Perfect ✅ thanks for clarifying.
So we will ignore file uploading (keep that React UI as-is but not connect it), and only connect the existing backend CRUD for Reports + ReportType with your React frontend.

Let's do this step by step:

🔹 Step 1: Backend (IntelliJ)

You already have:

Report entity

ReportType entity

ReportsController

ReportService

👉 You only need to enable CORS so your React frontend (running on port 3000) can talk to Spring Boot (port 8080).

Add this CORS config (create a new file):

📂 src/main/java/com/yourapp/config/CorsConfig.java

```
package com.yourapp.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class CorsConfig {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowedOrigins("http://localhost:3000")
                        .allowedMethods("GET", "POST", "PUT", "DELETE");
            }
        };
    }
```

}

✅ Now your frontend can call backend without CORS errors.

🔹 Step 2: Frontend (VS Code)

We'll replace your mock data (initialRules, reportTypes) with real API calls.

1. Install axios (inside your React app folder):

```
npm install axios
```

2. Update ReportsPage.js
At the very top:

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import Card from "../Card/Card";
import RuleModal from "../Modal/RuleModal";
import { FiPlus, FiEdit2, FiTrash2, FiUploadCloud } from "react-icons/fi";
import { useOutletContext } from "react-router-dom";
```

2A. Remove mock data

❌ Delete this line:

```
import { initialRules, reportTypes } from "../data/mockData";
```

2B. Add backend state + fetch

Replace your current state with:

```
const [rules, setRules] = useState([]); // Reports from backend
const [reportTypes, setReportTypes] = useState([]); // ReportTypes from backend
const [activeTab, setActiveTab] = useState("config");
const [currentRule, setCurrentRule] = useState({ id: null, name: "", path: "" });
const [showRuleModal, setShowRuleModal] = useState(false);
const [file, setFile] = useState(null);
const [selectedReportType, setSelectedReportType] = useState("");
const [isUploading, setIsUploading] = useState(false);
```

Then add inside your component:

```
useEffect(() => {
  fetchReports();
  fetchReportTypes();
```

```
}, []);

const fetchReports = async () => {
  try {
    const res = await axios.get("http://localhost:8080/api/reports");
    setRules(res.data);
  } catch (error) {
    console.error("Error fetching reports", error);
  }
};

const fetchReportTypes = async () => {
  try {
    const res = await axios.get("http://localhost:8080/api/report-types");
    setReportTypes(res.data);
    if (res.data.length > 0) {
      setSelectedReportType(res.data[0].name);
    }
  } catch (error) {
    console.error("Error fetching report types", error);
  }
};
```

2C. Update CRUD actions

Save report (add/update):

```
const handleSaveRule = async () => {
  try {
    if (currentRule.id) {
      // Update
      await axios.put(`http://localhost:8080/api/reports/${currentRule.id}`, currentRule);
      setNotification({ type: "success", message: "Report updated successfully!" });
    } else {
      // Add
      await axios.post("http://localhost:8080/api/reports", currentRule);
      setNotification({ type: "success", message: "Report added successfully!" });
    }
    fetchReports(); // refresh table
    setShowRuleModal(false);
  } catch (error) {
    setNotification({ type: "error", message: "Error saving report!" });
  }
};
```

Delete report:

```
const handleDeleteRule = async (ruleId) => {
```

```
  if (window.confirm("Are you sure you want to delete this report?")) {
    try {
      await axios.delete(`http://localhost:8080/api/reports/${ruleId}`);
      setNotification({ type: "success", message: "Report deleted successfully!" });
      fetchReports();
    } catch (error) {
      setNotification({ type: "error", message: "Error deleting report!" });
    }
  }
};
```

Add new report:

```
const handleAddRule = () => {
  setCurrentRule({ id: null, name: "", path: "" });
  setShowRuleModal(true);
};
```

🔹 Step 3: Verify

1. Start backend (mvn spring-boot:run or run from IntelliJ).

Test in Postman:

GET http://localhost:8080/api/reports → should return reports.

GET http://localhost:8080/api/report-types → should return report types.

2. Start frontend (npm start in VS Code).

Go to Reports Page.

The table should now load real DB data instead of mock data.

Add / Edit / Delete should work → confirm in pgAdmin.

✅ Now your frontend and backend are connected using existing DB logic.
❌ No file upload implemented (but upload button will just show notification for now).

Do you want me to also show how to modify RuleModal.js so it sends name + path + selected reportTypeId correctly to backend?