Employee Management System

Employee Management project built with MERN and Redux

Database Design

Use **MongoDB** for data store and management.

Onlu one model is needed in the Employee Manegement system: **Employee**.

A possible Employee Schema may look like this:

JavaScript

```
const HeroSchema = new Schema({
  name: {
    type: String,
    required: true
  },
  title: {
    type: String,
    required: true
  },
  sex: {
    type: Number,
    required: true
  },
  startDate: {
    type: Date,
    required: true
  },
  officePhone: {
    type: String,
    required: true
  },
  cellPhone: {
    type: String,
    required: true
  },
  email:  {
    type: String,
    required: true
  },
  manager: {
    type: String,
    required: true
  },
  dirReports: {
    type: Object,
    required: true
  }
  numberOfDirReport: {
    type: Number,
    required: true
  }
});
```

Database Explanation:

- name: employee's name, string
- title: employee's title, string
- sex: employee's gender, only three selections available
- startDate: employee's start working date at company, Date
- officePhone: employee's office phone, string that only in the right firmat
- cellPhone: employee's cell phone, string that only in the right firmat
- email: employee's email, must be in the right email format
- manager: employee's manager, a string which is the maneger's _id generated by MongoDB
- dirReports: employee's direct reports, an array which are all the reports' _id generated by MongoDB
- numberOfDirReport: the number of employee's direct employee

REST API

Use express router to handle data request, create and update.
The employee's router's url is '/api/employee'.
It can be used like this:

```
const employee = require('./routes/api/employee');
app.use('/api/employee', employee);
```

In the employee router, it has 4 functions: GET, PUT, POST, DELETE.

- GET:
  - ' / ': get all the employees, retuan as an Array
  - ' /:id ': get certain employee, retuan as Employee
- PUT: ' / ': add new employee to the database
- POST: ' /:id ': update existed employee to the database, when update is called and executed successfully, the database should update all other employee's relation with the deleted employee
- DELETE: ' /:id ': delete existed employee to teh database, when delete is called and executed successfully, the database should update all other employee's relation with the deleted employee

UI Design and React Component

Use **Material UI** to do the UI design.
For the React components, Reacr Router will be used to navigate through pages.

- ' / ': home page which will show all the employees, by clicking the employee can go to employee detail

page
    - ' /:employeeId ': employee detail page which will show the employee detail
        - ' /:employeeId/manager': detail info of employee manager, not available if no manager
        - ' /:employeeId/reports': all employee's direct reports

For the components hierarchy, it may look something like this:

- Home Page
    - Infinite scroll
    - Header
    - Search Bar
    - Add button
    - List of employees (--> Employee Detail)

In home page, we can navigate to employee detail page by clicking the employee list.

- Employee Detail
    - Back button
    - Basic info
    - Manager info (--> Manager(Employee) Detail)
    - Direct reports info (--> Direct Repoorts Detail)
    - Other info

In Employee page we can go to Manager detail page and Direct Reports page.

- Direct Reports
    - Back button
    - List of direct reports info (-->Employee Detail)