# DEPARTMENT OF INFORMATION TECHNOLOGY

**COURSE CODE: DJS22ITL604 DATE:**

**COURSE NAME: Full Stack Web Development Laboratory**  **CLASS: TYBTech**

**Name: Anish Sharma Div:IT-1 Roll no:I011**

**EXPERIMENT NO. 07**

**CO/LO:** CO1-Develop a full stack web application.

**AIM / OBJECTIVE:** Data Validation and Error Handling Implement input validation in React components and handle basic errors in Express.js.

**THEORY**:

Input validation and error handling are critical aspects of full-stack development. This lab will guide you through creating a full-stack application with React and Express.js, ensuring secure and wellvalidated user input.

**Technologies Used:**

Frontend: React.js, Formik, Yup

Backend: Express.js, Node.js

Other Dependencies: Axios (for API calls), CORS (Cross-Origin Resource Sharing), Helmet (security middleware)

**Step 1: Project Setup**

1. **Create a new project folder** mkdir my-newapp

&& cd my-new-app

1. **Create separate folders for frontend and backend mkdir frontend backend**

**Step 2: Setting Up the Backend (Express.js)**

Navigate to the backend folder and initialize Node.js cd backend npm init -y Install dependencies npm install express cors helmet

# DEPARTMENT OF INFORMATION TECHNOLOGY

dotenv Create necessary folders and files mkdir routes middleware

**COURSE CODE: DJS22ITL604 DATE:**

**COURSE NAME: Full Stack Web Development Laboratory**  **CLASS: TYBTech** cd routes && touch userRoutes.js && cd .. cd middleware && touch errorHandler.js && cd ..

touch server.js

**Backend Code Implementation backend/server.js** const express = require("express"); const cors = require("cors"); const helmet = require("helmet"); const dotenv = require("dotenv"); const userRoutes = require("./routes/userRoutes"); const errorHandler = require("./middleware/errorHandler"); dotenv.config(); const app = express(); app.use(express.json()); app.use(cors()); app.use(helmet()); app.use("/api/users", userRoutes); app.use(errorHandler); const PORT = process.env.PORT || 5000;

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

**backend/routes/userRoutes.js** const express = require("express"); const router = express.Router(); **COURSE**

**NAME: Full Stack Web**

**Development Laboratory**

**CLASS: TYBTech** router.get("/",

**DEPARTMENT OF INFORMATION TECHNOLOGY**

(req, res) => { res.json({ message: "Users route working!" }); **COURSE CODE: DJS22ITL604**

**DATE:**

}); router.post("/register", (req, res, next)

=> { const { name, email } = req.body; if

(!name || !email) { return next(new Error("Name and

Email are required"));

}

res.json({ message: "User registered successfully" });

});

module.exports = router; **backend/middleware/errorHandler.js** module.exports

= (err, req, res, next) => { console.error(err.stack); res.status(500).json({ error: err.message || "Internal Server Error" }); };

**Step 3: Setting Up the Frontend (React + Formik + Yup) Navigate back and create a React app** npx create-reactapp frontend cd frontend **Install dependencies** npm install formik yup axios

**DEPARTMENT OF INFORMATION TECHNOLOGY**

Create necessary folders and files mkdir src/components cd src/components && touch FormComponent.jsx && cd ..

touch src/App.js

**Frontend Code Implementation frontend/src/components/FormComponent.jsx** import

React from "react";

**COURSE CODE: DJS22ITL604 DATE:**

import { Formik, Form, Field, ErrorMessage } from "formik"; import \* as Yup from "yup"; import axios from "axios";

const validationSchema = Yup.object({ name: Yup.string().min(3, "Too Short!").required("Name is required"), email: Yup.string().email("Invalid email").required("Email is required"), });

const FormComponent = () => { const handleSubmit = async (values, { setSubmitting, setErrors }) => { try { const response = await axios.post("http://localhost:5000/api/users/register", values); alert(response.data.message);

} catch (error) { setErrors({ api: error.response?.data?.error

|| "Server Error" });

}

setSubmitting(false);

# DEPARTMENT OF INFORMATION TECHNOLOGY

};

return (

<Formik initialValues={{ name: "", email: "" }} validationSchema={validationSchema} onSubmit={handleSubmit}>

{({ errors, isSubmitting }) => (

<Form>

{errors.api && <div style={{ color: "red" }}>{errors.api}</div>}

<div>

**COURSE CODE: DJS22ITL604 DATE:**

<label>Name:</label>

<Field type="text" name="name" />

<ErrorMessage name="name" component="div" style={{ color: "red" }} />

</div>

<div>

<label>Email:</label>

<Field type="email" name="email" />

<ErrorMessage name="email" component="div" style={{ color: "red" }} />

</div>

<button type="submit" disabled={isSubmitting}>{isSubmitting ? "Submitting..." : "Submit"}</button>

</Form>

)}

</Formik>

# DEPARTMENT OF INFORMATION TECHNOLOGY

);

};

export default FormComponent; **frontend/src/App.js** import React from "react";

import FormComponent from "./components/FormComponent";

function App() { return (

<div>

<h1>User Registration</h1>

**COURSE CODE: DJS22ITL604 DATE:**

<FormComponent />

</div>

); } export default App;

**Step 4: Running the Application** Starting the Backend cd backend node server.js Starting the

Frontend cd frontend npm start

App.js import { BrowserRouter as Router, Routes, Route } from 'reactrouter-dom' import { ToastContainer } from 'react-toastify' import 'reacttoastify/dist/ReactToastify.css' import Header from './components/Header' import PrivateRoute from './components/PrivateRoute' import Home from

# DEPARTMENT OF INFORMATION TECHNOLOGY

'./pages/Home' import Login from './pages/Login' import Register from

'./pages/Register' import NewTicket from './pages/NewTicket' import Tickets from './pages/Tickets'

import Ticket from './pages/Ticket'

// NOTE: Here we have removed the nested routing as the path is the same

function App() { return (

<>

<Router>

<div className='container'>

<Header /> <Routes>

**COURSE CODE: DJS22ITL604 DATE:**

<Route path='/' element={<Home />} />

<Route path='/login' element={<Login />} />

<Route path='/register' element={<Register />} />

<Route path='/new-ticket' element={

<PrivateRoute>

<NewTicket />

</PrivateRoute>

}

/>

<Route path='/tickets' element={

<PrivateRoute>

<Tickets />

</PrivateRoute>

}

/>

# DEPARTMENT OF INFORMATION TECHNOLOGY

<Route path='/ticket/:ticketId'

element={

<PrivateRoute>

<Ticket />

</PrivateRoute>

}

/>

</Routes>

</div>

</Router>

<ToastContainer />

</>

)

}

export default App

**COURSE CODE: DJS22ITL604 DATE:**  server.js const path = require('path') const express = require('express') require('colors') require('dotenv').config() const { errorHandler } = require('./middleware/errorMiddleware') const connectDB = require('./config/db') const PORT = process.env.PORT || 5000

// Connect to database

connectDB()

const app = express()

app.use(express.json()) app.use(express.urlencoded({ extended: false }))

// Routes

# DEPARTMENT OF INFORMATION TECHNOLOGY

app.use('/api/users', require('./routes/userRoutes')) app.use('/api/tickets', require('./routes/ticketRoutes'))

// Serve Frontend if (process.env.NODE\_ENV

=== 'production') {

// Set build folder as static app.use(express.static(path.join(\_\_dirname, '../frontend/build')))

// FIX: below code fixes app crashing on refresh in deployment app.get('\*', (\_, res) => { res.sendFile(path.join(\_\_dirname, '../frontend/build/index.html'))

})

} else { app.get('/', (\_, res) => { res.status(200).json({ message: 'Welcome to the Support Desk API' })

})

}

app.use(errorHandler)

**COURSE CODE: DJS22ITL604 DATE:**

app.listen(PORT, () => console.log(`Server started on port ${PORT}`))

middleware.js const errorHandler = (error, \_, res, next) => {

// FIX: check for bad status codes, if it's a good status code then we want to send

// a bad status code i.e. 2xx should not be sent as error response const statusCode = res.statusCode < 400 ? 500 : res.statusCode

res.status(statusCode) res.json({ message: error.message, stack: process.env.NODE\_ENV === 'production' ? null : error.stack,

})

# DEPARTMENT OF INFORMATION TECHNOLOGY

}

module.exports = { errorHandler } authHandler.js const jwt = require('jsonwebtoken') const

asyncHandler = require('express-async-handler') const

User = require('../models/userModel')

const protect = asyncHandler(async (req, res, next) => { let token if

(

req.headers.authorization && req.headers.authorization.startsWith('Bearer')

) { try {

// Get token from header token = req.headers.authorization.split(' ')[1]

// Verify token const decoded = jwt.verify(token,

process.env.JWT\_SECRET)

// Get user from token req.user = await

User.findById(decoded.id).select('-password')

// NOTE: We need to check if a user was found

# DEPARTMENT OF INFORMATION TECHNOLOGY

**COURSE CODE: DJS22ITL604 DATE:**

// https://www.udemy.com/course/react-front-to-back-

2022/learn/lecture/30591026#questions/1784357

0 if (!req.user) { res.status(401) throw new Error('Not authorized')

}

next()

} catch (error) { console.log(error) res.status(401) throw new Error('Not authorized')

}

}

if (!token) { res.status(401) throw new Error('Not authorized')

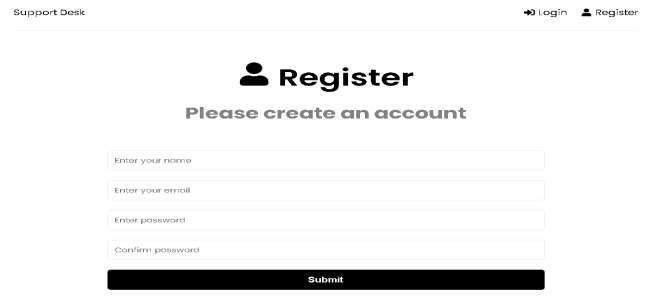
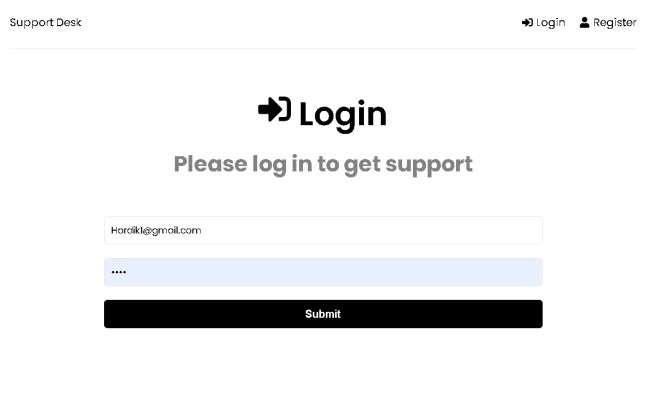
}

})

module.exports = { protect }

# DEPARTMENT OF INFORMATION TECHNOLOGY

**COURSE CODE: DJS22ITL604 DATE:**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJS22ITL604 DATE:**

Conclusion:Middlewares are used and implement error handling.

BOOKS AND WEB RESOURCES:

1. Official React Documentation: https://react.dev
2. Official Formik Documentation: https://formik.org
3. Yup Documentation: https://github.com/jquense/yup
4. Express.js Guide: [https://expressjs.com](https://expressjs.com/)
5. https://www.youtube.com/watch?v=tIdNeoHniEY
6. https://www.youtube.com/watch?v=Gbq66v4QulI