UIUX

```
// ReactJS Solutions

// 1. Counter Component
import React, { useState } from 'react';

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

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>Counter</h1>
      <h2>{count}</h2>
      <button onClick={increment} style={{ padding: '10px 20px', fontSize: '16px' }}>
        Increment
      </button>
    </div>
  );
}

export default Counter;
```

```
// 2. Passing Data with Props
import React from 'react';

function ChildComponent({ message }) {
  return <h2>{message}</h2>;
}

function ParentComponent() {
  const greeting = "Hello from Parent!";

  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>Parent Component</h1>
      <ChildComponent message={greeting} />
    </div>
  );
}

export default ParentComponent;

// 3. Form Handling in React
import React, { useState } from 'react';

function FormComponent() {
  const [formData, setFormData] = useState({ name: '', email: '' });

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };
```

```
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log('Form Submitted:', formData);
  };

  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>Form Handling</h1>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          name="name"
          placeholder="Name"
          value={formData.name}
          onChange={handleChange}
        />
        <br />
        <input
          type="email"
          name="email"
          placeholder="Email"
          value={formData.email}
          onChange={handleChange}
        />
        <br />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
}
```

```
export default FormComponent;

// Node.js with MySQL Solutions

// 4. Registration Page with Password Hashing
const express = require('express');
const mysql = require('mysql');
const bcrypt = require('bcrypt');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'testdb'
});

db.connect((err) => {
  if (err) {
    console.error('Database connection failed:', err);
  } else {
    console.log('Connected to MySQL');
  }
});

app.post('/register', async (req, res) => {
  const { username, email, password, confirmPassword } = req.body;
```

```
  if (password !== confirmPassword) {
    return res.status(400).send('Passwords do not match');
  }

  const hashedPassword = await bcrypt.hash(password, 10);

  const query = 'INSERT INTO users (username, email, password) VALUES (?, ?, ?)';
  db.query(query, [username, email, hashedPassword], (err, result) => {
    if (err) {
      console.error('Error inserting user:', err);
      res.status(500).send('Database error');
    } else {
      res.status(201).send('User registered successfully');
    }
  });
});

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

_*_

```
// 1. ReactJS - Simple Counter Component
import React, { useState } from 'react';

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

  const incrementCount = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Counter Value: {count}</p>
      <button onClick={incrementCount}>Increment</button>
    </div>
  );
}

export default Counter;

// 2. ReactJS - Pass Data from Parent to Child Component using Props
import React from 'react';

function ChildComponent(props) {
  return <p>Received data from parent: {props.message}</p>;
}

function ParentComponent() {
  const message = "Hello from the parent!";
```

```
  return (
    <div>
      <h1>Parent Component</h1>
      <ChildComponent message={message} />
    </div>
  );
}

export default ParentComponent;

// 3. ReactJS - Handle Form Input and Submit Data
import React, { useState } from 'react';

function FormComponent() {
  const [formData, setFormData] = useState({
    name: '',
    email: ''
  });

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setFormData((prevData) => ({
      ...prevData,
      [name]: value
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log('Form submitted:', formData);
```

```
  };

  return (
    <form onSubmit={handleSubmit}>
      <div>
        <label>Name:</label>
        <input
          type="text"
          name="name"
          value={formData.name}
          onChange={handleInputChange}
        />
      </div>
      <div>
        <label>Email:</label>
        <input
          type="email"
          name="email"
          value={formData.email}
          onChange={handleInputChange}
        />
      </div>
      <button type="submit">Submit</button>
    </form>
  );
}

export default FormComponent;

// Node.js and MySQL Section
```

```
// 4. Registration with Password Hashing using bcrypt
const express = require('express');
const bcrypt = require('bcrypt');
const mysql = require('mysql');
const app = express();

app.use(express.json());

const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'users_db'
});

db.connect();

app.post('/register', async (req, res) => {
  const { username, email, password, confirmPassword } = req.body;

  if (password !== confirmPassword) {
    return res.status(400).send("Passwords do not match");
  }

  const hashedPassword = await bcrypt.hash(password, 10);

  const query = 'INSERT INTO users (username, email, password) VALUES (?, ?, ?)';
  db.query(query, [username, email, hashedPassword], (err) => {
    if (err) {
      return res.status(500).send("Error registering user");
    }
```

```
    res.status(201).send("User registered successfully");
  });
});

app.listen(3000, () => {
  console.log('Server running on port 3000');
});

// 5. Login with Password Verification
app.post('/login', async (req, res) => {
  const { email, password } = req.body;

  db.query('SELECT * FROM users WHERE email = ?', [email], async (err, result) => {
    if (err || !result.length) {
      return res.status(400).send("User not found");
    }

    const isMatch = await bcrypt.compare(password, result[0].password);

    if (isMatch) {
      res.send("Welcome!");
    } else {
      res.status(400).send("Invalid password");
    }
  });
});

// 6. Store User Data in MySQL Table
// Similar to Question 4, just ensure that a table 'users' exists in MySQL.

// 7. Add Employee Details
```

```
app.post('/employees', (req, res) => {
  const { name, designation, salary, department } = req.body;

  const query = 'INSERT INTO employees (name, designation, salary, department) VALUES (?, ?, ?, ?)';
  db.query(query, [name, designation, salary, department], (err) => {
    if (err) {
      return res.status(500).send("Error adding employee");
    }
    res.status(201).send("Employee added successfully");
  });
});

// 8. List Employee Details
app.get('/employees', (req, res) => {
  db.query('SELECT * FROM employees', (err, result) => {
    if (err) {
      return res.status(500).send("Error retrieving employees");
    }
    res.json(result);
  });
});

// 9. Update Employee Details
app.put('/employees/:id', (req, res) => {
  const { name, designation, salary, department } = req.body;
  const { id } = req.params;

  const query = 'UPDATE employees SET name = ?, designation = ?, salary = ?, department = ? WHERE
id = ?';
  db.query(query, [name, designation, salary, department, id], (err) => {
    if (err) {
```

```
      return res.status(500).send("Error updating employee");
    }
    res.send("Employee updated successfully");
  });
});

// 10. Delete Employee Details
app.delete('/employees/:id', (req, res) => {
  const { id } = req.params;

  const query = 'DELETE FROM employees WHERE id = ?';
  db.query(query, [id], (err) => {
    if (err) {
      return res.status(500).send("Error deleting employee");
    }
    res.send("Employee deleted successfully");
  });
});

// 11. List All Products
app.get('/products', (req, res) => {
  db.query('SELECT * FROM products', (err, result) => {
    if (err) {
      return res.status(500).send("Error retrieving products");
    }
    res.json(result);
  });
});

// 12. Get Product Details by ID
app.get('/products/:id', (req, res) => {
```

```
  const { id } = req.params;

  db.query('SELECT * FROM products WHERE id = ?', [id], (err, result) => {
    if (err || !result.length) {
      return res.status(404).send("Product not found");
    }
    res.json(result[0]);
  });
});

// 13. Add New Product
app.post('/products', (req, res) => {
  const { name, price, category, description, stock } = req.body;

  const query = 'INSERT INTO products (name, price, category, description, stock) VALUES (?, ?, ?, ?,
?)';
  db.query(query, [name, price, category, description, stock], (err) => {
    if (err) {
      return res.status(500).send("Error adding product");
    }
    res.status(201).send("Product added successfully");
  });
});

// 14. Update Product Stock or Price
app.put('/products/:id', (req, res) => {
  const { id } = req.params;
  const { price, stock } = req.body;

  const query = 'UPDATE products SET price = ?, stock = ? WHERE id = ?';
  db.query(query, [price, stock, id], (err) => {
```

```
    if (err) {
      return res.status(500).send("Error updating product");
    }
    res.send("Product updated successfully");
  });
});

// 15. Delete Product
app.delete('/products/:id', (req, res) => {
  const { id } = req.params;

  const query = 'DELETE FROM products WHERE id = ?';
  db.query(query, [id], (err) => {
    if (err) {
      return res.status(500).send("Error deleting product");
    }
    res.send("Product deleted successfully");
  });
});

// 16. Add Student Details
app.post('/students', (req, res) => {
  const { name, rollNumber, class, grade } = req.body;

  const query = 'INSERT INTO students (name, rollNumber, class, grade) VALUES (?, ?, ?, ?)';
  db.query(query, [name, rollNumber, class, grade], (err) => {
    if (err) {
      return res.status(500).send("Error adding student");
    }
    res.status(201).send("Student added successfully");
  });
```

```
});

// 17. Update Student Grade
app.put('/students/:rollNumber', (req, res) => {
  const { rollNumber } = req.params;
  const { grade } = req.body;

  const query = 'UPDATE students SET grade = ? WHERE rollNumber = ?';
  db.query(query, [grade, rollNumber], (err) => {
    if (err) {
      return res.status(500).send("Error updating student grade");
    }
    res.send("Student grade updated successfully");
  });
});

// 18. Get All Students
app.get('/students', (req, res) => {
  db.query('SELECT * FROM students', (err, result) => {
    if (err) {
      return res.status(500).send("Error retrieving students");
    }
    res.json(result);
  });
});

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