Thiết Kế Phần Mềm

cho

<Dự Án>

**Phiên bản X.X được phê chuẩn**

**Được chuẩn bị bởi <Tác giả>**

**<Tổ chức>**

**<Ngày tạo ra tài liệu>**

**Mục Lục**

[1. Giới thiệu 1](#_Toc221185124)

[1.1 Mục đích 1](#_Toc221185125)

[1.2 Phạm vi 1](#_Toc221185126)

[1.3 Bảng chú giải thuật ngữ 2](#_Toc221185127)

[1.4 Tài liệu tham khảo 2](#_Toc221185128)

[1.5 Tổng quan về tài liệu 2](#_Toc221185129)

[2. Tổng quan hệ thống 2](#_Toc221185130)

[3. Kiến trúc hệ thống 4](#_Toc221185131)

[3.1 Thiết kế kiến trúc 4](#_Toc221185132)

[3.2 Mô tả sự phân rã 5](#_Toc221185133)

[3.3 Cơ sở thiết kế 7](#_Toc221185134)

[4. Thiết kế dữ liệu 8](#_Toc221185135)

[4.1 Mô tả dữ liệu 8](#_Toc221185136)

[4.2 Từ điển dữ liệu 10](#_Toc221185137)

[5. Thiết kế theo chức năng 10](#_Toc221185138)

[5.1 Chức năng Đăng ký tài khoản 10](#_Toc221185139)

[5.2 Chức năng thêm dự án 11](#_Toc221185140)

[5.3 Chức năng xóa dự án 12](#_Toc221185141)

[5.4 Chức năng chỉnh sửa dự án 13](#_Toc221185142)

[5.5 Chức năng đăng nhập 14](#_Toc221185143)

[5.6 Chức năng quản lý và phân quyền người dùng 15](#_Toc221185144)

[5.7 Chức năng nhận thông báo 17](#_Toc221185145)

[5.8 Chức năng tạo công việc 18](#_Toc221185146)

[5.9 Chức năng xem công việc 19](#_Toc221185147)

[5.10 Chức năng bình luận công việc 21](#_Toc221185148)

[5.11 Chức năng cập nhật trạng thái công việc 23](#_Toc221185149)

[5.12 Chức năng phân quyền dự án 24](#_Toc221185150)

[5.13 Chức năng xem thông kê 25](#_Toc221185151)

[6 Bảng tham khảo tới các yêu cầu 27](#_Toc221185152)

[7 Các phụ lục 27](#_Toc221185153)

**Theo dõi phiên bản tài liệu**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** | **Ngày** | **Lý do thay đổi** | **Phiên bản** |
|  |  |  |  |
|  |  |  |  |

# Giới thiệu

## Mục đích

Mục đích của tài liệu Thiết kế Phần mềm (Software Design Document - SDD) này là cung cấp một bản mô tả toàn diện về kiến trúc hệ thống và thiết kế chi tiết của dự án TeamWork Master (Ứng dụng quản lý công việc nhóm). Tài liệu này chuyển đổi các yêu cầu đã được xác định trong tài liệu "Đặc tả yêu cầu phần mềm" thành các biểu diễn kỹ thuật cụ thể, bao gồm: kiến trúc đa tầng, cơ sở dữ liệu, giao diện người dùng và các thuật toán xử lý nghiệp vụ.

Đặc biệt, tài liệu tập trung mô tả rõ các giải pháp thiết kế để đáp ứng các yêu cầu phi chức năng cốt lõi của môn học: tính tái sử dụng (design for reuse), tính dễ thích ứng với thay đổi (design for change) và khả năng mở rộng thông qua kiến trúc Plugin.

Tài liệu này được biên soạn dành cho các đối tượng độc giả sau:

* Đội ngũ phát triển (Developers): Sử dụng tài liệu làm kim chỉ nam để hiện thực hóa mã nguồn, đặc biệt là việc tuân thủ các Mẫu thiết kế (Design Patterns) và cơ chế nạp Plugin động.
* Đội ngũ kiểm thử (Testers): Hiểu rõ luồng dữ liệu và sự tương tác giữa các thành phần để xây dựng các kịch bản kiểm thử đơn vị (Unit Test) và kiểm thử tích hợp.
* Người bảo trì hệ thống: Nắm bắt cấu trúc hệ thống để thực hiện các nâng cấp hoặc sửa lỗi trong tương lai mà không phá vỡ kiến trúc sẵn có.
* Người đánh giá: Xem xét và đánh giá mức độ đáp ứng các nguyên lý xây dựng phần mềm hiện đại và chất lượng thiết kế của nhóm.

## Phạm vi

Mô tả và phạm vi phần mềm: Phần mềm TeamWork Master là một ứng dụng quản lý công việc nhóm quy mô vừa, được xây dựng trên nền tảng Java. Phạm vi của phần mềm bao gồm các chức năng chính sau:

* Quản lý dự án: Cho phép tạo mới, chỉnh sửa và quản lý vòng đời của dự án.
* Quản lý công việc: Thực hiện các thao tác tạo đầu việc, gán người phụ trách, thiết lập hạn chót (deadline) và cập nhật trạng thái công việc (To Do, In Progress, Done...).
* Quản lý người dùng: Hệ thống phân quyền chặt chẽ giữa các vai trò Admin, Manager và Member.
* Hệ thống thông báo & Tương tác: Tự động gửi cảnh báo khi trạng thái công việc thay đổi hoặc sắp đến hạn; hỗ trợ bình luận và thảo luận trực tiếp trong từng công việc.
* Thống kê & Báo cáo: Cung cấp cái nhìn trực quan về tiến độ dự án và hiệu suất làm việc.

Lợi ích của dự án:

* Cộng tác hiệu quả: Giúp các thành viên trong nhóm làm việc tập trung, nắm bắt kịp thời các thay đổi và phân công từ người quản lý.
* Tăng cường năng suất: Giảm thiểu sự chậm trễ thông qua cơ chế nhắc nhở deadline tự động và theo dõi trạng thái công việc theo thời gian thực.
* Quản lý tập trung: Lưu trữ toàn bộ lịch sử trao đổi, tài liệu và tiến độ của dự án tại một nơi duy nhất.

Mục đích và mục tiêu thiết kế: Dự án không chỉ hướng tới việc xây dựng một công cụ quản lý mà còn tập trung vào việc áp dụng các nguyên lý kỹ thuật phần mềm tiên tiến:

* Khả năng tái sử dụng (Design for reuse): Thiết kế các thành phần Core Framework độc lập để có thể sử dụng lại cho các ứng dụng khác.
* Dễ thích ứng với thay đổi (Design for change): Sử dụng kiến trúc Plugin và cơ chế nạp lớp động (Java Reflection) để thêm tính năng mới mà không cần chỉnh sửa mã nguồn cốt lõi.
* Áp dụng Mẫu thiết kế (Design Patterns): Triển khai linh hoạt các mẫu Observer, Strategy, Factory và Template Method để đảm bảo tính low coupling và high cohesion
* Đảm bảo chất lượng: Xây dựng hệ thống đa luồng (Concurrency) ổn định với Swing Worker và thực hiện kiểm thử đơn vị (Unit Testing) chặt chẽ.

## Bảng chú giải thuật ngữ

|  |  |  |
| --- | --- | --- |
| STT | Thuật ngữ / Từ viết tắt | Định nghĩa / Giải thích |
| 1 | OOP | Hướng đối tượng |

## Tài liệu tham khảo

<

* Đây là mục tùy chọn
* Liệt kê ra bất cứ tài liệu hay địa chỉ website nào mà tài liệu này tham khảo tới. Cung cấp đủ thông tin để người đọc có thể tìm bản sao của từng tài liệu tham khảo, bao gồm: tiêu đề, tác giả, số phát hành, ngày, nguồn hay nơi cung cấp. >

## Tổng quan về tài liệu

<Cung cấp một cái nhìn tổng quan về tài liệu này và sự tổ chức của nó.>

# Tổng quan hệ thống

### 2.1 Mô tả chức năng tổng quan

Hệ thống TeamWork Master là một giải pháp phần mềm quản lý công việc và cộng tác nhóm trên nền tảng Web, được thiết kế để hỗ trợ các đội nhóm quy mô vừa trong việc lập kế hoạch, phân công và theo dõi tiến độ dự án.

Các nhóm chức năng cốt lõi của hệ thống bao gồm:

* Quản lý dự án & Công việc: Cung cấp công cụ trực quan (Bảng Kanban, Danh sách) để khởi tạo dự án, phân rã công việc (Tasks), thiết lập thời hạn (Deadline) và cập nhật trạng thái tiến độ theo thời gian thực.
* Cộng tác nhóm trực tuyến: Tích hợp các tính năng tương tác như bình luận, đính kèm tài liệu và hệ thống thông báo tự động (Notification) giúp các thành viên nắm bắt thông tin kịp thời mà không phụ thuộc vào không gian địa lý.
* Quản trị & Phân quyền: Hệ thống quản lý danh tính người dùng tập trung với cơ chế phân quyền dựa trên vai trò (RBAC - Role Based Access Control) gồm: Quản trị viên (Admin), Quản lý dự án (Manager) và Thành viên (Member).
* Khả năng mở rộng: Hệ thống cho phép tích hợp thêm các tính năng mới (như Báo cáo thống kê, Xuất dữ liệu) thông qua cơ chế Plugin mà không làm gián đoạn hoạt động của phần mềm.

### 2.2 Ngữ cảnh hệ thống

Khác với các ứng dụng Desktop truyền thống, TeamWork Master vận hành theo mô hình Client-Server trên môi trường mạng:

* Phía Máy khách (Client - Frontend):
  + Là ứng dụng đơn trang (Single Page Application - SPA) được xây dựng bằng Vue.js.
  + Người dùng truy cập hệ thống thông qua các trình duyệt web phổ biến (Google Chrome, Firefox, Microsoft Edge, Safari) trên máy tính hoặc thiết bị di động mà không cần cài đặt phần mềm.
* Phía Máy chủ (Server - Backend):
  + Là ứng dụng Java chạy trên máy chủ, đóng vai trò trung tâm xử lý nghiệp vụ, lưu trữ dữ liệu và quản lý các Plugin.
  + Giao tiếp với Client thông qua chuẩn RESTful API bảo mật.
* Môi trường vận hành: Hệ thống yêu cầu kết nối Internet hoặc mạng nội bộ (LAN) ổn định để đồng bộ dữ liệu giữa người dùng và máy chủ.

### 2.3 Thiết kế tổng thể và Kiến trúc

Hệ thống áp dụng kiến trúc lai giữa Phân tầng (Layered Architecture) và Microkernel (Plugin-based) để đáp ứng các yêu cầu về hiệu năng và khả năng mở rộng:

1. Kiến trúc Phân tách (Decoupled Architecture):
   * Tầng Giao diện (Presentation Layer): Được tách biệt hoàn toàn tại phía Client (Vue.js). Nhiệm vụ duy nhất là hiển thị dữ liệu và thu thập tương tác người dùng.
   * Tầng Dịch vụ (Service Layer): Nằm tại phía Server (Java), cung cấp các API để xử lý logic nghiệp vụ.
   * Tầng Dữ liệu (Data Access Layer): Quản lý việc truy xuất cơ sở dữ liệu và hệ thống tệp tin.
2. Cơ chế Plugin phía Server (Server-side Microkernel):
   * Core Framework: Là hạt nhân của hệ thống Backend, quản lý vòng đời ứng dụng, kết nối cơ sở dữ liệu và bảo mật.
   * Plugin Loader: Sử dụng kỹ thuật Java Reflection để quét, nạp và kích hoạt các file .jar (Plugin) nằm trong thư mục mở rộng của máy chủ.
   * Lợi ích: Cho phép cập nhật hoặc thêm mới tính năng nghiệp vụ (ví dụ: thêm module "Chấm công") bằng cách tải lên file Plugin mới mà không cần biên dịch lại toàn bộ mã nguồn của hệ thống Core.

### 2.4 Thông tin nền tảng

Thiết kế của hệ thống tuân thủ các nguyên lý và tiêu chuẩn kỹ thuật sau:

* Giao thức giao tiếp: Sử dụng HTTPS để truyền tải dữ liệu dưới định dạng JSON, đảm bảo tính nhẹ và bảo mật.
* Cơ chế xác thực: Sử dụng JSON Web Token (JWT) để xác thực không trạng thái (Stateless), phù hợp với kiến trúc RESTful và khả năng mở rộng (Scalability) của ứng dụng Web.
* Mẫu thiết kế (Design Patterns): Áp dụng linh hoạt các mẫu thiết kế hướng đối tượng:
  + Strategy Pattern: Để thay đổi linh hoạt các thuật toán xử lý trong Plugin.
  + Observer Pattern: Để xử lý hệ thống thông báo sự kiện (Event-driven) giữa Core và Plugin.
  + DAO (Data Access Object): Để trừu tượng hóa việc truy cập dữ liệu.

# Kiến trúc hệ thống

## Thiết kế kiến trúc

Hệ thống TeamWork Master được xây dựng dựa trên mô hình kiến trúc phân tán Client-Server kết hợp với mẫu kiến trúc Microkernel (Plugin-based) tại phía máy chủ.

Mô hình này cho phép tách biệt hoàn toàn phần giao diện người dùng (Frontend) khỏi phần xử lý nghiệp vụ (Backend), đồng thời đảm bảo khả năng mở rộng linh hoạt thông qua các Plugin mà không cần can thiệp vào mã nguồn lõi.

### 3.1.1. Sơ đồ kiến trúc tổng quan (High-Level Architecture)

- Tầng Client (Presentation Layer):

* + Chạy trên trình duyệt của người dùng.
  + Công nghệ: Vue.js (Single Page Application).
  + Thành phần: Vue Router (điều hướng), Pinia (quản lý trạng thái), Axios (gọi API).
* Tầng Server (Application Layer - The Microkernel):
  + Chạy trên máy chủ (Web Server).
  + Công nghệ: Java Core Framework (tự xây dựng hoặc dùng Spring Boot).
  + Thành phần cốt lõi:
    - API Gateway / Controllers: Tiếp nhận yêu cầu RESTful từ Client.
    - Core System: Quản lý xác thực (Auth), kết nối CSDL, và ghi log.
    - Plugin Loader: Bộ phận chịu trách nhiệm quét, nạp và quản lý vòng đời của các Plugin (.jar).
* Tầng Dữ liệu (Data Layer):
  + Lưu trữ dữ liệu bền vững.
  + Công nghệ: MySQL (hoặc H2 cho môi trường nhúng).

### 3.1.2. Mô tả các thành phần và trách nhiệm

**a. Phân hệ Frontend (Vue.js Client)**

* Vai trò: Là điểm tiếp xúc duy nhất với người dùng. Chịu trách nhiệm hiển thị dữ liệu và thu thập thao tác.
* Trách nhiệm:
  + Render giao diện động (Dynamic UI) dựa trên dữ liệu JSON trả về từ Server.
  + Xử lý logic hiển thị (ví dụ: Kéo thả Kanban, Validate form).
  + Gửi các HTTP Request (GET, POST, PUT, DELETE) kèm theo Token xác thực tới Backend.

**b. Phân hệ Backend Core (The Host/Kernel)**

* Vai trò: Là bộ khung xương sống của hệ thống, hoạt động như một "Container" chứa các dịch vụ nền tảng và các Plugin.
* Trách nhiệm:
  + Cung cấp API: Expose các RESTful Endpoints để Frontend giao tiếp.
  + Quản lý Plugin: Sử dụng Java Reflection để nạp các file .jar từ thư mục /plugins khi khởi động.
  + Bảo mật: Xác thực người dùng thông qua JWT và phân quyền truy cập API.
  + Host Context: Cung cấp các giao diện (IHostContext) để Plugin có thể gọi ngược lại (Callback) nhằm truy xuất dữ liệu hoặc ghi log.

**c. Các Plugin mở rộng (Extension Modules)**

* **Vai trò:** Chứa các logic nghiệp vụ cụ thể hoặc các tính năng bổ sung.
* Đặc điểm:
  + Được đóng gói thành các file .jar độc lập.
  + Phải hiện thực giao diện IPlugin mà Core quy định.
* Ví dụ:
  + Plugin Thống kê: Tính toán số liệu và trả về JSON cho biểu đồ.
  + Plugin Xuất báo cáo: Tạo file PDF từ dữ liệu dự án.
  + Plugin Thông báo: Gửi email nhắc nhở.

### 3.1.3. Cơ chế cộng tác (Collaboration)

Quy trình xử lý một yêu cầu điển hình (ví dụ: "Xem thống kê dự án") diễn ra như sau:

* Client: Người dùng nhấn menu "Thống kê". Vue.js gửi Request GET /api/plugins/stats/project/1 tới Server.
* Core (Controller): Nhận Request, kiểm tra Token xác thực của người dùng. Nếu hợp lệ, chuyển tiếp yêu cầu tới bộ phận quản lý Plugin.
* Core (Plugin Manager): Tìm kiếm Plugin có ID là stats. Nếu Plugin đang hoạt động, Core gọi phương thức xử lý của Plugin đó.
* Plugin: Thực thi logic tính toán. Nếu cần dữ liệu thô từ Database, Plugin gọi hostContext.getDatabaseConnection().
* Core: Nhận kết quả từ Plugin, đóng gói thành JSON chuẩn và trả về cho Client.

- Client: Vue.js nhận JSON và vẽ biểu đồ lên màn hình.

**3.1.4. Các lựa chọn kiến trúc (Architectural Choices)**

* Tại sao chọn kiến trúc Web (Client-Server)?
  + Để đáp ứng nhu cầu truy cập mọi lúc mọi nơi của các nhóm làm việc từ xa, điều mà ứng dụng Desktop thuần túy khó đáp ứng.
  + Giúp việc cập nhật giao diện dễ dàng hơn (chỉ cần deploy lại Frontend, người dùng không cần cài lại App).
* Tại sao chọn Microkernel (Plugin-based) cho Backend?
  + Để thỏa mãn yêu cầu "Design for Change" và "Design for Reuse".
  + Cho phép phát triển song song: Một nhóm làm Core, các nhóm khác làm Plugin mà không dẫm chân lên code của nhau.
  + Giảm rủi ro: Lỗi trong một Plugin (ví dụ: Plugin báo cáo bị crash) sẽ được Core cô lập (try-catch), không làm sập toàn bộ hệ thống quản lý dự án chính.
* Tại sao dùng RESTful API?
  + Để giảm sự phụ thuộc (Decoupling) giữa Frontend và Backend. Team Vue.js và Team Java có thể làm việc độc lập, chỉ cần thống nhất tài liệu API (Swagger).

## Mô tả sự phân rã

### 3.2.1. Phân hệ Giao diện Người dùng (Frontend Subsystem)

Đây là phân hệ chạy trên trình duyệt (Client-side), được phát triển bằng Vue.js. Nó chịu trách nhiệm hiển thị và quản lý trạng thái giao diện. Phân hệ này được chia thành các gói (packages) sau:

* views (Màn hình): Chứa các thành phần giao diện cấp cao tương ứng với các trang của ứng dụng.
  + *DashboardView:* Hiển thị tổng quan dự án.
  + *ProjectDetailView:* Hiển thị bảng Kanban và chi tiết công việc.
  + *LoginView:* Màn hình đăng nhập.
* components (Thành phần tái sử dụng): Chứa các UI Component nhỏ, độc lập.
  + *Base:* Các nút bấm, ô nhập liệu chuẩn.
  + *Business:* Thẻ công việc (TaskCard), Cột trạng thái (KanbanColumn).
* stores (Quản lý trạng thái - Pinia): Nơi lưu trữ dữ liệu tạm thời của ứng dụng tại phía Client.
  + *AuthStore:* Lưu Token đăng nhập và thông tin User hiện tại.
  + *DataStore:* Lưu danh sách Dự án/Công việc để giảm tải việc gọi API liên tục.
* services (Giao tiếp API):
  + *AxiosClient:* Cấu hình HTTP Client để gửi Request kèm JWT Token.
  + *PluginRenderer:* Module đặc biệt giúp hiển thị động các thành phần giao diện (như Menu, Widget) dựa trên cấu hình JSON mà Backend trả về (hỗ trợ tính năng mở rộng của Plugin).

### 3.2.2. Phân hệ Lõi Backend (Backend Core Kernel)

Đây là hệ thống trung tâm chạy trên Server (Java), đóng vai trò là "Vật chứa" (Container) và quản lý nền tảng.

* com.teamwork.core.kernel (Hạt nhân):
  + *PluginManager:* Sử dụng Java Reflection để quét thư mục, nạp các file .jar và quản lý vòng đời (Start/Stop) của các Plugin.
  + *SecurityGuard:* Bộ lọc (Filter) kiểm tra JWT Token và phân quyền truy cập (RBAC) cho từng API Request.
* com.teamwork.core.api (Cổng giao tiếp):
  + *RestControllers:* Các lớp Controller tiếp nhận yêu cầu từ Frontend, validate dữ liệu và điều hướng đến Service xử lý.
* com.teamwork.core.host (Dịch vụ Host):
  + Hiện thực các dịch vụ dùng chung mà Core cung cấp cho Plugin sử dụng (như: *DatabaseConnection*, *Logger*, *EmailService*).

### 3.2.3. Phân hệ Giao diện Kết nối (Shared Contracts)

Đây là thành phần quan trọng nhất để đảm bảo tính "Design for Change". Nó là một thư viện chung (shared-lib.jar) mà cả Core và các Plugin đều phải tham chiếu tới.

* com.teamwork.contract.plugin:
  + Interface IPlugin: Bản hợp đồng bắt buộc cho mọi Plugin mở rộng. Định nghĩa các phương thức: initialize(), start(), stop(), getManifest().
* com.teamwork.contract.host:
  + Interface IHostContext: Bản hợp đồng quy định những gì Plugin được phép yêu cầu từ Core. Ví dụ: getUserService(), getLogService().

### 3.2.4. Phân hệ Các Plugin Nghiệp vụ (Business Plugins)

Các chức năng nghiệp vụ cụ thể được tách hoàn toàn khỏi Core và đóng gói thành các module riêng biệt (các file .jar riêng).

* plugin.project.jar (Quản lý Dự án):
  + Chứa logic CRUD Dự án, Công việc.
  + Chứa thuật toán xử lý logic bảng Kanban.
* plugin.statistics.jar (Thống kê & Báo cáo):
  + Chứa thuật toán tổng hợp dữ liệu và tính toán tiến độ.
* plugin.notification.jar (Thông báo):
  + Sử dụng mẫu Observer để lắng nghe các sự kiện thay đổi từ các Plugin khác và tạo thông báo đẩy.

## Cơ sở thiết kế

### 3.3.1. Các vấn đề then chốt (Key Drivers)

Các quyết định kiến trúc được thúc đẩy bởi bốn yêu cầu phi chức năng quan trọng nhất của hệ thống TeamWork Master:

* Tính linh hoạt và "Thiết kế cho sự thay đổi" (Design for Change):
  + Vấn đề: Trong môi trường phát triển Agile, các yêu cầu nghiệp vụ (như cách tính toán báo cáo, quy trình duyệt task) thường xuyên thay đổi.
  + Giải pháp: Kiến trúc Plugin cho phép đóng gói các logic nghiệp vụ vào các file .jar riêng biệt. Khi cần thay đổi logic, ta chỉ cần thay thế file Plugin mà không cần biên dịch lại hay dừng toàn bộ hệ thống Core.
* Khả năng tái sử dụng (Design for Reuse):
  + Vấn đề: Đội ngũ phát triển muốn xây dựng một nền tảng (Framework) có thể dùng lại cho các dự án tương lai (ví dụ: Hệ thống quản lý nhân sự, Quản lý kho).
  + Giải pháp: Việc tách biệt hoàn toàn Core Framework (chỉ chứa các dịch vụ nền tảng như Auth, Log, DB Connection) khỏi Business Plugins (chứa logic quản lý dự án) giúp bộ Core trở nên "sạch" và sẵn sàng tái sử dụng cho bất kỳ miền nghiệp vụ nào khác.
* Trải nghiệm người dùng hiện đại (Modern UX):
  + Vấn đề: Người dùng yêu cầu sự phản hồi tức thì và khả năng truy cập từ mọi nơi, điều mà ứng dụng Desktop truyền thống hạn chế.
  + Giải pháp: Sử dụng Vue.js (SPA) giúp tải trang một lần và chỉ cập nhật dữ liệu qua API, mang lại trải nghiệm mượt mà như ứng dụng Desktop nhưng vận hành trên trình duyệt Web.
* Sự cô lập lỗi (Fault Isolation):
  + Vấn đề: Một lỗi nhỏ trong tính năng phụ (như "Xuất báo cáo") không được phép làm sập toàn bộ hệ thống.
  + Giải pháp: Core System bao bọc việc thực thi Plugin trong các khối try-catch an toàn. Nếu Plugin bị lỗi, Core sẽ ghi log và thông báo lỗi, nhưng các chức năng khác vẫn hoạt động bình thường.

### 3.3.2. Các thỏa hiệp về thiết kế (Trade-offs)

Để đạt được tính linh hoạt và khả năng mở rộng, nhóm phát triển đã chấp nhận đánh đổi một số yếu tố sau:

* Độ phức tạp ban đầu cao hơn (Initial Complexity vs. Flexibility):
  + Đánh đổi: Việc xây dựng một hệ thống Plugin Loader sử dụng Java Reflection và định nghĩa các Interface trừu tượng tốn nhiều công sức hơn so với việc viết code "cứng" (Hard-coded) trong một kiến trúc nguyên khối (Monolithic).
  + Lý do chấp nhận: Chi phí ban đầu cao nhưng chi phí bảo trì và mở rộng về sau sẽ giảm đáng kể.
* Hiệu năng thực thi (Performance Overhead):
  + Đánh đổi: Việc gọi hàm qua Reflection và giao tiếp qua RESTful API (Serialize/Deserialize JSON) chậm hơn so với gọi hàm trực tiếp trong cùng bộ nhớ.
  + Lý do chấp nhận: Với sức mạnh phần cứng máy chủ hiện nay và bản chất của ứng dụng quản lý công việc (không phải ứng dụng giao dịch tần suất cao như chứng khoán), độ trễ này (vài mili-giây) là không đáng kể và người dùng không nhận thấy sự khác biệt.
* Thách thức trong kiểm thử (Testing Complexity):
  + Đánh đổi: Kiểm thử tích hợp (Integration Test) khó khăn hơn vì Core không biết trước Plugin nào sẽ được nạp.
  + Lý do chấp nhận: Có thể khắc phục bằng cách xây dựng bộ "Mock Plugins" để giả lập các kịch bản kiểm thử cho Core.

### 3.3.3. Lý do lựa chọn công nghệ cụ thể

* Tại sao chọn Java cho Backend?
  + Java có hệ sinh thái mạnh mẽ, tính ổn định cao và đặc biệt là khả năng Reflection (phản chiếu) xuất sắc, là yếu tố sống còn để hiện thực hóa cơ chế nạp Plugin động (Dynamic Loading) mà đồ án yêu cầu.
  + Hỗ trợ đa luồng (Multithreading) mạnh mẽ để xử lý nhiều Request đồng thời từ Client.
* Tại sao chọn Vue.js cho Frontend?
  + Vue.js nhẹ, dễ học và hỗ trợ kiến trúc Component-based. Điều này cho phép tái sử dụng các thành phần giao diện (như TaskCard, KanbanColumn) hiệu quả.
  + Khả năng Reactivity (Phản ứng) của Vue.js giúp cập nhật giao diện ngay lập tức khi dữ liệu thay đổi mà không cần reload trang.
* Tại sao chọn RESTful API & JWT?
  + RESTful là chuẩn giao tiếp phổ biến nhất, giúp giảm sự phụ thuộc (Decoupling) giữa team Frontend và Backend.
  + JWT (JSON Web Token) cho phép xác thực không trạng thái (Stateless), giúp Server không tốn tài nguyên lưu Session, dễ dàng mở rộng khi số lượng người dùng tăng lên.

# Thiết kế dữ liệu

## Mô tả dữ liệu

Hệ thống TeamWork Master quản lý dữ liệu theo mô hình tập trung, đảm bảo tính nhất quán và toàn vẹn trong môi trường đa người dùng. Dữ liệu được tổ chức qua ba tầng chuyển đổi: Tầng Lưu trữ (Database/File), Tầng Đối tượng (Backend Java Entities) và Tầng Trao đổi (JSON DTOs).

### 4.1.1. Chuyển đổi miền thông tin sang cấu trúc dữ liệu (Domain Mapping)

Các thực thể nghiệp vụ trong thế giới thực được ánh xạ thành các lớp đối tượng Java (POJO) và bảng cơ sở dữ liệu tương ứng:

* User (Người dùng):
  + Ánh xạ: Lớp User - Bảng TBL\_USERS.
  + Dữ liệu chính: Định danh (ID), Thông tin xác thực (Username, PasswordHash), Vai trò (Role Enum: ADMIN, MANAGER, MEMBER).
* Project (Dự án):
  + Ánh xạ: Lớp Project - Bảng TBL\_PROJECTS.
  + Dữ liệu chính: ID, Tên dự án, Mô tả, Thời gian (StartDate, EndDate), Trạng thái (Active/Closed), Chủ sở hữu (OwnerID).
* Task (Công việc):
  + Ánh xạ: Lớp Task - Bảng TBL\_TASKS.
  + Dữ liệu chính: ID, Tiêu đề, Mức độ ưu tiên (Priority), Hạn chót (Deadline), Trạng thái (Status: TODO, IN\_PROGRESS, DONE), Người thực hiện (AssigneeID), Dự án cha (ProjectID).
* Plugin Metadata (Thông tin Plugin):
  + Ánh xạ: Lớp PluginDescriptor - File cấu hình plugin.json (bên trong file .jar) hoặc Bảng TBL\_PLUGINS.
  + Dữ liệu chính: PluginID, Version, ClassPath, Trạng thái kích hoạt.

### 4.1.2. Cách thức xử lý và Tổ chức dữ liệu (Processing & Organization)

Dữ liệu được xử lý theo quy trình khép kín từ Frontend đến Backend:

1. Định dạng trao đổi (Data Exchange Format):
   * Sử dụng JSON (JavaScript Object Notation) làm chuẩn giao tiếp giữa Vue.js (Client) và Java (Server).
   * Dữ liệu không được gửi trực tiếp dưới dạng Entity mà thông qua các đối tượng chuyển giao dữ liệu (DTO - Data Transfer Objects) để ẩn giấu cấu trúc DB và chỉ truyền tải thông tin cần thiết.
   * Ví dụ: TaskDTO chỉ chứa AssigneeName thay vì toàn bộ đối tượng User.
2. Tổ chức trong bộ nhớ (In-Memory Organization):
   * Tại phía Backend, dữ liệu được quản lý bằng các Java Collections (List, Map, Set).
   * Sử dụng ConcurrentHashMap để lưu trữ các dữ liệu truy cập thường xuyên (như Cache cấu hình Plugin) nhằm đảm bảo an toàn luồng (Thread-safety).
3. Cơ chế truy xuất (Data Access Mechanism):
   * Áp dụng mẫu thiết kế DAO (Data Access Object) để trừu tượng hóa việc truy vấn. Core Framework cung cấp các interface như IProjectDAO, IUserDAO.
   * Nguyên tắc "Hộp đen" cho Plugin: Plugin không được phép truy cập trực tiếp vào java.sql.Connection. Mọi thao tác đọc/ghi dữ liệu phải thông qua IHostContext do Core cung cấp để đảm bảo tính toàn vẹn và kiểm soát Transaction.

### 4.1.3. Các mục lưu trữ dữ liệu (Data Storage Items)

Hệ thống sử dụng chiến lược lưu trữ lai (Hybrid Storage) để tối ưu hóa cho từng loại dữ liệu:

* Cơ sở dữ liệu Quan hệ (Relational Database - MySQL):
  + Mục đích: Lưu trữ dữ liệu nghiệp vụ cốt lõi có cấu trúc chặt chẽ và quan hệ ràng buộc.
  + Các bảng chính: users, projects, tasks, comments, activity\_logs.
  + Lý do: Đảm bảo tính nhất quán (ACID), hỗ trợ truy vấn phức tạp và Transaction an toàn.
* Hệ thống Tệp tin (File System Storage):
  + Thư mục /plugins: Lưu trữ các file thực thi mở rộng (.jar).
  + Thư mục /uploads: Lưu trữ các tài liệu đính kèm (Attachments) của công việc (Word, PDF, Images).
  + Lý do: Giảm tải cho Database, dễ dàng backup và quản lý dung lượng.
* Kho dữ liệu mở rộng cho Plugin (Extension Data Store):
  + Bảng TBL\_PLUGIN\_DATA: Lưu trữ dữ liệu cấu hình động của Plugin dưới dạng Key-Value (JSON String).
  + Ví dụ: Plugin "Thống kê" có thể lưu cấu hình màu sắc biểu đồ vào bảng này: Key="CHART\_COLOR", Value="{ 'todo': '#FF0000', 'done': '#00FF00' }".
  + Lý do: Cho phép Plugin lưu dữ liệu riêng mà không cần quyền ALTER TABLE để sửa đổi cấu trúc Database chính.
* Tệp cấu hình hệ thống (Configuration Files):
  + File application.properties: Lưu thông tin kết nối DB, cổng Server, cấu hình Log.
  + File security.key: Lưu trữ khóa bí mật (Secret Key) dùng để ký và xác thực JWT Token.

## Từ điển dữ liệu

<Liệt kê theo bảng chữ cái các thực thể của hệ thống hay dữ liệu chính cùng với kiểu và các mô tả chúng. Nếu ta đã sử dụng mô tả theo chức năng trong mục 3.2, ta sẽ liệt kê tất cả các chức năng và các đối số của chức năng. Nếu ta đã sử dụng mô tả theo hướng đối tượng, liệt kê tất cả các đối tượng, các thuộc tính, các phương thức và các đối số của phương thức theo từng đối tượng.>

# Thiết kế theo chức năng

## Chức năng Đăng ký tài khoản

* Mục đích: Cho phép người dùng khách (Guest) chưa có tài khoản thực hiện cung cấp thông tin cá nhân để tạo lập danh tính mới trong hệ thống. Tài khoản sau khi tạo sẽ được cấp quyền mặc định là Thành viên (Member) để tham gia vào các dự án.
* Giao diện: *Mô tả:* Màn hình hiển thị một Form nằm giữa trang (hoặc một Modal), bao gồm logo ứng dụng phía trên và các trường nhập liệu bên dưới.
* Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | *Textbox* | Rỗng | *Tên đăng nhập: Bắt buộc nhập. Kiểm tra tính duy nhất (Unique).* |
| 2 | *Textbox* | Rỗng | *Họ và tên: Hiển thị tên đầy đủ của người dùng.* |
| 3 | *Textbox* | Rỗng | *Email: Bắt buộc nhập. Phải đúng định dạng email (regex).* |
| 4 | *Password Field* | Rỗng | *Mật khẩu: Tối thiểu 6 ký tự. Hiển thị dạng dấu chấm (•).* |
| 5 | *Password Field* | Rỗng | *Xác nhận mật khẩu: Phải trùng khớp với trường Mật khẩu.* |
| 6 | *Button* | N/A | *Nút "Đăng ký": Chỉ kích hoạt (Enable) khi các trường dữ liệu hợp lệ.* |
| 7 | *Link/Button* | N/A | *Nút "Đăng nhập": Dùng để quay lại nếu đã có tài khoản.* |

* Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_USERS | x |  |  | X (Kiểm tra trùng lặp) |
| 2 | UserDTO (JSON) |  |  |  | Truyền dữ liệu từ Client -> Server |

* Cách xử lý:
  + 1. Tại Client (Vue.js):
* Người dùng nhập thông tin vào Form.
* Hệ thống thực hiện Validate thời gian thực (Client-side):
  + Kiểm tra rỗng.
  + Kiểm tra định dạng Email.
  + So sánh "Mật khẩu" và "Xác nhận mật khẩu".
* Nếu có lỗi: Hiển thị thông báo đỏ ngay dưới trường input tương ứng.
* Nếu hợp lệ: Người dùng nhấn "Đăng ký", Client gửi Request POST /api/auth/register kèm JSON chứa thông tin người dùng.
  + 1. Tại Backend (Java Core):
* Controller nhận Request, gọi xuống Service.
* Service kiểm tra trong Database:
  + SELECT count(\*) FROM TBL\_USERS WHERE username = ? OR email = ?
* Nếu tồn tại: Trả về lỗi "Tên đăng nhập hoặc Email đã tồn tại".
* Nếu không tồn tại:
  + Thực hiện mã hóa mật khẩu (Hashing) bằng BCrypt.
  + Gán vai trò mặc định là MEMBER.
  + Lưu bản ghi mới vào TBL\_USERS.
* Trả về phản hồi thành công (HTTP 201 Created).
  + 1. Phản hồi:
* Client nhận phản hồi thành công: Hiển thị thông báo Toast "Đăng ký thành công" và điều hướng về trang Đăng nhập.
* Hàm/ sự kiện (nếu có): mô tả giải thuật cho từng biến cố bằng sơ đồ hoặc bằng ngôn ngữ giả.
* Các ràng buộc (nếu có):
* Tuân thủ REQ-2 (Maudacta): Tên đăng nhập và Email không được trùng lặp.
* Tuân thủ REQ-6 (Maudacta): Mật khẩu phải được mã hóa trước khi lưu (Bảo mật).
* Tuân thủ REQ-5 (Maudacta): Vai trò mặc định luôn là Member.

## Chức năng thêm dự án

Mục đích: Cho phép người dùng (Manager) khởi tạo một không gian làm việc mới, thiết lập các thông tin cơ bản và thời gian dự kiến.

* Giao diện: hiển thị các ảnh giao diện từ góc nhìn của người sử dụng. Chúng có thể được vẽ bằng tay hay dùng công cụ vẽ tự động. Ta nên tạo ra chúng chính xác như có thể. Ta cũng có thể đánh số cho từng thành phần trong giao diện.
* Các thành phần trong giao diện: ghi vào bảng sau các mô tả về từng thành phần (đã được đánh số) của giao diện.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | *Textbox* | Rỗng | *Bắt buộc nhập. Giới hạn 100 ký tự* |
| 2 | *Textarea* | Rỗng | *Cho phép nhập nhiều dòng* |
| 3 | *DatePicker* | Ngày hiện tại |  |
| 4 | *DatePicker* | Rỗng | *Phải lớn hơn hoặc bằng Ngày bắt đầu* |
| 5 | *Checkbox* | Checked (True) | *Nếu chọn, hệ thống tự động insert dữ liệu mẫu vào bảng KanbanColumn* |
| 6 | *Button* | N/A | *Nút “Lưu lại”. Gọi hàm CreateProject.* |
| 7 | *Button* | N/A | *Nút “Hủy bỏ”. Đóng dialog.* |

* Dữ liệu được sử dụng: liệt kê các bảng trong cơ sở dữ liệu hoặc các cấu trúc dữ liệu được cần đến bởi chức năng này.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | PROJECT | X |  |  | Tạo bản ghi dự án mới |
| 2 | USER\_PROJECT | X |  |  | Gán người tạo là “OWNER” |
| 3 | KANBAN\_COLUMN | X |  |  | Tạo các cột mẫu |

* Cách xử lý: giải thích bằng lời hoặc vẽ sơ đồ mô tả dòng xử lý trên giao diện.

1. Người dùng nhấn nút “+” trên thanh Sidebar.
2. Hệ thống hiển thị Dialog nhập liệu.
3. Người dùng nhập thông tin và nhấn “Lưu lại”
4. Hệ thống Validate dữ liệu -> Lưu xuống DB -> Thông báo thành công -> Chuyển hướng sang trang chi tiết dự án.

* Hàm/ sự kiện (nếu có): mô tả giải thuật cho từng biến cố bằng sơ đồ hoặc bằng ngôn ngữ giả.
* Các ràng buộc (nếu có): ví dụ như tham khảo đặc tả nào của tài liệu đặc tả nào.

## Chức năng xóa dự án

Mục đích:Xóa bỏ hoàn toàn dự án và dữ liệu liên quan khỏi hệ thống. Đây là chức năng nhạy cảm cần quy trình xác thực kỹ.

* Giao diện: hiển thị các ảnh giao diện từ góc nhìn của người sử dụng. Chúng có thể được vẽ bằng tay hay dùng công cụ vẽ tự động. Ta nên tạo ra chúng chính xác như có thể. Ta cũng có thể đánh số cho từng thành phần trong giao diện.
* Các thành phần trong giao diện: ghi vào bảng sau các mô tả về từng thành phần (đã được đánh số) của giao diện.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | *Textbox* | Rỗng | *Yêu cầu nhập đúng chính xác tên dự án (Case-sensitive).* |
| 2 | *Button* | Disabled | *Chỉ Enable khi (1) khớp với tên dự án. Màu đỏ cảnh báo.* |
| 3 | *Button* | N/A | *Đóng dialog.* |

* **Dữ liệu được sử dụng**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | PROJECT |  |  | X | Xóa bản ghi chính |
| 2 | TASK |  |  | X | Xóa cascade tất cả task con |
| 3 | ATTACHMENT |  |  | X | Xóa file vật lý trên ổ cứng. |

* Cách xử lý:

1. Người dùng nhấn "Xóa dự án" ở góc khuất màn hình cài đặt.
2. Hệ thống đếm số lượng Task/File liên quan và hiển thị Dialog cảnh báo.
3. Người dùng nhập tên dự án để mở khóa nút "Xác nhận".
4. Khi nhấn "Xác nhận Xóa":

* Hệ thống kiểm tra quyền lần cuối (Server-side check).
* Gửi sự kiện: OnProjectDeleting(ID) cho các Plugin (để Plugin tự dọn dẹp dữ liệu của nó).
* Thực hiện xóa dữ liệu Core.
* Điều hướng về trang chủ.
* Hàm/ sự kiện (nếu có): mô tả giải thuật cho từng biến cố bằng sơ đồ hoặc bằng ngôn ngữ giả.
* Các ràng buộc (nếu có):
* Tham khảo REQ-8, REQ-9, REQ-10 trong tài liệu SRS.
* Phải đảm bảo tính toàn vẹn dữ liệu: Không được để lại "dữ liệu rác" (Orphan data) trong DB sau khi xóa dự án.

## 5.4 Chức năng chỉnh sửa dự án

Mục đích: Cho phép Manager cập nhật thông tin metadata, thay đổi trạng thái hoặc gia hạn thời gian dự án.

Giao diện: hiển thị các ảnh giao diện từ góc nhìn của người sử dụng. Chúng có thể được vẽ bằng tay hay dùng công cụ vẽ tự động. Ta nên tạo ra chúng chính xác như có thể. Ta cũng có thể đánh số cho từng thành phần trong giao diện.

Các thành phần trong giao diện: ghi vào bảng sau các mô tả về từng thành phần (đã được đánh số) của giao diện.

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | *Textbox* | Tên dự án cũ | *Bắt buộc nhaapjj.* |
| 2 | *Combobox* | Trạng thái hiện tại | *Các giá trị: Active, Paused, Archived, Closed.* |
| 3 | *DatePicker* | Ngày kết thúc cũ | *Cần kiểm tra logic với các Task con bên trong.* |
| 4 | *Button* | N/A | *Nút “Cập nhật”* |
| 5 | *Button* | N/A | *Quay về Dashboard.* |

* **Dữ liệu được sử dụng**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | PROJECT |  | X |  | Cập nhật thông tin dự án |
| 2 | TASK |  |  |  | Truy vấn (Read-only) để kiểm tra xung đột deadline. |
| 3 | ACTIVITY\_LOG | X |  |  | Ghi log thay đổi. |

* Cách xử lý:

1. Hệ thống load thông tin dự án hiện tại lên Form.
2. Người dùng thay đổi "Ngày kết thúc" sớm hơn dự kiến.
3. Khi nhấn "Cập nhật", hệ thống quét toàn bộ các Task trong dự án.
4. Nếu phát hiện có Task nào có Deadline > NewProjectEndDate, hiển thị cảnh báo.
5. Nếu người dùng xác nhận hoặc không có lỗi -> Lưu cập nhật.

* Hàm/ sự kiện (nếu có): mô tả giải thuật cho từng biến cố bằng sơ đồ hoặc bằng ngôn ngữ giả.
* Các ràng buộc (nếu có):
* Chỉ người Manager mới nhìn thấy màn hình này.

## Chức năng đăng nhập

Mục đích: Cho phép người dùng đã có tài khoản xác thực danh tính để truy cập vào hệ thống. Hệ thống sẽ cấp quyền truy cập vào các tài nguyên và chức năng tương ứng dựa trên vai trò (Admin, Manager, Member) đã được gán cho tài khoản đó. Đây là cổng kiểm soát an ninh chính, đảm bảo chỉ những người dùng hợp lệ mới có thể tương tác với dữ liệu dự án.

Giao diện: Mô tả: Màn hình được thiết kế tối giản, tập trung vào form đăng nhập nằm giữa màn hình. Bao gồm Logo ứng dụng, các trường nhập liệu và các liên kết điều hướng phụ (Đăng ký, Quên mật khẩu). (Tại đây bạn có thể chèn hình ảnh Mockup giao diện đăng nhập)

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Textbox | Rỗng | Tên đăng nhập: Bắt buộc nhập. |
| 2 | Password Field | Rỗng | Mật khẩu: Bắt buộc nhập. |
| 3 | Checkbox | Unchecked (False) | Ghi nhớ đăng nhập (Remember Me): Tùy chọn. Nếu chọn, lưu token vào LocalStorage lâu dài. |
| 4 | Button | N/A | Nút "Đăng nhập": Gửi yêu cầu xác thực về Server. |
| 5 | Hyperlink | N/A | Liên kết "Đăng ký ngay": Chuyển hướng sang màn hình Đăng ký. |
| 6 | Hyperlink | N/A | Liên kết "Quên mật khẩu": (Tùy chọn mở rộng). |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_USERS |  |  |  | X (Tìm user theo username & so khớp password) |
| 2 | AuthResponseDTO (JSON) |  |  |  | Trả về Token (JWT) và Role |

Cách xử lý:

1. Tại Client (Vue.js):
   * Người dùng nhập "Tên đăng nhập" và "Mật khẩu".
   * Nhấn nút "Đăng nhập".
   * Hệ thống kiểm tra sơ bộ (Client-side): Nếu bỏ trống trường nào thì báo lỗi ngay tại trường đó.
   * Gửi Request POST /api/auth/login chứa thông tin đăng nhập lên Backend.
2. Tại Backend (Java Core):
   * Nhận yêu cầu, tìm kiếm bản ghi trong bảng TBL\_USERS dựa trên username.
   * Nếu không tìm thấy User hoặc User bị khóa (isActive = falseTrả về lỗi.
   * Nếu tìm thấy, sử dụng thuật toán BCrypt để so sánh mật khẩu nhập vào với mật khẩu đã mã hóa trong DB.
   * Nếu sai mật khẩu Trả về lỗi "Thông tin đăng nhập không chính xác".
   * Nếu đúng Sinh mã JWT Access Token (chứa thông tin: UserID, Role, ExpirationTime).
   * Trả về Client mã Token và thông tin User cơ bản.
3. Phản hồi:
   * Client nhận Token, lưu vào localStorage hoặc cookie.
   * Lưu thông tin User vào Pinia Store (State Management).
   * Dựa vào Role trả về để điều hướng:
     + Nếu là Admin Chuyển đến trang Quản trị Hệ thống.
     + Nếu là Manager/Member Chuyển đến Dashboard Dự án.

Các ràng buộc:

* Tuân thủ REQ-2 (Maudacta): Mật khẩu phải được che dấu trên giao diện.
* Tuân thủ REQ-5 (Maudacta): Thông báo lỗi phải chung chung ("Tên đăng nhập hoặc mật khẩu sai") để tránh bị dò quét tài khoản.
* Tuân thủ SEC-1 (Maudacta): Sử dụng cơ chế xác thực không trạng thái (Stateless) với JWT.

## Chức năng quản lý và phân quyền người dùng

Mục đích: Cung cấp cho Quản trị viên (Administrator) khả năng giám sát và kiểm soát toàn bộ tài khoản trong hệ thống. Chức năng này cho phép tìm kiếm người dùng, thay đổi vai trò (phân quyền) để cấp quyền truy cập vào các tính năng quản lý, hoặc khóa tài khoản để ngăn chặn truy cập trái phép.

Giao diện: Mô tả: Màn hình chính là một danh sách (Table) hiển thị tất cả người dùng với các cột thông tin. Phía trên có thanh tìm kiếm và bộ lọc. Khi nhấn nút "Chỉnh sửa" trên một dòng, một hộp thoại (Modal) sẽ hiện ra cho phép thay đổi Vai trò và Trạng thái. (Tại đây có thể chèn hình ảnh Mockup giao diện Admin Panel)

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Textbox | Rỗng | Thanh tìm kiếm: Cho phép nhập Tên hoặc Email để lọc danh sách. |
| 2 | Table/Grid | N/A | Danh sách người dùng: Hiển thị các cột: ID, Username, Email, Role, Status. |
| 3 | Button (Icon) | N/A | Nút "Chỉnh sửa" (trên từng dòng): Mở Modal cập nhật thông tin. |
| 4 | Combobox (Modal) | Vai trò hiện tại | Chọn Vai trò: Chứa các giá trị: MEMBER, MANAGER, ADMIN. |
| 5 | Toggle/Switch (Modal) | Trạng thái hiện tại | Trạng thái hoạt động: ON (Active) / OFF (Banned) |
| 6 | Button | N/A | Nút "Lưu thay đổi": Gửi yêu cầu cập nhật về Server. |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_USERS |  | x |  | X (Lấy danh sách) |
| 2 | TBL\_ACTIVITY\_LOG | X |  |  |  |

Cách xử lý:

1. Tại Client (Admin Dashboard):
   * Admin truy cập menu "Quản lý người dùng". Hệ thống tải danh sách User từ API (GET /api/admin/users).
   * Admin nhập từ khóa vào ô tìm kiếm -> Danh sách tự động lọc lại (Client-side hoặc Server-side filtering).
   * Admin nhấn nút "Sửa" trên dòng của User A. Modal hiện ra với thông tin của User A.
   * Admin đổi Role từ "Member" sang "Manager" và nhấn "Lưu".
   * Client gửi Request PUT /api/admin/users/{id} chứa thông tin Role/Status mới.
2. Tại Backend (Java Core):
   * Middleware kiểm tra: Người gửi Request có phải là ADMIN không? (Nếu không -> 403 Forbidden) .
   * Kiểm tra logic nghiệp vụ: User đang bị sửa có phải là chính Admin đang đăng nhập không? (Tránh tự hạ quyền/tự khóa) .
   * Cập nhật thông tin vào TBL\_USERS.
   * Ghi lại hành động vào TBL\_ACTIVITY\_LOG.
   * Trả về kết quả thành công.

Các ràng buộc:

* Tuân thủ REQ-5 (Maudacta): Hệ thống phải chặn việc Admin tự khóa hoặc tự hạ quyền chính mình.
* Tuân thủ REQ-6 (Maudacta): Mọi thay đổi phải được ghi Log để tra cứu sau này.
* Tuân thủ SEC-2 (Maudacta): API này phải được bảo vệ nghiêm ngặt, chỉ cho phép Role = ADMIN truy cập.

## Chức năng nhận thông báo

Mục đích: Giúp người dùng nắm bắt kịp thời các sự kiện quan trọng liên quan đến công việc và dự án (như được gán việc mới, sắp đến hạn chót, hoặc có bình luận mới). Tính năng này đảm bảo sự cộng tác liên tục và giảm thiểu độ trễ trong quy trình làm việc nhóm.

Giao diện: Mô tả: Biểu tượng "Cái chuông" (Notification Bell) nằm cố định ở góc phải thanh Topbar. Khi có thông báo mới, một số đếm (Badge) màu đỏ sẽ xuất hiện. Khi nhấp vào chuông, một danh sách thả xuống (Dropdown) hiện ra chứa các thông báo gần nhất. (Tại đây có thể chèn hình ảnh Mockup của thanh Topbar và Dropdown thông báo)

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Button (Icon) | Hình cái chuông | Nút Chuông: Luôn hiển thị. Click để Toggle danh sách. |
| 2 | Label (Badge) | Ẩn (0) | Số lượng chưa đọc: Màu đỏ, bo tròn. Hiển thị số (1, 2, 99+). |
| 3 | List Container | Ẩn | Khung danh sách: Chứa các mục thông báo (Item). Scrollable. |
| 4 | List Item | N/A | Mục thông báo: Gồm Tiêu đề, Nội dung tóm tắt, Thời gian (ví dụ: "2 giờ trước"), Trạng thái (Đậm = Chưa đọc). |
| 5 | Hyperlink/Button | N/A | Nút "Đánh dấu tất cả là đã đọc": Tiện ích nhanh. |
| 6 | Toast Message | Ẩn | Thông báo nổi (Popup): Xuất hiện góc màn hình rồi tự biến mất khi có tin mới đến. |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_NOTIFICATIONS | X (Tự động) | X (Đổi trạng thái đã đọc) |  | X (Lấy danh sách theo UserID) |
| 2 | NotificationDTO (JSON) |  |  |  | Truyền qua WebSocket/API |

Cách xử lý:

1. Cơ chế Sinh thông báo (Trigger - Backend):
   * Khi một nghiệp vụ xảy ra (ví dụ: CreateTask), Service sẽ bắn một sự kiện (Event) vào NotificationService.
   * NotificationService tạo bản ghi vào TBL\_NOTIFICATIONS (trạng thái isRead = false).
   * Đồng thời, gửi tin nhắn thời gian thực qua WebSocket tới User đích.
2. Cơ chế Nhận và Hiển thị (Client - Vue.js):
   * Client lắng nghe kênh WebSocket (hoặc Polling định kỳ 30s/lần).
   * Khi nhận gói tin JSON mới:
     + Tăng biến đếm unreadCount.
     + Hiển thị Toast Message góc màn hình: "Bạn có thông báo mới".
     + Đẩy thông báo mới nhất lên đầu danh sách trong Dropdown.
3. Tương tác Người dùng:
   * Người dùng nhấp vào một thông báo trong danh sách.
   * Client gửi Request PUT /api/notifications/{id}/read.
   * Client điều hướng (Navigate) sang trang chi tiết (ví dụ: Trang chi tiết công việc).
   * Giảm số unreadCount.

Các ràng buộc:

* Tuân thủ REQ-2 (Maudacta): Hệ thống phải quét deadline và gửi thông báo tự động trước 24h.
* Tuân thủ REQ-6 (Maudacta): Phải phân biệt rõ ràng trạng thái Đã đọc (nhạt màu) và Chưa đọc (đậm màu).
* Tuân thủ PER-X: Việc nhận thông báo không được làm gián đoạn trải nghiệm người dùng (Non-blocking).

## Chức năng tạo công việc

Mục đích: Cho phép người dùng (Trưởng nhóm hoặc Thành viên được cấp quyền) khởi tạo một nhiệm vụ mới trong dự án. Tại đây, người dùng thiết lập các thông tin quan trọng như tên công việc, người chịu trách nhiệm, hạn hoàn thành và mức độ ưu tiên để phân phối khối lượng công việc cho nhóm.

Giao diện: Mô tả: Giao diện là một Hộp thoại (Modal Dialog) xuất hiện đè lên màn hình Chi tiết dự án. Việc sử dụng Modal giúp người dùng giữ được ngữ cảnh làm việc hiện tại mà không phải tải lại trang. Bố cục gồm tiêu đề hộp thoại, các trường nhập liệu được chia cột hợp lý và hàng nút tác vụ ở dưới cùng

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Textbox | Rỗng | Tên công việc: Bắt buộc nhập. Tối đa 255 ký tự. |
| 2 | Textarea | Rỗng | Mô tả: Nhập chi tiết yêu cầu công việc (Tùy chọn). |
| 3 | Combobox | "Unassigned" | Người thực hiện: Danh sách lấy từ thành viên của dự án hiện tại. |
| 4 | DatePicker | Rỗng | Hạn hoàn thành (Deadline): Bắt buộc. Phải >= Ngày hiện tại. |
| 5 | Radio Group | Medium (Trung bình) | Mức độ ưu tiên: Thấp, Trung bình, Cao. |
| 6 | Label (Hidden) | "To Do" | Trạng thái: Mặc định là "To Do" (không hiển thị để sửa) |
| 7 | Button | N/A | Nút "Lưu": Gửi dữ liệu tạo mới. |
| 8 | Button | N/A | Nút "Hủy": Đóng Modal, không lưu. |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_TASKS | X |  |  |  |
| 2 | TBL\_PROJECTS |  |  |  | X (Kiểm tra ngày kết thúc dự án) |
| 3 | UserDTO (List) |  |  |  | X (Load danh sách thành viên vào Combobox) |
| 4 | TBL\_NOTIFICATIONS | x |  |  | (Tạo thông báo cho người được gán) |

Cách xử lý:

1. Tại Client (Vue.js):
   * Người dùng nhấn nút "Thêm công việc" (+) trên bảng Kanban.
   * Hệ thống hiển thị Modal.
   * Người dùng nhập liệu và nhấn "Lưu".
   * Validate Client:
     + Tên công việc không được rỗng.
     + Hạn hoàn thành phải đúng định dạng ngày tháng.
   * Nếu hợp lệ, Client gửi Request POST /api/projects/{projectId}/tasks.
2. Tại Backend (Java Core):
   * Validate Server:
     + Kiểm tra AssigneeId có thuộc về Project này không.
     + Kiểm tra Deadline của Task có vượt quá EndDate của Project không (theo quy tắc BR-2).
   * Lưu dữ liệu:
     + Insert bản ghi vào TBL\_TASKS với trạng thái mặc định là TO\_DO.
   * Hậu xử lý:
     + Ghi Log hoạt động ("User A created Task B").
     + Kích hoạt Service gửi thông báo đến người được gán việc.
   * Trả về đối tượng Task vừa tạo (kèm ID).
3. Phản hồi UI:
   * Client nhận dữ liệu Task mới.
   * Thêm ngay Task đó vào cột "To Do" trên bảng Kanban (không cần reload trang).
   * Hiển thị thông báo "Tạo công việc thành công".
   * Đóng Modal.

Các ràng buộc:

* Tuân thủ REQ-2 (Maudacta): Các trường Tên, Người thực hiện, Hạn hoàn thành là bắt buộc.
* Tuân thủ BR-3 (Maudacta): Trạng thái khởi tạo bắt buộc là "To Do".
* Tuân thủ REQ-7 (Maudacta): Phải gửi thông báo cho người được gán việc ngay sau khi tạo.
* Tuân thủ REQ-8 (Maudacta): Kiểm tra trùng tên công việc trong cùng một dự án (nếu cấu hình bật).

## Chức năng xem công việc

Mục đích: Cho phép người dùng theo dõi tổng quan danh sách các công việc trong dự án, hỗ trợ các thao tác tìm kiếm, lọc và xem chi tiết nội dung của từng nhiệm vụ. Chức năng này giúp các thành viên nắm bắt được khối lượng công việc và tiến độ hiện tại để điều chỉnh kế hoạch làm việc phù hợp.

Giao diện: Mô tả: Màn hình hiển thị dưới dạng Danh sách (List View) hoặc tích hợp trong Bảng Kanban. Phía trên danh sách là thanh công cụ bao gồm ô tìm kiếm và các bộ lọc (Dropdown) theo trạng thái, người thực hiện và độ ưu tiên. Khi người dùng nhấp vào một công việc, hệ thống sẽ mở ra khung chi tiết (Detail Drawer/Modal) trượt từ bên phải sang hoặc hiển thị đè lên trung tâm.

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Textbox | Rỗng | Tìm kiếm: Nhập tên công việc. Hỗ trợ tìm kiếm theo thời gian thực (debounce) |
| 2 | Combobox | "Tất cả" | Lọc theo Trạng thái: To Do, In Progress, Done |
| 3 | Combobox | "Tất cả" | Lọc theo Người thực hiện: Danh sách thành viên dự án. |
| 4 | Data Table | N/A | Danh sách công việc: Hiển thị các cột: Tên task, Người được giao, Hạn chót, Mức độ ưu tiên, Trạng thái. |
| 5 | Pagination | Trang 1 | Phân trang: Hiển thị số trang nếu danh sách quá dà |
| 6 | Label | N/A | Thông báo trống: Hiển thị "Không tìm thấy dữ liệu" nếu kết quả lọc rỗng. |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_TASKS |  |  |  | X (Lấy danh sách task) |
| 2 | TBL\_USERS |  |  |  | X (Join để lấy tên người thực hiện) |
| 3 | TaskDTO (JSON) |  |  |  | Chứa thông tin tổng hợp gửi về Client |

Cách xử lý:

1. Tại Client (Vue.js):
   * Khi người dùng truy cập màn hình hoặc thay đổi bộ lọc.
   * Client thu thập các tham số: keyword, status, assigneeId, page.
   * Gửi Request GET /api/projects/{id}/tasks kèm theo query parameters.
   * Nhận về danh sách JSON TaskDTO.
   * Render dữ liệu lên bảng (Table). Nếu danh sách rỗng, hiển thị thông báo theo REQ-6.
2. Tại Backend (Java Core):
   * Controller nhận tham số lọc.
   * Service xây dựng câu truy vấn động (Dynamic Query) xuống Database (sử dụng Criteria API hoặc MyBatis dynamic SQL).
   * Kiểm tra quyền hạn: Đảm bảo User chỉ xem được task trong dự án họ tham gia (REQ-7).
   * Trả về danh sách kết quả kèm thông tin phân trang (Pagination Metadata).
3. Xem chi tiết:
   * Khi người dùng click vào một dòng.
   * Hệ thống gọi API lấy chi tiết GET /api/tasks/{taskId} (hoặc dùng dữ liệu đã cache trong Store).
   * Hiển thị đầy đủ mô tả, lịch sử và bình luận.

Các ràng buộc:

* REQ-7 (Phân quyền): Hệ thống phải lọc dữ liệu phía Server dựa trên UserID hiện tại, không được trả về dữ liệu của dự án mà User không tham gia.
* PER-3 (Hiệu năng): Nếu danh sách vượt quá 500 bản ghi, Frontend cần áp dụng kỹ thuật "Virtual Scrolling" hoặc phân trang (Pagination) để đảm bảo tốc độ cuộn mượt mà 60fps.
* REQ-5 (Real-time): Nếu có người khác cập nhật công việc khi người dùng đang xem danh sách, hệ thống (qua WebSocket) nên tự động cập nhật lại dòng đó mà không cần F5.

## Chức năng bình luận công việc

Mục đích: Cung cấp kênh giao tiếp trực tiếp ngay trên từng công việc, cho phép các thành viên trao đổi ý kiến, báo cáo tiến độ, hỏi đáp và đính kèm thông tin bổ sung. Toàn bộ lịch sử trao đổi được lưu trữ tập trung, giúp nhóm dễ dàng theo dõi dòng sự kiện của nhiệm vụ.

Giao diện:Mô tả: Khu vực bình luận (Comment Section) được đặt ở phần dưới cùng hoặc cột bên phải của Cửa sổ chi tiết công việc (Task Detail Modal). Giao diện bao gồm danh sách các bình luận lịch sử (hiển thị Avatar, Tên người gửi, Thời gian, Nội dung) và một khung nhập liệu ở dưới cùng để soạn thảo tin nhắn mới.

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | List Container | N/A | Danh sách bình luận: Có thanh cuộn (Scrollbar) nếu danh sách dài. Sắp xếp theo thời gian (Cũ nhất trên, Mới nhất dưới). |
| 2 | Image (Avatar) | Default Icon | Ảnh đại diện: Hiển thị bên cạnh mỗi bình luận để nhận diện người nói. |
| 3 | Label | N/A | Metadata: Hiển thị "Tên người gửi" và "Thời gian gửi" (ví dụ: 10 phút trước |
| 4 | Textarea | Rỗng | Khung nhập liệu: Cho phép nhập nhiều dòng. Tự động giãn chiều cao khi nhập dài. |
| 5 | Button (Icon) | Send Icon | Nút Gửi: Chỉ kích hoạt (Enable) khi khung nhập liệu không rỗng |
| 6 | Button (Icon) | Trash Icon | Nút Xóa: Chỉ hiển thị trên các bình luận do chính người dùng đó tạo ra |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_COMMENTS | x |  | x | X (Lấy theo TaskID) |
| 2 | TBL\_USERS |  |  |  | X (Lấy tên/avatar tác giả) |
| 3 | CommentDTO (JSON) |  |  |  | Chứa nội dung, senderId, timestamp |
| 4 | TBL\_NOTIFICATIONS | x |  |  | Tạo thông báo cho người liên quan |

Cách xử lý:

1. Tải danh sách (Khi mở Task):
   * Client gọi API GET /api/tasks/{taskId}/comments.
   * Server trả về danh sách JSON.
   * Vue.js render danh sách ra giao diện.
2. Gửi bình luận (Client - Vue.js):
   * Người dùng nhập nội dung và nhấn Enter hoặc nút Gửi.
   * Validate: Kiểm tra nội dung có rỗng không (Client-side).
   * Client gửi Request POST /api/tasks/{taskId}/comments với nội dung text.
3. Xử lý tại Server (Backend):
   * Server nhận Request, xác định userId từ Token.
   * Lưu bình luận vào TBL\_COMMENTS với thời gian hiện tại (NOW()).
   * Trigger Event: Gửi tín hiệu đến NotificationService để tạo thông báo cho:
     + Người được gán việc (Assignee).
     + Những người đã tham gia bình luận trước đó trong Task này.
   * Trả về đối tượng Comment vừa tạo (bao gồm cả ID và Timestamp).
4. Phản hồi UI:
   * Client nhận dữ liệu trả về.
   * Thêm ngay (Append) bình luận mới vào cuối danh sách.
   * Tự động cuộn xuống dưới cùng (Scroll to bottom).
   * Xóa trắng ô nhập liệu.

Các ràng buộc:

* REQ-18: Hệ thống tuyệt đối không cho phép lưu bình luận rỗng hoặc chỉ chứa khoảng trắng.
* REQ-20 (Authorization): API xóa bình luận phải kiểm tra quyền sở hữu tại Server (comment.authorId == currentUserId). Nếu người dùng cố tình xóa bình luận của người khác, trả về lỗi 403.
* REQ-21: Dữ liệu bình luận phải được lưu trữ vĩnh viễn (trừ khi bị xóa chủ động), không bị mất đi khi Task chuyển trạng thái Done.

## Chức năng cập nhật trạng thái công việc

Mục đích: Cho phép người dùng (Người được gán việc hoặc Quản lý) thay đổi tiến độ của một nhiệm vụ cụ thể (các trạng thái: To Do, In Progress, Done, Cancel). Hành động này là cơ sở để hệ thống tính toán phần trăm hoàn thành dự án và kích hoạt các luồng thông báo tự động cho các thành viên liên quan.

Giao diện: Mô tả: Chức năng này được thực hiện chủ yếu trên Bảng Kanban thông qua thao tác Kéo & Thả (Drag & Drop) trực quan. Ngoài ra, trong màn hình Chi tiết công việc (Modal), người dùng cũng có thể thay đổi trạng thái thông qua một Dropdown Menu (Combobox) nằm ở góc trên.

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Kanban Column | To Do, In Progress, Done | Vùng thả (Drop Zone): Nhận sự kiện khi thả chuột. |
| 2 | Draggable Card | N/A | Thẻ công việc: Cho phép dùng chuột kéo di chuyển. |
| 3 | Combobox | Trạng thái hiện tại | Chọn trạng thái: Chỉ hiển thị trong màn hình Chi tiết (Detail Modal). |
| 4 | Toast Message | Ẩn | Thông báo: Hiển thị "Cập nhật thành công" hoặc Lỗi sau khi thả. |
| 5 | Lock Icon | Ẩn | Biểu tượng khóa: Hiển thị khi Task ở trạng thái "Done" (nếu cấu hình khóa). |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_TASKS |  | X (Update Status) |  | X (Kiểm tra quyền) |
| 2 | TBL\_ACTIVITY\_LOG | x |  |  | Ghi log thay đổi |
| 3 | TBL\_NOTIFICATIONS | x |  |  | Gửi thông báo |

Cách xử lý:

1. Thao tác Kéo & Thả (Kanban Interaction):
   * Người dùng nhấn giữ chuột vào một Task Card ở cột A.
   * Kéo sang cột B và thả chuột.
   * Client (Vue.js):
     + Ngay lập tức cập nhật giao diện (Optimistic UI) để Task nằm ở cột mới (tạo cảm giác mượt mà).
     + Gửi Request ngầm PUT /api/tasks/{id}/status với body { "status": "NEW\_STATUS" }.
2. Xử lý tại Server (Backend):
   * Kiểm tra quyền (BR-3): User hiện tại có phải là Assignee hoặc Manager của dự án không? (Nếu không -> Trả lỗi 403).
   * Kiểm tra logic: Nếu Task đang ở trạng thái DONE và đã bị khóa, chỉ Manager mới được phép mở lại (Re-open).
   * Cập nhật cột status trong TBL\_TASKS.
   * Ghi Log: Lưu vào TBL\_ACTIVITY\_LOG ("User A moved Task B from To Do to In Progress").
   * Async Notification: Bắn sự kiện để thông báo cho những người theo dõi Task.
3. Xử lý lỗi (Rollback UI):
   * Nếu Server trả về lỗi (ví dụ: Mất mạng, Không đủ quyền):
   * Client hiển thị thông báo lỗi.
   * Tự động di chuyển Task Card trở lại cột cũ (Revert).

Các ràng buộc:

* REQ-UP-1: Phải hỗ trợ thao tác Kéo & Thả mượt mà (HTML5 Drag and Drop API).
* REQ-UP-2 & BR-3: Chỉ người làm (Assignee) hoặc Quản lý (Manager) mới đổi được trạng thái. Member khác chỉ được xem.
* REQ-UP-3: Khi trạng thái là "Done", các trường dữ liệu khác (Tên, Mô tả) sẽ bị khóa (Read-only), trừ khi chuyển lại trạng thái khác.

## Chức năng phân quyền dự án

Mục đích: Cho phép Quản lý dự án (Manager) hoặc Chủ dự án (Owner) thiết lập và điều chỉnh vai trò của các thành viên trong phạm vi một dự án cụ thể. Việc này nhằm kiểm soát quyền truy cập, thao tác dữ liệu và đảm bảo tính bảo mật dựa trên mô hình phân quyền theo vai trò (RBAC).

Giao diện: Mô tả: Chức năng này thường nằm trong tab "Thành viên" (Members) của màn hình Cài đặt Dự án. Giao diện hiển thị danh sách các thành viên hiện tại cùng với vai trò của họ (hiển thị dưới dạng Dropdown để dễ dàng thay đổi). Phía trên có ô tìm kiếm thành viên để lọc nhanh.

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Textbox | Rỗng | Tìm kiếm thành viên: Nhập tên hoặc email. |
| 2 | Table List | N/A | Danh sách thành viên: Hiển thị Avatar, Tên, Email và Cột Vai trò. |
| 3 | Combobox (trên từng dòng) | Vai trò hiện tại | Chọn vai trò: MEMBER (Thành viên), MANAGER (Quản lý). |
| 4 | Button (Icon) | Trash Icon | Xóa thành viên: Loại bỏ thành viên khỏi dự án. |
| 5 | Button | N/A | Nút "Mời thành viên": Mở Modal để thêm người mới vào dự án. |
| 6 | Modal (Confirm) | N/A | Hộp thoại xác nhận: Xuất hiện khi hạ quyền hoặc xóa thành viên. |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_USER\_PROJECT | X (Mời) | X (Đổi Role) | X (Kick) | X (Lấy DS) |
| 2 | TBL\_USERS |  |  |  | X (Lấy info hiển thị) |
| 3 | TBL\_ACTIVITY\_LOG | x |  |  | Ghi log thay đổi quyền |

Cách xử lý:

1. Tại Client (Vue.js):
   * Manager truy cập tab "Thành viên".
   * Client gọi API GET /api/projects/{id}/members.
   * Manager thay đổi giá trị trong Combobox Vai trò của một thành viên (ví dụ: từ Member lên Manager).
   * Hệ thống hiển thị Modal xác nhận: "Bạn có chắc muốn cấp quyền Quản lý cho thành viên này?".
   * Khi xác nhận, Client gửi Request PUT /api/projects/{id}/members/{userId} với body { "role": "MANAGER" }.
2. Tại Backend (Java Core):
   * Kiểm tra quyền (Security Check): Người thực hiện có phải là Owner hoặc Manager của dự án này không? (Nếu không -> 403) .
   * Kiểm tra logic an toàn (Safety Check):
     + Nếu người thực hiện đang tự hạ quyền của chính mình -> Kiểm tra xem dự án còn Manager nào khác không? Nếu không -> Chặn (tránh dự án không ai quản lý).
   * Cập nhật: Thay đổi role trong bảng liên kết TBL\_USER\_PROJECT.
   * Ghi Log: Lưu vào TBL\_ACTIVITY\_LOG ("User A changed role of User B to MANAGER").
   * Trả về kết quả thành công.
3. Phản hồi UI:
   * Hiển thị thông báo "Cập nhật quyền thành công".
   * Cập nhật lại trạng thái trên giao diện mà không cần reload trang.

Các ràng buộc:

* REQ-AUTH-2: Manager chỉ được phân quyền trong dự án mình quản lý.
* REQ-AUTH-5: Hệ thống phải ngăn chặn việc người quản lý tự tước quyền quản trị của chính mình nếu dự án đó chỉ còn một Manager.
* REQ-AUTH-4: Mọi thay đổi về nhân sự phải được ghi log đầy đủ (Ai làm, làm cho ai, thời gian)

## Chức năng xem thông kê

Mục đích: Cung cấp cho người dùng (đặc biệt là Quản lý dự án) cái nhìn trực quan về hiệu suất làm việc và tiến độ dự án thông qua các biểu đồ (hình tròn, hình cột). Đây là tính năng đặc biệt được thiết kế dưới dạng Plugin, cho phép hệ thống nạp thêm các loại báo cáo mới (như Biểu đồ Burn-down, Biểu đồ Gantt) mà không cần sửa đổi mã nguồn lõi.

Giao diện: Mô tả: Giao diện là một Dashboard tổng hợp (hoặc tab "Báo cáo" trong chi tiết dự án). Khu vực chính hiển thị các Widget biểu đồ được render từ các thư viện đồ họa (như Chart.js/ECharts). Phía trên có thanh công cụ cho phép chọn loại báo cáo (tương ứng với các Plugin thống kê đã cài) và bộ lọc thời gian.

Các thành phần trong giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại điều khiển | Giá trị mặc định | Lưu ý |
| 1 | Combobox | "Tiến độ công việc" | Chọn loại báo cáo: Danh sách này được tải động từ API GET /api/plugins. |
| 2 | DateRange Picker | Tuần này | Khoảng thời gian: Từ ngày - Đến ngày. |
| 3 | Button | N/A | Nút "Xem báo cáo": Gửi yêu cầu tính toán. |
| 4 | Chart Canvas | N/A | Khu vực vẽ biểu đồ: Nơi Vue.js render dữ liệu JSON thành hình ảnh. |
| 5 | Table | N/A | Bảng dữ liệu chi tiết: Hiển thị số liệu thô bên dưới biểu đồ. |
| 6 | Button (Icon) | Export Icon | Xuất dữ liệu: Tải về file Excel/PDF (nếu Plugin hỗ trợ). |

Dữ liệu được sử dụng:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên bảng / Cấu trúc dữ liệu | Phương thức | | | |
| Thêm | Sửa | Xóa | Truy vấn |
| 1 | TBL\_TASKS |  |  |  | X (Plugin truy vấn qua HostContext) |
| 2 | TBL\_PLUGINS |  |  |  | X (Lấy danh sách Plugin thống kê) |
| 3 | ChartDataDTO (JSON) |  |  |  | Cấu trúc dữ liệu trả về cho Frontend vẽ (Labels, Datasets) |

Cách xử lý:

1. Khởi tạo (Client):
   * Khi vào màn hình Thống kê, Vue.js gọi API GET /api/extensions/stats để lấy danh sách các Plugin thống kê đang hoạt động (ví dụ: "Task Distribution", "Member Performance").
   * Hiển thị danh sách này vào Combobox.
2. Thực thi (Client -> Server):
   * Người dùng chọn "Biểu đồ trạng thái công việc" và nhấn "Xem".
   * Client gọi API: GET /api/extensions/execute/stats-plugin-01?projectId=101.
3. Xử lý tại Backend (Core & Plugin):
   * Plugin Manager: Tìm kiếm Plugin có ID stats-plugin-01.
   * Isolation: Gọi phương thức execute(context, params) của Plugin trong một khối try-catch.
   * Plugin Logic:
     + Plugin sử dụng IHostContext để lấy danh sách Task từ Database.
     + Thực hiện thuật toán tổng hợp (ví dụ: Đếm số task theo trạng thái DONE/TODO).
     + Đóng gói kết quả về dạng JSON chuẩn: { "labels": ["Done", "Todo"], "data": [10, 5] }.
   * Backend trả JSON về cho Client.
4. Hiển thị (Client):
   * Vue.js nhận JSON.
   * Sử dụng thư viện Chart (như vue-chartjs) để vẽ biểu đồ tương ứng.

Các ràng buộc:

* REQ-STAT-1: Plugin phải tuân thủ đúng Interface, trả về dữ liệu JSON theo cấu trúc mà Frontend có thể hiểu được.
* REQ-STAT-4 (Security): Plugin không được tự ý kết nối DB riêng, bắt buộc phải dùng dữ liệu do hostContext.getProjectService() cung cấp.
* PER-1: Với các báo cáo phức tạp tốn > 3s xử lý, Backend phải xử lý bất đồng bộ và Client cần hiển thị Loading Spinner để người dùng không tưởng máy bị treo.

# Bảng tham khảo tới các yêu cầu

<Sử dụng theo định dạng bảng để chỉ ra thành phần nào của hệ thống đáp ứng yêu cầu chức năng nào trong tài liệu đặc tả yêu cầu phần mềm. Tham chiếu tới các yêu cầu chức năng thông qua mã số mà ta đã gán cho chúng trong tài liệu đặc tả.>

# Các phụ lục

<Tùy chọn. Các phụ lục cung cấp thêm thông tin chi tiết hỗ trợ cho việc hiểu tài liệu thiết kế phần mềm.>