Assignment: Xây dựng Ứng dụng Quản lý Nhân sự (HRM) Hoàn chỉnh bằng JavaScript Thuần  
Mô tả Tổng quát  
Trong assignment này, bạn sẽ xây dựng một ứng dụng quản lý nhân sự (Human Resource Management - HRM) hoàn chỉnh bằng JavaScript thuần (vanilla JavaScript), không sử dụng bất kỳ framework nào như React, Vue, Angular hoặc bất kỳ thư viện bên thứ ba nào. Ứng dụng phải chạy trên trình duyệt web và sử dụng localStorage để lưu trữ dữ liệu bền vững.  
Mục tiêu là thực hành các tính năng JavaScript nâng cao như:  
  
Cú pháp ES6+ (arrow functions, template literals, destructuring, spread/rest operators, v.v.).  
Class và module (sử dụng import/export để tổ chức mã nguồn thành các file riêng biệt).  
Async/await cho các tác vụ bất đồng bộ (ví dụ: giả lập delay khi lưu dữ liệu vào localStorage).  
Closure và higher-order functions để xử lý logic phức tạp.  
Thao tác DOM để xây dựng và cập nhật giao diện động.  
  
Ứng dụng phải bao gồm ít nhất 12 module riêng biệt, mỗi module xử lý một khía cạnh cụ thể của quản lý nhân sự. Bạn có thể thêm module bổ sung nếu cần để đảm bảo tính thống nhất và hoạt động mượt mà. Tổ chức mã nguồn thành các tệp .js riêng biệt cho từng module, và sử dụng một tệp app.js chính để import và tích hợp chúng.  
Về giao diện:  
  
Sử dụng HTML và CSS cơ bản để tạo cấu trúc và phong cách (giữ ở mức tối thiểu, tập trung vào logic JS).  
Tạo một dashboard chính với menu điều hướng (ví dụ: sidebar hoặc navbar) để chuyển đổi giữa các module.  
Sử dụng event listeners cho các tương tác người dùng (click, submit, input, v.v.).  
Đảm bảo ứng dụng responsive (sử dụng media queries cơ bản trong CSS) và xử lý lỗi mượt mà (kiểm tra dữ liệu đầu vào, hiển thị cảnh báo bằng alert hoặc modal, tránh crash ứng dụng).  
  
Dữ liệu sẽ được lưu trữ trong localStorage dưới dạng JSON (ví dụ: mảng đối tượng cho nhân viên, phòng ban, v.v.). Ứng dụng phải tự kiểm tra và khởi tạo dữ liệu mặc định nếu chưa tồn tại.  
Yêu cầu Kỹ thuật  
  
Ngôn ngữ và Công cụ: Chỉ sử dụng JavaScript thuần, HTML5, và CSS3. Không dùng thư viện bên ngoài.  
Cấu trúc Mã nguồn:  
  
index.html: File chính chứa cấu trúc HTML cơ bản (dashboard, menu), và các thẻ <script type="module"> để import app.js.  
style.css: CSS cơ bản cho giao diện (layout, form, table, button, v.v.). Liên kết qua <link> trong index.html.  
Các tệp .js riêng biệt cho từng module (ví dụ: authModule.js, employeeDbModule.js, v.v.).  
app.js: File chính để import tất cả module, khởi tạo ứng dụng, xử lý routing (chuyển module dựa trên menu), và gắn event listeners.  
  
  
Thực tiễn Tốt:  
  
Tuân thủ separation of concerns (tách biệt logic dữ liệu, giao diện, và xử lý sự kiện).  
Sử dụng comment ngắn gọn để giải thích mã nguồn.  
Mã phải modular, tái sử dụng (ví dụ: hàm chung cho validate input).  
Tự kiểm tra tính năng: Chạy ứng dụng trên trình duyệt, kiểm tra các trường hợp edge (dữ liệu rỗng, lỗi input, v.v.).  
  
  
Chạy Ứng dụng: Mở index.html trên trình duyệt (cần server local nếu dùng module, ví dụ: Live Server trên VS Code).  
  
Các Module Bắt buộc  
Dưới đây là danh sách 12 module bắt buộc. Mỗi module phải được triển khai trong một file .js riêng, export các hàm/class cần thiết, và import vào app.js. Tôi sẽ mô tả chi tiết chức năng của từng module để bạn dễ hiểu và triển khai. Tập trung vào logic JS; giao diện chỉ cần form/table đơn giản.  
  
Module Xác thực (AuthModule):  
  
Xử lý đăng nhập và đăng ký người dùng (admin hoặc HR manager).  
Lưu thông tin người dùng (username, password hashed đơn giản bằng closure) vào localStorage.  
Quản lý phiên làm việc: Kiểm tra token/session khi load app, logout để xóa session.  
Nếu chưa đăng nhập, hiển thị form login; sau đăng nhập, chuyển đến dashboard.  
Sử dụng async/await để giả lập delay kiểm tra credentials.  
  
  
Module Cơ sở dữ liệu Nhân viên (EmployeeDbModule):  
  
Quản lý lưu trữ dữ liệu nhân viên dưới dạng mảng đối tượng trong localStorage (mỗi nhân viên có thuộc tính: id, name, departmentId, positionId, salary, hireDate, v.v.).  
Các hàm: getAllEmployees(), getEmployeeById(id), saveEmployees(employees) – sử dụng JSON.parse/stringify.  
Khởi tạo dữ liệu mặc định nếu localStorage rỗng (ví dụ: 5 nhân viên mẫu).  
Sử dụng higher-order functions để filter/sort dữ liệu.  
  
  
Module Thêm Nhân viên (AddEmployeeModule):  
  
Tạo form DOM động (input cho name, id tự generate, department select, position select, salary, v.v.).  
Logic submit: Validate input (không rỗng, salary > 0), thêm vào db qua EmployeeDbModule.  
Hiển thị thông báo thành công/thất bại.  
Sử dụng event listeners cho form submit và input change.  
  
  
Module Sửa Nhân viên (EditEmployeeModule):  
  
Tạo form tìm kiếm (input id hoặc name) để load thông tin nhân viên hiện có.  
Hiển thị form edit với dữ liệu preload, cập nhật thông tin (gọi update từ EmployeeDbModule).  
Xử lý xác nhận trước khi save, validate input tương tự add.  
Sử dụng closure để lưu trạng thái edit.  
  
  
Module Xóa Nhân viên (DeleteEmployeeModule):  
  
Tìm kiếm nhân viên (tương tự edit), hiển thị confirm dialog (sử dụng window.confirm).  
Xóa khỏi db nếu xác nhận, cập nhật localStorage.  
Cập nhật dashboard hoặc list nếu cần (gọi refresh từ app.js).  
  
  
Module Tìm kiếm Nhân viên (SearchEmployeeModule):  
  
Tạo form tìm kiếm nâng cao: input cho name (regex), department, salary range (min-max).  
Logic: Sử dụng filter() với higher-order functions và RegExp để lọc dữ liệu từ db.  
Hiển thị kết quả dưới dạng table DOM, với sorting (ví dụ: theo salary).  
  
  
Module Quản lý Phòng ban (DepartmentModule):  
  
Quản lý danh sách phòng ban (mảng đối tượng: id, name, managerId).  
Các hàm: addDepartment(name), editDepartment(id, newName), deleteDepartment(id), getAllDepartments().  
Lưu vào localStorage riêng, liên kết với nhân viên qua departmentId.  
Hiển thị list table với button add/edit/delete.  
  
  
Module Quản lý Vị trí (PositionModule):  
  
Quản lý vị trí công việc (mảng: id, title, description, salaryBase).  
Các hàm: addPosition(title, desc), editPosition(id, updates), deletePosition(id), getAllPositions().  
Gán vị trí cho nhân viên khi add/edit (select dropdown từ list positions).  
Lưu vào localStorage, sử dụng async/await cho save.  
  
  
Module Quản lý Lương (SalaryModule):  
  
Tính toán lương: Cập nhật salary, thêm bonus/deduction (thuộc tính trong nhân viên).  
Tạo bảng lương: Generate table cho all employees với netSalary = salary + bonus - deduction.  
Các hàm: calculateNetSalary(employee), generatePayrollReport() – return array cho display.  
Sử dụng map/reduce cho tính toán.  
  
  
Module Theo dõi Chấm công (AttendanceModule):  
  
Ghi chấm công hàng ngày: Mảng attendance logs (date, employeeId, checkIn, checkOut).  
Các hàm: checkIn(employeeId), checkOut(employeeId), getAttendanceReport(employeeId, fromDate, toDate).  
Tính tổng giờ làm (sử dụng Date objects), lưu vào localStorage.  
Hiển thị calendar hoặc table cho report.  
  
  
Module Quản lý Nghỉ phép (LeaveModule):  
  
Xử lý yêu cầu nghỉ phép: Mảng leaves (employeeId, startDate, endDate, type: annual/sick, status: pending/approved).  
Các hàm: requestLeave(employeeId, dates, type), approveLeave(leaveId), getLeaveBalance(employeeId) – ví dụ: 20 ngày annual default.  
Theo dõi và cập nhật balance khi approve.  
Hiển thị list requests với button approve/reject.  
  
  
Module Đánh giá Hiệu suất (PerformanceModule):  
  
Thêm đánh giá: Mảng reviews (employeeId, date, rating: 1-5, feedback).  
Các hàm: addReview(employeeId, rating, feedback), getAverageRating(employeeId) – sử dụng reduce để tính trung bình.  
Hiển thị report: Table với average rating và list reviews.  
Sử dụng sort để hiển thị top performers.

Căn cứ vào yêu cầu project đang làm các bạn cần phải tự làm các phần sau  
1. (Xác thực)

* : Triển khai hàm hash password đơn giản sử dụng closure (ví dụ: closure để lưu salt và hash bằng cách kết hợp string đơn giản như password + salt và chuyển thành base64). Validation: Password phải có ít nhất 6 ký tự, không chứa khoảng trắng.
* : Tạo form đăng ký (register) với inputs cho username và password. Validation: Username không rỗng, không trùng với user hiện có (kiểm tra localStorage); password confirm phải khớp.
* : Tạo form đăng nhập (login) với inputs username/password. Sử dụng async/await để giả lập delay 1-2 giây kiểm tra credentials (so sánh hashed password). Validation: Thông báo lỗi nếu username/password sai.
* : Quản lý session: Lưu token (ví dụ: JSON object với username và expiry time) vào localStorage sau login thành công. Kiểm tra session khi load app (nếu hết hạn hoặc không tồn tại, hiển thị form login).
* : Hàm logout: Xóa session từ localStorage và redirect về login. Xử lý lỗi: Nếu không có session, không cho phép truy cập dashboard.
* : Tích hợp với app.js: Khi load, kiểm tra auth trước khi hiển thị các module khác. Sử dụng event listeners cho form submit.

2. (Cơ sở dữ liệu Nhân viên)

* : Định nghĩa cấu trúc object nhân viên: { id: string (UUID hoặc auto-increment), name: string, departmentId: string, positionId: string, salary: number, hireDate: string (ISO format), bonus: number (default 0), deduction: number (default 0) }. Lưu dưới dạng mảng JSON trong localStorage key 'employees'.
* : Triển khai hàm getAllEmployees(): JSON.parse(localStorage) hoặc trả mảng rỗng nếu lỗi. Khởi tạo 5 nhân viên mẫu nếu localStorage rỗng (ví dụ: name: 'John Doe', departmentId: '1', etc.).
* : Hàm getEmployeeById(id): Sử dụng find() HOF để tìm theo id. Validation: Throw error nếu id không tồn tại.
* : Hàm saveEmployees(employees): JSON.stringify và lưu vào localStorage. Sử dụng try-catch để xử lý lỗi parse/stringify.
* : Triển khai HOF cho filter/sort: Ví dụ, hàm filterEmployees(predicate) sử dụng array.filter(predicate); sortEmployees(comparator) sử dụng array.sort(comparator). Validation: Đảm bảo predicate/comparator là function.
* : Tích hợp với các module khác: Export các hàm để Add/Edit/Delete sử dụng.

3. (Thêm Nhân viên)

* : Tạo form DOM động: Sử dụng document.createElement để thêm inputs (name, salary, hireDate), selects cho department (load từ DepartmentModule.getAllDepartments()) và position (từ PositionModule.getAllPositions()).
* : Event listeners: Submit form để thêm employee. Change input để validate real-time (ví dụ: salary input onChange kiểm tra number > 0).
* : Validation: Name không rỗng (trim()), salary > 0 và là number, hireDate hợp lệ (sử dụng Date.parse), departmentId/positionId phải tồn tại (kiểm tra qua getAll). Hiển thị thông báo lỗi dưới mỗi input.
* : Logic submit: Generate id tự động (ví dụ: Date.now() + random), thêm vào db qua EmployeeDbModule.saveEmployees. Hiển thị alert thành công/thất bại.
* : Xử lý lỗi: Nếu save thất bại (localStorage full), retry hoặc thông báo.
* : Tích hợp: Sau add thành công, refresh dashboard (gọi hàm refresh từ app.js).

4. (Sửa Nhân viên)

* : Tạo form tìm kiếm: Input cho id hoặc name. Sử dụng event listener onSubmit để load employee qua EmployeeDbModule.getEmployeeById hoặc filter.
* : Hiển thị form edit: Preload dữ liệu vào inputs/selects tương tự AddModule. Sử dụng closure để lưu trạng thái employee đang edit (ví dụ: let currentEmployee = () => { return employee; }).
* : Validation: Tương tự AddModule, thêm kiểm tra id không thay đổi. Xác nhận trước save bằng window.confirm("Bạn chắc chắn muốn cập nhật?").
* : Logic update: Cập nhật object và save qua EmployeeDbModule. Hiển thị thông báo thành công.
* : Xử lý lỗi: Nếu employee không tồn tại, hiển thị lỗi; sử dụng try-catch cho update.
* : Tích hợp: Refresh UI sau edit.

5. (Xóa Nhân viên)

* : Tạo form tìm kiếm tương tự EditModule để load employee info.
* : Hiển thị confirm: Sử dụng window.confirm("Xác nhận xóa nhân viên này?") sau khi tìm thấy.
* : Logic delete: Nếu confirm, filter mảng để loại bỏ employee và save qua EmployeeDbModule.
* : Validation: Kiểm tra employee tồn tại trước delete; nếu liên kết với department/manager, cảnh báo nếu xóa ảnh hưởng (ví dụ: nếu là manager).
* : Xử lý lỗi: Thông báo nếu delete thất bại.
* : Tích hợp: Gọi refresh dashboard/list sau delete.

6. (Tìm kiếm Nhân viên)

* : Tạo form tìm kiếm: Inputs cho name (regex), select department, inputs min-max salary.
* : Logic filter: Sử dụng HOF filter với RegExp cho name (new RegExp(input, 'i')), departmentId === select, salary >= min && <= max.
* : Hiển thị kết quả: Tạo table DOM động với columns (id, name, etc.), sử dụng sort HOF cho salary (ascending/descending button).
* : Validation: Min <= max salary, regex hợp lệ (try-catch RegExp).
* : Xử lý lỗi: Nếu không kết quả, hiển thị "No results".
* : Tích hợp: Load data từ EmployeeDbModule.getAllEmployees().

7. (Quản lý Phòng ban)

* : Định nghĩa object: { id: string, name: string, managerId: string (link employee) }. Lưu mảng ở localStorage key 'departments'.
* : Hàm addDepartment(name): Generate id, push và save. Validation: Name không rỗng, không trùng.
* : Hàm editDepartment(id, newName): Find và update. Validation: Id tồn tại, newName không rỗng.
* : Hàm deleteDepartment(id): Filter loại bỏ, nhưng kiểm tra nếu có employee link (filter employees by departmentId), nếu có thì cảnh báo hoặc cấm xóa.
* : Hàm getAllDepartments(): Parse từ localStorage.
* : Hiển thị table với buttons add/edit/delete, event listeners cho buttons.
* : Tích hợp: Cập nhật selects ở Add/EditEmployee khi departments thay đổi.

8. (Quản lý Vị trí)

* : Object: { id: string, title: string, description: string, salaryBase: number }. Lưu ở 'positions'.
* : Hàm addPosition(title, desc): Async/await cho save (await new Promise(resolve => setTimeout(resolve, 500)) để giả lập delay).
* : Hàm editPosition(id, updates): Update object. Validation: Title không rỗng, salaryBase > 0.
* : Hàm deletePosition(id): Tương tự deleteDepartment, kiểm tra link với employees.
* : Hàm getAllPositions(): Parse.
* : Tích hợp: Load cho selects ở Add/EditEmployee.

9. (Quản lý Lương)

* : Hàm calculateNetSalary(employee): Return salary + bonus - deduction. Validation: Bonus/deduction >= 0.
* : Hàm generatePayrollReport(): Get all employees, map để tính netSalary, reduce để tổng lương (total payroll).
* : Hiển thị table: Columns employee name, salary, bonus, deduction, netSalary.
* : Validation: Khi update bonus/deduction ở EditEmployee, recalculate.
* : Tích hợp: Gọi từ dashboard để hiển thị report.

10. (Theo dõi Chấm công)

* : Object log: { date: string, employeeId: string, checkIn: timestamp, checkOut: timestamp }. Lưu mảng 'attendance'.
* : Hàm checkIn(employeeId): Add log với checkIn = Date.now(). Validation: Employee tồn tại, không checkIn 2 lần cùng ngày.
* : Hàm checkOut(employeeId): Update log ngày hôm nay với checkOut. Tính hours = (checkOut - checkIn)/3600000.
* : Hàm getAttendanceReport(id, from, to): Filter logs, reduce tính tổng hours.
* : Hiển thị table hoặc calendar (sử dụng Date objects).
* : Validation: Dates hợp lệ, from <= to.

11. (Quản lý Nghỉ phép)

* : Object leave: { id: string, employeeId: string, startDate: string, endDate: string, type: 'annual'|'sick', status: 'pending'|'approved' }. Lưu 'leaves'.
* : Hàm requestLeave(...): Add leave. Validation: Dates hợp lệ, start <= end, không overlap với leave hiện có.
* : Hàm approveLeave(leaveId): Update status, cập nhật balance (default 20 annual, trừ số ngày).
* : Hàm getLeaveBalance(id): Reduce leaves approved để tính remaining.
* : Hiển thị list với buttons approve/reject.
* : Tích hợp: Link với employee (thêm leaveBalance vào employee object?).

12. (Đánh giá Hiệu suất)

* : Object review: { employeeId: string, date: string, rating: 1-5, feedback: string }. Lưu 'reviews'.
* : Hàm addReview(...): Add. Validation: Rating 1-5, feedback không rỗng.
* : Hàm getAverageRating(id): Filter reviews, reduce(sum ratings)/count.
* : Hiển thị report: Table với average, sort top performers (sort by average descending).
* : Tích hợp: Thêm averageRating vào employee display.

Các Task Chung (App-wide)

* : Trong app.js, khởi tạo tất cả module, kiểm tra auth, render dashboard với menu link đến các module.
* : Xử lý lỗi toàn cục: Sử dụng window.onerror hoặc try-catch ở các hàm chính.
* : Refresh UI: Hàm refreshDashboard() để reload tables/lists sau thay đổi (sử dụng event dispatch nếu cần).
* : Testing: Viết unit tests đơn giản (console.log) cho validation, HOF, async.
* : Performance: Giới hạn localStorage (kiểm tra size trước save), sử dụng debounce cho search inputs.