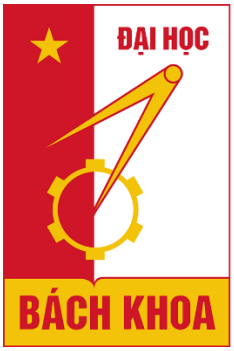
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI



SOICT

BÁO CÁO PROJECT-I

TaskTracker - Website quản lý công việc

Sinh viên thực hiện: Nguyễn Quang Đông 20225284

Giáo viên hướng dẫn: ThS. Lê Văn Đồng

Lời mở đầu

Báo cáo này trình bày chi tiết quá trình phân tích, thiết kế và các quyết định kỹ thuật đằng sau việc xây dựng hệ thống quản lý công việc "Task Tracker". Mục đích của tài liệu là cung cấp một cái nhìn tổng quan, toàn diện về kiến trúc hệ thống, từ yêu cầu ban đầu, lựa chọn công nghệ, cho đến thiết kế cơ sở dữ liệu và API. Báo cáo này sẽ phục vụ như một tài liệu tham khảo cốt lõi cho các nhà phát triển và các bên liên quan trong quá trình phát triển, bảo trì và mở rộng hệ thống trong tương lai, đảm bảo tính nhất quán và khả năng kế thừa của dự án.

MỤC LỤC

Chương 1. Đặt vấn đề

Trong bối cảnh phát triển phần mềm và quản lý dự án hiện đại, nhu cầu về các công cụ hỗ trợ làm việc nhóm hiệu quả ngày càng trở nên cấp thiết. Các đội nhóm thường xuyên đối mặt với thách thức trong việc theo dõi tiến độ, phân công nhiệm vụ một cách minh bạch và đảm bảo mọi thành viên đều nắm rõ trách nhiệm cũng như thời hạn công việc. Sự ra đời của các công cụ quản lý dự án đã giải quyết phần nào bài toán này, tuy nhiên, nhiều giải pháp lại quá phức tạp, cồng kềnh, gây khó khăn cho việc áp dụng trong các nhóm nhỏ và vừa. Nhận thấy nhu cầu về một giải pháp tinh gọn, tập trung vào sự cộng tác và trải nghiệm người dùng, dự án "Task Tracker" đã được hình thành.

1.1. Bối cảnh và Lý do chọn đề tài

Quản lý dự án truyền thống thường gặp phải các vấn đề như thông tin phân mảnh, khó khăn trong việc theo dõi tiến độ thực tế so với kế hoạch, và thiếu một kênh giao tiếp tập trung cho từng đầu việc. Các công cụ hiện có đôi khi đi kèm với một đường cong học tập dốc và nhiều tính năng không cần thiết, làm giảm hiệu suất thay vì tăng cường nó.

"Task Tracker" được phát triển để giải quyết trực tiếp những thách thức này. Hệ thống hướng đến việc cung cấp một nền tảng tập trung, nơi mọi thông tin về dự án, công việc và thành viên được quản lý một cách khoa học. Bằng cách ưu tiên một giao diện người dùng tối giản và các luồng công việc trực quan, "Task Tracker" mong muốn trở thành một công cụ làm việc hàng ngày hiệu quả, dễ tiếp cận và không gây xao lãng cho người dùng.

1.2. Mục tiêu của hệ thống

Hệ thống được xây dựng với các mục tiêu chính sau đây:

- **Xây dựng ứng dụng web quản lý tiến độ dự án:** Cung cấp một nền tảng tập trung để các nhóm có thể tạo dự án, phân chia công việc, và theo dõi tiến độ một cách trực quan. Điều này giúp tăng cường sự minh bạch và nâng cao hiệu suất làm việc chung của cả nhóm.
- **Tối ưu hóa trải nghiệm người dùng (UX):** Tập trung vào việc xây dựng một giao diện người dùng sạch sẽ, tối giản và dễ sử dụng. Một trải nghiệm người dùng tốt là yếu tố then chốt để người dùng gắn bó và sử dụng công cụ một cách thường xuyên, biến nó thành một phần tự nhiên trong quy trình làm việc.
- **Xây dựng hệ thống phân quyền rõ ràng:** Thiết lập một cơ cấu vai trò chặt chẽ (Owner, Leader, Member) để đảm bảo an ninh dữ liệu và trật tự trong quản lý. Việc phân quyền rõ ràng giúp giới hạn các thao tác nhạy cảm, tránh các thay đổi không mong muốn và đảm bảo mỗi thành viên chỉ có thể truy cập các chức năng phù hợp với vai trò của mình.
- **Tăng cường khả năng làm việc cộng tác:** Hỗ trợ các tính năng cốt lõi cho việc cộng tác như giao việc cho thành viên cụ thể, cập nhật trạng thái công việc theo thời gian thực, và các hệ thống thông báo tự động.

1.3. Đối tượng sử dụng và Phạm vi dự án

Hệ thống được thiết kế để phục vụ ba vai trò người dùng chính, mỗi vai trò có các trách nhiệm và quyền hạn riêng biệt.

Vai trò	Mô tả và trách nhiệm chính
OWNER	Người tạo ra dự án và có toàn quyền kiểm soát. Chịu trách nhiệm thiết lập dự án, quản lý thành viên (mời, xóa, phân quyền), và có thể xóa dự án.
LEADER	Được Owner chỉ định, chịu trách nhiệm quản lý các hoạt động hàng ngày của dự án. Có quyền tạo, sửa, xóa và giao việc cho các thành viên.
MEMBER	Là người trực tiếp thực thi các công việc. Có quyền xem các công việc (tùy theo cài đặt của Owner) và cập nhật trạng thái của các công việc được giao.

Phạm vi dự án:

- **Các chức năng trong phạm vi:**
  - Xác thực người dùng qua Google OAuth.
  - Quản lý dự án: Tạo, xem, cập nhật, xóa, và cài đặt dự án.
  - Quản lý công việc (Task): Tạo, sửa, xóa, giao việc, đính kèm tệp, cập nhật trạng thái và độ ưu tiên.
  - Quản lý thành viên: Mời, phân quyền, xóa thành viên khỏi dự án.
  - Hệ thống thông báo: Tự động gửi email nhắc nhở về hạn chót công việc.
- **Các chức năng ngoài phạm vi (hướng phát triển):**
  - Quản lý tài chính, ngân sách dự án.
  - Chức năng chấm công, theo dõi thời gian làm việc (timesheet).
  - Tích hợp với các kho mã nguồn (Git).
  - Xuất các báo cáo thống kê phức tạp.

1.4. Phương pháp tiếp cận và Đóng góp của đề tài

Dự án được tiếp cận theo mô hình thác nước. Việc lựa chọn stack công nghệ MERN (MongoDB, ExpressJS, ReactJS, Node.js) cũng phản ánh cách tiếp cận hiện đại, tận dụng sức mạnh của JavaScript trên cả frontend và backend.

Những đóng góp chính của đề tài bao gồm:

1. **Cung cấp giải pháp mã nguồn mở:** Tạo ra một công cụ quản lý công việc miễn phí, dễ dàng triển khai cho các nhóm nhỏ, startups hoặc các tổ chức phi lợi nhuận.
2. **Ứng dụng stack công nghệ hiện đại:** Trình bày một ví dụ thực tế về việc xây dựng một ứng dụng web đầy đủ chức năng bằng MERN stack, một trong những bộ công nghệ phổ biến nhất hiện nay.
3. **Tích hợp dịch vụ đám mây:** Tối ưu hóa chức năng và giảm tải cho hệ thống bằng cách tích hợp các dịch vụ bên thứ ba uy tín như Google OAuth cho xác thực, Cloudinary cho lưu trữ tệp và Gmail STMP cho việc gửi email.

1.5. Cấu trúc báo cáo

Báo cáo này được tổ chức thành 5 chương chính:

- **Chương 1 - Đặt vấn đề:** Giới thiệu tổng quan về bối cảnh, mục tiêu, phạm vi và những đóng góp của dự án.
- **Chương 2 - Cơ sở lý thuyết và Công nghệ sử dụng:** Phân tích nền tảng lý thuyết và các công nghệ được lựa chọn để xây dựng hệ thống.
- **Chương 3 - Phân tích và Thiết kế hệ thống:** Trình bày chi tiết bản thiết kế kỹ thuật, bao gồm yêu cầu, các biểu đồ UML, thiết kế cơ sở dữ liệu và kiến trúc hệ thống.
- **Chương 4 - Triển khai và Thử nghiệm:** Mô tả môi trường triển khai, các kịch bản kiểm thử và đánh giá chất lượng hệ thống.
- **Chương 5 - Tài liệu tham khảo:** Liệt kê các nguồn tài liệu đã được tham khảo trong quá trình thực hiện dự án.

Phần tiếp theo của báo cáo sẽ đi sâu vào cơ sở lý thuyết và các công nghệ đã được sử dụng.

Chương 2. Cơ sở lý thuyết và Công nghệ sử dụng

Việc lựa chọn một nền tảng công nghệ phù hợp là yếu tố nền tảng, quyết định trực tiếp đến sự thành công, hiệu suất và khả năng mở rộng của một dự án phần mềm. Chương này sẽ trình bày các khái niệm lý thuyết cốt lõi và phân tích chi tiết các công nghệ đã được lựa chọn để xây dựng hệ thống "Task Tracker", tạo nên một nền tảng vững chắc cho việc phát triển.

2.1. Cơ sở lý thuyết

- **Mô hình RESTful API:** Hệ thống giao tiếp giữa frontend (client) và backend (server) được xây dựng dựa trên kiến trúc REST (Representational State Transfer). RESTful API tuân thủ các nguyên tắc cốt lõi như kiến trúc Client-Server, giao tiếp không trạng thái (Stateless), và sử dụng các phương thức HTTP tiêu chuẩn (GET, POST, PATCH, DELETE) để thực hiện các thao tác trên tài nguyên. Cách tiếp cận này giúp tạo ra một API rõ ràng, dễ hiểu và dễ dàng tích hợp với nhiều loại client khác nhau.
- **Cơ sở dữ liệu NoSQL (Document-Oriented):** Dự án sử dụng MongoDB, một hệ quản trị cơ sở dữ liệu NoSQL hướng tài liệu. Khác với mô hình quan hệ (SQL) có cấu trúc cứng nhắc, mô hình hướng tài liệu cho phép lưu trữ dữ liệu dưới dạng các tài liệu BSON (tương tự JSON), mang lại sự linh hoạt cao trong việc định nghĩa cấu trúc dữ liệu. Điều này đặc biệt phù hợp với các ứng dụng web hiện đại, nơi yêu cầu về dữ liệu có thể thay đổi và phát triển nhanh chóng.
- **Xác thực với OAuth 2.0:** Để đơn giản hóa và tăng cường bảo mật cho quá trình đăng nhập, hệ thống tích hợp "Đăng nhập bằng Google" thông qua giao thức OAuth 2.0. OAuth 2.0 cho phép ứng dụng "Task Tracker" nhận được quyền truy cập giới hạn vào thông tin người dùng từ Google mà không cần phải xử lý hoặc lưu trữ mật khẩu của họ. Luồng hoạt động này cung cấp một phương thức đăng nhập an toàn, tiện lợi và được tin cậy rộng rãi.

2.2. Tổng quan về Stack công nghệ

Bảng dưới đây tóm tắt các công nghệ chính được sử dụng trong dự án và vai trò của chúng.

Thành phần	Công nghệ	Vai trò trong dự án
Frontend	ReactJS	Xây dựng giao diện người dùng (UI) dạng Single Page Application (SPA) linh hoạt và tương tác cao.
Backend	Node.js, ExpressJS	Xây dựng máy chủ ứng dụng và hệ thống RESTful API để xử lý logic nghiệp vụ.
Database	MongoDB	Lưu trữ và quản lý toàn bộ dữ liệu của ứng dụng một cách linh hoạt và hiệu quả.
Dịch vụ bên thứ ba	Google OAuth	Cung cấp tính năng đăng nhập/đăng ký an toàn và tiện lợi.
	Cloudinary	Dịch vụ lưu trữ và quản lý các tệp đính kèm (ảnh, tài liệu) trên đám mây.
	Gmail STMP	Dịch vụ gửi email tự động cho các chức năng như mời thành viên, nhắc nhở deadline.

2.3. Phân tích các công nghệ chính

- **ReactJS:** Là một thư viện JavaScript phổ biến để xây dựng giao diện người dùng, ReactJS được chọn cho phía frontend nhờ vào kiến trúc dựa trên component. Kiến trúc này cho phép chia nhỏ giao diện phức tạp thành các thành phần độc lập, có thể tái sử dụng, giúp mã nguồn trở nên dễ quản lý và bảo trì. Hơn nữa, cơ chế Virtual DOM của React giúp tối ưu hóa hiệu suất hiển thị, mang lại trải nghiệm mượt mà cho người dùng.
- **Node.js & ExpressJS:** Node.js là một môi trường thực thi JavaScript phía máy chủ, nổi bật với mô hình non-blocking I/O, rất phù hợp cho việc xây dựng các ứng dụng có nhiều tác vụ vào/ra như API server. ExpressJS, một framework tối giản và linh hoạt xây dựng trên nền tảng Node.js, được sử dụng để tăng tốc quá trình phát triển API, cung cấp các công cụ mạnh mẽ để định tuyến (routing), quản lý middleware và xử lý các yêu cầu HTTP.
- **MongoDB:** Sự lựa chọn MongoDB được quyết định bởi tính linh hoạt của mô hình dữ liệu. Cấu trúc dữ liệu của các dự án và công việc có thể thay đổi, và MongoDB cho phép dễ dàng điều chỉnh schema mà không làm gián đoạn hệ thống. Khả năng nhúng tài liệu (embedding) cũng giúp tối ưu hóa các truy vấn đọc, chẳng hạn như lấy thông tin dự án cùng với danh sách thành viên trong một lần truy vấn duy nhất.

2.4. Các thư viện và chuẩn liên quan

Dựa trên kiến trúc và chức năng của hệ thống, có thể suy luận một cách hợp lý về việc sử dụng các chuẩn và thư viện sau:

- **JSON Web Token (JWT):** Sau khi người dùng xác thực thành công qua Google OAuth, hệ thống backend có khả năng sẽ tạo ra một JWT để quản lý phiên đăng nhập của người dùng. Token này sẽ được gửi kèm trong mỗi yêu cầu API từ client để xác thực và phân quyền, đảm bảo tính bảo mật và không trạng thái của máy chủ.
- **Mongoose (ODM for MongoDB):** Để tương tác với MongoDB một cách có cấu trúc và an toàn từ môi trường Node.js, việc sử dụng một thư viện Object Data Modeling (ODM) như Mongoose là một giả định hợp lý. Mongoose cho phép định nghĩa các lược đồ (schema) cho dữ liệu, thực hiện xác thực (validation) và quản lý các mối quan hệ giữa các collection một cách dễ dàng.

Sự kết hợp của các công nghệ trên tạo ra một nền tảng vững chắc, hiện đại và có khả năng mở rộng, sẵn sàng cho giai đoạn phân tích và thiết kế chi tiết sẽ được trình bày ở chương tiếp theo.

Chương 3. Phân tích và Thiết kế hệ thống

Đây là chương quan trọng nhất, nơi các yêu cầu nghiệp vụ và ý tưởng ban đầu được chuyển đổi thành một bản thiết kế kỹ thuật chi tiết. Chất lượng của giai đoạn phân tích và thiết kế sẽ ảnh hưởng trực tiếp đến sự ổn định, khả năng bảo trì và thành công của sản phẩm cuối cùng. Chương này sẽ trình bày các yêu cầu hệ thống, mô hình hóa thông qua các biểu đồ UML, và thiết kế chi tiết kiến trúc cũng như cơ sở dữ liệu.

3.1. Yêu cầu hệ thống

3.1.1. Yêu cầu chức năng (Functional Requirements)

Dưới đây là danh sách các yêu cầu chức năng được hệ thống hóa từ tài liệu phân tích.

- **Nhóm chức năng Xác thực (F_AUTH)**
  - **F_AUTH-01:** Người dùng đăng nhập bằng tài khoản Google.
  - **F_AUTH-02:** Người dùng đăng xuất
- **Nhóm chức năng Quản lý Dự án (F_PROJ)**
  - **F_PROJ-01:** Người dùng có quyền phải có thể tạo một dự án mới với tên và mô tả.
  - **F_PROJ-02:** Người dùng xem danh sách tất cả các dự án mà họ là thành viên.
  - **F_PROJ-03:** Người dùng xem chi tiết dự án mà họ là thành viên.
  - **F_PROJ-04:** Owner phải có thể cập nhật thông tin cơ bản của dự án.
  - **F_PROJ-05:** Owner phải có thể cấu hình cài đặt dự án,
    - Quyền xem task của Member.
    - Bật/tắt tính năng nhắc hẹn.
  - **F_PROJ-06:** Owner phải có thể xóa dự án, với điều kiện phải nhập lại chính xác tên dự án để xác nhận.
- **Nhóm chức năng Quản lý Công việc (F_TASK)**
  - **F_TASK-01:** Leader/Owner thể tạo công việc.
  - **F_TASK-02:** Leader/Owner/Assignee cập nhật thông tin công việc.
    - Leader/Owner toàn quyền.
    - Assignee chỉ có thể thay đổi trạng thái.
  - **F_TASK-03:** Xem danh sách công việc.
    - Leader/Owner xem toàn bộ danh sách công việc.
    - Assignee xem danh sách công việc phụ trách.
    - Member xem toàn bộ danh sách công việc (phụ thuộc cấu hình dự án).
  - **F_TASK-04:** Leader/Owner/Assignee đính kèm tệp vào công việc.
  - **F_TASK-05:** Leader/Owner/Assignee xóa đính kèm tệp vào công việc.
  - **F_TASK-06**: Leader/Owner/Assignee xem chi tiết công việc.
  - **F_TASK-07**: Leader/Owner xóa công việc.
  - **F_TASK-08:** Hệ thống tự động gửi email nhắc nhở vào lúc 07:00 hàng ngày cho các công việc sắp đến hạn trong vòng 24 giờ hoặc quá hạn.
- **Nhóm chức năng Quản lý Thành viên (F_MEMBER)**
  - **F_MEMBER-01:** Owner mời thành viên mới vào dự án thông qua email.

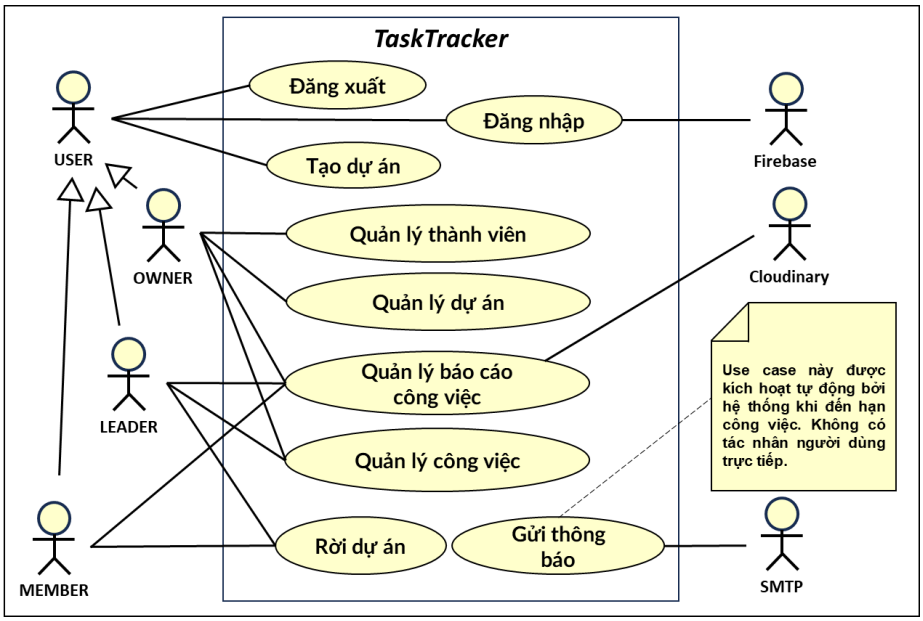
- **F_MEMBER-02:** Owner thay đổi vai trò của thành viên
  - Thăng cấp lên Leader.
  - Giáng cấp xuống Member.
- **F_MEMBER-03:** Owner xóa một thành viên khỏi dự án.
- **F_MEMBER-04:** Leader/Member tự rời khỏi dự án.
- **F_MEMBER-05**: Người dùng tham gia dự án qua email.

3.1.2. Yêu cầu phi chức năng (Non-Functional Requirements)

- **Hiệu năng (Performance):** Thời gian phản hồi trung bình của các yêu cầu API phải dưới 500ms trong điều kiện tải thông thường. Giao diện người dùng phải được tải và sẵn sàng tương tác trong vòng 3 giây.
- **Bảo mật (Security):** Toàn bộ dữ liệu truyền tải giữa client và server phải được mã hóa bằng giao thức HTTPS. Các mã token mời tham gia dự án phải có thời gian hết hạn để tránh lạm dụng.
- **Tính khả dụng (Availability):** Hệ thống phải đảm bảo độ sẵn sàng 99.5%, giảm thiểu thời gian chết không có kế hoạch.
- **Tính dễ sử dụng (Usability):** Giao diện phải được thiết kế trực quan, cho phép người dùng mới có thể thực hiện các thao tác cơ bản (tạo dự án, tạo task) mà không cần xem tài liệu hướng dẫn.

3.2. Biểu đồ Use Case

- **Các Actors chính:**
  - **Owner:** Chủ dự án.
  - **Leader:** Trưởng nhóm.
  - **Member:** Thành viên.
  - **STMP:** Tác nhân hệ thống, thực hiện các tác vụ tự động (Cron Job).
  - **Firebase:** Tác nhân hệ thống, hỗ trợ đăng nhập bằng Gmail.
  - **Cloudinary:** Tác nhân ngoài hệ thống, dùng để lưu trữ file báo cáo



Hình 3.2. Biểu đồ Use-case mức tổng quan

Bảng đặc tả Use-case

Use Case ID	UC-01
Tên Use Case	Đăng nhập bằng Gmail (Google Sign-In)
Mô tả	Cho phép người dùng đăng nhập vào hệ thống TaskTracker bằng tài khoản Google (Gmail)
Tác nhân chính	User
Tác nhân phụ	Google Sign-In (Google Identity Provider), Firebase Authentication
Tiền điều kiện	Người dùng đang ở màn hình đăng nhập.
Hậu điều kiện	Thành công: Điều hướng vào Dashboard
Luồng chính	1. User mở màn hình đăng nhập. 2. User chọn "Đăng nhập bằng Google". 3. Hệ thống mở luồng Google Sign-In để User chọn tài khoản và cấp quyền. 4. User chọn tài khoản Google và xác nhận. 5. Google trả về token đăng nhập (ID Token/Authorization Code). 6. Hệ thống gửi token lên Firebase Authentication để xác thực. 7. Firebase xác thực thành công và trả về thông tin user (UID, email, displayName, photoURL…). 8. Hệ thống kiểm tra hồ sơ user trong CSDL: nếu chưa có thì tạo mới, nếu có thì cập nhật (ví dụ lastLogin, avatar). 9. Hệ thống tạo phiên đăng nhập/lưu trạng thái đăng nhập. 10. Hệ thống chuyển User đến Dashboard/Trang chính.
Luồng thay thế	A1 - User huỷ đăng nhập: ➔ Hệ thống thông báo "Đăng nhập đã bị huỷ" và quay lại màn hình đăng nhập. A2 - Token không hợp lệ/hết hạn: ➔ Hệ thống thông báo "Phiên đăng nhập không hợp lệ, vui lòng thử lại" và quay lại bước 2.
Quy tắc nghiệp vụ	Chỉ cho phép đăng nhập qua Google (Gmail) theo cơ chế Google Sign-In.
Ghi chú	Use-case tổng quan

Bảng 3.1.1 Đặc tả Use-case "Đăng nhập"

Use Case ID	UC-02
Tên Use Case	Đăng xuất
Mô tả	Người dùng đăng xuất hệ thống
Tác nhân chính	USER
Tiền điều kiện	Người dùng đã đăng nhập
Hậu điều kiện	Đăng xuất thành công khỏi hệ thống
Luồng chính	1. Người dùng bấm tài khoản 2. Người dùng chọn Đăng xuất 3. Màn hình quay về trang đăng nhập
Luồng thay thế	A2. Lỗi ➔ Thông báo
Quy tắc nghiệp vụ	Chỉ khi đã đăng nhập

Bảng 3.1.2 Đặc tả Use-case "Đăng xuất"

Use Case ID	UC-03
Tên Use Case	Tạo dự án
Mô tả	Cho phép người dùng tạo một dự án mới để quản lý công việc
Tác nhân chính	USER
Tác nhân phụ	Không
Tiền điều kiện	USER đã đăng nhập hệ thống
Hậu điều kiện	Dự án được tạo thành công; USER trở thành OWNER của dự án
Luồng chính	1. USER chọn chức năng *Tạo dự án* 2. Hệ thống hiển thị form thông tin dự án 3. USER nhập thông tin dự án 4. USER xác nhận tạo 5. Hệ thống kiểm tra hợp lệ 6. Hệ thống tạo dự án và gán quyền OWNER
Luồng thay thế	A1. Thông tin không hợp lệ ➔ hệ thống báo lỗi A2. USER hủy thao tác ➔ không tạo dự án
Quy tắc nghiệp vụ	Tên dự án không được rỗng
Ghi chú	Use case tổng quát

Bảng 3.1.3 Đặc tả Use-case "Tạo dự án"

Use Case ID	UC-04
Tên Use Case	Quản lý thành viên
Mô tả	OWNER quản lý thành viên trong dự án
Tác nhân chính	OWNER
Tác nhân phụ	STMP (nếu có gửi thông báo)
Tiền điều kiện	OWNER đã đăng nhập và có quyền trên dự án
Hậu điều kiện	Danh sách thành viên/vai trò được cập nhật
Luồng chính	1. OWNER chọn *Quản lý thành viên* 2. Hệ thống hiển thị danh sách thành viên 3. OWNER thêm/mời thành viên 4. Hệ thống kiểm tra điều kiện 5. Hệ thống cập nhật danh sách
Luồng thay thế	A1. Thành viên không tồn tại ➔ báo lỗi A2. Thành viên đã tham gia ➔ thông báo
Quy tắc nghiệp vụ	OWNER không thể tự xóa chính mình
Ghi chú	Bao gồm thêm, xóa, phân quyền thành viên

Bảng 3.1.4 Đặc tả Use-case "Quản lý thành viên"

Use Case ID	UC-05
Tên Use Case	Quản lý dự án
Mô tả	OWNER cập nhật thông tin và cấu hình dự án
Tác nhân chính	OWNER
Tác nhân phụ	STMP (nếu cần thông báo)
Tiền điều kiện	OWNER đăng nhập và là chủ dự án
Hậu điều kiện	Thông tin dự án được cập nhật
Luồng chính	1. OWNER chọn *Quản lý dự án* 2. Hệ thống hiển thị thông tin dự án 3. OWNER chỉnh sửa thông tin 4. OWNER lưu thay đổi
Luồng thay thế	A1. Dữ liệu không hợp lệ ➔ không lưu A2. OWNER hủy thao tác ➔ Không lưu
Quy tắc nghiệp vụ	Chỉ OWNER mới được thay đổi cấu hình dự án
Ghi chú	Use case tổng quát

Bảng 3.1.5 Đặc tả Use-case "Quản lý dự án"

Use Case ID	UC-06
Tên Use Case	Quản lý công việc
Mô tả	OWNER, LEADER quản lý công việc trong dự án
Tác nhân chính	OWNER, LEADER
Tác nhân phụ	Không
Tiền điều kiện	LEADER thuộc dự án
Hậu điều kiện	Công việc được tạo/cập nhật/gán
Luồng chính	1. OWNER , LEADER chọn *Quản lý công việc* 2. Hệ thống hiển thị danh sách công việc 3. OWNER , LEADER tạo hoặc chỉnh sửa công việc 4. Hệ thống kiểm tra hợp lệ 5. Hệ thống lưu thay đổi
Luồng thay thế	A1. Deadline không hợp lệ ➔ báo lỗi A2. Người được gán không hợp lệ ➔ báo lỗi
Quy tắc nghiệp vụ	Chỉ gán công việc cho thành viên dự án
Ghi chú	Bao gồm tạo, sửa, gán, đổi trạng thái

Bảng 3.1.6 Đặc tả Use-case "Quản lý công việc"

Use Case ID	UC-07
Tên Use Case	Quản lý báo cáo công việc
Mô tả	OWNER, LEADER, MEMBER (được giao việc) theo dõi và tổng hợp báo cáo công việc
Tác nhân chính	OWNER, LEADER, MEMBER (được giao việc)
Tác nhân phụ	Không
Tiền điều kiện	LEADER, MEMBER thuộc dự án
Hậu điều kiện	Báo cáo được xem hoặc tổng hợp
Luồng chính	1. LEADER chọn *Xem chi tiết công việc* 2. Hệ thống hiển thị báo cáo 3. OWNER, LEADER, MEMBER (được giao việc) thêm/xem/sửa/xóa báo cáo
Luồng thay thế	A1. Không có báo cáo ➔ hiển thị rỗng
Quy tắc nghiệp vụ	Chỉ xem báo cáo trong dự án mình quản lý
Ghi chú	Use case tổng quát

Bảng 3.1.7 Đặc tả Use-case "Quản lý báo cáo công việc"

Use Case ID	UC-08
Tên Use Case	Rời dự án
Mô tả	LEADER, MEMBER tự rời khỏi dự án
Tác nhân chính	LEADER, MEMBER
Tác nhân phụ	Không
Tiền điều kiện	LEADER, MEMBER đang tham gia dự án
Hậu điều kiện	LEADER, MEMBER bị loại khỏi dự án
Luồng chính	1. LEADER, MEMBER chọn *Rời dự án* 2. Hệ thống yêu cầu xác nhận 3. LEADER, MEMBER xác nhận 4. Hệ thống cập nhật dữ liệu
Luồng thay thế	A1. LEADER, MEMBER xác nhận sai ➔ từ chối
Quy tắc nghiệp vụ	Công việc không được xóa khi thành viên rời
Ghi chú	Không áp dụng cho OWNER

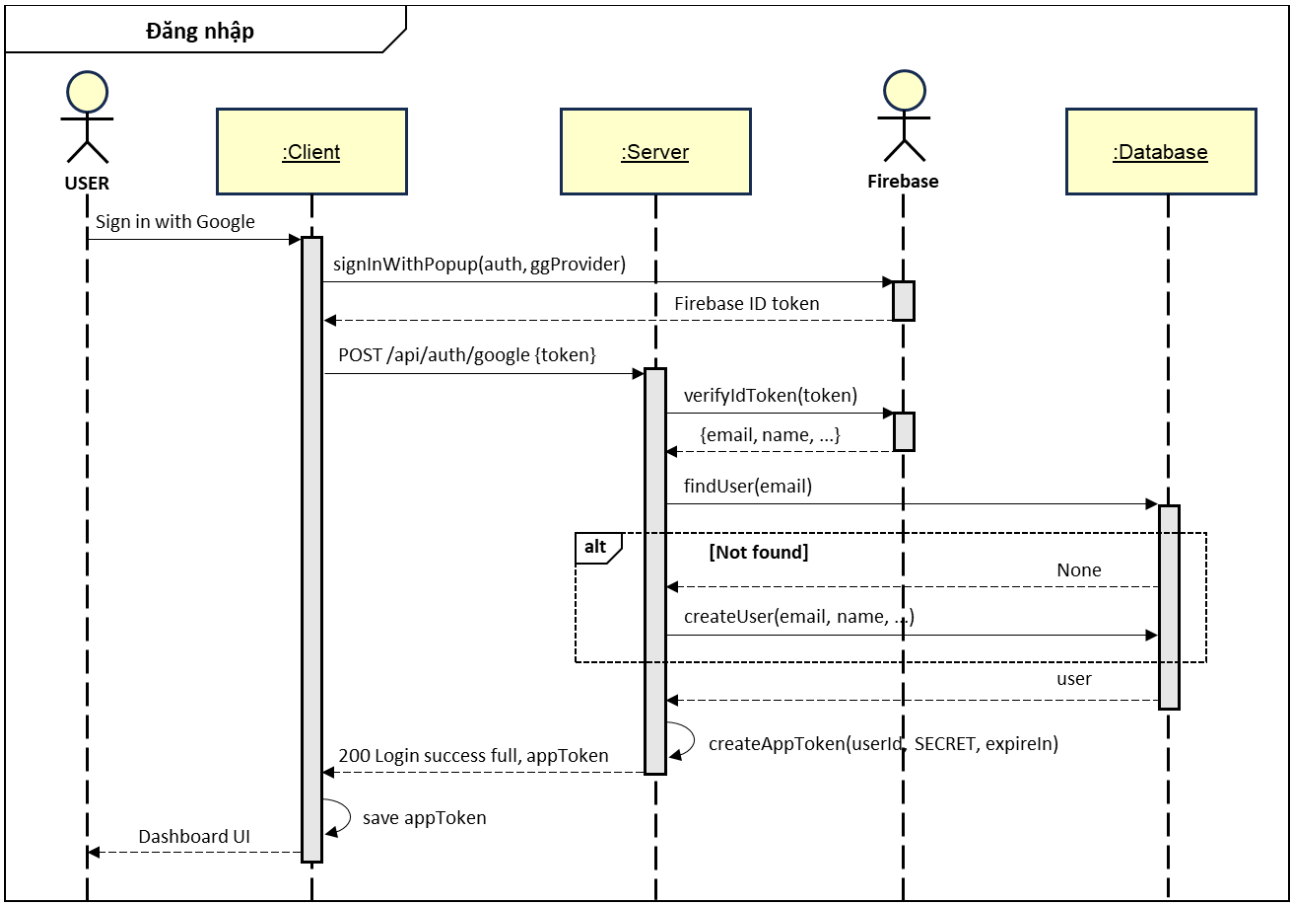
Bảng 3.1.8 Đặc tả Use-case "Rời dự án"

Use Case ID	UC-09
Tên Use Case	Gửi thông báo
Mô tả	Hệ thống tự động gửi thông báo khi đến hạn công việc
Tác nhân chính	Không có (kích hoạt nội bộ)
Tác nhân phụ	STMP
Tiền điều kiện	Có công việc đến hạn; nhắc hẹn được bật
Hậu điều kiện	Thông báo được gửi hoặc được ghi log
Luồng chính	1. Đến thời điểm nhắc hẹn 2. Hệ thống xác định công việc cần gửi 3. Tạo nội dung thông báo 4. Gửi qua STMP 5. Lưu trạng thái
Luồng thay thế	A1. STMP lỗi ➔ ghi log A2. Không có công việc cần nhắc
Quy tắc nghiệp vụ	Chỉ gửi cho thành viên còn trong dự án

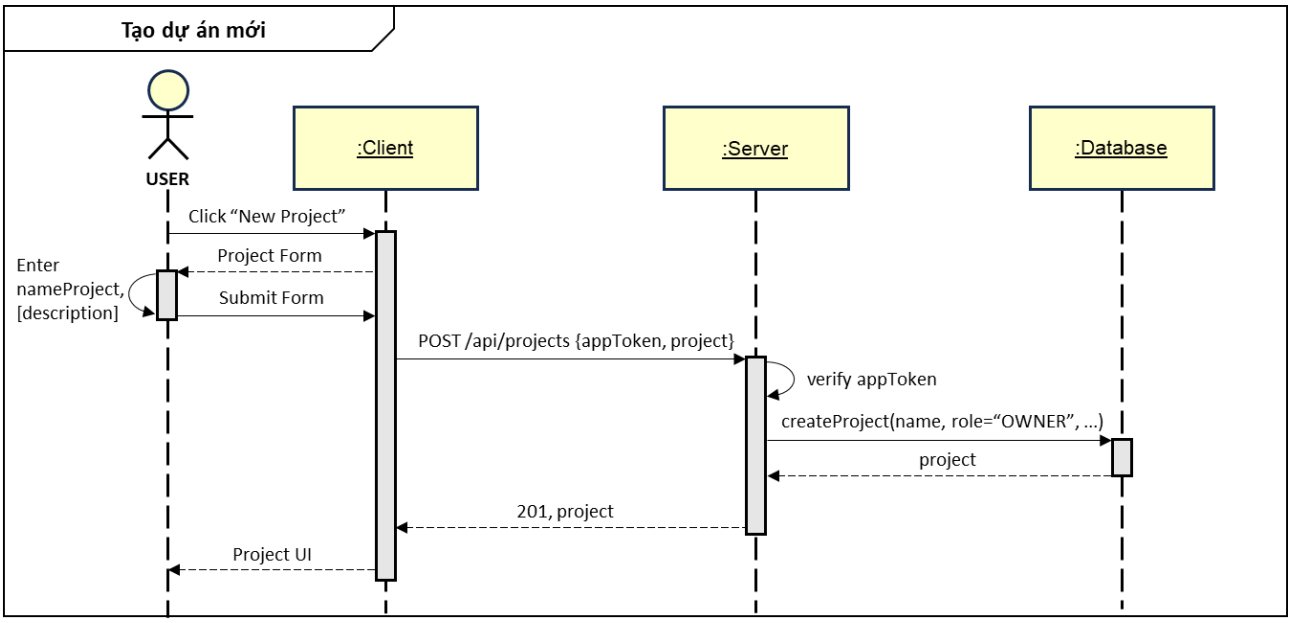
Bảng 3.1.9 Đặc tả Use-case "Gửi thông báo"

3.3. Biểu đồ tuần tự (Sequence Diagram) theo chức năng

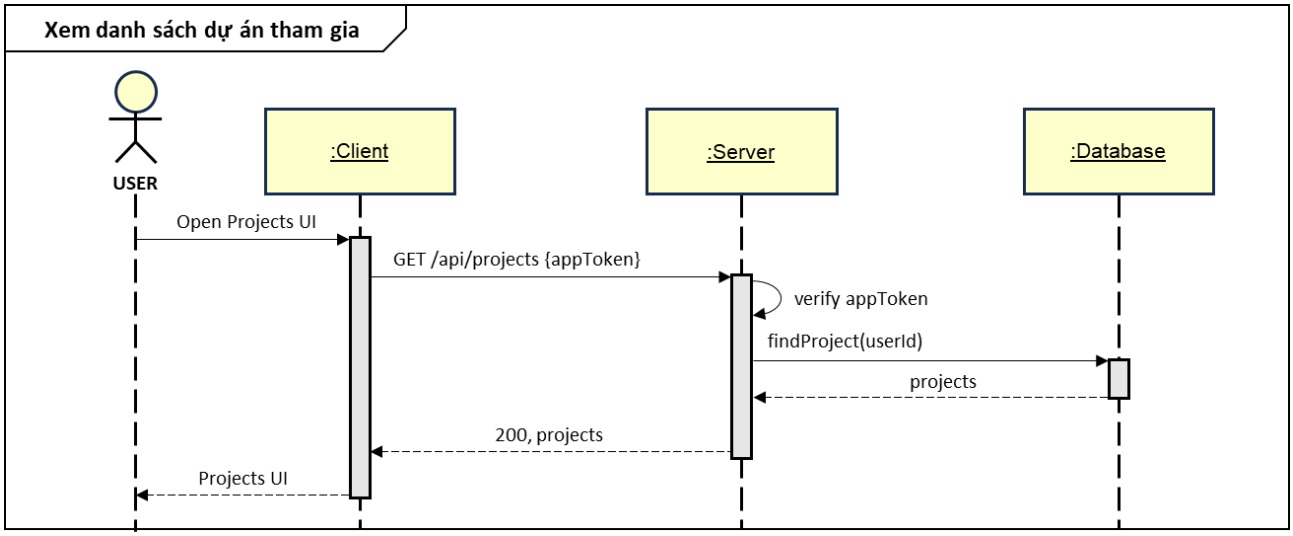
1. Đăng nhập



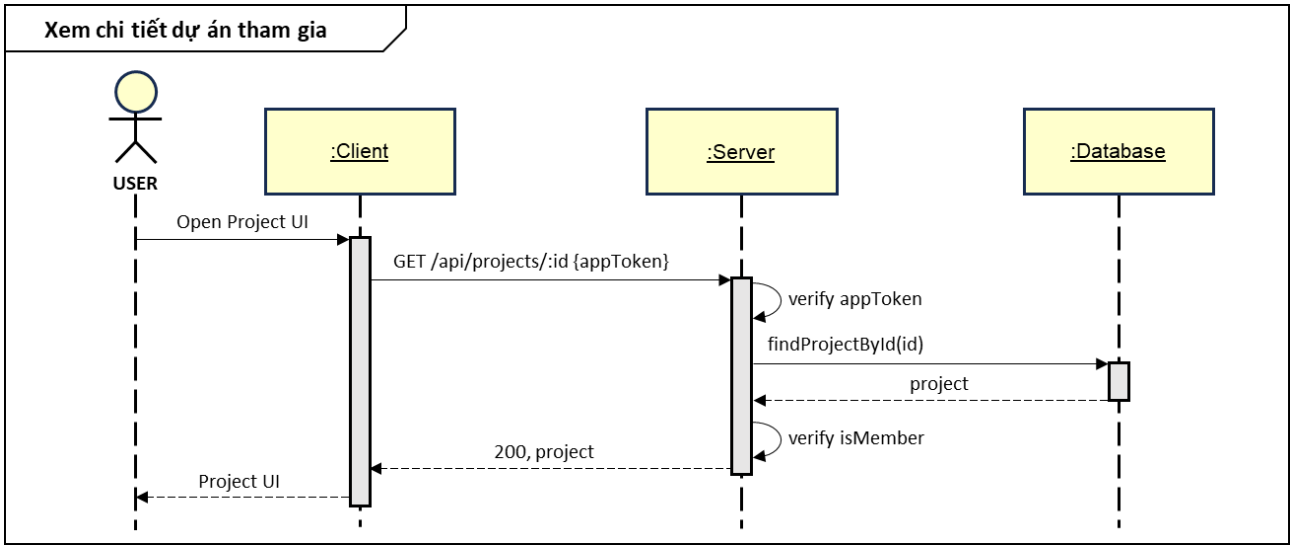
2. Tạo dự án mới



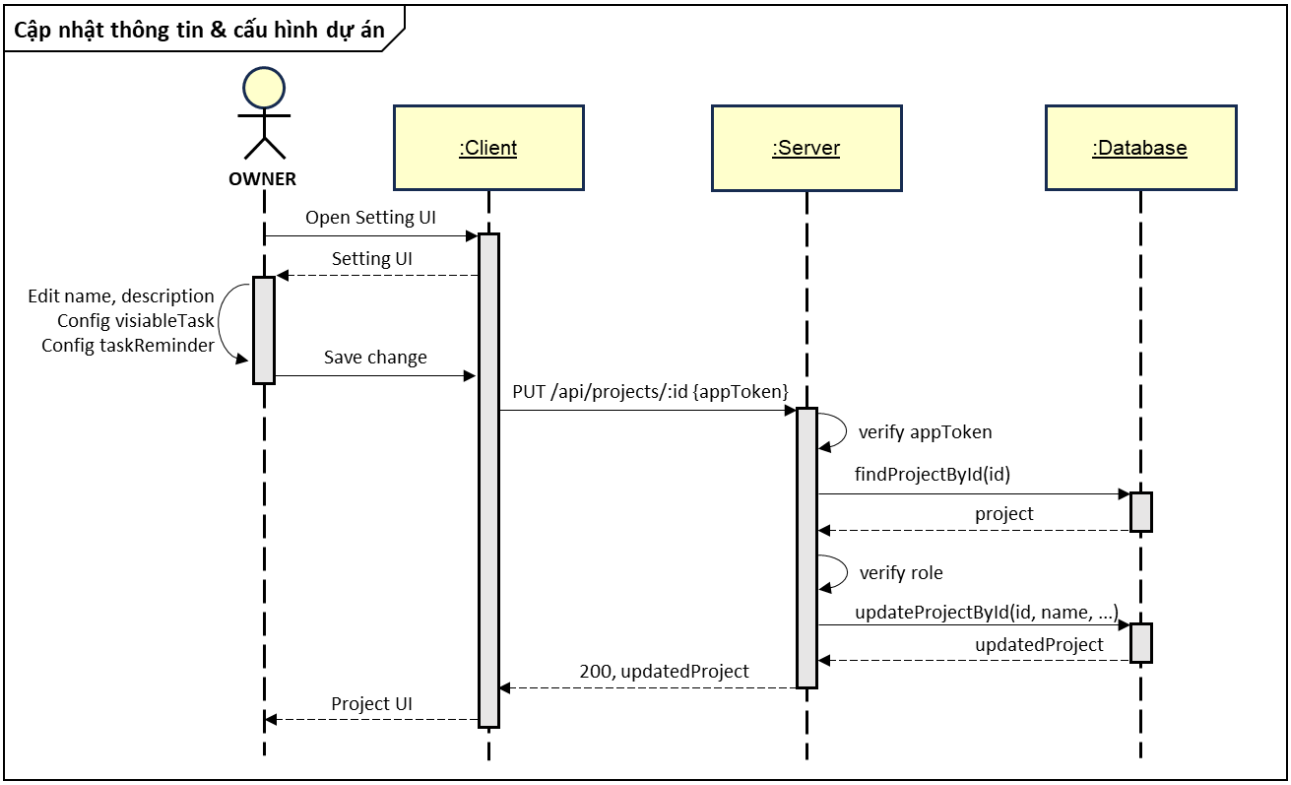
3. Xem danh sách dự án tham gia



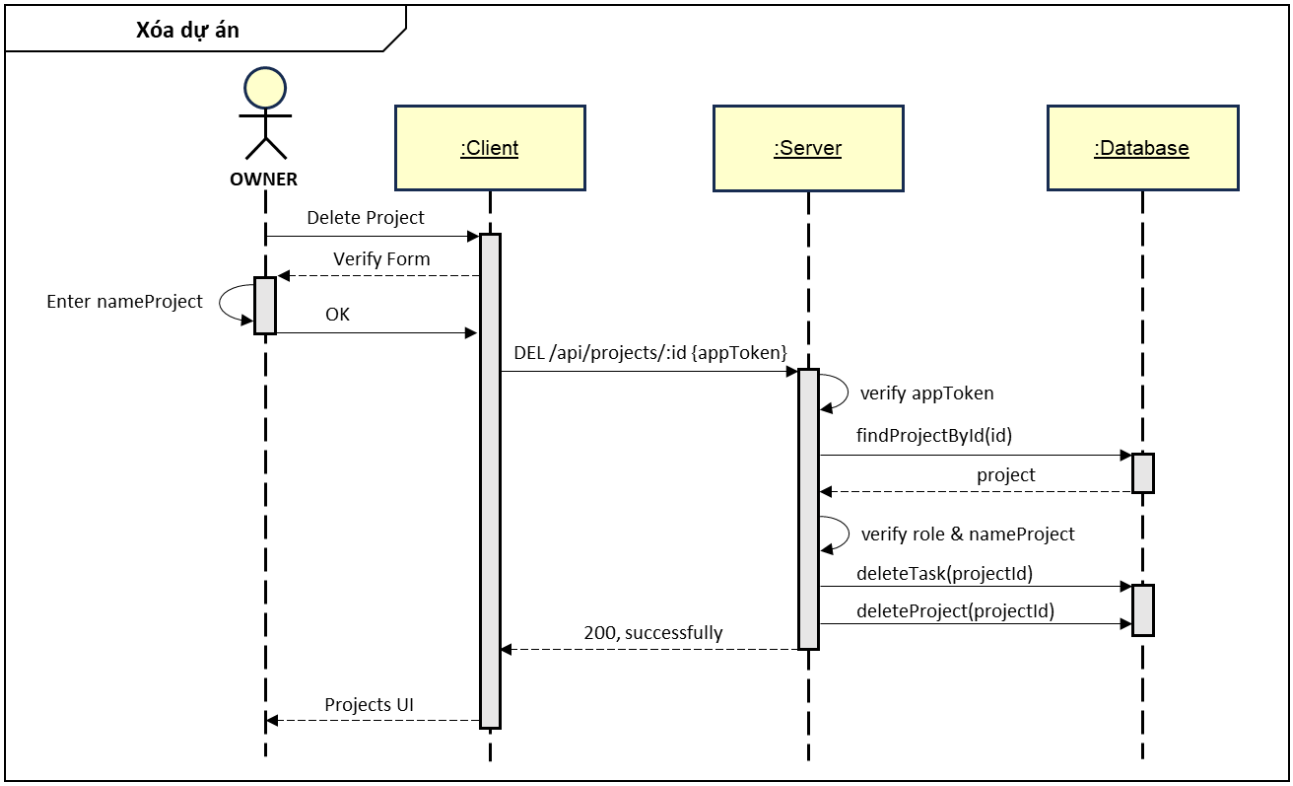
4. Xem chi tiết dự án tham gia



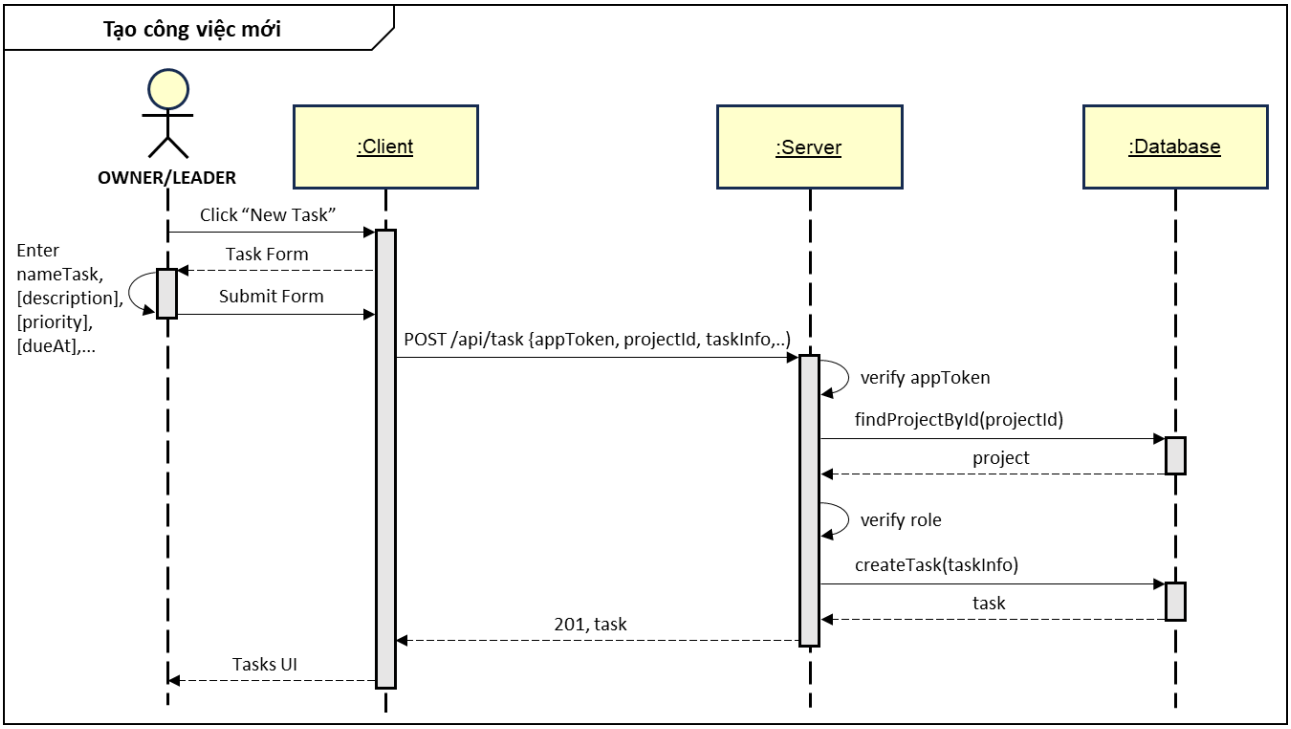
5. Cập nhật thông tin & cấu hình dự án



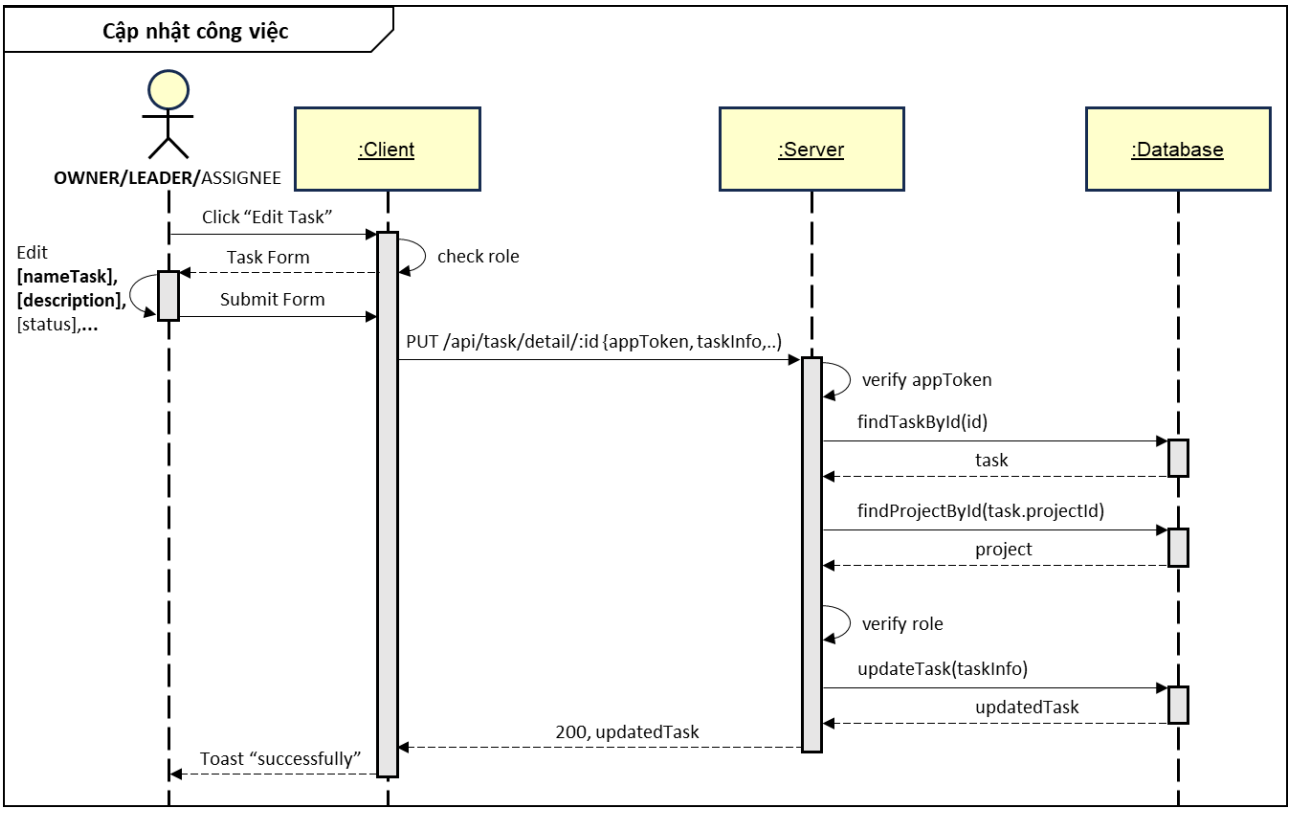
6. Xóa dự án



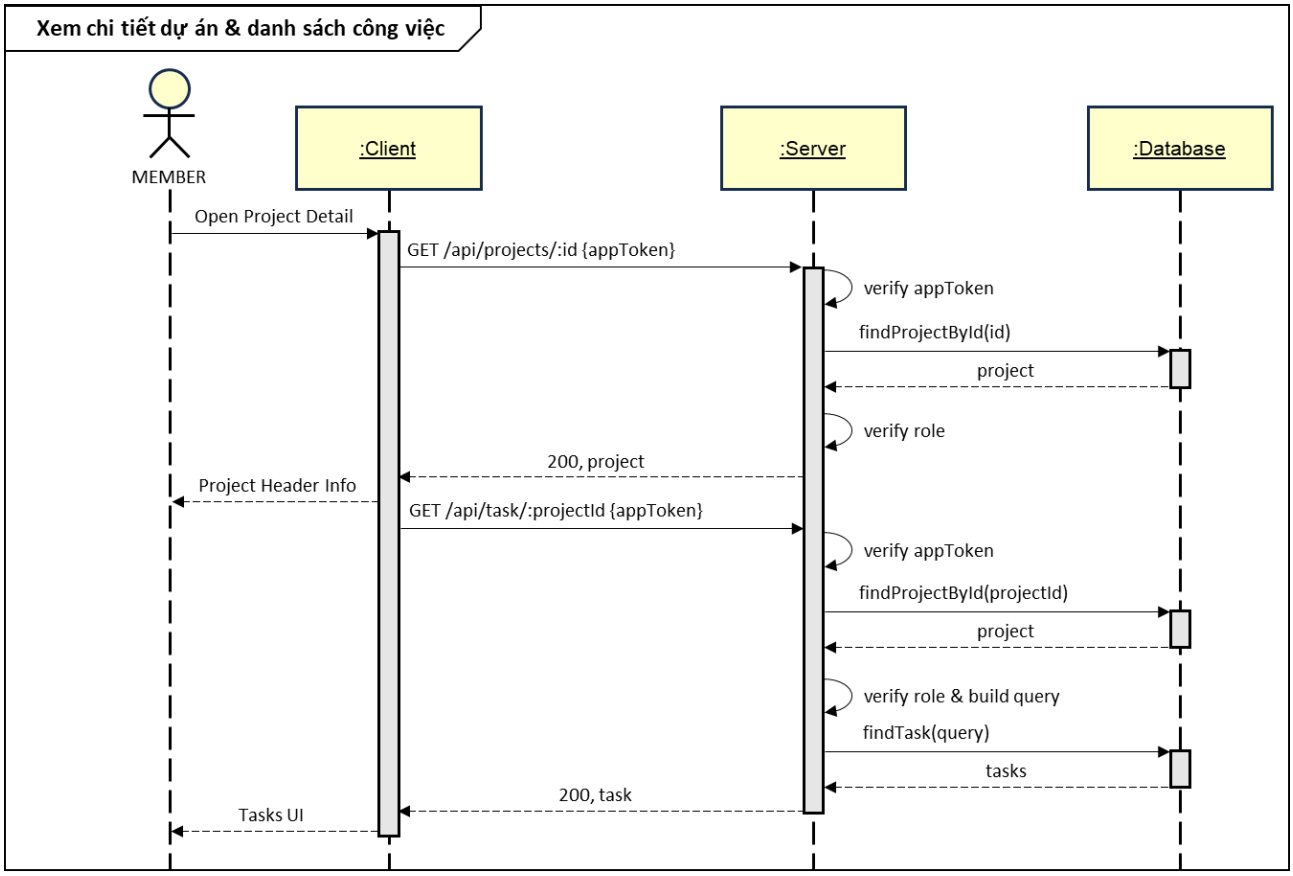
7. Tạo công việc



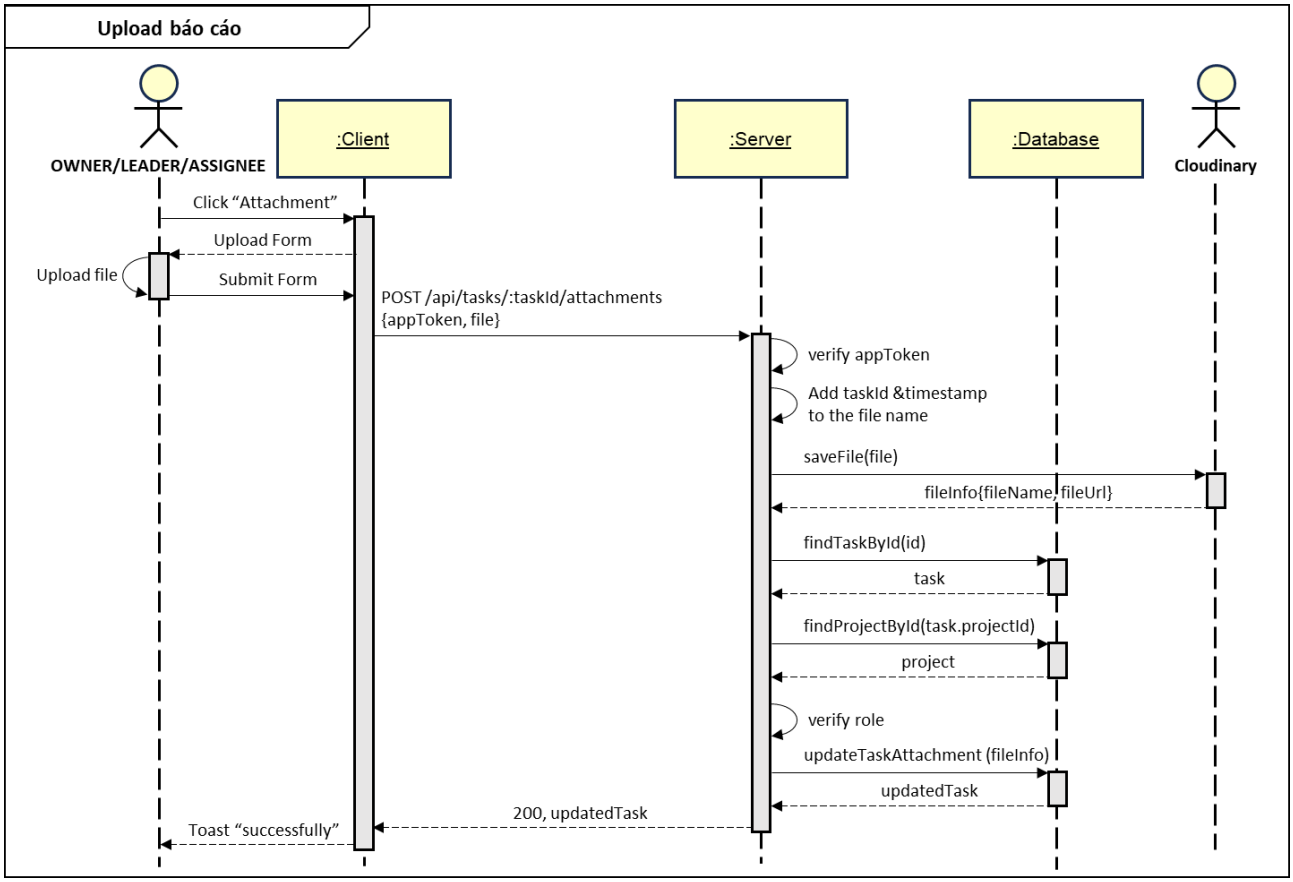
8. Cập nhật công việc



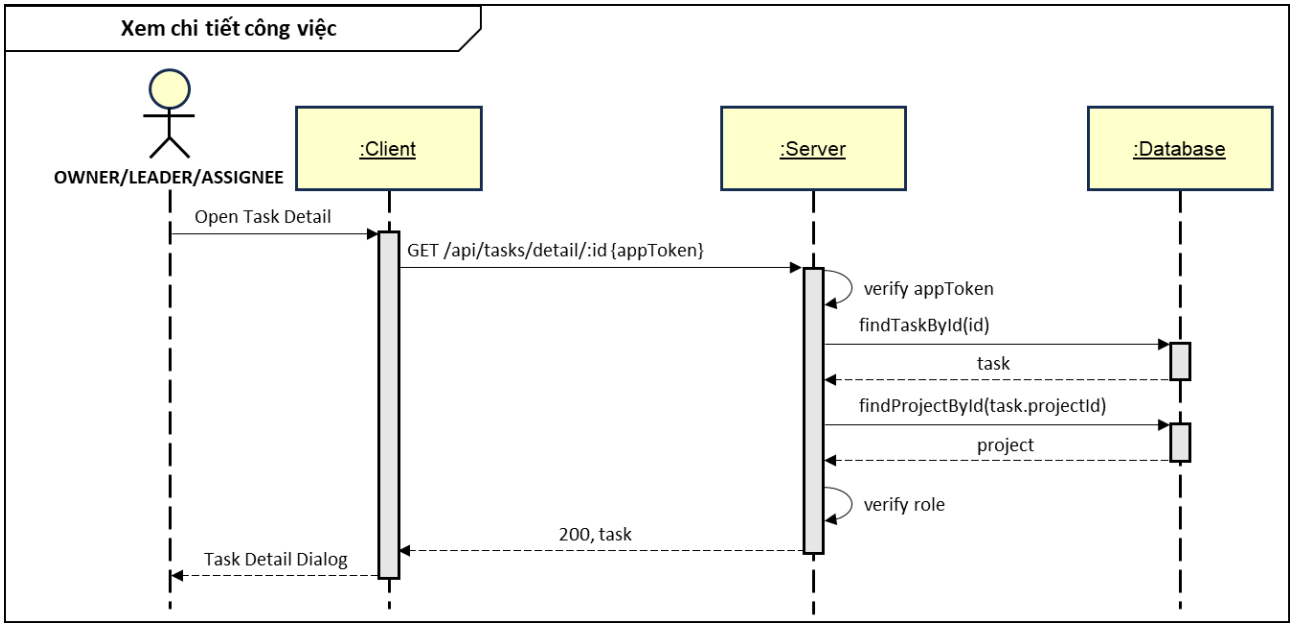
9. Xem chi tiết dự án & danh sách công việc



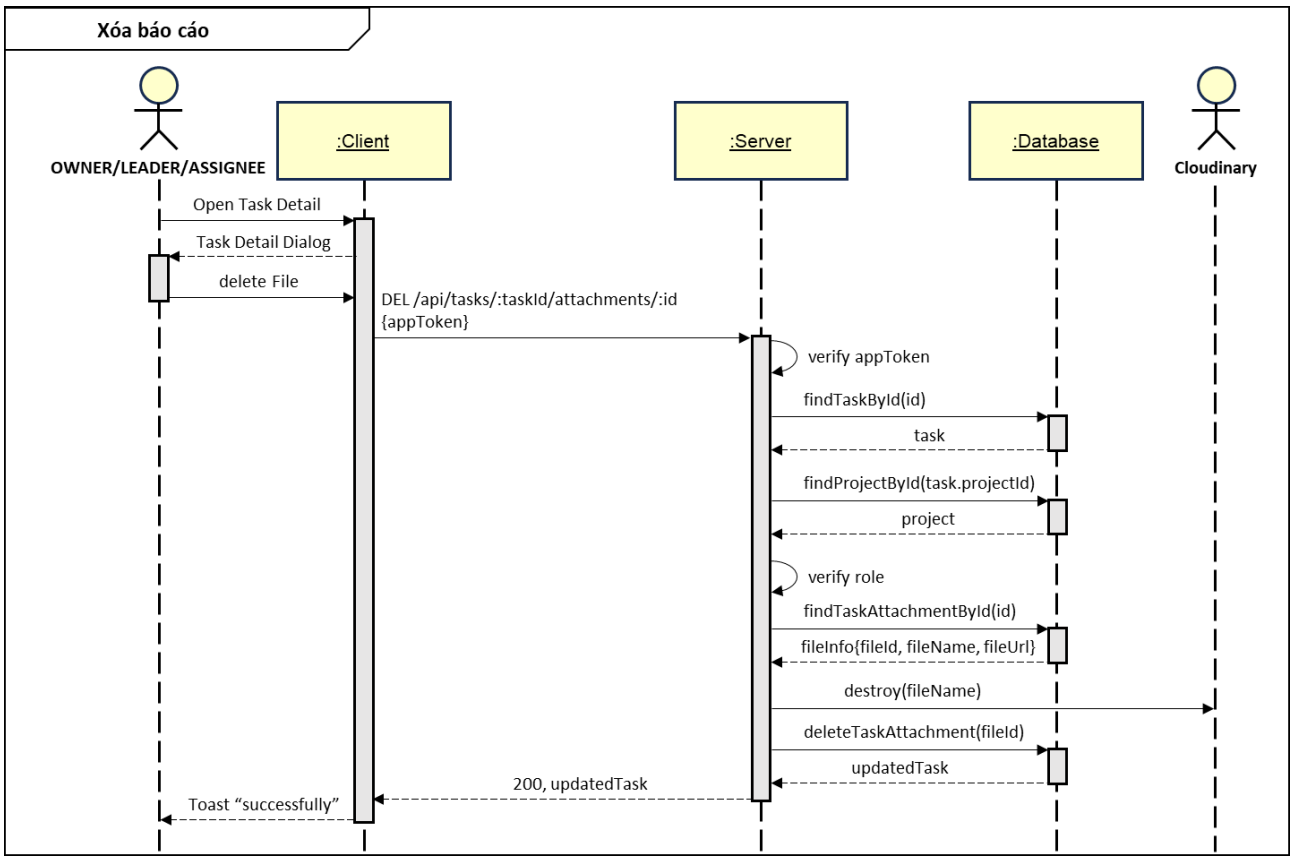
10. Upload báo cáo



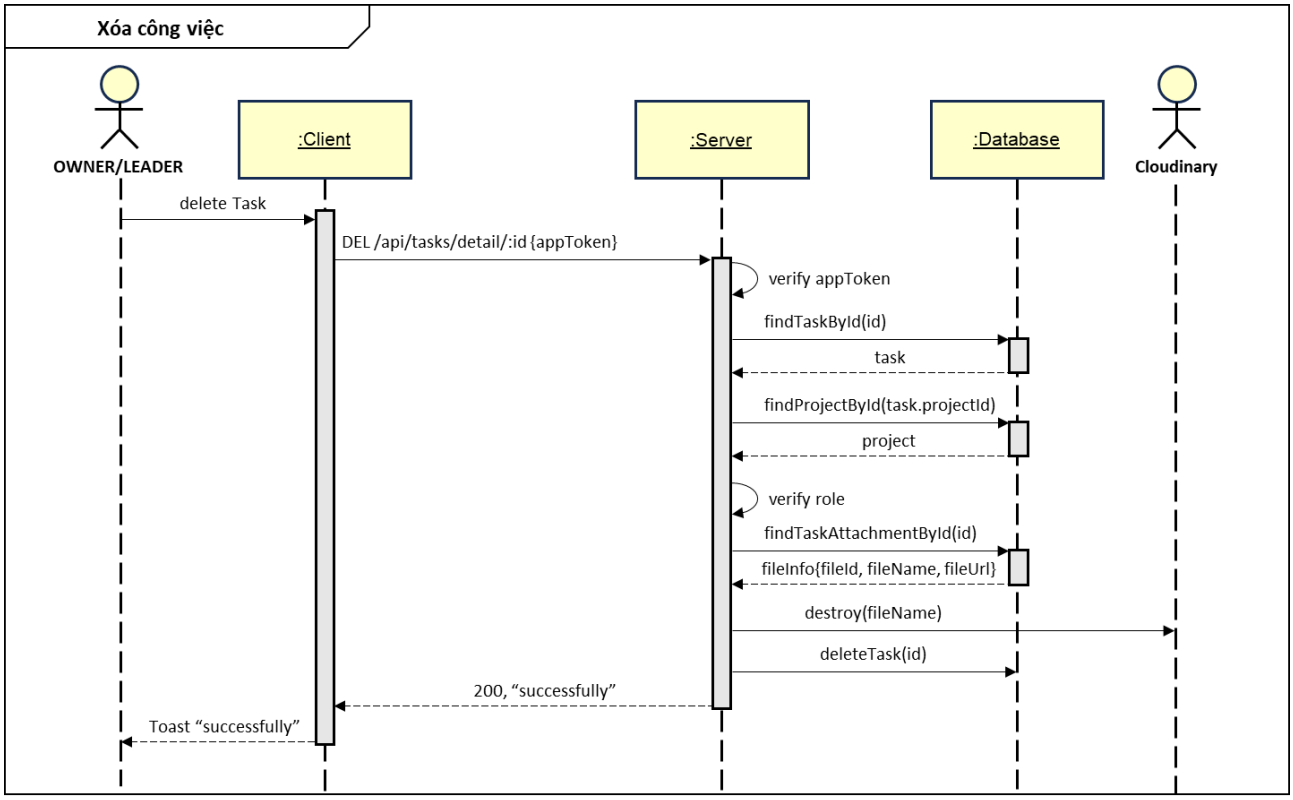
11. Xem chi tiết công việc



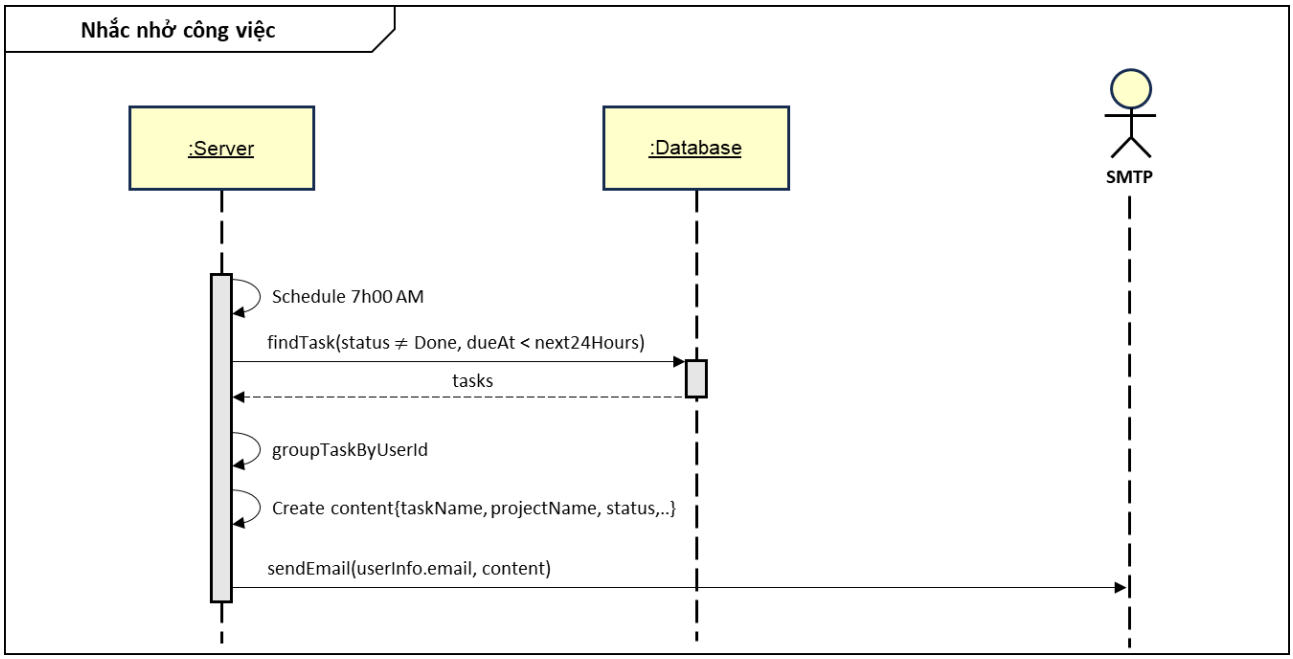
12. Xóa báo cáo



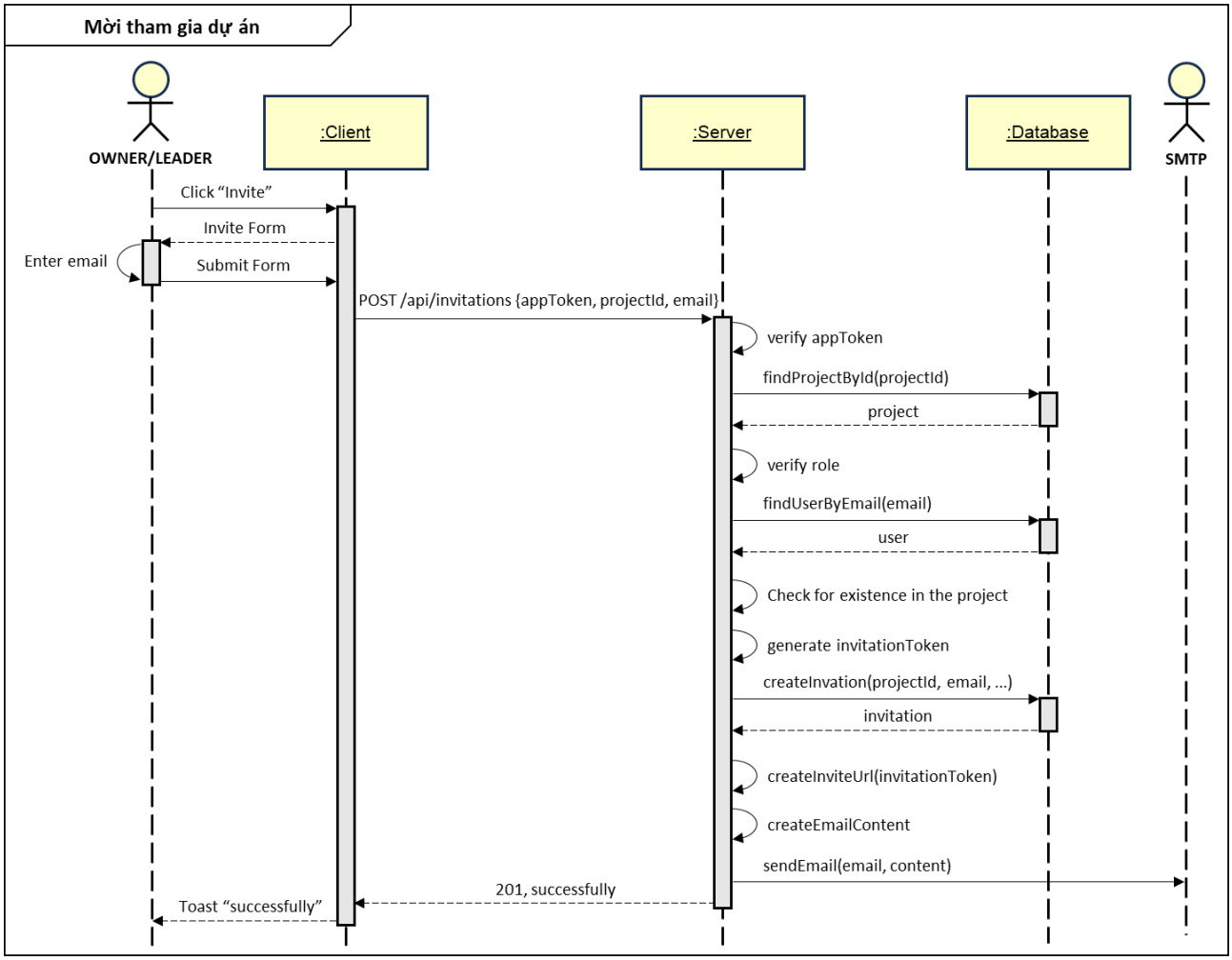
13. Xóa công việc



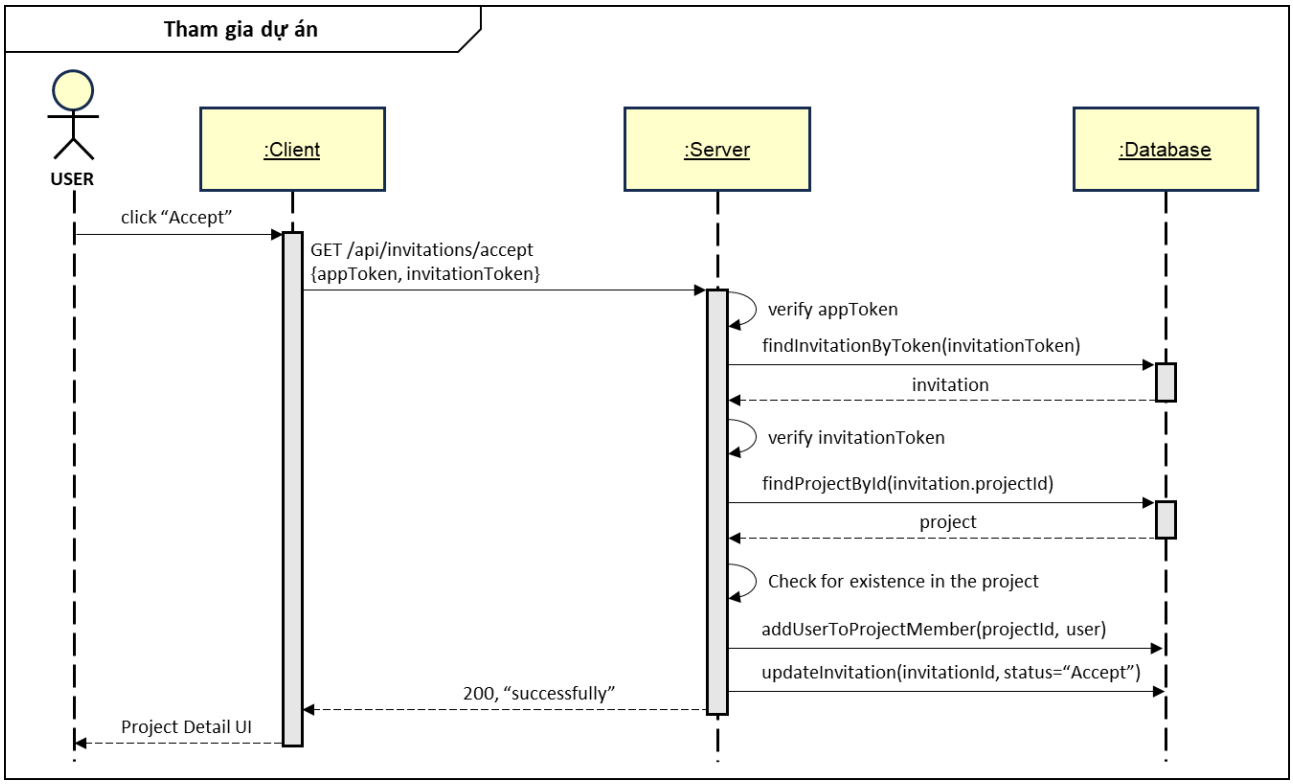
14. Nhắc nhở công việc



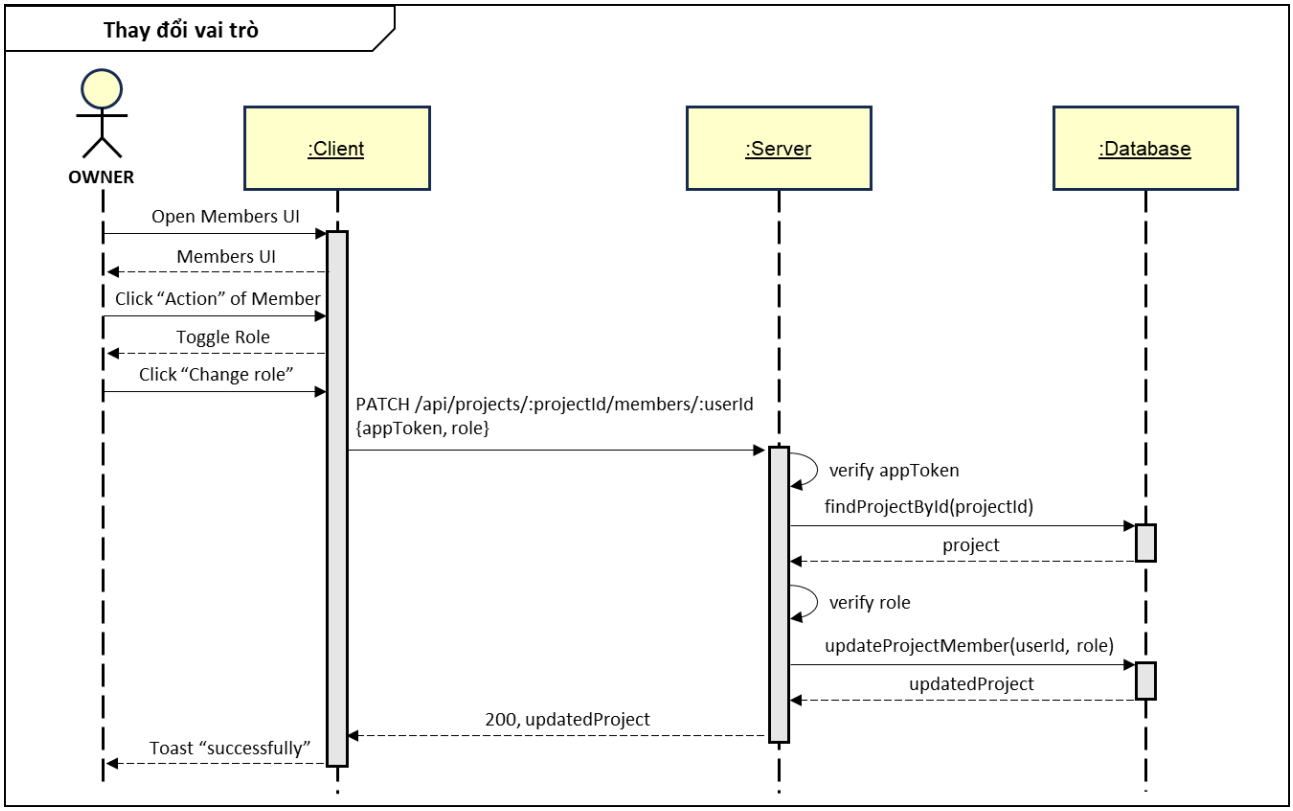
15. Mời tham gia dự án



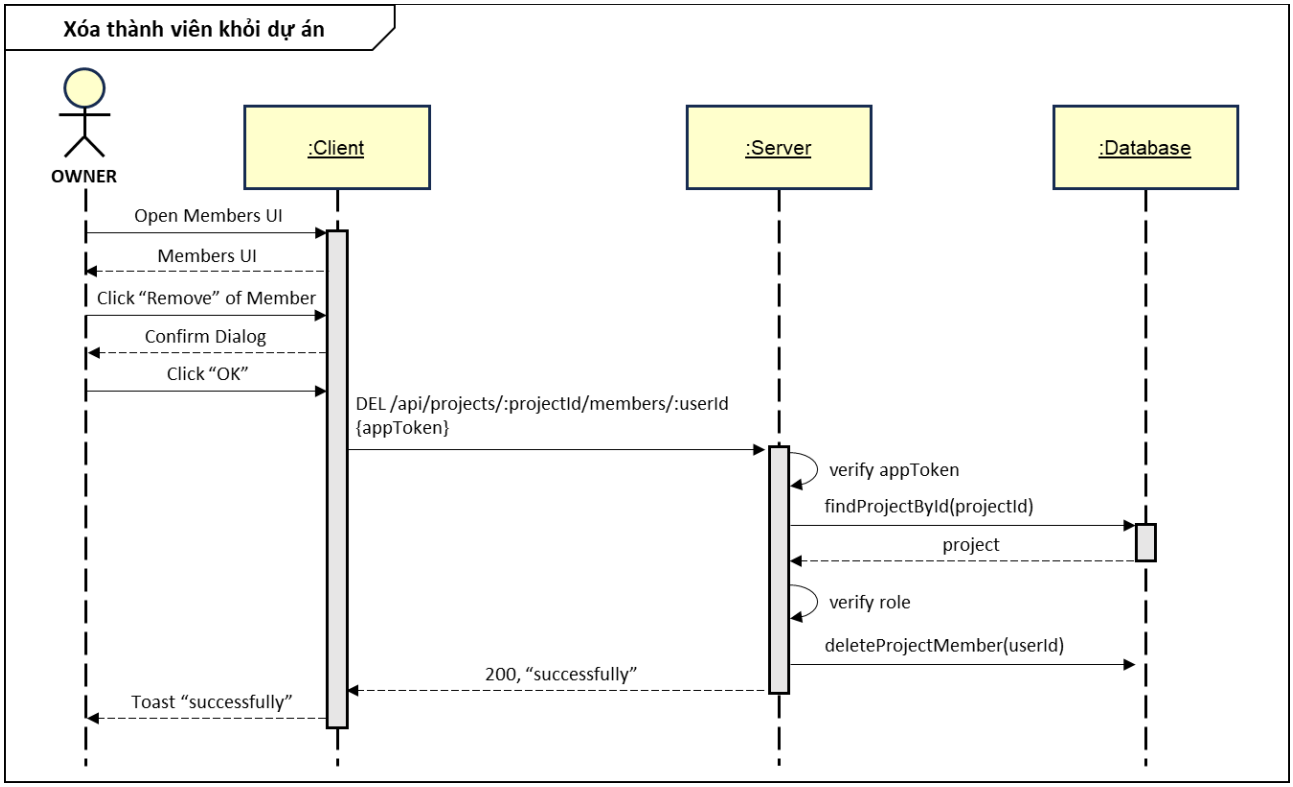
16. Tham gia dự án qua Gmail



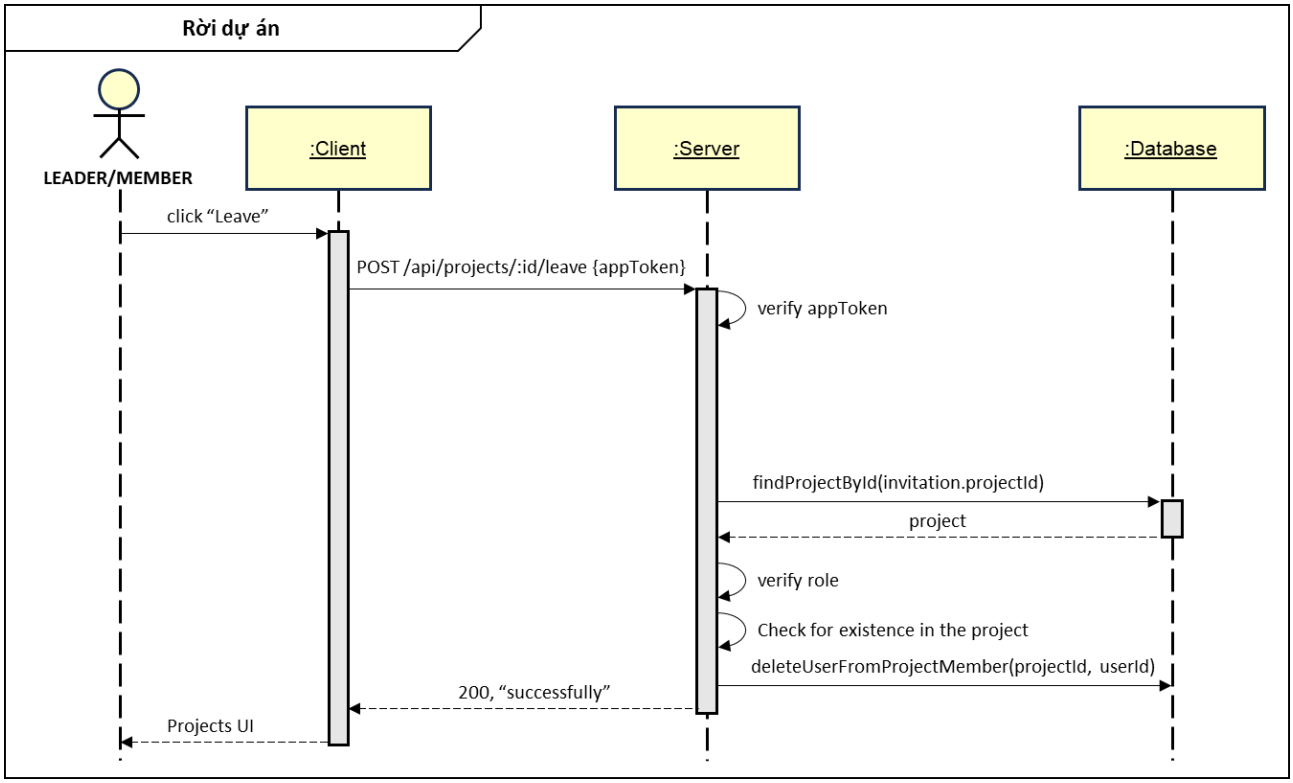
17. Thay đổi vai trò của thành viên



18. Xóa thành viên khỏi dự án



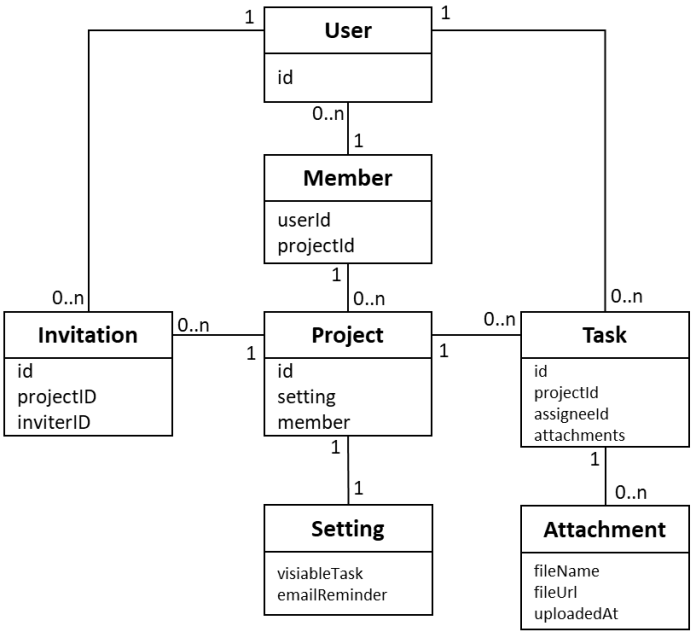
19. Rời dự án



3.4. Thiết kế Cơ sở dữ liệu

Lựa chọn hệ quản trị CSDL: MongoDB được chọn vì mô hình dữ liệu linh hoạt, phù hợp với cấu trúc bán cấu trúc của dự án và công việc. Chiến lược nhúng dữ liệu (embedding) được áp dụng cho danh sách thành viên và cài đặt trong collection Projects để tối ưu hóa hiệu suất đọc.

3.4.1 Sơ đồ quan hệ thực thể (ERD)



3.4.2 Mô tả các Collection

a) Users Collection

Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả & Ràng buộc
id	ObjectId	Có	Khóa chính (Primary Key)
googleId	String	Có	ID người dùng từ Google/Firebase
email	String	Có	Email của người dùng
name	String	Có	Tên hiển thị
avatarUrl	String	Không	URL ảnh đại diện
createdAt	Date	Có	Thời gian tạo tài khoản

b) Projects Collection

Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả & Ràng buộc
id	ObjectId	Có	Khóa chính
name	String	Có	Tên dự án
description	String	Không	Mô tả dự án
ownerId	ObjectId	Có	Tham chiếu Users.id
status	String	Có	Enum: ['ACTIVE', 'ARCHIVED']
settings	Object	Có	Cấu trúc nhúng (xem bên dưới)
members	Array<Object>	Có	Danh sách thành viên (xem bên dưới)
createdAt	Date	Có	Thời gian tạo dự án

Cấu trúc object settings

Tên trường con	Kiểu dữ liệu	Mô tả
allowMemberViewAllTasks	Boolean	true: Member xem tất cả task; false: chỉ xem task được giao
enableEmailReminders	Boolean	true: bật nhắc việc qua email; false: tắt

Cấu trúc phần tử trong mảng members

Tên trường con	Kiểu dữ liệu	Mô tả
userId	ObjectId	Tham chiếu Users.id
role	String	Enum: ['OWNER', 'LEADER', 'MEMBER']
joinedAt	Date	Ngày tham gia dự án

c) Tasks Collection

Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả & Ràng buộc
id	ObjectId	Có	Khóa chính
projectId	ObjectId	Có	Tham chiếu Projects.id
title	String	Có	Tên công việc
description	String	Không	Mô tả chi tiết
status	String	Có	Enum: ['TODO', 'IN_PROGRESS', 'REVIEW', 'DONE']
priority	String	Có	Enum: ['LOW', 'MEDIUM', 'HIGH']
assigneeId	ObjectId	Không	Tham chiếu Users.id
dueAt	Date	Không	Hạn chót hoàn thành
isReminded	Boolean	Không	Đã gửi email nhắc nhở chưa (mặc định false)
attachments	Array<Object>	Không	Danh sách tệp đính kèm (xem bên dưới)

Cấu trúc phần tử trong mảng attachments

Tên trường con	Kiểu dữ liệu	Mô tả
fileName	String	Tên file hiển thị
fileUrl	String	Đường dẫn file (Cloudinary)
uploadedAt	Date	Thời gian upload

d) Invitations Collection

Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả & Ràng buộc
id	ObjectId	Có	Khóa chính
projectId	ObjectId	Có	Tham chiếu Projects.id
inviterId	ObjectId	Có	Tham chiếu Users.id
email	String	Có	Email người được mời
role	String	Có	Vai trò dự kiến của người được mời
token	String	Có	Mã xác thực lời mời
expiresAt	Date	Có	Thời gian hết hạn token
createdAt	Date	Có	Thời gian tạo lời mời

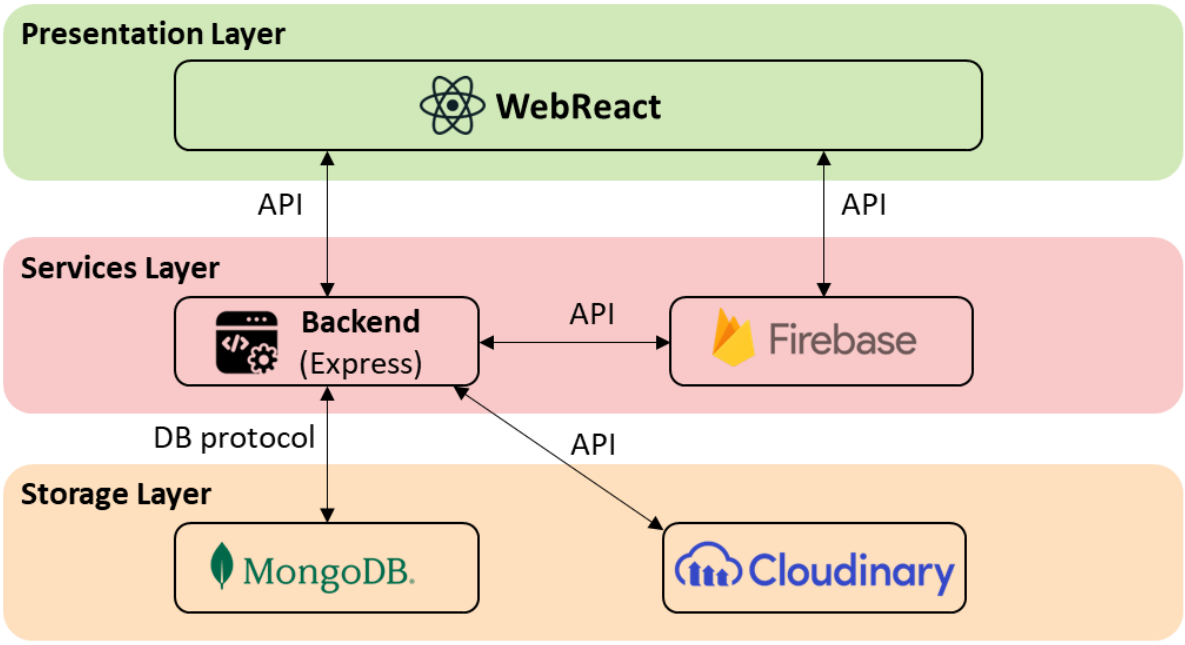
3.5. Thiết kế Kiến trúc hệ thống

3.5.1. Mô hình kiến trúc

Hệ thống được thiết kế theo **kiến trúc 3 lớp (3-Tier Architecture)**, một mô hình phổ biến cho các ứng dụng web hiện đại, giúp tách biệt các mối quan tâm và tăng cường khả năng bảo trì.

1. **Presentation Layer (Lớp trình diễn):** Được xây dựng bằng **ReactJS**, lớp này chịu trách nhiệm hiển thị giao diện người dùng và xử lý các tương tác từ người dùng. Nó giao tiếp với lớp ứng dụng thông qua các lời gọi API.
2. **Application Layer (Lớp ứng dụng - Backend):** Được xây dựng bằng **Node.js và ExpressJS**, đây là lõi logic của hệ thống. Lớp này tiếp nhận các yêu cầu từ lớp trình diễn, xử lý logic nghiệp vụ, xác thực, phân quyền và giao tiếp với lớp dữ liệu.
3. **Data Layer (Lớp dữ liệu):** Bao gồm cơ sở dữ liệu **MongoDB**, chịu trách nhiệm lưu trữ và truy xuất dữ liệu một cách bền vững.

3.5.2. Sơ đồ kiến trúc tổng thể



3.5.3. Thiết kế API

a) Auth & User

Method	Endpoint	Mô tả
POST	/api/auth/google	Đăng nhập bằng Google (Firebase ID Token)
GET	/api/auth/me	Lấy thông tin người dùng hiện tại (dựa trên JWT)

b) Projects

Method	Endpoint	Mô tả	Yêu cầu quyền
GET	/api/projects	Lấy danh sách các dự án mà người dùng tham gia	Member
POST	/api/projects	Tạo dự án mới	Member
GET	/api/projects/:id	Xem chi tiết thông tin dự án	Member
PATCH	/api/projects/:id	Cập nhật thông tin cơ bản và cấu hình dự án	Owner
DELETE	/api/projects/:id	Xóa dự án (yêu cầu xác nhận tên dự án)	Owner

c) Members

Method	Endpoint	Mô tả	Yêu cầu quyền
PATCH	/api/projects/:id/members/:userId	Thay đổi vai trò của thành viên trong dự án	Owner
DELETE	/api/projects/:id/members/:userId	Xóa thành viên khỏi dự án	Owner
POST	/api/projects/:id/leave	Thành viên tự rời khỏi dự án (xác nhận 2 bước)	Member

e) Tasks

Method	Endpoint	Mô tả	Yêu cầu quyền	Query Params
GET	/api/tasks	Lấy danh sách task theo bộ lọc	Member	projectId, assigneeId
POST	/api/tasks	Tạo task mới	Leader / Owner	
GET	/api/tasks/:id	Xem chi tiết task	Member	
PATCH	/api/tasks/:id	Cập nhật thông tin task	Leader / Owner	
DELETE	/api/tasks/:id	Xóa task	Leader / Owner	
POST	/api/tasks/:id/attachments	Upload tệp đính kèm cho task	Leader / Owner	

f) Invitations

Method	Endpoint	Mô tả	Yêu cầu quyền
POST	/api/invitations	Gửi email mời tham gia dự án	Owner
POST	/api/invitations/accept	Xác nhận tham gia dự án thông qua token mời	

3.5.4. Thiết kế Giao diện người dùng (UI)

Công cụ sử dụng: **Figma**

Giao diện (Mockup): TaskTrackerUI

Chương 4. Triển khai và Thử nghiệm

Giai đoạn này tập trung vào việc hiện thực hóa bản thiết kế đã được xây dựng ở chương 3, đồng thời tiến hành các hoạt động kiểm thử để đảm bảo chất lượng, sự ổn định và độ tin cậy của hệ thống trước khi đưa vào sử dụng.

4.1. Môi trường triển khai

- **Môi trường phát triển (Local Development):**
  - **Runtime:** Node.js (phiên bản LTS).
  - **Package Manager:** npm
  - **Database:** MongoDB Community Server
  - **Code Editor:** Visual Studio.
- **Môi trường sản phẩm (Production):**
  - **Frontend:** Triển khai trên các nền tảng chuyên dụng cho ứng dụng tĩnh/SPA như **Vercel** hoặc Netlify để tận dụng CDN và hiệu suất tối ưu.
  - **Backend:** Triển khai trên các nền tảng PaaS (Platform as a Service) như **Heroku**, AWS Elastic Beanstalk, hoặc DigitalOcean App Platform.
  - **Database:** Sử dụng dịch vụ **MongoDB Atlas** để đảm bảo tính sẵn sàng cao, khả năng sao lưu và bảo mật.
- **Hướng dẫn cài đặt:**
  1. Clone repository từ GitHub.
  2. Trong thư mục server, chạy npm install để cài đặt các dependency cho backend.
  3. Trong thư mục client, chạy npm install để cài đặt các dependency cho frontend.
  4. Tạo file .env trong thư mục server từ file mẫu .env.example.
  5. Điền các biến môi trường cần thiết: chuỗi kết nối MongoDB, Client ID và Client Secret của Google OAuth, API key của Cloudinary, và thông tin xác thực STMP.
  6. Chạy npm start trong cả hai thư mục để khởi động backend và frontend.

4.2. Kịch bản thử nghiệm (Test Case)

Bảng dưới đây trình bày một số kịch bản kiểm thử cho các chức năng cốt lõi.

ID	Tên chức năng	Các bước thực hiện	Kết quả mong đợi	Kết quả thực tế	Trạng thái
TC-01	Đăng nhập	1. Truy cập trang đăng nhập. 2. Nhấn nút "Đăng nhập với Google". 3. Chọn một tài khoản Google hợp lệ và xác thực.	Người dùng được chuyển hướng đến trang Dashboard, tên người dùng hiển thị chính xác.	Đúng theo mô tả	PASS

TC-02	Owner tạo dự án mới	1. Đăng nhập với vai trò Owner. 2. Tại Dashboard, nhấn nút "Tạo dự án mới". 3. Nhập tên, mô tả và nhấn "Tạo".	Dự án mới xuất hiện trong danh sách dự án. Người tạo tự động có vai trò Owner trong dự án đó.	Đúng theo mô tả	PASS
TC-03	Leader không thể xóa dự án	1. Đăng nhập với vai trò Leader. 2. Truy cập vào trang chi tiết dự án. 3. Tìm kiếm nút/chức năng "Xóa dự án".	Nút/chức năng "Xóa dự án" không hiển thị hoặc bị vô hiệu hóa. API xóa dự án trả về lỗi 403 Forbidden.	Đúng theo mô tả	PASS
TC-04	Member chỉ xem task được giao	1. Owner vào cài đặt, bật tùy chọn "Chỉ cho phép Member xem task được giao". 2. Đăng nhập với vai trò Member. 3. Truy cập trang dự án.	Member chỉ thấy các task có assigneeId là ID của chính mình trên Kanban board.	Đúng theo mô tả	PASS
TC-05	Xóa dự án với xác nhận sai	1. Đăng nhập với vai trò Owner. 2. Chọn xóa dự án. 3. Nhập một tên không khớp với tên dự án và xác nhận.	Hệ thống hiển thị thông báo lỗi, dự án không bị xóa.	Đúng theo mô tả	PASS

4.3. Đánh giá chất lượng mã nguồn

4.3.1 Đánh giá bằng SonarQube

a) SonarQube

SonarQube là một công cụ phân tích mã nguồn tĩnh tự động, giúp phát hiện lỗi (bugs), lỗ hổng bảo mật (vulnerabilities) và các vấn đề về chất lượng mã nguồn (code smells). Việc tích hợp SonarQube vào quy trình CI/CD giúp duy trì chất lượng mã nguồn ở mức cao.

b) Môi trường và công nghệ sử dụng

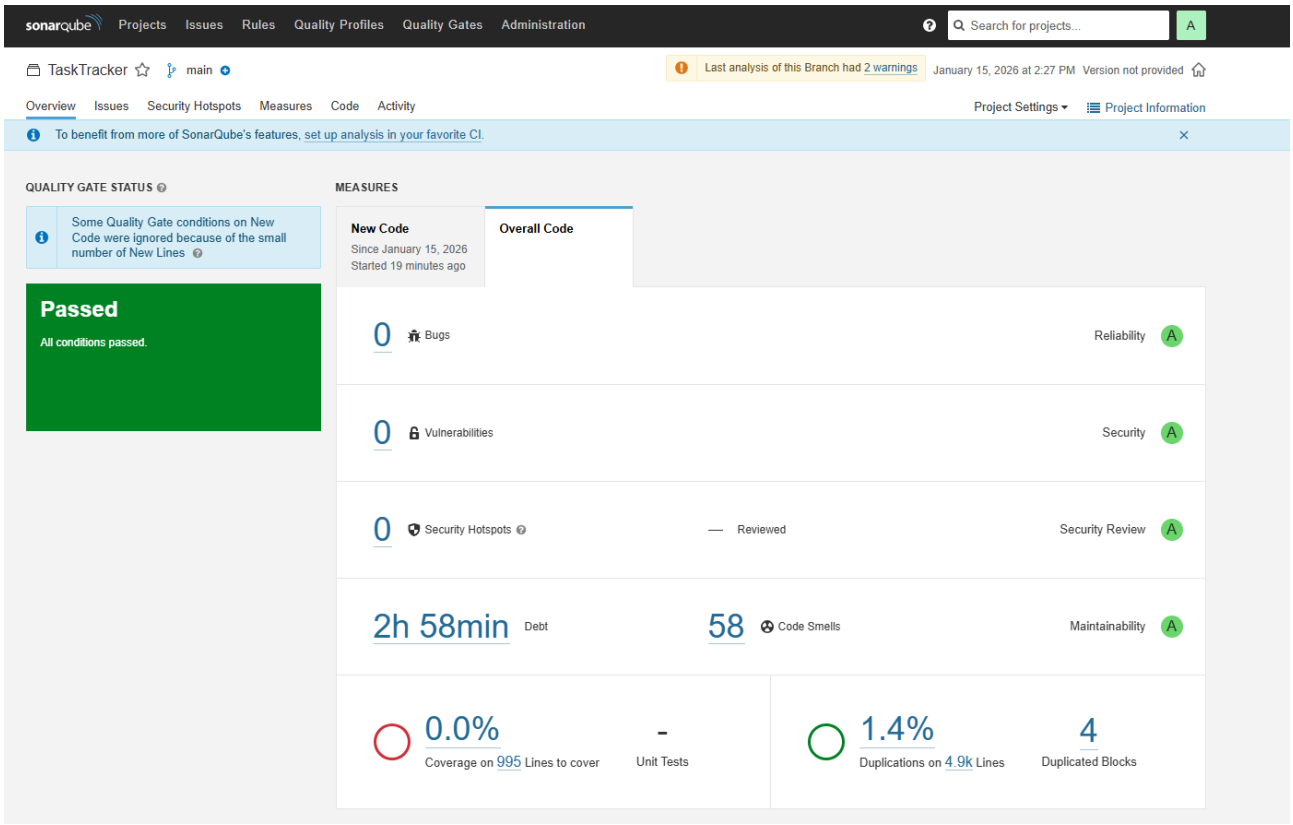
- Hệ điều hành: Windows
- Kiến trúc hệ thống: MERN (MongoDB, Express, React, Node.js)
- Công cụ kiểm thử: SonarQube Community Edition
- Phương thức triển khai: SonarQube chạy bằng Docker
- Phạm vi kiểm thử: mã nguồn phía client và server

c) Các bước kiểm thử

Quy trình kiểm thử được thực hiện theo các bước sau:

1. Triển khai SonarQube bằng Docker trên môi trường local.
2. Tạo project kiểm thử cho hệ thống.
3. Cấu hình file **sonar-project.properties** để xác định phạm vi mã nguồn cần phân tích.
4. Thực hiện phân tích mã nguồn bằng SonarScanner.
5. Thu thập và đánh giá kết quả kiểm thử trên giao diện SonarQube.

d) Kết quả thử nghiệm



Kết quả kiểm thử cho thấy hệ thống đạt Quality Gate với trạng thái **Passed**. Cụ thể:Bugs: 0

- Vulnerabilities: 0
- Security Hotspots: 0 (đã được review)
- Code Smells: 58
- Maintainability: A

e) Phân tích và xử lý lỗi

Trong quá trình kiểm thử, SonarQube phát hiện một số lỗi liên quan đến việc sử dụng React Hooks không đúng quy tắc (Hooks được gọi có điều kiện). Nhóm đã phân tích nguyên nhân và refactor lại

component để đảm bảo các Hook luôn được gọi ở top-level của component. Sau khi sửa lỗi và thực hiện kiểm thử lại, số lượng Bugs và Security Hotspots đã giảm về 0.

f) Nhận xét và đánh giá

Việc sử dụng SonarQube giúp phát hiện sớm các lỗi tiềm ẩn và cải thiện chất lượng mã nguồn. Công cụ đặc biệt hữu ích trong việc phát hiện lỗi logic và các vấn đề bảo mật ngay từ giai đoạn phát triển, góp phần nâng cao độ ổn định và khả năng bảo trì của hệ thống.

4.3.2 Checklist Đánh giá Thủ công:

Hạng mục	Tiêu chí kiểm tra
Bugs	Xử lý lỗi: Mọi lời gọi API, thao tác CSDL đều nằm trong khối try-catch hoặc xử lý promise .catch() Kiểm tra giá trị null/undefined trước khi truy cập thuộc tính.
Vulnerabilities	Không lưu trữ thông tin nhạy cảm (API keys, credentials) trực tiếp trong code, thay vào đó sử dụng biến môi trường. Xác thực đầu vào của người dùng (input validation) ở phía backend để chống lại các tấn công như XSS, Injection. Kiểm soát quyền truy cập API chặt chẽ, đảm bảo người dùng không thể thực hiện hành động ngoài vai trò của mình.
Code Smells	Trùng lặp mã: Các đoạn mã lặp lại nên được tái cấu trúc thành các hàm hoặc component có thể tái sử dụng. Hàm/Component quá lớn: Các hàm hoặc component có quá nhiều trách nhiệm cần được chia nhỏ. Đặt tên: Tên biến, hàm, component phải rõ ràng, có ý nghĩa và tuân theo quy ước chung.

Đề xuất cải tiến:

1. **Tái cấu trúc (Refactoring):** Xem xét các hàm xử lý logic nghiệp vụ phức tạp ở backend và chia nhỏ chúng thành các service module riêng biệt để tăng tính rõ ràng và khả năng kiểm thử.
2. **Tăng cường bảo mật:** Đảm bảo tất cả các biến môi trường được quản lý chặt chẽ thông qua file .env và không bị commit lên kho mã nguồn.
3. **Viết Unit Test:** Bổ sung các bài kiểm thử đơn vị (unit test) cho các hàm xử lý logic quan trọng ở backend (sử dụng Jest, Mocha) và các component ở frontend (sử dụng React Testing Library) để đảm bảo tính đúng đắn và ngăn ngừa lỗi hồi quy (regression).

Quá trình triển khai và thử nghiệm đã xác nhận rằng hệ thống hoạt động đúng theo thiết kế. Các đề xuất cải tiến sẽ được xem xét cho các phiên bản phát triển trong tương lai.

Chương 5. Tài liệu tham khảo

1. React Official Documentation.
2. Node.js Official Documentation
3. Express - Node.js web application framework
4. MongoDB Official Documentation
5. Google Identity - OAuth 2.0 for Web Server Applications
6. Cloudinary - Programmable Media Documentation

Phụ lục

Cấu trúc thư mục dự án

```
task-tracker/
├── client/                    # Frontend ReactJS
│   ├── public/
│   ├── src/
│   │   ├── api/               # Hàm gọi API
│   │   ├── assets/            # Tài nguyên tĩnh
│   │   ├── components/        # Component tái sử dụng
│   │   ├── hooks/             # Custom hooks
│   │   ├── pages/             # Các trang chính
│   │   └── App.js
│   └── package.json
├── server/                    # Backend Node.js/ExpressJS
│   ├── config/                # Cấu hình (DB, etc.)
│   ├── controllers/           # Logic xử lý request
│   ├── middleware/            # Middleware (auth, error)
│   ├── models/                # Mongoose schemas
│   ├── routes/                # Định tuyến API
│   └── server.js
└── README.md
```

File cấu hình mẫu (.env.example)

```
# MongoDB Connection
MONGO_URI=mongodb+srv://<user>:<password>@<cluster-url>/tasktracker?retryWrites=true&w=majority

# Google OAuth 2.0 Credentials
GOOGLE_CLIENT_ID=your_google_client_id.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=your_google_client_secret

# JSON Web Token
JWT_SECRET=your_super_secret_jwt_key
JWT_EXPIRES_IN=7d

# Cloudinary Credentials
CLOUDINARY_CLOUD_NAME=your_cloud_name
CLOUDINARY_API_KEY=your_api_key
CLOUDINARY_API_SECRET=your_api_secret

# Gmail STMP for Sending Emails
STMP_HOST=STMP.gmail.com
STMP_PORT=587
STMP_USER=your_email@gmail.com
STMP_PASS=your_app_password
```