APL - 2 (POE Programs)

1. Program for using registration form react.js class component

```
import "./style.css";

import React, { Component } from 'react';

class P1 extends Component {
    constructor(props) {
        super(props);
        this.state = {
            name: '',
            email: '',
            password: '',
            confirmPassword: '',
        };
    }

    handleChange = (event) => {
        this.setState({ [event.target.name]: event.target.value });
    };

    handleSubmit = (event) => {
        event.preventDefault();
        const { name, email, password, confirmPassword } = this.state;
        if (password !== confirmPassword) {
            alert("Passwords do not match");
        } else {
            alert(`Registration Successful!\nName: ${name}\nEmail: ${email}`);
        }
    };

    render() {
        return (
            <form onSubmit={this.handleSubmit}>
                <div>
                    <label>Name:</label>
                    <input
                        type="text"
                        name="fullName"
                        value={this.state.fullName}
                        onChange={this.handleChange}
                        required
                    />
                </div>
                <div>
                    <label>Email:</label>
```

```
                    <input
                        type="email"
                        name="email"
                        value={this.state.email}
                        onChange={this.handleChange}
                        required
                    />
                </div>
                <div>
                    <label>Password:</label>
                    <input
                        type="password"
                        name="password"
                        value={this.state.password}
                        onChange={this.handleChange}
                        required
                    />
                </div>
                <div>
                    <label>Confirm Password:</label>
                    <input
                        type="password"
                        name="confirmPassword"
                        value={this.state.confirmPassword}
                        onChange={this.handleChange}
                        required
                    />
                </div>
                <button type="submit">Register</button>
            </form>
        );
    }
}

export default P1;
```

2. Program for creating registration form using react.js function component.

```
import "./style.css";

import { useState } from 'react';

function P2() {
    const [name, setName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [confirmPassword, setConfirmPassword] = useState("");

    const handleSubmit = (event) => {
        event.preventDefault();
        if (password !== confirmPassword) {
```

```
            alert("Passwords do not match");
        } else {
            alert(`Registration Successful!\nName: ${name}\nEmail: ${email}`);
        }
    };


    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>Name:</label>
                <input
                    type="text"
                    name="fullName"
                    value={name}
                    onChange={(e)=>setName(e.target.value)}
                    required
                />
            </div>
            <div>
                <label>Email:</label>
                <input
                    type="email"
                    name="email"
                    value={email}
                    onChange={(e)=>setEmail(e.target.value)}
                    required
                />
            </div>
            <div>
                <label>Password:</label>
                <input
                    type="password"
                    name="password"
                    value={password}
                    onChange={(e)=>setPassword(e.target.value)}
                    required
                />
            </div>
            <div>
                <label>Confirm Password:</label>
                <input
                    type="password"
                    name="confirmPassword"
                    value={confirmPassword}
                    onChange={(e)=>setConfirmPassword(e.target.value)}
                    required
                />
            </div>
            <button type="submit">Register</button>
        </form>
```

```
    );

}

export default P2;
```

3. program for applying CSS style in react.js application

```
import "./style.css";

import { useState } from 'react';

function P2() {
    const [name, setName] = useState("");
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [confirmPassword, setConfirmPassword] = useState("");

    const handleSubmit = (event) => {
        event.preventDefault();
        if (password !== confirmPassword) {
            alert("Passwords do not match");
        } else {
            alert(`Registration Successful!\nName: ${name}\nEmail: ${email}`);
        }
    };


    return (
        <form onSubmit={handleSubmit}>
            <div>
                <label>Name:</label>
                <input
                    type="text"
                    name="fullName"
                    value={name}
                    onChange={(e)=>setName(e.target.value)}
                    required
                />
            </div>
            <div>
                <label>Email:</label>
                <input
                    type="email"
                    name="email"
                    value={email}
                    onChange={(e)=>setEmail(e.target.value)}
                    required
                />
            </div>
            <div>
```

```
                <label>Password:</label>
                <input
                    type="password"
                    name="password"
                    value={password}
                    onChange={(e)=>setPassword(e.target.value)}
                    required
                />
            </div>
            <div>
                <label>Confirm Password:</label>
                <input
                    type="password"
                    name="confirmPassword"
                    value={confirmPassword}
                    onChange={(e)=>setConfirmPassword(e.target.value)}
                    required
                />
            </div>
            <button type="submit">Register</button>
        </form>
    );

}

export default P2;
```

style.css - create a file

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

.App {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

h1 {
    text-align: center;
    color: #333;
    font-size: 2.5em;
    margin-bottom: 20px;
}
```

```css
.form-container {
    background-color: white;
    padding: 30px;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    width: 100%;
    max-width: 400px;
}

.form-group {
    margin-bottom: 20px;
}

label {
    font-weight: bold;
    display: block;
    margin-bottom: 8px;
    color: #333;
}

input {
    /* width: 100%; */
    padding: 10px;
    margin: 5px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

input:focus {
    border-color: #4CAF50;
    outline: none;
}

button {
    /* width: 100%; */
    padding: 12px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 1.1em;
}

button:hover {
    background-color: #45a049;
}

form{
```

```
    display: grid;
    grid-template-columns: auto;
    align-items: center;
    justify-content: center;
    margin-top: 50px;
}
```

4. program for display any 5 MUI Components

```
// npm install @mui/material @emotion/react @emotion/styled

import React, { useState } from 'react';
import { Typography, Button, TextField, Card, CardContent, Snackbar, Container } from '@mui

function P4() {
    const [open, setOpen] = useState(false);
    const [name, setName] = useState('');

    const handleSubmit = () => {
        if (name.trim()) {
            setOpen(true);
        }
    };

    const handleClose = () => {
        setOpen(false);
    };

    return (
        <Container component="main" maxWidth="xs">
            <div style={{ textAlign: 'center', marginTop: '50px' }}>
                <Typography variant="h4" gutterBottom>
                    MUI Component Demo
                </Typography>

                {/* TextField Component for Name Input */}
                <TextField
                    label="Enter Your Name"
                    fullWidth
                    value={name}
                    onChange={(e)=>setName(e.target.value)}
                    variant="outlined"
                    margin="normal"
                />

                <Button
                    variant="contained"
                    color="primary"
                    fullWidth
                    onClick={handleSubmit}
                >
```

```
                    Submit
                </Button>

                <Card sx={{ marginTop: 3 }}>
                    <CardContent>
                        <Typography variant="h6">Your Name</Typography>
                        <Typography variant="body1">{name || 'No name entered yet'}</Typogra
                    </CardContent>
                </Card>

                <Snackbar
                    open={open}
                    autoHideDuration={3000}
                    onClose={handleClose}
                    message="Name Submitted!"
                />
            </div>
        </Container>
    );
};

export default P4;
```

5 - 19 (Nodejs)

```
// Q5. program for printing hello on the browser using Node.js web module.

const http = require('http');

const server = http.createServer((req, res) => {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('Hello');
});

server.listen(3000, () => {
    console.log('Server is running at http://localhost:3000');
});
```

```
// Q6. Program for demonstrating the concept of callback function in Node.js

function addition(num1, num2, callback){
    result = num1 + num2;
    callback(result);
}

function printResult(result){
    console.log("Your Result is: ", result);
}

addition(100, 200, printResult);
```

```
// Q7. program to read the file contents using Node.js file system

const fs = require('fs');

const filePath = 'p7.txt';

// Read the file asynchronously
fs.readFile(filePath, 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading the file:', err);
        return;
    }
    console.log('File contents:');
    console.log(data);
});
```

```
// Q8. program to write the contents to the file using Node.js file system

const fs = require('fs');

const filePath = 'p8.txt';
const content = 'Hello, this content is written using Node.js!';

// Write the content to the file asynchronously
fs.writeFile(filePath, content, 'utf8', (err) => {
  if (err) {
    console.error('Error writing to the file:', err);
    return;
  }

  console.log('File written successfully!');
});
```

```
// Q9. Program to read the contents from the directory and display on console using Node.js

const fs = require('fs');
const dirPath = './';

fs.readdir(dirPath, (err, files) => {
    if (err) {
        console.error('Error reading the directory:', err);
        return;
    }

    console.log('Directory contents:');
    files.map((file)=>{
        console.log(file);
```

```
    })
});
```

```
// Q10. program for demonstrating any 5 functions of file systems

const fs = require('fs');

// 1. Create and Write to a File
fs.writeFileSync('sample.txt', 'This is a sample file.', 'utf8');
console.log('File created and written successfully.');

// 2. Read the File
const content = fs.readFileSync('sample.txt', 'utf8');
console.log('File contents:', content);

// 3. Append to the File
fs.appendFileSync('sample.txt', '\nAdding a new line.', 'utf8');
console.log('Content appended successfully.');

// 4. Check if the File Exists
if (fs.existsSync('sample.txt')) {
  console.log('File exists.');
}

// 5. Delete the File
fs.unlinkSync('sample.txt');
console.log('File deleted successfully.');
```

```
// Q11. Program for demonstrating any 5 functions of console global object

console.log('This is a regular log message.');

console.error('This is an error message.');

console.warn('This is a warning message.');

console.time('Timer');
for (let i = 0; i < 1e6; i++) {} // Simulating some code
console.timeEnd('Timer');

const data = [
  { id: 1, name: 'Abc', age: 25 },
  { id: 2, name: 'Xyz', age: 30 },
  { id: 3, name: 'Pqr', age: 35 },
];
console.table(data);
```

```
// Q12. program for demonstrating any 5 functions of process global object
```

```
console.log('Current Working Directory:', process.cwd());

console.log('Command-line arguments:', process.argv);

console.log('Environment Variables (sample):', process.env.PATH);

console.log('Process Uptime:', process.uptime(), 'seconds');

console.log('Exiting the process now...');
process.exit(0); // Exit with a success code (0)
```

```
// Q13. program for demonstrating any 5 functions of OS utility module

const os = require('os');

console.log('Operating System:', os.type());

console.log('Platform:', os.platform());

console.log('CPU Architecture:', os.arch());

console.log('Free Memory:', os.freemem() / (1024 * 1024), 'MB');
console.log('Total Memory:', os.totalmem() / (1024 * 1024), 'MB');

console.log('CPU Information:', os.cpus());
console.log('Number of CPU Cores:', os.cpus().length);
```

```
// Q14. Program for demonstrating any 5 functions of Path utility module

const path = require('path');

const joinedPath = path.join('/users', 'john', 'documents', 'file.txt');
console.log('Joined Path:', joinedPath);

const absolutePath = path.resolve('file.txt');
console.log('Absolute Path:', absolutePath);

const baseName = path.basename('/users/abc/documents/file.txt');
console.log('Base Name:', baseName);

const extName = path.extname('/users/abc/documents/file.txt');
console.log('File Extension:', extName);

const dirName = path.dirname('/users/abc/documents/file.txt');
console.log('Directory Name:', dirName);
```

```
// Q15. Program for demonstrating any 5 functions of Net utility module

const net = require('net');
```

```
const server = net.createServer((socket) => {
    console.log('Client connected');
    socket.write('Hello, client!\n');

    socket.on('data', (data) => {
        console.log('Received from client:', data.toString());
    });

    socket.on('end', () => {
        console.log('Client disconnected');
    });
});

server.listen(8080, () => {
    console.log('Server listening on port 8080');
});

const client = net.createConnection({ port: 8080 }, () => {
    console.log('Connected to server');
    client.write('Hello, server!');
});

client.on('data', (data) => {
    console.log('Server says:', data.toString());
    client.end();
});

client.on('end', () => {
    console.log('Disconnected from server');
});
```

```
// Q16. Program for demonstrating any 3 functions of DNS utility module

const dns = require('dns');

dns.lookup('www.google.com', (err, address, family) => {
    if (err) {
        console.log('Error:', err);
    } else {
        console.log('IP Address of www.google.com:', address);
        console.log('IP family:', family);
    }
});

dns.resolve('www.google.com', 'A', (err, addresses) => {
    if (err) {
        console.log('Error:', err);
    } else {
        console.log('A records for www.google.com:', addresses);
```

```
    }
});

dns.reverse('8.8.8.8', (err, hostnames) => {
    if (err) {
        console.log('Error:', err);
    } else {
        console.log('Hostnames for 8.8.8.8:', hostnames);
    }
});
```

```
// Q17. program for reading data from stream using Node.js

const fs = require('fs');

const readableStream = fs.createReadStream('p17.txt', { encoding: 'utf-8' });

readableStream.on('data', (chunk) => {
  console.log('Received chunk of data:');
  console.log(chunk);
});

readableStream.on('end', () => {
  console.log('Finished reading the file.');
});

readableStream.on('error', (err) => {
  console.error('Error reading file:', err.message);
});
```

```
// Q18. program for writing data to the stream using Node.js

const fs = require('fs');

const writableStream = fs.createWriteStream('p18.txt');

writableStream.write('Hello, this is the first line!\n');
writableStream.write('Here is the second line.\n');

writableStream.end('This is the last line.\n');

writableStream.on('finish', () => {
  console.log('Finished writing to the file.');
});

writableStream.on('error', (err) => {
  console.error('Error writing to the file:', err.message);
});
```

```
// Q19. program for creating a module for arithmetic operations and use it in another progra

// (Create maths.js File and Import it here)
const maths = require('./maths');

const a = 10;
const b = 5;

console.log(`Addition of ${a} and ${b}:`, maths.add(a, b));
console.log(`Subtraction of ${a} and ${b}:`, maths.subtract(a, b));
console.log(`Multiplication of ${a} and ${b}:`, maths.multiply(a, b));
console.log(`Division of ${a} and ${b}:`, maths.divide(a, b));
console.log(`Division of ${a} and 0:`, maths.divide(a, 0));
```

20-27 (express and mongdb one)

```
// Q20. program for demonstrating any 5 functions of request object in Express.js

const express = require('express');
const app = express();

app.use(express.json());

app.get('/', (req, res) => {
    res.send(`HTTP Method used: ${req.method}`);
});

app.get('/info', (req, res) => {
    res.send(`Request URL: ${req.url}`);
});

app.get('/user/:id', (req, res) => {
    const userId = req.params.id;
    res.send(`User ID from route: ${userId}`);
});

app.get('/search', (req, res) => {
    const searchQuery = req.query.q;
    res.send(`Search Query: ${searchQuery}`);
});

app.post('/submit', (req, res) => {
    const data = req.body;
    res.send(`Data received in body: ${JSON.stringify(data)}`);
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
```

```
    console.log(`Server running on http://localhost:${PORT}`);
});
```

```
// Q21. program for demonstrating any 5 functions of response object in Express.js

const express = require('express');
const app = express();

app.use(express.json());

app.get('/', (req, res) => {
    res.send('Hello, World! This is res.send() in action.');
});

app.get('/json', (req, res) => {
    res.json({ message: 'This is a JSON response using res.json()' });
});

app.get('/status', (req, res) => {
    res.status(201).send('Resource created successfully (HTTP 201)');
});

app.get('/redirect', (req, res) => {
    res.redirect('/json');
});

app.get('/headers', (req, res) => {
    res.set('X-Custom-Header', 'ExpressDemo');
    res.send('Custom header set using res.set()');
});

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});
```

```
// Q22. program for get HTTP method using Express.js

const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('Hello, World! This is a GET request.');
});

app.get('/user/:id', (req, res) => {
    const userId = req.params.id;
    res.send(`User ID received: ${userId}`);
});
```

```
const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

```
// Q23. program for post HTTP method using Express.js

const express = require('express');
const app = express();

app.use(express.json());

app.post('/', (req, res) => {
    res.send('Hello, World! This is a POST request.');
});

app.post('/submit', (req, res) => {
    const { name, age } = req.body; // Extract data from the request body
    res.send(`Received data: Name - ${name}, Age - ${age}`);
});

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

```
// Q24. program for put HTTP method using Express.js

const express = require('express');
const app = express();

app.use(express.json());

// Handle PUT request to update user information
app.put('/user/:id', (req, res) => {
  const userId = req.params.id;
  const { name, age } = req.body;

  res.send(`User with ID ${userId} updated. New details: Name - ${name}, Age - ${age}`);
});

// Start the server
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

```
// program for demonstrating the concept of Express.js router
const express = require('express');
```

```
const app = express();
const userRoutes = require('./userRoutes'); // Importing router module

app.use(express.json());
app.use('/users', userRoutes);

app.get('/', (req, res) => {
    res.send('Welcome to the Express.js Router Example!');
});

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});

// create another file which userRoutes
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
    res.send('List of users');
});

router.post('/', (req, res) => {
    const { name, age } = req.body;
    res.send(`New user created: Name - ${name}, Age - ${age}`);
});

router.put('/:id', (req, res) => {
    const userId = req.params.id;
    const { name, age } = req.body;
    res.send(`User with ID ${userId} updated: Name - ${name}, Age - ${age}`);
});

module.exports = router;
```

```
// Q26. program for demonstrating the use of app.use() in Express.js

const express = require('express');
const app = express();

app.use((req, res, next) => {
    console.log(`Request Method: ${req.method}, Request URL: ${req.url}`);
    next();
});

app.use(express.json());

app.get('/', (req, res) => {
    res.send('Welcome to the Express.js app!');
```

```
});

app.get('/users', (req, res) => {
    res.send('List of users');
});

const PORT = 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

```
// Q27.  Create the MongoDB database and insert the records either using interface or using

const mongoose = require("mongoose");

const url = 'mongodb://127.0.0.1:27017/p27';

async function dbConnect() {
    await mongoose.connect(url, {
        useNewUrlParser: true,
        useUnifiedTopology: true
    })
        .then(() => console.log("DB Connected Successfully!!"))
        .catch((error) => {
            console.log(error.message)
            process.exit(1);
        });
}

async function dbOperations() {

    await dbConnect();

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

    const User = mongoose.model('User', userSchema);
    const createUsers = async () => {
        try {

            const users = await User.insertMany([
                { name: 'Abc', age: 28, email: 'abc@gmail.com' },
                { name: 'Xyz', age: 24, email: 'xyz@gmail.com' }
            ]);

            console.log(`${users.length} users inserted`);
        } catch (err) {
```

```
            console.error('Error inserting users:', err);
        }
    }

    createUsers();
}

dbOperations();
```