1. Create a simple registration form with fields for Name, Email, and Password, along with a Submit button

```
<!DOCTYPE html>
<html>
<body>

<h2>Registration Form</h2>

<form>
  <label>Name:</label><br>
  <input type="text" required><br><br>

  <label>Email:</label><br>
  <input type="email" required><br><br>

  <label>Password:</label><br>
  <input type="password" required><br><br>

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

</body>
</html>
```

2. Design a basic calculator using JavaScript that performs addition, subtraction, multiplication, and division.

```
<!DOCTYPE html>
<html>
<body>

<h3>Simple Calculator</h3>

<input id="num1" type="number" placeholder="Number 1">
<input id="num2" type="number" placeholder="Number 2">

<select id="op">
  <option value="+">+</option>
  <option value="-">-</option>
  <option value="*">*</option>
  <option value="/">/</option>
</select>

<button onclick="calculate()">Calculate</button>

<h3>Result: <span id="result"></span></h3>

<script>
function calculate() {
```

```
  let a = Number(document.getElementById("num1").value);
  let b = Number(document.getElementById("num2").value);
  let op = document.getElementById("op").value;

  let res = eval(a + op + b);  // simple evaluation
  document.getElementById("result").innerText = res;
}
</script>

</body>
</html>
```

3. Build a responsive navigation bar with links for Home, About, and Contact pages

```
<!DOCTYPE html>
<html lang="en">
<head>
<style>
  body { margin: 0; font-family: Arial; }

  .navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background: #333;
    padding: 10px;
  }

  .navbar a {
    color: white;
    padding: 10px;
    text-decoration: none;
  }

  .navbar a:hover {
    background: #555;
  }

  /* Mobile menu */
  @media (max-width: 600px) {
    .navbar {
      flex-direction: column;
    }
  }
</style>
</head>

<body>

<div class="navbar">
```

```
  <a href="#">Home</a>
  <a href="#">About</a>
  <a href="#">Contact</a>
</div>

</body>
</html>
```

4. Create a webpage that changes its background color each time a button is clicked.

```
<!DOCTYPE html>
<html>
<body>

<button onclick="changeColor()">Change Background</button>

<script>
function changeColor() {
  document.body.style.backgroundColor =
    "#" + Math.floor(Math.random() * 16777215).toString(16);
}
</script>

</body>
</html>
```

5. Write a JavaScript program to display the current time and update it every second.

```
<!DOCTYPE html>
<html>
<body>
    <h2 id="time"></h2>

    <script>
        function showTime() {
            document.getElementById("time").innerText = new
Date().toLocaleTimeString();
        }
        setInterval(showTime, 1000);
        showTime();
    </script>
</body>
</html>
```

6. Develop a simple counter application with increment and decrement buttons using React.

```
import { useState } from "react";
```

```
export default function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <h2>{count}</h2>
      <button onClick={() => setCount(count + 1)}>+</button>
      <button onClick={() => setCount(count - 1)}>-</button>
    </div>
  );
}
```

7. Create a React component that displays a list of quotes from an array.

```
export default function Quotes() {
  const quotes = ["Be happy", "Work hard", "Stay positive"];

  return (
    <ul>
      {quotes.map((q, i) => (
        <li key={i}>{q}</li>
      ))}
    </ul>
  );
}
```

8. Build a React form that displays the entered user name dynamically as the user types.

```
import { useState } from "react";

export default function UsernameForm() {
  const [name, setName] = useState("");

  return (
    <div>
      <input
        type="text"
        placeholder="Enter name"
        onChange={(e) => setName(e.target.value)}
      />
      <h3>You typed: {name}</h3>
    </div>
  );
}
```

9. Fetch and display user data from an external API using React’s useEffect hook.

```
import { useEffect, useState } from "react";
```

```
export default function Users() {
  const [users, setUsers] = useState([]);

  useEffect(() => {
    fetch("https://jsonplaceholder.typicode.com/users")
      .then(res => res.json())
      .then(data => setUsers(data));
  }, []);

  return (
    <ul>
      {users.map(u => (
        <li key={u.id}>{u.name}</li>
      ))}
    </ul>
  );
}
```

10. Develop a To-Do List application in React that allows users to add and view tasks.

```
import { useState } from "react";

export default function Todo() {
  const [task, setTask] = useState("");
  const [list, setList] = useState([]);

  const addTask = () => {
    setList([...list, task]);
    setTask("");
  };

  return (
    <div>
      <input value={task} onChange={(e) => setTask(e.target.value)} />
      <button onClick={addTask}>Add</button>

      <ul>
        {list.map((t, i) => (
          <li key={i}>{t}</li>
        ))}
      </ul>
    </div>
  );
}
```

11. Create a basic Express server in Node.js that runs on port 3000 and returns “Hello World” as a response.

```
const express = require("express");
const app = express();

app.get("/", (req, res) => {
  res.send("Hello World");
});

app.listen(3000, () => console.log("Server running on port 3000"));
```

12. Develop a REST API endpoint in Node.js that returns a JSON message when accessed.

```
const express = require("express");
const app = express();

app.get("/api/message", (req, res) => {
  res.json({ msg: "Hello from API" });
});

app.listen(3000);
```

13. Write Node.js code to establish a connection to a MongoDB database using Mongoose.

```
const mongoose = require("mongoose");

mongoose.connect("mongodb://localhost:27017/mydb")
  .then(() => console.log("MongoDB Connected"))
  .catch(err => console.log(err));
```

14. Design a User schema using Mongoose and insert a new user record into the MongoDB collection.

```
const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({
  name: String,
  email: String
});

const User = mongoose.model("User", userSchema);

User.create({ name: "Abhinav", email: "abhi@example.com" })
  .then(() => console.log("User inserted"));
```

15. Create an API endpoint in Node.js to fetch and display all user data stored in a MongoDB collection.

```
const express = require("express");
```

```
const mongoose = require("mongoose");
const app = express();

mongoose.connect("mongodb://localhost:27017/mydb");

const User = mongoose.model("User", new mongoose.Schema({
  name: String,
  email: String
}));

app.get("/users", async (req, res) => {
  const users = await User.find();
  res.json(users);
});

app.listen(3000);
```