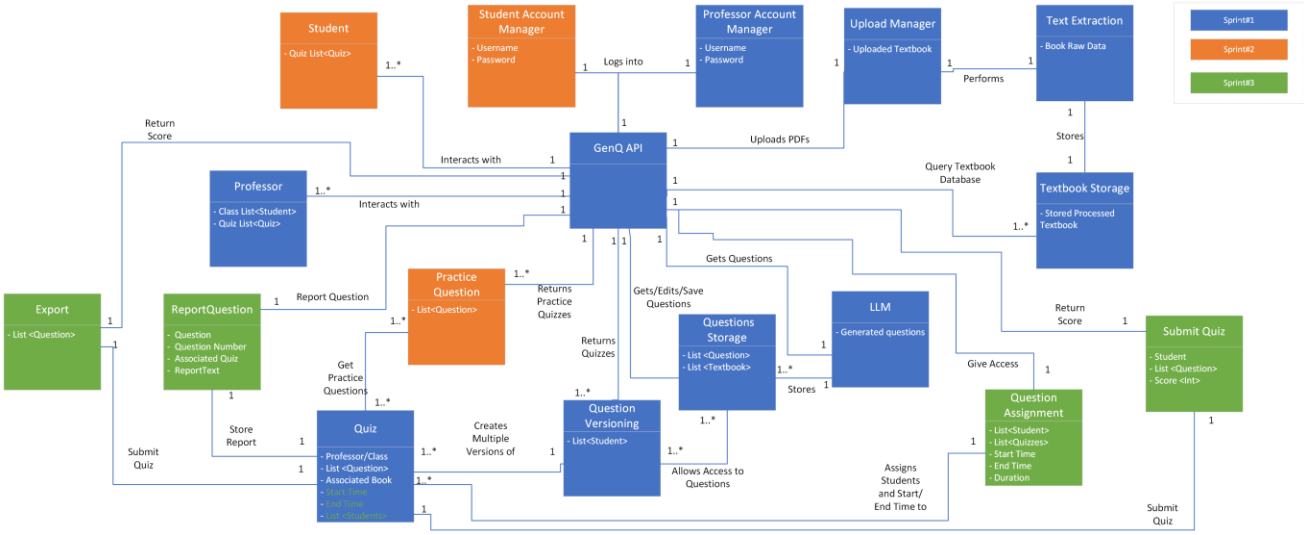
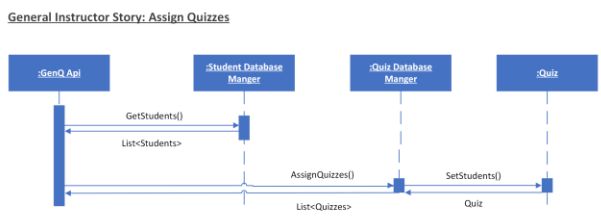
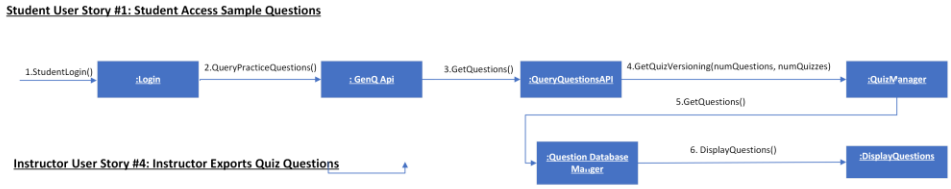
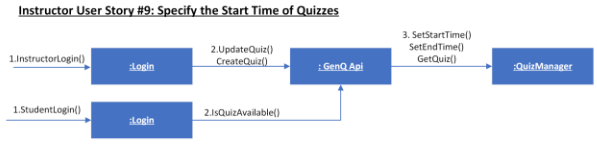
1. Domain Model



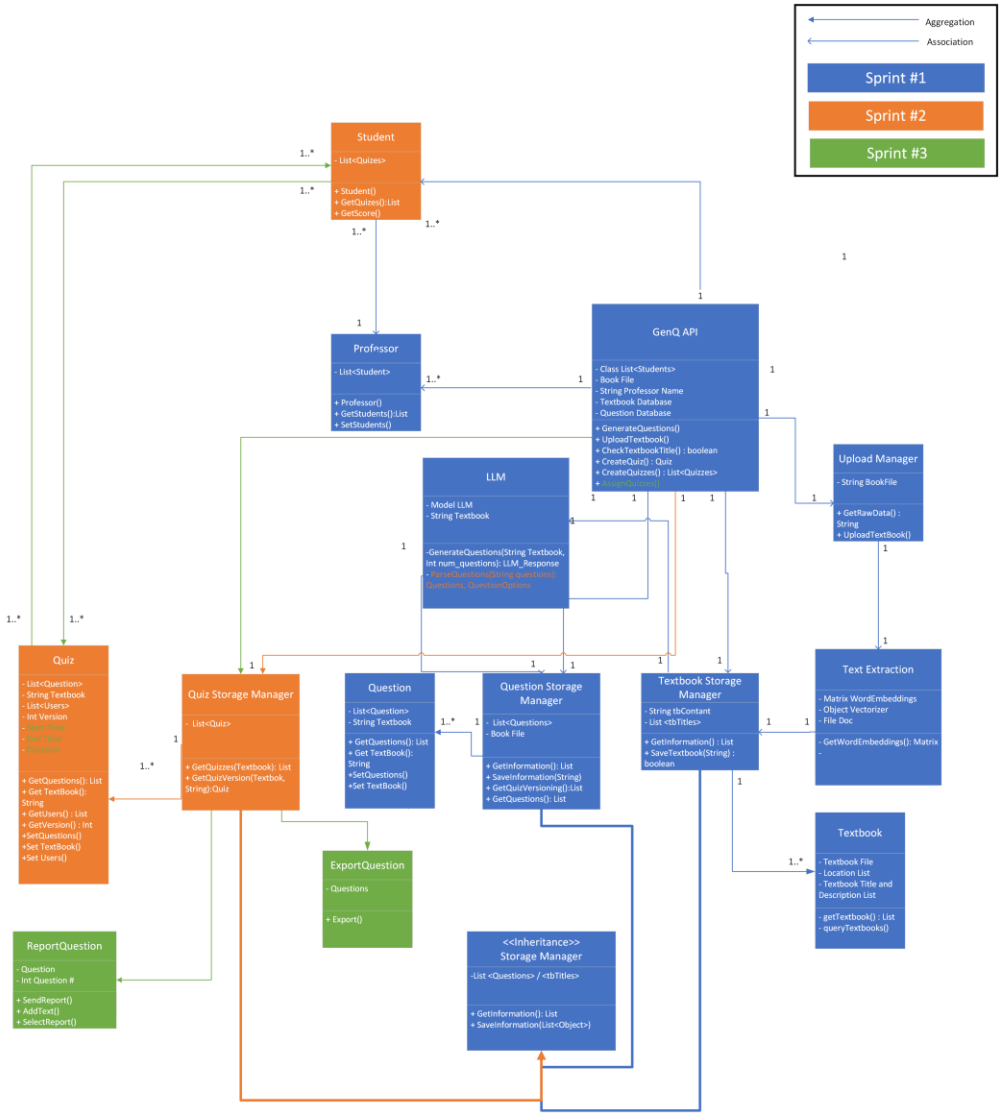
2. Interaction Diagrams





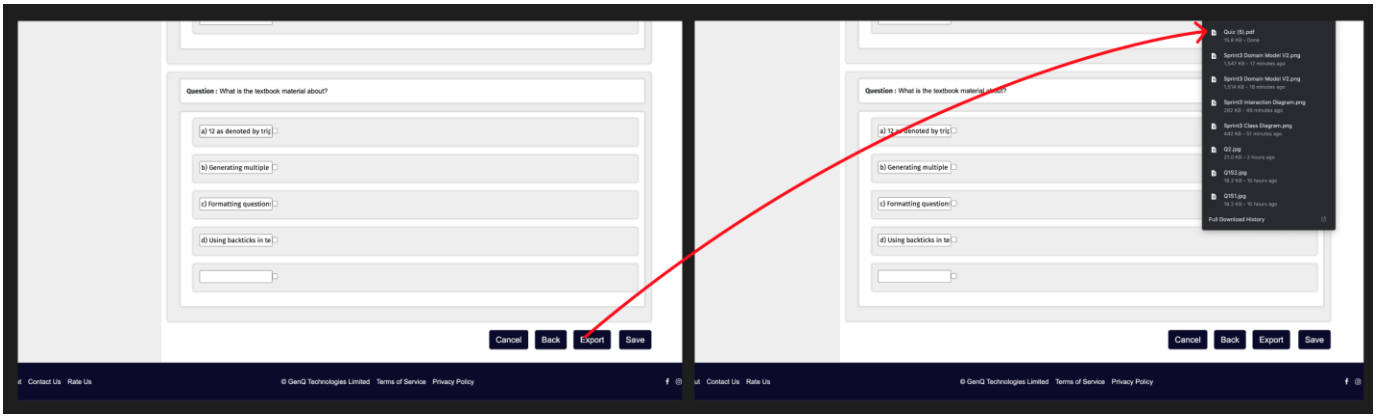


3. Design Class Diagram

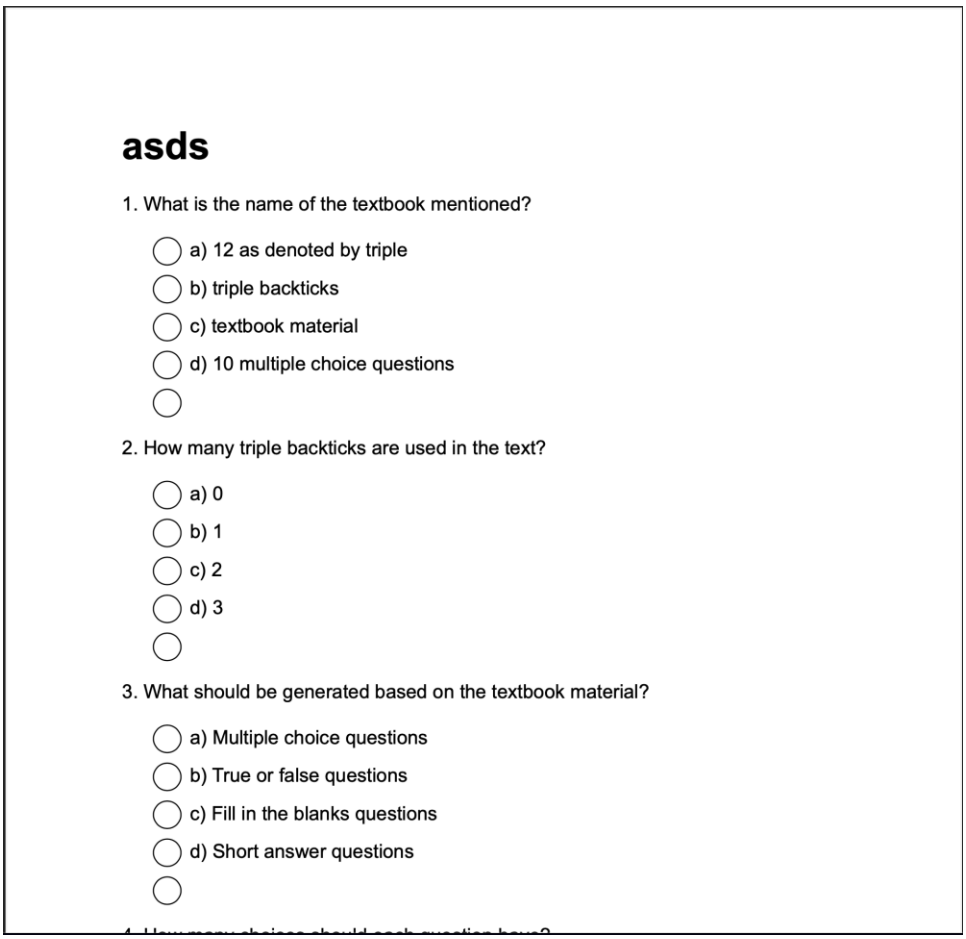


4. <u>Demo Working Product System</u>

   a. **Student US#1: Instructor Exports Quiz Questions**
   After the user generates quiz question, they would have the option to export them into a separate PDF file for physical handouts.



Below is a sample of the generated PDF.



asds

1. What is the name of the textbook mentioned?

- a) 12 as denoted by triple
- b) triple backticks
- c) textbook material
- d) 10 multiple choice questions

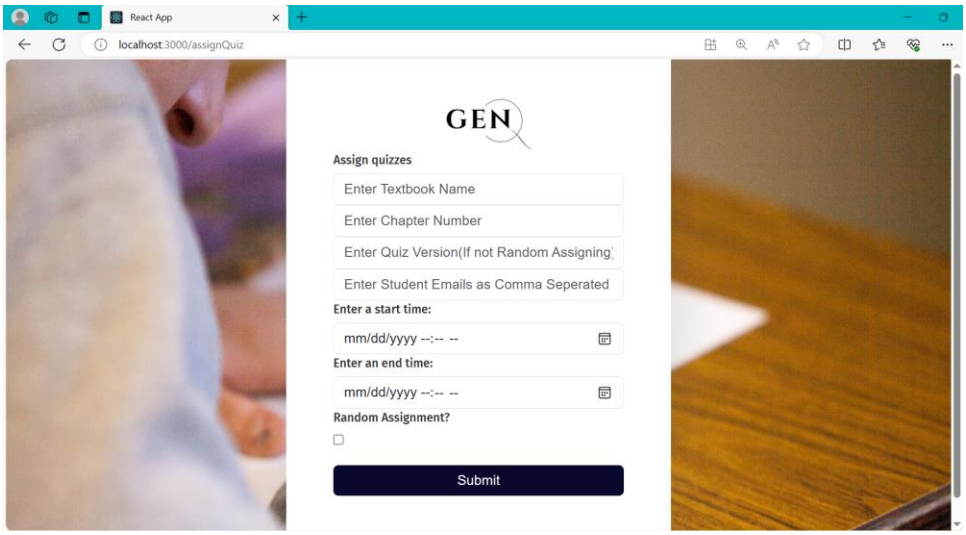
2. How many triple backticks are used in the text?

- a) 0
- b) 1
- c) 2
- d) 3

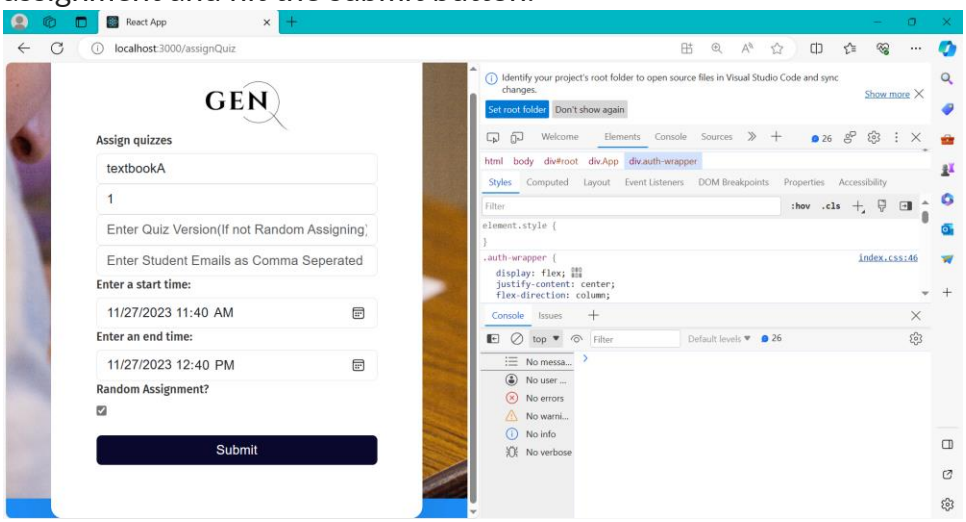
3. What should be generated based on the textbook material?

- a) Multiple choice questions
- b) True or false questions
- c) Fill in the blanks questions
- d) Short answer questions

b. General Instructor User Story: Assigning Quizzes



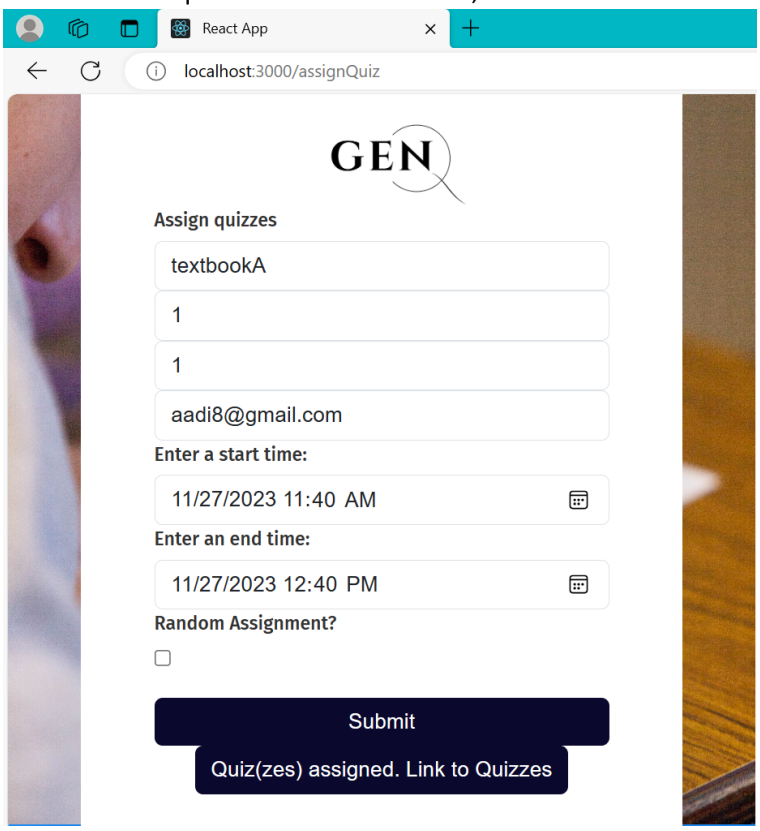
This is the first page that the professor accesses in order to assign quizzes. Here, they should enter the textbook name and chapter by default. Then, if they are randomly assigning quizzes, they just have to specify the start and end time and check the random assignment and hit the submit button.



Here is an example of a professor randomly assigning quizzes for textbookA, chapter 1, starting at 11:40 am and ending at 12:40 am.

```
▼ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
  ▼ 0:
    ▶ canAccess: (10) [1, 11, 21, 31, 41, 51, 61, 71, 81, 91]
      end_time: 202311271240
      id: 1
    ▶ questions: []
      quiz_duration: null
      start_time: 202311271140
      textbook_chapter: 1
      textbook_name: "textbookA"
      version: 1
    ▶ [[Prototype]]: Object
  ▼ 1:
    ▶ canAccess: (10) [2, 12, 22, 32, 42, 52, 62, 72, 82, 92]
      end_time: 202311271240
      id: 2
    ▶ questions: []
      quiz_duration: null
      start_time: 202311271140
      textbook_chapter: 1
      textbook_name: "textbookA"
      version: 2
    ▶ [[Prototype]]: Object
  ▶ 2: {id: 3, textbook_name: 'textbookA', textbook_chapter:
  ▶ 3: {id: 4, textbook_name: 'textbookA', textbook_chapter:
  ▶ 4: {id: 5, textbook_name: 'textbookA', textbook_chapter:
```


Here is a closeup of the value returned from the backend. All the quizzes have students equally assigned to each of the quiz versions. In addition, the start and end time have been specified.



Here, the professor specifically wants to assign the student with email aadi8@gmail.com to quiz version 1.

```
                                                    assignQuiz.js:44
▼ {id: 1, textbook_name: 'textbookA', textbook_chapter: 1, ve
  rsion: 1, start_time: 202311271140, …} i
  ▶ canAccess: (11) [1, 9, 11, 21, 31, 41, 51, 61, 71, 81, 91
    end_time: 202311271240
    id: 1
  ▶ questions: []
    quiz_duration: null
    start_time: 202311271140
    textbook_chapter: 1
    textbook_name: "textbookA"
    version: 1
  ▶ [[Prototype]]: Object
'newCreatedQuiz'
```

The backend API then updates just that quiz and inserts the student as having access to that quiz. Note that because python starts numbering at 0, and Django database starts at 1, the user id associated with aadi8@gmail.com is actually id 9.

---

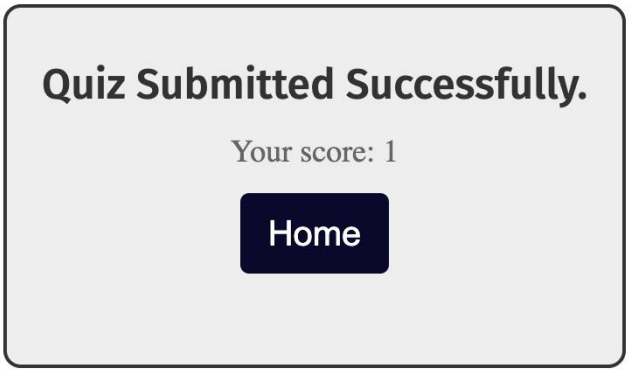
c. Student US#3: Student Submits Quiz

Once the user logs into the system and starts the quiz, user will have an option to select the answers for the quiz by clicking the radio button. Given below are two sample images for test quizzes.

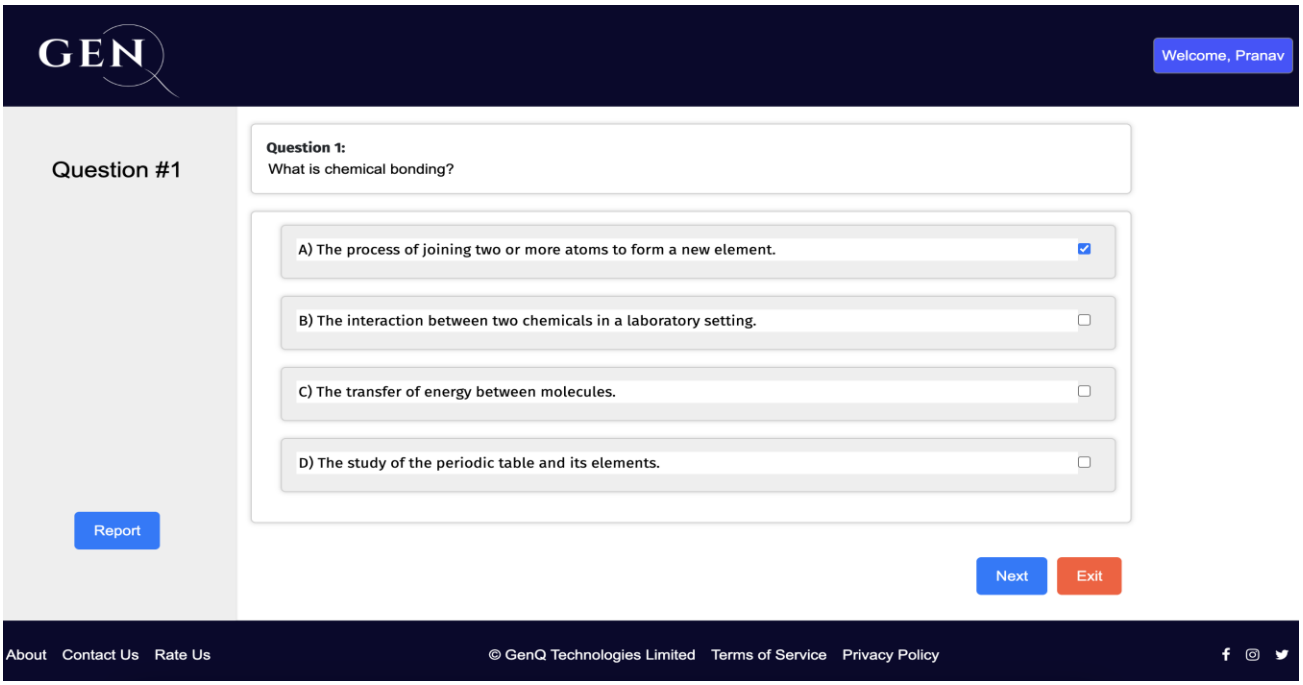
User selected the correct option for question 1

User selected the Incorrect option for Question 2

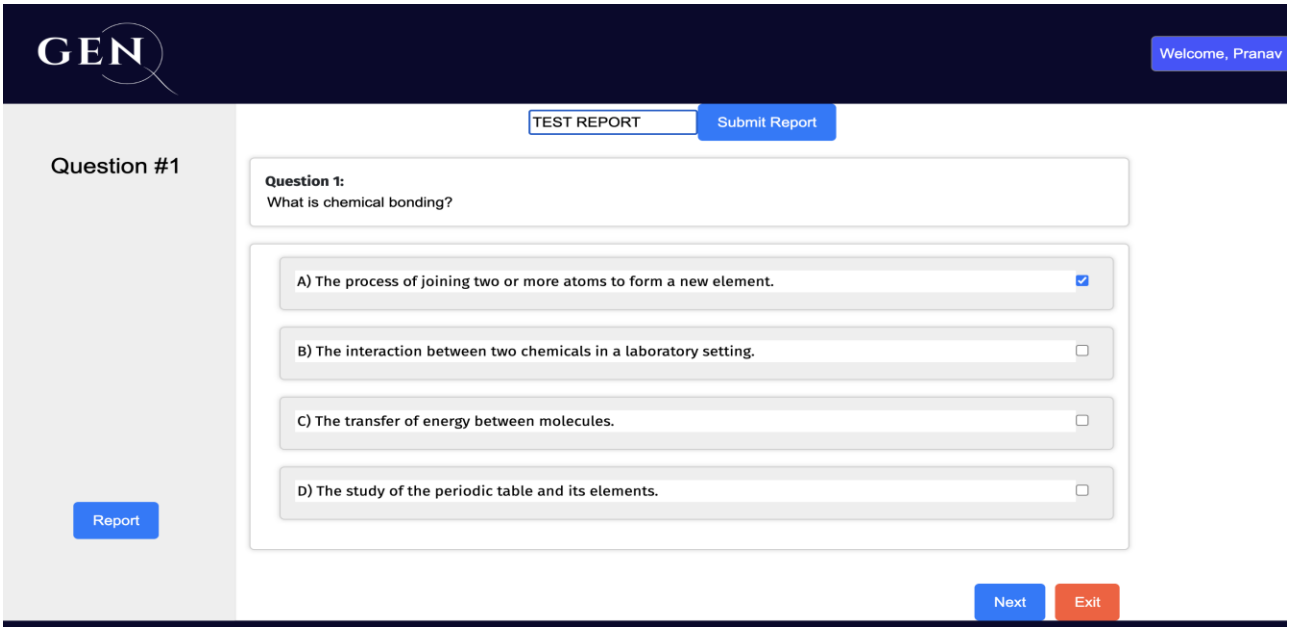
Once the user clicks on submit, he/she will be redirected to a new window in which the score of the quiz will be displayed.



d. Student US#2 : Student reports Quiz Questions



Upon accessing the generated sample quiz, the user will be directed to the sampleAccessQuiz frontend page where they will be presented a question with its respective answer choices. Here, we can see a report button that is available to select if the user wants to report anything about the question.

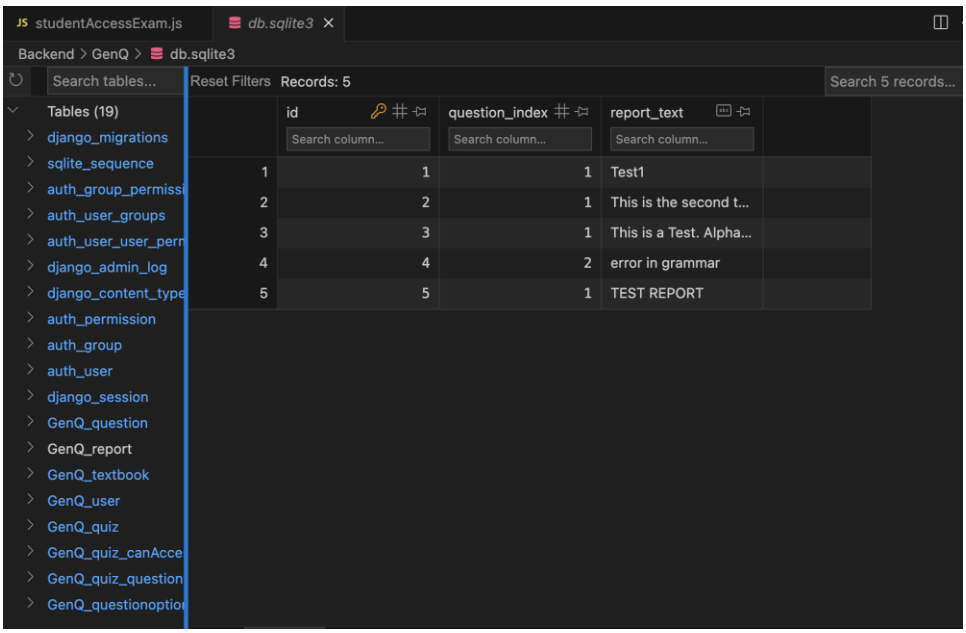


Upon selecting the report button, a text input box is presented to the user. Here, the text input will take the input from the user and translate it as a string datatype for the backend and it will also take the question index number as well. When the user hits the submit button, both of these data values will be sent to the backend.

```
urllib3/issues/3020
  warnings.warn(
System check identified no issues (0 silenced).
November 27, 2023 - 16:48:07
Django version 4.2.6, using settings 'GenQ.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

[27/Nov/2023 16:48:17] "GET /question/ HTTP/1.1" 200 3033
[27/Nov/2023 16:48:17] "GET /question/ HTTP/1.1" 200 3033
[27/Nov/2023 16:48:57] "OPTIONS /saveReport/ HTTP/1.1" 200 0
[27/Nov/2023 16:48:57] "POST /saveReport/ HTTP/1.1" 200 40
```

The image above shows the server log messages in reaction to the submit report button being clicked by the user. We can see that the OPTIONS /saveReport/ accepts the response with a 200 message and then the POST /saveReport/ responds with a successful 200 message as well as a 40 message to indicate the creation of the resource.



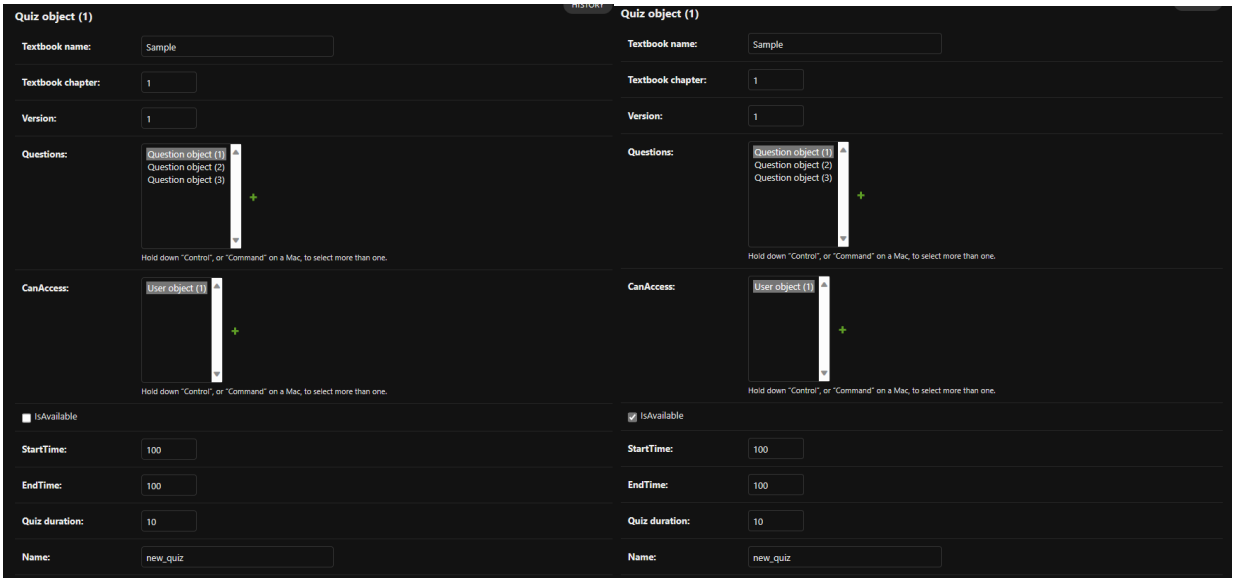
Finally, the image above shows the result of the report being saved in the database with a unique primary key id value as well as the two data values that were passed by the submitReport. The question_index column displays which question number the text belongs to and the report_text is the string content of the report itself.

Instructor User Story #9: Specify the starting time of quizzes

When the instructor creates a quiz, they specify a start time to the API. The start time automatically creates an 'at' command on the server to run the Django command which makes the quiz available. Below is the command line output of creating the 'at' command:

```
Capstone_Project/genq/Backend/GenQ/GenQ$ job 9 at Tue Nov 28 03:25:00 2023
```

When the server clock reaches that time, the command triggers, and the is_available field on the associated quiz object is set to True. This lets the front end know that the quiz is now available for students to view and take.



Shown on the left is the quiz object before the specified time, on the right, after the specified time. Note that the isAvailable field becomes marked as True.

5. Scrum

a. Sprint#3 User Stories:

i. **Student US#3:** As a student, I want to view my past practice exam scores so that I can gauge my progress. (3)

ii. Instructor Story 4: As an instructor, I want to export generated tests to a printable format so that I can distribute them to students offline. (3)

iii. Instructor Story 9: As an instructor, I want to start quizzes at certain times of the day so that I can make sure all students take the quiz at the same time. (3)

iv. Instructor Story 11: As an instructor, I would like to see usage and performance reports for my tests/quizzes so that I can gauge student performance and usage (5)

v. Student Story 2: As a student, I want to report questions that seem to have an incorrect answer so that my instructor can fix the question/answer. (3)

vi. General Story: Random assigning of quizzes to students and student access to quiz (assigned by quiz version) (5)

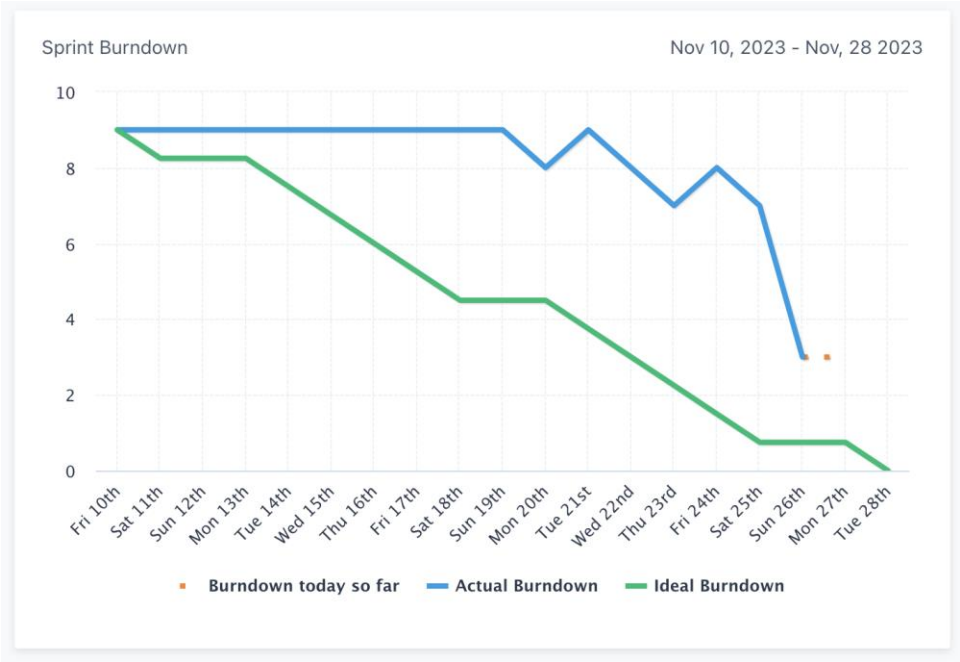
vii. Student US#1: Student Accesses Sample Questions (3)

viii. General Story: Instructor assigning of quizzes to students

ix. General Story: Student Access

x. General Story: Change Frontend routes to web flow interactions

b. **Sprint 3 burndown chart:**



c. **Sprint retrospective**

   i. *Did you implement all of the user stories you specified on the product roadmap for this release? If not, why not?*
      1. *Over committed too many stories*
      2. *Undervalued some stories in terms of story points*

   ii. *What impediments did you encounter? How did you address them?*
      1. *Readme for LLMAPI was not available for all team*
         a. *Solution: added a Readme for LLMAPI explaining how it works and how to use it.*
      2. *Limited Development Resources caused issues for some functionalities (Shared Development Environment)*
         a. *Solution: for now, we got it working on single computer for a production environment we would need more resources for full functionality.*

   iii. *What went well?*
      1. *Team communication went well even when some members were on holiday*
      2. *Code readability helped for complex functionality during the sprint*
      3.

   iv. *What could be improved?*

        1. *Time Management during the holiday season could be improved as many members as possible of our team had to travel and weren't able to finish up their stories in time.*

d. Product backlog (updated)

- i. Added User Stories
    1. General User Stories:
        a. Change Frontend routes to web flow interactions.
        b. Student Access
        c. Instructor assigning of quizzes to students

e. Issue Tracking

- i. Gitlab Issue tracking [LINK]