Team Directory Application

Sabre Software Intern - Adobe ColdFusion Technical Project

A full-stack application that demonstrates integration between a ColdFusion REST API backend and a modern React frontend.

👨‍💻 Author

Junaid
Application for: Software Intern - Adobe ColdFusion at MED49 Solutions Inc.

📋 Project Overview

This application retrieves employee data from a SQLite database via a RESTful ColdFusion API and displays it in an interactive React interface with real-time search functionality.

Features

- ✅ RESTful ColdFusion API with JSON responses
- ✅ React frontend using modern Hooks (useState, useEffect)
- ✅ SQLite database with sample employee data
- ✅ Real-time search/filter functionality
- ✅ CORS handling for cross-origin requests
- ✅ Security best practices with `<cfqueryparam>`
- ✅ Responsive, modern UI design
- ✅ Error handling and loading states

🗂️ Project Structure

```
SabreFinal/
├── src/
│   ├── Application.cfc        # ColdFusion application configuration
│   ├── Employee.cfc           # REST API component
│   └── fix.cfm                # Database test file
├── team-directory-react/
│   ├── src/
```

```
│   │   ├── App.jsx          # Main React component
│   │   ├── App.css          # Component styles
│   │   ├── main.jsx         # React entry point
│   │   └── index.css        # Base styles
│   ├── index.html           # HTML template
│   ├── package.json         # Dependencies
│   └── vite.config.js       # Vite configuration
├── team_directory.db        # SQLite database
├── team_directory.sql       # Database setup script
├── server.json              # CommandBox server config
└── README.md                # This file
```

🚀 Setup Instructions

Prerequisites

- CommandBox CLI (for ColdFusion/Lucee server)
- Node.js (v16 or higher) and npm
- SQLite3 (optional, for manual database inspection)

Part 1: Database Setup

1. **Create the SQLite database:**

```bash
cd C:\Users\JUNAID\OneDrive\Desktop
sqlite3 team_directory.db < team_directory.sql
```

Or manually:

```bash
sqlite3 team_directory.db
```

Then paste the contents of `team_directory.sql` and press Enter.

2. **Verify database creation:**

```bash
```

```
sqlite3 team_directory.db
SELECT * FROM Employees;
.exit
```

Part 2: ColdFusion Backend Setup

1. **Navigate to your project directory:**

```
bash

cd C:\Users\JUNAID\OneDrive\Desktop\SabreFinal
```

2. **Start the CommandBox server:**

```
bash

box start
```

The server will start on `http://127.0.0.1:8080`

3. **Test the database connection:**
   - Open: `http://127.0.0.1:8080/src/fix.cfm`
   - You should see a list of employees
4. **Test the REST API:**
   - Open: `http://127.0.0.1:8080/rest/employee/list`
   - You should see JSON response with employee data
5. **Test search functionality:**
   - Try: `http://127.0.0.1:8080/rest/employee/list?search=john`

Part 3: React Frontend Setup

1. **Navigate to the React app directory:**

```
bash

cd team-directory-react
```

2. **Install dependencies:**

```bash
npm install
```

3. **Start the development server:**

```bash
npm run dev
```

The React app will open at `http://localhost:3000`

4. **Build for production (optional):**

```bash
npm run build
```

🧪 Testing the Application

Manual Testing Steps:

1. **Backend Testing:**
   - Open `http://127.0.0.1:8080/rest/employee/list` in your browser
   - Verify JSON response contains all 7 employees
   - Test search: `http://127.0.0.1:8080/rest/employee/list?search=developer`
2. **Frontend Testing:**
   - Open `http://localhost:3000` in your browser
   - Verify all employees are displayed in cards
   - Test search bar with various terms:
     - Try names: "John", "Sarah"
     - Try roles: "Developer", "Manager"
     - Try departments: "Engineering"
   - Test "Clear" button functionality
   - Verify responsive design on mobile view
3. **CORS Testing:**

- Open browser DevTools (F12)
- Check Network tab for API calls
- Verify no CORS errors in Console

🔒 Security Features

1. **SQL Injection Prevention:**
   - All database queries use `<cfqueryparam>` for parameterized queries
   - Input sanitization with `lcase()` and `trim()`
2. **CORS Configuration:**
   - Proper CORS headers set in REST API
   - Handles OPTIONS preflight requests
3. **Error Handling:**
   - Try-catch blocks in React for API failures
   - Graceful error messages for users
   - No sensitive information exposed in errors

🛠️ Technologies Used

Backend:

- **ColdFusion/Lucee 5.4.8.2** - Server-side scripting
- **SQLite** - Lightweight database
- **REST API** - JSON communication

Frontend:

- **React 18.2** - UI framework
- **Vite 4.3** - Build tool and dev server
- **Modern JavaScript (ES6+)** - Async/await, fetch API
- **CSS3** - Styling with Flexbox and Grid

Tools:

- **CommandBox** - CFML server and package manager
- **npm** - Package manager for JavaScript

📊 Database Schema

```sql
Employees Table:
- ID (INTEGER, PRIMARY KEY, AUTOINCREMENT)
- FirstName (VARCHAR(50), NOT NULL)
- LastName (VARCHAR(50), NOT NULL)
- Role (VARCHAR(100), NOT NULL)
- Email (VARCHAR(100))
- Department (VARCHAR(50))
- HireDate (DATE)
```

Sample Data: 7 employees across various departments (Engineering, Product, Design, QA, Infrastructure, Business)

🔧 Configuration Notes

server.json

- **Port:** 8080
- **Engine:** Lucee 5
- **Datasource:** SQLite at `C:/Users/JUNAID/OneDrive/Desktop/team_directory.db`

Vite Configuration

- **Dev Server Port:** 3000
- **Auto-open browser:** Enabled

🎯 Project Requirements Checklist

- [x] SQLite database with Employees table
- [x] At least 5 employee records (provided 7)
- [x] ColdFusion REST API endpoint
- [x] JSON response format
- [x] CORS headers properly configured
- [x] React app using Vite
- [x] useState Hook for state management
- [x] useEffect Hook for API calls
- [x] Clean, responsive UI with card layout
- [x] **BONUS:** Search/filter functionality implemented
- [x] Security: cfqueryparam usage
- [x] Code organization and readability
- [x] Complete README with setup instructions
- [x] SQL script for database recreation

🐛 Troubleshooting

Issue: "datasource [teamDSN] doesn't exist"

Solution: Ensure `server.json` is in the project root and contains correct database path. Restart CommandBox server.

Issue: CORS errors in browser console

Solution: Verify CORS headers in `Employee.cfc`. Ensure REST API is accessible at `http://127.0.0.1:8080/rest/employee/list`

Issue: React app shows "Error Loading Data"

Solution:

1. Confirm ColdFusion server is running on port 8080
2. Test API endpoint directly in browser
3. Check browser console for specific errors

Issue: Search not working

Solution: Ensure data is loaded successfully. Check that column names match (FIRSTNAME, LASTNAME,

etc. in uppercase)

📝 Code Quality Highlights

1. **Modular Structure:** Separate concerns between API, UI, and data
2. **Reusable Components:** Single-responsibility principle
3. **Error Boundaries:** Graceful degradation on failures
4. **Responsive Design:** Mobile-first CSS approach
5. **Accessibility:** Semantic HTML and proper labels
6. **Performance:** Efficient filtering with React hooks
7. **Maintainability:** Clear variable names and comments

📹 Video Walkthrough

A 2-minute video demonstration is included showing:

1. Project structure overview
2. Database and API setup
3. React app functionality
4. Search feature demonstration
5. Code organization highlights

📧 Contact

Junaid
Location: Rawalpindi, Punjab, Pakistan
Project: Sabre Software Intern - Adobe ColdFusion Application

---

🙏 Acknowledgments

Thank you to MED49 Solutions Inc. and the Sabre team for the opportunity to work on this technical assessment. This project demonstrates my ability to bridge modern frontend technologies with ColdFusion backend systems, following industry best practices for security, code organization, and user experience.

Project Completion Time: 8-10 hours
Submission Date: January 2026