P05 Design Document

404NotFound

Jayden Zhang, Margie Cao, Danny Huang, Kyle Lee

Target Ship Date: {2025-06-06}

Overview

As successful students and productive members of society, it is important for us to improve ourselves and for us to keep on learning throughout life. It therefore is crucial to be able to access these resources that will allow us to become better selves (I.e., textbooks). However, sometimes it is extremely difficult, time consuming, inefficient and oftentimes costly for us to obtain these keys to success. 404NotFound aims to solve this problem by crawling and scraping through the web for you to compile an all in one, one step textbook archive where you will be able to find all your textbooks in minimal time, maximum efficiency, all in one search! Through our application you will be able to find your textbooks, save them to your account and download them to your local device. All for free !!! (This application is for educational purposes, please don't copyright us)

Components

Front End:

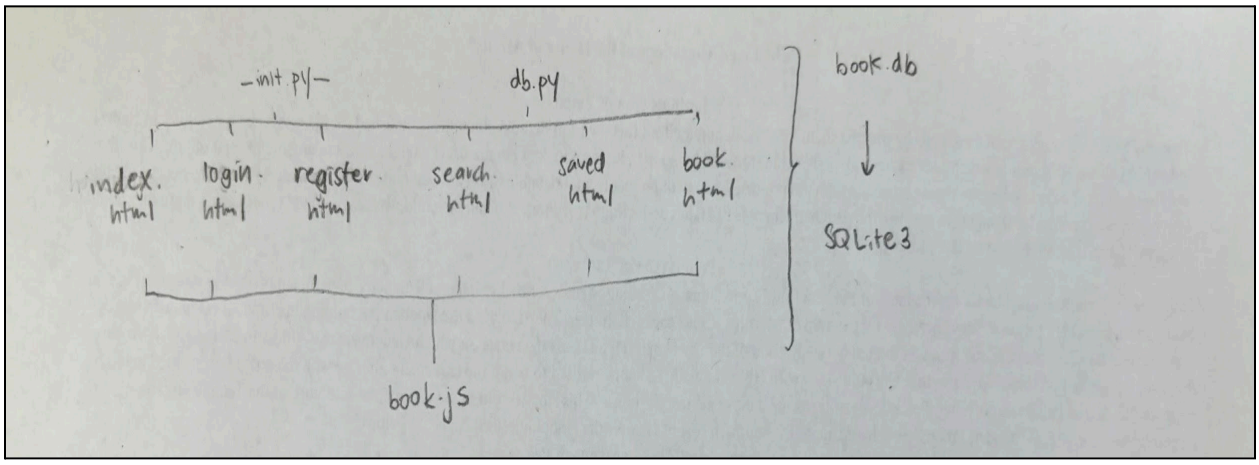
- index.html
    - Landing page, features site introduction and buttons to redirect. Displays a clean UI with buttons to log in, register, or search textbooks.
- login.html
    - Users will be able to log into their existing accounts found within the database. Uses hashed passwords.
- register.html
    - Users will be able to register a new account. Necessary to use the website, otherwise access is restricted.
- search.html
    - Users will be able to search for different textbooks. This will prompt the middleware to search and scrape for that textbook.
- saved.html
    - Users will be able to get a list of different textbooks that they have saved to easily access them for future reference.
- book.html

- Users will be able to view specific textbooks and ask questions within the pdf. For instance, question 7 on page 475 can be retrieved. This will also display an AI generated answer.

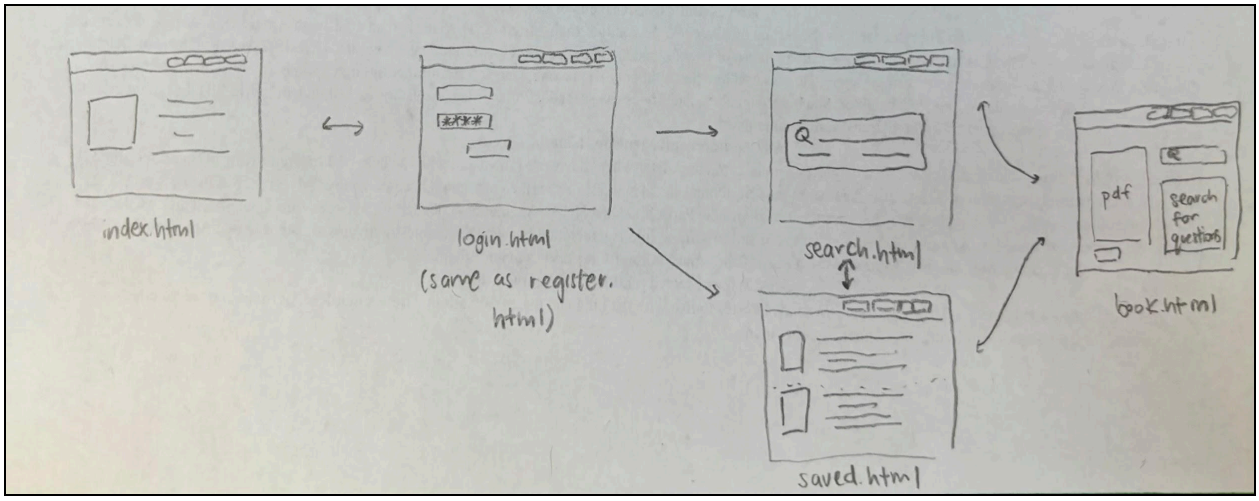
Back End:

- book.db:
  - Database that will store user accounts, favorited content, questions, answers, etc. Handles all logic in SQLite3.
- book.js:
  - Facilitates javascript functionalities that will be needed for the APIs. JS is also faster than python, meaning it would be better for our purposes.
- __init__.py
  - Renders website through flask and manages the entire directory of the website. Adds a functionality for camera access / pdf upload / question upload.
- db.py
  - Database methods that are separated from **init**.py to maintain a clean working environment.

Component Map



Site Map



Database - SQLite3

users:

username	password	school	admin
TEXT	TEXT	TEXT	Boolean

{username}_saved

Textbook	ISBN	QUESTIONS	ANSWERS
TEXT	TEXT	TEXT [returns a file directory]	TEXT [returns a file directory]

archive

Textbook	content
TEXT [returns a file directory holding a pdf]	TEXT [returns a file directory]

APIs

DeepAI API:

We will be using this API to generate solutions for problems that users may ask related to the content of the textbook. Once the user asks a question, we will use this API to generate an AI solution to their question.

Frontend Framework - Tailwind

We believe that Tailwind will be the best option for us to help achieve the site that we have envisioned because of its unparalleled capabilities in customization.

- Utilize tailwind for buttons in user interface, page sections, and prebuilt page designs.
- This will ensure consistent design throughout the entire website and create an appealing user experience.

Libraries/Modules/Packages

- Selenium (in KB already)
    - Used to help scrape different websites in order to find links for textbooks

---

Task Assignments

TASK	PMJZ	DMC	DDH	DKL
Flask Setup	X	X		
Database Organization		X		
HTML Templates			X	
Front-end (Tailwind+CSS)			X	X
Front-end (JS)				X
Web Scraping	X			
Final Testing and Bug Fixing	X	X	X	X

Reach Goals: Git-like Engine

We aim to create an engine for our project that stores scraped textbook questions, tracks changes, logs AI-generated answers, and allows the user to revisit any previous content or version. It mimics Git by using content hashes and a commit log, and adds functionalities like the AI API question-answer design that will be implemented.

File Structure

engine.py

- Handles saving, retrieving, and logging content and answers.

manager.py

- Command interface for the engine. Houses all of the functions that the user can perform with the engine.

hash.py

- Houses all helper functions such as the hashing function used for questions and answers.

commits.json

- Handles all commit messages. Provides a list for changes, updates, and audit history.

/storage directory

- Houses the route of all remembered and hashed questions and answers for the users.

*Designed to be made in python and personalized for specific users.