1. Demo working product system

Screenshot Demo:

- **Instructor US#10: Access Previously Generated Questions**

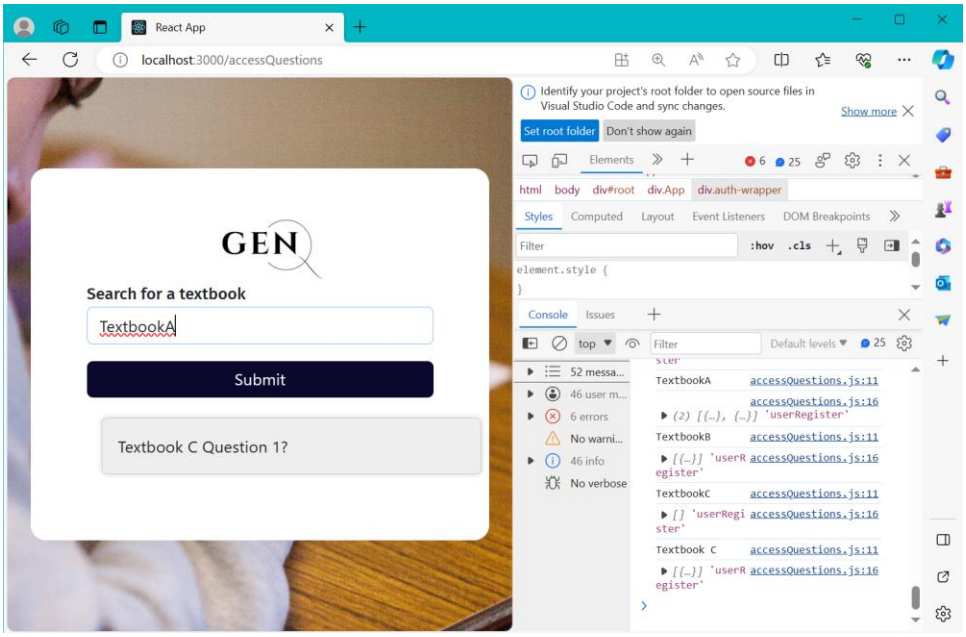


Image 1: Here, we see that we enter TextbookA as the name for the textbook to search in the search bar (disregard the text below the submit button as this is from a previous query).

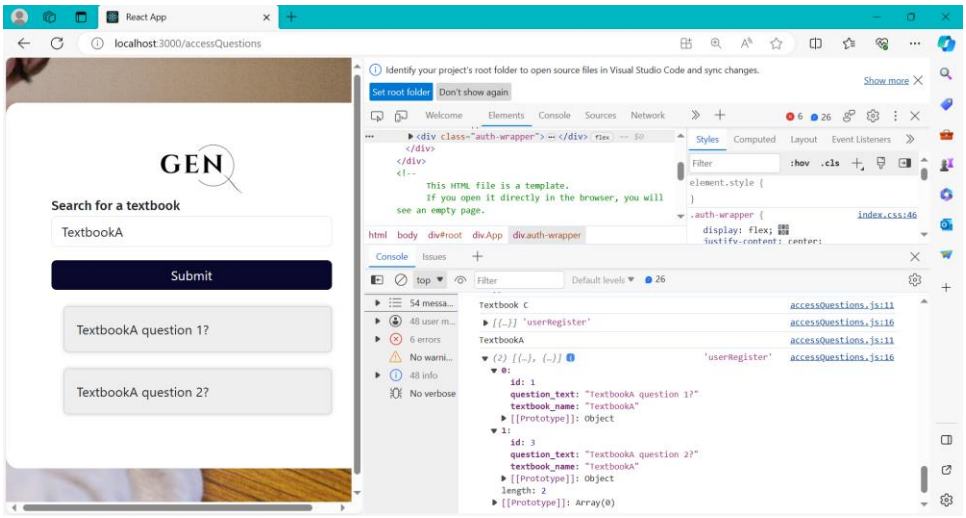


Image 2: When we hit submit, we see on the bottom right in the console that we log the response from the backend server, which is the questions associated with the textbook, and we also see that those questions (only their text) are shown below the submit button.

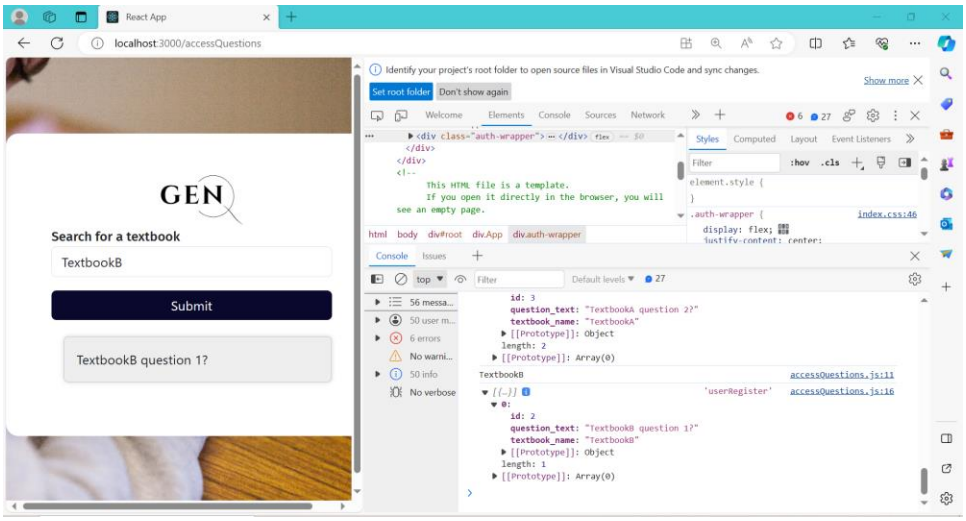


Image 3: Similarly, when we search for “TextbookB”, we get a response from the backend server with the appropriate questions (as shown in the bottom right corner of the console), and those questions are presented below the submit button.

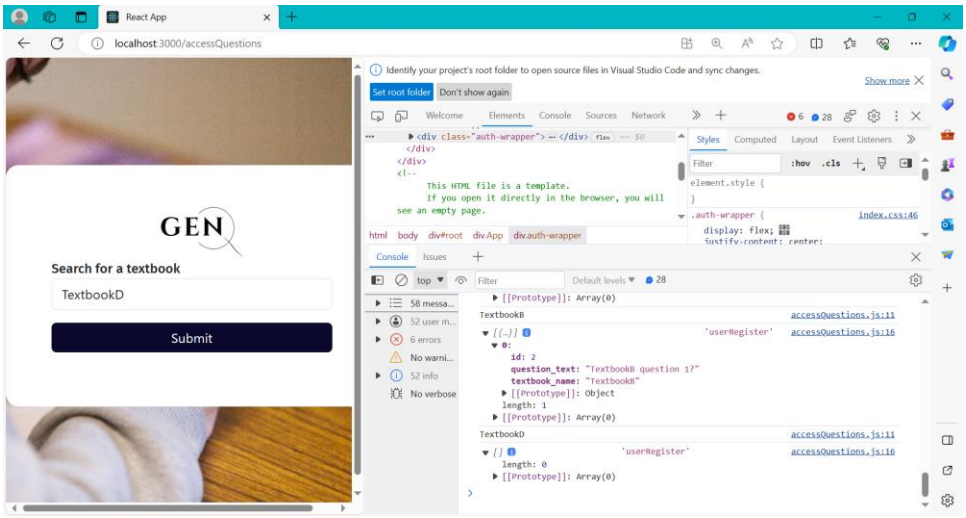


Image 4: Here, we see that when we search “TextbookD”, which is a textbook name not present in the database, we get an empty list of questions back from the backend server (as shown in the bottom right corner of the console). Thus, there are no questions present below the submit button.

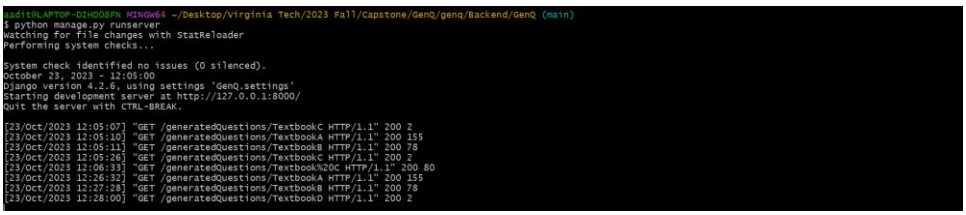


Image 5: Shown above is the log from the backend server that shows it is receiving the requests.

- **Instructor US#3: Create various Quiz Versions**

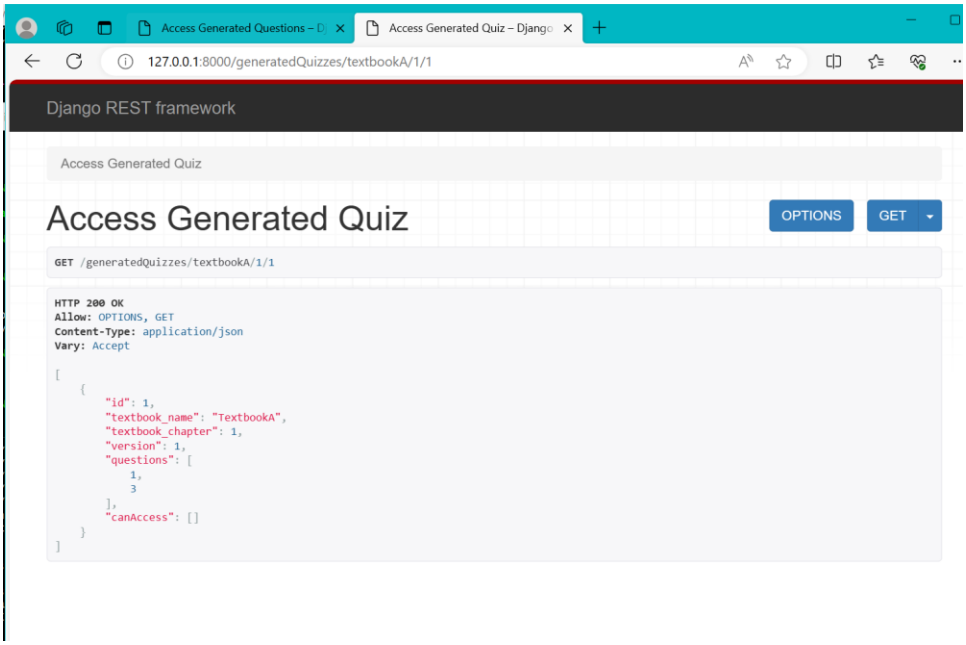


Image 1: The GET API for a quiz, given the textbook name, textbook chapter, and the version. Here we see that we get a response back from the backend with the ID for the questions associated with that quiz.

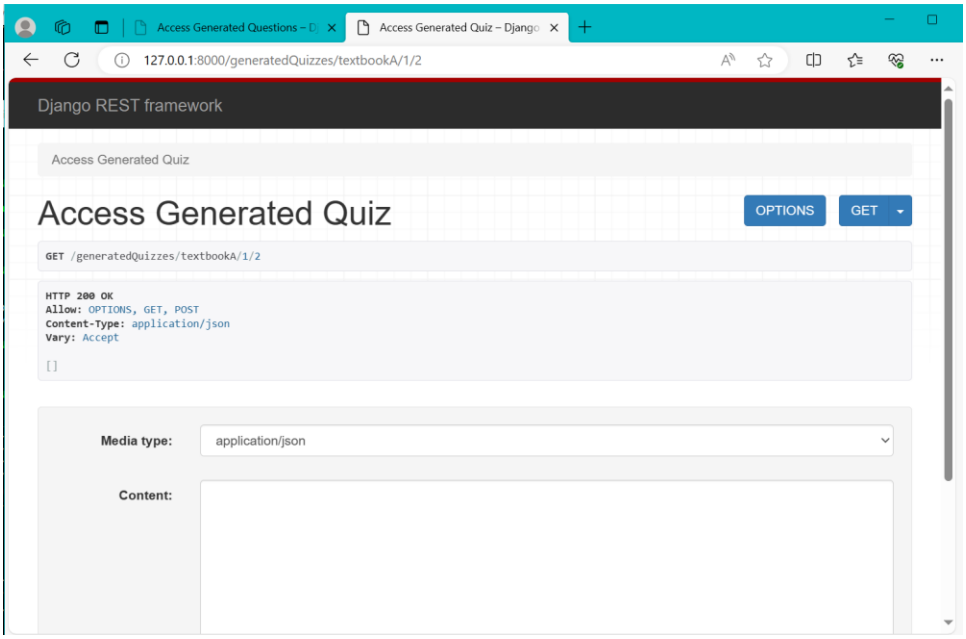


Image 2: Here, we see that when we call a GET request with the same textbook name, chapter, and the 2nd version (which doesn't exist), we get an empty list back from the backend server indicating this quiz does not exist

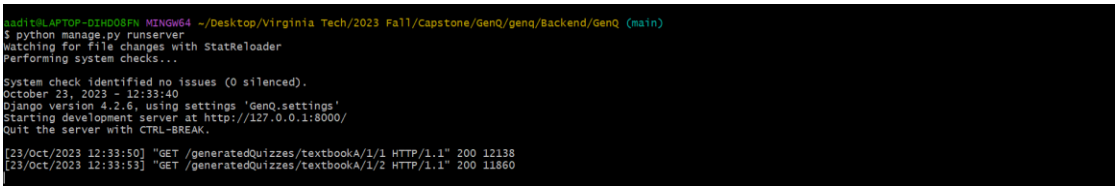


Image 3: Here, we see the log from the backend indicating that it is receiving the query from the Django API.

- **Instructor US#2: Generate Question**

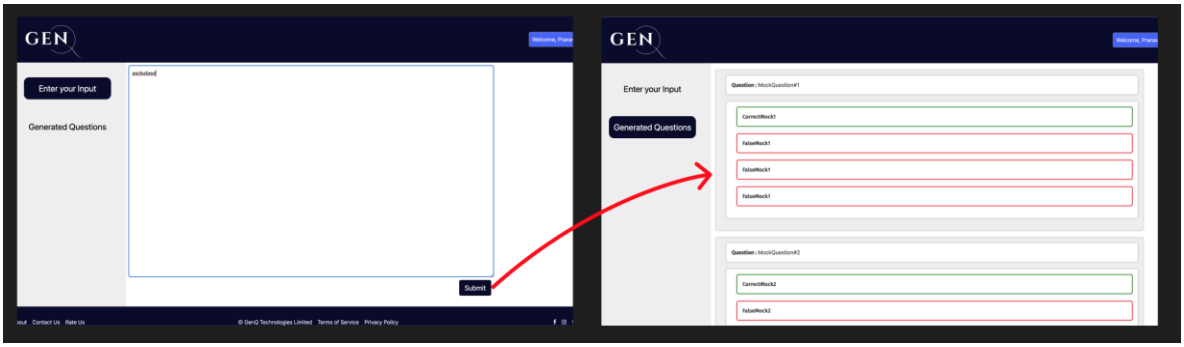


Image 1: After entering the generate questions page, the user would be promoted to enter the textbook information (will be changed in next sprint) after which they would submit it to start the generation process. Then the page would move to the generated questions page, the user (instructor) can read through all the generated questions and options.

Image 2: Once the user has finished looking over the generated questions, they could save the questions.

Instructor US#1: Upload Textbook:

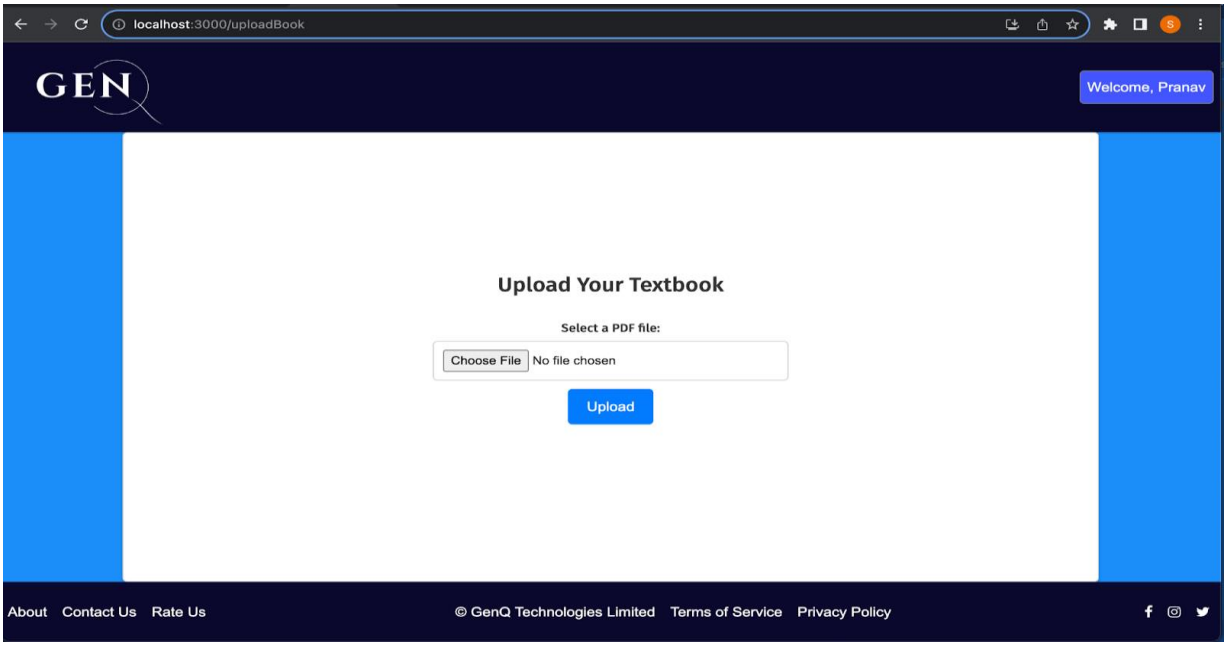


Image 1: Upon going to the uploadBook route, the prompt along with necessary buttons are displayed to upload the textbook. The ability to choose a pdf file of a textbook and then upload said textbook is given in the form of buttons.

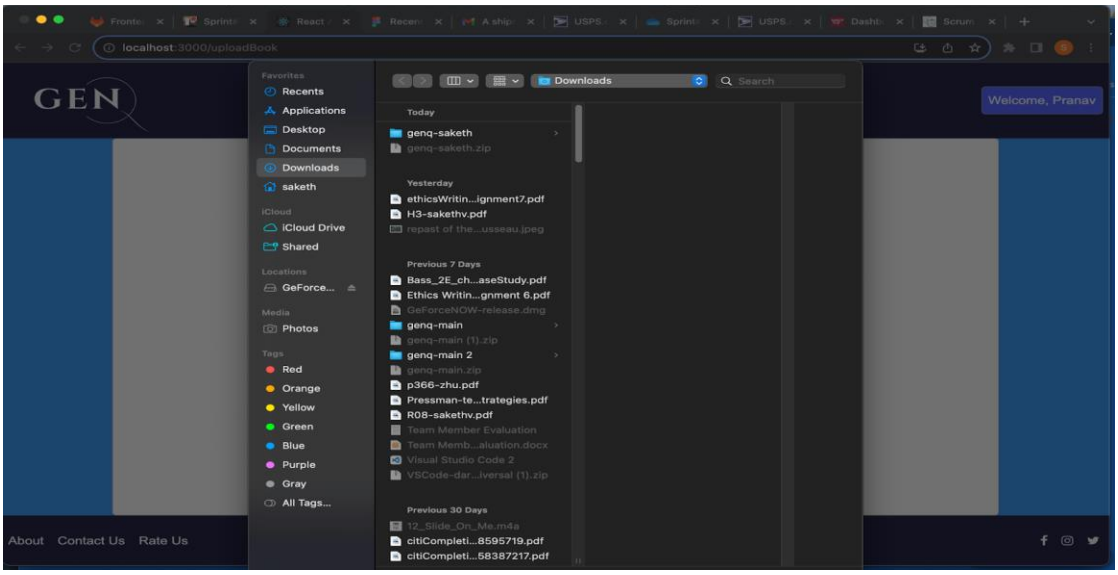


Image 2: This image represents the screen that is displayed when the Upload button is selected. The files that are available are pdf files and any other file type will not be available for the user to pick. A requirement for the system was to allow only pdf files for the textbook. Upon selecting confirm, the pdf will be sent to the server where if processed appropriately, will return a success message to the platform. Otherwise, a failure message will be displayed.

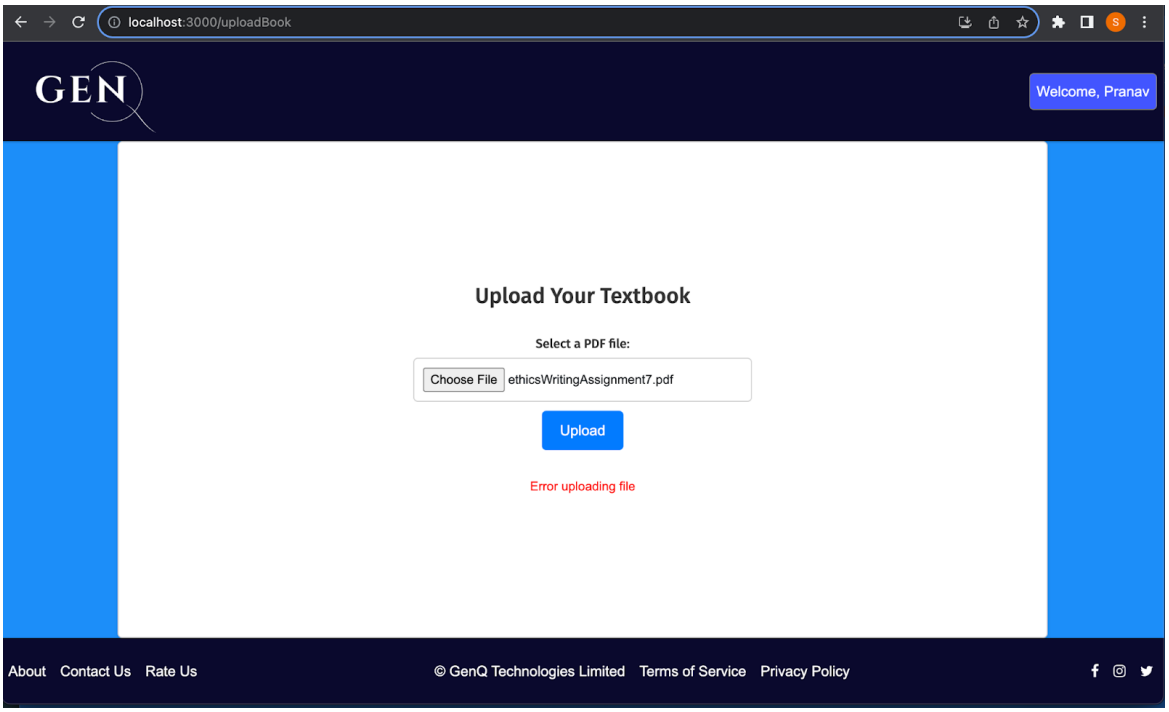


Image 3: Above is an error message being displayed on the platform in response to the server saying that there was a problem in the transmission of the pdf file from the user to the software. This provides a check in the system to ensure that all the data that is stored on the server is appropriate to the software.

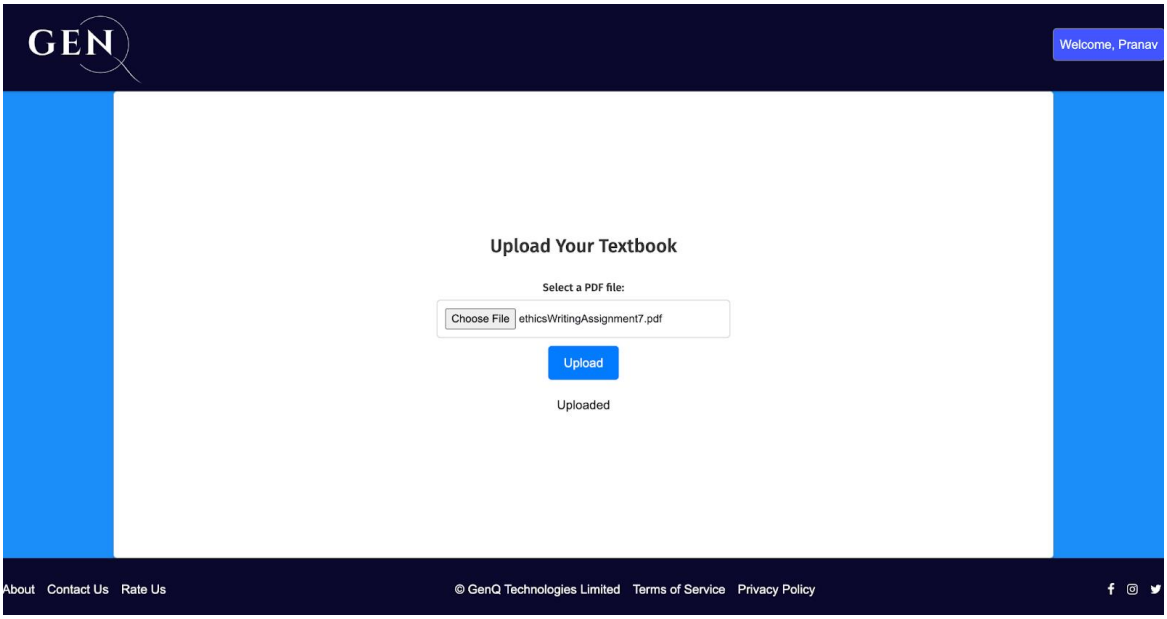


Image 4: Here is an example of a pdf document being successfully uploaded to the platform with a return message from the server stating “Uploaded” as a reassurance to the user. This completes a positive working use case of the upload textbook.

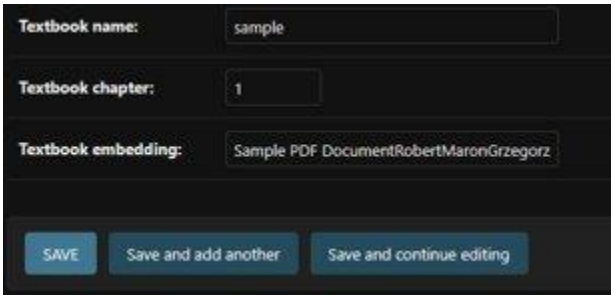
A Python JSON object with the textbook information is submitted through the API to the /upload/ endpoint.

```
{
        data : {
                'name': 'sample'
                'chapter': '1'
                }

        files : {
                'textbook': Python File Object
                }

}
```

When viewing the database, we can see a corresponding textbook object is created. The text from the file object is extracted and stored in a field.



We can see that the text in the database matches the text from the sample PDF shown below. One improvement would be to save the file server-side and store a path to the file in the database to improve performance.

Sample PDF Document

Robert Maron
Grzegorz Grudziński

February 20, 1999

- User Authentication

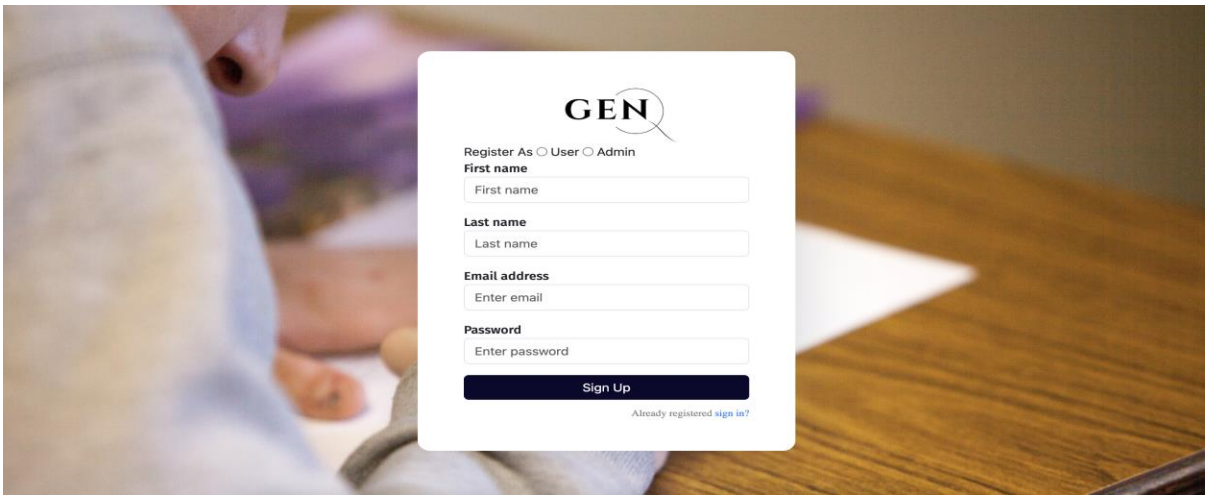


Image 1: This page is our User Registration Page. When the user is registering for the first time on the website, he needs to choose if he is an instructor or a student and enter other relevant information like Name, Email and Password. This data is stored in the database and will be used to authenticate the user during login.

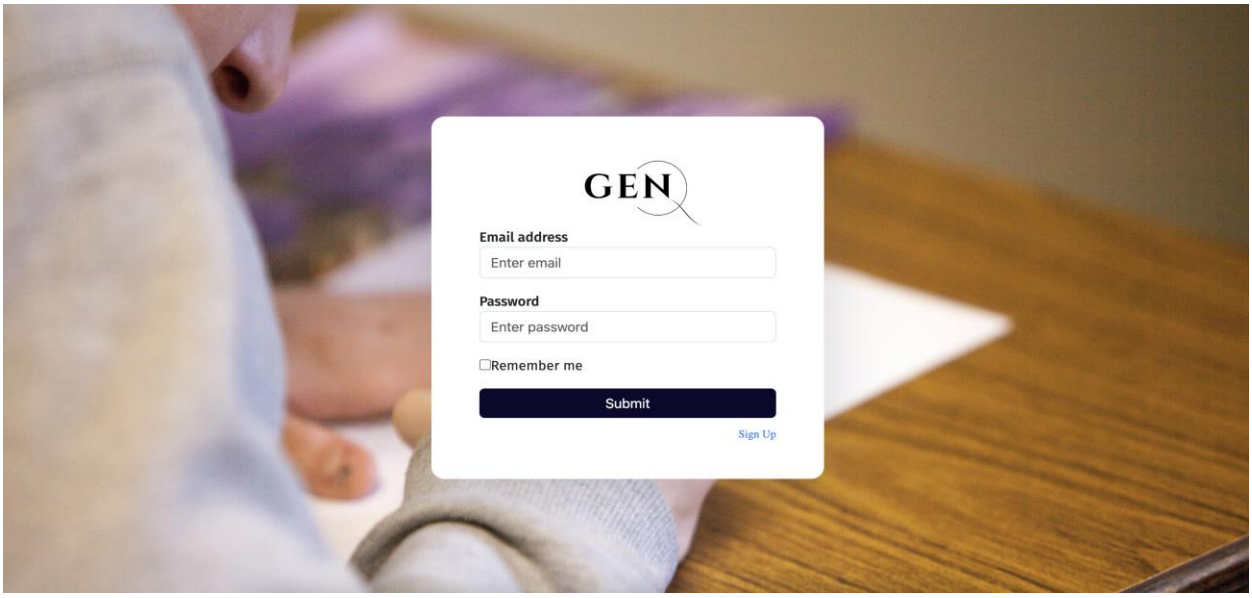


Image 2: This page is our User Sign-up Page. When the user needs to login on the website, he needs to enter data such as email and password. The data stored in the database during registration will be used to authenticate the user during login.

- **LLM API**

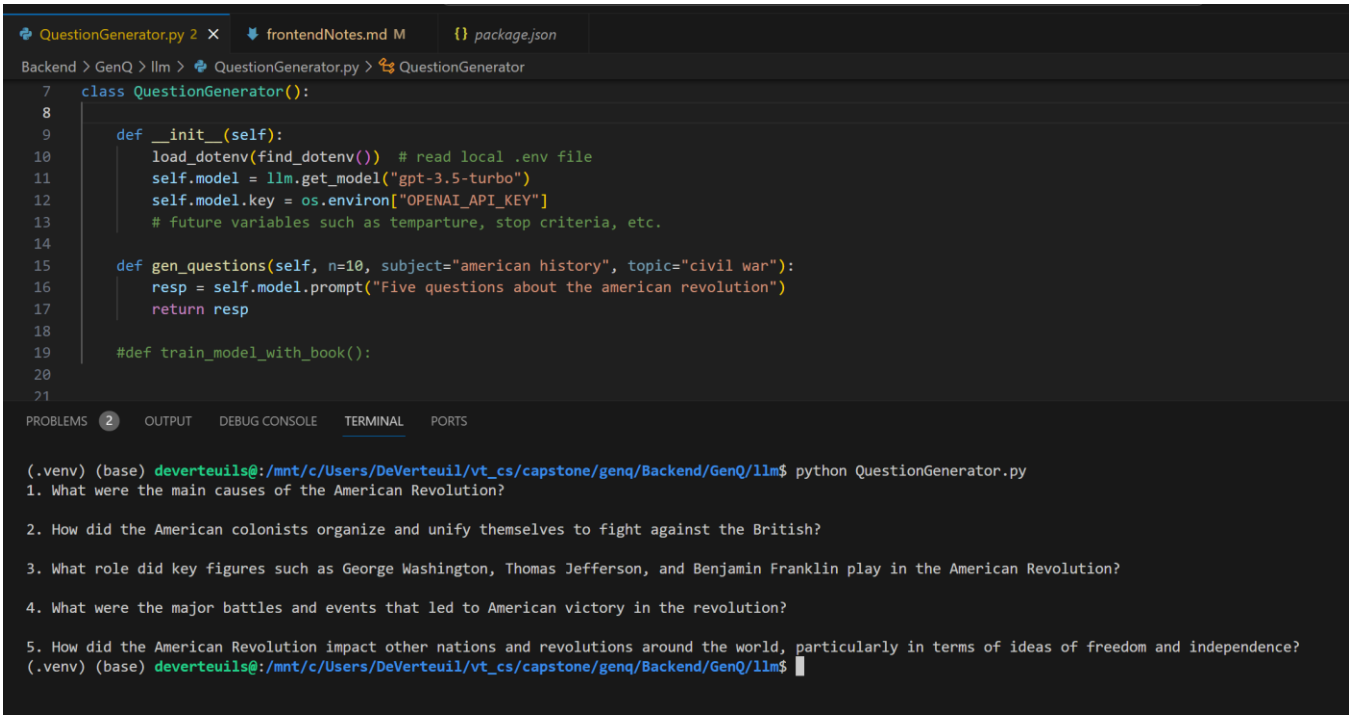
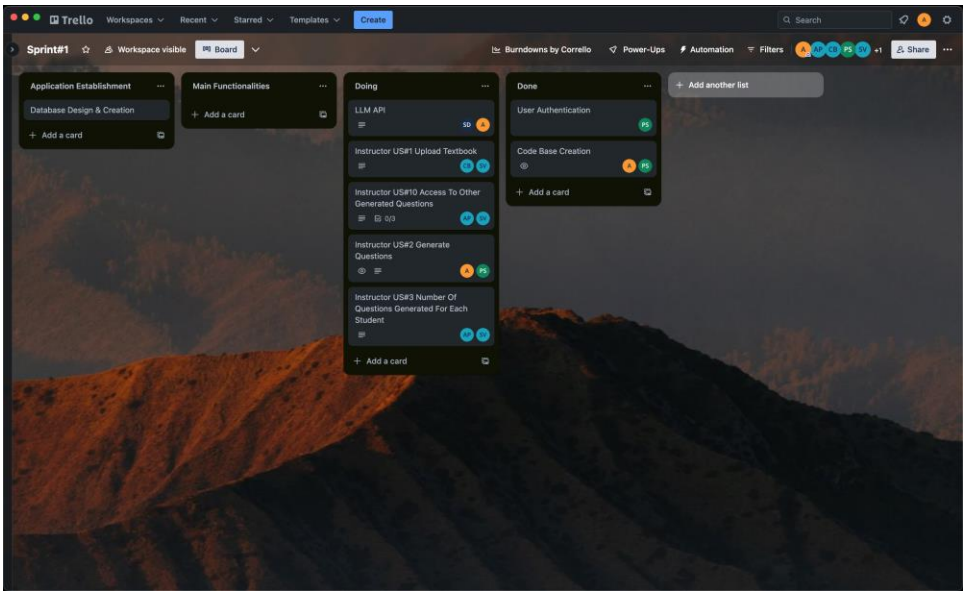


Image 1:

As can be seen in the screenshot above, a mock LLM has been created. At present, the user is not able to access these generated questions. The prompt is hard-coded, and the output is printed to the terminal for the time being. In the next sprint, we plan to integrate the LLM response with our front end. Additionally, we will allow for specific textbook based downstream training of the LLM so that the user can receive robust questions based on their desired input.

2. Collaboration Space

- We are using Trello as a project management tool. Trello has streamlined our workflow by allowing each team member to focus on their respective task while also keeping them aware of other team members' progress thought the sprint. This makes our team more efficient and increased our sprint speed, as team members who have dependencies could start working on their respective tasks while having visibility on the progress of their dependency. Also, with Trello we can easily track and manage our project's tasks, set priorities, assign responsibilities, and monitor progress after each daily sprint and sprint review.



3. Scrum
    - Sprint 1 backlog:
        - Instructor US#2: Generate Questions
            - Tasks
                - Create Associated Backend API endpoints
                    - Hours: 0.5
                - Send a request to LLM API to start generating questions
                    - Hours: 1
                - Create Page to present generated questions
                    - Hours: 2
                - Save generated questions button action and redirection to success page
                    - Hours: 0.5
                - Save generated questions API endpoint
                    - Hours: 0.5
        - Instructor US#1: Upload textbook
            - Tasks

- Create function in views.py that allows for API endpoint access to upload textbook file
  - Hours 1.5
- Add uploaded textbook to database
  - Hours 0.5
- Create test function to verify API endpoints
  - Hours 0.5
- Create page to present upload textbook prompt
  - Hours 1
- Connect frontend to backend
  - Hours 0.5

- Instructor US#10: Access to other generated questions
  - Tasks
    - Create function in views.py that allows for API endpoint access to pull questions given the textbook name
      - Hours: 1
    - Add endpoint to Django URLs file
      - Hours: .25
    - Create Frontend component as the UI
      - Hours: 2
    - Connect Frontend component to backend component and parse results
      - Hours: 1
- Instructor US#3: Number of Generated Questions for each student
  - Tasks
    - Create function in views.py that allows for API endpoint access to pull a specific quiz version for a textbook
      - Hours: .5
    - Create function in views.py that allows for API endpoint access to post a specific quiz version for a textbook given a list of question IDs
      - Hours: 1 (unfinished)
    - Create Frontend component as the UI
      - Hours: 0 -> Pushed to Sprint 2
    - Connect Frontend component to backend component and parse results
      - Hours: 0 -> Pushed to Sprint 2
- User Authentication
  - Tasks
    - Create user interface
      - Hours: 0.5
    - Create backend end points

        - Hours: 1
    - Code Base Creation
        - Tasks
            - Create a react frontend
                - Hours: 0.5
            - Create a backend
                - Hours: 0.5
    - LLM API
        - Tasks
            - Create a temporary Mock API
                - Hours 0.5
            - Research LLM creation processes
                - Hours: 4

- Include your Sprint 1 burndown chart (as described in lecture)

- Sprint retrospective (round-robin type discussion):
    - Your sprint retrospective should talk about what you accomplished during the sprint.
        - Did we implement all the user stories?
            - Implemented US
                - Instructor US#1
                - Instructor US#2
                - Instructor US#10
                - User Authentication
                - Code Base Creation

        - What roadblocks did we encounter? How did we address them?
            - Code Running Issues

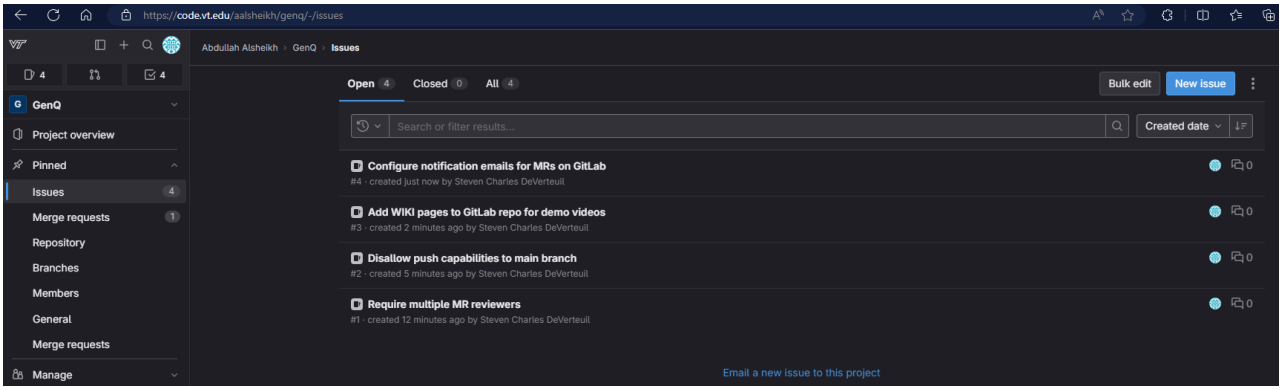
        - Update Start script
        - Added a readme file to explain how to execute code
    - Database Corruption
        - Database migration

- What went well?
    - Well Selected Code Stack because it's easy to use and straightforward in agile developing
    - SQLite is integrated with Django, no need to develop a shared database for now

- What could be improved?
    - Organizing the repository for push requests (Branches for each US)

- Product backlog (updated):
    - Targeted Sprint#2 User Stories:
        - General User Story: Integrating Generating Question with Uploaded textbook (2)
        - General User Story: Integrate LLM API with Model (3)
        - Instructor US#3: Number of Questions for each student (3)
        - Instructor US#5: As an instructor, I want to select and upload sample questions to the student portal so that they can have practice questions. (3)
        - Instructor US#6: As an instructor, I want to edit generated questions so that I can ensure questions are up to the standard that I have. (5)
        - Student US#1: As a student, I want to access practice exams so that I can practice for my class's exams/quizzes. (5)
        - Student US#3: As a student, I want to view my past practice exam scores so that I can gauge my progress. (3)

- Issue Tracking:
    - Issue should be available in our Gitlab repository (LINK)


https://code.vt.edu/aalsheikh/genq/-/issues
Abdullah Alsheikh › GenQ › Issues
4
4
GenQ
Project overview
Pinned
Issues 4
Merge requests 1
Repository
Branches
Members
General
Merge requests
Manage
Open 4
Closed 0
All 4
Bulk edit
New issue
Search or filter results...
Created date
Configure notification emails for MRs on GitLab
#4 · created just now by Steven Charles DeVerteuil
0
Add WIKI pages to GitLab repo for demo videos
#3 · created 2 minutes ago by Steven Charles DeVerteuil
0
Disallow push capabilities to main branch
#2 · created 5 minutes ago by Steven Charles DeVerteuil
0
Require multiple MR reviewers
#1 · created 12 minutes ago by Steven Charles DeVerteuil
0
Email a new issue to this project