Computer Science benefits the most from this, given its wide range of topics. Finally, some more work for Math was completed. Math will have the AI generate the expressions in a format that can be displayed in markdown, and sent to sympy to be solved / verified. All in all, a majority of the website is completed at this point. The remaining time will be dedicated to implementing math, implementing the legacy flashcards feature, and further tweaking of AI prompting.</p>
<h2>What was Accomplished:</h2>
  <ul>
    <li>Flashcards created</li>
    <li>Subjects further tweaked</li>
    <li>Math further troubleshooted</li>
  </ul>
<h2>Problems Encountered:</h2>
  <ul>
    <li>How to display math problems</li>
  </ul>
<h2>Plans to Overcome Problems:</h2>
  <p>Math problems are a little unique, and must be displayed a certain way due to their use of symbols. We will be working around this by displaying a markdown renderer on the web page, and placing the math expressions inside of it.</p>
<h2>Plans for this Week:</h2>
  <ul>
    <li>Further test website</li>
    <li>Play more with breakpoints / prompts / points gained / points lost values</li>
    <li>Create legacy Flashcards</li>
    <li>Get Math online</li>
  </ul>
<h2>Member Contributions:</h2>
  <ul>
    <li>Cameron Calhoun - Weekly Report, Presentation, Flashcards, Math, bug-fixing, prompt engineering.</li>
    <li>Gage Keslar - Further tested Math class</li>
    <li>Seth Morgan - Further tested Math class</li>
    <li>Jonathan Buckel - Further tested Math class, experimented with markdown</li>
  </ul>

</div>
`,
week12: `
<div class="article-typography">
<h1>Week 12</h1>
<h2>Summary:</h2>
  <p>We are nearly complete with the majority of the program. To begin, lets talk about legacy flashcards. Flashcards has now been split into two separate functions. Legacy flashcards, and interactive flashcards. Interactive flashcards is the function we implemented last week, where a user can save questions generated when using ‘Study’, and review them later as a flashcard. Legacy flashcards is a new feature, which allows users to generate six questions of a given subject (given their current ELO in the subject) and export the questions to a PDF file, which can then be printed and used as flashcards. The formatting of this document is designed so that it can be folded along a line, where the front of the card has the question, and the back has the answer. This is in line with our original vision for flashcards as described in our design document. In addition to flashcards, we began to finalize Math, and have found a direction to take in implementing it. All text boxes in the program which would store questions and answers before have been replaced with Markdown renderer objects. These objects support markdown rendering, and when utilizing another package to convert LaTeX format to markdown, we can directly render math expressions on our website as you would find them in a math textbook. With a format in mind that works, we have tweaked the Math subject class to generate questions in accordance to this format. Currently, the program is able to identify expressions and variables generated from these questions. This is relevant to how Sympy solves math expressions. The only remaining function for the program is to get the actual Math subject online, which should be a brief task with a little bit of bug fixing.</p>
<h2>What was Accomplished:</h2>
  <ul>
    <li>Flashcards separated into Legacy Flashcards, and Interactive Flashcards</li>
    <li>Legacy Flashcards created</li>
    <li>Markdown Renderer implemented to display math equations</li>
    <li>Math subject class tweaked to generate questions in the form of LaTeX syntax</li>
  </ul>
```

```
<h2>Problems Encountered:</h2>
  <ul>
    <li>Regular expressions to match variable names</li>
  </ul>
<h2>Plans to Overcome Problems:</h2>
  <p>In our program, we use regular expressions to match the portions of the response generated by ChatGPT. This is done to consistently extract each portion from one large chunk of text, where the contents usually differ. We ran into an issue where we could not consistently match the variables of some expressions. This is required, as declaring the variables in an expression is needed by Sympy. If you were to say, match for any letter, A-Z, an operator like sqrt() would get consumed as a variable. This was circumvented, by restricting the characters that can be used as variables, and changing our regular expression to account for that.</p>
<h2>Plans for this Week:</h2>
  <ul>
    <li>Further test website</li>
    <li>Play more with breakpoints / prompts / points gained / points lost values</li>
    <li>Get Math online</li>
  </ul>
<h2>Member Contributions:</h2>
  <ul>
    <li>Cameron Calhoun - Weekly Report, Presentation, Flashcards separation, Legacy Flashcards, Math rendering thru Markdown, bug-fixing, prompt engineering.</li>
    <li>Gage Keslar - Further tested Math class</li>
    <li>Seth Morgan - Further tested Math class</li>
    <li>Jonathan Buckel - Further tested Math class, tweaked regular expression</li>
  </ul>

</div>
`,
week13: `
<div class="article-typography">
<h1>Week 13</h1>
<h2>Summary:</h2>
```

```
    <p>The program is now complete with respect to the design documents. The big ticket item of this week is that Math is now online and functional. The program currently provides three tiers of Algebraic equations for the user to solve. With Math being online, everything outlined in the design documents has been implemented, and all remaining changes are additional. The first big additional change added was Interactive flashcards. This provides users with a more intuitive way to utilize flashcards then we initially designed. However, the utility of exporting your questions to a PDF from legacy flashcards proved to be very handy. We added this feature to Interactive flashcards, so a user can print out all of their saved flashcards from a given subject. Now that the program is complete, all remaining weeks will be focused on additional quality of life changes, and bug fixes.</p>
  <h2>What was Accomplished:</h2>
    <ul>
      <li>Math subject finished</li>
      <li>Interactive flashcards printable</li>
      <li>Various bug fixes</li>
    </ul>
  <h2>Problems Encountered:</h2>
    <ul>
      <li>Math</li>
    </ul>
  <h2>Plans to Overcome Problems:</h2>
    <p>Like many weeks prior, Math was the big problem creator of the week. First, we removed Sympy in favor of a Javascript library, Nerdamer. This is due to the fact that we are already extracting from the Ai’s response in the frontend, and recreating this in the backend would be a little wasteful. Nerdamer functions the exact same way as Sympy, solving symbolic equations when variables are provided to it. Due to how the data needed to be passed to Nerdamer, making sure the expressions were provided by the AI in a precise consistent manner was more important here then anywhere else in the program. This is why we have limited the scope of Math to Algebra. There are many edge cases to account for, such as ChatGPT generating the question with the correct answer, but not specifying it as the correct answer, generating the question without the correct answer present, generating questions where the answer contains multiple values, generating questions that are logically unsolvable, and contain no solution, and more than that! We’ve narrowed down most of these, but more could be present. Fallback logic was created to regenerate a question with a fresh set of context in order to address any edge cases not yet found. In addition to the above, the way Chatbots initial context is defined now varies on the subject. If the subject is Math, the context has a series of rules appended to restrict the prompts as described above</p>
  <h2>Plans for this Week:</h2>
    <ul>
      <li>Further test website</li>
      <li>Play more with breakpoints / prompts / points gained / points lost values</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
      <li>Cameron Calhoun - Weekly Report, Presentation, printing Interactive Flashcards to PDF, removal of Sympy and inclusion of math.js, implemented final Math subject version</li>
      <li>Gage Keslar - Further tested Math class</li>
      <li>Seth Morgan - Further tested Math class</li>
      <li>Jonathan Buckel - Further tested Math class, found a lot of issues in regards to question generation</li>
    </ul>

  </div>
  `,
  week14: `
  <div class="article-typography">
  <h1>Week 14</h1>
  <h2>Summary:</h2>
    <p>With the program being “complete” last week, this week was focused entirely on quality of life changes, and polishing the edges. The progress page was something we designed in week 2, and thus, is outdated. We polished it up by styling it to be consistent with the rest of the program, and adding links to all documents written in the creation of this program. Additionally, the website is now navigable to the user, through the about page. You can also return to ASC through progress which was also unavailable before. With about, it is a similar story. About was fleshed out to provide instructions on using the application, and links to the progress page and a new feature, doxygen. Doxygen is an application that auto generates documents on classes found within your program. This is not required, but we figured it would be a nice addition to those more interested in the technical nature of the program. Additionally, the flashcards menu page had descriptions added in order to assist users in more easily navigating the website. In the same vein, the card on the home page now updates its messaging based on the hovered button. The final change made was to replace the browser alerts with less intrusive, more thematically cohesive toasts. These fit the design of the website, and do not interrupt the user as the alerts once did.</p>
  <h2>What was Accomplished:</h2>
    <ul>
```

```
      <li>Bunch of small quality of life tweaks</li>
      <li>Progress page updated</li>
      <li>About page updated</li>
      <li>Doxygen added</li>
      <li>Flashcards menu updated</li>
      <li>Browser alerts replaced with toasts</li>
      <li>Home page card assists users in navigation</li>
    </ul>
  <h2>Member Contributions:</h2>
    <ul>
      <li>Cameron Calhoun - Weekly Report, Presentation, User manual, final presentation, QOL Tweaks, Progress page, About page, Flashcards, Toasts, Home page card</li>
      <li>Gage Keslar - User manual, final presentation</li>
      <li>Seth Morgan - User manual, final presentation</li>
      <li>Jonathan Buckel - Doxygen, User manual, final presentation</li>
    </ul>

  </div>
  `

}

export default articles
```

File: AiStudyCompanion\ASC\src\pages\About.jsx

```
import { useState } from 'react'
import { useNavigate } from "react-router-dom";
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'

```

```
function About() {
    const navigate = useNavigate();

    return (
        <>
            {/* Full Page Layout */}
            <div className="grid grid-rows-[auto_1fr] min-h-screen">
                <TopBar title="About ASC" />
                <div>
                    {/* About Article */}
                    <div className="p-10 max-w-6xl mx-auto">
                        <h2 className="text-3xl font-bold mb-4">What is AI Study Companion?</h2>
                        <p className="text-lg mb-4">
                            AI Study Companion (ASC) is a collaborative project spawned from PennWest California's Senior Project: Software Engineering course.
                            This project is headed by four Computer Science students of PennWest California:</p>
                        <ul className="list-disc pl-6 text-lg space-y-2 mb-4">
                            <li>Cameron Calhoun</li>
                            <li>Gage Keslar</li>
                            <li>Jonathan Buckel</li>
                            <li>Seth Morgan</li>
                        </ul>
                        <p className="text-lg mb-4">
                            AI Study Companion is a website built to utilize the latest large language model capabilities to provide a convenient
                            solution to studying given subjects. AI Study Companion provides users with a set of subjects in which they can interactively test their skills in
                            with ASC adjusting the difficulty based on your individual progress.
                        </p>


                        <h2 className="text-3xl font-bold mt-8 mb-4">Project Progress</h2>
                        <div className="border-3 p-4 border-pwred rounded-xl p-6 rounded-2xl shadow-md text-lg space-y-2">
                            <p>
                              Curious about how ASC has developed over time? We maintain a transparent development log to track our goals, updates, and milestones.
                            </p>
                            <p>
                              Visit our <span className="text-blue-600 cursor-pointer underline" onClick={() => navigate("/progress")}>Progress</span> page to follow along with the journey and see how the site has evolved since its inception.
                            </p>
                        </div>

                        <h2 className="text-3xl font-bold mt-8 mb-4">Why Use AI Study Companion?</h2>
                        <ul className="list-disc pl-6 text-lg space-y-2">
                            <li>Personalized learning paths based on your progress.</li>
                            <li>Instant explanations and interactive problem-solving.</li>
                            <li>Access to a vast library of study materials.</li>
                            <li>Time-efficient and engaging study sessions.</li>
                        </ul>

                        <h2 className="text-3xl font-bold mt-8 mb-4">How It Works</h2>
                        <p className="text-lg mb-4">
                            Simply visit the website, select your subject of interest, and start interacting with AI to get
                            step-by-step guidance. The more you use the tool, the better it adapts to your learning style.
                            Need to study on the go? Just export your progress to a portable file and import it on any other machine.
                        </p>
```

```

            <h2 className="text-3xl font-bold mt-8 mb-4">Explore the Features</h2>
            <div className="text-lg space-y-4">
              <p><strong>Study:</strong> Practice with AI-generated questions tailored to your progress and performance.</p>
              <p><strong>Flashcards:</strong> Create and study randomized flashcards, or review ones you've saved for targeted practice.</p>
              <p><strong>Subjects:</strong> Manage your subjects, add new topics, and view detailed scores for each to track your growth.</p>
              <p><strong>Import & Export:</strong> Use the navbar buttons to save your progress as a portable file or load existing progress easily.</p>
            </div>

            <h2 className="text-3xl font-bold mt-8 mb-4">More Resources</h2>
            <div className="text-lg space-y-4">
              <p>
                Need help navigating ASC or want technical details about how it works? Check out our <span className="text-blue-600 cursor-pointer underline" onClick={() => window.location.href = "/docs/html/index.html"}>Docs</span> page for guides, FAQs, and contributor info.
              </p>
            </div>
            {/* Back Button */}
            <div className="mt-10 flex justify-center">
              <Button text="Back" onClick={() => navigate("/")} />
            </div>
          </div>
        </div>
      </div>


    </>
  );
```

```
}

export default About;

```

File: AiStudyCompanion\ASC\src\pages\Flashcards.jsx

```
import { useState } from 'react'
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'
import { useNavigate } from "react-router-dom"

function Flashcards() {
    const navigate = useNavigate()
    return (
        <>
            {/* Full Page Layout */}
```

```
      <div className="grid grid-rows-[auto_1fr] min-h-screen">
        <TopBar title="Flashcards" />
        <div className='flex justify-center items-center h-full'>
          <div className='flex w-full gap-80 p-30 h-full justify-between items-center'>
            <div className='flex flex-col items-center w-1/2 h-1/2'>
              <Button
                text='Legacy Flashcards'
                className='p-20'
                onClick={() => navigate('/legacy_flashcards')} />
              <h1 className='text-center text-2xl w-2/3 pt-5 border-3 p-4 border-pwred rounded-xl m-10'>
                Randomly generate five questions of a chosen subject that match your skill level, and export them as a PDF!
              </h1>
            </div>

            <div className='flex flex-col items-center w-1/2 h-1/2'>
              <Button
                text='Interactive Flashcards'
                className='p-20'
                onClick={() => navigate('/interactive_flashcards')} />
              <h1 className='text-center text-2xl w-2/3 pt-5 border-3 p-4 border-pwred rounded-xl m-10'>                Utilize flashcards saved during Study. Interact with them here, or export them as a PDF!
              </h1>
            </div>
          </div>
        </div>
      </div >
    </>
  );
}
```

```

export default Flashcards;

```

File: AiStudyCompanion\ASC\src\pages\Home.jsx

```
import axios from "axios"
import { useEffect, useState } from "react"
import { useNavigate } from "react-router-dom"
import '../styles.css'
import Button from '../components/Button.jsx'
import Card from '../components/Card.jsx'
import TopBar from '../components/TopBar.jsx'
import { useToast } from "../contexts/ToastContext"

import { API_HELLO } from "../config/api.js"
import { handleExport, handleImport } from "../utils/fileHandlers"


```

```
function Home() {
    const navigate = useNavigate()
    const { addToast } = useToast()

    const [message, setMessage] = useState("")
    const [loading, setLoading] = useState(true)
    const [delayed, setDelayed] = useState(false)
    const [cardMessage, setCardMessage] = useState("Make a selection on the left to begin!")


    useEffect(() => {
        setLoading(true)
        setDelayed(false)

        const delayTimer = setTimeout(() => setDelayed(true), 5000)

        axios.get(API_HELLO)
            .then((response) => {
                setMessage(response.data.message)
            })
            .catch((error) => {
                console.error("Error fetching data:", error)
            })
            .finally(() => {
                clearTimeout(delayTimer)
                setLoading(false)
            })
    }, [])

    return (
<>
  {loading &&
    <div className="flex flex-col justify-center items-center text-center min-h-screen">
      <h1 className="text-pwblue text-6xl font-semibold py-10">ASC is loading, please wait</h1>
      <div class="col-3">
        <div class="snippet" data-title="dot-flashing">
          <div class="stage">
            <div class="dot-flashing"></div>
          </div>
        </div>
      </div>
    </div>}
  {/* Full Page Layout */}
  {!loading &&
    <div className="grid grid-rows-[auto_1fr] min-h-screen">
      <TopBar title={`Welcome to ASC!`} />
      {/* Buttons + Card */}
      <div className="my-25 grid grid-cols-[53.75%_46.25%]">
        {/* Buttons */}
        <div className="flex items-center justify-center">
          <div className="grid grid-cols-2 gap-20 w-full h-full p-20">
            <Button text="Study"
              onClick={() => navigate("/study")}
              onMouseEnter={() => setCardMessage("Study your tracked subjects with the help of AI!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
            <Button text="Flashcards"
              onClick={() => navigate("/flashcards")}
              onMouseEnter={() => setCardMessage("Review saved questions in flashcard format, or let ASC generate some for you!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
            <Button text="Import"
              onClick={() => handleImport(addToast)}
              onMouseEnter={() => setCardMessage("Import your saved data to pick up where you left off!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
            <Button text="Export"
              onClick={() => handleExport(addToast)}
              onMouseEnter={() => setCardMessage("Export your data to study on the go!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
            <Button text="Subjects"
              onClick={() => navigate("/subjects")}
              onMouseEnter={() => setCardMessage("Add new subjects, and track your progress!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
            <Button text="About"
              onClick={() => navigate("/about")}
              onMouseEnter={() => setCardMessage("Learn more about this app!")}
              onMouseLeave={() => setCardMessage("Make a selection on the left to begin!")} />
          </div>
        </div>
        {/* Card */}
        <div className='flex items-center justify-center'>
          <Card text={cardMessage} />
        </div>
```

```
                    </div>
                </div>
            }
        </>
    )
}

export default Home

```

File: AiStudyCompanion\ASC\src\pages\LegacyFlashcards.jsx

```
import { useEffect, useState } from 'react'
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'
import { API_GENERATE_FLASHCARDS } from '../config/api'
import axios from 'axios';
import { generatePDF } from '../utils/pdfHandlers'
import { useNavigate } from 'react-router-dom'

function LegacyFlashcards() {
    const navigate = useNavigate()
    const [subjects, setSubjects] = useState([])
    const [loading, setLoading] = useState(false)
    const [selectedSubject, setSelectedSubject] = useState("")
    const [newChat, setNewChat] = useState(Boolean)
```

```

// useEffect to load subjects, and listen for subject changes
useEffect(() => {
   const loadSubjects = () => {
      const storedData = sessionStorage.getItem("importedSubjects")
      setSubjects(storedData ? JSON.parse(storedData) : [])
   }

   loadSubjects()

   const handleUpdate = () => loadSubjects()

   window.addEventListener("subjectsUpdated", handleUpdate)
   return () => window.removeEventListener("subjectsUpdated", handleUpdate)
}, [])

// on subject change, set selected subject, and set new chat to true
const handleSubjectChange = (event) => {
   setSelectedSubject(event.target.value);
   setNewChat(true)
}

const handleGenerateFlashcards = async (selectedSubject) => {
   if (!selectedSubject) {
      console.error("No subject selected!")
      return
   }

   const currentSubjects = JSON.parse(sessionStorage.getItem("importedSubjects")) || []
   try {
```

```
setLoading(true)
const payload = {
    subjects: currentSubjects,
    curSubject: selectedSubject,
    newChat: newChat,
}
console.log(payload)
const response = await axios.post(API_GENERATE_FLASHCARDS, payload, {
    headers: { "Content-Type": "application/json" }
})
const ai_responses = response.data.ai_response
console.log(ai_responses)



const questionRegex = /^Question:\s([\s\S]+?)^(?=[A-D]\))/m;

const answerChoicesRegex = /([A-D])\)\s(.+)/g
const correctAnswerRegex = /Answer:\s([A-D])/

const parsedFlashcards = ai_responses.map(ai_response => {

    const matchQuestion = ai_response.match(questionRegex)
    const matchAnswerChoices = [...ai_response.matchAll(answerChoicesRegex)]
        .map(m => `${m[1]}) ${m[2].trim()}`)

    const matchCorrectAnswer = ai_response.match(correctAnswerRegex)
    const limitedAnswerChoices = matchAnswerChoices.slice(0, 4)
    return {
        question: matchQuestion ? matchQuestion[1].trim() : "",
                choices: limitedAnswerChoices,
                answer: matchCorrectAnswer ? matchCorrectAnswer[1] : "",
            }
        })
        console.log(parsedFlashcards)
        generatePDF(parsedFlashcards, selectedSubject)
        setLoading(false)

    }
    catch (error) {
        console.error("Error generating question: ", error)
    }
}




return (
    <>
        {/* Full Page Layout */}
        <div className="grid grid-rows-[auto_1fr] min-h-screen">
            <TopBar title="Legacy Flashcards" />
            <div>
                <div className='flex justify-between border-3 p-4 border-pwred rounded-xl
items-center mb-8 gap-5 h-1/8'>
                    {subjects.length === 0 &&
                        <div className="w-1/3 h-full flex justify-center items-center
                                text-shadow text-3xl font-bold text-[#F3F4F6] text-center gap-2">
                            <Button text="Add a subject." onClick={() => navigate("/subjects")} />
                        </div>
                    }
```

```

{subjects.length !== 0 &&
  <form className='w-1/3 pl-17'>
    <label for="subjects" className="block mb-2 text-xl font-semibold text-gray-900"></label>
    <select id="subjects"
      name="subjects"
      className='bg-gray-50 border border-gray-300 text-gray-900 text-lg rounded-lg focus:ring-pwblue focus:border-blue-500 block w-full p-2.5'
      onChange={handleSubjectChange}>
      <option value="" hidden disabled selected>
        {subjects.length === 0 ? 'Please add a subject in "Subjects".' : "Subjects"}
      </option>
      {subjects.map((subject, index) => (
        <option key={index} value={subject.subjectName}>
          {subject.subjectName}
        </option>
      ))}
    </select>
  </form>

}
<div className="w-1/3 h-full flex justify-center items-center
        text-shadow text-3xl font-bold text-[#F3F4F6] text-center gap-2">
  <Button
    text='Generate Flashcards'
    onClick={() => { handleGenerateFlashcards(selectedSubject) }} />
</div>

<div className="w-1/3 h-full bg-pwblue rounded-xl shadow-xl flex justify-center items-center
```

```
          text-shadow text-3xl font-bold text-[#F3F4F6] text-center">
    {subjects.length === 0 ? (
      <h1>No subject found</h1>

    ) : !selectedSubject ? (
      <h1>No subject selected</h1>
    ) : (
      <div className='w-full flex'>
        <h1 className="w-1/2">
          {selectedSubject}
        </h1>
        <h1 className="w-1/2">
          Elo: {subjects.find(subject => subject.subjectName === selectedSubject)?.subjectElo || ""}
        </h1>
      </div>
    )}
  </div>
</div>
{loading &&
  <div className='flex flex-col justify-center items-center pt-8'>
    <h1 className="text-pwblue text-6xl font-semibold py-10">Generating Flashcards</h1>
    <div className="col-3">
      <div className="snippet" data-title="dot-flashing">
        <div className="stage">
          <div className="dot-flashing"></div>
        </div>
      </div>
    </div>
  </div>
```

```
                    }
                </div>


            </div>
        </>
    );
}

export default LegacyFlashcards;

```

File: AiStudyCompanion\ASC\src\pages\NewFlashcards.jsx

```
import { useState, useEffect } from 'react'
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'
import ReactMarkdown from 'react-markdown';
import remarkMath from 'remark-math';
import rehypeKatex from 'rehype-katex';
import "katex/dist/katex.min.css"
import { generatePDF } from '../utils/pdfHandlers';

// EXPORT CUR SUBJECTS FLASHCARDS TO PDF SIMILAR TO LEGACY FLASHCARDS
//
// for flashcards[currentSubject] {
//  generatePdf()?
//  may need to tweek legacy flashcards syntax such that the same function can be used
// }

function NewFlashcards() {
```

```
const [flashcards, setFlashcards] = useState(() => {
  const flashcardsData = sessionStorage.getItem("flashcards")
  return flashcardsData ? JSON.parse(flashcardsData) : {}
})

const subjects = Object.keys(flashcards)

const [lines, setLines] = useState("")
const [currentSubjectIndex, setCurrentSubjectIndex] = useState(0)
const [currentFlashcardIndex, setCurrentFlashcardIndex] = useState(0)
const [answerDisplayed, setAnswerDisplayed] = useState(Boolean)
const handleQuestionDisplayedState = () => {
  if (answerDisplayed == false) {
    setAnswerDisplayed(true)
  } else {
    setAnswerDisplayed(false)
  }
}

const currentSubject = subjects[currentSubjectIndex];
const currentFlashcard = flashcards[currentSubject] && flashcards[currentSubject][currentFlashcardIndex];

const choices = currentFlashcard ? currentFlashcard.choices : []


const prevSubject = () => {
  if (answerDisplayed) { handleQuestionDisplayedState() }
  setCurrentSubjectIndex((prev) => (prev > 0 ? prev - 1 : subjects.length - 1));
  setCurrentFlashcardIndex(0)
}
```

```

const nextSubject = () => {
    if (answerDisplayed) { handleQuestionDisplayedState() }
    setCurrentSubjectIndex((prev) => (prev < subjects.length - 1 ? prev + 1 : 0));
    setCurrentFlashcardIndex(0)
}

const prevFlashcard = () => {
    const subject = subjects[currentSubjectIndex]
    setCurrentFlashcardIndex((prev) => (prev > 0 ? prev - 1 : flashcards[subject].length - 1))
    if (answerDisplayed) { handleQuestionDisplayedState() }
};

const nextFlashcard = () => {
    if (answerDisplayed) { handleQuestionDisplayedState() }
    const subject = subjects[currentSubjectIndex];
    setCurrentFlashcardIndex((prev) => (prev < flashcards[subject].length - 1 ? prev + 1 : 0));
};


const removeFlashcard = () => {
    const subject = subjects[currentSubjectIndex]
    if (!flashcards[subject]) return

    const updatedFlashcards = { ...flashcards }

    if (flashcards[subject].length > 0) {
        updatedFlashcards[subject].splice(currentFlashcardIndex, 1)

    if (updatedFlashcards[subject].length === 0) {
      delete updatedFlashcards[subject]

      setCurrentSubjectIndex((prev) => (subjects.length > 1 ? (prev > 0 ? prev - 1 : 0) : 0))
    } else {
      setCurrentFlashcardIndex((prev) => (prev > 0 ? prev - 1 : 0))
    }
  }

  if (answerDisplayed) { handleQuestionDisplayedState() }
  setFlashcards(updatedFlashcards)
  console.log(flashcards)
  sessionStorage.setItem("flashcards", JSON.stringify(updatedFlashcards))
}

useEffect(() => {
  if (currentSubject === "Math" && currentFlashcard?.question) {
    const splitLines = currentFlashcard.question.split('\n')
    setLines([splitLines[0] || "", splitLines[1] || ""])
  } else {
    setLines("")
  }
}, [currentFlashcard, currentSubject])

return (
  <>
    {/* Full Page Layout */}
    <div className="grid grid-rows-[auto_1fr] min-h-screen">
      <TopBar title="Interactive Flashcards" />

```

```
<div className='flex justify-center'>
  <div className="p-4 text-center">
    {/* Display subjects if available */}
    {subjects.length > 0 ? (
      <div className='w-full flex justify-center pb-4'>
        <div className="w-3/5 gap-20 flex items-center justify-between">
          <div className='w-[180px]'>
            <Button text="<" onClick={prevSubject} />
          </div>
          <h2 className="text-4xl font-semibold text-pwblue  whitespace-nowrap">{subjects[currentSubjectIndex]}</h2>
          <div className='w-[180px]'>
            <Button text=">" onClick={nextSubject} />
          </div>

        </div>
      </div>

    ) : (
      <p className="text-gray-500">No flashcards available.</p>
    )}

    {currentFlashcard ? (
      <div className="w-full">
        <div className="grid grid-cols-[1fr_3fr_1fr] gap-4">

          <div className="flex justify-end items-center w-full h-full">
            <div className="w-1/3 h-12">
              <Button text="<" onClick={prevFlashcard} />
            </div>
          </div>
```

```

<div className="w-full p-5 border-3 border-pwred bg-pwblue text-[#F3F4F6] rounded-xl shadow-xl flex items-center justify-center gap-8
  hover:bg-[#0055a4] hover:border-[#ff4a3d] hover:text-[#ffffff] min-h-[200px] min-w-[700px]"
  onClick={handleQuestionDisplayedState}>
  {!answerDisplayed && (
    <>
      <div className="text-shadow text-4xl font-bold  w-1/2 text-center">
        {currentSubject != 'Math' &&
          <ReactMarkdown
            remarkPlugins={[remarkMath]}
            rehypePlugins={[rehypeKatex]}
          >
            {currentFlashcard.question}
          </ReactMarkdown>
        }
        {currentSubject == 'Math' &&
          <>
            <ReactMarkdown
              remarkPlugins={[remarkMath]}
              rehypePlugins={[rehypeKatex]}
            >
              {lines[0]}
            </ReactMarkdown>
            <ReactMarkdown
              remarkPlugins={[remarkMath]}
              rehypePlugins={[rehypeKatex]}
            >
              {lines[1]}
      </ReactMarkdown>
    </>
  }

</div>
<div className="text-shadow text-2xl font-bold w-1/2 text-left">
  <div className="py-1">
    <ReactMarkdown
      remarkPlugins={[remarkMath]}
      rehypePlugins={[rehypeKatex]}
    >
      {choices[0]}
    </ReactMarkdown>
  </div>
  <div className="py-1">
    <ReactMarkdown
      remarkPlugins={[remarkMath]}
      rehypePlugins={[rehypeKatex]}
    >
      {choices[1]}
    </ReactMarkdown>
  </div>
  <div className="py-1">
    <ReactMarkdown
      remarkPlugins={[remarkMath]}
      rehypePlugins={[rehypeKatex]}
    >
      {choices[2]}
    </ReactMarkdown>
  </div>
```

```
          <div className="py-1">
            <ReactMarkdown
              remarkPlugins={[remarkMath]}
              rehypePlugins={[rehypeKatex]}
            >
              {choices[3]}
            </ReactMarkdown>
          </div>
        </div>
      </>)}
    {answerDisplayed && (
      <div className="text-shadow text-4xl font-bold  w-1/2 text-center"><ReactMarkdown
        remarkPlugins={[remarkMath]}
        rehypePlugins={[rehypeKatex]}
      >
        {currentFlashcard.answer}
      </ReactMarkdown></div>
    )}
  </div>

  <div className="flex justify-start items-center w-full h-full">
    <div className="w-1/3 h-12">
      <Button text=">" onClick={nextFlashcard} />

    </div>
  </div>

</div>
<div className='flex justify-center items-center w-full h-full flex-col p-4 gap-8'>
```

```
                  <h3>Click on Flashcard to flip sides.</h3>
                  <div className="w-1/3 h-12">
                    <Button text="Remove Flashcard" onClick={removeFlashcard} />
                  </div>
                  <div className="w-1/3 h-12">
                    <Button text={`Save ${currentSubject} cards to PDF`} onClick={() => generatePDF(flashcards[currentSubject], currentSubject)} />
                  </div>

                </div>

              </div>


            ) : (
              <p>No flashcards in this subject</p>
            )}
          </div>
        </div>
      </div>
    </>
  );
}

export default NewFlashcards;
```

File: AiStudyCompanion\ASC\src\pages\Progress.jsx

```
import { useState } from "react";
import '../styles.css'
import articles from "./progressFiles/articles"
import { useNavigate } from "react-router-dom";
import Button from "../components/Button";

const sections = Object.keys(articles).map((id) => ({ id, label: id.charAt(0).toUpperCase() + id.slice(1) }));

export default function Progress() {
    const [selectedSection, setSelectedSection] = useState("about");
    const navigate = useNavigate();

    return (
        <div className="h-screen flex flex-col bg-bgwhite text-pwblue">
            {/* Top Bar */}
            <header className="w-full h-1/8 bg-pwblue flex items-center justify-center">
                <div className="w-3/4 h-1/2 bg-bgwhite rounded-xl shadow-2xl flex items-center justify-center">
                    <h1 className="text-3xl font-bold">
                        <span className="text-pwred">ASC</span>{' '}
                        <span className="text-pwblue">Progress Reports</span>
                    </h1>
                </div>
            </header>

```

```
      {/* Main Content */}
      <div className="flex flex-1 overflow-hidden my-4">

        {/* Side Bar */}
        <aside className="w-1/6 h-full bg-bgwhite flex flex-col items-center px-3 py-4
          border-3 p-4 border-pwred rounded-xl overflow-y-auto">

          {/* Home Button */}
          <button
            className="w-full mb-4 bg-transparent border-3 border-pwblue rounded-full shadow-2xl flex items-center justify-center py-3
        hover:border-pwred hover:shadow-3xl hover:scale-105
        active:border-pwblue active:scale-95 active:shadow-md transition-all duration-300"
            onClick={() => navigate("/")}
          >           <h1 className="text-2xl font-bold text-white">
              <span className="text-pwred">A</span>
              <span className="text-pwblue">SC</span>
            </h1>
          </button>

          {/* Section Buttons */}
          {sections.map((section) => (
            <button
              key={section.id}
              className="w-full mt-2 bg-pwblue rounded-full shadow-2xl flex items-center justify-center py-3
            hover:bg-red-400 hover:shadow-3xl hover:scale-105
            active:bg-pwred active:scale-95 active:shadow-md transition-all duration-300"
              onClick={() => setSelectedSection(section.id)}
            >
              <h1 className="text-2xl font-bold text-white">{section.label}</h1>
            </button>
          ))}
        </aside>

        {/* Article Section */}
        <main className="w-full h-full overflow-y-auto bg-bgwhite p-5">
          <article
            id="main-article"
            className="prose max-w-screen-xl mx-auto w-full p-8 bg-white shadow-lg rounded-lg"
            dangerouslySetInnerHTML={{ __html: articles[selectedSection] }}
          />
        </main>
      </div>

      {/* Sticky Download Section */}
      <div className="sticky bottom-0 bg-bgwhite border-t-4 border-pwred p-3 flex items-center justify-between space-x-4">
        <h2 className="text-xl font-bold text-pwblue">Download Relevant Documents</h2>
        <div className="flex space-x-4">
          <a
            href="/pdfs/project_proposal.pdf"
            download
            className="text-pwblue underline hover:text-pwred transition-colors duration-300"
          >
            Project Proposal
          </a>
          <a
            href="/pdfs/requirements_document.pdf"
```

```
            download
            className="text-pwblue underline hover:text-pwred transition-colors duration-300"
          >
            Requirements Document
          </a>
          <a
            href="/pdfs/specification_document.pdf"
            download
            className="text-pwblue underline hover:text-pwred transition-colors duration-300"
          >
            Specification Document
          </a>
          <a
            href="/pdfs/design_document.pdf"
            download
            className="text-pwblue underline hover:text-pwred transition-colors duration-300"
          >
            Design Document
          </a>
          <a
            //href="/pdfs/user_manual.pdf"
            download
            //className="text-pwblue underline hover:text-pwred transition-colors duration-300"
            className="text-gray-400 cursor-not-allowed"
          >
            User Manual
          </a>
        </div>
```

```
            </div>
        </div>
    );
}


```

File: AiStudyCompanion\ASC\src\pages\Study.jsx

```
import { useEffect, useRef, useState } from 'react'
import { useNavigate } from "react-router-dom";
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'
import axios from 'axios';
import { API_GENERATE_QUESTION } from '../config/api';
import ReactMarkdown from 'react-markdown';
import remarkMath from 'remark-math';
import rehypeKatex from 'rehype-katex';
import "katex/dist/katex.min.css"
import latex2js from 'latex-to-js/dist';
import nerdamer from 'nerdamer/all';
import 'nerdamer/Algebra';
import 'nerdamer/Solve';
import 'nerdamer/Calculus';
import { convertToLatex } from '../utils/convertLatex';
import { useToast } from '../contexts/ToastContext';

const markdown = `
$$
\\int_0^1 x^2 \\,dx = \\frac{1}{3}
$$
`
function Study() {
    const navigate = useNavigate()
    const { addToast } = useToast()

    const answerToNumKey = { "A": 0, "B": 1, "C": 2, "D": 3 }
    const [question, setQuestion] = useState("")
    const [lines, setLines] = useState("")
    const [answerChoices, setAnswerChoices] = useState([])
    const [answerSelection, setAnswerSelection] = useState("")
    const [correctAnswer, setCorrectAnswer] = useState("")
    const [explanation, setExplanation] = useState("")
    const [subjectsRight, setSubjectsRight] = useState([])
    const [subjectsWrong, setSubjectsWrong] = useState([])
    const [answerStatus, setAnswerStatus] = useState("")
    const prevAnswerStatusRef = useRef("")
    const [streak, setStreak] = useState(false)
    const [delta, setDelta] = useState(1)


    const [selectedSubject, setSelectedSubject] = useState("")
    const handleSubjectChange = (event) => {
       setSelectedSubject(event.target.value);
       setAnswerSelection("")
       setQuestion("")
       setLines("")
       setAnswerChoices([])
       setCorrectAnswer("")
       setExplanation("")
       setSubjectsRight([])
       setSubjectsWrong([])
       handleNewChatState(true)
       setStreak(false)
    };

    const [newChat, setNewChat] = useState(Boolean)
    const handleNewChatState = (set) => {
      setNewChat(set)
    }

    const [loading, setLoading] = useState(false)

    const [subjects, setSubjects] = useState([])

    useEffect(() => {
      const loadSubjects = () => {
        const storedData = sessionStorage.getItem("importedSubjects")
        setSubjects(storedData ? JSON.parse(storedData) : [])
      }

      loadSubjects()

      const handleUpdate = () => loadSubjects()

      window.addEventListener("subjectsUpdated", handleUpdate)
      return () => window.removeEventListener("subjectsUpdated", handleUpdate)
    }, [])

    useEffect(() => {
      if (answerSelection !== "") {
        const isCorrect = answerSelection === correctAnswer;

        if (isCorrect) {
          sessionStorage.setItem("importedSubjects", JSON.stringify(subjectsRight));

          if (prevAnswerStatusRef.current === "Correct") {
            setStreak(true);
          } else {
            setStreak(false);
          }

          setAnswerStatus("Correct");
          prevAnswerStatusRef.current = "Correct";
        } else {
          sessionStorage.setItem("importedSubjects", JSON.stringify(subjectsWrong));

          if (prevAnswerStatusRef.current === "Incorrect") {
            setStreak(true);
          } else {
            setStreak(false);
          }

          setAnswerStatus("Incorrect");
          prevAnswerStatusRef.current = "Incorrect";
        }

        window.dispatchEvent(new Event("subjectsUpdated"));
      }
    }, [answerSelection]);

    const handleGenerateQuestion = async (selectedSubject) => {
      if (!selectedSubject) {
        console.error("No subject selected!")
        addToast("No subject selected!")
        return
      }

      if (answerSelection != "") {
        setAnswerStatus("")
      }

      setAnswerSelection("")
      setQuestion("")
      setLines("")
      setAnswerChoices([])
      setCorrectAnswer("")
      setExplanation("")
      setSubjectsRight([])
      setSubjectsWrong([])

      const currentSubjects = JSON.parse(sessionStorage.getItem("importedSubjects")) || []
      try {
        setLoading(true)
        const payload = {
          subjects: currentSubjects,
          curSubject: selectedSubject,
          newChat: newChat,
          delta: delta,
          streak: streak

        }
        console.log(payload)
        const response = await axios.post(API_GENERATE_QUESTION, payload, {
          headers: { "Content-Type": "application/json" }
        })

        let question = ""
        let answerChoices = []
        let correctAnswer = ""
        let explanation = ""

        console.log("Raw data: ", response.data.ai_response)
        const ai_response = response.data.ai_response





        const questionRegex = /^Question:\s([\s\S]+?)^(?=[A-D]\))/m;

        const answerChoicesRegex = /([A-D])\)\s(.+)/g

        const correctAnswerRegex = /Answer:\s([A-D])/
        const explanationRegex = /Explanation:\s\{(.*?)\}/s

        const matchQuestion = ai_response.match(questionRegex)
        const matchAnswerChoices = [...ai_response.matchAll(answerChoicesRegex)]
        const matchCorrectAnswer = ai_response.match(correctAnswerRegex)
        const matchExplanation = ai_response.match(explanationRegex)

        if (matchQuestion) {
          question = matchQuestion[1].trim();
        }

        if (matchAnswerChoices) {
          answerChoices = matchAnswerChoices.map(match => match[0].trim())
```

```
}

if (matchCorrectAnswer) {
  correctAnswer = matchCorrectAnswer[1]
}

if (matchExplanation) {
  explanation = matchExplanation[1].trim()
}

console.log("Question: ", question)
console.log("Answer Choices:", answerChoices)
console.log("Correct Answer:", correctAnswer)
console.log("Explanation:", explanation)
console.log("Delta: ", response.data.delta)

if (selectedSubject == "Math") {
  try {
    const qregex = /Solve for \$\$(\w+)\$\$\s*\$\$([^\n]+)\$\$/
    const match = question.match(qregex)
    const varToSolveFor = match[1]
    const expression = match[2]
    console.log("Var: ", varToSolveFor)
    console.log("Expression: ", expression)



    const acregex = /([A-D])\)\s*\${1,2}(\w+)\s*=\s*([-\w\\{}\/^+\d]+)\${1,2}/
    const answers = []

```

```
for (let i = 0; i < answerChoices.length; i++) {
    const match = answerChoices[i].match(acregex)
    if (match) {
        const letter = match[1]
        let value = match[3]

        // Optional LaTeX cleanup
        value = value.replace(/\\frac{(\d+)}{(\d+)}/, "$1/$2")
        answers.push({ letter, value })
    }
}

console.log("Answers: ", answers)
const jsExpression = latex2js(expression).toString()
console.log("JS Expr: ", jsExpression)

let solution
let jssolution
let solutionFound = true
try {
    jssolution = nerdamer.solveEquations(jsExpression,
varToSolveFor).toString()
    console.log("JS Solution: ", jssolution)

    solution = convertToLatex(jssolution)
    console.log("Solution: ", solution)
} catch {
    solution = "No solution"
    solutionFound = false
}
let validAnswerFound = false
```

```
        for (let i = 0; i < 4; i++) {
          // If the valid answer is found
          if (answers[i].value === solution) {
            validAnswerFound = true

            // If the correctAnswer matches, change nothing
            if (answers[i].letter === correctAnswer) {
              console.log('Correct answer alr set, no change needed')
              break
            }

            // If correctAnswer doesnt match, swap values
            if (answers[i].letter !== correctAnswer) {
              console.log('Correct answer found but in the wrong spot')
              correctAnswer = answers[i].letter

            }
          }
        }

        if (!validAnswerFound) {
          // Find the answer corresponding to the correctAnswer letter and update its value
          console.log('Correct answer not found, replacing correct answers value')
          for (let i = 0; i < answers.length; i++) {
            if (answers[i].letter === correctAnswer) {
              answers[i].value = solution  // Update the value to the solution

              if (solutionFound) {
                const beforelatex = `${correctAnswer}) $$${varToSolveFor} = ${solution}$$`
```

```
                    answerChoices[i] = convertToLatex(beforelatex)
                } else {
                    answerChoices[i] = `${correctAnswer}) $$${varToSolveFor} =$$ No Solution`
                }


            }
            else {
                if (jssolution.includes('/') && !answers[i].value.includes('/')) {
                    const [, denominator] = jssolution.match(/\/\s*(-?\d+)/) || []
                    if (denominator) {
                        const fakeFraction = `${answers[i].value}/${denominator}`
                        const beforeLatex = `${answers[i].letter}) $$${varToSolveFor} = ${fakeFraction}$$`
                        answerChoices[i] = convertToLatex(beforeLatex)
                    }
                }
            }
        }
    }
    const lines = question.split('\n')
    setLines(lines)
} catch (error) {
    //GEN NEW QUESTION
    console.error("Catastrophic error, genning new question with newchat true", error)
    handleNewChatState(true)
    handleGenerateQuestion(selectedSubject)
}
console.log(correctAnswer)
```

```
            }

            handleNewChatState(false)

            setDelta(response.data.delta)
            setQuestion(question)
            setAnswerChoices(answerChoices)
            setCorrectAnswer(correctAnswer)
            setExplanation(explanation)
            setSubjectsRight(response.data.subjects_right)
            setSubjectsWrong(response.data.subjects_wrong)
            setLoading(false)
        }
        catch (error) {
            console.error("Error generating question:", error)
        }


    }

    const handleSaveAsFlashcard = async (question, answerChoices, correctAnswer,
selectedSubject) => {

        console.log("Correct Answer Letter:", correctAnswer);
        console.log("Index from answerToNumKey:", answerToNumKey[correctAnswer]);
        console.log("Answer Choices:", answerChoices);
        let answer = answerChoices[answerToNumKey[correctAnswer]]

        let choices = answerChoices.slice(0, 4)

        try {
```

```
      const flashcardsData = sessionStorage.getItem("flashcards")
      let flashcards = flashcardsData ? JSON.parse(flashcardsData) : {}

      if (!flashcards[selectedSubject]) {
        flashcards[selectedSubject] = []
      }

      const exists = flashcards[selectedSubject].some(flashcard => flashcard.question === question)
      if (exists) {
        addToast("Flashcard already exists!")
        return
      }

      const newFlashcard = { question, choices, answer }

      flashcards[selectedSubject].push(newFlashcard)

      sessionStorage.setItem("flashcards", JSON.stringify(flashcards))

      console.log("Flashcard added")
      console.log(newFlashcard)
      addToast("Flashcard added")
    } catch (error) {
      console.error("ERROR adding flashcard: ", error)
      addToast("Failed to add flashcard.")
    }
  }
  return (
    <>
      {/* Full Page Layout */}
    <div className="grid grid-rows-[auto_1fr] min-h-screen">
      <TopBar title="Study" />
      <div className="flex flex-col p-4">

        {/* Header */}
        <div className='flex justify-between border-3 p-4 border-pwred rounded-xl items-center mb-8 gap-5'>
          {subjects.length === 0 &&
            <div className="w-1/3 h-full flex justify-center items-center
                  text-shadow text-3xl font-bold text-[#F3F4F6] text-center gap-2">
              <Button text="Add a subject." onClick={() => navigate("/subjects")} />
            </div>
          }

          {subjects.length !== 0 &&
            <div className='w-1/3 pl-17'>
              <label for="subjects" className="block mb-2 text-xl font-semibold text-gray-900"></label>
              <select id="subjects"
                name="subjects"
                className='bg-gray-50 border border-gray-300 text-gray-900 text-lg rounded-lg focus:ring-pwblue focus:border-blue-500 block w-full p-2.5'
                onChange={handleSubjectChange}>
                <option value="" hidden disabled selected>
                  {subjects.length === 0 ? 'Please add a subject in "Subjects".' : "Subjects"}
                </option>
                {subjects.map((subject, index) => (
                  <option key={index} value={subject.subjectName}>
                    {subject.subjectName}
                  </option>
                ))}
              </select>
            </div>

          }
          <div className="w-1/3 h-full flex justify-center items-center
                  text-shadow text-3xl font-bold text-[#F3F4F6] text-center gap-2">
            <Button
              text='Generate question'
              onClick={() => { handleGenerateQuestion(selectedSubject) }} />
          </div>

          <div className="w-1/3 h-full bg-pwblue rounded-xl shadow-xl flex justify-center items-center
                  text-shadow text-3xl font-bold text-[#F3F4F6] text-center">
            {subjects.length === 0 ? (
              <h1>No subject found</h1>

            ) : !selectedSubject ? (
              <h1>No subject selected</h1>
            ) : (
              <div className='w-full flex'>
                <h1 className="w-1/2">
                  {selectedSubject}
                </h1>
                <h1 className="w-1/2">
                  Elo: {subjects.find(subject => subject.subjectName === selectedSubject)?.subjectElo || ""}
                </h1>
              </div>
            )}
          </div>
        </div>
        {loading &&
          <div className='flex flex-col justify-center items-center pt-8'>
            <h1 className="text-pwblue text-6xl font-semibold py-10">Generating
question</h1>
            <div className="col-3">
              <div className="snippet" data-title="dot-flashing">
                <div className="stage">
                  <div className="dot-flashing"></div>
                </div>
              </div>
            </div>
          </div>
        }

        {/* Question */}
        {question && !answerSelection &&
          <div className="flex justify-center items-center m-auto ">
            <div className="w-full min-w-300 h-full p-5 border-3 border-pwred bg-
pwblue rounded-xl shadow-xl flex items-center justify-center gap-8 ">
              <div className="text-shadow text-4xl font-bold text-[#F3F4F6]  w-1/2
text-center">
                {selectedSubject != 'Math' &&
                  <ReactMarkdown
                    remarkPlugins={[remarkMath]}
                    rehypePlugins={[rehypeKatex]}
                  >
                    {question}
                  </ReactMarkdown>
                }
                {selectedSubject == 'Math' &&
                  <>
                    <ReactMarkdown
                      remarkPlugins={[remarkMath]}
                      rehypePlugins={[rehypeKatex]}
                    >
                      {lines[0]}
                    </ReactMarkdown>
                    <ReactMarkdown
                      remarkPlugins={[remarkMath]}
                      rehypePlugins={[rehypeKatex]}
                    >
                      {lines[1]}
                    </ReactMarkdown>
                  </>
                }
              </div>
              <div className='flex flex-col w-1/2 items-center'>
                <div className="text-shadow text-2xl font-bold text-[#F3F4F6] w-2/3 text-left">
                  <div className="py-1">
                    <ReactMarkdown
                      remarkPlugins={[remarkMath]}
                      rehypePlugins={[rehypeKatex]}
                    >
                      {answerChoices[0]}
                    </ReactMarkdown>
                  </div>
                  <div className="py-1">
                    <ReactMarkdown
                      remarkPlugins={[remarkMath]}
                      rehypePlugins={[rehypeKatex]}
```

```
                    >
                        {answerChoices[1]}
                    </ReactMarkdown>
                </div>
                <div className="py-1">
                    <ReactMarkdown
                        remarkPlugins={[remarkMath]}
                        rehypePlugins={[rehypeKatex]}
                    >
                        {answerChoices[2]}
                    </ReactMarkdown>
                </div>
                <div className="py-1">
                    <ReactMarkdown
                        remarkPlugins={[remarkMath]}
                        rehypePlugins={[rehypeKatex]}
                    >
                        {answerChoices[3]}
                    </ReactMarkdown>
                </div>
            </div>
        </div>
    </div>

</div>}

{/* Answer choices */}
{answerChoices.length !== 0 && !answerSelection &&
    <div className='border-3 border-pwred rounded-xl p-12 grid grid-cols-2 gap-20 w-3/5 m-auto'>
        <div className="flex flex-col gap-10">
```

```
            <Button text="A" onClick={() => setAnswerSelection("A")} />
            <Button text="C" onClick={() => setAnswerSelection("C")} />
        </div>
        <div className="flex flex-col gap-10">
            <Button text="B" onClick={() => setAnswerSelection("B")} />
            <Button text="D" onClick={() => setAnswerSelection("D")} />
        </div>
    </div>
}

{answerSelection && explanation &&
    <div className="flex flex-col justify-center items-center gap-12 p-12">
        <div className="w-full h-full p-5 border-3 border-pwred bg-pwblue rounded-xl shadow-xl flex flex-col items-center justify-center gap-8 ">
            <h1 className="text-shadow text-4xl font-bold text-[#F3F4F6] text-center">{answerSelection == correctAnswer ? 'Correct!' : 'Incorrect.'}</h1>
            {selectedSubject != "Math" &&
                <h1 className="text-shadow text-2xl font-bold text-[#F3F4F6] text-center">{explanation}</h1>
            }
            {console.log(selectedSubject)}
        </div>
        <div className='w-1/3 h-20'>
            <Button text="Save question as Flashcard"
                onClick={() => handleSaveAsFlashcard(question, answerChoices, correctAnswer, selectedSubject)} />
        </div>
    </div>

}


```

```

        </div>
      </div>
    </>
  );
}

export default Study;

```

File: AiStudyCompanion\ASC\src\pages\Subjects.jsx

```
import { useState } from 'react'
import '../styles.css'
import Button from '../components/Button.jsx'
import TopBar from '../components/TopBar.jsx'
import SubjectList from '../components/SubjectList.jsx'
import { API_ADD_SUBJECT } from '../config/api';
import axios from 'axios';
import { useToast } from '../contexts/ToastContext'
```

```

function Subjects() {
    const [selectedSubjects, setSelectedSubjects] = useState(new Set())
    const { addToast } = useToast()

    const handleSubjectSelection = (subjectName, isChecked) => {
        setSelectedSubjects(prev => {
            const newSelection = new Set(prev);
            isChecked ? newSelection.add(subjectName) : newSelection.delete(subjectName);
            return newSelection;
        });
    }

    const [forceUpdate, setForceUpdate] = useState(false)
    const handleRemoveSubjects = () => {
        let subjects = JSON.parse(sessionStorage.getItem("importedSubjects")) || [];
        subjects = subjects.filter(sub => !selectedSubjects.has(sub.subjectName));

        sessionStorage.setItem("importedSubjects", JSON.stringify(subjects));
        setSelectedSubjects(new Set()); // Clear selection after removal

        setForceUpdate(prev => !prev)
        window.dispatchEvent(new Event("subjectsUpdated"));
    };


    const handleAddSubject = async (newSubject) => {
        if (!selectedSubject) {
            console.error("No subject selected!")
            addToast("No subject selected!")
            return
        }

        const currentSubjects = JSON.parse(sessionStorage.getItem("importedSubjects")) || []
        console.log("Current subjects from sessionStorage:", currentSubjects)
        console.log("New Subject:", newSubject)

        try {
            const payload = {
                subjects: currentSubjects,
                newSubject: newSubject
            }
            const response = await axios.post(API_ADD_SUBJECT, payload, {
                headers: { "Content-Type": "application/json" }
            })

            sessionStorage.setItem("importedSubjects", JSON.stringify(response.data.subjects))
            console.log("Subjects updated:", response.data.subjects)

            window.dispatchEvent(new Event("subjectsUpdated"))
        } catch (error) {
            console.error("Error adding subject:", error)
        }
    }


    const [selectedSubject, setSelectedSubject] = useState("")
    const handleSubjectChange = (event) => {
        setSelectedSubject(event.target.value);
    };
```

```
  return (
    <>
      {/* Full Page Layout */}
      <div className="grid grid-rows-[auto_1fr] min-h-screen">
        <TopBar title="Subjects" />
        <div>
          {/* Subjects */}
          <div className="p-10 max-w-6xl mx-auto">
            <SubjectList onSubjectSelect={handleSubjectSelection} key={forceUpdate} />
            {/* REMOVE BUTTON */}
            <div className='h-12 w-1/5 ml-auto '>
              {selectedSubjects.size !== 0 &&
                <Button
                  text="Remove Subject"
                  onClick={handleRemoveSubjects}
                  className="" />
              }
            </div>
            {/* ADD BUTTON & LIST */}
            <div className="w-1/3 mt-10 flex justify-end relative -translate-y-23 translate-x-69 z-0">
              <div className=">

                <form className='max-w-sm mx-auto '>
                  <label for="subjects" className=" text-center block mb-2 text-xl font-semibold text-gray-900">Add a subject:</label>
                  <select id="subjects"
                    name="subjects"
                    className='bg-gray-50 border border-gray-300 text-gray-900 text-lg rounded-lg focus:ring-pwblue focus:border-blue-500 block w-full p-2.5'
                    onChange={handleSubjectChange}>
                    <option value="" hidden disabled selected>Subjects</option>
                    <option value="English">English</option>
                    <option value="History">History</option>
                    <option value="Geography">Geography</option>
                    <option value="Computer Science">Computer Science</option>
                    <option value="Math">Math</option>
                  </select>
                </form>
                <div className='h-1/3 z-10 mt-2'>
                  <Button text="Add Subject"
                    className="mx-auto "
                    onClick={() => handleAddSubject(selectedSubject)} />
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </>
  );
}

export default Subjects;

```

File: AiStudyCompanion\ASC\src\utils\convertLatex.js

```
export function convertToLatex(input) {
    let latex = input;

    latex = latex.replace(/\b(-?\d+)\s*\/\s*(-?\d+)\b/g, '\\frac{$1}{$2}')

    latex = latex
        .replace(/i/g, '\\i')
        .replace(/sqrt/g, '\\sqrt')
        .replace(/(\d)([a-zA-Z])/g, '$1 $2')
        .replace(/(\d)([+\-])/g, '$1 $2')
        .replace(/\*/g, '')

    return latex;
}





```

File: AiStudyCompanion\ASC\src\utils\fileHandlers.js

```
export const handleExport = (addToast) => {
    try {
        const subjectsData = sessionStorage.getItem("importedSubjects");
        const flashcardsData = sessionStorage.getItem("flashcards");

        if (!subjectsData || subjectsData === "null") {
            addToast("No subjects to export!");
            return;
        }

        // Ensure flashcardsData is a valid object
        const flashcards = flashcardsData ? JSON.parse(flashcardsData) : {};

        const exportData = {
            importedSubjects: JSON.parse(subjectsData),
            flashcards: flashcards
        };

        const jsonString = JSON.stringify(exportData, null, 2);
        const blob = new Blob([jsonString], { type: "application/json" });
        const url = window.URL.createObjectURL(blob);

        const a = document.createElement("a");
        a.href = url;
        a.download = "ascData.json";
        a.style.display = "none";
        document.body.appendChild(a);
        a.click();
        document.body.removeChild(a);

        // Free URL memory
        window.URL.revokeObjectURL(url);
    } catch (error) {
        console.error("ERROR exporting subjects: ", error);
        addToast("Failed to export subjects.");
    }
};

export const handleImport = (addToast) => {
    const input = document.createElement("input");
```

```
input.type = "file";
input.accept = ".json";

input.addEventListener("change", (event) => {
    const file = event.target.files[0];
    if (!file) return;

    const reader = new FileReader();
    reader.onload = (e) => {
        try {
            const jsonData = JSON.parse(e.target.result);

            // Ensure importedSubjects is a valid array
            if (!Array.isArray(jsonData.importedSubjects)) {
                throw new Error("Invalid file format.");
            }

            sessionStorage.setItem("importedSubjects", JSON.stringify(jsonData.importedSubjects));

            // Ensure flashcards is a valid object before setting it
            sessionStorage.setItem("flashcards", JSON.stringify(jsonData.flashcards || {}));

            window.dispatchEvent(new Event("subjectsUpdated"));
            addToast("Subjects and flashcards successfully imported!");
        } catch (error) {
            console.error("ERROR importing data: ", error);
            addToast("Failed to import subjects.");
        }
    };
    reader.readAsText(file);
```

```

        input.remove();
    });

    input.click();
};


```

File: AiStudyCompanion\ASC\src\utils\pdfHandlers.js

```
import jsPDF from 'jspdf'
export function generatePDF(questions, selectedSubject) {
    const doc = new jsPDF({
        orientation: "landscape",
        unit: "mm",
        format: "a4"
    })

    const pageWidth = doc.internal.pageSize.width
    const pageHeight = doc.internal.pageSize.height
    const margin = 10
    const centerX = pageWidth / 2
    const sectionWidth = (pageWidth / 2) - (margin * 1.5)

    questions.forEach((q, index) => {
        if (index > 0) doc.addPage()

        // Draw center line for folding
        doc.setDrawColor(0)
        doc.setLineWidth(0.5)
        doc.line(centerX, margin, centerX, pageHeight - margin)

        // Draw left and right section borders
        doc.rect(margin, margin, sectionWidth, pageHeight - 2 * margin)
        doc.rect(centerX + margin / 2, margin, sectionWidth, pageHeight - 2 * margin)

        // Left Side: Question and Answer Choices
        doc.setFontSize(14)
        let yPosition = margin + 10

        const wrappedQuestion = doc.splitTextToSize(q.question, sectionWidth - 10)
        doc.text(`Q${index + 1}:`, margin + 5, yPosition)
        yPosition += 6
        doc.text(wrappedQuestion, margin + 5, yPosition)
        yPosition += wrappedQuestion.length * 6 + 4

        doc.setFontSize(12)
        q.choices.forEach((choice) => {
          const wrappedChoice = doc.splitTextToSize(choice, sectionWidth - 10)
          doc.text(wrappedChoice, margin + 5, yPosition)
          yPosition += wrappedChoice.length * 6
```

```
        })

        // Right Side: Correct Answer
        doc.setFontSize(14)
        doc.text("Correct Answer:", centerX + margin + 5, margin + 10)
        doc.setFontSize(16)
        doc.text(q.answer, centerX + margin + 5, margin + 25)
    })

    doc.save(selectedSubject.toLowerCase() + "Flashcards.pdf")
}

```