DAY-1_Events_HTML5_API_ Hands-On_Uday

- **1)Problem Statement : Personal Notes Saver using LocalStorage**

- **HTML CODE :**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Personal Note Page</title>

  <style>

    body{
      font-family: Arial;
      background:#f0fff4;
    }

    .box{
      width:600px;
      margin:auto;
      margin-top:40px;
      background:white;
      padding:20px;
      border-radius:10px;
      box-shadow:0 0 10px gray;
    }

    h2{
      color:#1e8449;
    }

    textarea{
      width:100%;
      border:2px solid #27ae60;
      border-radius:5px;
      padding:10px;
    }

    button{
      background:#27ae60;
      color:white;
      border:none;
      padding:8px 15px;
      border-radius:5px;
      cursor:pointer;
    }

    button:hover{
      background:#1e8449;
    }
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

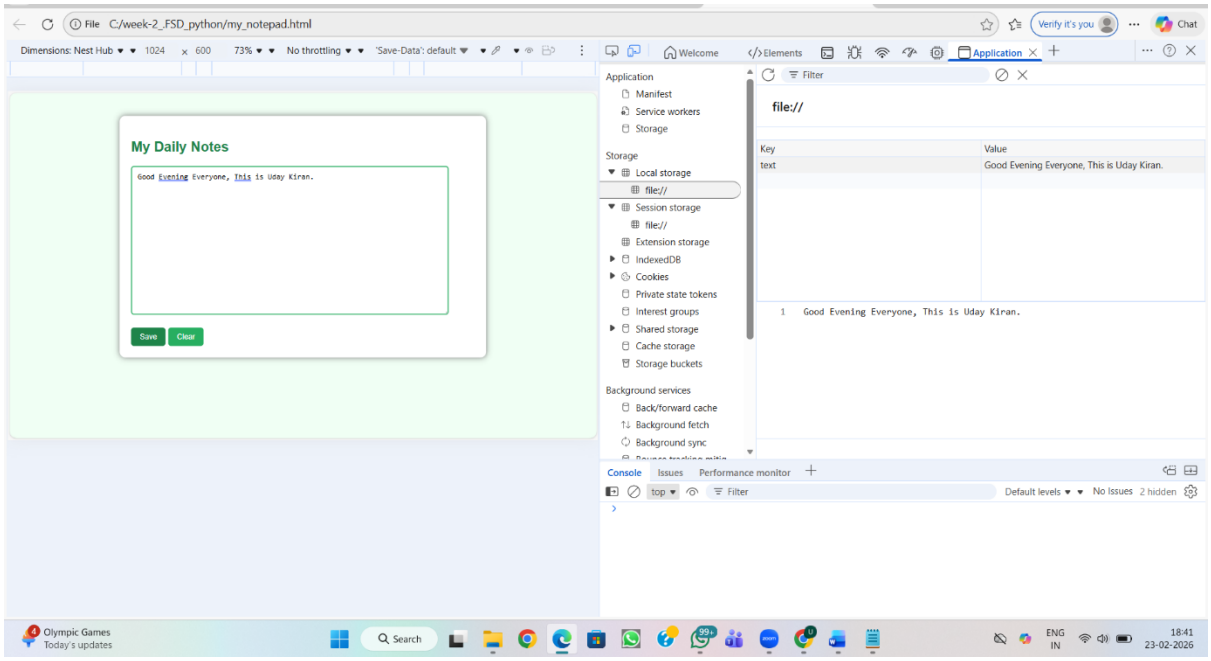
```
  </style>

</head>
<body onload="loadNote()">

<div class="box">

  <h2>My Daily Notes</h2>

  <textarea id="note" rows="10" coloumns="60" placeholder="Write your text
Here"></textarea>
  <br><br>

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

</div>

<script>

function saveNote()
{
  let data = document.getElementById("note").value;

  if(data.trim() === "")
  {
    alert("kindly write something before saving");
    return;
  }

  localStorage.setItem("text", data);
  alert("Note is saved successfully ");
}

function loadNote()
{
  let savedData = localStorage.getItem("text");

  if(savedData !== null)
  {
    document.getElementById("note").value = savedData;
  }
}

function clearNote()
{
  localStorage.removeItem("text");
  document.getElementById("note").value = "";
  alert("Note cleared");
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
}

</script>

</body>
</html>
```

Technical Explanation : I developed a simple client-side note storage application using HTML and vanilla JavaScript that utilizes the browser's localStorage API to persist data. A <textarea> element is used to capture user input, allowing users to type and edit their notes. When the Save button is clicked, the note is stored in the browser using localStorage.setItem() as a key–value pair, ensuring the data remains available even after refreshing the page. On page load, the application automatically retrieves the saved note using localStorage.getItem() and displays it in the textarea. The Clear button removes the stored note using localStorage.removeItem() and resets the user interface. All DOM manipulation is handled using document.getElementById() to access and update elements dynamically.

- **OUTPUT:**



2) Problem Statement: Live Form Validation with Events

- **HTML CODE :**

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

  <style>
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
body{
   font-family: Arial;
   background-color: #f2f2f2;
}

h2{
   color: #2c3e50;
}

.box{
   background: white;
   padding: 20px;
   width: 320px;
   border-radius: 10px;
   box-shadow: 0 0 10px gray;
}

label{
   display: inline-block;
   width: 80px;
   font-weight: bold;
}

.row{
   margin-bottom: 12px;
}

input{
   padding: 5px;
}

.success{
   color: green;
   font-weight: bold;
}

.error{
   color: red;
   font-weight: bold;
}

button{
   background: #27ae60;
   color: white;
   border: none;
   padding: 8px 15px;
   border-radius: 5px;
   cursor: pointer;
}
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
      button:hover{
        background: #219150;
      }

    </style>

</head>
<body>

<center>

<div class="box">

<h2>Registration</h2>

<div class="row">
  <label>Name :</label>
  <input type="text" id="name" onchange="checkName()">
  <span id="nameMsg"></span>
</div>

<div class="row">
  <label>Email :</label>
  <input type="text" id="email" onchange="checkEmail()">
  <span id="emailMsg"></span>
</div>

<div class="row">
  <label>Age :</label>
  <input type="number" id="age" onchange="checkAge()">
  <span id="ageMsg"></span>
</div>

<br>

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

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

</div>

</center>

<script>

function checkName()
{
  let n = document.getElementById("name").value;
  let namePattern = /^[A-Za-z ]+$/;
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
  if(n === "")
  {
    alert("Name cannot be empty");
    nameMsg.innerHTML = "Required";
    nameMsg.className = "error";
  }
  else if(!namePattern.test(n))
  {
    alert("Name must contain only letters");
    nameMsg.innerHTML = "Only letters allowed";
    nameMsg.className = "error";
  }
  else
  {
    nameMsg.innerHTML = "OK";
    nameMsg.className = "success";
  }
}

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

  if(!e.includes("@"))
  {
    alert("Enter valid email");
    emailMsg.innerHTML = "Invalid";
    emailMsg.className = "error";
  }
  else
  {
    emailMsg.innerHTML = "OK";
    emailMsg.className = "success";
  }
}

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

  if(a <= 18 || a === "")
  {
    alert("Age must be greater than 18");
    ageMsg.innerHTML = "Not allowed";
    ageMsg.className = "error";
  }
  else
  {
    ageMsg.innerHTML = "OK";
```

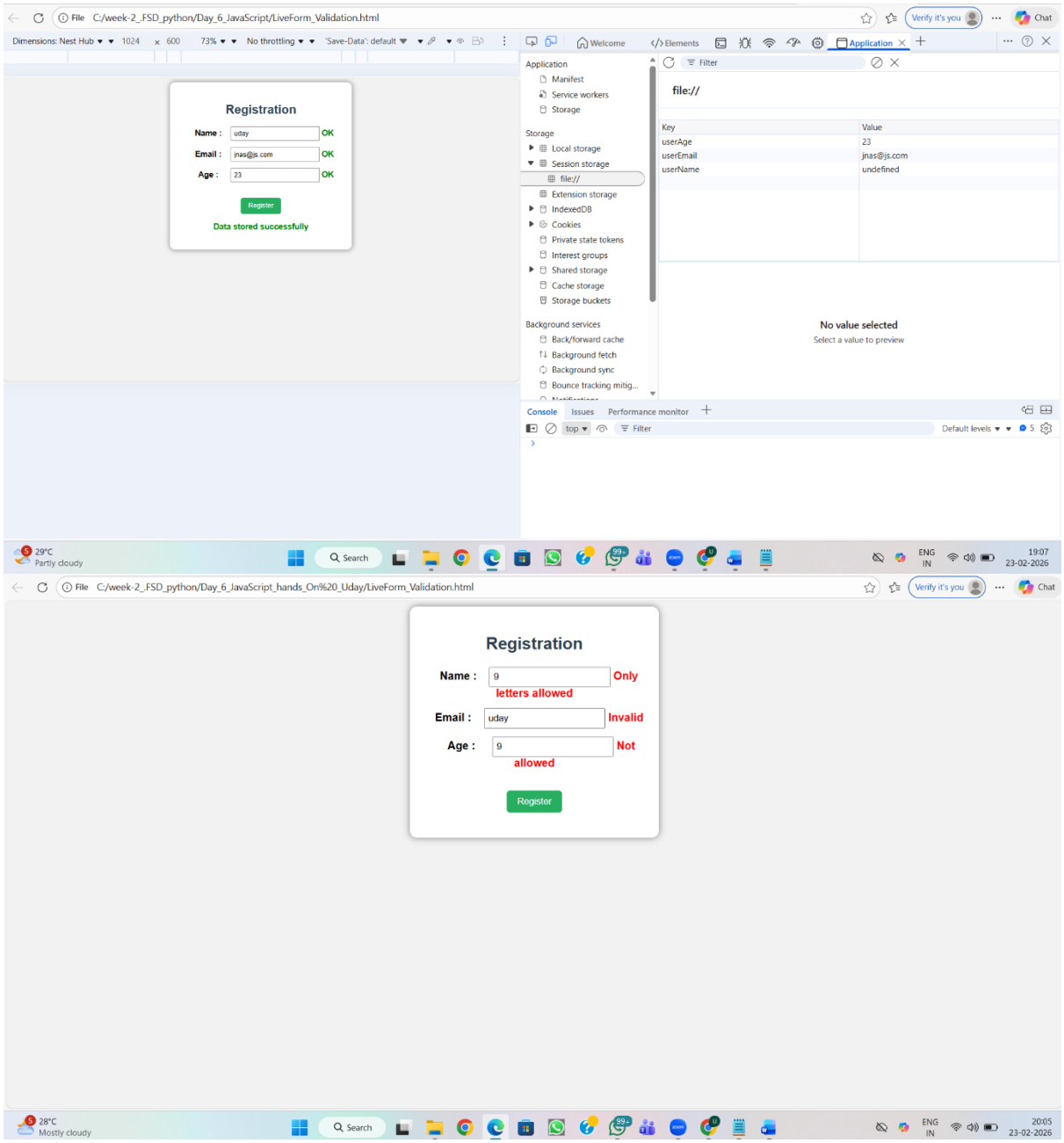
```
    ageMsg.className = "success";
  }
}

function saveData()
{
  let n = name.value;
  let e = email.value;
  let a = age.value;

  if(n === "" || e === "" || a === "")
  {
    alert("All fields are required");
    return;
  }

  sessionStorage.setItem("userName", n);
  sessionStorage.setItem("userEmail", e);
  sessionStorage.setItem("userAge", a);

  finalMsg.innerHTML = " Data stored successfully";
  finalMsg.className = "success";
}

</script>

</body>
</html>
```

- **Technical Explanation :**

  I developed a client-side registration form using HTML and basic JavaScript that validates user inputs for Name, Email, and Age through inline onchange events. The validation rules ensure that the Name field is not empty, the Email field contains an “@” symbol, and the Age entered is greater than 18. Validation messages are displayed and updated dynamically using DOM manipulation to provide immediate feedback to the user. On clicking the submit button (onclick), if all fields meet the validation criteria, the valid data is stored in the browser using sessionStorage.setItem(). The entire implementation is built using only basic JavaScript without any external libraries.

- **OUTPUT:**

DAY-1_Events_HTML5_API_ Hands-On_Uday



DAY-1_Events_HTML5_API_ Hands-On_Uday

3) Problem Statement : Location-Based Weather Logger

- **HTML CODE :**

```
<!DOCTYPE html>
<html>
<head>
  <title>My Geo Location Logger</title>

  <style>
    body{
      font-family: Arial;
      background:linear-gradient(to right,#00c6ff,#0072ff);
      text-align:center;
      margin-top:40px;
      color:white;
    }

    .box{
      background:white;
      color:black;
      width:340px;
      margin:auto;
      padding:20px;
      border-radius:15px;
      box-shadow:0 0 15px rgba(0,0,0,0.3);
    }

    button{
      padding:8px 15px;
      border:none;
      background:#0072ff;
      color:white;
      border-radius:6px;
      cursor:pointer;
    }

    button:hover{
      background:#0056d6;
    }

    ul{
      text-align:left;
    }
  </style>
</head>

<body onload="loadHistory()">

<div class="box">

<h3>Location Logger</h3>
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

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

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

<h4>Last 5 Locations</h4>
<ul id="history"></ul>

</div>

<script>

let output = document.getElementById("output");
let historyList = document.getElementById("history");

function getLocation()
{
  if(navigator.geolocation)
  {
    navigator.geolocation.getCurrentPosition(showPosition, showError);
  }
  else
  {
    output.innerHTML = "Geolocation not supported";
  }
}

function showPosition(position)
{
  let lat = position.coords.latitude;
  let lon = position.coords.longitude;

  output.innerHTML = "Latitude : " + lat + "<br>Longitude : " + lon;

  saveLocation(lat, lon);
function showError(error)
{
  if(error.code == 1)
    output.innerHTML = " Permission denied";
  else if(error.code == 2)
    output.innerHTML = " Location unavailable";
  else if(error.code == 3)
    output.innerHTML = " Timeout";
}

function saveLocation(lat, lon)
{
  let locations = JSON.parse(localStorage.getItem("locations")) || [];

  let newEntry = {
```

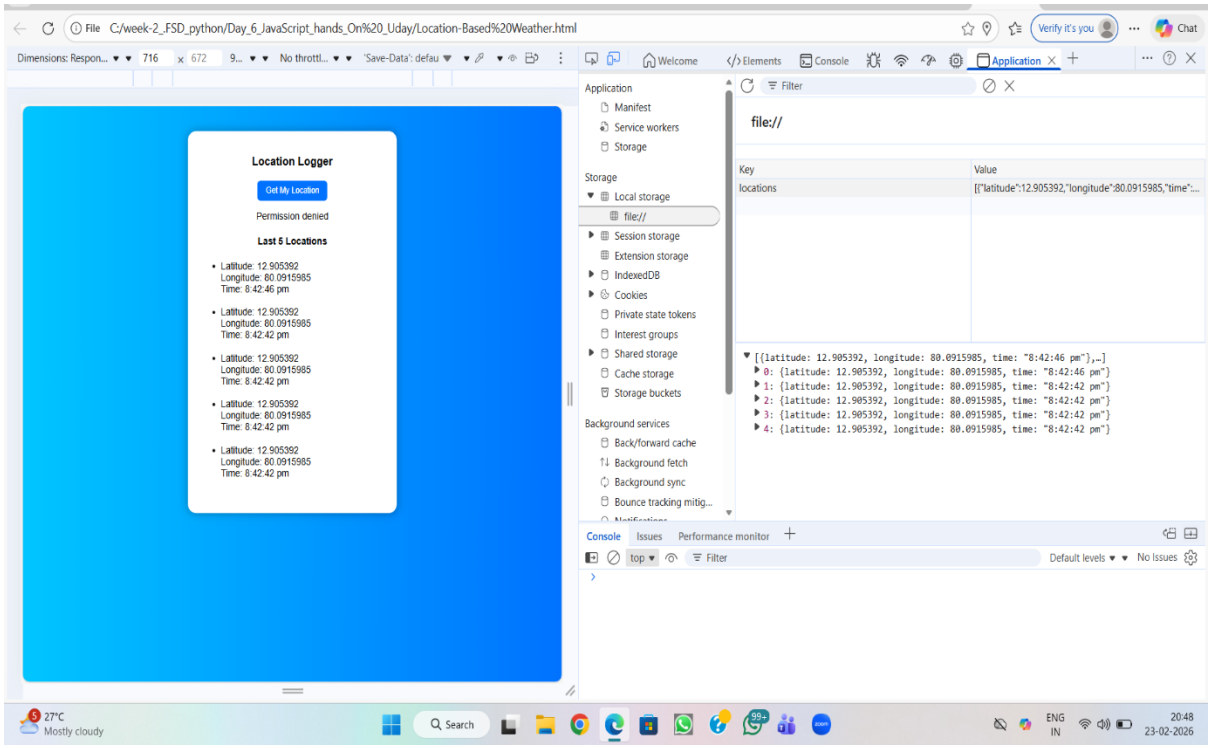
```
      latitude : lat,
      longitude : lon,
      time : new Date().toLocaleTimeString()
    };

    locations.unshift(newEntry);

    //  Keep only last 5
    if(locations.length > 5)
    {
      locations = locations.slice(0,5);
    }

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

    loadHistory();
}

function loadHistory()
{
    let locations = JSON.parse(localStorage.getItem("locations")) || [];

    historyList.innerHTML = "";

    locations.forEach(function(loc)
    {
      let li = document.createElement("li");

      li.innerHTML =
      "latitude " + loc.latitude + " , longitude " + loc.longitude +
      "<br>time " + loc.time;

      historyList.appendChild(li);
    });
}

</script>

</body>
</html>
```

- **Technical Explanation :**

  I developed a client-side geolocation web application using HTML and vanilla JavaScript that retrieves the user's latitude and longitude through navigator.geolocation.getCurrentPosition() when a button is clicked. The application uses a success callback function to display the user's current location on the page and store the data in localStorage. An error callback function is implemented to handle scenarios such as permission denial, request timeout, or location unavailability. The location details are stored as an array of objects by converting them using JSON.stringify() while saving and JSON.parse() while retrieving. The application maintains only the last five location records by updating the array before storing it again. On page load,

the saved location history is automatically retrieved from localStorage and displayed dynamically using DOM manipulation.

OUTPUT:



4) Problem Statement : : Mini Expense Tracker using Client-Side Database

- **HTML CODE :**

```
<!DOCTYPE html>
<html>
<head>
  <title>Mini Expense Tracker</title>

  <style>
    body{font-family:Arial;background:#f2f2f2;}
    .box{
      background:white;
      width:420px;
      margin:auto;
      margin-top:30px;
      padding:20px;
      border-radius:10px;
      box-shadow:0 0 10px gray;
    }
    input,button{margin:5px;padding:6px;}
    button{background:#2ecc71;color:white;border:none;border-radius:4px;}
    button:hover{background:#27ae60;}
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
    li{margin:6px 0;}
  </style>

</head>
<body onload="openDB()">

<div class="box">

<h3>Expense Tracker (Indexed_DB)</h3>

<input type="text" id="title" placeholder="Expense Title"><br>
<input type="number" id="amount" placeholder="Amount"><br>
<input type="date" id="date"><br>

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

<ul id="list"></ul>

</div>

<script>

let db;

function openDB()
{
  let request = indexedDB.open("expenseDB",1);

  request.onupgradeneeded = function(e)
  {
    db = e.target.result;
    db.createObjectStore("expenses",{keyPath:"id",autoIncrement:true});
  };

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

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

function addExpense()
{
  let t = title.value;
  let a = amount.value;
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
  let d = date.value;

  if(t=="" || a=="" || d=="")
  {
    alert("fill all fields");
    return;
  }

  let tx = db.transaction("expenses","readwrite");

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

  store.add({title:t,amount:a,date:d});

  tx.oncomplete = function()
  {
    alert("expense added");
    title.value="";
    amount.value="";
    date.value="";
  };

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

function viewExpenses()
{
  list.innerHTML="";

  let tx = db.transaction("expenses","readonly");

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

  let request = store.openCursor();

  request.onsuccess = function(e)
  {
    let cursor = e.target.result;

    if(cursor)
    {
      let li = document.createElement("li");

      li.innerHTML =
      cursor.value.title+" - ₹"+
      cursor.value.amount+" - "+
      cursor.value.date+
```

DAY-1_Events_HTML5_API_ Hands-On_Uday

```
        " <button onclick='deleteExpense("+cursor.value.id+")'>Delete</button>";

        list.appendChild(li);

        cursor.continue();
      }
   };

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

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

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

   store.delete(id);

   tx.oncomplete = function()
   {
      viewExpenses();
   };

   tx.onerror = function()
   {
      alert("delete error");
   };
}

</script>

</body>
</html>
```

- **Technical Explanation :**

I developed a Mini Expense Tracker using **IndexedDB** as the client-side database to store and manage expenses directly in the browser without any backend support. The application opens the database using indexedDB.open() on page load and creates an object store named **"expenses"** with keyPath and autoIncrement during the onupgradeneeded event. Users can enter Expense Title, Amount, and Date, and on clicking **Add Expense**, a readwrite transaction is created to store the data using objectStore.add(). To display records, a readonly transaction is used along with openCursor() to iterate through all stored expenses dynamically and render them in the DOM as list items. Each expense includes a Delete button that removes the specific record using objectStore.delete() within a transaction. Proper transaction handling is implemented using oncomplete and onerror events to manage successful operations and errors. This project

DAY-1_Events_HTML5_API_ Hands-On_Uday

demonstrates client-side database management, transaction handling, structured data storage, and dynamic DOM rendering using pure HTML and JavaScript.

- **OUTPUT:**