

**Lab # 08**

**Web Engineering  
Fall 2020**

****

|  |  |
| --- | --- |
| Instructor |  |
| Student Name |  |
| CMSID |  |
| Department |  |
| Semester |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lesson Set 8** | **CRUD Operations Using**  **JavaScript** | | | |
| **Purpose** | 1. To get a basic awareness of CRUD 2. To understand CRUD and why we are using it. 3. To learn the basics of JavaScript and create simple pages. 4. File Handling | | | |
| **Procedure** | 1. Students should read the Pre-lab Reading assignment before coming to the lab. 2. Students should complete the Pre-lab Writing assignment before entering the lab. 3. In the lab, students should complete Labs 8.1 through 8.2 in sequence. Your instructor will give further instructions as to grading and completion of the lab. 4. Students should complete the set of lab tasks before the next lab and get them checked by their lab instructor. | | | |
|  | **Contents** | **Pre-requisites** | **Completion Time** | **Page Number** |
|  | Pre-lab Reading Assignment | - | 20 min | 3 |
|  | Pre-lab Writing Assignment | Pre-lab Reading | 10 min | 4 |
|  | **Lab 8** | | | |
|  | **Lab 8.1**  CRUD Operations | Pre-lab reading | 30 min | 5 |
|  | **Lab 8.2**  Lab Tasks | Awareness with JavaScript | - | 9 |

|  |  |
| --- | --- |
| **PRE-LAB READING ASSIGNMENT** | |
| **JavaScript CRUD** | CRUD is an acronym that stands for Create, Read, Update, and Delete. It represents the four basic operations performed on data in most database systems. In the context of JavaScript, CRUD operations refer to manipulating and managing data using JavaScript code.  Let's go through each operation in detail with examples:  1. **Create (C):**  Creating refers to adding new data to a system or database. In JavaScript, you can create new data by capturing user input, processing it, and adding it to the desired data structure or storage.  Example:  *// Create an array to store user data*  *let users = [];*  *// Function to add a new user to the array*  *function createUser(name, email) {*  *const user = {*  *name: name,*  *email: email*  *};*  *users.push(user);*  *}*  *// Call the createUser function with user input*  *createUser('John Doe', 'johndoe@example.com');*    **2. Read (R):**  Reading involves retrieving and accessing data from a system or database. In JavaScript, you can read data by accessing and retrieving values from data structures, APIs, or external sources.  Example:  *// Read data from an array*  *function getUsers() {*  *return users;*  *}*  *// Call the getUsers function to get all users*  *const allUsers = getUsers();*  *console.log(allUsers);*    **3. Update (U):**  Updating means modifying or changing existing data in a system or database. In JavaScript, you can update data by identifying the specific data you want to modify and then applying the necessary changes.  Example:  *// Update user data in the array*  *function updateUser(index, newName) {*  *if (index >= 0 && index < users.length) {*  *users[index].name = newName;*  *}*  *}*  *// Call the updateUser function to update a user's name*  *updateUser(0, 'Jane Doe');*    **4. Delete (D):**  Deleting refers to removing data from a system or database. In JavaScript, you can delete data by locating the specific data you want to remove and then removing it from the data structure or storage.  Example:  *// Delete user data from the array*  *function deleteUser(index) {*  *if (index >= 0 && index < users.length) {*  *users.splice(index, 1);*  *}*  *}*  *// Call the deleteUser function to delete a user*  *deleteUser(0);*  CRUD operations are fundamental for managing data in various applications, including web development, where you often need to create, read, update, and delete data in response to user interactions or system requirements. These operations allow you to manipulate data dynamically, update user interfaces, and interact with backend systems or databases. |
|  |  |

|  |  |
| --- | --- |
| **PRELAB WRITING ASSIGNMENT** | |
| **Fill in the blanks** | 1. CRUD operations are the \_\_\_\_\_\_\_\_ operations used to \_\_\_\_\_\_\_\_data in a system. 2. Create operation is used to \_\_\_\_\_\_\_\_new data or \_\_\_\_\_\_\_\_to a database or data structure. 3. Read operation is used to \_\_\_\_\_\_\_\_or \_\_\_\_\_\_\_\_existing data from a database or data structure. 4. Update operation is used to \_\_\_\_\_\_\_\_or \_\_\_\_\_\_\_\_existing data in a database or data structure. 5. Delete operation is used to \_\_\_\_\_\_\_\_or \_\_\_\_\_\_\_\_existing data from a database or data structure. |

|  |
| --- |
| **Lab 8.2 LAB TASK** |

**Task: Recipe Management System (Using HTML, CSS, and JavaScript)**

**Description:**

You are tasked with creating a Recipe Management System using only HTML, CSS, and JavaScript. The system should allow users to add, view, update, and delete recipes. You will need to implement both the front-end interface and the back-end logic using JavaScript.

**Requirements:**

1**. Front-End:**

- Design a user-friendly interface using HTML and CSS.

- Create an HTML form to add new recipes with fields such as title, ingredients, instructions, and an image upload option.

- Display a list of existing recipes on the main page using HTML elements.

- Provide options to view, edit, and delete individual recipes using buttons or links.

- Implement client-side validation to ensure required fields are filled in correctly.

**2. Back-End (Using JavaScript):**

- Store the recipe data in an array or object within your JavaScript code.

- Create functions to handle the CRUD operations:

- Add a new recipe to the data structure.

- Retrieve and display the list of recipes on the main page.

- Implement functionality to view, edit, and delete individual recipes.

- Update the data structure and the displayed recipes accordingly after each operation.

**3. Additional Features (Optional):**

- Implement JavaScript search functionality to allow users to search for recipes based on title or ingredients.

- Add pagination to the recipe list to handle large amounts of data.

- Store the data locally using the browser's LocalStorage API to persist the recipes across page reloads.

- Enhance the user interface with CSS transitions, animations, or responsive design techniques.

Notes:

- Upload into GitHub pages and provide a link in the given box below.

- Focus on writing clean, modular, and maintainable code.

- Test your application thoroughly to ensure it works correctly and handles edge cases.

- Make the user interface visually appealing and user-friendly using CSS.

|  |
| --- |
|  |

Good luck with the task, and enjoy building your Recipe Management System using HTML, CSS, and JavaScript!