

**ASSIGNMENT REPORT**

# HRM

| Semester: | Programming Fundamentals |
| --- | --- |
| Class: | WD1113 |
| Instructor: | Cuong Nguyen Dinh |
| Members: | Phan Nhat Quan (Alexander Phan) |

### **1. Giới Thiệu Mục Đích Báo Cáo**

Báo cáo này nhằm mục đích trình bày chi tiết về quá trình xây dựng ứng dụng Quản lý Nhân sự (HRM) bằng JavaScript thuần. Nội dung chính tập trung vào ba phần: (1) Cách thức triển khai từng module chức năng, (2) Những thách thức kỹ thuật gặp phải và giải pháp, và (3) Phương pháp kiểm thử đã được áp dụng để đảm bảo chất lượng của ứng dụng.

### **2. Mô Tả Quá Trình Triển Khai Các Module**

* **Cách triển khai:**
  + **EmployeeDbModule.js** được xây dựng như một lớp trừu tượng (abstraction layer) để giao tiếp với localStorage. Nó đóng vai trò là "cơ sở dữ liệu" phía client, cung cấp các phương thức CRUD (Create, Read, Update, Delete) cho tất cả các module khác. Dữ liệu được lưu dưới dạng chuỗi JSON, do đó các hàm JSON.stringify() và JSON.parse() được sử dụng thường xuyên.
  + **AuthModule.js** xử lý logic đăng ký và đăng nhập. Một điểm nhấn kỹ thuật là việc sử dụng **closure** để tạo ra một hàm "mã hóa" mật khẩu đơn giản. Mặc dù không mang lại khả năng bảo mật thực sự, đây là một cách hiệu quả để áp dụng kiến thức đã học vào thực tế, ngăn logic xử lý mật khẩu bị lộ ra ở phạm vi toàn cục (global scope).
* **Thách thức:** Quản lý dữ liệu dạng chuỗi trong localStorage và phải chuyển đổi qua lại liên tục là một việc dễ gây nhầm lẫn ban đầu.

#### **2.2. Module Quản lý Nhân viên (Thêm, Sửa, Xóa, Tìm kiếm)**

* **Cách triển khai:**
  + **AddEmployeeModule.js**: Sử dụng các trình lắng nghe sự kiện (addEventListener) để thu thập dữ liệu từ form. Việc xác thực (validate) dữ liệu đầu vào như email và số điện thoại được thực hiện theo thời gian thực bằng **Regular Expressions (Regex)**, giúp cải thiện trải nghiệm người dùng.
  + **EditEmployeeModule.js**: Khi người dùng chọn sửa một nhân viên, module sẽ tìm kiếm nhân viên đó, sau đó tự động điền (preload) thông tin hiện có vào form. Kỹ thuật **closure** được áp dụng để "ghi nhớ" ID của nhân viên đang được sửa, đảm bảo việc cập nhật được thực hiện chính xác.
  + **SearchEmployeeModule.js**: Đây là một trong những module phức tạp nhất, cho phép lọc dữ liệu theo nhiều tiêu chí. Logic cốt lõi là sử dụng hàm bậc cao Array.prototype.filter() kết hợp với các điều kiện để trả về một mảng dữ liệu đã được lọc. Chức năng sắp xếp khi nhấn vào tiêu đề cột được xử lý bằng Array.prototype.sort().
* **Thách thức:** Việc cập nhật giao diện (UI) một cách đồng bộ sau mỗi thao tác CRUD là thách thức lớn nhất khi không có framework. Giải pháp là viết các hàm render() riêng, chịu trách nhiệm xóa và vẽ lại bảng dữ liệu mỗi khi có sự thay đổi trong localStorage.

#### **2.3. Module Nghiệp Vụ (Lương, Chấm công, Nghỉ phép, Hiệu suất)**

* **Cách triển khai:**
  + **SalaryModule.js**: Tận dụng triệt để các hàm bậc cao. Hàm map() được dùng để biến đổi mảng dữ liệu nhân viên thành một mảng mới chứa lương thực nhận của từng người. Hàm reduce() được dùng để tính tổng lương của toàn công ty một cách hiệu quả.
  + **AttendanceModule.js** và **LeaveModule.js**: Tập trung vào việc sử dụng đối tượng Date của JavaScript để tính toán thời gian làm việc, xử lý ngày bắt đầu/kết thúc và theo dõi số ngày phép còn lại.
  + **DepartmentModule.js**: Logic quan trọng nhất là kiểm tra ràng buộc. Trước khi cho phép xóa một phòng ban, module sẽ kiểm tra xem có nhân viên nào đang thuộc phòng ban đó không. Nếu có, hành động xóa sẽ bị chặn.

### **3. Những Thách Thức Chung và Giải Pháp**

1. **Hệ thống Module ES6 và Môi trường chạy:**
   * **Thách thức:** Khi mới bắt đầu, việc chạy file index.html trực tiếp trên trình duyệt gây ra lỗi CORS do chính sách bảo mật không cho phép tải các file module (import/export) từ file://.
   * **Giải pháp:** Sử dụng một máy chủ HTTP cục bộ. Công cụ **Live Server** của VS Code đã giải quyết triệt để vấn đề này, tạo ra một môi trường phát triển chuyên nghiệp hơn.
2. **Quản lý Trạng thái (State Management):**
   * **Thách thức:** Đây là vấn đề nan giải nhất. Khi không có các thư viện như Redux hay Vuex, việc đảm bảo dữ liệu trên giao diện luôn khớp với dữ liệu trong localStorage qua nhiều module khác nhau rất phức tạp.
   * **Giải pháp:** Áp dụng nguyên tắc "Single Source of Truth", coi localStorage là nguồn dữ liệu duy nhất và đáng tin cậy. Sau mỗi hành động làm thay đổi dữ liệu (thêm, sửa, xóa), giao diện sẽ được render lại hoàn toàn từ dữ liệu mới nhất được đọc từ localStorage. Tuy không tối ưu về hiệu suất, phương pháp này đảm bảo tính nhất quán.

### **4. Phương Pháp Kiểm Thử (Testing)**

Do quy mô dự án, phương pháp kiểm thử chính được áp dụng là **kiểm thử thủ công (Manual Testing)** dựa trên các kịch bản sử dụng (test cases) đã được xác định trước.

**Bảng tóm tắt một số trường hợp kiểm thử tiêu biểu:**

| **Mã TC** | **Tên Kịch bản** | **Mô tả** | **Kết quả Mong đợi** |
| --- | --- | --- | --- |
| **AUTH-01** | Đăng nhập thành công | Người dùng nhập đúng tài khoản admin và mật khẩu admin123. | Đăng nhập thành công và được chuyển hướng đến trang quản trị. |
| **EMP-01** | Thêm nhân viên - Dữ liệu không hợp lệ | Người dùng nhập email sai định dạng vào form thêm nhân viên. | Hệ thống hiển thị thông báo lỗi ngay bên dưới ô nhập liệu, không cho phép gửi form. |
| **EMP-02** | Xóa nhân viên thành công | Người dùng tìm và xóa một nhân viên khỏi danh sách. | Nhân viên bị xóa khỏi localStorage và biến mất khỏi bảng danh sách trên UI. |
| **DEPT-01** | Xóa phòng ban có ràng buộc | Người dùng cố gắng xóa một phòng ban vẫn còn nhân viên. | Hành động bị chặn và một thông báo cảnh báo được hiển thị. |
| **SEARCH-01** | Tìm kiếm nâng cao | Người dùng lọc nhân viên theo cả phòng ban và khoảng lương. | Danh sách kết quả chỉ hiển thị những nhân viên thỏa mãn cả hai điều kiện. |

Quá trình kiểm thử cũng bao gồm các trường hợp đặc biệt (edge cases) như nhập dữ liệu rỗng, số âm, hoặc thực hiện các hành động không theo tuần tự để đảm bảo ứng dụng không gặp lỗi nghiêm trọng.