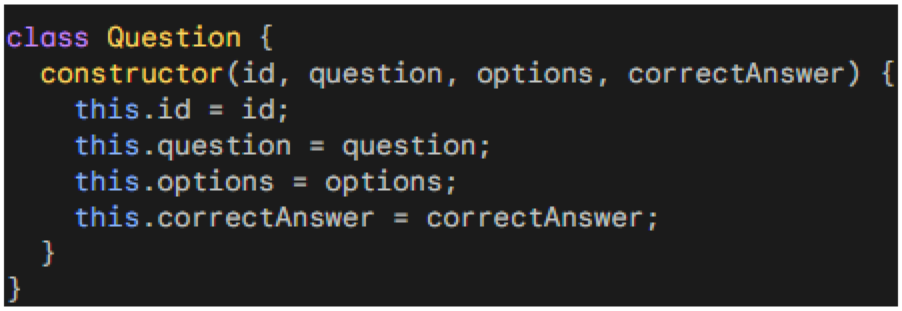
Problem 2: Online Quiz System

Logical design of the quiz application:

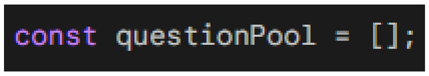
Data Structure

Create a Question class to represent each question:



```
class Question {
  constructor(id, question, options, correctAnswer) {
    this.id = id;
    this.question = question;
    this.options = options;
    this.correctAnswer = correctAnswer;
  }
}
```

Create a QuestionPool array to store the Question objects:



```
const questionPool = [];
```

Question Generation

Create 50 Question objects with unique IDs, questions, options, and correct answers.

Shuffle the questionPool using a JavaScript algorithm to randomize the order of the questions.

Use a loop to display a question from the shuffled questionPool.

Quiz Interface

Create an HTML structure for the quiz interface using HTML elements like div, h1, p, and input.

Style the quiz interface using CSS to make it visually appealing and user-friendly.

Use JavaScript to dynamically generate the question and answer options based on the current question from the pool.

Allow users to select an answer using radio buttons or other appropriate input elements.

Scoring System

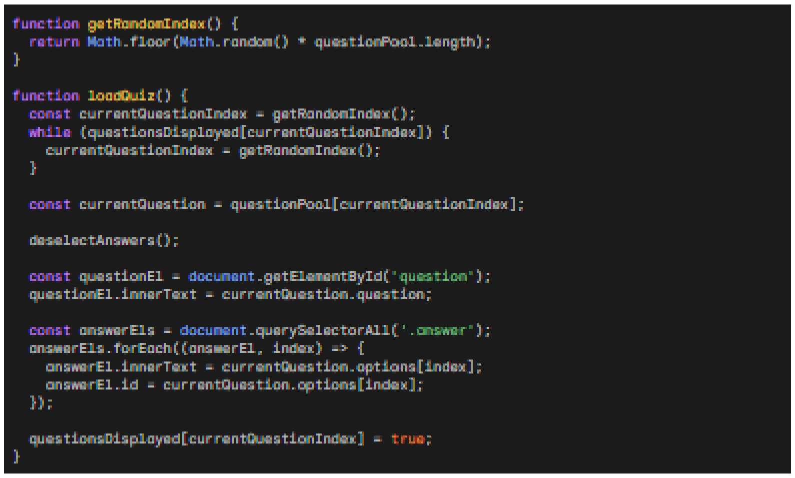
Keep track of the user's score by counting the number of correct answers.

After the user submits their answer, compare their selected option to the correct answer.

If the selected option matches the correct answer, increment the user's score.

Display the user's final score at the end of the quiz.

Logic and Algorithm for Randomized Question Selection



```javascript
function getRandomIndex() {
  return Math.floor(Math.random() * questionPool.length);
}

function loadQuiz() {
  const currentQuestionIndex = getRandomIndex();
  while (questionsDisplayed[currentQuestionIndex]) {
    currentQuestionIndex = getRandomIndex();
  }

  const currentQuestion = questionPool[currentQuestionIndex];

  deselectAnswers();

  const questionEl = document.getElementById('question');
  questionEl.innerText = currentQuestion.question;

  const answerEls = document.querySelectorAll('.answer');
  answerEls.forEach((answerEl, index) => {
    answerEl.innerText = currentQuestion.options[index];
    answerEl.id = currentQuestion.options[index];
  });

  questionsDisplayed[currentQuestionIndex] = true;
}
```

Ensure No Question Repetition

Before displaying a new question, check if the question has already been displayed.

If the question has been displayed, use the getRandomIndex() function to select a different question.

This ensures that each question is displayed only once throughout the quiz.

3. Class and Database Representation (Explanation Only):

Class Diagram:

Question

id (unique)

question

options (array of strings)

correctAnswer (string)

User

id (unique)

name

progress (array of question IDs)

score (integer)

Quiz

id (unique)

questions (array of Question objects)

user (User object)

Question -> Quiz (many-to-one relationship)

User -> Quiz (one-to-many relationship)

Database Schema:

```
create table questions (
  id int primary key,
  question varchar(255),
  options varchar(255),
  correct_answer varchar(255)
);

create table users (
  id int primary key,
  name varchar(255),
  progress varchar(255),
  score int
);

create table quizzes (
  id int primary key,
  questions int references questions(id),
  user int references users(id)
);
```

Flow of Data during the Quiz

The user logs in to the quiz system and creates a new quiz.

The quiz system retrieves 50 questions from the database and stores them in a question pool.

The quiz system randomly selects a question from the pool and displays it to the user.

The user selects an answer from the provided options.

The quiz system verifies the user's answer and updates the user's progress and score.

The quiz system continues to select questions from the pool until all questions have been answered.

The quiz system displays the user's final score and progress.

Relationships between Entities

The Question entity represents a single question in the quiz. It contains the question itself, the options, and the correct answer.

The User entity represents a user who is taking the quiz. It contains the user's name, progress, and score.

The Quiz entity represents a single instance of the quiz. It contains the quiz ID, the questions, and the user.

The Question entity is related to the Quiz entity in a many-to-one relationship. This means that a quiz can have many questions, but a question can only belong to one quiz. The User entity is related to the Quiz entity in a one-to-many relationship. This means that a user can take many quizzes, but a quiz can only be taken by one user.