**SET 1 (Q1–Q3)**

**Q1. Node.js (Sequelize) — List students with marks > 60**

**Required Installation**

sudo apt update

sudo apt install -y mysql-server nodejs npm

mkdir node-q1 && cd node-q1

npm init -y

npm i express sequelize mysql2

**📂 Folder Structure**

node-q1/

├── package.json

├── server.js

├── config/

│ └── db.js

├── models/

│ └── Student.js

└── routes/

└── student.js

**DB Setup (Terminal)**

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;

**Files + 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 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("http://localhost:3000"));

});

**Run**

cd node-q1

node server.js

curl http://localhost:3000/students

**Expected**

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

**Q2. Node.js — Convert “Hello World!” to uppercase**

**Install & Structure**

node-q2/

└── app.js

app.js

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

**Run**

node app.js

**Expected**

HELLO WORLD!

**Q3. React — List of students (random data)**

**Install**

npx create-react-app react-q3

cd react-q3

**📂 Structure**

react-q3/

└── src/

├── App.js

└── components/

└── StudentList.js

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/>; }

**Run**

npm start

**Expected**: Page shows list.

**SET 2 (Q4–Q6)**

**Q4. Node.js (Sequelize) — Delete student by roll no (id)**

**Install & Structure**

node-q4/

├── server.js

├── config/db.js

├── models/Student.js

└── routes/student.js

*(reuse Q1’s db.js and Student.js)*

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("http://localhost:3001")));

**DB Step**: use same school.students from Q1.

**Run**

node server.js

curl -X DELETE http://localhost:3001/students/3

**Expected**

{"deleted":1}

**Q5. Node.js — Open requested file, else 404**

**Install & Structure**

node-q5/

└── server.js

cd node-q5

npm init -y

npm i express

server.js

const express = require("express");

const fs = require("fs");

const path = require("path");

const app = express();

app.get("/:file", (req, res) => {

const p = path.join(\_\_dirname, req.params.file);

fs.readFile(p, "utf8", (err, data) => {

if (err) return res.status(404).send("404 Not Found");

res.type("text/plain").send(data);

});

});

app.listen(3002, () => console.log("http://localhost:3002"));

**Run**

echo "sample" > hello.txt

node server.js

curl http://localhost:3002/hello.txt

**Expected**

sample

**Q6. JavaScript — Callback demo**

**Structure**

js-q6/

└── callback.js

callback.js

function fetchData(cb){

setTimeout(() => cb(null, { ok: true, data: [1,2,3] }), 300);

}

fetchData((err, result) => {

if (err) return console.error(err);

console.log("Callback got:", result);

});

**Run**

node callback.js

**Expected**

Callback got: { ok: true, data: [ 1, 2, 3 ] }

**SET 3 (Q7–Q9)**

**Q7. Node.js (Sequelize) — Insert teacher data**

**Install & Structure**

node-q7/

├── server.js

├── config/db.js

├── models/Teacher.js

└── routes/teacher.js

cd node-q7

npm init -y

npm i express sequelize mysql2

**DB Setup**

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;

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 t = await Teacher.create(req.body);

res.status(201).json(t);

});

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("http://localhost:3003")));

**Run**

node server.js

curl -X POST http://localhost:3003/teachers -H "Content-Type: application/json" \

-d '{"name":"Ayesha","subject":"Math"}'

**Expected**

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

**Q8. Node.js (Sequelize) — Create DB & table (sync)**

**Structure**

node-q8/

└── init.js

cd node-q8

npm init -y

npm i sequelize mysql2

**DB Step**

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

init.js

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

(async()=>{

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

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

await sequelize.sync();

console.log("Tables ready");

})();

**Run**

node init.js

**Expected**

Tables ready

**Q9. React — Hook demo (useState)**

**Structure**

react-q9/

└── src/components/Counter.js

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>);

}

**Run**

* Render <Counter/> in App.js, then npm start.

**Expected**: Counter increments.

**SET 4 (Q10–Q12)**

**Q10. Node.js Express — Generate JWT on login**

**Install & Structure**

node-q10/

└── server.js

cd node-q10

npm init -y

npm i express jsonwebtoken bcryptjs

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";

const user = { email: "test@site.com", pass: bcrypt.hashSync("123456", 8) };

app.post("/login", (req, res) => {

const { email, password } = req.body;

if (email !== user.email || !bcrypt.compareSync(password, user.pass))

return res.status(401).json({ error: "Invalid credentials" });

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

res.json({ token });

});

app.listen(3004, () => console.log("http://localhost:3004"));

**Run**

node server.js

curl -X POST http://localhost:3004/login -H "Content-Type: application/json" \

-d '{"email":"test@site.com","password":"123456"}'

**Expected**

{"token":"<JWT\_TOKEN\_HERE>"}

**Q11. React — Login component**

**Structure**

react-q11/

└── src/components/Login.js

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(JSON.stringify(form)); };

return (

<form onSubmit={submit}>

<input name="email" type="email" placeholder="Email" onChange={change}/>

<input name="password" type="password" placeholder="Password" onChange={change}/>

<button>Login</button>

</form>

);

}

**Run**: Render in App.js, then npm start.

**Q12. JavaScript — Promise & resolve**

**Structure**

js-q12/

└── promise.js

promise.js

function doWork(){

return new Promise(resolve => setTimeout(()=>resolve("OK"), 400));

}

doWork().then(console.log);

**Run**

node promise.js

**Expected**

OK

# SET 5 (Q13–Q15)

## Q13. Node.js (Sequelize) — Search employee by email

**Install**

mkdir node-q13 && cd node-q13

npm init -y

npm i express sequelize mysql2

**📂 Structure**

node-q13/

├── server.js

├── config/db.js

├── models/Employee.js

└── routes/employee.js

**DB Setup**

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;

**Files**

config/db.js

const { Sequelize } = require("sequelize");

module.exports = new Sequelize("company","root","",{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:"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("http://localhost:3005")));

**Run**

node server.js

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

**Expected**

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

## Q14. React — Register component

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(JSON.stringify(form));};

return(

<form onSubmit={submit}>

<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>Register</button>

</form>

);

}

**Expected**: Form alert with entered data.

## Q15. JavaScript — Async/Await

async.js

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

async function run(){

console.log("Start");

await delay(1000);

console.log("End after 1s");

}

run();

**Expected**

Start

End after 1s

# SET 6 (Q16–Q18)

## Q16. Node.js (Sequelize) — Insert employee

**📂 Structure**

node-q16/

├── server.js

├── config/db.js

├── models/Employee.js

└── routes/employee.js

**DB Setup**

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)

);

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("http://localhost:3006")));

**Run**

curl -X POST http://localhost:3006/employees -H "Content-Type: application/json" \

-d '{"name":"Sara","email":"sara@mail.com"}'

**Expected**

{"id":2,"name":"Sara","email":"sara@mail.com"}

## Q17. React — Render & update username (useState + useEffect)

User.js

import {useState,useEffect} from "react";

export default function User(){

const [name,setName]=useState("Guest");

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

return(

<div>

<input value={name} onChange={e=>setName(e.target.value)}/>

<p>Hello {name}</p>

</div>

);

}

**Expected**: Updates text + page title as name changes.

## Q18. React — List of employees

EmployeeList.js

export default function EmployeeList(){

const employees=[{id:1,name:"Ali"},{id:2,name:"Sara"}];

return <ul>{employees.map(e=><li key={e.id}>{e.name}</li>)}</ul>;

}

# SET 7 (Q19–Q21)

## Q19. React Router — Multi-page app

**Install**

npm install react-router-dom

App.js

import {BrowserRouter,Routes,Route,Link} from "react-router-dom";

function Home(){return <h2>Home</h2>}

function About(){return <h2>About</h2>}

function Contact(){return <h2>Contact</h2>}

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>

);

}

## Q20. React — Dropdown with useRef

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>A</li><li>B</li></ul>}

</div>

);

}

## Q21. React Router — Login & Register routes

import {BrowserRouter,Routes,Route,Link} from "react-router-dom";

function Login(){return <form><input placeholder="Email"/><input placeholder="Password"/><button>Login</button></form>}

function Register(){return <form><input placeholder="Name"/><input placeholder="Email"/><input placeholder="Password"/><button>Register</button></form>}

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>

);

}

# SET 8 (Q22–Q24)

## Q22. Node.js (Sequelize) — List all employees

**DB Setup**

USE company;

INSERT INTO employees(name,email) VALUES('Omar','omar@mail.com');

routes/employee.js

const express=require("express");

const Employee=require("../models/Employee");

const router=express.Router();

router.get("/", async (\_req,res)=>{

const rows=await Employee.findAll();

res.json(rows);

});

module.exports=router;

**Run**

node server.js

curl http://localhost:3006/employees

## Q23. React — Update username with useState+useEffect (same as Q17)

(Reuse Q17)

## Q24. React — useState illustration

Counter.js

import {useState} from "react";

export default function Counter(){

const [c,setC]=useState(0);

return <div><p>{c}</p><button onClick={()=>setC(c+1)}>+1</button></div>;

}

✅ That covers **Q13–Q24** (Sets 5–8).

👉 Say **“next”** and I’ll give you **Sets 9–12 (Q25–Q36)**.

# SET 9 (Q25–Q27)

## Q25. Node.js (Sequelize) — Register REST API

**Install & Structure**

node-q25/

├── server.js

├── config/db.js

├── models/User.js

└── routes/user.js

**DB Setup**

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)

);

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 u=await User.create(req.body);

res.json(u);

});

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(3007,()=>console.log("http://localhost:3007")));

**Run**

curl -X POST http://localhost:3007/api/register \

-H "Content-Type: application/json" \

-d '{"name":"Ali","email":"ali@mail.com","password":"123"}'

## Q26. React Router — Login & Register pages

(Reuse Q21 React Router pattern with two routes */login* and */register*.)

## Q27. React — Dropdown with useRef

(Reuse Q20 *Dropdown.js*.)

# SET 10 (Q28–Q30)

## Q28. Node.js (Sequelize) — Login REST API

**Install**

npm i bcryptjs jsonwebtoken

**DB Setup**

USE authdb;

-- ensure a user exists

INSERT INTO users(name,email,password) VALUES('Sara','sara@mail.com','123');

routes/user.js

const express=require("express");

const jwt=require("jsonwebtoken");

const bcrypt=require("bcryptjs");

const User=require("../models/User");

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) return res.status(404).json({error:"No user"});

if(password!==user.password) return res.status(401).json({error:"Bad creds"}); // simple (for exam)

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

res.json({token});

});

module.exports=router;

## Q29. React — Display employee list

EmployeeList.js

export default function EmployeeList(){

const employees=[{id:1,name:"Ali"},{id:2,name:"Sara"}];

return <ul>{employees.map(e=><li key={e.id}>{e.name}</li>)}</ul>;

}

## Q30. React — Update username with hooks

(Reuse Q17 *User.js*)

# SET 11 (Q31–Q33)

## Q31. Node.js (Sequelize) — Select all customers + delete specific

**DB Setup**

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@shop.com"),("Sara","sara@shop.com");

models/Customer.js

const { DataTypes }=require("sequelize");

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

module.exports=sequelize.define("Customer",{

name:DataTypes.STRING,

email:DataTypes.STRING

},{tableName:"customers",timestamps:false});

routes/customer.js

const express=require("express");

const Customer=require("../models/Customer");

const router=express.Router();

router.get("/",async(\_req,res)=>{ res.json(await Customer.findAll()); });

router.delete("/:id",async(req,res)=>{

const count=await Customer.destroy({where:{id:req.params.id}});

res.json({deleted:count});

});

module.exports=router;

## Q32. JavaScript — Async/Await

(Reuse Q15 *async.js*)

## Q33. React — Register component

(Reuse Q14 *Register.js*)

# SET 12 (Q34–Q36)

## Q34. Node.js (Sequelize) — Insert multiple students

**DB Setup**

USE school;

CREATE TABLE IF NOT EXISTS students2(

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(50),

marks INT

);

bulkInsert.js

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

const sequelize = new Sequelize("school","root","",{dialect:"mysql"});

const Student = sequelize.define("Student2",{

name:DataTypes.STRING,

marks:DataTypes.INTEGER

},{tableName:"students2",timestamps:false});

(async()=>{

await sequelize.sync();

await Student.bulkCreate([

{name:"Ali",marks:70},

{name:"Sara",marks:80},

{name:"Omar",marks:65}

]);

console.log("Inserted");

})();

## Q35. JavaScript — Promise & resolve

(Reuse Q12 *promise.js*)

## Q36. React — Login component

(Reuse Q11 *Login.js*)

**SET 17 (Q49–Q51)**

**Q49. HTML + JS — Student Registration form validation**

student.html

<form onsubmit="return validate()">

<input id="fname" placeholder="First Name"/><br>

<input id="lname" placeholder="Last Name"/><br>

<input id="age" type="number" placeholder="Age"/><br>

<button>Submit</button>

</form>

<script>

function validate(){

let fname=document.getElementById("fname").value;

let lname=document.getElementById("lname").value;

let age=parseInt(document.getElementById("age").value);

if(!/^[A-Za-z]+$/.test(fname) || !/^[A-Za-z]+$/.test(lname)){

alert("Names must be alphabets only"); return false;

}

if(age<18 || age>50){ alert("Age must be 18–50"); return false; }

return true;

}

</script>

**Q50. Node.js — Open file and return content, else 404**

*(Reuse Q5 server.js)*

**Q51. JavaScript — Callback demo**

*(Reuse Q6 callback.js)*

**SET 18 (Q52–Q54)**

**Q52. Node.js (Sequelize) — List students with marks > 60**

*(Reuse Q1 server.js with /students route)*

**Q53. Node.js (Sequelize) — Create DB & table**

*(Reuse Q8 init.js)*

**Q54. React — Hook demo**

*(Reuse Q9 Counter.js or Timer.js)*

**SET 19 (Q55–Q57)**

**Q55. Node.js (Sequelize) — Delete student by roll no**

*(Reuse Q4 delete route)*

**Q56. React — Login component**

*(Reuse Q11 Login.js)*

**Q57. JavaScript — Promise & resolve**

*(Reuse Q12 promise.js)*

**SET 20 (Q58–Q60)**

**Q58. Node.js (Sequelize) — Insert teacher data**

*(Reuse Q7 teacher.js + server.js)*

**Q59. React — Register component**

*(Reuse Q14 Register.js)*

**Q60. JavaScript — Async/Await**

*(Reuse Q15 async.js)*