Database Management System Lab

Name: Tonmoy Biswas

Roll No: 002110501133

Class: BCSE 3rd Year 2nd Sem

Section: A3

Assignment No: 0.1 and 0.2

PROBLEM :- 1

- **Problem statement :-** Develop an application as follows. User will enter two numbers . Provide button to add or to find the difference. Depending on the option result will be shown in result box . Add a suitable title to the screen . When ever the form is loaded , also display current date in the title bar.

- **Software Used :-** The software used in this project is a React application. Additionally, it seems like it's built using Vite, a fast build tool for modern web development that enhances the development experience for React applications.

- **GUI :-** The GUI consists of a single component, Card, which represents a calculation card with inputs, buttons, and a result display. The card has a title, date, two input fields, buttons for addition and finding the difference, and an area to display the result.

- **Component Properties :-**

  The “Card” component has the following state properties :

  - input1: Holds the value of the first input field.
  - input2: Holds the value of the second input field.
  - result: Holds the result of the calculation.
  - date: Holds the formatted date string.

- **EVENT-HANDLER (PSEUDO CODE) :-**

```
➢ const handleAddition = () =>
  {
  // Convert inputs to numbers
  const val1 = Number(input1);
  const val2 = Number(input2);
  // Add values
  const sum = val1 + val2;
  // Set result
  setResult(sum);
  }
➢ const handleDifference = () =>
  {
  // Convert inputs to numbers
  const val1 = Number(input1);
  const val2 = Number(input2);
  // Find the absolute difference
  const difference = Math.abs(val1 - val2);
  // Set result
setResult(difference);
}
```

- ➢ useEffect(() => {

```
// Get the current date
const currentDate = new Date();
// Format the date
const formattedDate = currentDate.toLocaleString("en-GB", {
day: "numeric",
month: "short",
year: "numeric",
hour: "numeric",
minute: "2-digit" });
// Set the formatted date in the state
setDate(formattedDate);
},[])
```

- **INTERFACE LAYOUT :-**

  The interface layout can be described as follows:

  - ➢ A titlebar displaying the title "Calculation" and the date.
  - ➢ Two input fields for user input (labeled as 'input 1' and 'input 2').
  - ➢ Buttons for addition and finding the difference.
  - ➢ A section to display the result or a waiting message.
  - ➢ The interface is structured using Flexbox and has a responsive design.

--XXX--

PROBLEM :- 2

- **Problem statement :-** Consider list of departments(deptcode and name) , list of students (roll , dept code , name , address and phone) preloaded in array . Now develop an application for the following. User may add / search / edit / delete / display all student record . While adding , ensure roll must be unique , a list of dept name to be shown from which user selects one and corresponding dept code to be stored . On collecting the data user may choose CANCEL/ SAVE button to decide course of action . For searching user provides roll . If it exists details are shown else suitable message to be displayed. To delete user provides roll . If it doesnot exist then suitable message is to be displayed . To edit also user provides roll . If it exists user may be allowed to edit any field except roll . User may select CANCEL/ SAVE to decide course of action. To display all records , at a time five records are to be shown . IT will also have PREV / NEXT button to display previous set and next set respectively . When first set is displayed PREV button must be disabled and atlast set NEXT button must be disabled .

- **Software Used :-** The software used in this project is a React application. Additionally, it seems like it's built using Vite, a fast build tool for modern web development that enhances the development experience for React applications.

- **GUI :-** The GUI is a web-based user interface built using React components. It consists of a navigation bar at the top with buttons for Display, Add, Edit, Delete, and Search pages. The main content area changes dynamically based on the selected page.

- **Component Properties** :-
  - ➢ The components (DisplayPage, AddPage, EditPage, DeletePage, and SearchPage) are functional components in React.
  - ➢ They utilize state hooks (useState) for managing local component state. The components use Recoil for state management (useRecoilState).
  - ➢ Tailwind CSS classes are applied for styling.

- **EVENT-HANDLER (PSEUDO CODE) :-**
  - ➢ App Component:
    setPage: Sets the current page when a navigation button is clicked.

    Event-handler :-
    ```
    ` function setPageHandler(page) {
      setPage(page);
    } `
    ```
  - ➢ DisplayPage Component:
    next: Increases the index n to display the next set of students.

    Event-handler :-
    ```
    ` function nextHandler() {
      setCurWindow(students.slice(n, n + 5));
      n = n + 5;
    } `
    ```

    prev: Decreases the index n to display the previous set of students.

    Event-handler :-
    ```
    ` function prevHandler() {
     n -= 10;
    setCurWindow(students.slice(n, n + 5));
    n += 5;
    } `
    ```

getDepartmentName: Retrieves the department name based on the department code.

- AddPage Component:

  changeHandler: Updates the local state (data) as the user inputs data.

  Event-handler :-

```
` function changeHandler(e) {
setData({ ...data, [e.target.name]: e.target.value });
} `
```

  clickHandler: Adds a new student to the list with a unique roll and updates the roll value.

  Event-handler :-

```
function clickHandler() {
setStudents([...students, { ...data, roll: roll }]);
setRoll(roll + 1);
alert("Student added");
}`
```

- EditPage Component:

  clickHandler: Searches for a student based on the roll number.

  Event-handler :-

```
` function clickHandler() {
const s = students.find((x) => x.roll == data);
if (!s) {
alert("No student found");
return;
}
setStudent({ ...s });
setEditedData({ ...s });
}`
```

  changeHandler: Updates the local state (editedData) as the user edits data.

  Event-handler :-

```
` function changeHandler(e) {
 setEditedData({ ...editedData, [e.target.name]: e.target.value });
}`
```

  clickHandler2: Edits the student details and updates the state.

Event-handler :-
` function clickHandler2() {
const updatedStudents = students.map((s) => s.roll === editedData.roll ? { ...editedData } : s);
setStudents([...updatedStudents]); alert("Edited");
}`

- SearchPage Component: clickHandler: Searches for a student based on the roll number.

  Event-handler :-
  ` function clickHandler() {
  const s = students.find((x) => x.roll == data); if (!s) {
  alert("No student found");
  return;
  }
  setStudent({ ...s }); }`
- DeletePage Component :
  Event-handler :-
  ` function clickHandler() {
  const s=students.find(x=>x.roll==data);if(!s){
  alert("No student found");
  return ;
  }
  let l=students.filter(s=>s.roll!=data);
  setStudents([...l]);
  alert("Deleted");
  }`

- **INTERFACE LAYOUT :-**
  - Display Page: Displays a list of students with pagination controls (Prev and Next buttons).
  - Add Page: Form to input student details (name, address, phone, and department). Add button to add a new student.
  - Edit Page: Search input for roll number. Displays student details with editable fields. Edit button to save changes.
  - Delete Page: search for roll number . delete the student for that particular roll no. If not roll number is not valid then display the alert message .
  - Search Page: Search input for roll number. Displays student details.