Mock Lab Exam: A Restaurant Page using HTML, CSS, and JS with Python

Duration: 90 minutes

Instructions:

- This lab exam assesses your ability to create a basic restaurant page using HTML, CSS, and JavaScript, with Flask Python for server-side functionality.
- You have 90 minutes to complete the lab exam.
- The focus is on creating a responsive and visually appealing restaurant page that incorporates basic functionality.

Requirements:

HTML:

1. Create a main HTML file named `index.html`.
2. Use semantic HTML5 elements like `<header>`, `<nav>`, `<section>`, `<footer>`, etc.
3. Include a navigation menu with links to Home, Menu, About, and Contact sections.
4. Design a homepage section with a welcome message, a brief introduction to the restaurant, and a call-to-action button (a prominently visible button). Its functionality is explained in the JS section.
5. Create a Menu section displaying at least three menu items with images, descriptions, and prices.
6. Include an About section with information about the restaurant, its history, and mission.
7. Add a Contact section with a form including fields for Name, Email, and Message.

CSS:

1. Create a CSS file named `style.css`.
2. Implement a responsive design using CSS Grid or Flexbox.
3. Style the navigation menu to be horizontally aligned and responsive.

4. Design a visually appealing layout for the homepage, menu items, about section, and contact form.
5. Use appropriate colors, fonts, and spacing to enhance the visual presentation.

JavaScript:

1. Create a JavaScript file named `script.js`.
2. Add functionality to the call-to-action button to scroll smoothly to the Menu section.
3. Validate the contact form fields for Name and Email to ensure they are not empty.
4. Implement a basic form submission function to handle form data.

Flask Python:

1. Install Flask using pip (`pip install Flask`).
2. Create a Flask app file named `app.py`.
3. Set up routes for serving the main HTML file and handling form submission.
4. Use Flask to render the `index.html` template and handle POST requests from the contact form.
5. Implement basic server-side validation for form data (e.g., ensuring email format is valid).

Submission:

1. Ensure all HTML, CSS, JavaScript, and Flask Python files are included.
2. Zip the entire project directory and name it as <RollNumber>.zip.
3. Upload the zip file on Moodle.