Malnad College of Engineering

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

Hassan – 573202, Karnataka, India

"FULL STACK WEB DEVELOPMENT"

"LABORATORY PROGRAMS"

Report

Submitted By:

NAME	USN
Ankitha K N	4MC22CS012

Submitted To:

Mr. M S Prapulla Kumar
Assistant Professor
Department of Computer Science and Engineering

Program 1

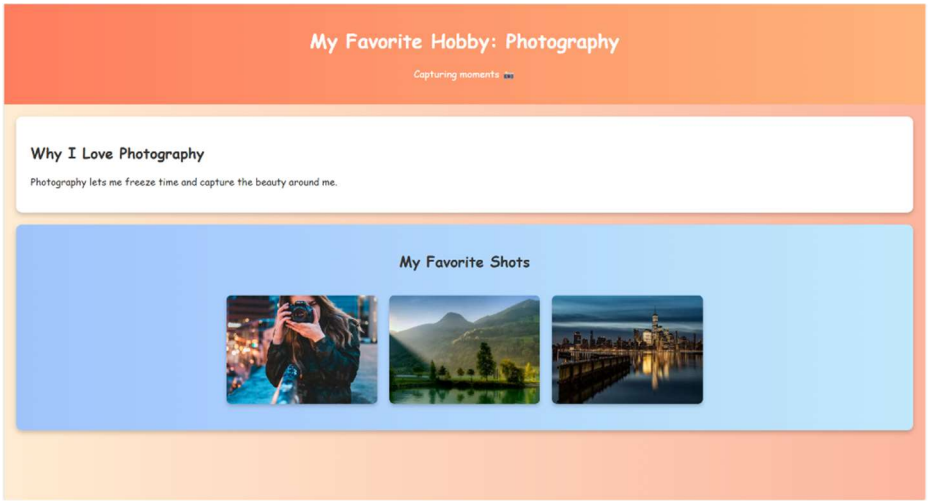
HTML and CSS Webpage: Create a simple webpage that showcases your favourite hobby. Use HTML to structure the content and CSS to style the page, including adding colors, fonts, and images.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>My Favorite Hobby: Photography</title>
   <style>
      body {
         font-family: 'Comic Sans MS', cursive, sans-serif;
         background: linear-gradient(to right, #ffecd2, #fcb69f);
         margin: 0;
         color: #333;
      }
      header {
         background: linear-gradient(to right, #ff7e5f, #feb47b);
         color: white;
         text-align: center;
         padding: 1.5rem;
      }
      section {
         background: #fff;
         margin: 20px;
         padding: 1.5rem;
         border-radius: 10px;
         box-shadow: 0 4px 8px rgba(0,0,0,0.2);
      }
      .gallery {
         background: linear-gradient(to right, #a1c4fd, #c2e9fb);
         text-align: center;
      }
```

```
        .gallery-images {
            display: flex;
            justify-content: center;
            flex-wrap: wrap;
            gap: 20px;
            padding: 20px 0;
        }
        .gallery-images img {
            width: 250px;
            height: 180px;
            object-fit: cover;
            border-radius: 8px;
            box-shadow: 0 4px 6px rgba(0,0,0,0.2);
        }
    </style>
</head>
<body>
    <header>
        <h1>My Favorite Hobby: Photography</h1>
        <p>Capturing moments 📸</p>
    </header>
    <section>
        <h2>Why I Love Photography</h2>
        <p>Photography lets me freeze time and capture the beauty around
me.</p>
    </section>
    <section class="gallery">
        <h2>My Favorite Shots</h2>
        <div class="gallery-images">
            <img src="C:\Users\LENOVO\OneDrive\Desktop\aa.jpeg"
alt="Camera">
            <img src="C:\Users\LENOVO\OneDrive\Desktop\bb.jpeg"
alt="Nature">
            <img src="C:\Users\LENOVO\OneDrive\Desktop\cc.jpeg" alt="City">
```

```
        </div>
    </section>
</body>
</html>
```

OUTPUT:



Program 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>
```

OUTPUT:

Program 3

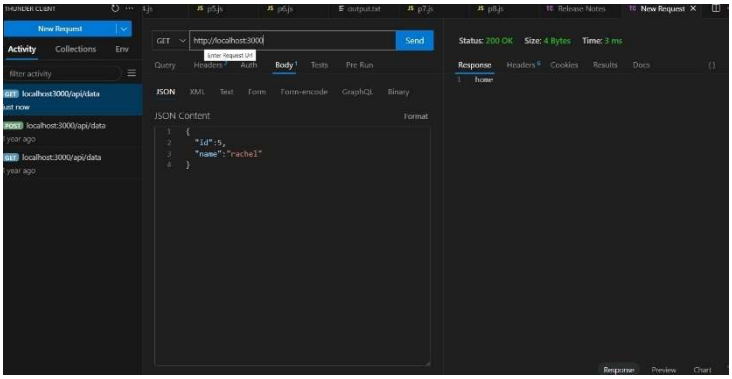
Node.js Server with Express: Build a basic server using Node.js and Express. Create routes to handle HTTP requests like GET and POST and respond with simple JSON data. Steps:

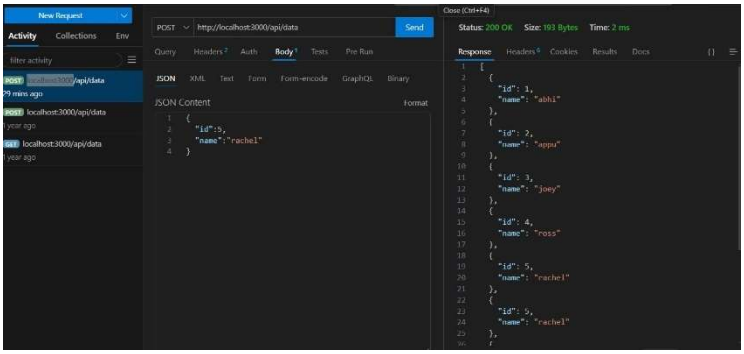
- npm install express
- node server.js

```
// Import required packages
 const express = require('express');

// Initialize Express app
const app = express();
const PORT = process.env.PORT || 3000;
app.use(express.json());
```

```
// Sample data - in a real app, this would likely be a database let
users = [
    { id: 1, name: 'abhi'},
    { id: 2, name: 'appu' },
    { id: 3, name: 'joey'},
    { id: 4, name: 'ross' },

];

// GET all users
app.get("/",(req,res)=>{
res.send("Home");

 });

app.get("/api/data",(req,res)=>{
res.json(data);

});



// POST - Create a new user
app.post('/api/users', (req, res) => {
const nItem = req.body;
data.push(nItem);

 res.json(data);

});


// Start the server
app.listen(PORT, (err)=>{
 if(err) console.log(err);
 console.log("Server running on port: ",PORT);
});
```

OUTPUT:





Program 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.

Steps:

```
project/
├── server.js
```

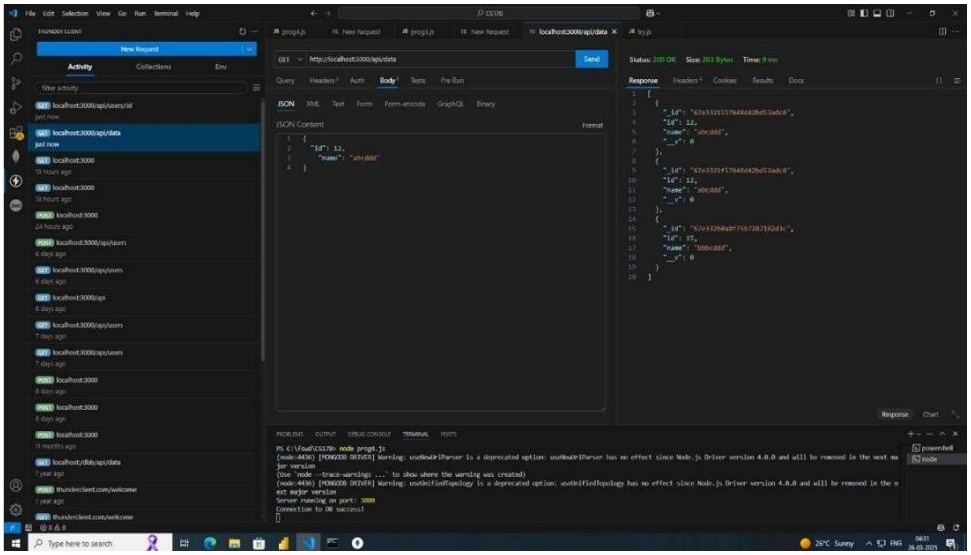
npm init -y

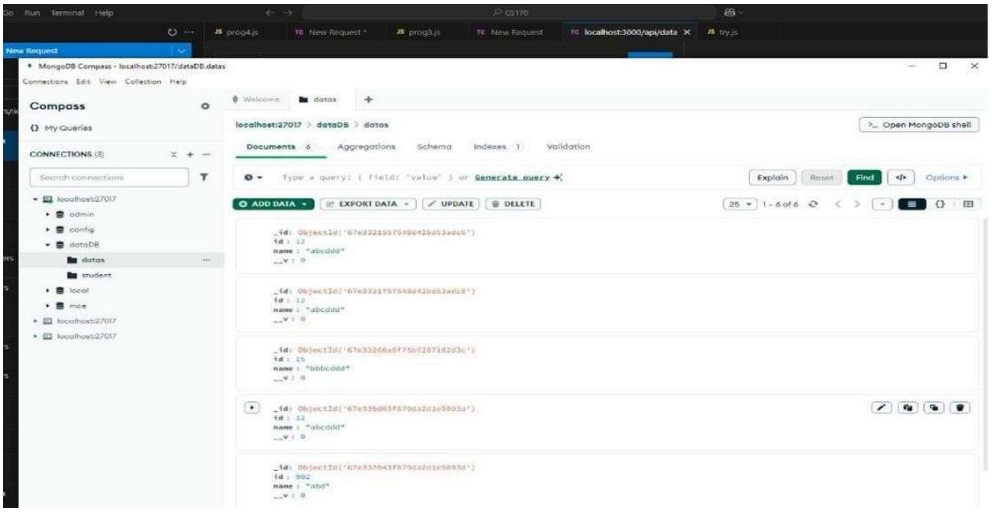
npm install mongoose

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();
const PORT =process.env.PORT || 3000;
app.use(express.json());
 //Connect to MongoDB
mongoose.connect("mongodb://localhost:27017/MCE2", {
useNewUrlParser: true,
useUnifiedTopology: true,
})
.then(()=>{ console.log("Connection to DB Successful !");
})
 .catch((err)=>{
 console.log("Connection to DB Failed !");
})
// Schema and Model
const dataSchema = new mongoose.Schema({
name: String,
 usn: String,
sem: String });
const Data = mongoose.model("Data", dataSchema);
// Get users
app.get("/", (req, res) => {
res.send("Home");
});
 app.get("/api/data", async (req, res) => {
```

```
try { const allData = await Data.find();MongoDB
res.json(allData);
} catch (err) {
console.error(err);
res.status(500).json({ message: "Internal Server Error" });
internal server error }
});
// Post user
app.post('/api/data', async (req, res)
=> {
try {
const nItem = req.body;
const newData = new Data(nItem);
await newData.save();
res.json(newData);
} catch (err) {
console.error(err);
res.status(500).json({ message:
"Internal Server Error" });
}
});
// Start server
app.listen(PORT, (err) => {
if (err) console.log(err); console.log("Server running
on port: ", PORT);
});
```

OUTPUT:





Program 5

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

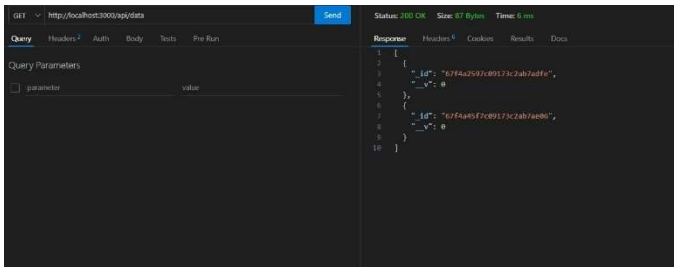
```
// server.js
const express = require('express');
const mongoose = require('mongoose');
const app = express(); // Middleware
const PORT = process.env.PORT ||
3000; default to 3000
app.use(express.json()); // Connect to
MongoDB
mongoose.connect("mongodb://localhost:27017/dataDB_1',{ ||
 useNewUrlParser:true,
useUnifiedTopology: true
})
.then(() =>{
 console.log('Connected to MongoDB');
})
.catch((err) =>{
 console.error('Connection to DB failed);
})
// Schema and model
 const dataSchema = new mongoose.Schema({
id: Number,
name: String,
});
```

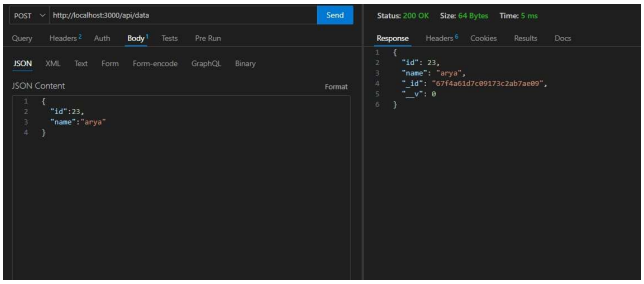
```
const Data = mongoose.model("Data",
dataSchema);

// GET
app.get("/api/data", async (req, res) => {
 try {
 const allData = await Data.find(); res.json(allData);
} catch (err) {
 console.error(err); res.status(500).json({ message: "Internal Server Error" });
}
});

// POST
app.post('/api/data', async (req, res) => {
 try {
const nItem = req.body; const newData = new
Data(nItem); await newData.save();
res.json(newData);
} catch (err) {
 console.error(err); res.status(500).json({
message: "Internal Server Error" });
 }
});
```

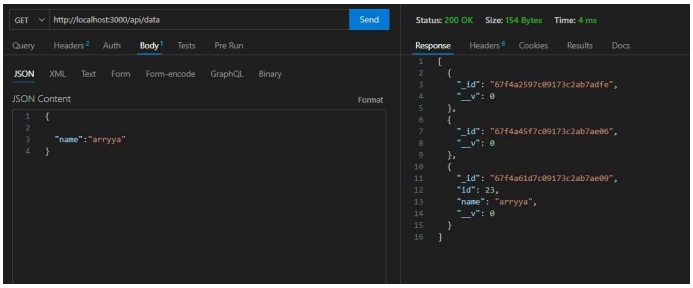
```
// PUT
 app.put('/api/data/:id', async (req, res) => {
try {
const id = req.params.id;
 const updatedItem = req.body;
 await Data.findOneAndUpdate({ id },
updatedItem);
res.json(updatedItem);
 } catch (err) {
console.error(err);
 res.status(500).json({ message: "Internal
Server Error" });
}
});
// DELETE
app.delete('/api/data/:id', async (req, res) => {
try {
const id = req.params.id;
 await Data.findOneAndDelete({ id });
res.json({ message: "Data deleted" });
} catch (err) {
console.error(err);
res.status(500).json({ message: "Internal
Server Error" });
 }
 });
```

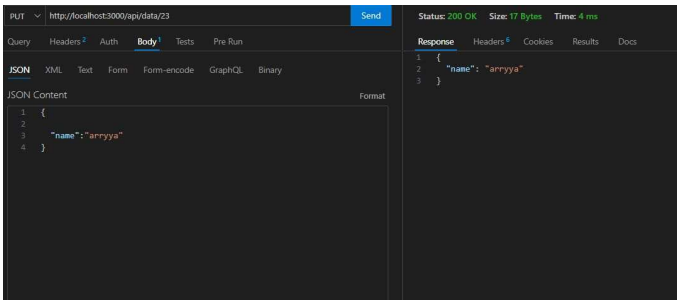
```
// Start server
app.listen(PORT, (err) => {
if (err) console.log(err); console.log("Server running on port: ",
PORT);
});
```

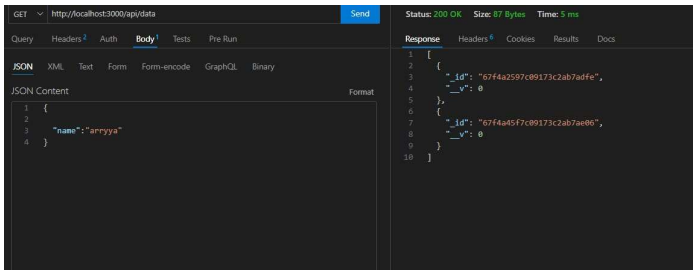
OUTPUT:

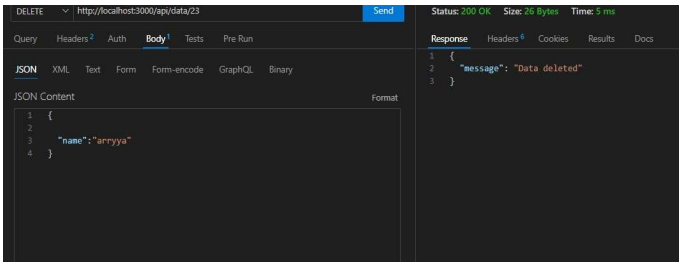












Program 6

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. Steps:

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';
 import PropTypes from 'prop-types';
 const Button=({ onClick,label})=>{
return(
```

```
<button onCilck={onClick}className="button">
  {label}
  </button>
  );
  };
Button.propTypes={
onClick:PropTypes.func.isRequired,
label:PropTypes.string.isRequired,
};
export default Button;
```

Card.js (Reusable Card Component)

```
 import React from 'react'; import PropTypes from
'prop-types';     const     Card=({title,content})=>{
return(
<div className="card">
            <h2>{title}</h2>
            <p>{content}</p>
</div>
);
};
Card.PropTypes={
title:PropTypes.string.isRequired,
content:PropTypes.string.isRequired, };
 export default Card;
```

Modal.js (Reusable Modal Component)

```
import React from 'react'; import PropTypes from
'prop-types'; const Modal = ({ isOpen, onClose,
children }) => {
 return (
 isOpen && (
<div className="modal-overplay">
  <div className="modal">
          <button
onClick={onClose}className="close-button">
          </button>
{children}
  </div>
</div>
)
);
};
 Modal.propTypes = {
 isOpen: PropTypes.bool.isRequired,
 onClose: PropTypes.func.isRequired,
 children: PropTypes.node.isRequired, };
export default Modal;
```

Step 3:

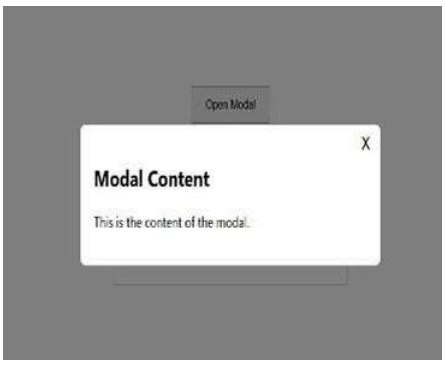
Use Components in App.js and Replace the default code in src/App.js:

```
// src/App.js
import React,{useState} from 'react';
import'./App.css';
 import Button from './components/Button';
import Card from './components/Card';
 import Modal from './components/Modal';
 const App=()=>{
 const [isModalOpen,setIsModalOpen]=useState(false);
return(
< div className="app">
  <Button label= "Open Modal" onClick={()=> setIsModalOpen (true)}/>
  <Card title= "Sample Card" content={()=> "this is sample component"/>
  <Modal isOpen= {isModalOpen}onClose={()=> setIsModalOpen(flase)}>
  <h2>Modal content</h2>
  <p>this is the content of modal</p>
  </Modal>
  </div>
  );
  };
  export default App;


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

Step 3:

To run the program: **npm start**

OUTPUT:



Program 7

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

Steps:

- npx create-react-app calculator
- cd calculator
- Replace calculator.js & App.js file in src folder with below file

calculator.js

```
import React, { useState } from
"react";
 export default function Calculator() {
const [input, setInput] = useState("");
const [result, setResult] = useState('');
const handleClick = (value) => {
 if (value === '=') {
 try {
```

```
setResult(eval(input).toString());
} catch {
setResult('Error');
 }
 } else if (value === 'C') {
 setInput('');
setResult('');
 } else {
setInput(input + value);
}
};
const buttons = ['7', '8', '9', '/', '4', '5', '6', '*', '1', '2', '3', '-', '0', '.', '=', '+', 'C'];
return (
        <div style={styles.container}>
           <h2>React Calculator</h2>
        <div style={styles.display}>
        <div>{input}</div>
        <div>= {result}</div>
        </div>
        <div style={styles.buttons}>
        {buttons.map((btn) => (
                <buttonkey={btn}style={styles.buttons}{onClick={()=
                   >handleClick(btn) }}>
                   {btn}
                </button>
              ))}
```

```
            </div>
        </div>
      );
    }
const styles={
  container:{
  max-width: 300px;
   margin: 50px auto;
   padding: 20px;
   background: #f0f0f0;
  border-radius: 12px;
   box-shadow: 0 0 10px #ddd;
  text-align: center;
  },
  display:{
  background:#f4f5f5;
   margin-bottom:10px;
  padding: 5px;
  font-size: 20;
   minHeight: 50;
  },
  buttons:{
   display: grid;
   grid-template-columns:  repeat(4,  1fr);
  gap: 10px;
   } ,
```

```
button :{
    padding: 15px;
    font-size: 18;
    cursor: pointer;
    transition: background 0.2s;
} ;
```

App.js

```
import React from 'react';
 import Calculator from './components/calculator';
 function App() {
return (
    <div className="App">
          <h2>Calculator</h2>
     </div>
);
}
export default App;
```

To run the program: **npm start**

OUTPUT:

Program 8

Create a Simple Login form using React JS. Steps:

- npx create-react-app login-form
- cd login-form
- Replace Form.js & App.js file in src folder with below file

Form.js

```
import React, { useState } from 'react';
export default 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 style={styles.container}>
    <h2>Login Form</h2>
    <form            onSubmit={handleSubmit}
      style={styles.form}>
     <input
        type="text"
        placeholder=”Username”
         value={username}
         onChange={(e) => setUsername(e.target.value)}
          style={styles.input}
        />
        <input
         type="password"
        placeholder=”Password”
         value={password}
         onChange={(e) => setPassword(e.target.value)}
        style={styles.input}
        />
```

```
            <button type="submit"
              style={styles.button}>Login</button>

          </form>
{message&&<p>{message}</p>
        </div>
      );
      }
   const styles={
     container:{
     margin: 50px auto;
     maxWidth:300;
      width: 300px;
       text-align:center;
       padding:20px;
       boxShadow:0
       010px#eee;
        border-radius: 10px;
     },
     form{
       display:flex,
       flexDirection:column,
       gap:10,
     },
     input{
       padding: 8px,
```

```
  border:1px solid#ccc,
  font-size:16,
  box-Radius: 4,
},
button{
  padding: 10px;
   width: 100%;
   background-color:
  #007bff;
   color: white;
   border: none;
   border-radius: 5px;
};
```

App.js

```
import React from 'react';
 import Form from './components/form';
 function App() {
return (
    <div className="App">
          <h2>Form</h2>
     </div>
);
}
export default App;
```

To run the program: **npm start**

OUTPUT:

Program 9

Full-Stack Task Manager: Develop a task manager application with React on the front end and Node.js/Express on the back end. Allow users to add, update, and delete tasks.

```
Task-manager/
├── backend/
│  ├── server.js
│  └── routes/
│      └── tasks.js
├── frontend/
│  └── (React App)
```

Steps:

1. Create Project Directory
   - mkdir task-manager
   - cd task-manager
2. Initialize backend
   - mkdir backend
   - cd backend
   - npm init -y
3. Install Dependencies
   - body-parser nodemon
4. Create server.js
   - echo> server.js

server.js

```
const express = require('express');
 const cors = require('cors');
 const bodyParser = require('body-
parser');
```

```
const app = express();
const PORT = 5000;
 app.use(cors());
app.use(bodyParser.json());
let tasks = [];
 let currentId = 1;
 app.get('/tasks',       (req,       res)       =>
res.json(tasks));
app.post('/tasks', (req, res) => {
  const task = { id: currentId++, ...req.body
  };
  tasks.push(task);
 res.status(201).json(task);
});
 app.put('/tasks/:id', (req, res) =>
{
  const { id } = req.params;
 const index = tasks.findIndex(t => t.id == id);
  if (index !== -1) {
    tasks[index] = { ...tasks[index], ...req.body };
    res.json(tasks[index]);
  }  else  {  res.status(404).send('Task  not
    found');
  }
   });
```

```
app.delete('/tasks/:id', (req, res) =>
{
  const { id } = req.params;
  tasks = tasks.filter(t => t.id != id);
res.status(204).send();
});
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

Add a start script in package.json

```
"scripts": {
  "start": "nodemon server.js" }
```

Frontend Setup (React) Steps:

- cd task-manager
- npx create-react-app frontend
- cd frontend
- Install Axios: npm install axios
- Create Task Manger Components
- Inside front/src/ , replace App.js with:

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
 function App() {
  const [tasks, setTasks] = useState([]);
  const [input, setInput] = useState('');
```

```
const [editId, setEditId] = useState(null);
useEffect(() => {
  fetchTasks();
}, []);
const fetchTasks = async () => {
  const res = await axios.get('http://localhost:5000/tasks'); setTasks(res.data);
};
const addOrUpdateTask = async () => {
  if (editId !== null) {
   await axios.put(`http://localhost:5000/tasks/${editId}`, { title: input });
                                                                  setEditId(null);
  } else { await axios.post('http://localhost:5000/tasks', { title:
    input });
  }
  setInput('');
  fetchTasks()
  ;
}; const deleteTask = async (id) =>
{
  await axios.delete(`http://localhost:5000/tasks/${id}`); fetchTasks();
}; const startEdit = (task)
=> {
  setInput(task.title);
  setEditId(task.id);
}; return
(
```

```
  <div style={{ padding: 20 }}>
    <h2>Task Manager</h2>
    <input   value={input}   onChange={(e)   =>
      setInput(e.target.value)}
      placeholder="Enter task"
    />
    <button    onClick={addOrUpdateTask}>{editId    ?    'Update'    :    'Add'}
Task</button>
    <ul>
      {tasks.map((task) => (
        <li key={task.id}>

          {task.title}{' '}
          <button onClick={() => startEdit(task)}>Edit</button>{' '}
          <button onClick={() => deleteTask(task.id)}>Delete</button>
        </li>
      ))}
    </ul>
  </div>
 ); }
export default App;
```

Run the frontend: npm start

OUTPUT:

Program 10

Real-Time Chat Application: Build a real-time chat application using React for the front end and Node.js with Socket.IO for the back end. Enable users to send and receive messages in real-time.

Steps:

- Create a Folder Called "Outer". Inside that create two folders called "Server" and "Client".
- Go to "Server" folder and create dependencies by hitting the command "npm install express socket.io" .
- Inside "Outer" enter the command "npx create-react-app client"
- Go to "Client" (or the react app client inside “Outer” folder) folder and create dependencies by hitting the command "npm install socket.io- client"
- inside “Server” folder create a file called “server.js”
- Inside client app Replace src/App.js with following code:

App.js:

```
import React, { useState, useEffect } from 'react';
import io from 'socket.io-client';
const socket = io('http://localhost:3001');
function App() {
  const [message, setMessage] = useState('');
  const [chat, setChat] = useState([]);
   const sendMessage = (e) => {
```

```
  e.preventDefault();
  if (message.trim()) {
    socket.emit('chat message', message);
    setMessage('');
  } };
useEffect(() =>
{
  socket.on('chat message', (msg) => {
    setChat((prevChat) => [...prevChat, msg]);
  }); return () => socket.off('chat
  message');
}, []); return
(
  <div className="p-4 max-w-md mx-auto">
    <h1 className="text-2xl font-bold mb-4">Real-Time Chat</h1>
    <div className="border rounded p-4 h-64 overflow-y-scroll mb-4">
      {chat.map((msg, index) => (
        <div key={index} className="mb-2">{msg}</div>
      ))}
    </div>
    <form onSubmit={sendMessage} className="flex gap-2">
      <input className="border rounded p-2 flex-
        grow" value={message} onChange={(e) =>
        setMessage(e.target.value)}
        placeholder="Type a message"
```

```
        />
        <button className="bg-blue-500 text-white rounded p-2" type="submit">Send</button>
      </form>
    </div>
  ); } export default
App;
```

server.js

```
const express = require('express');
const http = require('http');
const { Server } = require('socket.io');
 const cors = require('cors');
const app = express();
app.use(cors());
const server = http.createServer(app);
const io = new Server(server, { cors:
{
    origin: 'http://localhost:3000', methods:
    ['GET', 'POST']
  } }); io.on('connection',
(socket) => {
  console.log('A user connected:', socket.id);
  socket.on('chat message', (msg) => {
  io.emit('chat message', msg);
```

```
  });
   socket.on('disconnect',  ()  =>
   {
    console.log('User disconnected:', socket.id); });
});
const PORT = 3001; server.listen(PORT,
() => {
 console.log(`Server listening on http://localhost:${PORT}`);
});
```

OUTPUT: