JS Capstone Project Marks 70 Time 3:00 Hrs

**Question 1: Web Form Validation**

1. **Design a Web Form: 5 Marks**
   * Create an HTML page containing a form with the following input fields:
     + **Name** (text input)
     + **Email** (email input)
     + **Password** (password input)
     + **Confirm Password** (password input)
     + **Phone Number** (text input)
     + **Date of Birth** (date input)
     + **Gender** (radio buttons: Male, Female, Other)
     + **Country** (dropdown/select list)
     + **Agree to Terms and Conditions** (checkbox)
   * Add a submit button at the end of the form.
2. **Basic HTML Validation:**
   * Ensure that appropriate HTML5 validation attributes (e.g., required, type, minlength, maxlength, pattern) are used for each input field where applicable.

**Part 2: Implement Client-Side JavaScript Validation 15 Marks**

1. **Validate Name Field:**
   * Ensure the name is not empty and contains only alphabetic characters and spaces.
   * Display an error message if the input is invalid.
2. **Validate Email Field:**
   * Use a regular expression to validate that the email is in the correct format (e.g., example@domain.com).
   * Display an error message if the email format is incorrect.
3. **Validate Password Fields:**
   * Ensure the password is at least 8 characters long and contains at least one uppercase letter, one lowercase letter, one digit, and one special character.
   * Validate that the "Confirm Password" field matches the "Password" field.
   * Display appropriate error messages for both conditions.
4. **Validate Phone Number Field:**
   * Ensure the phone number contains only digits and is 10-15 characters long.
   * Optionally, include validation for specific country formats if relevant.
   * Display an error message if the input is invalid.
5. **Validate Date of Birth:**
   * Ensure the user is at least 18 years old based on the selected date of birth.
   * Display an error message if the user is under 18.
6. **Validate Gender Selection:**
   * Ensure that one of the gender options is selected.
   * Display an error message if no option is selected.
7. **Validate Country Selection:**
   * Ensure a country is selected from the dropdown list.
   * Display an error message if no country is selected.
8. **Validate Terms and Conditions Checkbox:**
   * Ensure that the user has checked the "Agree to Terms and Conditions" checkbox.
   * Display an error message if the checkbox is not checked.
9. **Notification: 5 Marks**

* Show Notification to the User he had successfully registered.

1. **Display Error Messages:**

* Ensure that error messages are displayed in a user-friendly manner near the relevant input field.
* Use CSS to style the error messages and highlight invalid fields.

1. **Disable Submit Button: 5 Marks**

* Disable the form's submit button by default.
* Enable the submit button only when all fields are valid

**Question 2:** 25 Marks

1. **Create a Simple Web Page: 5 Marks**
   * Design a basic HTML page with a button labelled "Fetch Data" and an empty div element where the fetched data will be displayed.
2. **Implement a GET Request: 5 Marks** 
   * Write JavaScript code to create an XMLHttpRequest object.
   * Implement a GET request to retrieve JSON data from a public API (e.g., <https://restcountries.com/v3.1/all>).
   * Display the fetched data inside the div element when the "Fetch Data" button is clicked.
3. **Implement a POST Request: 5 Marks**
   * Extend your web page to include a form with input fields for title and body.
   * Write JavaScript code to create and send a POST request using XMLHttpRequest to submit the form data to a server (e.g., https://restcountries.com/v3.1/all).
   * Display the response from the server on the webpage.
4. **Implement PUT and DELETE Requests: 10 Marks**
   * Add functionality to update and delete a post using PUT and DELETE requests.
   * Use XMLHttpRequest to send these requests and update the webpage based on the server's response.

**Question 3: To Do App 15 Marks**

**Create the Basic Structure:**

* Develop an HTML page that includes:
  + An input field and a button to add new tasks.
  + A section to display the list of tasks.
* Use CSS to style the to-do list, making sure it is visually appealing and user-friendly.

1 Implement **Task Addition:**

* Write JavaScript code to capture the user's input when the "Add Task" button is clicked.
* Add the new task to the task list dynamically.
* Clear the input field after the task is added.

2 Delete **Tasks:**

* Add a delete button next to each task.
* Write JavaScript to remove the task from the list when the delete button is clicked.