1) **<u>Frontend Question:</u>**
Html requirements:

a) Add *aria-required, aria-describedby,* and *aria-live* attributes for accessibility.
b) Include fields for Name, Email, and Message with proper labels and hints.
c) Ensure the *submit* button has *aria-label*.
d) Add role="main" to main element

JavaScript Functionality

e) *validateForm* Function: Clear existing error messages, validate Name (not empty), Email (not empty/invalid), and Message (not empty). Display error messages and set focus on the first invalid field.

Question:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,
initial-scale=1.0">

  <!-- Set the page title as Accessible Contact Form -->



  <link rel="stylesheet" href="index.css">

</head>

<body>

  <header>

    <h1>Contact Us</h1>

  </header>
  <!-- Add role as "main"-->

  <main>



    <!-- Error Messages: Set aria-live as "polite"-->

    <div id="error-message" class="visually-hidden" ></div>



    <form id="contact-form" aria-labelledby="form-heading">

      <h2 id="form-heading" class="visually-hidden">Contact
Form</h2>



      <div class="form-group">

        <label for="name">Name</label>

        <!-- Create input field for name -->

        <!-- Add id as "name" -->

        <!-- Set type to "text" -->

        <!-- Add placeholder "Enter your name" -->

        <!-- Add aria-required as "true" -->

        <!-- Add aria-describedby pointing to the "name-hint"
element -->

        <span id="name-hint" class="visually-hidden">Your full name
is required.</span>

      </div>





      <div class="form-group">
```

```
        <label for="email">Email</label>

        <!-- Create input field for email -->

        <!-- Add id as "email" -->

        <!-- Set type to "email" -->

        <!-- Add placeholder "Enter your email" -->

        <!-- Add aria-required as "true" -->

        <!-- Add aria-describedby pointing to the "email-hint"
element -->

        <span id="email-hint" class="visually-hidden">A valid email
address is required.</span>

      </div>




      <div class="form-group">

        <label for="message">Message</label>

        <!-- Create textarea for message -->

        <!-- Add id as "message" -->

        <!-- Add placeholder "Type your message here" -->

        <!-- Set rows to "5" -->

        <!-- Add aria-required as "true" -->

        <!-- Add aria-describedby pointing to the "message-hint"
element -->

        <span id="message-hint" class="visually-hidden">Please enter
your message.</span>

      </div>
```

```
    <!-- Add aria-label in submit button "Submit Contact Form" -->

    <button type="submit" >Submit</button>

    </form>

  </main>



  <footer>

    <p>Keyboard shortcuts: Tab to navigate, Enter to submit.</p>

  </footer>



  <script src="index.js"></script>

</body>

</html>
```

<u>Sample Code</u>: Here you can observe that how based on mentioned comments, we can write the code:

```
  <!-- Add role as "main"-->
  <main role="main" >

    <!-- Error Messages: Set aria-live as "polite"-->
    <div id="error-message" aria-live="polite"   class="visually-hidden" ></div>

    <form id="contact-form" aria-labelledby="form-heading">
      <h2 id="form-heading" class="visually-hidden">Contact Form</h2>

      <div class="form-group">

        <label for="name">Name</label>
        <input id="name" type="text" placeholder="Enter your name" aria-required="true"
aria-describedby="name-hint">
```

```
        <!-- Create input field for name -->
        <!-- Add id as "name" -->
        <!-- Set type to "text" -->
        <!-- Add placeholder "Enter your name" -->
        <!-- Add aria-required as "true" -->
        <!-- Add aria-describedby pointing to the "name-hint" element -->
        <span id="name-hint" class="visually-hidden">Your full name is required.</span>
      </div>


      <div class="form-group">
        <label for="email">Email</label>
        <input id='email' type='email' placeholder='Enter your email' aria-required='true'
aria-describedby='email-hint'/>
        <!-- Create input field for email -->
        <!-- Add id as "email" -->
        <!-- Set type to "email" -->
        <!-- Add placeholder "Enter your email" -->
        <!-- Add aria-required as "true" -->
        <!-- Add aria-describedby pointing to the "email-hint" element -->
        <span id="email-hint" class="visually-hidden">A valid email address is
required.</span>
      </div>


      <div class="form-group">
        <label for="message">Message</label>
        <input id='message' placeholder='Type your message here' rows='5'
aria-required='true' aria-describedby='message-hint'/>
        <!-- Create textarea for message -->
        <!-- Add id as "message" -->
        <!-- Add placeholder "Type your message here" -->
        <!-- Set rows to "5" -->
        <!-- Add aria-required as "true" -->
        <!-- Add aria-describedby pointing to the "message-hint" element -->
        <span id="message-hint" class="visually-hidden">Please enter your
message.</span>
      </div>

     <!-- Add aria-label in submit button "Submit Contact Form" -->
```

```
    <button type="submit" aria-label='Submit Contact Form' >Submit</button>
    </form>
  </main>
```

2) **Frontend Question:**

Complete codes in *index.html*, *index.css* and *index.js* files present in the *website* folder

1. HTML Tasks
   - Set the page title to "*Recipe Ingredient Calculator*".
   - Create a dropdown menu with id="recipe" and class="form-select" for selecting recipes.
   - Add an unordered list with id="ingredients-list" and class="list-group list-group-flush" for displaying ingredients.
2. CSS Tasks
   - Style list items (.list-group-item) with:
     - Font size: 0.95rem.
     - Font weight: 400.
   - Style card components (.card) with:
     - Subtle box shadow.
     - Rounded corners (border-radius).
3. JavaScript Tasks
   - Implement functionality to dynamically update the ingredients list:
   - Retrieve the selected recipe and scale ingredient quantities based on the servings.
   - Create styled list items (<li>) and append them to the ingredients list.

Do practice the code based on above instructions.

3) Full Stack Question: can be like Event management Application,Task Management Application

Instructions:

Backend:

Implement the below functions in backend>src>controller.js

i) getAllEvents/getAllTasks- need to use dbmodel.find()

ii)AddEvent/AddTask - need to use dbmodel.save() or dbmodel.insertOne() or dbmodel.create()

iii)RemoveEvent/RemoveTask- need to use dbmodel.FindByIdAndDelete(id)

iv)Update Event/Update Task- need to use dbmode.FindByIdAndUpdate();

Note:

So if a particular filename is mentioned then do edit only the mentioned files, and wherever the comments are mentioned do edit only that part.You dont need to implement whole project code, do implement only based on instructions mentioned

FrontEnd:

Connect frontend with the backend using the backend url- on clicking the add button of frontend- the AddEvent/AddTask should get implemented from backend.

Similarly perform update button,remove button,display button.

Reference:

https://youtu.be/cOY_4JrtT20?si=_edXbeIYpYbo46Fg

https://youtu.be/hZmtWdhawg8?si=euMAQNx0XkSQ23k1