Employee Registration Form Documentation

Overview

The **Employee Registration Form** is a React-based application using Material UI and Firebase Firestore for managing employee data. This document provides an overview of the implementation, including the UI components, Firebase integration, and form handling.

Features

- User-friendly registration form for employees
- Data validation and structured input fields
- Firestore database integration for storing employee details
- Success message display after form submission

Technologies Used

• React: Frontend framework

• Material UI: UI components

• Firebase Firestore: Database for storing employee records

• **Date Fns**: Date management library

Installation & Setup

Clone the repository: git clone https://github.com/your-repo.git

- 1. cd your-repo
- 2. Install dependencies: npm install
- 3. Configure Firebase:
 - Create a Firebase project
 - Set up Firestore database
 - Initialize Firebase in the project (firebase.js)
 - Enter your Firebase credentials in firebase.js

Firebase Configuration (firebase.js)

Create a firebase. js file and enter your Firebase credentials:

import { initializeApp } from 'firebase/app';

```
import { getFirestore } from 'firebase/firestore';

const firebaseConfig = {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_AUTH_DOMAIN",
    projectId: "YOUR_PROJECT_ID",
    storageBucket: "YOUR_STORAGE_BUCKET",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
    appId: "YOUR_APP_ID"
};

const app = initializeApp(firebaseConfig);
const db = getFirestore(app);

export { db };
```

Code Explanation

Component Imports

The form utilizes Material UI components (TextField, Button, Grid, etc.) and Firebase Firestore methods.

```
import React, { useState } from 'react';
import { Container, Box, Typography, TextField, MenuItem, Button, Grid, Alert } from
'@mui/material';
import { DatePicker, LocalizationProvider } from '@mui/x-date-pickers';
import { AdapterDateFns } from '@mui/x-date-pickers/AdapterDateFns';
import { db, collection, addDoc } from './firebase';
```

State Management

The useState hook manages form input fields and handles changes dynamically.

```
const [formData, setFormData] = useState({
  firstName: ",
  lastName: ",
  email: ",
  phone: ",
  department: ",
  position: ",
  hireDate: null,
});
```

Handling Input Changes

Updates form values in response to user input.

```
const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};
```

Handling Date Selection

Uses DatePicker for selecting the hire date.

```
const handleDateChange = (date) => {
  setFormData({ ...formData, hireDate: date });
};
```

Form Submission

Upon submission, the form data is sent to Firestore.

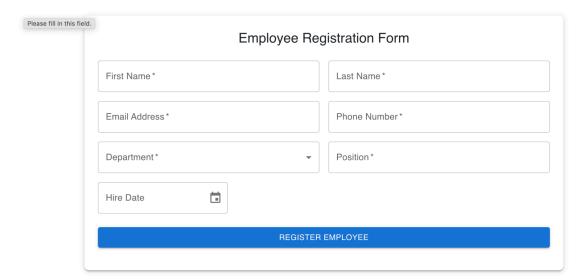
```
const handleSubmit = async (e) => {
    e.preventDefault();
    setSubmitted(true);
    try {
        await addDoc(collection(db, "employees"), formData);
        setFormData({ firstName: ", lastName: ", email: ", phone: ", department: ", position: ",
        hireDate: null });
    } catch (error) {
        console.error("Error adding employee: ", error);
    }
};
```

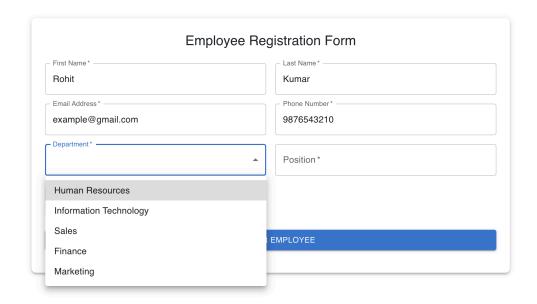
UI Layout

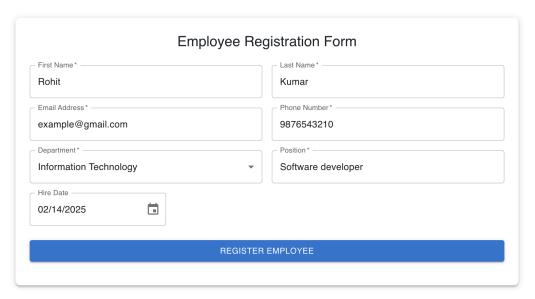
The form consists of input fields structured in a grid layout.

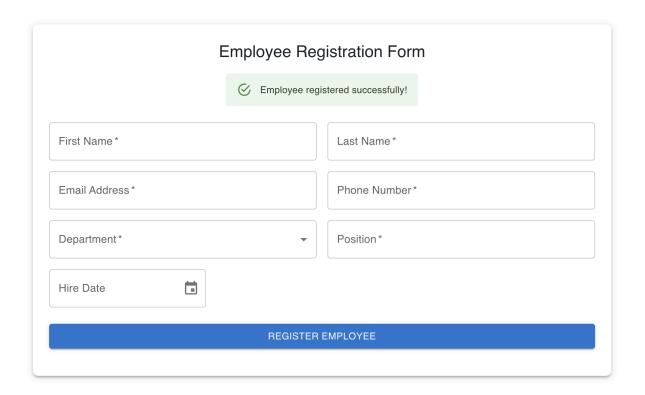
- Name, Email, Phone Number
- Department Selection (MenuItem dropdown)
- Position Input
- Hire Date Picker
- Submit Button

Screenshots









Conclusion

This Employee Registration Form offers a simple and effective way to collect and store employee details in Firestore. The use of Material UI ensures a clean and responsive design, while Firebase provides a scalable backend solution.

For further enhancements, consider adding:

- Form validation for email and phone fields
- Improved error handling for Firestore operations
- Integration with authentication for secure access