EXPERIMENT 06

**Aim**: Design a web application using JavaScript.

**Objective**: Students will be able to:

 Understand and implement Quiz Web Application using Javascript.

 Understand javascript basics and DOM.

**Theory**:

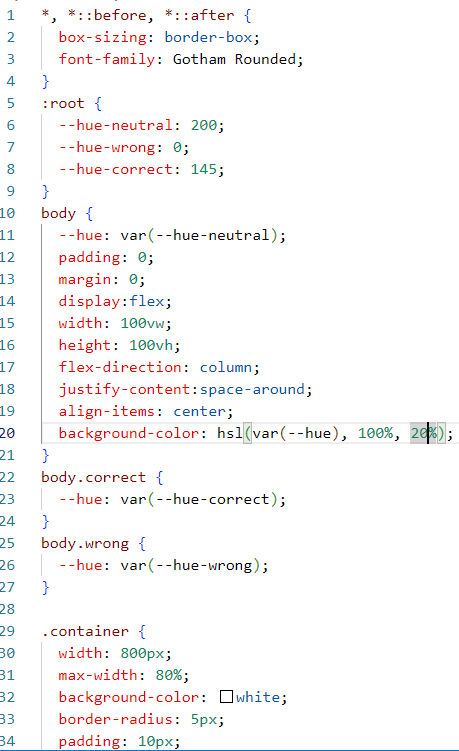
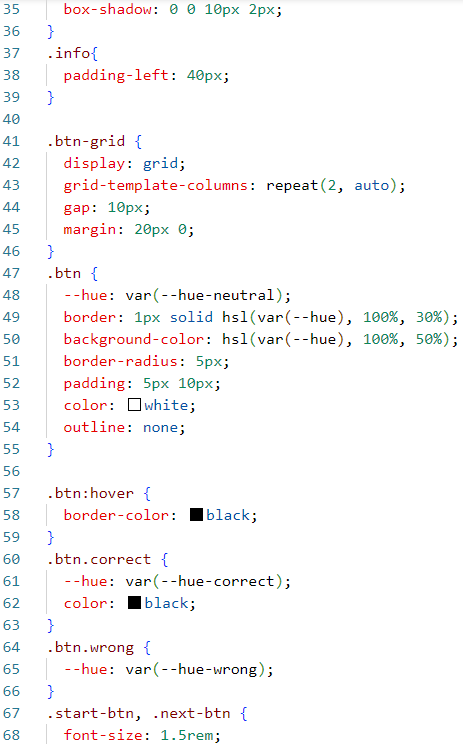
The below code is an implementation of a simple quiz application using HTML, CSS, and JavaScript. It allows users to take a quiz with multiple-choice questions, provides feedback on correct and incorrect answers, and allows them to move through the questions.

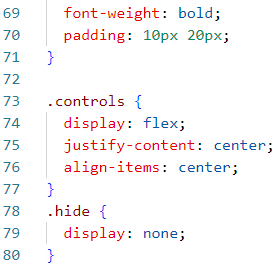
1. **HTML Structure**:
   * The HTML file starts with a standard document declaration and defines the document’s structure.
   * It includes metadata like character encoding, viewport settings, and the document’s title.
   * External CSS and JavaScript files are linked to the HTML file.
2. **CSS Styles**:
   * The CSS code defines the styles for various elements in the quiz app.
   * It uses CSS variables for color management and sets up the overall layout, fonts, and button styles.
   * The layout is designed with containers for information, questions, and answer buttons.
   * Buttons have different styles depending on whether they are correct or wrong.
3. **JavaScript Implementation**:
   * JavaScript code is used to add functionality to the quiz app.
   * Event listeners are set up for the “Start” button and the “Next” button.
   * The startGame function is called when the “Start” button is clicked.
4. **startGame Function**:
   * This function hides the “Start” button, shuffles the quiz questions randomly, and initializes the current question index.
   * It also reveals the question container by removing the ‘hide’ class.
5. **setNextQuestion Function**:
   * This function is responsible for setting up the next question in the quiz.
   * It calls the resetState function to clear previous question data.
   * Then, it calls showQuestion to display the current question.
6. **showQuestion Function**:
   * This function displays the question and answer choices on the screen.
   * It dynamically creates buttons for each answer choice and attaches event listeners to them.
   * If an answer is correct, it sets a custom attribute (data-correct) on the button.
7. **selectAnswer Function**:
   * This function is triggered when a user selects an answer.
   * It checks whether the selected answer is correct using the data-correct attribute.
   * It updates the visual feedback by adding CSS classes (‘correct’ or ‘wrong’) to the selected answer button and all answer buttons.
   * If there are more questions, it shows the “Next” button; otherwise, it shows a “Restart” button.
8. **setStatusClass and clearStatusClass Functions**:
   * These functions add and remove CSS classes to set the visual feedback for correct and wrong answers.
9. **Quiz Data**:
   * Quiz questions and answers are stored in an array of objects, where each object represents a question.
   * Each question object includes the question text and an array of answer choices, each with text and a correctness flag.
10. **Event Listeners**:
    * Event listeners are used to respond to user interactions, such as clicking the “Start” and “Next” buttons and selecting answer choices.

In summary, this quiz app implementation combines HTML for the structure, CSS for styling, and JavaScript for interactivity. It manages the flow of questions, tracks correct and incorrect answers, and provides visual feedback to the user. It’s a simple but functional example of a quiz application.

**Source code:**

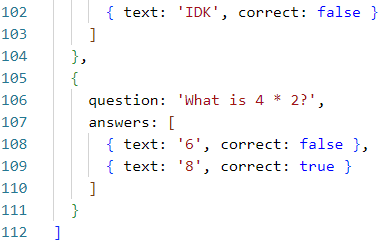
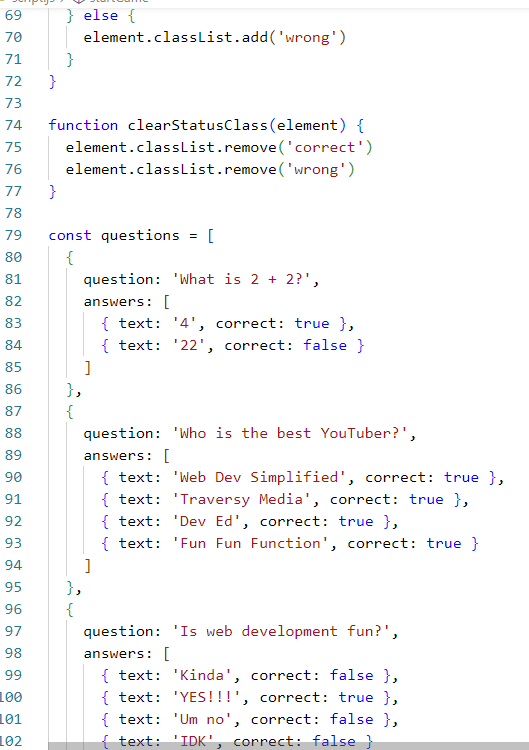


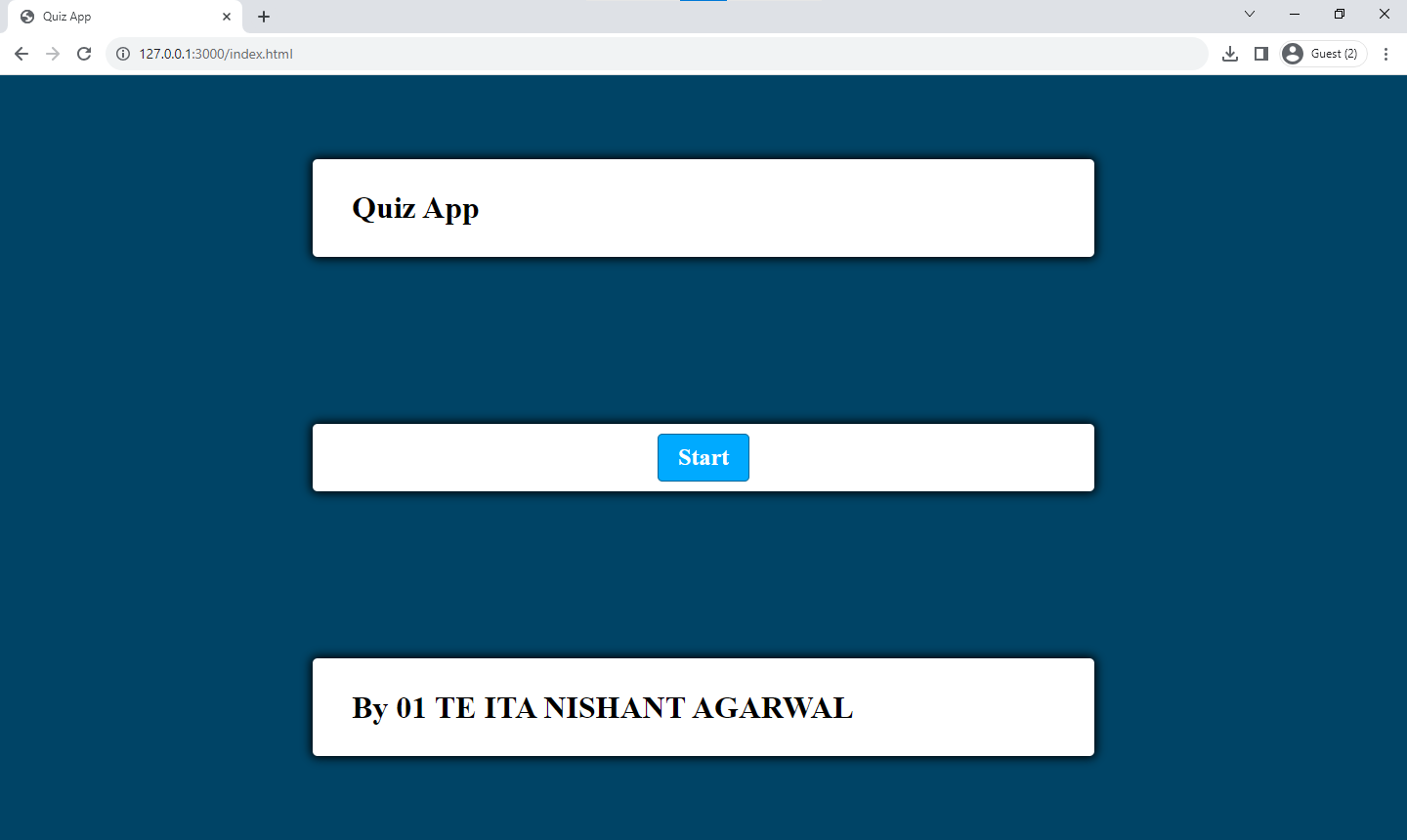


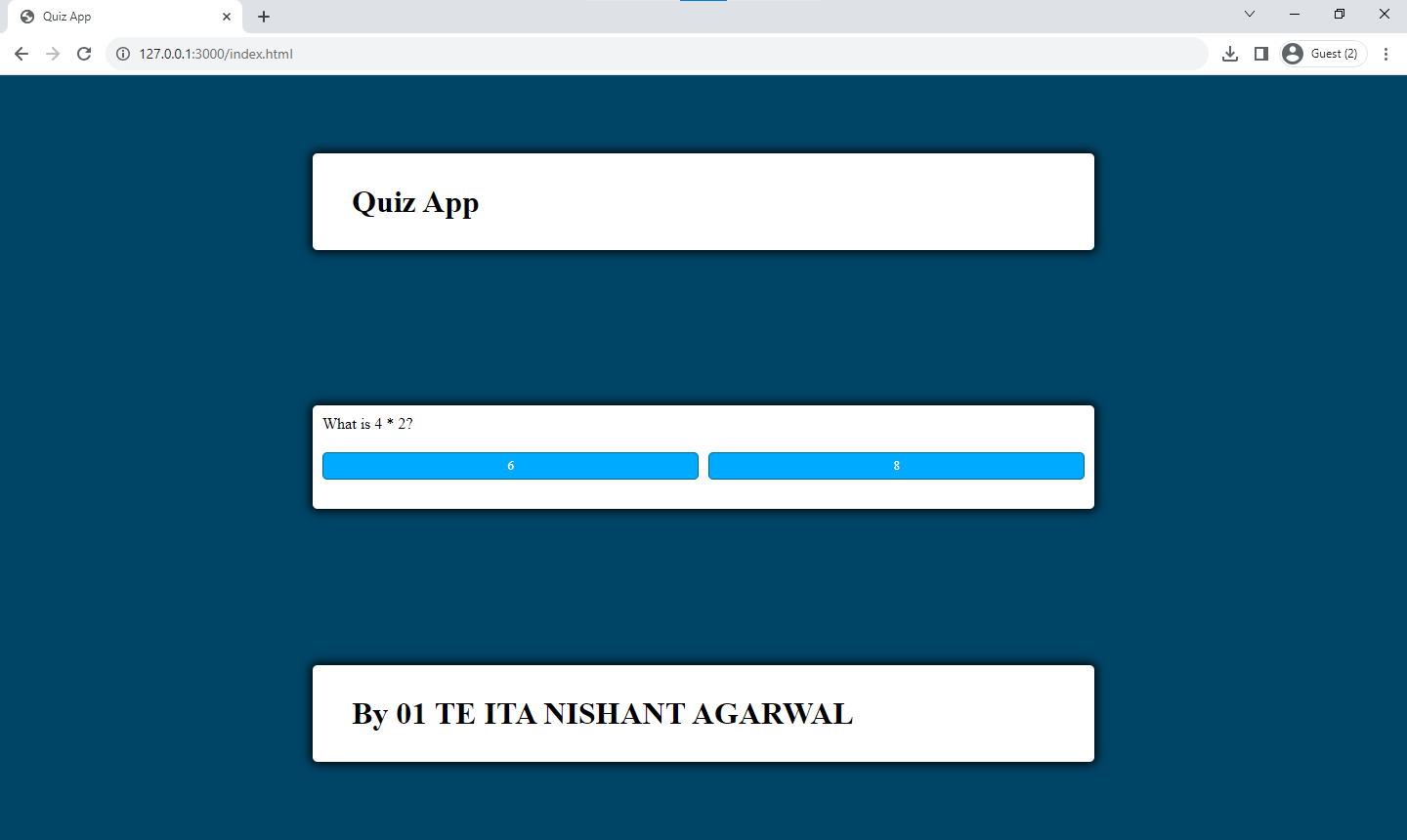




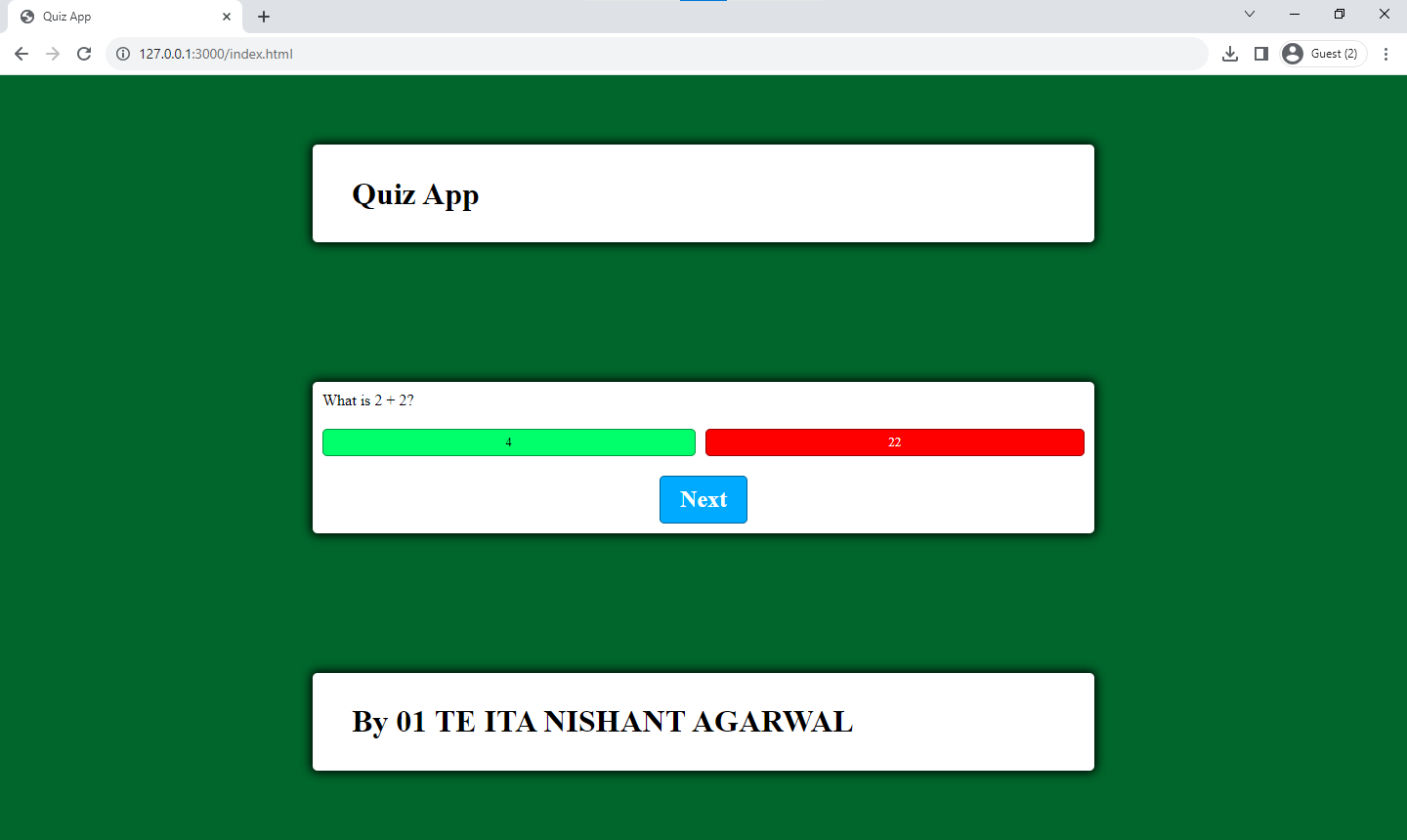


Output (web page):









Conclusion:

In conclusion, this implementation offers a functional and interactive quiz application using HTML, CSS, and JavaScript. It effectively handles the presentation of questions and answer choices, tracks correct and incorrect answers, and provides visual feedback to the user. The code's modular structure allows for easy management of quiz data and user interactions, making it a suitable starting point for more complex quiz applications. Additionally, the use of CSS variables enhances the app's visual appeal and maintainability. Overall, this implementation serves as a practical example of how to create a basic quiz application on the web.

Lab Outcome: Students were able to:

 Implement Quiz Web Application using Javascript.

 Understood javascript basics and DOM.

**COs attained:**

CO1 (Implement interactive web page(s) using HTML)

CO2 (Create Responsive Web Design with CSS & Bootstrap)

CO3 (Design and develop web applications using JavaScript) **POs attained:**

PO 1: ENGINEERING KNOWLEDGE (Apply Knowledge of Mathematics, Science,

engineering fundamentals and an engineering specialization to the solution of complex engineering problems.)

PO 2: PROBLEM ANALYSIS (Identify, formulate, research literature and analyse complex

engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.)

PO 3: DESIGN / DEVELOPMENT OF SOLUTIONS (Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.)

PO 5:MODERN TOOL USAGE (Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.)

**PEOs achieved:**

PEO 1: To prepare learners with a strong foundation in the area of Information Technology required solving real life problems arising from software technology. (Knowledge)(CURRICULAR)

PEO 3: To prepare learners to understand the need for lifelong learning with effective written

and oral communication skills and to be able to readily adapt to new software engineering environments. (PRESENTATION AND GROWTH)