
SET 1 (Q1–Q3)

Q1. Node.js (Sequelize) — List students with marks greater than 60

1. Creation Steps (Terminal)

```
mkdir node-q1 && cd node-q1
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q1/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Student.js
└── routes/
    └── student.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Student.js
- routes/student.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
CREATE TABLE IF NOT EXISTS students(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  marks INT
);
INSERT INTO students(name,marks) VALUES ('Ali',75),('Sara',82),('Rohit',58);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",   // ☑ using localhost
  dialect: "mysql"
});
```

models/Student.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Student", {
  name: DataTypes.STRING,
  marks: DataTypes.INTEGER
}, { tableName: "students", timestamps: false });
```

routes/student.js

```
const express = require("express");
const { Op } = require("sequelize");
const Student = require("../models/Student");

const router = express.Router();

router.get("/", async (_req, res) => {
  const rows = await Student.findAll({ where: { marks: { [Op.gt]: 60 } } });
  res.json(rows);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const studentRoutes = require("./routes/student");

const app = express();
app.use(express.json());
app.use("/students", studentRoutes);

sequelize.sync().then(() => {
  app.listen(3000, () => console.log("☑ Server running on
http://localhost:3000"));
});
```

7. Running Steps

node server.js

Test:

curl http://localhost:3000/students

8. Expected Output

```
[
  {"id":1,"name":"Ali","marks":75},
  {"id":2,"name":"Sara","marks":82}
]
```

Q2. Node.js — Convert "Hello World!" to UPPERCASE

1. Creation Steps (Terminal)

mkdir node-q2 && cd node-q2

2. Folder Structure

```
node-q2/
 └── app.js
```

3. Required Installation

- Only Node.js

4. Files to Create

- app.js

5. DB Setup

- ❌ Not required

6. Code

```
console.log("Hello World!".toUpperCase());
```

7. Running Steps

node app.js

8. Expected Output

HELLO WORLD!

Q3. React — Display list of students (random data)

1. Creation Steps (Terminal)

```
npx create-react-app react-q3
cd react-q3
```

2. Folder Structure

```
react-q3/
 └── src/
     ├── App.js
     └── components/
         └── StudentList.js
```

3. Required Installation

- Node.js
- React via create-react-app

4. Files to Create

- src/components/StudentList.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/StudentList.js

```
export default function StudentList(){
  const students = [
    { id: 1, name: "Ali", marks: 75 },
    { id: 2, name: "Sara", marks: 82 },
    { id: 3, name: "Rohit", marks: 58 }
  ];

  return (
    <div>
      <h2>Students</h2>
      <ul>
        {students.map(s => (
          <li key={s.id}>{s.name} - {s.marks}</li>
        ))}
      </ul>
    </div>
  );
}
```

src/App.js

```
import StudentList from "./components/StudentList";

export default function App(){
  return <StudentList/>;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A webpage showing:

Students
Ali - 75
Sara - 82
Rohit - 58

SET 2 (Q4–Q6)

Q4. Create a Node.js server to delete student record by roll no using MySQL (Sequelize)

1. Creation Steps (Terminal)

```
mkdir node-q4 && cd node-q4
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q4/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Student.js
└── routes/
    └── student.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Student.js
- routes/student.js
- server.js

5. DB Setup (MySQL)

Use the same school.students table from Q1.

```
mysql -u root -p
USE school;
SELECT * FROM students;
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",   // ☑ localhost
  dialect: "mysql"
});
```

models/Student.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Student", {
  name: DataTypes.STRING,
  marks: DataTypes.INTEGER
}, { tableName: "students", timestamps: false });
```

routes/student.js

```
const express = require("express");
const Student = require("../models/Student");
const router = express.Router();

router.delete("/:id", async (req, res) => {
  const deleted = await Student.destroy({ where: { id: req.params.id } });
  res.json({ deleted });
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
```

```
const studentRoutes = require("./routes/student");

const app = express();
app.use(express.json());
app.use("/students", studentRoutes);

sequelize.sync().then(() => {
  app.listen(3001, () => console.log("☑ Server running on
http://localhost:3001"));
});
```

7. Running Steps

```
node server.js
curl -X DELETE http://localhost:3001/students/1
```

8. Expected Output

```
{"deleted":1}
```

Q5. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error

1. Creation Steps (Terminal)

```
mkdir node-q5 && cd node-q5
npm init -y
npm install express
```

2. Folder Structure

```
node-q5/
 └── server.js
```

3. Required Installation

- Node.js
- Express

4. Files to Create

- server.js

5. DB Setup

- ❌ Not required

6. Code

server.js

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

app.get("/:file", (req, res) => {
  const filePath = path.join(__dirname, req.params.file);
  fs.readFile(filePath, "utf8", (err, data) => {
    if (err) return res.status(404).send("404 File Not Found");
    res.type("text/plain").send(data);
  });
});

app.listen(3002, () => console.log("☑ Server running on
http://localhost:3002"));
```

7. Running Steps

```
echo "Hello Rashida" > test.txt
node server.js
curl http://localhost:3002/test.txt
```

8. Expected Output

```
Hello Rashida
```

Q6. Create a JavaScript application to illustrate the use of callback under functions

1. Creation Steps (Terminal)

```
mkdir js-q6 && cd js-q6
```

2. Folder Structure

```
js-q6/
 └── callback.js
```

3. Required Installation

- Only Node.js

4. Files to Create

- callback.js

5. DB Setup

- ❌ Not required

6. Code

callback.js

```
function fetchData(callback){
  console.log("Fetching data...");
  setTimeout(() => {
    callback(null, { id: 1, name: "Rashida" });
  }, 1000);
}

fetchData((err, result) => {
  if (err) return console.error("Error:", err);
  console.log("Callback result:", result);
});
```

7. Running Steps

node callback.js

8. Expected Output

Fetching data...
Callback result: { id: 1, name: 'Rashida' }

SET 3 (Q7–Q9)

Q7. Create a Node.js server to insert teacher data into teacher table using MySQL (Sequelize)

1. Creation Steps (Terminal)

mkdir node-q7 && cd node-q7
npm init -y
npm install express sequelize mysql2

2. Folder Structure

```
node-q7/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Teacher.js
└── routes/
    └── teacher.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Teacher.js
- routes/teacher.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
CREATE TABLE IF NOT EXISTS teachers(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  subject VARCHAR(50)
);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Teacher.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Teacher", {
  name: DataTypes.STRING,
  subject: DataTypes.STRING
}, { tableName: "teachers", timestamps: false });
```

routes/teacher.js

```
const express = require("express");
const Teacher = require("../models/Teacher");
const router = express.Router();

router.post("/", async (req, res) => {
```

```
  const teacher = await Teacher.create(req.body);
  res.status(201).json(teacher);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const teacherRoutes = require("./routes/teacher");

const app = express();
app.use(express.json());
app.use("/teachers", teacherRoutes);

sequelize.sync().then(() => {
  app.listen(3003, () => console.log("☑ Server running on
http://localhost:3003"));
});
```

7. Running Steps

```
node server.js
curl -X POST http://localhost:3003/teachers -H "Content-Type: application/json"
\
 -d '{"name":"Ayesha","subject":"Math"}'
```

8. Expected Output

```
{"id":1,"name":"Ayesha","subject":"Math"}
```

Q8. Create a Node.js file that demonstrates create database and table in MySQL (Sequelize)

1. Creation Steps (Terminal)

```
mkdir node-q8 && cd node-q8
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q8/
  └── init.js
```

3. Required Installation

- Node.js
- Sequelize

- MySQL2

4. Files to Create

- init.js

5. DB Setup (MySQL)

```
mysql -u root -p -e "CREATE DATABASE IF NOT EXISTS demo;"
```

6. Code

init.js

```
const { Sequelize, DataTypes } = require("sequelize");

(async () => {
  const sequelize = new Sequelize("demo", "root", "", {
    host: "localhost",
    dialect: "mysql"
  });

  const User = sequelize.define("User", {
    name: DataTypes.STRING
  }, { tableName: "users", timestamps: false });

  await sequelize.sync();
  console.log("☑ Database & Table created successfully");
})();
```

7. Running Steps

```
node init.js
```

8. Expected Output

☑ Database & Table created successfully

Q9. Create any React component to illustrate the React hook (useState)

1. Creation Steps (Terminal)

```
npx create-react-app react-q9
cd react-q9
```

2. Folder Structure

```
react-q9/
 └── src/
     └── components/
```

└── Counter.js

3. Required Installation

- Node.js
- React via create-react-app

4. Files to Create

- src/components/Counter.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Counter.js

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

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

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

7. Running Steps

- Add <Counter /> inside App.js
- Run:

```
npm start
```

8. Expected Output

A page showing a counter that increments by 1 each time the button is clicked.

SET 4 (Q10–Q12)

Q10. Create a Node.js Express server to generate JWT token in the response of login API

1. Creation Steps (Terminal)

```
mkdir node-q10 && cd node-q10
npm init -y
npm install express jsonwebtoken bcryptjs
```

2. Folder Structure

```
node-q10/
 └── server.js
```

3. Required Installation

- Node.js
- Express
- jsonwebtoken for JWT
- bcryptjs for password hashing

4. Files to Create

- server.js

5. DB Setup

- ❌ Not required (we will hardcode a sample user)

6. Code

server.js

```
const express = require("express");
const jwt = require("jsonwebtoken");
const bcrypt = require("bcryptjs");

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

const SECRET = "exam-secret"; // secret key
const user = {
  email: "test@site.com",
  password: bcrypt.hashSync("123456", 8) // hashed password
};

app.post("/login", (req, res) => {
  const { email, password } = req.body;

  if (email !== user.email || !bcrypt.compareSync(password, user.password)) {
    return res.status(401).json({ error: "Invalid credentials" });
  }

  const token = jwt.sign({ email }, SECRET, { expiresIn: "1h" });
```

```
  res.json({ token });
});

app.listen(3004, () => console.log("☑ Server running on
http://localhost:3004"));
```

7. Running Steps

```
node server.js
curl -X POST http://localhost:3004/login -H "Content-Type: application/json" \
 -d '{"email":"test@site.com","password":"123456"}'
```

8. Expected Output

```
{"token":"<JWT_TOKEN_HERE>"}
```

Q11. Create a Login component using React.js (with appropriate fields)

1. Creation Steps (Terminal)

```
npx create-react-app react-q11
cd react-q11
```

2. Folder Structure

```
react-q11/
 └── src/
     └── components/
         └── Login.js
```

3. Required Installation

- Node.js
- React via create-react-app

4. Files to Create

- src/components/Login.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Login.js

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

export default function Login(){
```

```
  const [form, setForm] = useState({ email: "", password: "" });

  const change = (e) => setForm({ ...form, [e.target.name]: e.target.value });
  const submit = (e) => {
    e.preventDefault();
    alert(`Email: ${form.email}, Password: ${form.password}`);
  };

  return (
    <form onSubmit={submit}>
      <h2>Login</h2>
      <input
        type="email"
        name="email"
        placeholder="Enter Email"
        onChange={change}
      />
      <input
        type="password"
        name="password"
        placeholder="Enter Password"
        onChange={change}
      />
      <button type="submit">Login</button>
    </form>
  );
}
```

src/App.js

```
import Login from "./components/Login";

export default function App(){
  return <Login />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A login form with email and password fields. Submitting shows alert with values.

Q12. Create a JavaScript program to illustrate the use of Promise and Resolve

1. Creation Steps (Terminal)

mkdir js-q12 && cd js-q12

2. Folder Structure

```
js-q12/
└── promise.js
```

3. Required Installation

- Node.js only

4. Files to Create

- promise.js

5. DB Setup

- ❌ Not required

6. Code

promise.js

```
function doWork() {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve("Task completed successfully"), 1000);
  });
}

doWork().then(result => console.log(result));
```

7. Running Steps

node promise.js

8. Expected Output

Task completed successfully

☑ That's **Set 4 (Q10–Q12)** fully regenerated.

👉 Do you want me to continue with **Set 5 (Q13–Q15)** in the same format?

Perfect Rashida 👐
Here's **Set 5 (Q13–Q15)** in the full exam format with localhost in Sequelize DB configs.

SET 5 (Q13–Q15)

Q13. Create a Node.js Express server to search an employee by email using MySQL (Sequelize)

1. Creation Steps (Terminal)

```
mkdir node-q13 && cd node-q13
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q13/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Employee.js
└── routes/
    └── employee.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Employee.js
- routes/employee.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS company;
USE company;
CREATE TABLE IF NOT EXISTS employees(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);
INSERT INTO employees(name,email) VALUES
('Ali','ali@mail.com'),('Sara','sara@mail.com');
```

EXIT;

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("company", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Employee.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Employee", {
  name: DataTypes.STRING,
  email: DataTypes.STRING
}, { tableName: "employees", timestamps: false });
```

routes/employee.js

```
const express = require("express");
const Employee = require("../models/Employee");
const router = express.Router();

router.get("/:email", async (req, res) => {
  const emp = await Employee.findOne({ where: { email: req.params.email }
});
  if (!emp) return res.status(404).json({ error: "Employee not found" });
  res.json(emp);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const employeeRoutes = require("./routes/employee");

const app = express();
app.use("/employees", employeeRoutes);

sequelize.sync().then(() => {
  app.listen(3005, () => console.log("☑ Server running on
http://localhost:3005"));
});
```

7. Running Steps

node server.js
curl http://localhost:3005/employees/ali@mail.com

8. Expected Output

{"id":1,"name":"Ali","email":"ali@mail.com"}

Q14. Create a Register component using React.js

1. Creation Steps (Terminal)

npx create-react-app react-q14
cd react-q14

2. Folder Structure

react-q14/
└── src/
    └── components/
        └── Register.js

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Register.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Register.js

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

export default function Register(){
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const change = (e) => setForm({ ...form, [e.target.name]: e.target.value });
  const submit = (e) => {
    e.preventDefault();
    alert(`Name: ${form.name}, Email: ${form.email}, Password:
${form.password}`);
```

```
  };

  return (
    <form onSubmit={submit}>
      <h2>Register</h2>
      <input name="name" placeholder="Name" onChange={change} />
      <input name="email" type="email" placeholder="Email"
onChange={change} />
      <input name="password" type="password" placeholder="Password"
onChange={change} />
      <button type="submit">Register</button>
    </form>
  );
}
```

src/App.js

```
import Register from "./components/Register";

export default function App(){
  return <Register />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

Form with name, email, and password fields. Submitting shows an alert with entered values.

Q15. Create a JavaScript program to illustrate the use of Async and Await

1. Creation Steps (Terminal)

```
mkdir js-q15 && cd js-q15
```

2. Folder Structure

```
js-q15/
 └── async.js
```

3. Required Installation

- Node.js only

4. Files to Create

- async.js

5. DB Setup

- ❌ Not required

6. Code

async.js

```
function delay(ms){
  return new Promise(resolve => setTimeout(resolve, ms));
}

async function run(){
  console.log("Start");
  await delay(1000);
  console.log("End after 1 second");
}

run();
```

7. Running Steps

```
node async.js
```

8. Expected Output

```
Start
End after 1 second
```

SET 6 (Q16–Q18)

Q16. Create a Node.js Express server to insert employee into database (Sequelize + MySQL)

1. Creation Steps (Terminal)

```
mkdir node-q16 && cd node-q16
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q16/
 ├── package.json
 ├── server.js
 ├── config/
```

```
│   └── db.js
├── models/
│   └── Employee.js
└── routes/
    └── employee.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Employee.js
- routes/employee.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS company;
USE company;
CREATE TABLE IF NOT EXISTS employees(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("company", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Employee.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Employee", {
  name: DataTypes.STRING,
  email: DataTypes.STRING
}, { tableName: "employees", timestamps: false });
```

routes/employee.js

```
const express = require("express");
const Employee = require("../models/Employee");
const router = express.Router();

router.post("/", async (req, res) => {
  const emp = await Employee.create(req.body);
  res.json(emp);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const employeeRoutes = require("./routes/employee");

const app = express();
app.use(express.json());
app.use("/employees", employeeRoutes);

sequelize.sync().then(() => {
  app.listen(3006, () => console.log("☑ Server running on
http://localhost:3006"));
});
```

7. Running Steps

```
node server.js
curl -X POST http://localhost:3006/employees -H "Content-Type:
application/json" \
 -d '{"name":"Sara","email":"sara@mail.com"}'
```

8. Expected Output

```
{"id":1,"name":"Sara","email":"sara@mail.com"}
```

Q17. Create a React component to render user name and update when it changes (useState + useEffect)

1. Creation Steps (Terminal)

```
npx create-react-app react-q17
cd react-q17
```

2. Folder Structure

```
react-q17/
 └── src/
     └── components/
```

└── User.js

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/User.js

5. DB Setup

- ❌ Not required

6. Code

src/components/User.js

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

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

  useEffect(() => {
    document.title = `Hello ${name}`;
  }, [name]);

  return (
    <div>
      <h2>User: {name}</h2>
      <input
        value={name}
        onChange={(e) => setName(e.target.value)}
        placeholder="Enter name"
      />
    </div>
  );
}
```

src/App.js

```
import User from "./components/User";

export default function App(){
  return <User />;
}
```

7. Running Steps

npm start

8. Expected Output

- Initially shows **User: Guest**
- Typing in input updates the displayed name and also updates the browser tab title.

Q18. Create a React component to display list of employees (random data)

1. Creation Steps (Terminal)

```
npx create-react-app react-q18
cd react-q18
```

2. Folder Structure

```
react-q18/
 └── src/
     └── components/
         └── EmployeeList.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/EmployeeList.js

5. DB Setup

- ❌ Not required

6. Code

src/components/EmployeeList.js

```
export default function EmployeeList(){
  const employees = [
    { id: 1, name: "Ali", email: "ali@mail.com" },
    { id: 2, name: "Sara", email: "sara@mail.com" },
    { id: 3, name: "Rashida", email: "rashida@mail.com" }
  ];

  return (
    <div>
      <h2>Employees</h2>
      <ul>
```

```
        {employees.map(e => (
          <li key={e.id}>{e.name} - {e.email}</li>
        ))}
      </ul>
    </div>
  );
}
```

src/App.js

```
import EmployeeList from "./components/EmployeeList";

export default function App(){
  return <EmployeeList />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A webpage showing a list of employees with names and emails.

SET 7 (Q19–Q21)

Q19. Build a simple multi-page React application using React Router

1. Creation Steps (Terminal)

```
npx create-react-app react-q19
cd react-q19
npm install react-router-dom
```

2. Folder Structure

```
react-q19/
 └── src/
     ├── App.js
     └── pages/
         ├── Home.js
         ├── About.js
         └── Contact.js
```

3. Required Installation

- Node.js

- React (create-react-app)
- React Router DOM

4. Files to Create

- src/pages/Home.js
- src/pages/About.js
- src/pages/Contact.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/pages/Home.js

```
export default function Home(){
  return <h2>Home Page</h2>;
}
```

src/pages/About.js

```
export default function About(){
  return <h2>About Page</h2>;
}
```

src/pages/Contact.js

```
export default function Contact(){
  return <h2>Contact Page</h2>;
}
```

src/App.js

```
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import Home from "./pages/Home";
import About from "./pages/About";
import Contact from "./pages/Contact";

export default function App(){
  return (
    <BrowserRouter>
      <nav>
        <Link to="/">Home</Link> |{" "}
        <Link to="/about">About</Link> |{" "}
        <Link to="/contact">Contact</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home/>}/>
```

```
        <Route path="/about" element={<About/>}/>
        <Route path="/contact" element={<Contact/>}/>
      </Routes>
    </BrowserRouter>
  );
}
```

7. Running Steps

npm start

8. Expected Output

A React app with **Home**, **About**, and **Contact** pages navigated via links.

Q20. Implement a simple dropdown menu using useRef in React

1. Creation Steps (Terminal)

npx create-react-app react-q20
cd react-q20

2. Folder Structure

```
react-q20/
 └── src/
     └── components/
         └── Dropdown.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Dropdown.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Dropdown.js

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

```
export default function Dropdown(){
  const ref = useRef();
  const [open, setOpen] = useState(false);

  useEffect(() => {
    const handler = (e) => {
      if (ref.current && !ref.current.contains(e.target)) setOpen(false);
    };
    document.addEventListener("click", handler);
    return () => document.removeEventListener("click", handler);
  }, []);

  return (
    <div ref={ref}>
      <button onClick={() => setOpen(!open)}>Menu</button>
      {open && (
        <ul>
          <li>Option A</li>
          <li>Option B</li>
        </ul>
      )}
    </div>
  );
}
```

src/App.js

```
import Dropdown from "./components/Dropdown";

export default function App(){
  return <Dropdown />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A button “Menu” that toggles a dropdown list with two options. Clicking outside closes the dropdown.

Q21. Create login and register components and routes using React Router

1. Creation Steps (Terminal)

```
npx create-react-app react-q21
cd react-q21
```

npm install react-router-dom

2. Folder Structure

```
react-q21/
 └── src/
     ├── App.js
     └── components/
         ├── Login.js
         └── Register.js
```

3. Required Installation

- Node.js
- React
- React Router DOM

4. Files to Create

- src/components/Login.js
- src/components/Register.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Login.js

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

export default function Login(){
  const [form, setForm] = useState({ email: "", password: "" });

  const handleChange = (e) => setForm({ ...form, [e.target.name]:
e.target.value });
  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Login with ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Login</h2>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Login</button>
```

```
    </form>
  );
}
```

src/components/Register.js

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

export default function Register(){
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const handleChange = (e) => setForm({ ...form, [e.target.name]:
e.target.value });
  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Registered: ${form.name}, ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Register</h2>
      <input name="name" placeholder="Name" onChange={handleChange}/>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Register</button>
    </form>
  );
}
```

src/App.js

```
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import Login from "./components/Login";
import Register from "./components/Register";

export default function App(){
  return (
    <BrowserRouter>
      <nav>
        <Link to="/login">Login</Link> | <Link to="/register">Register</Link>
      </nav>
      <Routes>
        <Route path="/login" element={<Login/>}/>
        <Route path="/register" element={<Register/>}/>
      </Routes>
    </BrowserRouter>
  );
}
```

7. Running Steps

npm start

8. Expected Output

- Navigation links: Login | Register
- Clicking Login → shows login form
- Clicking Register → shows register form

SET 8 (Q22–Q24)

Q22. Create a Node.js Express server to list all employees from the employee table

1. Creation Steps (Terminal)

```
mkdir node-q22 && cd node-q22
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q22/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Employee.js
└── routes/
    └── employee.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Employee.js
- routes/employee.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS company;
USE company;
CREATE TABLE IF NOT EXISTS employees(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);
INSERT INTO employees(name,email) VALUES
('Ali','ali@mail.com'),('Sara','sara@mail.com');
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("company", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Employee.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Employee", {
  name: DataTypes.STRING,
  email: DataTypes.STRING
}, { tableName: "employees", timestamps: false });
```

routes/employee.js

```
const express = require("express");
const Employee = require("../models/Employee");
const router = express.Router();

router.get("/", async (_req, res) => {
  const employees = await Employee.findAll();
  res.json(employees);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const employeeRoutes = require("./routes/employee");
```

```
const app = express();
app.use("/employees", employeeRoutes);

sequelize.sync().then(() => {
  app.listen(3007, () => console.log("☑ Server running on
http://localhost:3007"));
});
```

7. Running Steps

```
node server.js
curl http://localhost:3007/employees
```

8. Expected Output

```
[
  {"id":1,"name":"Ali","email":"ali@mail.com"},
  {"id":2,"name":"Sara","email":"sara@mail.com"}
]
```

Q23. Create a React component to render username and update when it changes (useState + useEffect)

1. Creation Steps (Terminal)

```
npx create-react-app react-q23
cd react-q23
```

2. Folder Structure

```
react-q23/
 └── src/
     └── components/
         └── User.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/User.js

5. DB Setup

- ❌ Not required

6. Code

src/components/User.js

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

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

  useEffect(() => {
    document.title = `Welcome ${name}`;
  }, [name]);

  return (
    <div>
      <h2>User: {name}</h2>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
    </div>
  );
}
```

src/App.js

```
import User from "./components/User";

export default function App(){
  return <User />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

- Shows **User: Guest** initially
- Typing in input updates both the UI and the browser tab title.

Q24. Create a React component which will illustrate the use of useState

1. Creation Steps (Terminal)

```
npx create-react-app react-q24
cd react-q24
```

2. Folder Structure

```
react-q24/
 └── src/
     └── components/
         └── Counter.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Counter.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Counter.js

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

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

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

src/App.js

```
import Counter from "./components/Counter";

export default function App(){
  return <Counter />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A counter that increases/decreases when buttons are clicked.

SET 9 (Q25–Q27)

Q25. Create a Node.js Express server to provide Register REST API

1. Creation Steps (Terminal)

```
mkdir node-q25 && cd node-q25
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q25/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── User.js
└── routes/
    └── user.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/User.js
- routes/user.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS authdb;
USE authdb;
CREATE TABLE IF NOT EXISTS users(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50),
  password VARCHAR(100)
);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("authdb", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/User.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("User", {
  name: DataTypes.STRING,
  email: DataTypes.STRING,
  password: DataTypes.STRING
}, { tableName: "users", timestamps: false });
```

routes/user.js

```
const express = require("express");
const User = require("../models/User");
const router = express.Router();

router.post("/register", async (req, res) => {
  const user = await User.create(req.body);
  res.status(201).json(user);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const userRoutes = require("./routes/user");

const app = express();
app.use(express.json());
app.use("/api", userRoutes);

sequelize.sync().then(() => {
  app.listen(3008, () => console.log("☑ Server running on
http://localhost:3008"));
});
```

7. Running Steps

node server.js
curl -X POST http://localhost:3008/api/register -H "Content-Type: application/json" \
 -d '{"name":"Rashida","email":"rashida@mail.com","password":"123456"}'

8. Expected Output

{"id":1,"name":"Rashida","email":"rashida@mail.com","password":"123456"}

Q26. Create Login and Register components and routes using React Router

1. Creation Steps (Terminal)

npx create-react-app react-q26
cd react-q26
npm install react-router-dom

2. Folder Structure

react-q26/
└── src/
    ├── App.js
    └── components/
        ├── Login.js
        └── Register.js

3. Required Installation

- Node.js
- React
- React Router DOM

4. Files to Create

- src/components/Login.js
- src/components/Register.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Login.js

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

export default function Login(){
```

```
  const [form, setForm] = useState({ email: "", password: "" });

  const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value
});
  const handleSubmit = e => {
    e.preventDefault();
    alert(`Login with ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Login</h2>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Login</button>
    </form>
  );
}
```

src/components/Register.js

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

export default function Register(){
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const handleChange = e => setForm({ ...form, [e.target.name]: e.target.value
});
  const handleSubmit = e => {
    e.preventDefault();
    alert(`Registered ${form.name}, ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Register</h2>
      <input name="name" placeholder="Name" onChange={handleChange}/>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Register</button>
    </form>
  );
}
```

src/App.js

```
import { BrowserRouter, Routes, Route, Link } from "react-router-dom";
import Login from "./components/Login";
```

```
import Register from "./components/Register";

export default function App(){
  return (
    <BrowserRouter>
      <nav>
        <Link to="/login">Login</Link> | <Link to="/register">Register</Link>
      </nav>
      <Routes>
        <Route path="/login" element={<Login/>}/>
        <Route path="/register" element={<Register/>}/>
      </Routes>
    </BrowserRouter>
  );
}
```

7. Running Steps

```
npm start
```

8. Expected Output

- Navbar with **Login** and **Register** links
- Clicking shows respective forms

Q27. Implement a simple dropdown menu using useRef in React

1. Creation Steps (Terminal)

```
npx create-react-app react-q27
cd react-q27
```

2. Folder Structure

```
react-q27/
 └── src/
     └── components/
         └── Dropdown.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Dropdown.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Dropdown.js

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

export default function Dropdown(){
  const ref = useRef();
  const [open, setOpen] = useState(false);

  useEffect(() => {
    const handler = (e) => {
      if (ref.current && !ref.current.contains(e.target)) setOpen(false);
    };
    document.addEventListener("click", handler);
    return () => document.removeEventListener("click", handler);
  }, []);

  return (
    <div ref={ref}>
      <button onClick={() => setOpen(!open)}>Menu</button>
      {open && (
        <ul>
          <li>Option A</li>
          <li>Option B</li>
        </ul>
      )}
    </div>
  );
}
```

src/App.js

```
import Dropdown from "./components/Dropdown";

export default function App(){
  return <Dropdown />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A **Menu button** that toggles a dropdown with “Option A” and “Option B”.
Clicking outside closes the menu.

SET 10 (Q28–Q30)

Q28. Create a Node.js Express server to provide Login REST API

1. Creation Steps (Terminal)

```
mkdir node-q28 && cd node-q28
npm init -y
npm install express sequelize mysql2 bcryptjs jsonwebtoken
```

2. Folder Structure

```
node-q28/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── User.js
└── routes/
    └── user.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/User.js
- routes/user.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS authdb;
USE authdb;
CREATE TABLE IF NOT EXISTS users(
  id INT AUTO_INCREMENT PRIMARY KEY,
  email VARCHAR(50),
  password VARCHAR(100)
);

-- Insert one user (password: 123456)
INSERT INTO users(email,password) VALUES ("test@mail.com","123456");
```

EXIT;

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("authdb", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/User.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("User", {
  email: DataTypes.STRING,
  password: DataTypes.STRING
}, { tableName: "users", timestamps: false });
```

routes/user.js

```
const express = require("express");
const User = require("../models/User");
const jwt = require("jsonwebtoken");

const router = express.Router();
const SECRET = "exam-secret";

router.post("/login", async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ where: { email } });
  if (!user || user.password !== password) {
    return res.status(401).json({ error: "Invalid credentials" });
  }
  const token = jwt.sign({ email }, SECRET, { expiresIn: "1h" });
  res.json({ token });
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const userRoutes = require("./routes/user");

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

```
app.use("/api", userRoutes);

sequelize.sync().then(() => {
  app.listen(3009, () => console.log("☑ Server running on
http://localhost:3009"));
});
```

7. Running Steps

```
node server.js
curl -X POST http://localhost:3009/api/login -H "Content-Type: application/json"
\
 -d '{"email":"test@mail.com","password":"123456"}'
```

8. Expected Output

```
{"token":"<JWT_TOKEN>"}
```

Q29. Create a React component to display list of employees (random data)

1. Creation Steps (Terminal)

```
npx create-react-app react-q29
cd react-q29
```

2. Folder Structure

```
react-q29/
 └── src/
     └── components/
         └── EmployeeList.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/EmployeeList.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/EmployeeList.js

```
export default function EmployeeList(){
  const employees = [
    { id: 1, name: "Ali", dept: "HR" },
    { id: 2, name: "Sara", dept: "Finance" },
    { id: 3, name: "Rashida", dept: "IT" }
  ];

  return (
    <div>
      <h2>Employee List</h2>
      <ul>
        {employees.map(e => (
          <li key={e.id}>{e.name} - {e.dept}</li>
        ))}
      </ul>
    </div>
  );
}
```

src/App.js

```
import EmployeeList from "./components/EmployeeList";

export default function App(){
  return <EmployeeList />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A webpage showing employees with their department.

Q30. Create a React component to render username and update on change (useEffect + useState)

1. Creation Steps (Terminal)

```
npx create-react-app react-q30
cd react-q30
```

2. Folder Structure

```
react-q30/
 └── src/
     └── components/
         └── User.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/User.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/User.js

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

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

  useEffect(() => {
    document.title = `Hello, ${name}`;
  }, [name]);

  return (
    <div>
      <h2>User: {name}</h2>
      <input
        type="text"
        value={name}
        onChange={(e) => setName(e.target.value)}
      />
    </div>
  );
}
```

src/App.js

```
import User from "./components/User";

export default function App(){
  return <User />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

- Initially shows **User: Guest**
- Typing updates username in UI and browser title

SET 11 (Q31–Q33)

Q31. Create a Node.js file that Selects all records from the "customers" table, and deletes a specified record

1. Creation Steps (Terminal)

```
mkdir node-q31 && cd node-q31
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q31/
 └── app.js
```

3. Required Installation

- Node.js
- Sequelize
- MySQL2

4. Files to Create

- app.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS shop;
USE shop;
CREATE TABLE IF NOT EXISTS customers(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);
INSERT INTO customers(name,email) VALUES
('Ali','ali@mail.com'),('Sara','sara@mail.com');
EXIT;
```

6. Code

app.js

```
const { Sequelize, DataTypes } = require("sequelize");
const sequelize = new Sequelize("shop", "root", "", {
  host: "localhost",
  dialect: "mysql"
});

const Customer = sequelize.define("Customer", {
  name: DataTypes.STRING,
  email: DataTypes.STRING
}, { tableName: "customers", timestamps: false });

(async () => {
  await sequelize.sync();

  const all = await Customer.findAll();
  console.log("All customers:", all.map(c => c.toJSON()));

  const deleted = await Customer.destroy({ where: { id: 1 } });
  console.log("Deleted count:", deleted);
})();
```

7. Running Steps

node app.js

8. Expected Output

All customers: [{ id: 1, name: 'Ali', email: 'ali@mail.com' }, { id: 2, name: 'Sara', email: 'sara@mail.com' }]
Deleted count: 1

Q32. Create a JavaScript program to illustrate the use of async and await

1. Creation Steps (Terminal)

mkdir js-q32 && cd js-q32

2. Folder Structure

```
js-q32/
 └── async.js
```

3. Required Installation

- Node.js only

4. Files to Create

- async.js

5. DB Setup

- ❌ Not required

6. Code

async.js

```
function fetchData(){
  return new Promise(resolve => {
    setTimeout(() => resolve("Data received"), 1000);
  });
}

async function main(){
  console.log("Start");
  const data = await fetchData();
  console.log(data);
}

main();
```

7. Running Steps

```
node async.js
```

8. Expected Output

```
Start
Data received
```

Q33. Create a Register component using React.js (with appropriate fields)

1. Creation Steps (Terminal)

```
npx create-react-app react-q33
cd react-q33
```

2. Folder Structure

```
react-q33/
 └── src/
     └── components/
         └── Register.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Register.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Register.js

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

export default function Register(){
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const handleChange = (e) => setForm({ ...form, [e.target.name]:
e.target.value });
  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Registered ${form.name}, ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Register</h2>
      <input name="name" placeholder="Name" onChange={handleChange}/>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Register</button>
    </form>
  );
}
```

src/App.js

```
import Register from "./components/Register";

export default function App(){
  return <Register />;
}
```

7. Running Steps

npm start

8. Expected Output

A registration form with Name, Email, and Password fields. Submitting shows an alert with values.

SET 12 (Q34–Q36)

Q34. Create a Node.js file that inserts multiple records in "student" table, and display the result object on console

1. Creation Steps (Terminal)

```
mkdir node-q34 && cd node-q34
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q34/
 └── insert.js
```

3. Required Installation

- Node.js
- Sequelize
- MySQL2

4. Files to Create

- insert.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
CREATE TABLE IF NOT EXISTS students(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  marks INT
);
EXIT;
```

6. Code

insert.js

```
const { Sequelize, DataTypes } = require("sequelize");
const sequelize = new Sequelize("school", "root", "", {
  host: "localhost",
  dialect: "mysql"
});

const Student = sequelize.define("Student", {
  name: DataTypes.STRING,
  marks: DataTypes.INTEGER
}, { tableName: "students", timestamps: false });

(async () => {
  await sequelize.sync();
  const result = await Student.bulkCreate([
    { name: "Ali", marks: 70 },
    { name: "Sara", marks: 85 },
    { name: "Rashida", marks: 92 }
  ]);
  console.log("Inserted records:", result.map(r => r.toJSON()));
})();
```

7. Running Steps

node insert.js

8. Expected Output

```
Inserted records: [
 { id: 1, name: 'Ali', marks: 70 },
 { id: 2, name: 'Sara', marks: 85 },
 { id: 3, name: 'Rashida', marks: 92 }
]
```

Q35. Create a JavaScript program to illustrate the use of Promise and Resolve

1. Creation Steps (Terminal)

mkdir js-q35 && cd js-q35

2. Folder Structure

```
js-q35/
  └── promise.js
```

3. Required Installation

- Node.js

4. Files to Create

- promise.js

5. DB Setup

- ❌ Not required

6. Code

promise.js

```
function jobDone(){
  return new Promise(resolve => {
    setTimeout(() => resolve("Work finished"), 1000);
  });
}

jobDone().then(msg => console.log(msg));
```

7. Running Steps

node promise.js

8. Expected Output

Work finished

Q36. Create a Login component using React.js

1. Creation Steps (Terminal)

```
npx create-react-app react-q36
cd react-q36
```

2. Folder Structure

```
react-q36/
 └── src/
     └── components/
         └── Login.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Login.js

- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Login.js

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

export default function Login(){
  const [form, setForm] = useState({ email: "", password: "" });

  const handleChange = (e) => setForm({ ...form, [e.target.name]:
e.target.value });
  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Login: ${form.email}`);
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Login</h2>
      <input name="email" type="email" placeholder="Email"
onChange={handleChange}/>
      <input name="password" type="password" placeholder="Password"
onChange={handleChange}/>
      <button type="submit">Login</button>
    </form>
  );
}
```

src/App.js

```
import Login from "./components/Login";

export default function App(){
  return <Login />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A login form with email and password fields. Submitting shows an alert with entered email.

SET 13 (Q37–Q39)

Q37. Create a Node.js file that Select all records from the "customers" table, and display the result object on console

1. Creation Steps (Terminal)

```
mkdir node-q37 && cd node-q37
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q37/
 └── select.js
```

3. Required Installation

- Node.js
- Sequelize
- MySQL2

4. Files to Create

- select.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS shop;
USE shop;
CREATE TABLE IF NOT EXISTS customers(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50)
);
INSERT INTO customers(name,email) VALUES
('Ali','ali@mail.com'),('Sara','sara@mail.com');
EXIT;
```

6. Code

select.js

```
const { Sequelize, DataTypes } = require("sequelize");
```

```
const sequelize = new Sequelize("shop", "root", "", {
  host: "localhost",
  dialect: "mysql"
});

const Customer = sequelize.define("Customer", {
  name: DataTypes.STRING,
  email: DataTypes.STRING
}, { tableName: "customers", timestamps: false });

(async () => {
  await sequelize.sync();
  const customers = await Customer.findAll();
  console.log("All customers:", customers.map(c => c.toJSON()));
})();
```

7. Running Steps

```
node select.js
```

8. Expected Output

```
All customers: [
 { id: 1, name: 'Ali', email: 'ali@mail.com' },
 { id: 2, name: 'Sara', email: 'sara@mail.com' }
]
```

Q38. Create any React component to illustrate the React hook (useState)

1. Creation Steps (Terminal)

```
npx create-react-app react-q38
cd react-q38
```

2. Folder Structure

```
react-q38/
 └── src/
     └── components/
         └── Counter.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Counter.js

- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Counter.js

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

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

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

src/App.js

```
import Counter from "./components/Counter";

export default function App(){
  return <Counter />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A counter that increments or decrements when buttons are clicked.

Q39. Create a Node.js file that demonstrates create database and table in MySQL

1. Creation Steps (Terminal)

```
mkdir node-q39 && cd node-q39
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q39/
 └── init.js
```

3. Required Installation

- Node.js
- Sequelize
- MySQL2

4. Files to Create

- init.js

5. DB Setup (MySQL)

```
mysql -u root -p -e "CREATE DATABASE IF NOT EXISTS demo;"
```

6. Code

init.js

```
const { Sequelize, DataTypes } = require("sequelize");

(async () => {
  const sequelize = new Sequelize("demo", "root", "", {
    host: "localhost",
    dialect: "mysql"
  });

  const Product = sequelize.define("Product", {
    name: DataTypes.STRING,
    price: DataTypes.FLOAT
  }, { tableName: "products", timestamps: false });

  await sequelize.sync();
  console.log("☑ Database & Table created successfully");
})();
```

7. Running Steps

```
node init.js
```

8. Expected Output

☑ Database & Table created successfully

SET 14 (Q40–Q42)

Q40. Create a Node.js file that writes an HTML form, with an upload field

1. Creation Steps (Terminal)

```
mkdir node-q40 && cd node-q40
npm init -y
npm install express multer
```

2. Folder Structure

```
node-q40/
 ├── package.json
 ├── server.js
 └── uploads/
```

3. Required Installation

- Node.js
- Express
- Multer (for file upload handling)

4. Files to Create

- server.js

5. DB Setup

- ❌ Not required

6. Code

server.js

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

const app = express();
const upload = multer({ dest: "uploads/" });

app.get("/", (req, res) => {
  res.send(`
    <form action="/upload" method="post" enctype="multipart/form-data">
      <input type="file" name="file"/>
      <button type="submit">Upload</button>
    </form>
  `);
});
```

```
app.post("/upload", upload.single("file"), (req, res) => {
  res.send(`File uploaded: ${req.file.originalname}`);
});

app.listen(3010, () => console.log("☑ Server running on
http://localhost:3010"));
```

7. Running Steps

node server.js

Open http://localhost:3010 in browser, upload a file.

8. Expected Output

File uploaded: myfile.txt

Q41. Create JavaScript application to illustrate the use of callback under functions

1. Creation Steps (Terminal)

mkdir js-q41 && cd js-q41

2. Folder Structure

```
js-q41/
└── callback.js
```

3. Required Installation

- Node.js only

4. Files to Create

- callback.js

5. DB Setup

- ❌ Not required

6. Code

callback.js

```
function calculate(a, b, callback){
  console.log("Calculating...");
  setTimeout(() => {
    const sum = a + b;
    callback(sum);
```

```
  }, 1000);
}

calculate(5, 7, result => {
  console.log("Result:", result);
});
```

7. Running Steps

```
node callback.js
```

8. Expected Output

```
Calculating...
Result: 12
```

Q42. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error

1. Creation Steps (Terminal)

```
mkdir node-q42 && cd node-q42
npm init -y
npm install express
```

2. Folder Structure

```
node-q42/
 └── server.js
```

3. Required Installation

- Node.js
- Express

4. Files to Create

- server.js

5. DB Setup

- ❌ Not required

6. Code

server.js

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

```
const app = express();

app.get("/:file", (req, res) => {
  const filePath = path.join(__dirname, req.params.file);
  fs.readFile(filePath, "utf8", (err, data) => {
    if (err) return res.status(404).send("404 File Not Found");
    res.type("text/plain").send(data);
  });
});

app.listen(3011, () => console.log("☑ Server running on
http://localhost:3011"));
```

7. Running Steps

```
echo "Hello Rashida" > hello.txt
node server.js
curl http://localhost:3011/hello.txt
```

8. Expected Output

```
Hello Rashida
```

SET 15 (Q43–Q45)

Q43. Create an HTML form for Login and write a JavaScript to validate email ID using Regular Expression

1. Creation Steps (Terminal)

```
mkdir q43 && cd q43
```

2. Folder Structure

```
q43/
 └── index.html
```

3. Required Installation

- Only a browser (no server required)

4. Files to Create

- index.html

5. DB Setup

- ❌ Not required

6. Code

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <script>
    function validateEmail() {
      const email = document.getElementById("email").value;
      const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
      if (!regex.test(email)) {
        alert("Invalid Email ID");
        return false;
      }
      alert("Valid Email ID");
      return true;
    }
  </script>
</head>
<body>
  <form onsubmit="return validateEmail()">
    <h2>Login</h2>
    <input type="email" id="email" placeholder="Enter Email" required/>
    <input type="password" placeholder="Enter Password" required/>
    <button type="submit">Login</button>
  </form>
</body>
</html>
```

7. Running Steps

- Open index.html in browser

8. Expected Output

- Entering rashida@com → Invalid Email
- Entering rashida@mail.com → Valid Email

Q44. Create a React component to display list of students (random data)

1. Creation Steps (Terminal)

```
npx create-react-app react-q44
cd react-q44
```

2. Folder Structure

```
react-q44/
 └── src/
     └── components/
         └── StudentList.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/StudentList.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/StudentList.js

```
export default function StudentList(){
  const students = [
    { id: 1, name: "Ali", marks: 75 },
    { id: 2, name: "Sara", marks: 82 },
    { id: 3, name: "Rashida", marks: 65 }
  ];

  return (
    <div>
      <h2>Students</h2>
      <ul>
        {students.map(s => (
          <li key={s.id}>{s.name} - {s.marks}</li>
        ))}
      </ul>
    </div>
  );
}
```

src/App.js

```
import StudentList from "./components/StudentList";

export default function App(){
  return <StudentList />;
}
```

7. Running Steps

npm start

8. Expected Output

Students
Ali - 75
Sara - 82
Rashida - 65

Q45. Create a Node.js file that will convert the output "Hello World!" into uppercase

1. Creation Steps (Terminal)

mkdir node-q45 && cd node-q45

2. Folder Structure

node-q45/
└── app.js

3. Required Installation

- Node.js

4. Files to Create

- app.js

5. DB Setup

- ❌ Not required

6. Code

app.js

```
console.log("Hello World!".toUpperCase());
```

7. Running Steps

node app.js

8. Expected Output

HELLO WORLD!

☑ That's **Set 15 (Q43–Q45)** complete.

👉 Do you want me to continue with **Set 16 (Q46–Q48)** next?

Perfect Rashida 👐
Here's **Set 16 (Q46–Q48)** in the same exam-ready style with Sequelize using localhost.

SET 16 (Q46–Q48)

Q46. Create an HTML form that contains the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary

1. Creation Steps (Terminal)

mkdir q46 && cd q46

2. Folder Structure

q46/
└── index.html

3. Required Installation

- Browser only

4. Files to Create

- index.html

5. DB Setup

- ❌ Not required

6. Code

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Employee Registration</title>
  <script>
    function validateForm() {
```

```
      const dob = new Date(document.getElementById("dob").value);
      const joinDate = new Date(document.getElementById("join").value);
      const salary = parseFloat(document.getElementById("salary").value);

      if (dob >= joinDate) {
        alert("Joining date must be after DOB");
        return false;
      }
      if (salary <= 0) {
        alert("Salary must be greater than 0");
        return false;
      }
      alert("Form submitted successfully!");
      return true;
    }
  </script>
</head>
<body>
  <form onsubmit="return validateForm()">
    <h2>Employee Registration</h2>
    <label>DOB: <input type="date" id="dob" required></label><br>
    <label>Joining Date: <input type="date" id="join" required></label><br>
    <label>Salary: <input type="number" id="salary" required></label><br>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

7. Running Steps

- Open index.html in a browser

8. Expected Output

- If DOB ≥ Joining Date → Error
- If Salary ≤ 0 → Error
- Else → “Form submitted successfully!”

Q47. Create a Node.js file that will convert "Hello World!" into uppercase

1. Creation Steps (Terminal)

```
mkdir node-q47 && cd node-q47
```

2. Folder Structure

```
node-q47/
 └── app.js
```

3. Required Installation

- Node.js only

4. Files to Create

- app.js

5. DB Setup

- ❌ Not required

6. Code

app.js

```
console.log("Hello World!".toUpperCase());
```

7. Running Steps

node app.js

8. Expected Output

HELLO WORLD!

Q48. Create a React component to display list of students (random data)

1. Creation Steps (Terminal)

```
npx create-react-app react-q48
cd react-q48
```

2. Folder Structure

```
react-q48/
 └── src/
     └── components/
         └── StudentList.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/StudentList.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/StudentList.js

```
export default function StudentList(){
  const students = [
    { id: 1, name: "Ali", marks: 78 },
    { id: 2, name: "Sara", marks: 88 },
    { id: 3, name: "Rashida", marks: 92 }
  ];

  return (
    <div>
      <h2>Students</h2>
      <ul>
        {students.map(s => (
          <li key={s.id}>{s.name} - {s.marks}</li>
        ))}
      </ul>
    </div>
  );
}
```

src/App.js

```
import StudentList from "./components/StudentList";

export default function App(){
  return <StudentList />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

```
Students
Ali - 78
Sara - 88
Rashida - 92
```

SET 17 (Q49–Q51)

Q49. Create an HTML form that contains the Student Registration details and write a JavaScript to validate Student first and last name (alphabets only) and age (between 18–50)

1. Creation Steps (Terminal)

mkdir q49 && cd q49

2. Folder Structure

q49/
└── index.html

3. Required Installation

- Browser only

4. Files to Create

- index.html

5. DB Setup

- ❌ Not required

6. Code

index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Registration</title>
  <script>
    function validateForm() {
      const fname = document.getElementById("fname").value;
      const lname = document.getElementById("lname").value;
      const age = parseInt(document.getElementById("age").value);

      const regex = /^[A-Za-z]+$/;
      if (!regex.test(fname) || !regex.test(lname)) {
        alert("First and Last name must contain only alphabets");
        return false;
      }
      if (age < 18 || age > 50) {
        alert("Age must be between 18 and 50");
        return false;
      }
      alert("Form submitted successfully!");
      return true;
    }
```

```
  </script>
</head>
<body>
  <form onsubmit="return validateForm()">
    <h2>Student Registration</h2>
    <input id="fname" placeholder="First Name" required><br>
    <input id="lname" placeholder="Last Name" required><br>
    <input id="age" type="number" placeholder="Age" required><br>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

7. Running Steps

- Open index.html in browser

8. Expected Output

- If invalid → Error message
- If valid → "Form submitted successfully!"

Q50. Create a Node.js file that opens the requested file and returns the content to the client. If anything goes wrong, throw a 404 error

1. Creation Steps (Terminal)

```
mkdir node-q50 && cd node-q50
npm init -y
npm install express
```

2. Folder Structure

```
node-q50/
 └── server.js
```

3. Required Installation

- Node.js
- Express

4. Files to Create

- server.js

5. DB Setup

- ❌ Not required

6. Code

server.js

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

const app = express();

app.get("/:file", (req, res) => {
  const filePath = path.join(__dirname, req.params.file);
  fs.readFile(filePath, "utf8", (err, data) => {
    if (err) return res.status(404).send("404 File Not Found");
    res.type("text/plain").send(data);
  });
});

app.listen(3012, () => console.log("☑ Server running on
http://localhost:3012"));
```

7. Running Steps

```
echo "Hello Rashida" > test.txt
node server.js
curl http://localhost:3012/test.txt
```

8. Expected Output

```
Hello Rashida
```

Q51. Create a JavaScript application to illustrate the use of callback under functions

1. Creation Steps (Terminal)

```
mkdir js-q51 && cd js-q51
```

2. Folder Structure

```
js-q51/
 └── callback.js
```

3. Required Installation

- Node.js only

4. Files to Create

- callback.js

5. DB Setup

- ❌ Not required

6. Code

callback.js

```
function greet(name, callback){
  console.log("Preparing greeting...");
  setTimeout(() => {
    callback(`Hello, ${name}!`);
  }, 1000);
}

greet("Rashida", message => {
  console.log(message);
});
```

7. Running Steps

node callback.js

8. Expected Output

Preparing greeting...
Hello, Rashida!

SET 18 (Q52–Q54)

Q52. Create a Node.js server to list all the students having marks greater than 60 using MySQL (Sequelize)

1. Creation Steps (Terminal)

mkdir node-q52 && cd node-q52
npm init -y
npm install express sequelize mysql2

2. Folder Structure

```
node-q52/
 ├── package.json
 ├── server.js
 ├── config/
 │    └── db.js
 ├── models/
 │    └── Student.js
```

```
└── routes/
    └── student.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Student.js
- routes/student.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
CREATE TABLE IF NOT EXISTS students(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  marks INT
);
INSERT INTO students(name,marks) VALUES
('Ali',75),('Sara',82),('Rohit',55);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Student.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Student", {
  name: DataTypes.STRING,
  marks: DataTypes.INTEGER
}, { tableName: "students", timestamps: false });
```

routes/student.js

```
const express = require("express");
const { Op } = require("sequelize");
const Student = require("../models/Student");
const router = express.Router();

router.get("/", async (_req, res) => {
  const students = await Student.findAll({ where: { marks: { [Op.gt]: 60 } } });
  res.json(students);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const studentRoutes = require("./routes/student");

const app = express();
app.use("/students", studentRoutes);

sequelize.sync().then(() => {
  app.listen(3013, () => console.log("☑ Server running on
http://localhost:3013"));
});
```

7. Running Steps

```
node server.js
curl http://localhost:3013/students
```

8. Expected Output

```
[
  {"id":1,"name":"Ali","marks":75},
  {"id":2,"name":"Sara","marks":82}
]
```

Q53. Create a Node.js file that demonstrates create database and table in MySQL

1. Creation Steps (Terminal)

```
mkdir node-q53 && cd node-q53
npm init -y
npm install sequelize mysql2
```

2. Folder Structure

```
node-q53/
```

└── init.js

3. Required Installation

- Node.js
- Sequelize
- MySQL2

4. Files to Create

- init.js

5. DB Setup

- ❌ Not required manually (Sequelize will handle creation)

6. Code

init.js

```
const { Sequelize, DataTypes } = require("sequelize");

(async () => {
  const sequelize = new Sequelize("testdb", "root", "", {
    host: "localhost",
    dialect: "mysql"
  });

  const Teacher = sequelize.define("Teacher", {
    name: DataTypes.STRING,
    subject: DataTypes.STRING
  }, { tableName: "teachers", timestamps: false });

  await sequelize.sync();
  console.log("☑ Database & teachers table created");
})();
```

7. Running Steps

node init.js

8. Expected Output

☑ Database & teachers table created

Q54. Create any React component to illustrate the React hook (useState)

1. Creation Steps (Terminal)

```
npx create-react-app react-q54
cd react-q54
```

2. Folder Structure

```
react-q54/
 └── src/
     └── components/
         └── Counter.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Counter.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Counter.js

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

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

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

src/App.js

```
import Counter from "./components/Counter";

export default function App(){
  return <Counter />;
}
```

7. Running Steps

npm start

8. Expected Output

A counter app with a button to increase the number.

SET 19 (Q55–Q57)

Q55. Create a Node.js server to delete student record by roll no using MySQL (Sequelize)

1. Creation Steps (Terminal)

```
mkdir node-q55 && cd node-q55
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q55/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Student.js
└── routes/
    └── student.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Student.js
- routes/student.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
```

```
CREATE TABLE IF NOT EXISTS students(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  rollno INT UNIQUE,
  marks INT
);
INSERT INTO students(name,rollno,marks) VALUES
('Ali',101,75),('Sara',102,55),('Rashida',103,85);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Student.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Student", {
  name: DataTypes.STRING,
  rollno: DataTypes.INTEGER,
  marks: DataTypes.INTEGER
}, { tableName: "students", timestamps: false });
```

routes/student.js

```
const express = require("express");
const Student = require("../models/Student");
const router = express.Router();

router.delete("/:rollno", async (req, res) => {
  const deleted = await Student.destroy({ where: { rollno: req.params.rollno }
});
  res.json({ deleted });
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const studentRoutes = require("./routes/student");
```

```
const app = express();
app.use("/students", studentRoutes);

sequelize.sync().then(() => {
  app.listen(3014, () => console.log("☑ Server running on
http://localhost:3014"));
});
```

7. Running Steps

```
node server.js
curl -X DELETE http://localhost:3014/students/102
```

8. Expected Output

```
{"deleted":1}
```

Q56. Create a Login component using React.js (with appropriate fields)

1. Creation Steps (Terminal)

```
npx create-react-app react-q56
cd react-q56
```

2. Folder Structure

```
react-q56/
 └── src/
     └── components/
         └── Login.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Login.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Login.js

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

```
export default function Login(){
  const [form, setForm] = useState({ email: "", password: "" });

  const change = e => setForm({ ...form, [e.target.name]: e.target.value });
  const submit = e => {
    e.preventDefault();
    alert(`Login: ${form.email}`);
  };

  return (
    <form onSubmit={submit}>
      <h2>Login</h2>
      <input type="email" name="email" placeholder="Email"
onChange={change}/>
      <input type="password" name="password" placeholder="Password"
onChange={change}/>
      <button type="submit">Login</button>
    </form>
  );
}
```

src/App.js

```
import Login from "./components/Login";

export default function App(){
  return <Login />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

Login form with email and password fields → shows alert with entered email when submitted.

Q57. Create a JavaScript program to illustrate the use of Promise and Resolve

1. Creation Steps (Terminal)

```
mkdir js-q57 && cd js-q57
```

2. Folder Structure

```
js-q57/
```

└── promise.js

3. Required Installation

- Node.js

4. Files to Create

- promise.js

5. DB Setup

- ❌ Not required

6. Code

promise.js

```
function task(){
  return new Promise(resolve => {
    setTimeout(() => resolve("Task Completed"), 1000);
  });
}

task().then(msg => console.log(msg));
```

7. Running Steps

node promise.js

8. Expected Output

Task Completed

SET 20 (Q58–Q60)

Q58. Create a Node.js server to insert teacher data into teacher table using MySQL (Sequelize)

1. Creation Steps (Terminal)

```
mkdir node-q58 && cd node-q58
npm init -y
npm install express sequelize mysql2
```

2. Folder Structure

```
node-q58/
├── package.json
├── server.js
├── config/
│   └── db.js
├── models/
│   └── Teacher.js
└── routes/
    └── teacher.js
```

3. Required Installation

```
sudo apt update
sudo apt install -y nodejs npm mysql-server
```

4. Files to Create

- config/db.js
- models/Teacher.js
- routes/teacher.js
- server.js

5. DB Setup (MySQL)

```
mysql -u root -p
CREATE DATABASE IF NOT EXISTS school;
USE school;
CREATE TABLE IF NOT EXISTS teachers(
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  subject VARCHAR(50)
);
EXIT;
```

6. Code

config/db.js

```
const { Sequelize } = require("sequelize");
module.exports = new Sequelize("school", "root", "", {
  host: "localhost",
  dialect: "mysql"
});
```

models/Teacher.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/db");

module.exports = sequelize.define("Teacher", {
  name: DataTypes.STRING,
```

```
  subject: DataTypes.STRING
}, { tableName: "teachers", timestamps: false });
```

routes/teacher.js

```
const express = require("express");
const Teacher = require("../models/Teacher");
const router = express.Router();

router.post("/", async (req, res) => {
  const teacher = await Teacher.create(req.body);
  res.status(201).json(teacher);
});

module.exports = router;
```

server.js

```
const express = require("express");
const sequelize = require("./config/db");
const teacherRoutes = require("./routes/teacher");

const app = express();
app.use(express.json());
app.use("/teachers", teacherRoutes);

sequelize.sync().then(() => {
  app.listen(3015, () => console.log("☑ Server running on
http://localhost:3015"));
});
```

7. Running Steps

```
node server.js
curl -X POST http://localhost:3015/teachers -H "Content-Type: application/json"
\
 -d '{"name":"Ayesha","subject":"Physics"}'
```

8. Expected Output

```
{"id":1,"name":"Ayesha","subject":"Physics"}
```

Q59. Create a Register component using React.js

1. Creation Steps (Terminal)

```
npx create-react-app react-q59
cd react-q59
```

2. Folder Structure

```
react-q59/
 └── src/
     └── components/
         └── Register.js
```

3. Required Installation

- Node.js
- React

4. Files to Create

- src/components/Register.js
- src/App.js

5. DB Setup

- ❌ Not required

6. Code

src/components/Register.js

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

export default function Register(){
  const [form, setForm] = useState({ name: "", email: "", password: "" });

  const change = e => setForm({ ...form, [e.target.name]: e.target.value });
  const submit = e => {
    e.preventDefault();
    alert(`Registered: ${form.name}, ${form.email}`);
  };

  return (
    <form onSubmit={submit}>
      <h2>Register</h2>
      <input name="name" placeholder="Name" onChange={change}/>
      <input name="email" type="email" placeholder="Email"
onChange={change}/>
      <input name="password" type="password" placeholder="Password"
onChange={change}/>
      <button type="submit">Register</button>
    </form>
  );
}
```

src/App.js

```
import Register from "./components/Register";
```

```
export default function App(){
  return <Register />;
}
```

7. Running Steps

```
npm start
```

8. Expected Output

A registration form with Name, Email, and Password fields. Shows alert on submit.

Q60. Create a JavaScript program to illustrate the use of async and await

1. Creation Steps (Terminal)

```
mkdir js-q60 && cd js-q60
```

2. Folder Structure

```
js-q60/
 └── async.js
```

3. Required Installation

- Node.js

4. Files to Create

- async.js

5. DB Setup

- ❌ Not required

6. Code

async.js

```
function delay(ms){
  return new Promise(resolve => setTimeout(resolve, ms));
}

async function runTask(){
  console.log("Start Task");
  await delay(1000);
  console.log("Task Completed");
```

```
}

runTask();
```

7. Running Steps

```
node async.js
```

8. Expected Output

```
Start Task
Task Completed
```
