Short Answer:

Answer the following questions with complete sentences in **your own words**. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers.

1. What are static websites?
2. What are dynamic websites?
3. What are the major differences between static and dynamic websites?
4. What is first contentful paint and time to interactive?
5. What is SSR, CSR, and SSG? When would you use each one?
6. How do they affect page performance?
7. What are template engines and their advantages?
8. What is DRY?
9. What is CRUD?
10. What is a seed file for?

Coding Questions:

Use HTML/CSS/JS to solve the following problems. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. **To-Do App (Express.js, EJS, SCSS)**
   Use EJS & SCSS to create the frontend for yesterday's to-do application. It should be fully-functional and connected (can send requests) to the server. Postman screenshots are not needed.
   - The to-do app frontend should include the following feature requirements (based on the same routes). Refreshing the page should show the most updated to-do list.

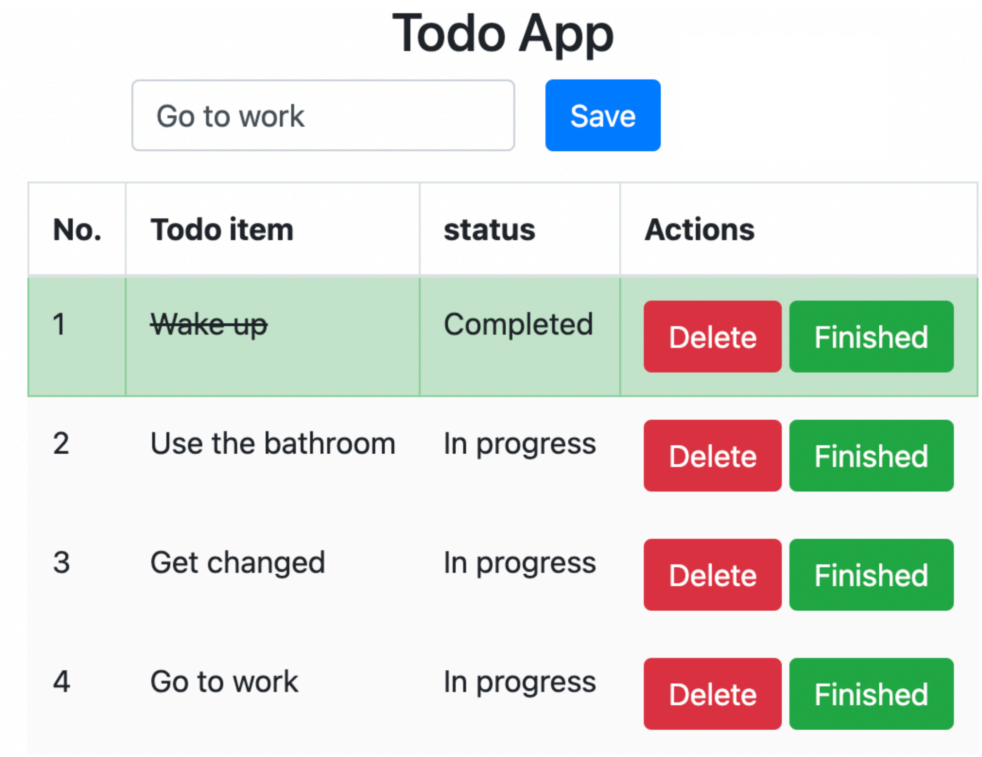
   1. `POST: /todo/{filename}`
      When the user fills out the form to add a new to-do item and clicks "Save", it should send the POST request to the server.
      - Title & description are input text fields, and status & priority are dropdowns.
      - If any of the text fields are empty, it should not send the request, and notify the user appropriately.
      - If there is an error from the server response, display an appropriate error message on the page.
      - If the request was successfully completed, display an appropriate success message on the page.

   2. `GET: /todo/{filename}`
      When the user navigates to this page, the server should render the corresponding EJS template. It should display all the existing to-do items in the file.
      - If there is an error from the server response, display an appropriate error message on the page and redirect to the home page (can be anything).
      - If the request was successfully completed, display an appropriate success message on the page.

3. `PUT: /todo/{filename}`
   Each to-do item should have an “Edit” button that when clicked, allows them to modify the to-do item (which becomes text fields & dropdowns). When the user clicks “Save”, it should send the PUT request to the server.
   - If there is an error from the server response, display an appropriate error message on the page.
   - If the request was successfully completed, display an appropriate success message on the page.

4. `DELETE: /todo/{filename}`
   When the user clicks a “Delete” button to remove this to-do item, it should send the DELETE request to the server.
   - If there is an error from the server response, display an appropriate error message on the page.
   - If the request was successfully completed, display an appropriate success message on the page.

5. Follow the styling below. You should still have the other input fields and display their properties in the table row.



No.	Todo item	status	Actions
1	~~Wake up~~	Completed	Delete Finished
2	Use the bathroom	In progress	Delete Finished
3	Get changed	In progress	Delete Finished
4	Go to work	In progress	Delete Finished

Paired Programming:

Use JS to solve the following problems with your partner. Please remember to label who was acting as the driver and navigator for each problem and write down your feedback on their performance. **Feedback will be confidential**.

Leetcode #409: Longest Palindrome
Leetcode #104: Maximum Binary Tree