### MỤC LỤC

[CHƯƠNG 1. GIỚI THIỆU 1](#_bookmark0)

* 1. [Lý do chọn đề tài 1](#_bookmark1)
  2. [Mục tiêu dự án 1](#_bookmark2)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 3](#_bookmark3)

* 1. [Phương pháp luận phát triển: Agile và Scrum 3](#_bookmark4)
  2. [Kiến trúc hệ thống 4](#_bookmark6)
  3. [Công nghệ sử dụng 5](#_bookmark8)
     1. [Công nghệ phía Backend 5](#_bookmark9)
     2. [Công nghệ phía Frontend 5](#_bookmark10)
     3. [Công nghệ tích hợp Trí tuệ nhân tạo (AI) 6](#_bookmark11)
     4. [Công nghệ và Công cụ khác 6](#_bookmark13)

[CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 8](#_bookmark14)

* 1. [Phân tích Yêu cầu 8](#_bookmark15)
     1. [Yêu cầu chức năng 8](#_bookmark16)
     2. [Yêu cầu phi chức năng 9](#_bookmark17)
  2. [Thiết kế kiến trúc tổng thể 9](#_bookmark18)
     1. [Thiết kế Giao diện (UI/UX) 10](#_bookmark19)
        1. [Trang Đăng nhập / Đăng ký 10](#_bookmark20)
        2. [Trang chủ (Dashboard) 11](#_bookmark23)
        3. [Quản lý danh sách công việc 12](#_bookmark25)
        4. [Lịch công việc 12](#_bookmark27)
        5. [Quản lý dự án 13](#_bookmark29)
        6. [Thống kê 13](#_bookmark31)
     2. [Thiết kế xử lý 14](#_bookmark33)
        1. [Mô hình ERD 14](#_bookmark34)
        2. [Mô tả chi tiết các bảng 14](#_bookmark36)
        3. [Mô hình hệ thống và các luồng xử lý chính 16](#_bookmark37)
     3. [Thiết kế API 19](#_bookmark39)

[CHƯƠNG 4. QUẢN LÝ VÀ HIỆN THỰC HÓA DỰ ÁN 20](#_bookmark41)

* 1. [Quản lý dự án với Jira 20](#_bookmark42)
  2. [Quản lý mã nguồn với Git và GitHub 21](#_bookmark44)

[CHƯƠNG 5. KIỂM THỬ VÀ TRIỂN KHAI 24](#_bookmark47)

* 1. [Kiểm thử hệ thống (Testing) 24](#_bookmark48)
  2. [Đóng gói và Triển khai (Deployment) 25](#_bookmark50)

[CHƯƠNG 6. KẾT QUẢ, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN 27](#_bookmark51)

* 1. [Kết quả đạt được 27](#_bookmark52)
  2. [Hướng phát triển 28](#_bookmark55)

[TÀI LIỆU THAM KHẢO 29](#_bookmark56)

**DANH MỤC HÌNH ẢNH – BẢNG BIỂU**

[Hình 2.1 Sơ đồ quy trình Scrum 3](#_bookmark5)

[Hình 2.2 Sơ đồ kiến trúc Client-Server của ứng dụng WebsiteTimE 4](#_bookmark7)

[Hình 2.3 Sơ đồ tương tác với Google Gemini AI 6](#_bookmark12)

[Hình 3.1 Giao diện đăng nhập 10](#_bookmark21)

[Hình 3.2 Giao diện đăng ký 11](#_bookmark22)

[Hình 3.3 Giao diện trang chủ 11](#_bookmark24)

[Hình 3.4Giao diện quản lý danh sách công việc 12](#_bookmark26)

[Hình 3.5 Giao diện lịch công việc 12](#_bookmark28)

[Hình 3.6 Giao diện quản lý dự án 13](#_bookmark30)

[Hình 3.7 Giao diện thống kê 13](#_bookmark32)

[Hình 3.8 Mô hình ERD 14](#_bookmark35)

[Hình 3.9 Mô hình hệ thống và các luồng xử lý chính 16](#_bookmark38)

[Hình 3.10 Giao diện tài liệu API của WebsiteTimE trên Swagger 19](#_bookmark40)

[Hình 4.1 GitHub Repository của dự án 22](#_bookmark45)

[Hình 4.2 Lịch sử các Pull Request trên GitHub Repository của dự án 23](#_bookmark46)

[Hình 5.1 Kết quả kiểm thử API tạo công việc mới bằng Postman 25](#_bookmark49)

[Hình 6.1 Tính năng AI gợi ý thông minh khi tạo công việc mới 27](#_bookmark53)

[Hình 6.2 Giao diện trang Thống kê với các biểu đồ trực quan 28](#_bookmark54)

[Bảng 4.1 Bảng kế hoạch thực hiện các Sprint 20](#_bookmark43)

# CHƯƠNG 1. GIỚI THIỆU

## Lý do chọn đề tài

Trong thời đại công nghệ phát triển nhanh chóng như hiện nay, khối lượng công việc và thông tin mà mỗi người phải xử lý ngày càng nhiều. Điều này khiến việc quản lý thời gian và sắp xếp công việc hợp lý trở thành một kỹ năng quan trọng. Tuy nhiên, hầu hết các công cụ hỗ trợ hiện có vẫn còn rời rạc, thiếu sự thông minh và khó cá nhân hóa. Người dùng thường phải sử dụng nhiều ứng dụng khác nhau như lịch, ghi chú, danh sách việc cần làm…, gây bất tiện và dễ mất tập trung.

Xuất phát từ thực tế đó, nhóm quyết định xây dựng dự án "WebsiteTimE – Hệ thống Quản lý Thời gian Thông minh". Khác với các ứng dụng ghi chú đơn thuần, WebsiteTimE hướng đến một nền tảng toàn diện có tích hợp Trí tuệ nhân tạo (AI), hoạt động như một trợ lý ảo giúp người dùng ghi nhớ công việc, phân tích dữ liệu và đưa ra gợi ý sắp xếp thời gian một cách hợp lý. Qua đó, người dùng có thể làm việc hiệu quả hơn và duy trì được sự cân bằng giữa công việc và cuộc sống.

Thực hiện đề tài này cũng là dịp để nhóm áp dụng các kiến thức đã học trong môn Công nghệ Phần mềm vào một dự án thực tế – từ khâu phân tích yêu cầu, lựa chọn kiến trúc hệ thống, áp dụng quy trình phát triển phần mềm theo mô hình Agile/Scrum, đến việc sử dụng các công nghệ hiện đại như Next.js, NestJS và Docker để triển khai hệ thống.

## Mục tiêu dự án

Dự án được xây dựng với các mục tiêu chính sau:

Mục tiêu sản phẩm:

Xây dựng một ứng dụng web hoàn chỉnh với kiến trúc Client-Server, cho phép người dùng quản lý công việc (Tasks), dự án (Projects), và lịch trình (Calendar) một cách trực quan và hiệu quả.

Tích hợp thành công trợ lý AI "Dr.AITime" sử dụng Google Gemini AI, có khả năng phân tích, tóm tắt công việc, trò chuyện và đưa ra các gợi ý thông minh về độ ưu tiên và thời hạn.

Cung cấp giao diện người dùng hiện đại, đáp ứng (responsive) và có khả năng cá nhân hóa cao (chế độ sáng/tối, cài đặt hiển thị).

Mục tiêu học tập và kỹ thuật:

Vận dụng thành thạo quy trình phát triển phần mềm theo mô hình Agile và phương pháp Scrum, sử dụng Jira để quản lý và theo dõi tiến độ.

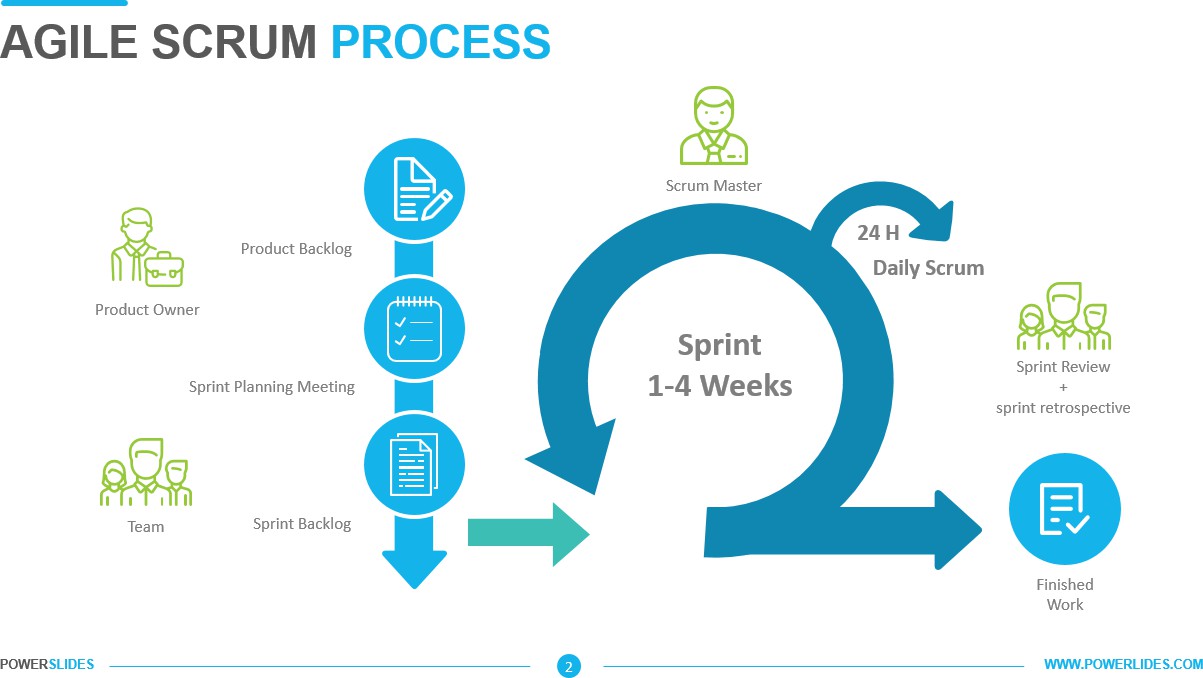
Xây dựng thành công một ứng dụng full-stack với Backend API sử dụng NestJS và Frontend sử dụng Next.js, giao tiếp qua RESTful API.

Thực hành quản lý mã nguồn chuyên nghiệp với Git/GitHub và triển khai ứng dụng bằng Docker, đảm bảo môi trường phát triển và vận hành nhất quán

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

## Phương pháp luận phát triển: Agile và Scrum

Agile là một phương pháp luận phát triển phần mềm linh hoạt, tập trung vào việc lặp lại (iterative) và tăng trưởng (incremental). Thay vì lập kế hoạch chi tiết từ đầu, Agile chia dự án thành các chu kỳ phát triển ngắn, cho phép nhóm nhanh chóng thích ứng với sự thay đổi yêu cầu và liên tục nhận phản hồi từ khách hàng.



Hình 2.1 Sơ đồ quy trình Scrum

Scrum là một framework quản lý dự án phổ biến nhất trong hệ sinh thái Agile. Nhóm đã áp dụng Scrum vì nó cung cấp một cấu trúc rõ ràng để quản lý các công việc phức tạp, đặc biệt phù hợp với một dự án có nhiều tính năng và yêu cầu có thể thay đổi như WebsiteTimE. Các thành phần chính của Scrum được áp dụng bao gồm:

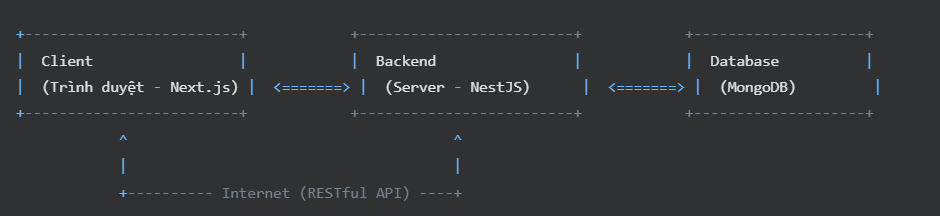
Các vai trò: Product Owner (đại diện cho người dùng), Scrum Master (hỗ trợ quy trình), và Development Team (nhóm phát triển).

Các sự kiện: Sprint (chu kỳ phát triển 2-4 tuần), Sprint Planning (lập kế hoạch cho Sprint), Daily Scrum (họp hàng ngày), Sprint Review (đánh giá sản phẩm cuối Sprint), và Sprint Retrospective (cải tiến quy trình).

Các công cụ: Product Backlog (danh sách yêu cầu) và Sprint Backlog (công việc trong một Sprint).

Công cụ hỗ trợ: Nhóm đã sử dụng Jira để quản lý Product Backlog, lên kế hoạch Sprints và theo dõi tiến độ qua các biểu đồ Burndown Chart.

## Kiến trúc hệ thống

****

Hình 2.2 Sơ đồ kiến trúc Client-Server của ứng dụng WebsiteTimE

WebsiteTimE được xây dựng dựa trên kiến trúc Client-Server tách biệt rõ ràng:

Client (Frontend): Là một ứng dụng Single-Page Application (SPA) được xây dựng bằng Next.js, chạy trên trình duyệt của người dùng. Client chịu trách nhiệm hiển thị giao diện, xử lý tương tác người dùng và gửi yêu cầu đến Server.

Server (Backend): Là một API server được xây dựng bằng NestJS. Server chịu trách nhiệm xử lý logic nghiệp vụ, xác thực người dùng, tương tác với cơ sở dữ liệu và trả về dữ liệu cho Client.

Dự án được tổ chức theo mô hình Monorepo, tức là toàn bộ mã nguồn của cả Frontend và Backend đều nằm trong một kho chứa (repository) duy nhất. Nhóm sử dụng Pnpm Workspace để quản lý mô hình này, mang lại các lợi ích:

Quản lý phụ thuộc tập trung: Dễ dàng quản lý các thư viện và phiên bản cho cả hai phần của dự án.

Tái sử dụng mã: Dễ dàng chia sẻ các module hoặc tệp định nghĩa (types) giữa Frontend và Backend trong tương lai.

Quy trình làm việc thống nhất: Các lệnh build, test, lint có thể được thực thi đồng thời cho toàn bộ dự án từ thư mục gốc.

Frontend và Backend giao tiếp với nhau thông qua một bộ các RESTful API. API được thiết kế theo các nguyên tắc của REST, sử dụng các phương thức HTTP (GET, POST, PUT, DELETE) để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên các tài nguyên của hệ thống như Users, Tasks, Projects.

## Công nghệ sử dụng

### Công nghệ phía Backend

NestJS được nhóm chọn làm framework chính cho phần Backend vì nó có kiến trúc dạng module rõ ràng, hỗ trợ tốt TypeScript và áp dụng được các nguyên tắc lập trình hướng đối tượng (OOP) lẫn lập trình hàm (FP). Trong quá trình làm, nhóm thấy việc tách chức năng và tổ chức mã nguồn bằng NestJS giúp dễ kiểm soát và mở rộng hệ thống sau này.

Fastify được tích hợp thay cho Express.js – vốn là mặc định trong NestJS – vì nhóm ưu tiên hiệu năng cao và độ trễ thấp. Khi thử nghiệm, Fastify cho tốc độ phản hồi API tốt hơn nên nhóm quyết định sử dụng xuyên suốt dự án.

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL mà nhóm lựa chọn, vì nó lưu trữ dữ liệu dưới dạng tài liệu (document) nên linh hoạt hơn trong việc thay đổi cấu trúc, đặc biệt hữu ích khi dự án vẫn đang trong quá trình phát triển và còn nhiều cập nhật.

Để làm việc với MongoDB, nhóm sử dụng thư viện Mongoose. Thư viện này giúp định nghĩa schema, truy vấn dữ liệu dễ dàng và có hỗ trợ kiểm tra dữ liệu (validation) ngay từ phía ứng dụng – điều này giúp hạn chế lỗi trước khi dữ liệu được lưu xuống DB.

### Công nghệ phía Frontend

React 19 là thư viện JavaScript chính mà nhóm dùng để xây dựng giao diện người dùng. Việc chia nhỏ các thành phần (component) giúp dễ tái sử dụng mã và quản lý trạng thái hiệu quả trong quá trình phát triển.

Nhóm sử dụng Next.js 15 làm framework cho phía Frontend vì nó hỗ trợ nhiều tính năng hữu ích như Server-Side Rendering (SSR), Static Site Generation (SSG) và hệ thống App Router. Những tính năng này giúp cải thiện hiệu năng, hỗ trợ SEO tốt hơn và giúp việc triển khai cũng thuận tiện hơn.

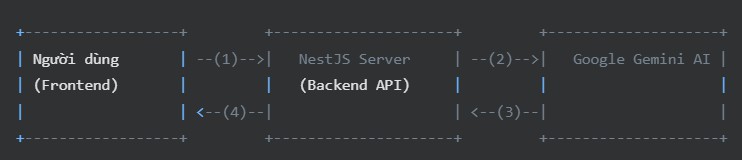
Dự án được viết hoàn toàn bằng TypeScript ở cả Frontend và Backend. Việc dùng TypeScript giúp nhóm dễ phát hiện lỗi khi code và giữ cho mã nguồn rõ ràng, dễ đọc, đặc biệt là khi làm việc theo nhóm hoặc mở rộng chức năng về sau.

Để xây dựng giao diện, nhóm chọn Tailwind CSS vì nó hoạt động theo hướng “utility-first”, cho phép thiết kế trực tiếp bằng cách kết hợp các class có sẵn ngay trong HTML hoặc JSX. Cách này giúp đẩy nhanh tốc độ làm giao diện mà không phải viết nhiều CSS riêng.

Ngoài ra, nhóm còn dùng Shadcn UI – một tập hợp các component dựng sẵn dựa trên Tailwind và Radix UI. Thay vì dùng thư viện như một dependency, nhóm sao chép mã của từng component cần dùng vào dự án và chỉnh sửa theo nhu cầu, giúp linh hoạt hơn trong việc tuỳ biến giao diện.

### Công nghệ tích hợp Trí tuệ nhân tạo (AI)

Gemini là một họ các mô hình ngôn ngữ lớn (LLM) đa phương thức của Google. Trong dự án, nhóm sử dụng mô hình Gemini để xử lý ngôn ngữ tự nhiên, thực hiện các tác vụ như tóm tắt văn bản, trả lời câu hỏi và đưa ra các gợi ý dựa trên ngữ cảnh công việc của người dùng.



Hình 2.3 Sơ đồ tương tác với Google Gemini AI

Để tích hợp Gemini vào backend NestJS một cách có cấu trúc, nhóm đã sử dụng Genkit. Đây là một framework mã nguồn mở từ Google, giúp đơn giản hóa việc xây dựng, triển khai và giám sát các luồng xử lý AI, quản lý prompts và kết nối tới các mô hình ngôn ngữ.

### Công nghệ và Công cụ khác

Docker: Được sử dụng để "container hóa" ứng dụng Frontend và Backend. Việc này đảm bảo ứng dụng chạy trong một môi trường nhất quán, độc lập với máy phát triển, giúp loại bỏ các lỗi do khác biệt môi trường.

Docker Compose: Dùng để định nghĩa và khởi chạy đồng thời nhiều container (Frontend, Backend, MongoDB) chỉ bằng một lệnh duy nhất, đơn giản hóa quá trình thiết lập môi trường phát triển.

# CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Phân tích Yêu cầu

### Yêu cầu chức năng

Quản lý Người dùng và Xác thực

* Người dùng có thể đăng ký tài khoản mới.
* Người dùng có thể đăng nhập vào hệ thống bằng email và mật khẩu.
* Hệ thống sử dụng JWT để xác thực và phân quyền truy cập.
* Người dùng có thể xem và cập nhật thông tin cá nhân (tên, avatar). Quản lý Công việc (Tasks)
* Cho phép người dùng tạo, xem, sửa, xóa công việc.
* Thiết lập thuộc tính cho công việc: tiêu đề, mô tả, ngày hết hạn, mức độ ưu tiên (Thấp, Trung bình, Cao).
* Gán công việc vào một Danh mục (Category) hoặc Dự án (Project) cụ thể.
* Theo dõi trạng thái công việc theo quy trình Scrum: Backlog, To Do, Doing, Done.

Quản lý Dự án (Projects)

* Cho phép tạo, xem, sửa, xóa dự án.
* Hiển thị danh sách công việc của từng dự án dưới dạng bảng Scrum.
* Theo dõi tiến độ tổng thể của dự án.

Lịch và Khối thời gian (Calendar & Time Blocks)

* Hiển thị lịch theo các chế độ: ngày, tuần, tháng.
* Tạo, xem, sửa, xóa các khối thời gian (time blocks).
* Cho phép liên kết khối thời gian với công việc cụ thể.

Tích hợp Trí tuệ nhân tạo (AI - Dr.AITime)

* AI Chatbox: giao diện trò chuyện để nhận lời khuyên quản lý thời gian.
* AI Assistant: tóm tắt tình hình công việc hiện tại.
* Gợi ý thông minh: đề xuất tự động ngày hết hạn và mức độ ưu tiên khi tạo công việc mới.

Thống kê và Báo cáo

* Hiển thị biểu đồ về tỉ lệ hoàn thành công việc.
* Thống kê công việc theo danh mục và mức độ ưu tiên.
* Phân tích thời gian sử dụng dựa trên các khối thời gian.

Cá nhân hóa và Cài đặt

* Cho phép chuyển đổi giao diện (Sáng/Tối/Hệ thống).
* Tùy chỉnh ngày bắt đầu của tuần trong lịch.
* Cấu hình các tuỳ chọn hiển thị và nhận thông báo.

### Yêu cầu phi chức năng

* Hiệu năng (Performance)
* Ứng dụng cần có thời gian tải trang nhanh.
* API Backend phải phản hồi dưới 500ms cho các thao tác cơ bản.
* Bảo mật (Security)
* Mật khẩu người dùng được mã hoá trước khi lưu vào cơ sở dữ liệu.
* Các API yêu cầu xác thực được bảo vệ bằng JWT.
* Dữ liệu người dùng phải được kiểm tra và lọc đầu vào để tránh lỗ hổng bảo mật.

## Thiết kế kiến trúc tổng thể

Hệ thống WebsiteTimE được thiết kế theo kiến trúc Client-Server, tổ chức dưới dạng Monorepo và giao tiếp qua RESTful API. Kiến trúc này đảm bảo sự tách biệt rõ ràng giữa logic giao diện (Frontend) và logic nghiệp vụ (Backend), giúp dễ dàng phát triển, bảo trì và mở rộng độc lập.

### Thiết kế Giao diện (UI/UX)

Giao diện người dùng được thiết kế trên công cụ Figma, tập trung vào trải nghiệm người dùng (UX) hiện đại, sạch sẽ và trực quan. Luồng người dùng (User Flow) được xây dựng để đảm bảo các thao tác chính như tạo công việc, lên lịch được thực hiện một cách nhanh chóng và dễ dàng.

* + - 1. *Trang Đăng nhập / Đăng ký*

### Giao diện xác thực người dùng gồm:

Form đăng nhập với email và mật khẩu Liên kết chuyển sang màn hình đăng ký Giao diện gọn gàng, dễ thao tác

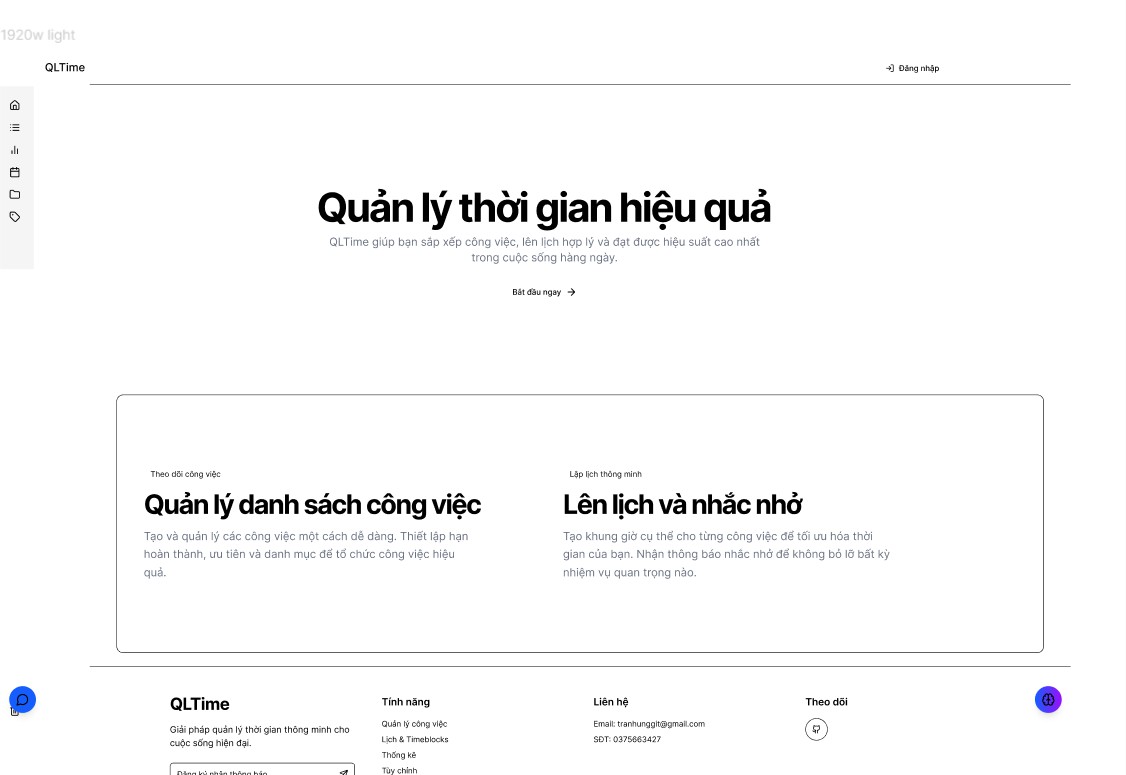


Hình 3.1 Giao diện đăng nhập



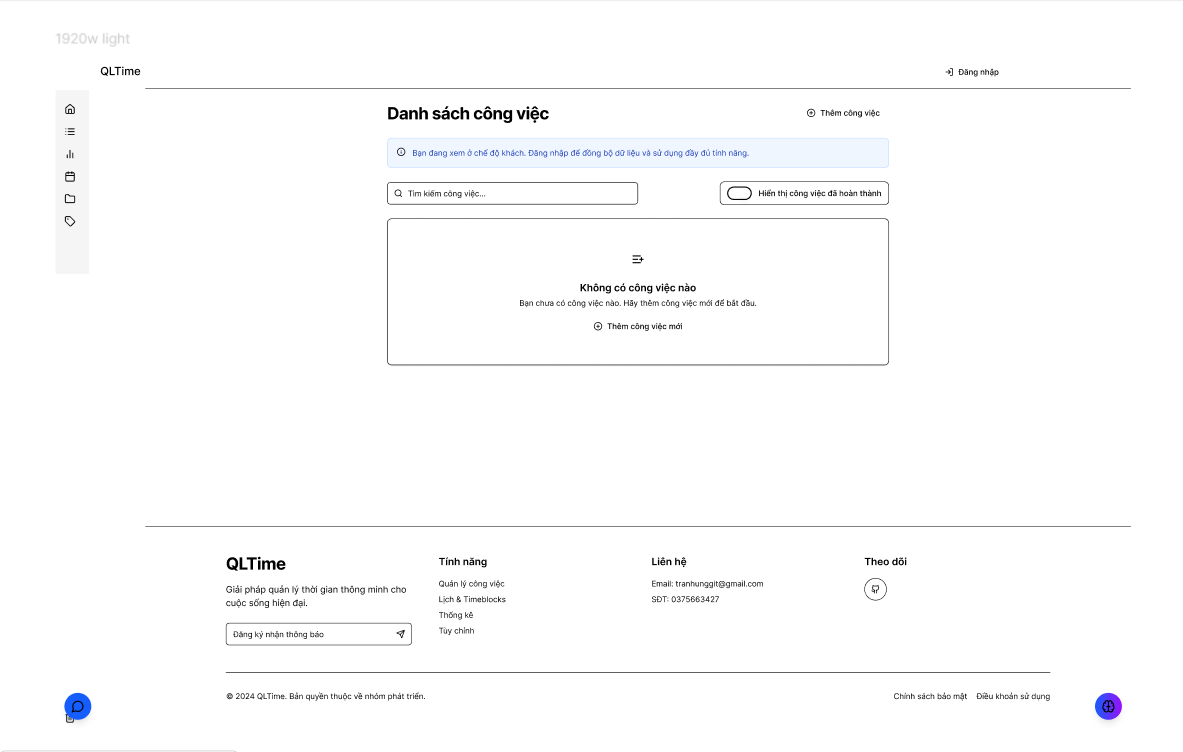
Hình 3.2 Giao diện đăng ký

* + - 1. *Trang chủ (Dashboard)*

**

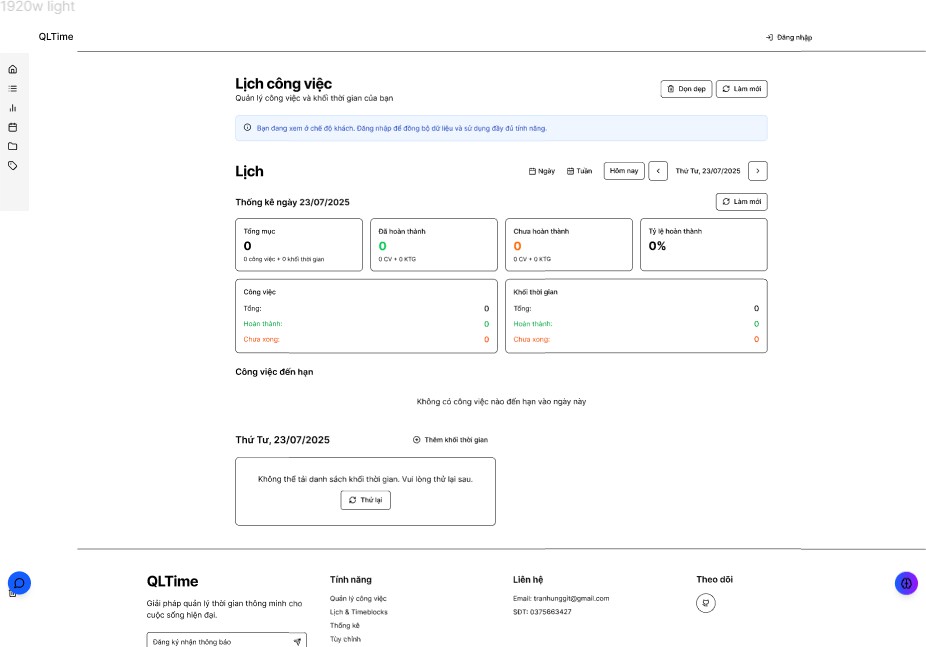
Hình 3.3 Giao diện trang chủ

* + - 1. *Quản lý danh sách công việc*

**

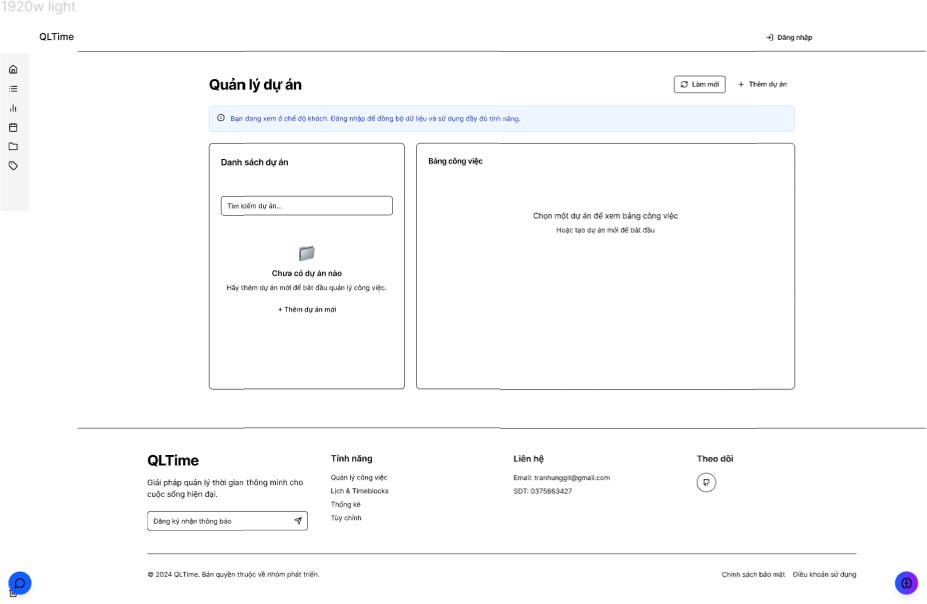
Hình 3.4Giao diện quản lý danh sách công việc

* + - 1. *Lịch công việc*

**

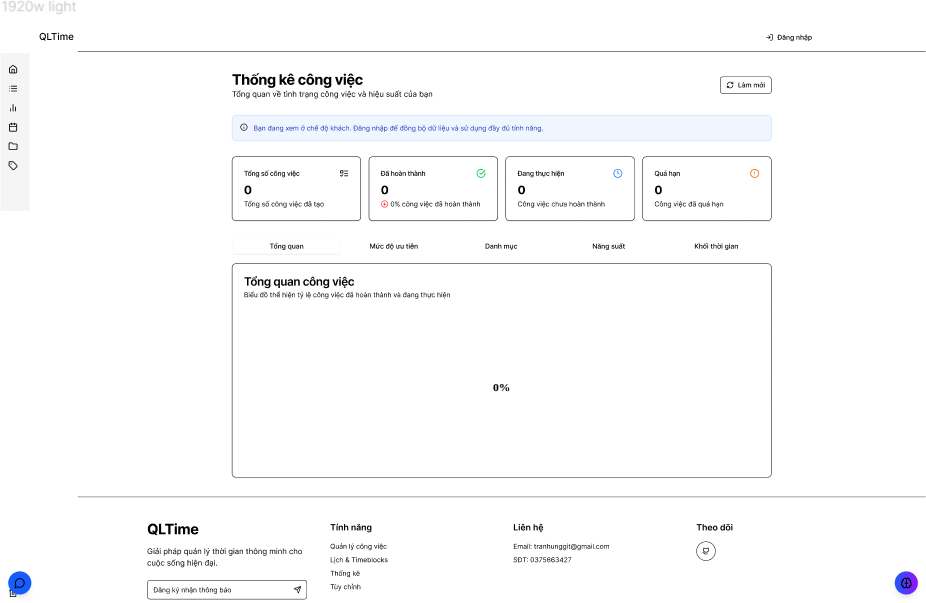
Hình 3.5 Giao diện lịch công việc

* + - 1. *Quản lý dự án*

**

Hình 3.6 Giao diện quản lý dự án

* + - 1. *Thống kê*

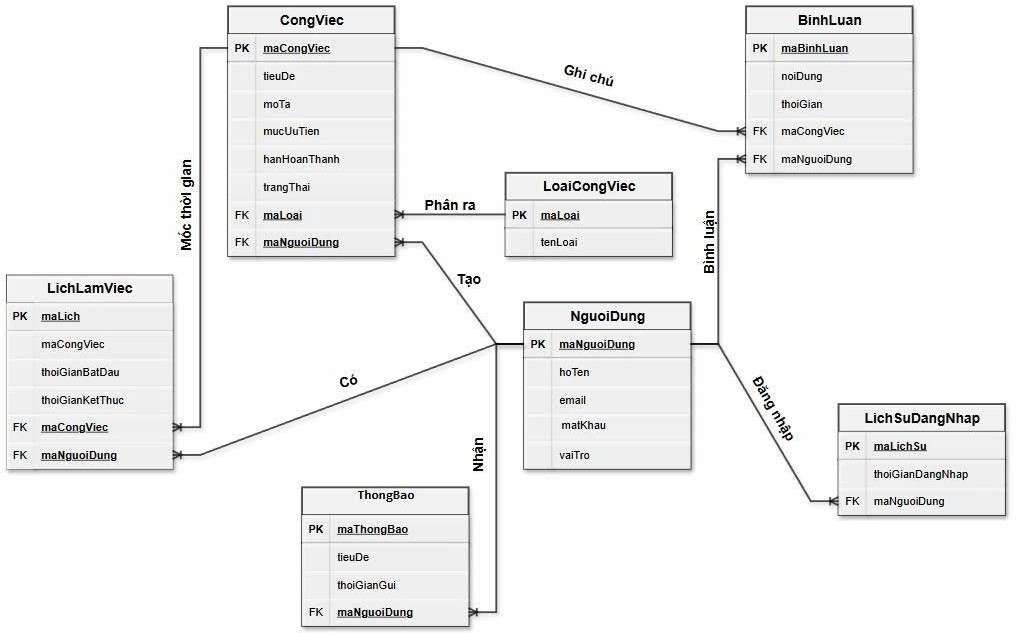
**

Hình 3.7 Giao diện thống kê

### Thiết kế xử lý

* + - 1. *Mô hình ERD*

Cấu trúc cơ sở dữ liệu được thiết kế theo dạng quan hệ (Entity Relationship Diagram – ERD), đảm bảo khả năng mở rộng, dễ quản lý và truy vấn dữ liệu.



Hình 3.8 Mô hình ERD

* + - 1. *Mô tả chi tiết các bảng*

### NguoiDung

Lưu thông tin người dùng trong hệ thống.

* + - * + maNguoiDung (PK): Mã định danh người dùng
        + hoTen: Họ tên người dùng
        + email: Địa chỉ email đăng nhập (duy nhất)
        + matKhau: Mật khẩu đã mã hóa
        + vaiTro: Vai trò trong hệ thống (user/admin)

### CongViec

Quản lý các nhiệm vụ/công việc của người dùng.

* + - * + maCongViec (PK)
        + tieuDe: Tên công việc
        + moTa: Mô tả nội dung
        + mucUuTien: Mức độ ưu tiên
        + hanHoanThanh: Hạn chót
        + trangThai: Trạng thái thực hiện (Backlog, To Do, Doing, Done)
        + maLoai (FK): Liên kết với bảng LoaiCongViec
        + maNguoiDung (FK): Người sở hữu công việc

### LoaiCongViec

Phân loại các công việc theo danh mục cụ thể.

* + - * + maLoai (PK)
        + tenLoai: Tên loại công việc (học tập, cá nhân, công việc...)

### LichLamViec

Lưu trữ các khối thời gian người dùng đã lập lịch.

* + - * + maLich (PK)
        + maCongViec (FK): Liên kết công việc tương ứng
        + maNguoiDung (FK): Người sở hữu lịch
        + thoiGianBatDau, thoiGianKetThuc: Mốc thời gian thực hiện

### BinhLuan

Cho phép người dùng để lại ghi chú hoặc bình luận cho công việc. maBinhLuan (PK)

noiDung: Nội dung bình luận thoiGian: Thời điểm tạo maCongViec (FK) maNguoiDung (FK)

### ThongBao

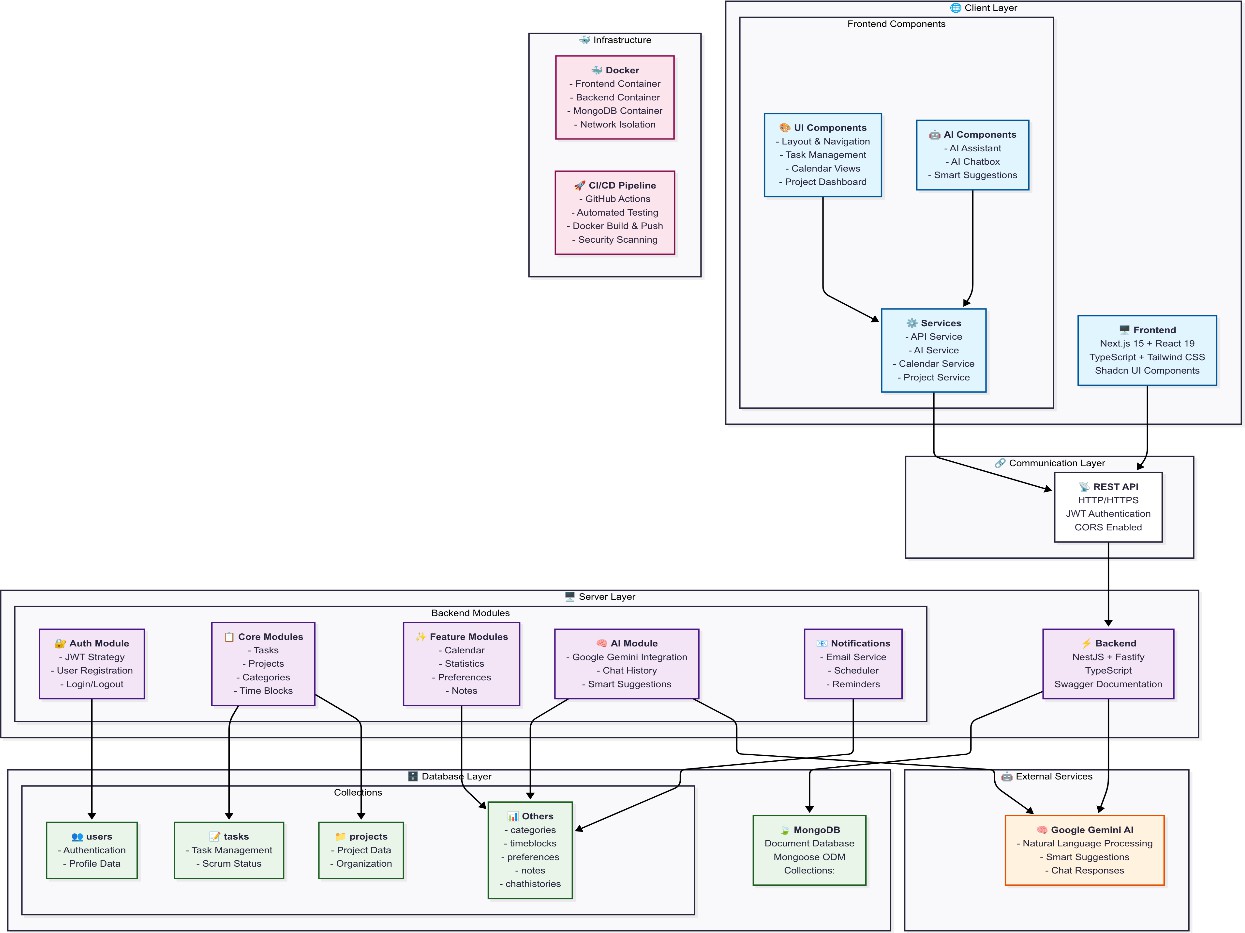
Thông báo gửi đến người dùng (nhắc việc, trạng thái...).

* + - * + maThongBao (PK)
        + tieuDe: Tiêu đề thông báo
        + thoiGianGui: Thời gian gửi
        + maNguoiDung (FK): Người nhận

### LichSuDangNhap

Lưu vết lịch sử truy cập hệ thống của người dùng.

* + - * + maLichSu (PK)
        + thoiGianDangNhap: Thời điểm đăng nhập
        + maNguoiDung (FK)
      1. *Mô hình hệ thống và các luồng xử lý chính*

**

Hình 3.9 Mô hình hệ thống và các luồng xử lý chính

### Tổng quan hệ thống

Hệ thống được thiết kế theo kiến trúc phân lớp rõ ràng, bao gồm:

Lớp Client (Giao diện người dùng): Nơi người dùng tương tác trực tiếp. Lớp Giao tiếp: Cầu nối giữa giao diện người dùng và phần xử lý chính. Lớp Máy chủ: Nơi xử lý logic nghiệp vụ và dữ liệu.

Lớp Cơ sở dữ liệu: Nơi lưu trữ tất cả dữ liệu.

Dịch vụ Bên ngoài: Các dịch vụ hỗ trợ từ bên ngoài hệ thống. Cơ sở hạ tầng: Cách thức triển khai và vận hành hệ thống.

### Các thành phần chính

Giao diện người dùng (Frontend):

Được xây dựng bằng công nghệ hiện đại như Next.js, React, TypeScript.

Bao gồm các thành phần giao diện (UI Components) như bố cục, điều hướng, quản lý tác vụ, lịch và bảng điều khiển.

Có các thành phần liên quan đến Trí tuệ nhân tạo (AI Components) như trợ lý AI, hộp trò chuyện và gợi ý thông minh.

Phần dịch vụ (Services) giúp giao diện người dùng kết nối với các tính năng phía sau như API, AI, lịch và dự án.

API (Giao tiếp):

Là cầu nối chính giữa giao diện người dùng và máy chủ. Sử dụng giao thức an toàn HTTP/HTTPS.

Bảo vệ bằng xác thực JWT và cho phép truy cập từ nhiều nguồn (CORS). Máy chủ (Backend):

Được xây dựng bằng NestJS và Fastify với TypeScript. Cung cấp tài liệu API rõ ràng (Swagger Documentation). Bao gồm các module:

Xác thực (Auth Module): Xử lý đăng ký, đăng nhập, đăng xuất và bảo mật người dùng.

Lõi (Core Modules): Quản lý các chức năng chính như công việc (tasks), dự án (projects), danh mục (categories) và khối thời gian (time blocks).

Tính năng bổ sung (Feature Modules): Cung cấp lịch, thống kê, tùy chỉnh và ghi chú.

AI Module: Tích hợp với Google Gemini AI để cung cấp các tính năng thông

minh.

Thông báo (Notifications): Xử lý email, lịch trình và nhắc nhở. Cơ sở dữ liệu (MongoDB):

Là loại cơ sở dữ liệu NoSQL, linh hoạt cho việc lưu trữ tài liệu. Sử dụng Mongoose để dễ dàng tương tác.

Lưu trữ dữ liệu về người dùng, công việc, dự án và các thông tin khác như

danh mục, ghi chú, lịch sử trò chuyện.

Dịch vụ bên ngoài (Google Gemini AI):

Là dịch vụ AI từ Google, được sử dụng để xử lý ngôn ngữ, đưa ra gợi ý thông minh và phản hồi trong cuộc trò chuyện.

Cơ sở hạ tầng:

Docker: Đóng gói ứng dụng thành các "hộp" riêng biệt (container) cho Frontend, Backend và MongoDB, giúp dễ dàng triển khai và quản lý.

CI/CD Pipeline (GitHub Actions): Quy trình tự động để kiểm thử, xây dựng và triển khai ứng dụng, đảm bảo chất lượng và an toàn liên tục.

Các luồn hoạt động gồm có:

* Luồng Xác thực và Quản lý Người dùng
* Luồng Quản lý Công việc
* Luồng Quản lý Dự án
* Luồng Quản lý Lịch & Time Blocks
* Luồng Tích hợp AI (Dr.AITime)
* Luồng Thống kê & Báo cáo

### Thiết kế API

Hệ thống API được thiết kế theo chuẩn RESTful, tài liệu hóa bằng Swagger (OpenAPI) để đảm bảo sự rõ ràng và dễ dàng tích hợp giữa Backend và Frontend.

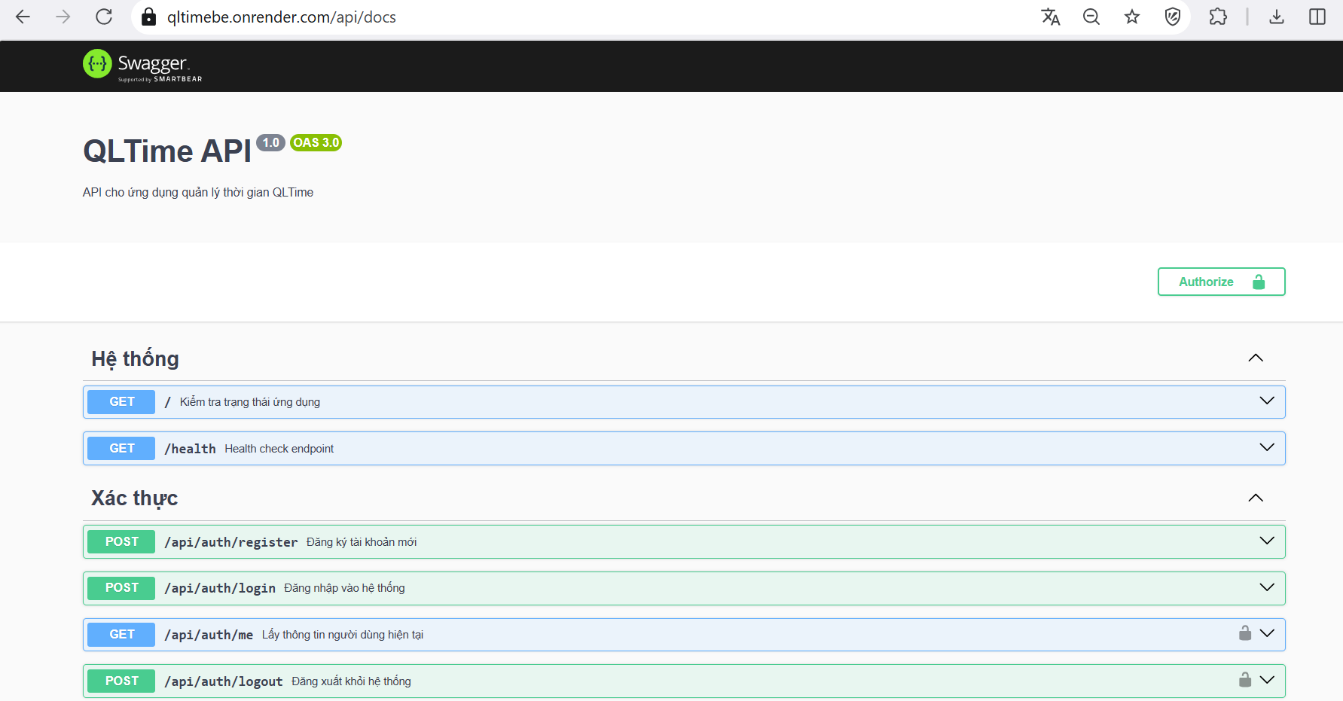
Các API endpoint chính bao gồm:

POST /api/auth/register: Đăng ký tài khoản. POST /api/auth/login: Đăng nhập.

GET /api/tasks: Lấy danh sách công việc. POST /api/tasks: Tạo công việc mới.

PUT /api/tasks/:id: Cập nhật công việc. DELETE /api/tasks/:id: Xóa công việc.

POST /api/ai/chat: Tương tác với AI Chatbot.



Hình 3.10 Giao diện tài liệu API của WebsiteTimE trên Swagger

# CHƯƠNG 4. QUẢN LÝ VÀ HIỆN THỰC HÓA DỰ ÁN

Chương này mô tả chi tiết quy trình quản lý và triển khai dự án WebsiteTimE theo phương pháp Scrum. Nhóm đã sử dụng các công cụ hiện đại như Jira để lập kế hoạch, Git/GitHub để quản lý mã nguồn và Pnpm Workspace để tổ chức dự án, nhằm đảm bảo tiến độ, chất lượng và sự phối hợp hiệu quả giữa các thành viên.

## Quản lý dự án với Jira

Jira được chọn làm công cụ trung tâm để quản lý toàn bộ vòng đời của dự án, từ việc xây dựng Product Backlog cho đến theo dõi tiến độ từng Sprint.

Khi bắt đầu, nhóm đã tạo một dự án Scrum trên Jira. Product Backlog được xây dựng bằng cách chuyển đổi các User Story (đã nêu ở Chương 3) thành các Issue trong Jira. Mỗi Issue được phân loại (Task, Story, Bug), gán Story Points để ước tính độ phức tạp, và giao cho thành viên chịu trách nhiệm thực hiện.

Dự án được chia thành nhiều Sprint, mỗi Sprint kéo dài từ 1 đến 2 tuần. Đầu mỗi Sprint, nhóm thực hiện buổi họp Sprint Planning để chọn các Issue quan trọng nhất từ Product Backlog và đưa vào Sprint Backlog. Trong suốt Sprint, tiến độ được theo dõi qua bảng Scrum Board (Kanban board) với các cột như To Do, In Progress, Done.

Bảng 4.1 Bảng kế hoạch thực hiện các Sprint

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **Mục tiêu chính** | **Các công việc** | **Người thực hiện** | | **Thời gian** |
| Sprint 1 | Xây dựng nền tảng Backend & xác thực | * Thiết lập dự án NestJS   + Cấu hình DB MongoDB   + Xây dựng API Đăng ký/Đăng nhập   + Tích hợp JWT | | Trần Tấn Hưng Lâm Nhật Hào Lê Tuấn Kha | Tuần 1 -  Tuần 2 |
| Sprint 2 | Xây dựng giao diện cơ bản & | - Thiết lập dự án Next.js | | Trần Tấn Hưng Lâm Nhật Hào | Tuần 3 -  Tuần 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | quản lý công việc | - Tạo layout chính  - Xây dựng giao diện CRUD cho Tasks  - Kết nối API Tasks | Lê Tuấn Kha |  |
| Sprint 3 | Hoàn thiện tính năng Dự án và Lịch biểu | * Xây dựng API & giao diện CRUD cho Projects * Xây dựng giao diện Lịch * Tích hợp hiển thị Tasks/TimeBloc   ks | Trần Tấn Hưng Lâm Nhật Hào Lê Tuấn Kha | Tuần 5 -  Tuần 6 |
| Sprint 4 | Tích hợp AI & Thống kê | * Xây dựng API & giao diện cho AI Chat, Gợi ý * Xây dựng trang Thống kê   - Tích hợp biểu đồ | Trần Tấn Hưng Lâm Nhật Hào Lê Tuấn Kha | Tuần 7 -  Tuần 8 |
| Sprint 5 | Triển khai Docker và hoàn thiện | - Viết Dockerfile, docker- compose.yml  - Tối ưu hóa UI/UX  - Sửa lỗi và kiểm thử toàn diện | Trần Tấn Hưng Lâm Nhật Hào Lê Tuấn Kha | Tuần 9 -  Tuần 10 |

## Quản lý mã nguồn với Git và GitHub

Toàn bộ mã nguồn của dự án được quản lý trên GitHub. Nhóm đã áp dụng quy trình làm việc chuyên nghiệp dựa trên feature branching:

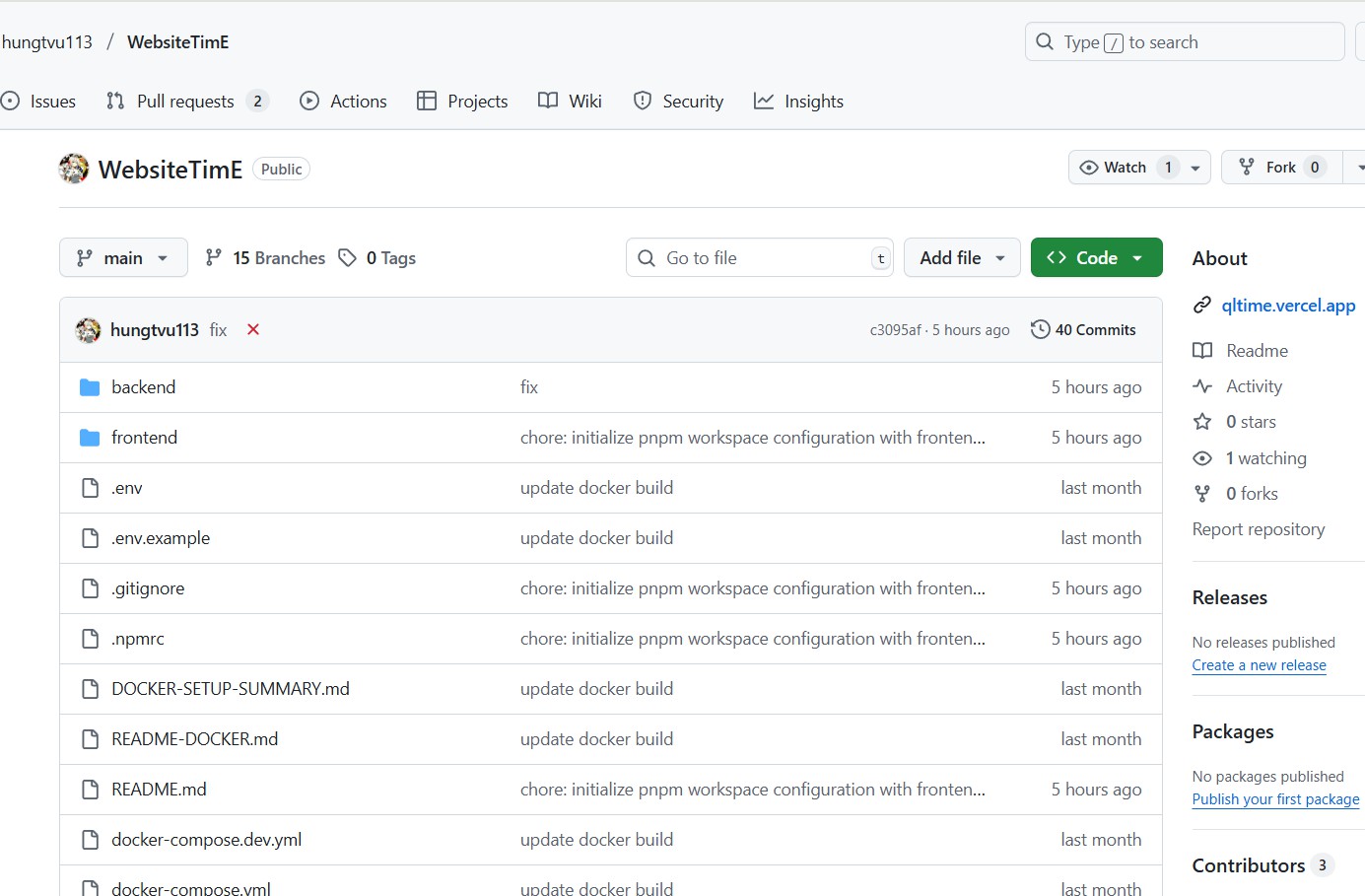
Main Branch: Luôn ở trạng thái ổn định, chỉ chứa mã nguồn đã được kiểm thử và sẵn sàng cho release.

Develop Branch: Nhánh phát triển chính, nơi tích hợp các tính năng đã hoàn

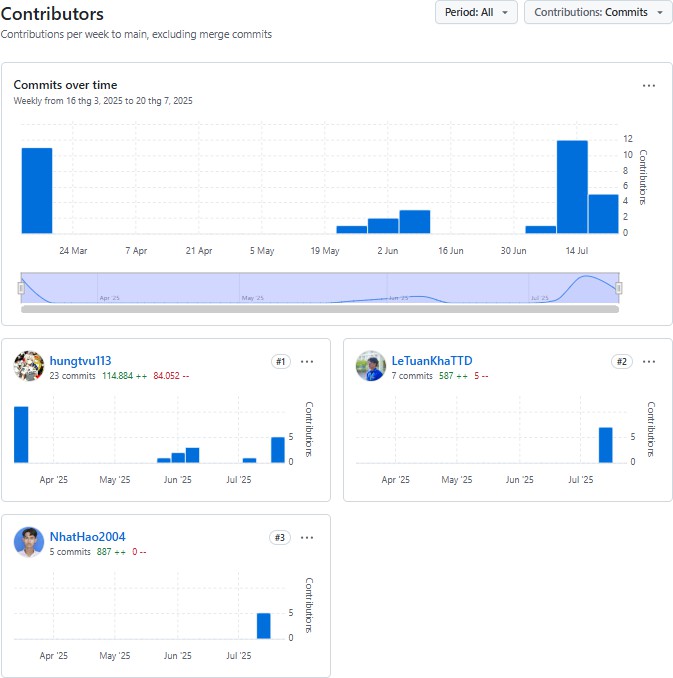
thành.

Feature Branches: Mỗi khi phát triển một tính năng mới (ví dụ: feature/login, feature/ai-chat), một nhánh mới sẽ được tạo ra từ develop.

Pull Request (PR): Sau khi hoàn thành tính năng, thành viên sẽ tạo một Pull Request để các thành viên khác review code trước khi hợp nhất (merge) vào nhánh develop. Quy trình này giúp đảm bảo chất lượng mã nguồn và giảm thiểu xung đột.



Hình 4.1 GitHub Repository của dự án



Hình 4.2 Lịch sử các Pull Request trên GitHub Repository của dự án

# CHƯƠNG 5. KIỂM THỬ VÀ TRIỂN KHAI

Sau khi hoàn thành các tính năng chính, giai đoạn kiểm thử và triển khai là bước cuối cùng và quan trọng để đảm bảo hệ thống WebsiteTimE hoạt động ổn định, đúng chức năng và sẵn sàng để sử dụng. Nhóm đã thực hiện kiểm thử API bằng Postman và đóng gói toàn bộ ứng dụng bằng Docker để đơn giản hóa việc triển khai.

## Kiểm thử hệ thống (Testing)

Kiểm thử là một phần không thể thiếu để đảm bảo chất lượng phần mềm. Trong phạm vi dự án, nhóm tập trung vào việc kiểm thử API phía Backend để xác minh tính đúng đắn của logic nghiệp vụ.

Postman được sử dụng làm công cụ chính để gửi các yêu cầu HTTP đến các API endpoint của Backend và kiểm tra kết quả trả về. Nhóm đã xây dựng một bộ sưu tập (Collection) các kịch bản kiểm thử cho những luồng chức năng quan trọng.

Kịch bản 1: Đăng ký và Đăng nhập

Gửi yêu cầu POST /api/auth/register với thông tin người dùng mới. Kiểm tra response trả về status 201 Created và có chứa token.

Dùng thông tin vừa đăng ký để gửi yêu cầu POST /api/auth/login. Kiểm tra response trả về status 200 OK và có chứa token.

Lưu lại token để sử dụng cho các yêu cầu cần xác thực. Kịch bản 2: Quản lý Công việc (CRUD)

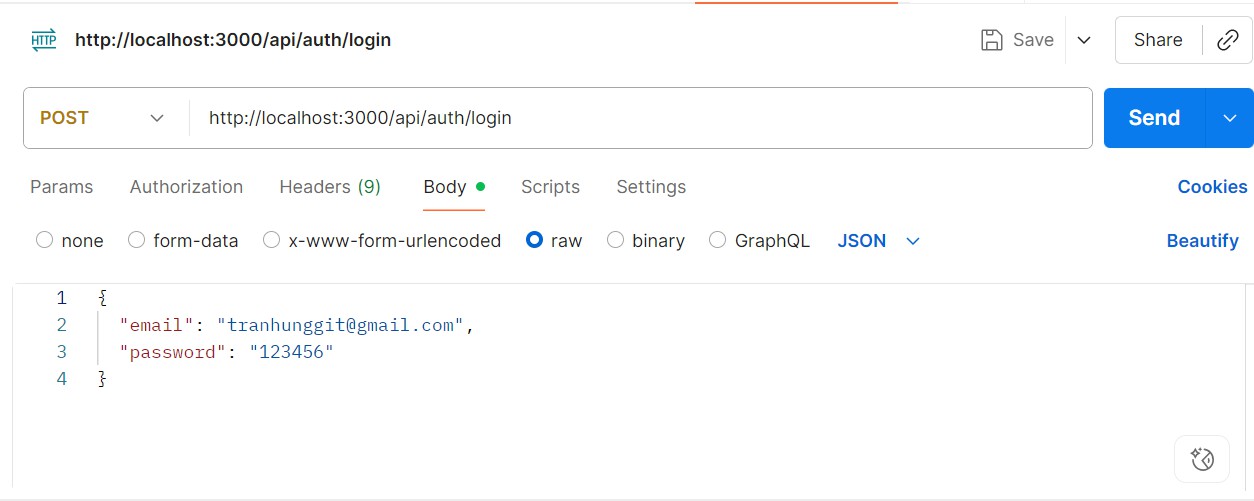
Tạo công việc: Gửi yêu cầu POST /api/tasks với token xác thực và dữ liệu công việc. Kiểm tra status 201 Created và dữ liệu công việc trả về là chính xác.

Lấy công việc: Gửi yêu cầu GET /api/tasks/:id với ID của công việc vừa tạo.

Kiểm tra status 200 OK và dữ liệu nhận được khớp với dữ liệu đã tạo.

Cập nhật công việc: Gửi yêu cầu PUT /api/tasks/:id để thay đổi tiêu đề. Kiểm tra status 200 OK và tiêu đề đã được cập nhật.

Xóa công việc: Gửi yêu cầu DELETE /api/tasks/:id. Kiểm tra status 200 OK. Gửi lại yêu cầu GET cho ID đó và kiểm tra status 404 Not Found.



Hình 5.1 Kết quả kiểm thử API tạo công việc mới bằng Postman

## Đóng gói và Triển khai (Deployment)

Để đơn giản hóa quá trình cài đặt và đảm bảo tính nhất quán của môi trường vận hành, nhóm đã sử dụng Docker để đóng gói toàn bộ ứng dụng.

Dự án được đóng gói thành 3 container chính, được quản lý bởi Docker Compose:

Container Frontend (frontend): Chứa ứng dụng Next.js. Container Backend (backend): Chứa ứng dụng NestJS API. Container Database (mongodb): Chứa cơ sở dữ liệu MongoDB. Cấu trúc Dockerfile:

Nhóm đã xây dựng hai file Dockerfile riêng biệt cho Frontend và Backend. Cả hai đều sử dụng multi-stage builds:

Stage 1 (Builder): Cài đặt toàn bộ dependencies (bao gồm cả devDependencies) và build mã nguồn ra phiên bản production.

Stage 2 (Production): Chỉ cài đặt production dependencies và sao chép các tệp đã được build từ stage 1. Cách làm này giúp giảm đáng kể kích thước của image cuối cùng, tối ưu cho việc triển khai.

Cấu trúc docker-compose.yml:

File docker-compose.yml định nghĩa 3 services (frontend, backend, mongodb), thiết lập các biến môi trường, ánh xạ cổng (port mapping) và định nghĩa network để các container có thể giao tiếp với nhau.

Backend service phụ thuộc (depends\_on) vào MongoDB service để đảm bảo cơ sở dữ liệu khởi động trước.

Frontend service phụ thuộc vào Backend service.

# CHƯƠNG 6. KẾT QUẢ, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

Sau một thời gian nghiên cứu, thiết kế và phát triển, dự án "WebsiteTimE - Hệ thống Quản lý Thời gian Thông minh" đã hoàn thành các mục tiêu cốt lõi đề ra. Chương này sẽ tổng kết các kết quả đạt được, đưa ra những đánh giá khách quan về dự án và đề xuất các hướng phát triển trong tương lai.

## Kết quả đạt được

Sản phẩm cuối cùng là một ứng dụng web full-stack hoạt động ổn định, cung cấp đầy đủ các chức năng quản lý thời gian và được tích hợp trợ lý AI thông minh.

Hệ thống đã hiện thực hóa thành công các nhóm chức năng chính:

Quản lý Công việc và Dự án: Người dùng có thể tạo, sửa, xóa, và theo dõi công việc một cách trực quan qua danh sách hoặc bảng Scrum trong từng dự án.

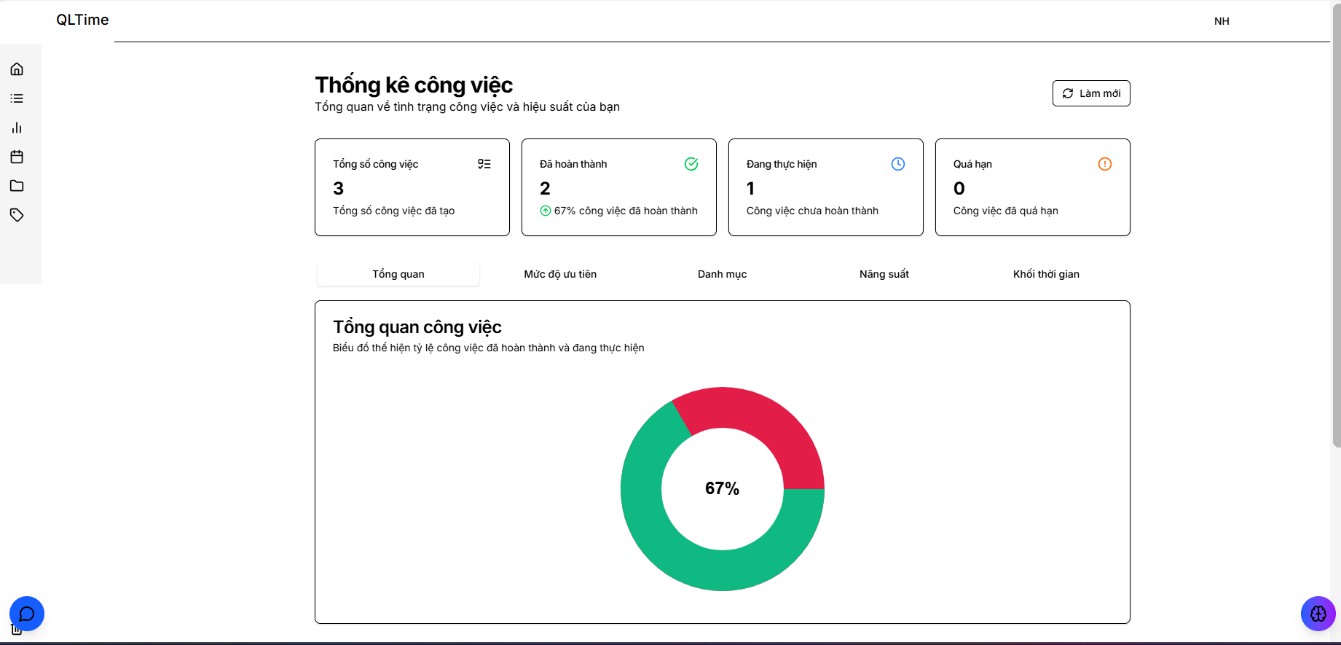
Lịch biểu và Time Blocking: Giao diện lịch cho phép người dùng xem công việc theo ngày/tuần, đồng thời tạo các khối thời gian (time blocks) để lên kế hoạch chi tiết.

Tích hợp AI - Dr.AITime: Trợ lý AI đã được tích hợp thành công với các tính năng: gợi ý độ ưu tiên/ngày hết hạn khi tạo task và cung cấp giao diện chat để tương tác.



Hình 6.1 Tính năng AI gợi ý thông minh khi tạo công việc mới

Thống kê và Báo cáo: Trang thống kê cung cấp các biểu đồ trực quan về tiến độ công việc, phân loại theo mức độ ưu tiên và danh mục, giúp người dùng nắm bắt hiệu suất của mình.



Hình 6.2 Giao diện trang Thống kê với các biểu đồ trực quan

## Hướng phát triển

Dự án WebsiteTimE có tiềm năng lớn để tiếp tục phát triển và hoàn thiện.

Dưới đây là một số hướng đi trong tương lai: Mở rộng tính năng AI:

AI tự động lập lịch: Phân tích danh sách công việc và tự động đề xuất các khối thời gian (time blocks) tối ưu trên lịch.

Phân tích năng suất sâu hơn: AI có thể phân tích thói quen làm việc của người dùng và đưa ra các báo cáo, gợi ý cá nhân hóa để cải thiện hiệu suất.

Phát triển ứng dụng di động: Xây dựng phiên bản Mobile App (sử dụng React Native hoặc Flutter) để người dùng có thể quản lý công việc mọi lúc, mọi nơi.

Tích hợp tính năng làm việc nhóm: Cho phép người dùng tạo không gian làm việc (workspace), mời thành viên, giao việc và cùng nhau theo dõi tiến độ dự án.

Hoàn thiện CI/CD và Kiểm thử tự động: Xây dựng một quy trình Tích hợp và Triển khai liên tục (CI/CD) hoàn chỉnh với GitHub Actions, đồng thời bổ sung bộ unit test và e2e test để đảm bảo chất lượng phần mềm một cách tự động.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Atlassian, "What is a backlog?," Atlassian, 17 05 2022. [Online]. Available: [https://www.atlassian.com/agile/project-management/backlogs.](http://www.atlassian.com/agile/project-management/backlogs) [Accessed 23  02 2025]. |
| [2] | N. Eyal, "The Psychology Behind Time Management," NirAndFar, 10 06 2021. [Online]. Available: htt[ps://www.nirandfar.com/psychology-of-time-](http://www.nirandfar.com/psychology-of-time-) management/. [Accessed 11 03 2025]. |
| [3] | MongoDB, "Introduction to MongoDB," MongoDB Inc., 14 08 2023. [Online]. Available: https://[www.mongodb.com/docs/manual/introduction/.](http://www.mongodb.com/docs/manual/introduction/) [Accessed 21 03 2025]. |
| [4] | Figma, "Figma Design Systems," Figma, 01 12 2023. [Online]. Available: [https://www.figma.com/design-systems/.](http://www.figma.com/design-systems/) [Accessed 5 02 2025 ]. |