SWE 363

Quiz Backend Challenge

Create a backend for the Quiz application you developed in the JavaScript module. Follow these steps

Step 1 create a database and tables

scores	[table]
id	INTEGER
	auto-incremented
"user"	TEXT
score	REAL

questions	[table]
id	INTEGER
	auto-incremented
text	TEXT
choice1	TEXT
choice2	TEXT
choice3	TEXT
choice4	TEXT
answer_index	INTEGER

Step 2: create QuizController module to do the following

1. Connect to the database using `better-sqlite3`
2. Create a method named getQuestions() that returns an array of all the question columns except the answer_index
3. Create a method named addQuestion(info) to insert into the question table.
4. Create a method named getScores()
5. Create a method named checkAnswer(answer) to query the database on the questions table and returns true if the answer[“answer”] is the correct answer for the question with id = answer[“id”]
6. Create a method named addScore(data) to insert in the scores table data[“user”] and data[“score”]

Step 2 create route /admin to enable admin to add questions

1. When receiving a GET request, present a form to enter the question details
2. When receiving a POST request call addQuestion to store the question then show whether or not the insertion was done successfully

Step 3 create /questions_api route

1. When receiving POST request, decode the JSON encoded data and use it to call checkAnswer method then encode the return value as JSON and send it
2. When receiving GET call getQuestions to get an array of questions. Iterate over this array to change it to match the format used in app.js questions array. Then encode it and send it as JSON

The return of getQuestions	The required JSON
{ id : 2 text : What is 2 x 2, choice1 : 2, choice2 : 8, choice3 : 4, choice4 : 6 }	{"id":"2", "text":"What is 2 x 2", "choices":[{"choice":"2"}, {"choice":"8"}, {"choice":"4"}, {"choice":"6"}] }

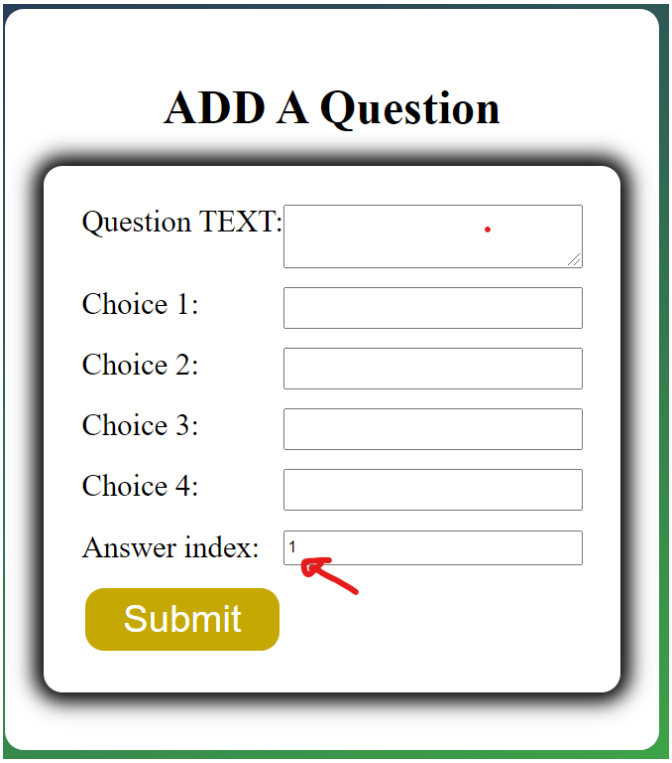

Figure 1 admin.njk all fields are required, the default value for answer index is 1

Step 4 create /scores_api route

1. When receiving GET request call getScores then encode the returned array in JSON and send it
2. When receiving POST call addScore then encode and send the returned value

Step 5 modify index.html to enable the user to enter his name before starting the quiz

Step 6 modify app.js to make use of the backend

1. Replace the hardcoded questions array with an empty one
2. On start function
    a. Check if the user entered his name. if not, set the username as Anonymous
    b. Make AJAX GET request to /questions_api then parse and assign the returned array to the questions variable and calculate the maxPoints.
3. In the createQuestion function set the id attribute of the wrapper div to be the id of the question
4. In the answer() function

a. Create a data object {id: the id of the question, answer: the chosen answer}
b. Make an AJAX POST request to /questions_api sending the JSON representation of the object. When receiving the reply use it to decide on adding the correct or incorrect class.

5. In the showNext() function if there are no more questions to show
    a. Create a data object {user: username, score: the computed score}
    b. Make an AJAX POST request to /scores_api sending the JSON encoded object.
    c. When receiving the response make an AJAX Get request to /scores_api and use the reply to show a table of users' scores.