Step 1: Setting Up the Basic Structure

- **Goal:** Create the HTML structure for the quiz application.
- **Tasks:**
  - Add a container (`div`) for rendering quiz questions with an ID like `quizContainer`.
  - Add a submit button for submitting answers with an ID like `submitQuiz`.
  - Create a hidden section (`div`) to display the score and unanswered questions, with an ID like `scoreContainer`.
- **Practice:**
  - Create basic HTML with the necessary IDs and ensure they are styled for visibility.

Step 2: Adding Questions

- **Goal:** Implement functionality to add questions to the quiz.
- **Tasks:**
  - Write the `addQuestion` method to add a question object to the `questions` array.
  - Include fields like `id`, `text`, `options`, and `correctAnswer` in each question object.
- **Practice:**
  - Add multiple questions to test the quiz setup.

Step 3: Randomizing Questions

- **Goal:** Randomize the order of questions for each quiz attempt.
- **Tasks:**
  - Write the `randomizeQuestions` method to shuffle the `questions` array.
  - Use the Fisher-Yates algorithm or a simple random sort for shuffling.
- **Practice:**
  - Test multiple quiz loads to ensure question order changes.

Step 4: Rendering the Quiz

- **Goal:** Dynamically display the quiz questions and options.
- **Tasks:**

    - Write the `renderQuiz` method to loop through the `questions` array and render each question with its options.
    - Use radio buttons for the options, grouped by the question index.
- **Practice:**
    - Style the questions and options for a better appearance.

Step 5: Collecting User Answers

- **Goal:** Capture the user's selected answers for each question.
- **Tasks:**
    - Write the `collectAnswers` method to gather selected options using `document.querySelector`.
    - Store the answers in the `userAnswers` object, keyed by question ID.
- **Practice:**
    - Test collecting answers with various selections, including unanswered questions.

Step 6: Validating and Scoring the Quiz

- **Goal:** Validate user answers and calculate the quiz score.
- **Tasks:**
    - Write the `calculateScore` method to compare user answers with the correct answers.
    - Count the correct answers for the score.
    - Identify unanswered questions and return them along with the score.
- **Practice:**
    - Test scoring with all correct, some correct, and no answers.

Step 7: Displaying the Score

- **Goal:** Show the user's score and any unanswered questions.
- **Tasks:**
    - Write the `displayScore` method to populate the `scoreContainer` with the score.
    - Display a list of unanswered questions, if any.
    - Ensure the `scoreContainer` is visible after submission.
- **Practice:**
    - Test score display with complete and incomplete answers.

Step 8: Adding Sample Questions

- **Goal:** Add a set of sample questions for the quiz.
- **Tasks:**
  - Use the `addQuestion` method to add questions with diverse formats (e.g., single-choice, true/false).
- **Practice:**
  - Test the quiz with a variety of questions.

Step 9: Submitting the Quiz

- **Goal:** Handle quiz submission and display results.
- **Tasks:**
  - Add a click event listener to the `submitQuiz` button.
  - Call `collectAnswers`, `calculateScore`, and `displayScore` in sequence.
- **Practice:**
  - Test quiz submissions with various user inputs.

Step 10: Polishing the User Interface

- **Goal:** Enhance the visual design and usability of the application.
- **Tasks:**
  - Style the quiz container, questions, options, and score display for a professional look.
  - Use responsive design techniques to ensure usability on mobile and desktop devices.
- **Practice:**
  - Test the application on different screen sizes and devices.

Bonus Steps

1. **Add Timer:**
   - Implement a countdown timer for quiz completion.
   - Auto-submit the quiz when the timer ends.
2. **Add Local Storage:**
   - Save the user's answers in `localStorage` to persist progress in case of page reload.

3. **Add Question Review:**
    - Allow users to review their answers with correct and incorrect markings after submission.
4. **Generate Quiz Report:**
    - Create a downloadable report of the user's answers and score.
5. **Dark Mode:**
    - Add a toggle for switching between light and dark themes.

Practicing Each Step Implement and test each step independently. Once all steps are complete, integrate them to create a fully functional **Quiz Application**.