**Full Stack Web Development Laboratory** **22CS606B**

1. **HTML and CSS Webpage: Create a simple webpage that showcases your favorite hobby. Use HTML to structure the content and CSS to style the page, including adding colors, fonts, and images.**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Photography - My Favorite Hobby</title>
<style>
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
background-color: #f4f4f4;
text-align: center;
}
header {
background-color: #333;
color: white;
padding: 20px;
font-size: 24px;
}
.container {
padding: 20px;
}
.hobby-img {
width: 60%;
max-width: 600px;
border-radius: 10px;
margin-top: 20px;
}
```

```
p {
font-size: 18px;
color: #555;
line-height: 1.6;
}
</style>
</head>
<body>
<header>Photography - Capturing Moments</header>
<div class="container">
<p>Photography allows me to capture the beauty of the world and preserve
memories forever.
It is my creative outlet and a way to express emotions through images.</p>
<img src="https://source.unsplash.com/600x400/?photography,camera"
alt="Photography Hobby" class="hobby-img">
</div>
</body>
</html>
```

2. **JavaScript Form Validation: Develop a web form with fields for name, email, and password. Implement JavaScript validation to ensure that all fields are filled correctly before submitting the form.**

```
<!DOCTYPE html>
<html>
<head>
 <title>Form Validation</title>
 <style>
  body {
   font-family: Arial, sans-serif;
   margin: 40px;
  }
  .error {
   color: red;
  }
  input {
   margin-bottom: 10px;
   padding: 8px;
   width: 250px;
  }
  button {
   padding: 8px 15px;
  }
 </style>

</head>
<body>

 <h2>Registration Form</h2>
 <form id="myForm" onsubmit="return validateForm()">
  <div>
   <label>Name:</label><br>
```

```
    <input type="text" id="name" />
    <div class="error" id="nameError"></div>
  </div>

  <div>
    <label>Email:</label><br>
    <input type="text" id="email" />
    <div class="error" id="emailError"></div>
  </div>

  <div>
    <label>Password:</label><br>
    <input type="password" id="password" />
    <div class="error" id="passwordError"></div>
  </div>

  <button type="submit">Submit</button>
</form>

<script>
  function validateForm() {
    // Get field values
    const name = document.getElementById("name").value.trim();
    const email = document.getElementById("email").value.trim();
    const password = document.getElementById("password").value;

    // Clear previous errors
    document.getElementById("nameError").innerText = "";
    document.getElementById("emailError").innerText = "";
    document.getElementById("passwordError").innerText = "";
```

```
    let valid = true;

    // Name validation
    if (name === "") {
      document.getElementById("nameError").innerText = "Name is required.";
      valid = false;
    }

    // Email validation
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (email === "") {
      document.getElementById("emailError").innerText = "Email is required.";
      valid = false;
    } else if (!emailRegex.test(email)) {
      document.getElementById("emailError").innerText = "Invalid email format.";
      valid = false;
    }

    // Password validation
    if (password === "") {
      document.getElementById("passwordError").innerText = "Password is required.";
      valid = false;
    } else if (password.length < 6) {
      document.getElementById("passwordError").innerText = "Password must be at least 6
characters.";
      valid = false;
    }
    return valid;
  }
 </script>
</body>
</html>
```

**3. Node.js Server with Express: Build a basic server using Node.js and Express. Create routes to handle HTTP requests like GET and POST and respond with simple JSON data.**

```
// Import required packages
const express = require('express');
const bodyParser = require('body-parser');
// Initialize Express app
const app = express();
const port = 3000;
// Middleware for parsing JSON bodies
app.use(bodyParser.json());
app.use(express.urlencoded({ extended: true }));
// Sample data - in a real app, this would likely be a database
let users = [
  { id: 1, name: 'Alice', email: 'alice@example.com' },
  { id: 2, name: 'Bob', email: 'bob@example.com' }
];
// Root route
app.get('/', (req, res) => {
  res.json({ message: 'Welcome to the Express Server API' });
});

// GET all users
app.get('/api/users', (req, res) => {
res.json(users);
});

// GET user by ID
app.get('/api/users/:id', (req, res) => {
const id = parseInt(req.params.id);
  const user = users.find(user => user.id === id);

  if (!user) {
    return res.status(404).json({ message: 'User not found' });
  }
  res.json(user);
});
 POST - Create a new user
app.post('/api/users', (req, res) => {
```

```
  const { name, email } = req.body;

  // Simple validation
  if (!name || !email) {
    return res.status(400).json({ message: 'Name and email are required' });
  }

  const newId = users.length > 0 ? Math.max(...users.map(user => user.id)) + 1 : 1;
  const newUser = { id: newId, name, email };

  users.push(newUser);
  res.status(201).json(newUser);
});


// Start the server
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

```
mkdir my-node-server
cd my-node-server

npm init -y

npm install express

node server.js
```

4. **Database Integration: Extend the previous Node.js server by integrating a database (e.g., SQLite or MongoDB). Implement endpoints to perform CRUD operations on a dataset.**

```
project/
│
├── server.js
└── database.sqlite (auto-created)
```

```
npm init -y
npm install express sqlite3
```

**server.js**

```
const express = require("express");
const sqlite3 = require("sqlite3").verbose();
const app = express();

const PORT = 3000;

app.use(express.json());

// Connect to SQLite database
const db = new sqlite3.Database("./database.db", (err) => {
  if (err) console.error(err.message);
  else console.log("Connected to SQLite database.");
});

// Create Table
db.run(`CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT,
age INTEGER)`);

// CRUD Endpoints
app.post("/users", (req, res) => {
  const { name, age } = req.body;
  db.run("INSERT INTO users (name, age) VALUES (?, ?)", [name, age], function (err) {
    if (err) return res.status(500).json({ error: err.message });
    res.json({ id: this.lastID, name, age });
  });
```

```
});

app.get("/users", (req, res) => {
  db.all("SELECT * FROM users", [], (err, rows) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(rows);
  });
});

app.put("/users/:id", (req, res) => {
  const { name, age } = req.body;
  db.run("UPDATE users SET name = ?, age = ? WHERE id = ?", [name, age, req.params.id],
function (err) {
    if (err) return res.status(500).json({ error: err.message });
    res.json({ updated: this.changes });
  });
});

app.delete("/users/:id", (req, res) => {
  db.run("DELETE FROM users WHERE id = ?", [req.params.id], function (err) {
    if (err) return res.status(500).json({ error: err.message });
    res.json({ deleted: this.changes });
  });
});

app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

**5. RESTful API: Design and implement a RESTul API using Node.js, Express, and a database of your choice. Define endpoints for managing resources, such as creating, reading, updating, and deleting data.**

```
// server.js
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const app = express();

// Middleware
app.use(express.json());

// Connect to MongoDB
mongoose.connect(process.env.MONGO_URI || 'mongodb://localhost:27017/bookdb', {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch((err) => console.error('Connection error:', err));

// Schema and model
const bookSchema = new mongoose.Schema({
  title: { type: String, required: true },
  author: String,
  year: Number
});
const Book = mongoose.model('Book', bookSchema);

// Routes
// GET all books
app.get('/api/books', async (req, res) => {
  const books = await Book.find();
  res.json(books);
});

// GET one book by ID
app.get('/api/books/:id', async (req, res) => {
try {
    const book = await Book.findById(req.params.id);
    if (!book) return res.status(404).send('Book not found');
    res.json(book);
  } catch {
    res.status(400).send('Invalid ID');
```

```
  }
});

// POST create a new book
app.post('/api/books', async (req, res) => {
const book = new Book(req.body);
  try {
    await book.save();
    res.status(201).json(book);
  } catch (err) {
    res.status(400).send(err.message);
  }
});

// PUT update a book
app.put('/api/books/:id', async (req, res) => {
try {
    const book = await Book.findByIdAndUpdate(req.params.id, req.body, { new: true });
    if (!book) return res.status(404).send('Book not found');
    res.json(book);
  } catch {
    res.status(400).send('Invalid  update');
  }
});

// DELETE a book
app.delete('/api/books/:id', async (req, res) => {
  try {
    const book = await Book.findByIdAndDelete(req.params.id);
    if (!book) return res.status(404).send('Book not found');
    res.send('Book deleted');
  } catch {
    res.status(400).send('Invalid ID');
  }
});

// Start server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`□ Server running at http://localhost:${PORT}`);
});
```

**6. React Component Library: Create a library of reusable React components. Build components like buttons, cards, and modals and use them in a sample React application.**

## Step 1: Create the React Workspace and Component Library

**i. Create a new React project using Create React App (CRA):**

npx create-react-app component-lib

cd component-lib

**ii. Inside the project, create a components folder:**

mkdirsrc/components

This folder will store all reusable components like `Button`, `Card`, and `Modal`.

**Step 2: Create Reusable Components**

```
src/
└── components/
├── Button.js
├── Card.js
└── Modal.js
```

**Button.js (Reusable Button Component)**

```
import React from 'react';
// A functional component that accepts props for text and click behavior
const Button = ({ label, onClick, type = "button", style = {} }) => {
return (
   // Button element with prop
<button type={type} onClick={onClick} style={style}>
    {label}
</button>
 );
};
export default Button; // Exporting to use in other files
```

**Card.js (Reusable Card Component)**

```
import React from 'react';
// A simple card component with optional title and children content
const Card = ({ title, children, style = {} }) => {
return (
<div style={{
```

```
border: '1px solid #ccc',
borderRadius: '8px',
padding: '16px',
margin: '16px 0',
boxShadow: '0 2px 4px rgba(0,0,0,0.1)',

    ...style
  }}>
    {title&&<h3>{title}</h3>}
    {children}
</div>
 );
};
export default Card;
```

**Modal.js (Reusable Modal Component)**

```
import React from 'react';
// A modal component that appears as a popup when 'isOpen' is true
const Modal = ({ isOpen, onClose, children }) => {
if (!isOpen) return null; // Don't render if not open
return (
<div style={{
position: 'fixed',
top: 0, left: 0, right: 0, bottom: 0,
backgroundColor: 'rgba(0, 0, 0, 0.5)',
display: 'flex', justifyContent: 'center', alignItems: 'center',
zIndex: 1000
  }}>
<div style={{
backgroundColor: '#fff',
padding: '20px',
borderRadius: '10px',
minWidth: '300px',
maxWidth: '500px'
    }}>
<button onClick={onClose} style={{ float: 'right' }}>X</button>
      {children}
</div>
</div>
```

```
  );
};
export default Modal;
```

**Step 3:** Use Components in App.js

Replace the default code in src/App.js:

```
// src/App.js

import React, { useState } from 'react';
import Button from './components/Button';
import Card from './components/Card';
import Modal from './components/Modal';
function App() {
const [isModalOpen, setModalOpen] = useState(false); // State tocontrol modal visibility

return (

<div style={{ padding: '2rem' }}>
<h1>Reusable React Component Library Demo</h1>

{/* Card with some content */}

<Card title="Sample Card">
<p>This is some content inside a reusable Card component.</p>
<Button label="Open Modal" onClick={() =>setModalOpen(true)} />
</Card>
    {/* Modal popup */}
<Modal isOpen={isModalOpen} onClose={() =>setModalOpen(false)}>
<h2>This is a Modal</h2>
<p>You can close it by clicking the X.</p>
</Modal>
</div>
  );
}
export default App;
```

```
react-component-library/
├── src/
│   └── App.js
```

To run the program

```
npm start
```

## 7. Write a program to create a simple calculator Application using React JS.

npx create-react-app calculator

cd calculator

Replace App.js & App.css file in src folder with below file

**App.js**

```
import React, { useState } from "react";
import "./App.css";

function App() {
  const [input, setInput] = useState("");

  const handleClick = (value) => {
    setInput((prev) => prev + value);
  };

  const handleClear = () => {
    setInput("");
  };

  const handleCalculate = () => {
    try {
      const result = eval(input); // Warning: eval() can be unsafe in real apps
      setInput(result.toString());
    } catch (error) {
      setInput("Error");
    }
  };

  return (
    <div className="calculator">
      <h2>React Calculator</h2>
      <input type="text" value={input} readOnly />

      <div className="buttons">
```

```
      {["7", "8", "9", "/", "4", "5", "6", "*", "1", "2", "3", "-", "0", ".", "=",
"+"].map((btn, i) => (
        <button
          key={i}
          onClick={() => {
            btn === "=" ? handleCalculate() : handleClick(btn);
          }}
        >
          {btn}
        </button>
      ))}
      <button onClick={handleClear} className="clear">
      </button>
    </div>

  </div>
 );
}

export default App;
```

**App.css**

```
.calculator {
  max-width: 300px;
  margin: 50px auto;
  padding: 20px;
  background: #f0f0f0;
  border-radius: 12px;
  box-shadow: 0px 0px 15px rgba(0, 0, 0, 0.1);
  text-align: center;
}

.calculator input {
  width: 100%;
  height: 40px;
  font-size: 1.5rem;
  text-align: right;
  margin-bottom: 10px;
```

```
  padding: 5px;
  border-radius: 6px;
  border: 1px solid #ccc;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 10px;
}

button {
  padding: 15px;
  font-size: 1.2rem;
  border: none;
  border-radius: 8px;
  background: #ddd;
  cursor: pointer;
  transition: background 0.2s;
}

button:hover {
  background: #bbb;
}

.clear {
  grid-column: span 4;
  background-color: #ff6666;
  color: white;
}
```

**To run the program**

**npm start**

## 8. Create a Simple Login form using React JS.

```
npx create-react-app login-form
cd login-form
```

Replace App.js & App.css file in src folder with below file

**App.js**

```
import React, { useState } from 'react';
import './App.css';
function App() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');
  const handleSubmit = (e) => {
    e.preventDefault();
    if (username === 'admin' && password === '1234') {
      setMessage('Login successful!');
    } else {
      setMessage('Invalid credentials!');
    }
  };
  return (
    <div className="App">
      <h2>Login Form</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Username:</label><br />
          <input
            type="text"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            required
          />
        </div>
        <div>
          <label>Password:</label><br />
```

```
        <input
          type="password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
      </div>
      <button type="submit">Login</button>
    </form>
    <p>{message}</p>
  </div>
 );
}
export default App;
```

**App.css**

```
.App {
 font-family: Arial, sans-serif;
 margin: 50px auto;
 width: 300px;
 text-align: left;
 padding: 20px;
 border: 1px solid #ccc;
 border-radius: 10px;
}
input {
 width:  100%;
 padding:  8px;
 margin: 8px 0;
 box-sizing: border-box; }
button {
 padding: 10px;
 width: 100%;
 background-color: #007bff;
 color: white;
 border: none;
 border-radius: 5px;
} To run the program npm start
```