JavaScript_Day – 1_Week2_Hands_On -- Kaveri Harish Babu

Problem 1

Problem Statement:

Personal Notes Saver using LocalStorage (Level-1)

Scenario
You are building a simple web page where users can write daily notes and save them in their browser without using a server.

📌 **Requirements**

- **A textarea for writing notes.**
- **A Save button (using onclick).**
- **A Clear button.**
- **Notes must:**

  Be stored in localStorage

  Automatically load when the page refreshes
- **Display stored note on page load.**

🛠️ **Technical Constraints**

- **Must use:**
  - **onclick inline event**
  - **localStorage.setItem()**
  - **localStorage.getItem()**
  - **localStorage.removeItem()**
- **No backend/database.**
- **Pure HTML + JavaScript only.**
- **Data stored as key-value pair.**

🎯 Learning Outcome

You should be able to:

- **Inline event handling (onclick)**
- **Browser Storage APIs**
- **Storing & retrieving key-value data**
- **Page load event handling**
- **Basic DOM manipulation**

Source Code:

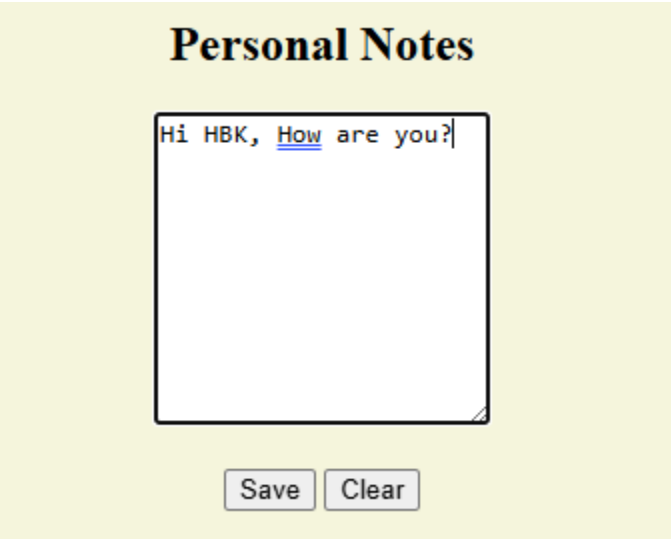
File Name: Index.html

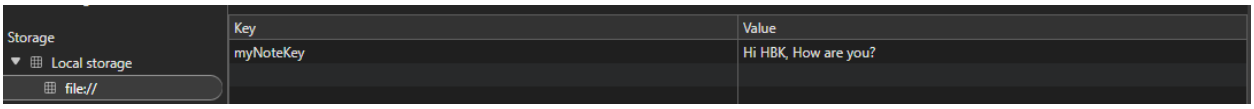
```
<!--JS_Day6_Hands_on_Problem_Statement1_HarishBabuKaveri-->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Personal Notes</title>

    <style>
        body{
            text-align: center;
            background-color: beige;
        }
    </style>
</head>
<body onload="loadSavedNote()">

    <h2>Personal Notes</h2>

    <textarea id="noteBox" rows="10" cols="20" placeholder="Write Your Notes Here"></textarea>
    <br><br>

    <button onclick="saveNote()">Save</button>

    <button onclick="clearNote()">Clear</button>

    <script>
```

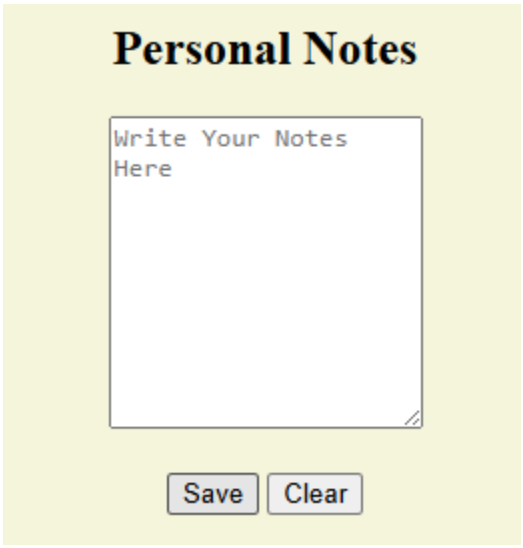
```
    <script>
    function saveNote() {
        var noteText = document.getElementById("noteBox").value.trim();

        if (noteText === "") {
            alert("Please Enter Some Text Before Saving!");
            return;
        }

        localStorage.setItem("myNoteKey", noteText);
        alert("Note Saved!");
    }

    function loadSavedNote() {
        var storedNote = localStorage.getItem("myNoteKey");
        if (storedNote !== null) {
            document.getElementById("noteBox").value = storedNote;
        }
    }

    function clearNote() {
        localStorage.removeItem("myNoteKey");
        document.getElementById("noteBox").value = "";
        alert("Note Cleared!");
    }
    </script>

</body>
</html>
```

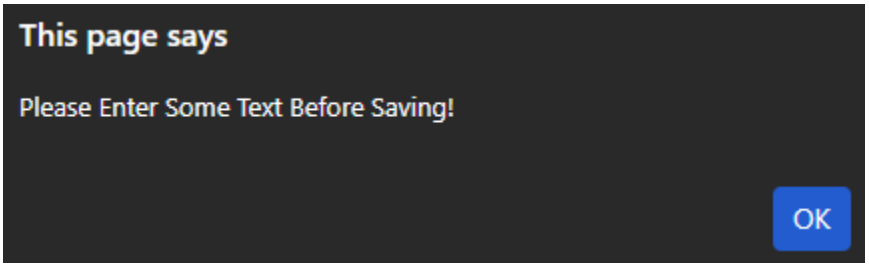
Output:

Output File: index.html









Explanation:

This code creates a simple **Personal Notes app** using a textarea and buttons. When you click **Save**, it stores your note in the browser's localStorage, so it stays even after refreshing the page. When the page loads, it automatically shows the saved note, and if you click **Clear**, it deletes the note. It also shows an alert if you try to save an empty note.

Problem 2

Problem Statement:

Live Form Validation with Events (Level-1)

Scenario

Create a simple registration form that validates user input when fields change.

📌 **Requirements**

- **Fields:**
  - **Name**
  - **Email**
  - **Age**
- **Use:**
  - **onchange**
  - **onclick**
- **Validate:**
  - **Name cannot be empty**
  - **Email must contain "@"**
  - **Age must be greater than 18**
- **Display validation message dynamically.**
- **Store valid user data in sessionStorage.**

🛠️ **Technical Constraints**

- **Must use inline onchange events.**
- **Store valid data using:**
  - **sessionStorage.setItem()**
- **No external libraries.**
- **Use basic JavaScript only.**

🎯 Learning Outcome

You will be able to:

- **onchange event usage**
- **Form validation logic**
- **Difference between localStorage and sessionStorage**
- **Dynamic DOM updates**

Source Code:

File Name: index.html

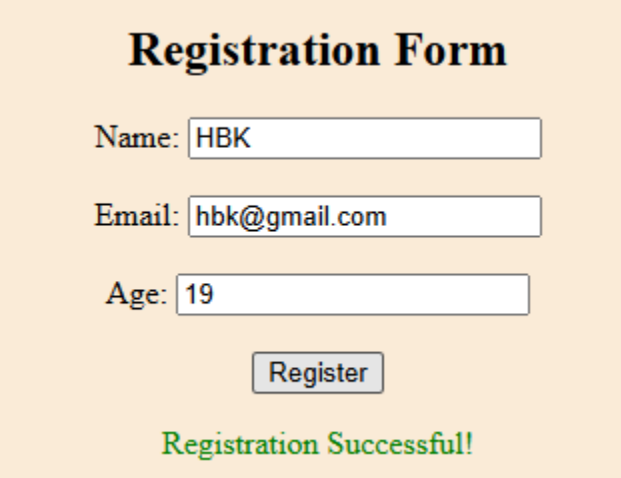
```
<!--JS_Day6_Hands_on_Problem_Statement2_HarishBabuKaveri-->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Registration Form</title>

    <style>
        body{
            text-align: center;
            background-color: antiquewhite;
        }
    </style>
</head>
<body>
    <h2>Registration Form</h2>

    Name:
    <input type="text" id="name" onchange="validateName()">
    <span id="nameMsg" style="color:red;"></span>
    <br><br>

    Email:
    <input type="text" id="email" onchange="validateEmail()">
    <span id="emailMsg" style="color:red;"></span>
    <br><br>

    Age:
    <input type="number" id="age" onchange="validateAge()">
    <span id="ageMsg" style="color:red;"></span>
    <br><br>

    <button onclick="submitForm()">Register</button>

    <p id="successMsg" style="color:green;"></p>

    <p id="WrongMsg" style="color:red;"></p>

    <script>
```

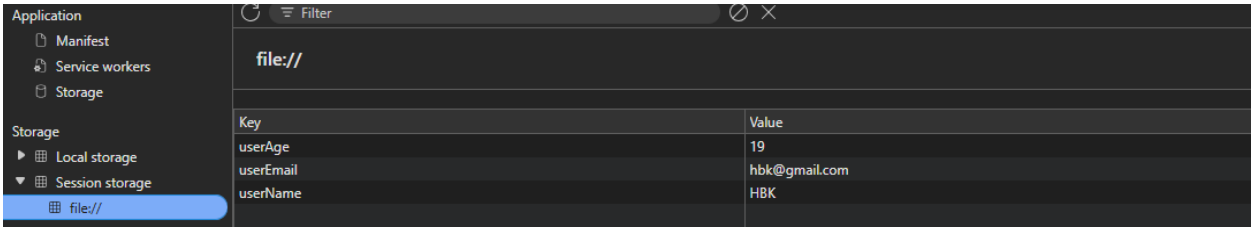
```
    <script>

    function validateName() {
        var name = document.getElementById("name").value;
        if (name === "") {
            document.getElementById("nameMsg").innerHTML = "Name cannot be empty";
            return false;
        } else {
            document.getElementById("nameMsg").innerHTML = "";
            return true;
        }
    }

    function validateEmail() {
        var email = document.getElementById("email").value;
        if (email === "") {
            document.getElementById("emailMsg").innerHTML = "Email cannot be empty";
            return false;
        } else if (email.indexOf("@") === -1) {
            document.getElementById("emailMsg").innerHTML = "Email must contain @";
            return false;
        } else {
            document.getElementById("emailMsg").innerHTML = "";
            return true;
        }
    }

    function validateAge() {
        var age = document.getElementById("age").value;
        if (age === "") {
            document.getElementById("ageMsg").innerHTML = "Age cannot be empty";
            return false;
        } else if (age <= 18) {
            document.getElementById("ageMsg").innerHTML = "Age must be greater than 18";
            return false;
        } else {
            document.getElementById("ageMsg").innerHTML = "";
            return true;
        }
    }

```

```
    }

    function submitForm() {

        var isNameValid = validateName();
        var isEmailValid = validateEmail();
        var isAgeValid = validateAge();

        if (isNameValid && isEmailValid && isAgeValid) {

            var name = document.getElementById("name").value;
            var email = document.getElementById("email").value;
            var age = document.getElementById("age").value;

            sessionStorage.setItem("userName", name);
            sessionStorage.setItem("userEmail", email);
            sessionStorage.setItem("userAge", age);

            document.getElementById("successMsg").innerHTML = "Registration Successful!";
            document.getElementById("WrongMsg").innerHTML = "";

        } else {
            document.getElementById("WrongMsg").innerHTML = "Please enter the correct details before submitting.";
            document.getElementById("successMsg").innerHTML = "";
        }
    }

</script>
</body>
</html>
```

Output:

Output File: index.html





Explanation:

This code creates a **Registration Form** with validation using JavaScript. It checks if the name, email, and age are filled correctly (email must contain "@" and age must be above 18). If everything is correct, it shows "Registration Successful!" and stores the details in sessionStorage. If not, it displays error messages in red and asks the user to enter valid details.

Problem 3

Problem Statement:

Location-Based Weather Logger (Level-2)

Scenario
Create a web application that fetches the user's geographic location and stores location history in localStorage.

📌 **Requirements**

- Button: **Get My Location**
- Use:

  navigator.geolocation.getCurrentPosition()
- Display:

  Latitude

  Longitude
- Handle:

  Permission denied

  Timeout

Location unavailable

- Save last 5 location entries in localStorage.
- Display location history on page load.

🛠️ **Technical Constraints**

- Must handle:

  Success callback

  Error callback

- Use browser permission handling.
- Store data as JSON using:

  JSON.stringify()

  JSON.parse()
- Use inline event (onclick).

🎯 **Learning Outcome**

Learners should be able to:

- Geolocation API
- Handling browser permissions
- Error handling in APIs
- Managing structured data in localStorage
- JSON parsing and stringifying

Source Code:

File Name: index.html

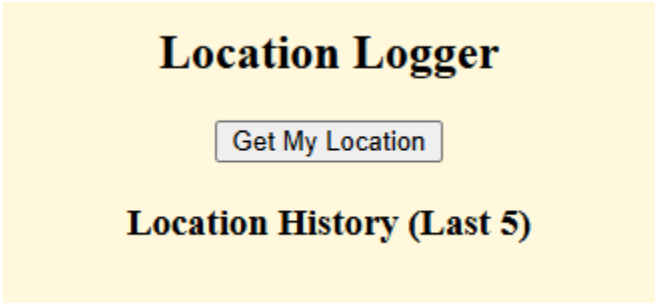
```html
<!--JS_Day6_Hands_on_Problem_Statement3_HarishBabuKaveri-->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Location-Based Weather Logger</title>

    <style>
        body{
            text-align: center;
            background-color: cornsilk;
        }
    </style>
</head>

<body onload="displayHistory()">

    <h2>Location Logger</h2>

    <button onclick="getMyLocation()">Get My Location</button>

    <p id="output"></p>
    <p id="error" style="color:red;"></p>

    <h3>Location History (Last 5)</h3>
    <p id="history"></p>

    <script>

        function getMyLocation() {

            if (navigator.geolocation) {

                navigator.geolocation.getCurrentPosition(
                    successCallback,
                    errorCallback,
                    { timeout: 8000 }
                );

            } else {
```

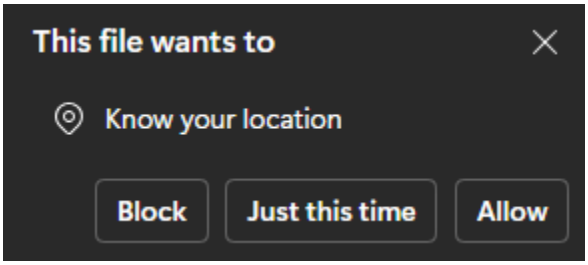
```
        } else {
            document.getElementById("error").innerHTML =
                "Geolocation not supported.";
        }
    }

    function successCallback(position) {

        var lat = position.coords.latitude;
        var lon = position.coords.longitude;

        document.getElementById("output").innerHTML =
            "Latitude: " + lat + "<br>Longitude: " + lon;

        document.getElementById("error").innerHTML = "";


        var locations = JSON.parse(localStorage.getItem("locations")) || [];

        locations.push({
            latitude: lat,
            longitude: lon,
            date: new Date().toLocaleString()
        });

        if (locations.length > 5) {
            locations = locations.slice(locations.length - 5);
        }

        localStorage.setItem("locations", JSON.stringify(locations));

        displayHistory();
    }

    function errorCallback(error) {

        var message = "";

        if (error.code === 1) {
            message = "Permission denied by user.";
        }
```

```
            message = "Permission denied by user.";
        }
        else if (error.code === 2) {
            message = "Location unavailable.";
        }
        else if (error.code === 3) {
            message = "Request timed out.";
        }
        else {
            message = "Unknown error occurred.";
        }

        document.getElementById("error").innerHTML = message;
    }

    function displayHistory() {

        var historyList = document.getElementById("history");
        historyList.innerHTML = "";

        var storedLocations = JSON.parse(localStorage.getItem("locations")) || [];

        for (var i = 0; i < storedLocations.length; i++) {

            var li = document.createElement("li");

            li.innerHTML =
                "Lat: " + storedLocations[i].latitude +
                ", Lon: " + storedLocations[i].longitude +
                " (" + storedLocations[i].date + ")";

            historyList.appendChild(li);
        }
    }
</script>

</body>
</html>
```

Output:

Output File: index.html





Location Logger

Get My Location

Latitude: 14.4108585
Longitude: 79.948925

Location History (Last 5)

Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:01:44 PM)

Location Logger

Get My Location

Latitude: 14.4108585
Longitude: 79.948925

Location History (Last 5)

Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:01:44 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:04 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:08 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:11 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:14 PM)

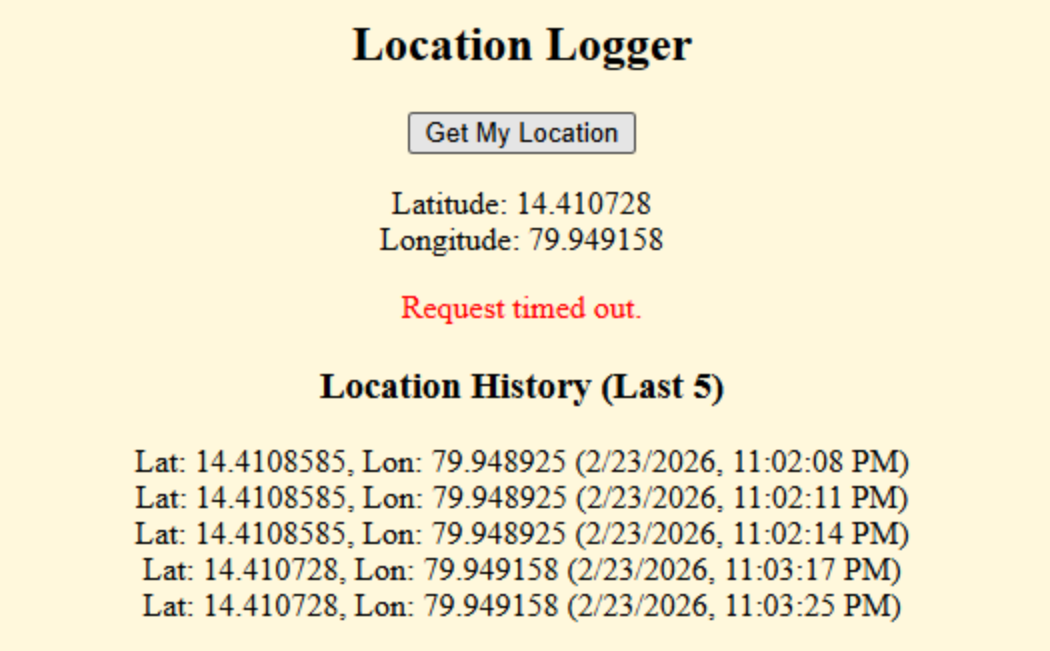
Location Logger

Get My Location

Latitude: 14.410728
Longitude: 79.949158

Location History (Last 5)

Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:08 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:11 PM)
Lat: 14.4108585, Lon: 79.948925 (2/23/2026, 11:02:14 PM)
Lat: 14.410728, Lon: 79.949158 (2/23/2026, 11:03:17 PM)
Lat: 14.410728, Lon: 79.949158 (2/23/2026, 11:03:25 PM)



Explanation:

This code creates a **Location Logger** that uses the browser's geolocation feature to get your current latitude and longitude when you click "Get My Location." It asks for permission, shows your location on the screen, and handles errors like timeout or permission denied. It also saves the last 5 locations in localStorage and displays them as history, so you can see your recent location logs even after refreshing the page.

Problem 4

Problem Statement:

Mini Expense Tracker using Client-Side Database (Level-2)

Scenario
Develop a client-side expense tracker where users can add, view, and delete expenses using a browser-supported database.

📌 **Requirements**

1. **Fields:**
    - **Expense Title**
    - **Amount**
    - **Date**
2. **Buttons:**
    - **Add Expense**
    - **View Expenses**
    - **Delete Expense**
3. **Use:**
    - **Client-side database (Web SQL or IndexedDB)**
4. **Execute SQL-like queries:**
    - **CREATE TABLE**
    - **INSERT**
    - **SELECT**
    - **DELETE**
5. **Maintain transaction handling.**
6. **Display expense list dynamically.**

🛠️ Technical Constraints

- **Must use:**
  - **Client-side DB transactions**
  - **SQL execution methods**
- **Must handle:**
  - **Transaction errors**
  - **Query errors**
- **No server/database allowed.**
- **Pure JavaScript + HTML.**

🎯 Learning Outcome

You will be able to:

- **Client-side database concepts**
- **Executing SQL queries in browser**
- **Managing transactions**
- **Advanced DOM rendering**
- **Persistent structured storage**

Source Code:

File Name: index.html

```html
<!--JS_Day6_Hands_on_Problem_Statement4_HarishBabuKaveri-->
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mini Expense Tracker</title>
    <style>
        body {
            text-align: center;
            background-color: gainsboro;
            font-family: Arial;
            padding: 20px;
        }
        input {
            margin: 5px;
        }
        button {
            margin: 5px;
        }
        table {
            border-collapse: collapse;
            margin: 20px auto;
        }
        th, td {
            border: 1px solid black;
            padding: 8px;
        }
    </style>
</head>
<body>

<h2>Mini Expense Tracker</h2>

<!-- Input Fields -->
<input type="text" id="title" placeholder="Expense Title">
<input type="number" id="amount" placeholder="Amount">
<input type="date" id="date">

<br>

```

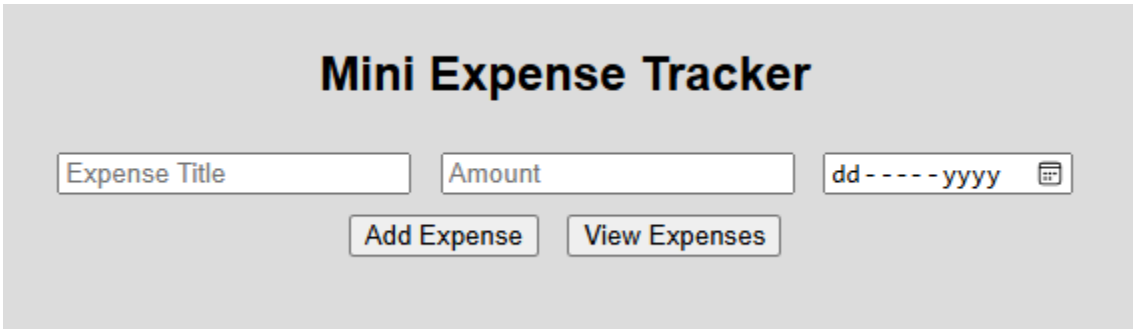
```
41
42  <!-- Buttons -->
43  <button onclick="addExpense()">Add Expense</button>
44  <button onclick="viewExpenses()">View Expenses</button>
45
46  <!-- Expense List -->
47  <div id="expenseList"></div>
48
49  <script>
50  let db;
51
52  //CREATE DATABASE (Open DB)
53  let request = indexedDB.open("ExpenseDB", 1);
54
55  request.onupgradeneeded = function(event) {
56      db = event.target.result;
57
58      //CREATE TABLE → objectStore
59      if (!db.objectStoreNames.contains("expenses")) {
60          db.createObjectStore("expenses", { keyPath: "id", autoIncrement: true });
61          console.log("ObjectStore Created");
62      }
63  };
64
65  request.onsuccess = function(event) {
66      db = event.target.result;
67      console.log("Database opened successfully");
68  };
69
70  request.onerror = function(event) {
71      console.error("Database error:", event.target.error);
72  };
73
74  //INSERT → add()
75  function addExpense() {
76      let title = document.getElementById("title").value;
77      let amount = document.getElementById("amount").value;
78      let date = document.getElementById("date").value;
79
80      let transaction = db.transaction(["expenses"], "readwrite");
81
```

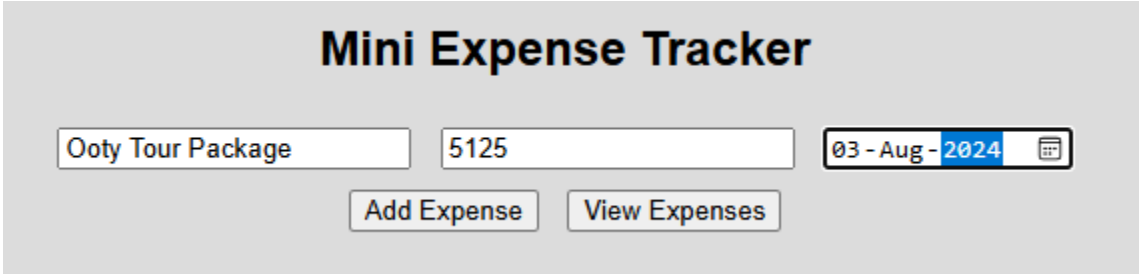
```
81
82      transaction.onerror = function() {
83          alert("Transaction failed!");
84      };
85
86      let store = transaction.objectStore("expenses");
87
88      let expense = { title, amount, date };
89
90      let addRequest = store.add(expense);
91
92      addRequest.onsuccess = function() {
93          alert("Expense Added Successfully!");
94      };
95
96      addRequest.onerror = function() {
97          alert("Error Adding Expense!");
98      };
99  }
100
101 //SELECT → getAll()
102 function viewExpenses() {
103     let transaction = db.transaction(["expenses"], "readonly");
104
105     let store = transaction.objectStore("expenses");
106
107     let getRequest = store.getAll();
108
109     getRequest.onsuccess = function() {
110         let data = getRequest.result;
111         displayExpenses(data);
112     };
113
114     getRequest.onerror = function() {
115         alert("Error Fetching Data!");
116     };
117 }
118
119 //DELETE → delete()
120 function deleteExpense(id) {
121     let transaction = db.transaction(["expenses"], "readwrite");
```

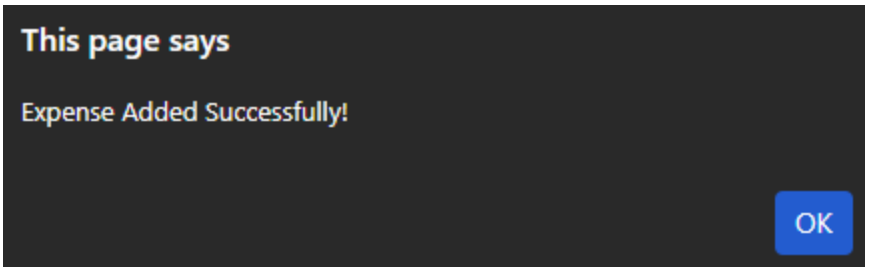
```
function deleteExpense(id) {
    let transaction = db.transaction(["expenses"], "readwrite");

    let store = transaction.objectStore("expenses");

    let deleteRequest = store.delete(id);

    deleteRequest.onsuccess = function() {
        alert("Expense Deleted!");
        viewExpenses();
    };

    deleteRequest.onerror = function() {
        alert("Delete Failed!");
    };
}

//To Display the Content
function displayExpenses(expenses) {
    let output = "<table>";
    output += "<tr><th>Title</th><th>Amount</th><th>Date</th><th>Action</th></tr>";

    expenses.forEach(exp => {
        output += `
            <tr>
                <td>${exp.title}</td>
                <td>${"₹"+exp.amount}</td>
                <td>${exp.date}</td>
                <td><button onclick="deleteExpense(${exp.id})">Delete</button></td>
            </tr>
        `;
    });

    output += "</table>";

    document.getElementById("expenseList").innerHTML = output;
}
</script>
</body>
</html>
```

Output:

Output File: index.html





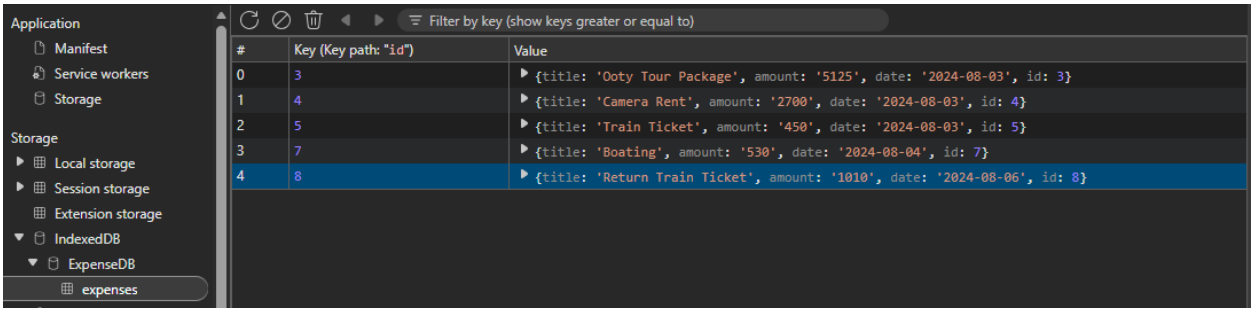


Mini Expense Tracker

Return Train Ticket | 1010 | 06 - Aug - 2024

Add Expense | View Expenses

Title	Amount	Date	Action
Ooty Tour Package	₹5125	2024-08-03	Delete
Camera Rent	₹2700	2024-08-03	Delete
Train Ticket	₹450	2024-08-03	Delete
Boating	₹530	2024-08-04	Delete
Return Train Ticket	₹1010	2024-08-06	Delete



Explanation:

This code creates a **Mini Expense Tracker** using IndexedDB in the browser. You can add an expense (title, amount, date), store it in a database, view all expenses in a table, and delete any record. The output shows your entered expenses neatly in a table, and the data is saved in the browser's IndexedDB, so it stays even after refreshing the page.