Javascript _Day - 6 Hands On _Sana Naveen

Problem 1

Problem Statement :
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.

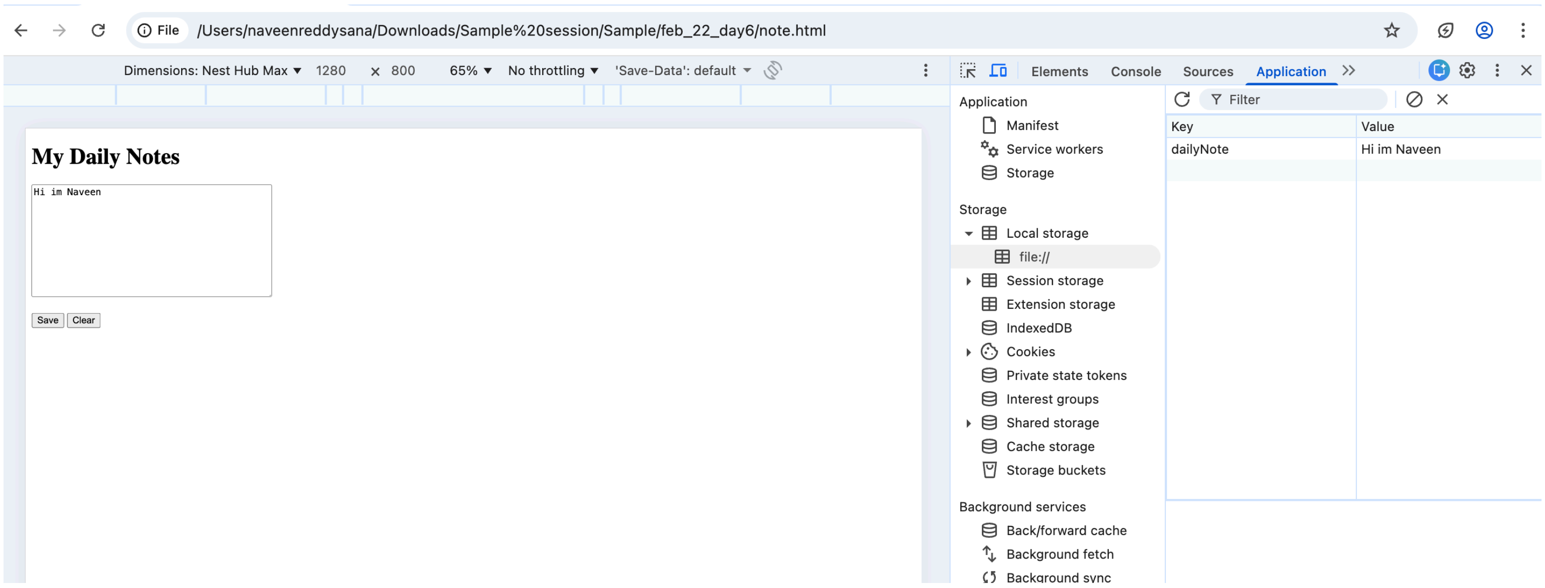
Technical Constraints
- Must use:
    - onclick inline event
    - localStorage.setItem()
    - localStorage.getItem()
    - localStorage.removeItem()
- No backend/database.
- Pure HTML + JavaScript only.
- Data stored as key-value pair.

Code :

```
<!DOCTYPE html>
<html>
<head>
   <title>Personal Notes Saver</title>
</head>
<body onload="loadNote()">

   <h1>My Daily Notes</h1>

   <!-- Textarea -->
   <textarea id="noteArea" rows="10" cols="40" placeholder="Write your note here..."></
textarea>
   <br><br>

   <!-- Save Button (inline onclick) -->
   <button onclick="saveNote()">Save</button>

   <!-- Clear Button -->
   <button onclick="clearNote()">Clear</button>

   <script>

      // Save note to localStorage
      function saveNote() {
         let note = document.getElementById("noteArea").value;
         localStorage.setItem("dailyNote", note);
         alert("Note Saved!");
```

```
        }

        // Load note when page refreshes
        function loadNote() {
            let savedNote = localStorage.getItem("dailyNote");

            if (savedNote !== null) {
                document.getElementById("noteArea").value = savedNote;
            }
        }

        // Clear note
        function clearNote() {
            localStorage.removeItem("dailyNote");
            document.getElementById("noteArea").value = "";
            alert("Note Cleared!");
        }

    </script>

</body>
</html>
```

Output Screenshot:


File /Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_22_day6/note.html

Dimensions: Nest Hub Max 1280 x 800 65% No throttling 'Save-Data': default

My Daily Notes

```
Hi im Naveen
```

Key	Value
dailyNote	Hi im Naveen

Explanation :

This application uses inline event handling (onclick) to trigger JavaScript functions for saving and clearing notes. The saveNote() function retrieves the textarea value using getElementById() and stores it in localStorage as a key-value pair using localStorage.setItem(). The loadNote() function runs on page load (onload) and retrieves stored data using localStorage.getItem() to repopulate the textarea. The clearNote() function removes the stored data using localStorage.removeItem() and resets the DOM element value. The implementation demonstrates DOM manipulation and browser storage API usage without any backend.

Problem 2:

Problem Statement :

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:

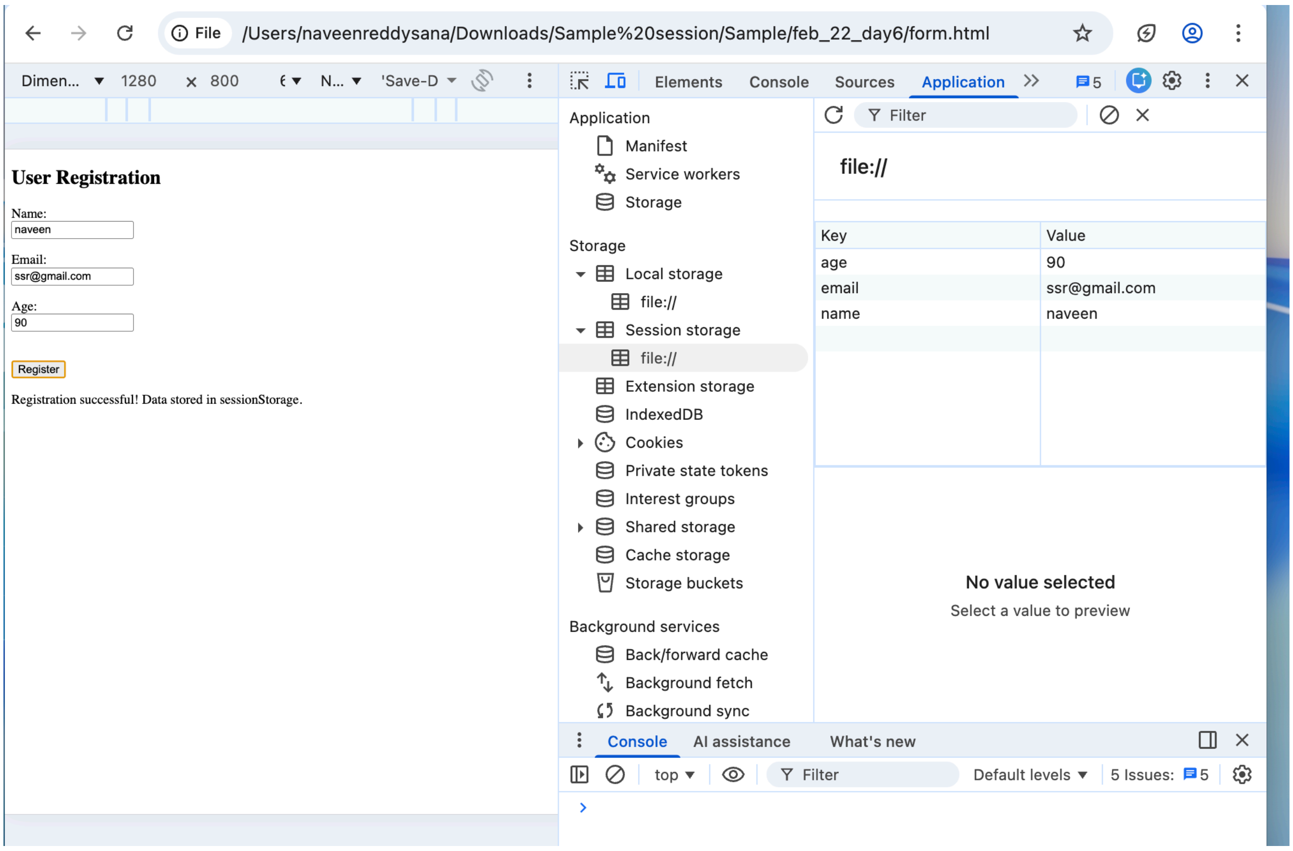
- onchange event usage
- Form validation logic
- Difference between localStorage and sessionStorage
- Dynamic DOM updates

Code :

```
<!DOCTYPE html>
<html>
<head>
   <title>Registration Form</title>
</head>
<body>

<h2>User Registration</h2>

<form>

   <!-- Name -->
   <label>Name:</label><br>
   <input type="text" id="name" onchange="validateName()">
   <p id="nameMsg"></p>

   <!-- Email -->
   <label>Email:</label><br>
   <input type="email" id="email" onchange="validateEmail()">
```

```
    <p id="emailMsg"></p>

    <!-- Age -->
    <label>Age:</label><br>
    <input type="number" id="age" onchange="validateAge()">
    <p id="ageMsg"></p>

    <br>
    <button type="button" onclick="saveData()">Register</button>

    <p id="finalMsg"></p>

</form>

<script>

    function validateName() {
        let 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() {
        let email = document.getElementById("email").value;

        if (!email.includes("@")) {
            document.getElementById("emailMsg").innerHTML = "Email must contain @";
            return false;
        } else {
            document.getElementById("emailMsg").innerHTML = "";
            return true;
        }
    }

    function validateAge() {
        let age = document.getElementById("age").value;

        if (age <= 18) {
            document.getElementById("ageMsg").innerHTML = "Age must be greater than 18";
            return false;
        } else {
            document.getElementById("ageMsg").innerHTML = "";
            return true;
        }
    }

    function saveData() {

        if (validateName() && validateEmail() && validateAge()) {

            sessionStorage.setItem("name", document.getElementById("name").value);
            sessionStorage.setItem("email", document.getElementById("email").value);
            sessionStorage.setItem("age", document.getElementById("age").value);
            document.getElementById("finalMsg").innerHTML =
               "Registration successful! Data stored in sessionStorage.";
         } else {
            document.getElementById("finalMsg").innerHTML =
               "Please fix validation errors.";
         }
      }

</script>

</body>
</html>
```

Output Screenshot :



/Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_22_day6/form.html

User Registration

Name: naveen

Email: ssr@gmail.com

Age: 90

Register

Registration successful! Data stored in sessionStorage.

file://

Key	Value
age	90
email	ssr@gmail.com
name	naveen

No value selected

Select a value to preview

Explanation :

This registration form uses inline onchange events to validate user input dynamically whenever a field value changes. JavaScript functions check that the name is not empty, the email contains "@", and the age is greater than 18. Validation messages are displayed in real time using DOM manipulation (innerHTML). When the Register button (using onclick) is clicked and all validations pass, the user data is stored in the browser using sessionStorage.setItem() as key-value pairs for the current session.

Problem 3:

Problem Statement :

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:

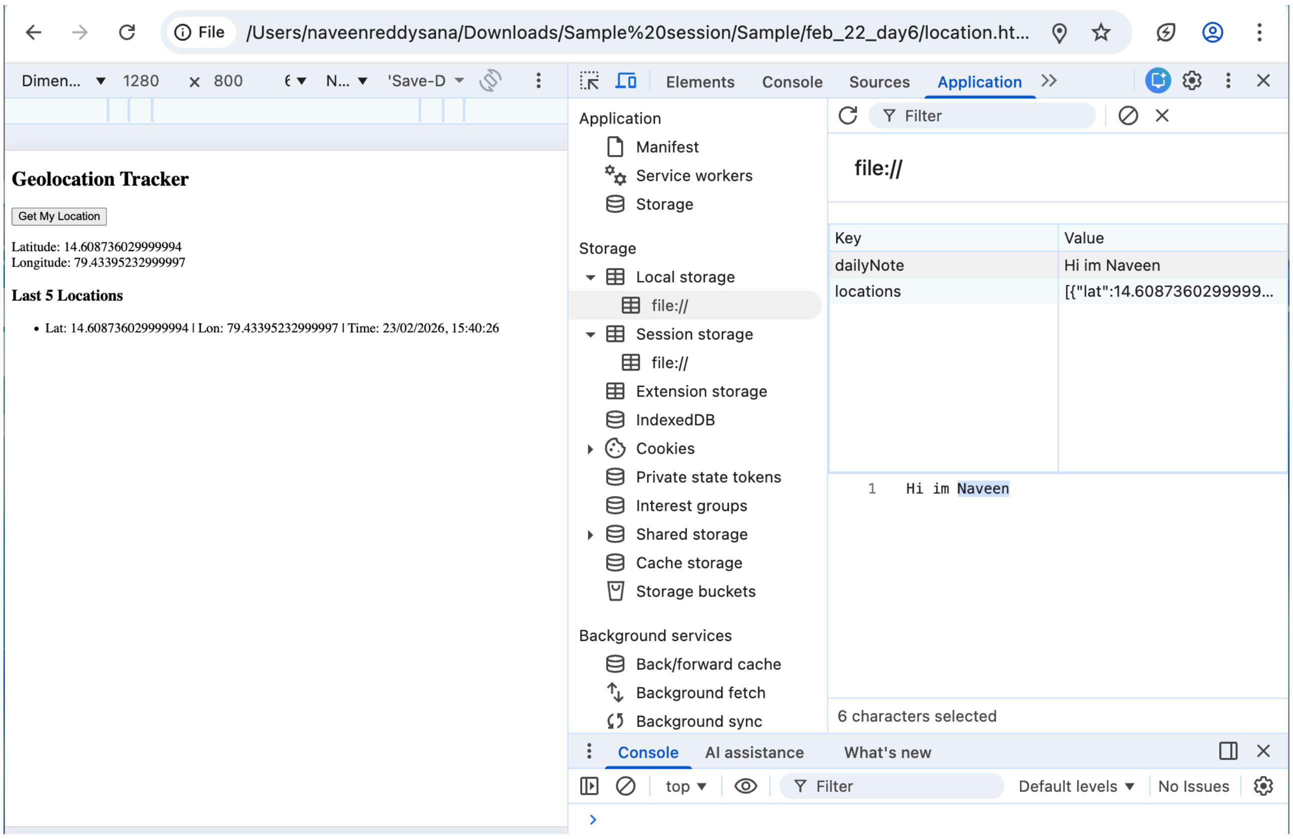
JSON.stringify()
JSON.parse()

- Use inline event (onclick).

Code :

```
<!DOCTYPE html>
<html>
<head>
  <title>Location Tracker</title>
</head>
<body onload="loadHistory()">

  <h2>Geolocation Tracker</h2>

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

  <p id="output"></p>

  <h3>Last 5 Locations</h3>
  <ul id="historyList"></ul>

<script>

function getLocation() {
  if (!navigator.geolocation) {
      document.getElementById("output").innerHTML = "Geolocation not supported.";
      return;
  }

  navigator.geolocation.getCurrentPosition(successCallback, errorCallback, {
      timeout: 10000
  });
}

function successCallback(position) {

  let latitude = position.coords.latitude;
  let longitude = position.coords.longitude;

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

  let locationData = {
      lat: latitude,
      lon: longitude,
      time: new Date().toLocaleString()
  };

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

  history.unshift(locationData);   // Add new location at start

  if (history.length > 5) {
      history.pop();  // Keep only last 5
  }

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

  loadHistory();
}

function errorCallback(error) {

  switch (error.code) {
      case error.PERMISSION_DENIED:
          document.getElementById("output").innerHTML = "Permission denied.";
          break;
      case error.POSITION_UNAVAILABLE:
          document.getElementById("output").innerHTML = "Location unavailable.";
          break;
      case error.TIMEOUT:
          document.getElementById("output").innerHTML = "Request timed out.";
          break;
      default:
          document.getElementById("output").innerHTML = "Unknown error occurred.";
  }
}

function loadHistory() {

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

  let list = document.getElementById("historyList");
  list.innerHTML = "";

  history.forEach(function(loc) {
      let li = document.createElement("li");
      li.innerHTML = "Lat: " + loc.lat +
                     " | Lon: " + loc.lon +
                     " | Time: " + loc.time;
      list.appendChild(li);
  });
}

</script>

</body>
</html>
```

Code Screenshots :



Geolocation Tracker

Latitude: 14.608736029999994
Longitude: 79.43395232999997

Last 5 Locations

- Lat: 14.608736029999994 | Lon: 79.43395232999997 | Time: 23/02/2026, 15:40:26

file://

Key	Value
dailyNote	Hi im Naveen
locations	[{"lat":14.6087360299999...

```
Hi im Naveen
```

6 characters selected

Explanation :
This web application uses the Geolocation API (navigator.geolocation.getCurrentPosition()) to retrieve the user’s latitude and longitude through success and error callback functions. The retrieved location data is stored as a JavaScript object and saved in localStorage as a JSON string using JSON.stringify(). On page load, the stored data is retrieved using JSON.parse() and displayed dynamically in the DOM. The application maintains only the last five location records and properly handles permission denied, timeout, and location unavailable errors.

Problem 4:

Problem Statement :
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

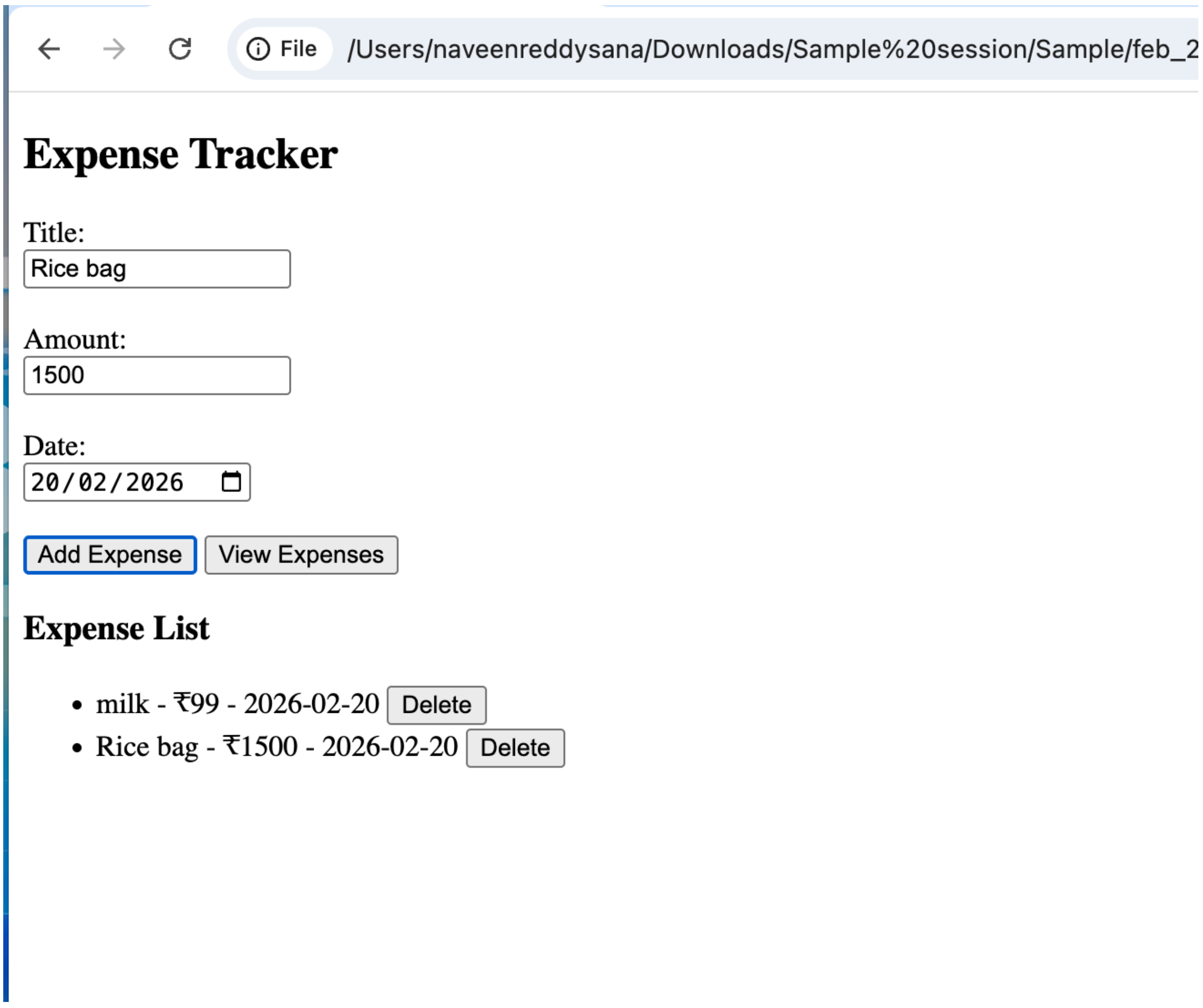
- Must use:
  - Client-side DB transactions
  - SQL execution methods
- Must handle:
  - Transaction errors
  - Query errors
- No server/database allowed.
- Pure JavaScript + HTML.

Code :

```
<!DOCTYPE html>
<html>
<head>
   <title>Expense Tracker - IndexedDB</title>
</head>
<body>

<h2>Expense Tracker</h2>

<label>Title:</label><br>
<input type="text" id="title"><br><br>

<label>Amount:</label><br>
<input type="number" id="amount"><br><br>

<label>Date:</label><br>
<input type="date" id="date"><br><br>

<button onclick="addExpense()">Add Expense</button>
<button onclick="viewExpenses()">View Expenses</button>

<h3>Expense List</h3>
<ul id="expenseList"></ul>
```

```
<script>

let db;

// Open / Create Database
let request = indexedDB.open("ExpenseDB", 1);

request.onupgradeneeded = function(event) {
    db = event.target.result;

    // CREATE TABLE equivalent
    if (!db.objectStoreNames.contains("expenses")) {
        db.createObjectStore("expenses", { keyPath: "id", autoIncrement: true });
    }
};

request.onsuccess = function(event) {
    db = event.target.result;
};

request.onerror = function() {
    alert("Database error");
};

// INSERT → add()
function addExpense() {

    let transaction = db.transaction(["expenses"], "readwrite");
    let store = transaction.objectStore("expenses");

    let expense = {
        title: document.getElementById("title").value,
        amount: document.getElementById("amount").value,
        date: document.getElementById("date").value
    };

    let addRequest = store.add(expense);

    addRequest.onsuccess = function() {
        alert("Expense Added");
        viewExpenses();
    };

    addRequest.onerror = function() {
        alert("Insert error");
    };

    transaction.onerror = function() {
        alert("Transaction error");
    };
}

// SELECT → getAll()
function viewExpenses() {

    let transaction = db.transaction(["expenses"], "readonly");
    let store = transaction.objectStore("expenses");

    let getRequest = store.getAll();
    getRequest.onsuccess = function() {

        let list = document.getElementById("expenseList");
        list.innerHTML = "";

        getRequest.result.forEach(function(item) {

            let li = document.createElement("li");
            li.innerHTML =
                item.title + " - ₹" + item.amount + " - " + item.date +
                " <button onclick='deleteExpense(" + item.id + ")'>Delete</button>";

            list.appendChild(li);
        });
    };

    getRequest.onerror = function() {
        alert("Query error");
    };
}

// DELETE → delete()
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 error");
    };

    transaction.onerror = function() {
        alert("Transaction error");
    };
}

</script>

</body>
</html>
```

Output Screenshot :



/Users/naveenreddysana/Downloads/Sample%20session/Sample/feb_2

Expense Tracker

Title: Rice bag

Amount: 1500

Date: 20/02/2026

Expense List

- milk - ₹99 - 2026-02-20 Delete
- Rice bag - ₹1500 - 2026-02-20 Delete

Explanation :
This expense tracker uses IndexedDB, a browser-supported client-side database, to store expense records. The database is initialized using indexedDB.open() and an object store is created in onupgradeneeded (equivalent to SQL CREATE TABLE). Data insertion is performed using add() (INSERT), retrieval using getAll() (SELECT), and deletion using delete() (DELETE). All operations are executed within transactions (readwrite or readonly) to ensure data consistency. Transaction and request error handling are implemented to manage database failures. The expense list is dynamically updated in the DOM.