

Project Proposal



Bộ môn công nghệ phần mềm
Khoa công nghệ thông tin
Đại học khoa học tự nhiên TP HCM

MỤC LỤC

Các nội dung chính	1
1 Bảng đánh giá thành viên	2
2 Phát biểu bài toán sơ lược	3
3 Giải pháp đề xuất	4
4 Kế hoạch phát triển	5
5 Kế hoạch nhân sự và chi phí	6
6 Thiết lập công cụ	7






Project Proposal

Các nội dung chính

Mục tiêu tài liệu tập trung vào các chủ đề:

- ✓ Tạo ra tài liệu Project Proposal.
- ✓ Hoàn chỉnh tài liệu Project Proposal với các nội dung:
 - Phát biểu bài toán sơ lược
 - Giải pháp đề xuất
 - Kế hoạch phát triển
 - Kế hoạch nhân sự & chi phí
- ✓ Đọc hiểu tài liệu Project Proposal.

1 Bảng đánh giá sinh viên

ID	Name	Contribution (%)	Signature
23120200	Nguyễn Hưng Thịnh	100%	
23120222	Lê Thành Công	100%	
23120232	Lê Thượng Đế	100%	
23120386	Phan Khắc Trường	100%	
23120400	Cao Quốc Tý	100%	

2 Phát biểu bài toán sơ lược

Trong thời đại **trí tuệ nhân tạo (AI)** phát triển mạnh mẽ, đặc biệt là các mô hình **tạo ảnh từ văn bản** như **Stable Diffusion**, **DALL·E**, và **Midjourney**, nhu cầu sáng tạo hình ảnh dựa trên mô tả ngôn ngữ ngày càng trở nên phổ biến. Tuy nhiên, hầu hết các nền tảng hiện nay chỉ cung cấp **chức năng tạo ảnh cơ bản**, thiếu môi trường tương tác hoặc không hỗ trợ các **thao tác chỉnh sửa hình ảnh nâng cao**. Người dùng thông thường thường gặp khó khăn khi muốn **tách đối tượng, chỉnh sửa chi tiết, thay đổi khuôn mặt, hoặc thêm yếu tố mới** vào hình ảnh mà không cần sử dụng các công cụ thiết kế đồ họa phức tạp.

Dựa trên thực tế đó, nhóm của chúng em đề xuất phát triển một **nền tảng web dành cho tạo và chỉnh sửa hình ảnh bằng AI**, giúp người dùng **tạo và tinh chỉnh hình ảnh một cách dễ dàng, nhanh chóng và mang tính cá nhân hóa cao**.

Dự án hiện tại tập trung vào hai chức năng chính:

- **Text-to-Image Generation:**
Người dùng nhập mô tả (prompt), và hệ thống sẽ **tạo ra hình ảnh tương ứng** bằng AI.
- **AI Image Editing:**
Người dùng có thể thực hiện các thao tác như **tách đối tượng, chỉnh sửa khuôn mặt, thêm chi tiết**, hoặc **chuyển đổi phong cách** bằng cách cung cấp thêm prompt.

Hệ thống sẽ áp dụng mô hình token-based để giới hạn lượt sử dụng hằng ngày, đảm bảo hệ thống hoạt động ổn định:

- **Người dùng thường:** khoảng 100–200 tokens/ngày, tùy theo chi phí dịch vụ AI.
- **Người dùng premium:** 500–1000 tokens/ngày, không tích lũy

Mỗi thao tác tạo hoặc chỉnh sửa hình ảnh sẽ **tiêu tốn một lượng token nhất định**, tùy theo **độ phức tạp của yêu cầu và chi phí xử lý AI**.

Trong các phiên bản tương lai, nền tảng có thể mở rộng thêm **tính năng xã hội (social features)** như **chia sẻ hình ảnh, bình luận, thả cảm xúc, hoặc tái sử dụng hình ảnh của người dùng khác**.

Môi trường hoạt động

Frontend:

- Xây dựng bằng **React + Vite**, sử dụng **JavaScript (ES6)** kết hợp với **TailwindCSS** và **CSS thuần** cho giao diện người dùng.
- Chạy trực tiếp trên các **trình duyệt hỗ trợ HTML5**.

Backend:

Hệ thống được chia thành hai backend độc lập để tối ưu khả năng mở rộng và bảo trì:

1. AI Backend:

- Xây dựng bằng **Django (Python)**.
- Chịu trách nhiệm **tương tác với API của các dịch vụ AI bên ngoài** (hoặc nguồn AI miễn phí nếu có).
- Cung cấp **internal API** cho **Social Backend** để yêu cầu tạo hoặc chỉnh sửa hình ảnh.

2. Social Backend:

- Là cầu nối giữa người dùng và AI Backend.
- Xử lý **xác thực người dùng (authentication)**, **quản lý token**, **kiểm soát truy cập**, **lưu trữ prompt và hình ảnh**, cùng các **tính năng xã hội**.
- Dự kiến phát triển bằng **Go** hoặc **Java** để tận dụng hiệu năng cao và khả năng **xử lý song song (parallel processing)**.

Database:

- Cơ sở dữ liệu chính: **PostgreSQL**
- Tùy chọn sử dụng thêm **MongoDB**, **Redis**, hoặc **Neo4j** cho các kịch bản triển khai linh hoạt.
- Trong giai đoạn phát triển ban đầu, sử dụng **SQLite**.

Môi trường triển khai:

- Chạy trên máy chủ **Linux (Ubuntu)**.
- Triển khai thông qua **Nginx + Gunicorn** (cho Django) và **Go/Java runtime environment**.

Tích hợp AI:

- Các mô hình AI được tích hợp thông qua **external API** như **Stable Diffusion API**, **HuggingFace**, hoặc **OpenAI API**.
- Hệ thống hỗ trợ **thay thế linh hoạt** khi thay đổi nhà cung cấp AI.

Thiết kế & Ràng buộc triển khai

Ngôn ngữ lập trình:

- **Frontend:** JavaScript (ES6)
- **AI Backend:** Python (Django)
- **Social Backend:** Go (Gin) / Java (Springboot)
- **UI Framework:** React (Vite), TailwindCSS
- **Database:** PostgreSQL / SQLite

Quy chuẩn mã nguồn:

- **Python:** tuân theo PEP8
- **JavaScript:** tuân theo ESLint
- **Go/Java:** theo **chuẩn coding convention** của ngôn ngữ
- **Tài liệu:** viết bằng **Markdown**, lưu trong **docs/** hoặc xuất ra **PDF**

Ràng buộc tài nguyên:

- **Chi phí sử dụng AI có thể thay đổi tùy theo dịch vụ, do đó lượng token hằng ngày cho từng loại người dùng có thể được điều chỉnh linh hoạt.**

3 Giải pháp đề xuất

3.1 Phần mềm

3.1.1. Danh sách các chức năng

Nhu cầu	Yêu cầu
Là người dùng, tôi muốn có thể nhập mô tả (prompt) và nhận lại hình ảnh được AI sinh ra.	Sinh ảnh từ mô tả văn bản (Text → Image)
Là người dùng, tôi muốn có thể xóa hoặc thay đổi phông nền của ảnh.	Xóa / thay phông nền (Background Remove / Replace)
Là người dùng, tôi muốn thay đổi phong cách ảnh (chuyển phong cách ảnh thành anime, tranh sơn dầu...).	Đổi phong cách ảnh (Style Transfer)
Là người dùng, tôi muốn có thể thay đổi khuôn mặt.	Thay đổi gương mặt (Face Swap)
Là người dùng, tôi muốn có thể thay đổi độ tuổi trong ảnh.	Thay đổi tuổi (Face Swap)

Là người dùng mới, tôi muốn đăng nhập nhanh bằng tài khoản Google, Facebook để không tốn thời gian đăng ký.	Đăng nhập đa nền tảng (Social Login)
Là người dùng quốc tế, tôi muốn trang web hiển thị bằng ngôn ngữ của tôi (ví dụ: tiếng Anh, tiếng Việt) để dễ dàng sử dụng.	Hỗ trợ đa ngôn ngữ
Là người dùng, tôi muốn tải ảnh đã xử lý về máy với nhiều định dạng	Chức năng xuất (Export)
Là người dùng, tôi muốn có thể trả phí (qua VietQR, PayPal).	Giao diện thanh toán
Là người dùng, tôi muốn xem lại tất cả các ảnh tôi đã tạo và quản lý hồ sơ cá nhân của mình.	Trang cá nhân, lịch sử đã tạo
Là người dùng, tôi muốn chia sẻ những tấm hình thú vị được tạo ra bởi AI, và chia sẻ cách tạo ra ảnh đó. Bạn bè cũng có thể tương tác bằng cách thích (like) và bình luận (comment) vào bài đăng.	Chia sẻ và tương tác trên một nền tảng chung
Là người dùng, tôi muốn biến phong cách ảnh của mình thành các phong cách độc đáo, thú vị (anime, cartoon, ...).	Hệ thống chuyển đổi ảnh sang nhiều phong cách khác nhau
Là người dùng, tôi không biết cách gõ prompt, tôi cần được hỗ trợ.	Hệ thống hỗ trợ tạo prompt để sinh ảnh tự động.
Là người dùng, tôi có prompt nhưng chưa rõ ý, hãy giúp tôi chỉnh sửa.	Hệ thống gợi ý và chỉnh sửa prompt theo ý người dùng.

Là người dùng, tôi muốn sinh ảnh theo trí tưởng tượng của mình.	Hệ thống sinh ảnh theo mô tả tưởng tượng của người dùng.
Là người dùng, tôi muốn hòa mình vào các khung cảnh mà tôi mong muốn.	Hệ thống ghép người dùng vào khung cảnh mong muốn.
Là người dùng, tôi muốn phục hồi bức ảnh bị hư hỏng, mờ, ô vàng hoặc rách.	Hệ thống phục hồi ảnh bị mờ, rách hoặc ô vàng.

3.1.2. Kiến trúc phần mềm

3.1.2.1 Tổng quan hệ thống

Hệ thống được xây dựng theo **kiến trúc microservices**, chia thành nhiều nhóm dịch vụ độc lập, mỗi nhóm phụ trách một chức năng chính.

Mục tiêu của kiến trúc là:

- Đảm bảo khả năng **mở rộng độc lập (scalability)** cho từng mảng (AI, social, frontend).
- Tăng tính **linh hoạt trong phát triển** giữa các team.
- Giảm thiểu ảnh hưởng khi triển khai, nâng cấp hoặc sửa lỗi.

Hệ thống bao gồm ba nhóm chính:

Nhóm	Công nghệ chính	Chức năng chính
Frontend Team	React / Vite / Tailwind	Xây dựng giao diện web, gọi API Gateway, chịu trách nhiệm tăng trải nghiệm của người dùng
Backend AI Team	Python / Django / Docker	Định dạng lại đầu vào và đầu ra cho API AI
Backend Social Team	Spring Boot + Gin	Quản lý người dùng, bài đăng, tương tác, thông báo, bảo mật, API Gateway

3.1.2.2 Kiến trúc chi tiết Backend Social

- Mục tiêu:

Backend Social chịu trách nhiệm xử lý toàn bộ các chức năng mạng xã hội của ứng dụng, bao gồm quản lý người dùng, bài viết, bình luận, lượt thích, và kết nối giữa người dùng.

- Công nghệ sử dụng:

Hệ thống áp dụng mô hình lai giữa Spring Boot và Gin:

+ Spring Boot: Quản lý các service nền tảng như Authentication, User Service, Notification, và API Gateway.

+ Gin: Xử lý các service có yêu cầu hiệu năng cao như Post Service, Comment Service, Feed Service, và Follow Service.

- Sơ đồ tổng quan (mô tả logic)

API Gateway → Auth Service (Spring Boot) → User Service (Spring Boot) → Feed/Post/Comment/Follow Services (Gin) → Database Layer

↳ Các service giao tiếp với nhau qua REST, Message queue để đảm bảo hiệu năng và tính mở rộng.

- Các thành phần chính

- + API Gateway: Định tuyến và xác thực các request, giúp che giấu kiến trúc nội bộ.
- + Auth Service: Xác thực người dùng, phát hành JWT Token, và quản lý session.
- + User Service: Lưu trữ thông tin hồ sơ, trạng thái người dùng.
- + Post Service: Cho phép người dùng đăng bài viết, ảnh, và chia sẻ nội dung.
- + Comment Service: Quản lý các bình luận trên bài viết.
- + Follow Service: Quản lý quan hệ theo dõi giữa người dùng.
- + Feed Service: Tổng hợp nội dung từ những người mà người dùng theo dõi, tối ưu truy vấn bằng cache Redis.
- + Notification Service: Gửi thông báo theo thời gian thực khi có tương tác.

Trong tương lai phát triển có thể có thêm các service hỗ trợ

- Cơ chế giao tiếp nội bộ

Các service trong Backend Social giao tiếp thông qua hai cơ chế:

- + RESTful API cho các thao tác cơ bản giữa các service.
- + Message queue (Kafka/RabbitMQ) cho các tác vụ bất đồng bộ như gửi thông báo, cập nhật feed, ghi log hoạt động.

- Cơ sở dữ liệu và lưu trữ

- + PostgreSQL được dùng cho dữ liệu quan hệ như người dùng, bài viết, bình luận.
- + Redis được dùng làm cache cho Feed và Session.
- + MongoDB (hoặc MinIO) được dùng để lưu trữ metadata của ảnh và nội dung động.
- + Neo4j được dùng tận dụng tìm ra điểm chung hay mối quan hệ giữa các user (optional).

- Bảo mật và quản lý truy cập

- + Xác thực bằng JWT, refresh token quản lý bởi Auth Service.
- + Sử dụng HTTPS/TLS để mã hóa truyền thông.
- + Áp dụng Role-Based Access Control (RBAC) để phân quyền người dùng.

- **Lợi ích của kiến trúc Microservices**

- + Dễ mở rộng từng phần độc lập.
- + Cho phép nhóm làm việc song song và triển khai riêng biệt.
- + Tối ưu hiệu năng với ngôn ngữ phù hợp từng service.
- + Giảm rủi ro khi nâng cấp hoặc thay đổi một module cụ thể.

3.1.2.2 Kiến trúc chi tiết Backend AI

- **Mục tiêu:**

Backend AI chịu trách nhiệm xử lý toàn bộ các tác vụ liên quan AI trong hệ thống, bao gồm:

- + Nhận và xử lý prompt từ người dùng.
- + Chuẩn hóa hoặc tối ưu prompt (refine prompt).
- + Sinh ảnh hoặc chỉnh sửa ảnh dựa trên prompt thông qua mô hình AI.
- + Giao tiếp với Backend Social để lưu trữ metadata hội thoại, ảnh và kết quả sinh.

Mục tiêu của backend AI là tách riêng phần xử lý mô hình ra khỏi hệ thống mạng xã hội, giúp dễ mở rộng, thay thế model hoặc tối ưu hiệu năng mà không ảnh hưởng tới các nhóm khác.

Tương lai sẽ có thể fine-tuning lại model để sử dụng tốt hơn nếu có đủ thời gian

- **Công nghệ sử dụng:**

Ngôn ngữ & Framework: Python, Django(cung cấp RESTful API, dễ mở rộng và nhẹ).
Triển khai: Docker + Gunicorn/Uvicorn để container hóa và tối ưu hiệu suất.

Mô hình AI:

+ API của Gemini (Refine Prompt): Chuẩn hóa, mở rộng hoặc diễn đạt lại prompt để đạt chất lượng sinh ảnh tốt hơn.

+ API của Gemini Banana (Image Generation / Editing): Sinh ảnh, chỉnh ảnh, hoặc thực hiện các tác vụ AI khác theo prompt đầu vào.

+ Tương lai sẽ có thêm một số API hỗ trợ mạnh khác

Task queue: Celery + Redis. Hỗ trợ xử lý bất đồng bộ, queue sinh ảnh nặng.

- Sơ đồ tổng quan (mô tả logic)

AI Gateway → Refine Prompt Service (Gemini API)



AI Image Generation Service (Gemini Banana / Model khác)



AI respond handler



(Metadata gửi về Backend Social)

- Các thành phần chính

+ AI Gateway: Định tuyến và điều phối các yêu cầu AI từ hệ thống chính, thống nhất dữ liệu request/response và che giấu kiến trúc nội bộ.

+ Refine Prompt Service: Sử dụng Gemini API để chuẩn hóa và tối ưu prompt người dùng, giúp tăng chất lượng đầu vào cho quá trình sinh ảnh.

+ AI Image Generation Service: Gọi Gemini Banana API để sinh ảnh từ prompt đã được tối ưu, xử lý và trả kết quả về cho Gateway.

+ AI Response Handler: Tổng hợp kết quả ảnh, xử lý hậu kỳ output (chèn watermark, nén ảnh), sinh metadata và gửi về Backend Social để lưu trữ.

- Cơ chế giao tiếp nội bộ

Backend AI → Backend Social: Giao tiếp qua REST/Message queue để đồng bộ thông tin hội thoại, metadata ảnh.

Nội bộ Backend AI: Các module có thể gọi nhau nội bộ hoặc qua event queue nếu xử lý bất đồng bộ (Celery + Redis).

- **Bảo mật và quản lý truy cập**
+ Sử dụng HTTPS/TLS để mã hóa (prompt, ảnh).

3.2 Phần cứng

Hạng mục	Mô tả chi tiết	Ghi chú
Máy chủ Backend AI	<ul style="list-style-type: none"> - CPU: 4 nhân trở lên (Intel i5 / Xeon Silver / AMD Ryzen 5 trở lên) - RAM: 16GB trở lên - GPU: Chưa cần (nếu chưa cần phải tự tinh chỉnh model) - Ổ cứng SSD \geq 200GB trở lên 	Dùng để chạy và gọi các mô hình sinh ảnh (Stable Diffusion, SDXL, v.v.)
Máy chủ Backend Social(Spring Boot + Gin)	<ul style="list-style-type: none"> - CPU: 4 nhân trở lên - RAM: 8–16GB - Ổ cứng SSD: \geq 200GB 	Đảm nhận xử lý logic mạng xã hội, API Gateway, Auth,...
Máy chủ Frontend / Reverse Proxy	<ul style="list-style-type: none"> - CPU: 2 nhân - RAM: 4GB - Ổ cứng SSD: \geq 100GB - Có thể sử dụng dịch vụ đám mây 	Dùng để chạy Nginx, React/Vite build, hoặc làm proxy điều phối

Máy tính lập trình viên (Development)	<ul style="list-style-type: none"> - CPU: Intel i5 hoặc tương đương - RAM: 16GB - Ổ cứng SSD: $\geq 512\text{GB}$ 	Dùng cho các thành viên nhóm phát triển, chạy môi trường local
--	---	--

4 Kế hoạch phát triển

4.1 Phân tích Yêu cầu (Requirements Analysis)

Mục tiêu: Xác định và làm rõ tất cả các yêu cầu chức năng và phi chức năng của hệ thống.

Công việc chính:

- Thu thập và chi tiết hóa các nhu cầu người dùng (User Stories).
- Phân tích và định nghĩa rõ ràng phạm vi của dự án, tập trung vào các chức năng cốt lõi (Text-to-Image, AI Editing) và các tính năng xã hội.
- Xác định các yêu cầu phi chức năng: bảo mật (xác thực, JWT) , hiệu năng (xử lý song song) , và hệ thống token.
- Nghiên cứu và lựa chọn các nhà cung cấp External AI API (Stable Diffusion, OpenAI, Gemini).

Kết quả:

- Tài liệu Đặc tả Yêu cầu Phần mềm (Software Requirements Specification - SRS).
- Tài liệu phân tích và so sánh các AI API.

4.2 Thiết kế phần mềm

Mục tiêu: Xây dựng bản thiết kế kỹ thuật chi tiết cho toàn bộ hệ thống dựa trên kiến trúc microservices.

Công việc chính:

- **Thiết kế kiến trúc:** Hoàn thiện sơ đồ kiến trúc tổng thể , định nghĩa rõ ràng ranh giới và cách giao tiếp (Internal API) giữa Frontend, Backend Social, và Backend AI.

- **Thiết kế Database:** Thiết kế lược đồ quan hệ (ERD) cho PostgreSQL (quản lý user, post, comment) và xem xét cấu trúc cho MongoDB/Redis (metadata, cache).
- **Thiết kế API:** Đặc tả chi tiết API (OpenAPI/Swagger) cho cả ba thành phần:
 - + API Gateway (phía BE Social) cho Frontend gọi.
 - + API nội bộ giữa BE Social và BE AI.
- **Thiết kế giao diện (UI/UX):** Xây dựng wireframes và mockups cho các màn hình chính (trang chủ, trang tạo ảnh, trang cá nhân, feed).

Kết quả:

- Tài liệu Thiết kế Kiến trúc (Software Architecture Design - SAD).
- Lược đồ cơ sở dữ liệu chi tiết.
- Tài liệu đặc tả API.
- Bộ thiết kế UI/UX (Figma hoặc tương đương).

4.3 Triển khai (Implementation)

Giai đoạn này được chia thành 3 Sprint chính, mỗi Sprint tập trung vào một nhóm chức năng cụ thể.

Sprint 1: Xây dựng lõi hệ thống và Xác thực

- **Mục tiêu:** Thiết lập môi trường, xây dựng nền tảng cho cả 3 service và hoàn thiện chức năng xác thực, quản lý người dùng.
- **Frontend (React/Vite):**
 - + Thiết lập dự án, cấu trúc thư mục, TailwindCSS.
 - + Xây dựng các trang: Đăng nhập, Đăng ký, Quên mật khẩu.
 - + Tích hợp Social Login (Google, Facebook).
 - + Xây dựng trang Quản lý hồ sơ cá nhân (User Profile).
- **Backend Social (Go/Java):**
 - + Thiết lập dự án (Spring Boot + Gin), kết nối Database (PostgreSQL/SQLite).
 - + Hoàn thiện Auth Service: cấp JWT token, Refresh token, xác thực Social Login.
 - + Hoàn thiện User Service: CRUD thông tin người dùng.
 - + Xây dựng hệ thống quản lý Token (API) cơ bản.
- **Backend AI (Python/Django):**
 - + Thiết lập dự án, tạo API Gateway nội bộ.

- + Xây dựng service wrapper cho 1 chức năng AI cơ bản (ví dụ: Text-to-Image) sử dụng Gemini Banana.
- **Kết quả Sprint 1:** Người dùng có thể đăng ký, đăng nhập (thường và social), xem/cập nhật thông tin cá nhân. Hệ thống API Gateway và kết nối cơ bản giữa 3 service được thông suốt.

Sprint 2: Hoàn thiện tính năng AI và Mạng xã hội cơ bản

- **Mục tiêu:** Triển khai tất cả các tính năng xử lý AI cốt lõi và các tính năng xã hội cơ bản (đăng bài, xem feed).
- **Frontend (React/Vite):**
 - + Xây dựng giao diện tạo ảnh: Nhập prompt, upload ảnh.
 - + Xây dựng giao diện chỉnh sửa ảnh: Các công cụ (xóa nền, đổi mặt...).
 - + Hiển thị kết quả ảnh, cho phép tải về (Export).
 - + Xây dựng trang Feed (xem bài đăng) và trang Lịch sử tạo ảnh.
- **Backend Social (Go/Java):**
 - + Hoàn thiện Post Service (đăng bài, ảnh).
 - + Hoàn thiện Feed Service (lấy danh sách bài đăng).
 - + Tích hợp API gọi sang BE AI để tạo/chỉnh sửa ảnh và lưu kết quả (metadata).
 - + Trừ token người dùng sau mỗi lần gọi AI thành công.
- **Backend AI (Python/Django):**
 - + Hoàn thiện tất cả các service AI còn lại: Refine Prompt (Gemini) , Background Remove, Style Transfer, Face Swap, Image Restoration.
 - + Tối ưu hóa đầu vào/đầu ra API.
- **Kết quả Sprint 2:** Người dùng có thể sử dụng đầy đủ các tính năng tạo và chỉnh sửa ảnh AI. Ảnh sau khi tạo có thể được đăng lên "feed" chung và xem lại trong lịch sử.

Sprint 3: Hoàn thiện tính năng Nâng cao và Đa ngôn ngữ

- **Mục tiêu:** Hoàn thiện các tính năng xã hội nâng cao, thanh toán và các tính năng phụ trợ.
- **Frontend (React/Vite):**
 - + Triển khai hệ thống tương tác: Like, Comment.
 - + Xây dựng giao diện Thông báo
 - + Tích hợp đa ngôn ngữ (Tiếng Anh, Tiếng Việt,...).
 - + Xây dựng giao diện thanh toán (nâng cấp Premium).
- **Backend Social (Go/Java):**

- + Hoàn thiện Comment Service , Follow Service , Notification Service.
- + Tích hợp thanh toán (VietQR, PayPal).
- + Tích hợp Redis cache cho Feed Service và Session.
- **Backend AI (Python/Django):**
 - + Tích hợp Celery + Redis để xử lý các tác vụ AI nặng (image generation) một cách bất đồng bộ.
- **Kết quả Sprint 3:** Hệ thống hoàn chỉnh với đầy đủ tính năng xã hội (theo dõi, thả tim, bình luận), hệ thống thanh toán và xử lý AI bất đồng bộ, cải thiện trải nghiệm người dùng.

4.4 *Kiểm thử (Testing)*

- **Mục tiêu:** Đảm bảo chất lượng phần mềm, sửa lỗi và xác minh hệ thống đáp ứng đúng yêu cầu.
- **Công việc chính:**
 - + **Unit Test:** Lập trình viên tự thực hiện Unit Test cho các hàm/module mình viết (như kế hoạch nhân sự).
 - + **Integration Test:** Kiểm tra sự giao tiếp giữa các microservices:
 - FE ↔ BE Social (API Gateway).
 - BE Social ↔ BE AI (Internal API).
 - BE Social/AI ↔ Databases (PostgreSQL, Redis).
 - + **Functional Test:** Kiểm thử toàn bộ các User Stories (Đăng nhập, tạo ảnh, xóa nền, đăng bài, bình luận...).
 - + **Performance Test:** Kiểm tra thời gian phản hồi của API, đặc biệt là các API AI và Feed.
 - + **Security Test:** Kiểm tra các lỗ hổng bảo mật cơ bản (OWASP), phân quyền (RBAC).
- **Kết quả:**
 - + Tài liệu Kế hoạch Kiểm thử (Test Plan) và Bộ Ca kiểm thử (Test Cases).
 - + Báo cáo Kết quả Kiểm thử (Test Report) và danh sách lỗi (bugs) đã được xử lý.

4.5 *Triển khai và bảo trì (Deployment and Maintenance)*

- **Mục tiêu:** Đưa ứng dụng vào hoạt động trên môi trường production và duy trì hoạt động ổn định.
- **Công việc chính:**
 - + **Deployment:**

- Cấu hình máy chủ Linux (Ubuntu).
- Cài đặt Nginx làm Reverse Proxy.
- Triển khai các service Backend (Django + Gunicorn, Go/Java runtime).
- Cấu hình cơ sở dữ liệu PostgreSQL production.
- Thiết lập CI/CD (GitHub Actions) để tự động hóa quy trình build và deploy.
- + **Maintenance:**
 - Theo dõi (monitoring) log hệ thống và hiệu suất máy chủ.
 - Sao lưu (backup) cơ sở dữ liệu định kỳ.
 - Tiếp nhận phản hồi người dùng và sửa các lỗi phát sinh.
 - Cập nhật và điều chỉnh lượng token/ngày dựa trên chi phí AI thực tế.
- **Kết quả :**
 - + Hệ thống chạy ổn định trên môi trường production.
 - + Tài liệu hướng dẫn triển khai và vận hành.
 - + Kế hoạch sao lưu và bảo trì.

5 Kế hoạch nhân sự và chi phí

5.1 Kế hoạch nhân sự

STT	Chức năng	Đảm nhiệm giai đoạn đầu	Đảm nhiệm giai đoạn sau	Ngôn ngữ + công cụ	Mô tả sơ bộ
1	Project Manager	Công	Công		Quản lý tiến độ, phân công, giám sát toàn bộ dự án
2	Frontend	Tý	Trưởng, Đế, Tý (captain)	ReactJS, Tailwind CSS, MUI, Shaden ...	Thực hiện tiếp nhận ý kiến và nhu cầu người dùng triển khai thiết kế giao diện, ban đầu Tý làm chính về

					sau sẽ được Đế và Trường hỗ trợ
3	Backend Social	Thịnh, Công (captain)	Thịnh, Công (captain)	Go / Java	Thực hiện viết backend cho các phần cốt lõi và thiết kế mạng xã hội, messa
4	Backend AI	Trường, Đế (captain)		Python (Django)	Viết lại module AI thành service riêng, chuẩn hóa đầu vào/đầu ra để các module khác dễ tích hợp. Bởi vì chỉ cần chuẩn hóa lại đầu vào đầu ra nên về sau sẽ hỗ trợ Tý làm giao diện
5	Database	Thịnh	Công, Thịnh (captain)	PostgreSQL, MongoDB, Neo4j	Thực hiện quản lý toàn bộ database, chịu trách nhiệm thiết kế và lưu trữ khi thao tác với database
6	BA (Business Analyst)	Công, Đế			Tiếp nhận nhu cầu người dùng và triển khai cho team nhỏ cũng như toàn bộ thành viên trong nhóm
7	Tester	Tất cả			Mỗi thành viên sẽ tự test các chức năng do các hàm mình đảm nhiệm viết

5.2 Chi phí nhân sự

STT	Họ và tên	Số giờ	(VNĐ/giờ)	Thành tiền
1	Project Manager	30	40.000	1.200.000
1.1	Lê Thành Công	30	40.000	1.200.000
2	FE	100	35.000	3.500.000
2.1	Cao Quốc Tỷ	60	35.000	2.100.000
2.2	Lê Thượng Đế	20	35.000	700.000
2.3	Phan Khắc Trường	20	35.000	700.000
3	Social BE	100	35.000	3.500.000
3.1	Lê Thành Công	50	35.000	1.750.000
3.2	Nguyễn Hưng Thịnh	50	35.000	1.750.000
4	AI BE	70	35.000	2.100.000
4.1	Lê Thượng Đế	35	35.000	1.225.000
4.2	Phan Khắc Trường	35	35.000	1.225.000
5	Database	30	35.000	1.050.000
5.1	Lê Thành Công	15	35.000	525.000
5.2	Nguyễn Hưng Thịnh	15	35.000	525.000
6	Business Analyst	30	35.000	1.050.000

6.1	Lê Thành Công	15	35.000	525.000
6.2	Lê Thượng Đế	15	35.000	525.000
7	Tester	40	30.000	1.200.000
7.1	Cao Quốc Tỷ	8	30.000	240.000
7.2	Lê Thành Công	8	30.000	240.000
7.3	Nguyễn Hưng Thịnh	8	30.000	240.000
7.4	Lê Thượng Đế	8	30.000	240.000
7.5	Phan Khắc Trường	8	30.000	240.000

Tổng chi phí nhân sự = 13.600.000 VNĐ

5.3 Chi phí Tool + thiết bị

Tool

STT	Hạng mục	Chi tiết	Chi phí ước tính (VNĐ)	Ghi chú
1	Phần mềm/ Công cụ phát triển	ReactJS, Visual Studio Code,	0	Mã nguồn mở.
2	Quản lý dự án và cộng tác	GitHub, Jira, Trello, Zalo, ..	0	Sử dụng bản miễn phí, gói giáo dục.
3	Cơ sở dữ liệu	PostgreSQL	0	Mã nguồn mở.
4	Dịch vụ AI	Tin dụng gọi API (Gemini, OpenAI, ...)	300.000	Ưu tiên các nguồn miễn phí nếu có thể.
5	Tên miền		200.000	

6	Dự phòng		1.000.000	Các chi phí không lường trước.
---	----------	--	-----------	--------------------------------

Tổng chi phí Tool: 1.500.000 VNĐ

Server

STT	Hạng mục	Chi tiết	Chi phí ước tính (VNĐ)	Ghi chú
1	CPU	CPU: 4 nhân trở lên (Intel i5 / Xeon Silver / AMD Ryzen 5 trở lên)	5.000.000	Cần iGPU để debug vì ko xài GPU rời.
2	RAM	16 GB DDR4	1.500.000	Chọn 2×8GB chạy dual-channel
3	Ổ cứng	512 GB SSD	1.000.000	Ưu tiên NVMe M.2 Gen3 x4 để tốc độ cao.
4	Hệ điều hành	Linux	0	Ubuntu LTS 24.04 để được hỗ trợ lâu dài
5	Nguồn, Mainboard	Phù hợp với CPU	2.000.000	Nguồn khoảng 500W, Mainboard nên có 4 khe ram hỗ trợ lâu dài trong tương lai.

6	Case, Quạt tản nhiệt	Case cỡ nhỏ + 2 - 3 quạt 120mm.	700.000	Tạo luồng khí vào ra, giảm nhiệt, tăng tuổi thọ cho server
---	----------------------	------------------------------------	---------	--

Tổng chi phí server: 10.200.000 VNĐ

5.4 Tổng chi phí:

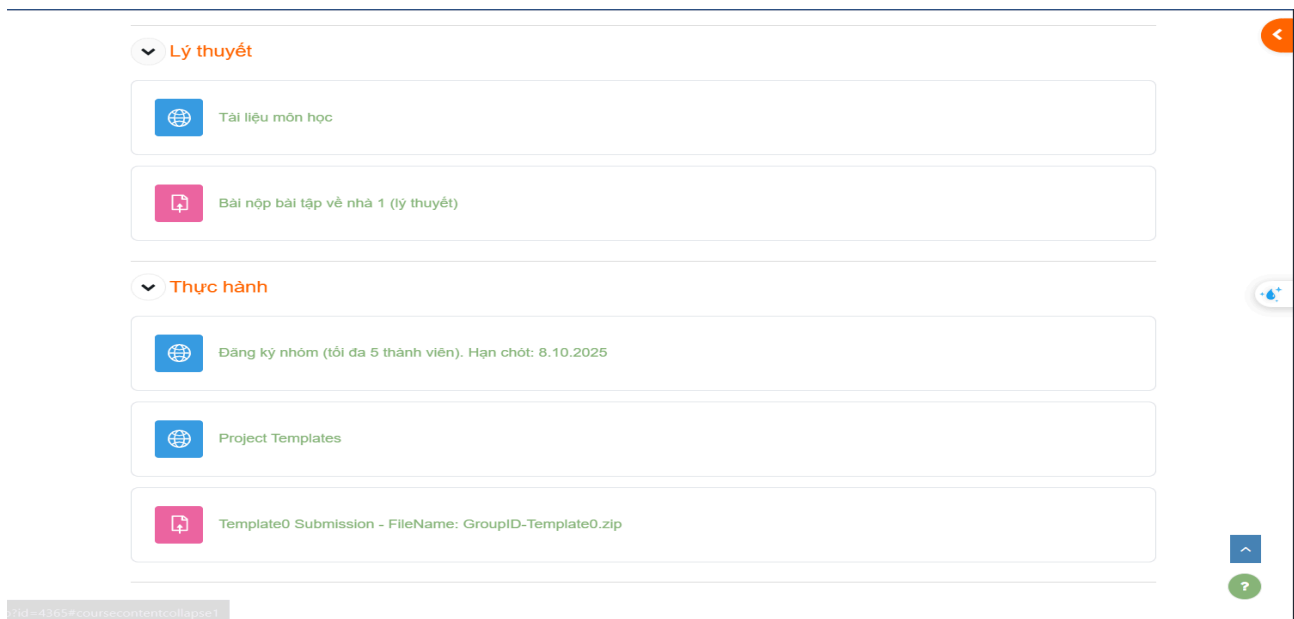
- 25.300.300 VNĐ
- Chưa kể chi phí bảo trì và điện.

6 Thiết lập công cụ

Để đảm bảo việc phối hợp hiệu quả và quản lý tiến độ đồ án, nhóm đã lựa chọn và thiết lập các công cụ hỗ trợ như sau:

1. Moodle

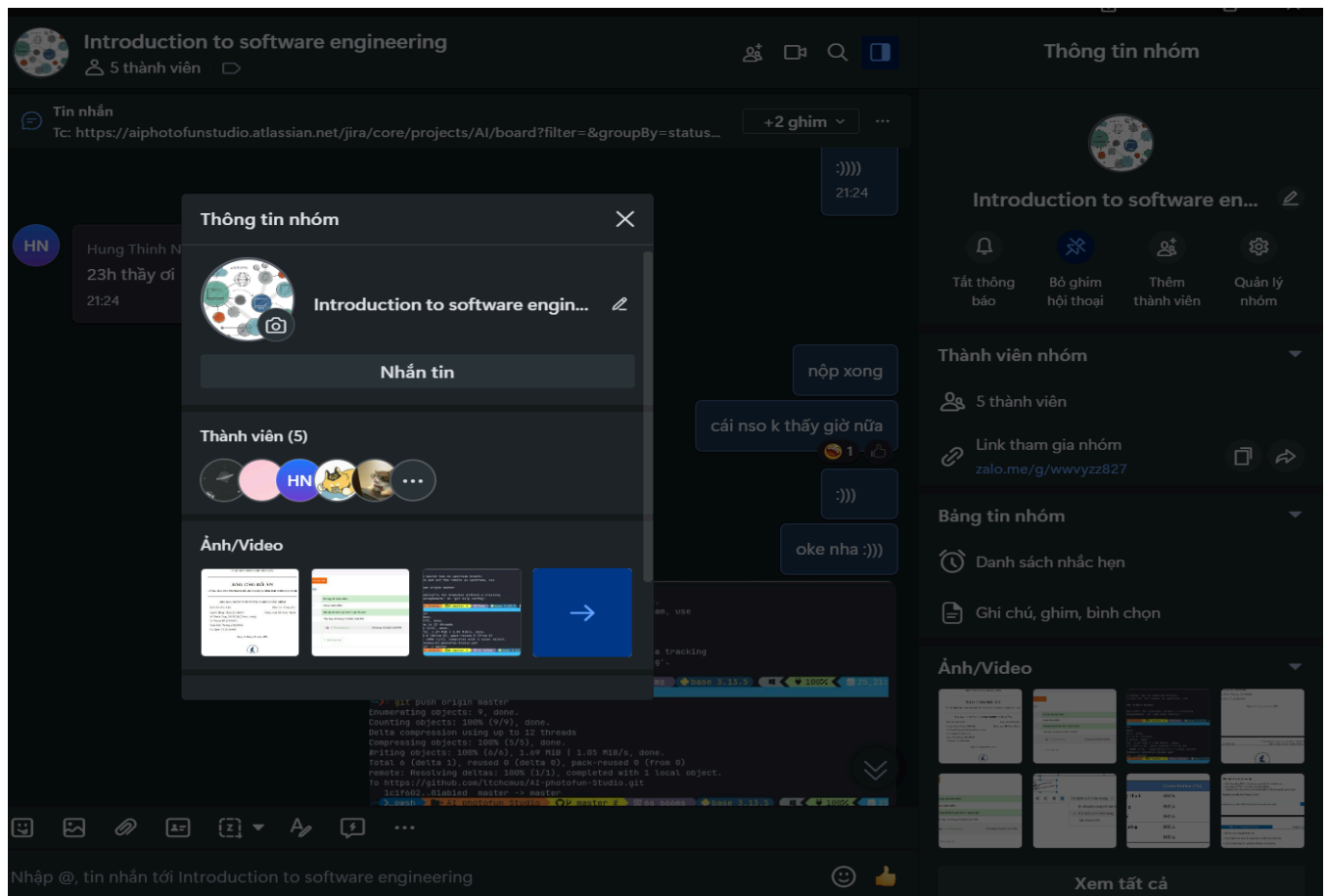
Nhóm sử dụng hệ thống Moodle của trường để theo dõi thông báo, nộp bài tập, và tải các tài liệu hướng dẫn từ giảng viên và trợ giảng. Mọi thành viên đều thường xuyên kiểm tra Moodle để đảm bảo cập nhật kịp thời các yêu cầu và deadline của đồ án.



2. Zalo (dùng để trao đổi và cập nhật tiến độ nhóm)

Nhóm lựa chọn **Zalo** làm công cụ chính để **trao đổi thông tin, thảo luận và cập nhật tiến độ làm việc**.

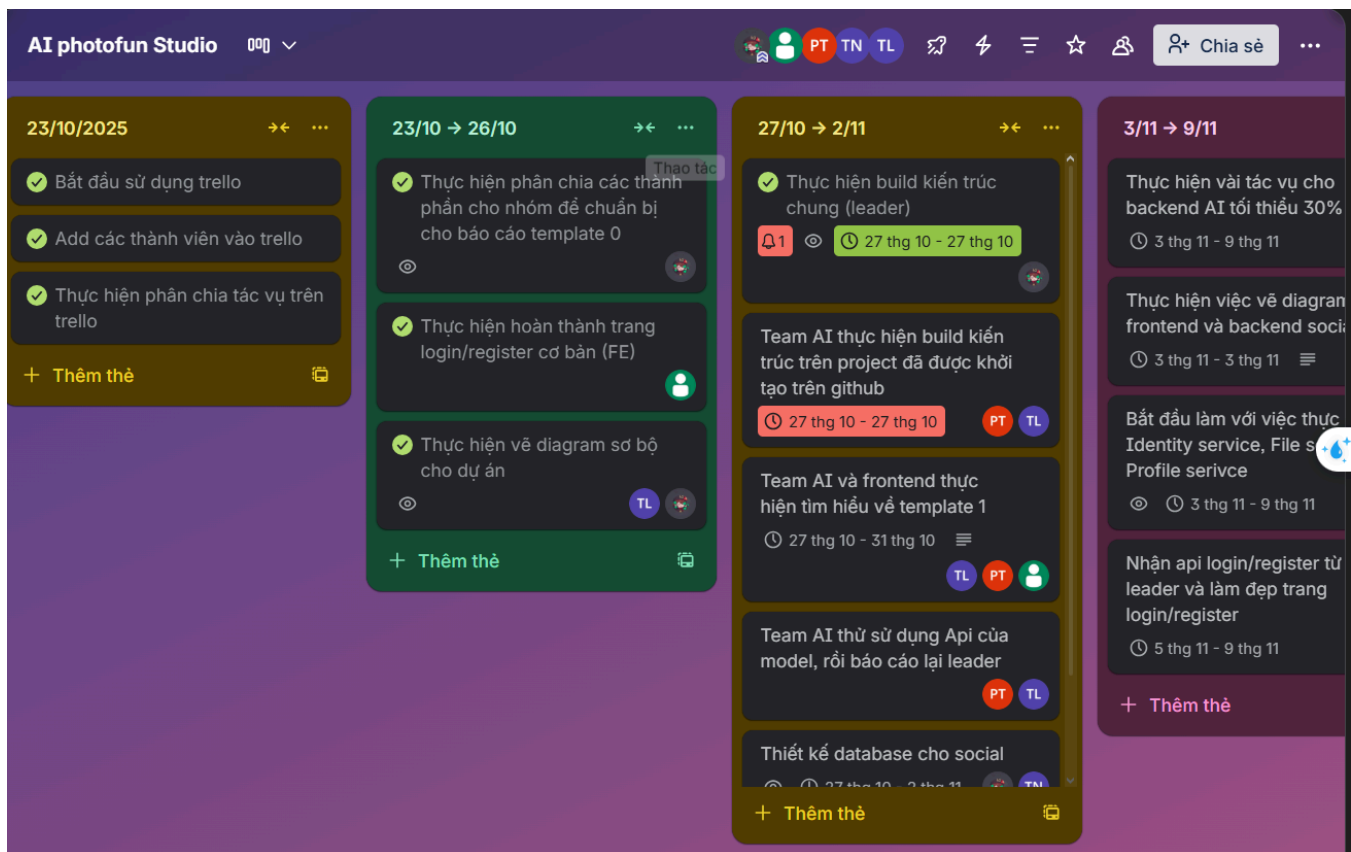
- Nhóm Zalo được tạo riêng cho các thành viên trong nhóm để thảo luận các vấn đề kỹ thuật, phân chia công việc và báo cáo tiến độ hằng ngày.
- Các thành viên bật thông báo Zalo để đảm bảo không bỏ lỡ thông tin quan trọng.
- Nhóm cũng thống nhất đăng tải các thông tin cập nhật từ **Trello** (về task và tiến độ) trực tiếp trong nhóm Zalo để mọi người dễ theo dõi.

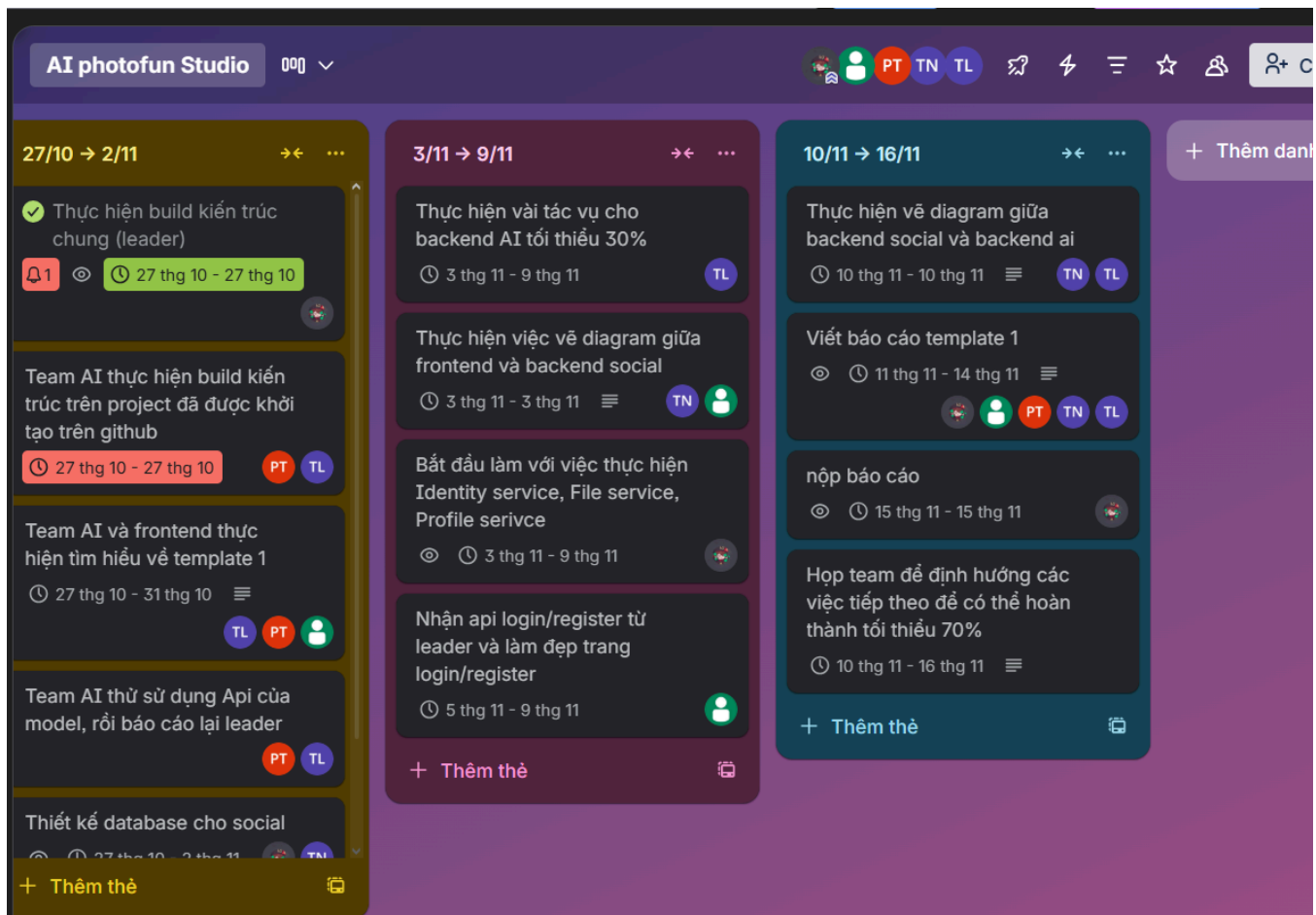


3. Trello

Nhóm sử dụng **Trello** để quản lý công việc và theo dõi tiến độ dự án.

- Các task được chia nhỏ theo sprint (theo mô hình Agile).
- Mỗi thẻ (card) trên Trello thể hiện một nhiệm vụ cụ thể, có người chịu trách nhiệm, thời hạn và trạng thái (To Do – Doing – Done).
- Mỗi khi có cập nhật, nhóm trưởng sẽ thông báo lại trong nhóm Zalo để đồng bộ thông tin.





Link trello: [Trello](#)

4. GitHub

Nhóm sử dụng **GitHub** làm hệ thống quản lý mã nguồn (source control).

Cấu trúc repository gồm:

/src → Mã nguồn chương trình

/docs → Tài liệu đề án

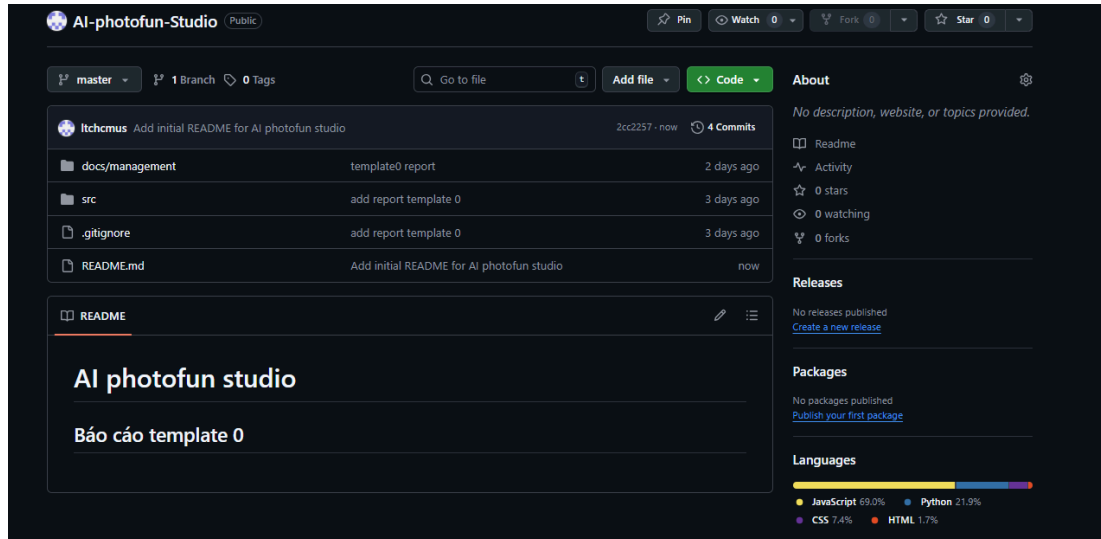
+ management → Báo cáo tuần, kế hoạch, biên bản họp

+ requirements → Tài liệu yêu cầu, use case, vision document

+ analysis_design → Mô hình UML, kiến trúc phần mềm, thiết kế giao diện

+ test → Test plan, test case, test report

/pa → Các bài nộp (PA1, PA2, ...)



Tất cả các thành viên được phân quyền push/pull để đảm bảo cập nhật liên tục. Khi thực hiện một task nào đó sẽ pull và checkout sang branch khác để thực hiện khi hoàn thành sẽ push lên và yêu cầu merge vào master và sẽ được leader kiểm duyệt

Link repository: [AI photofun Studio](#)

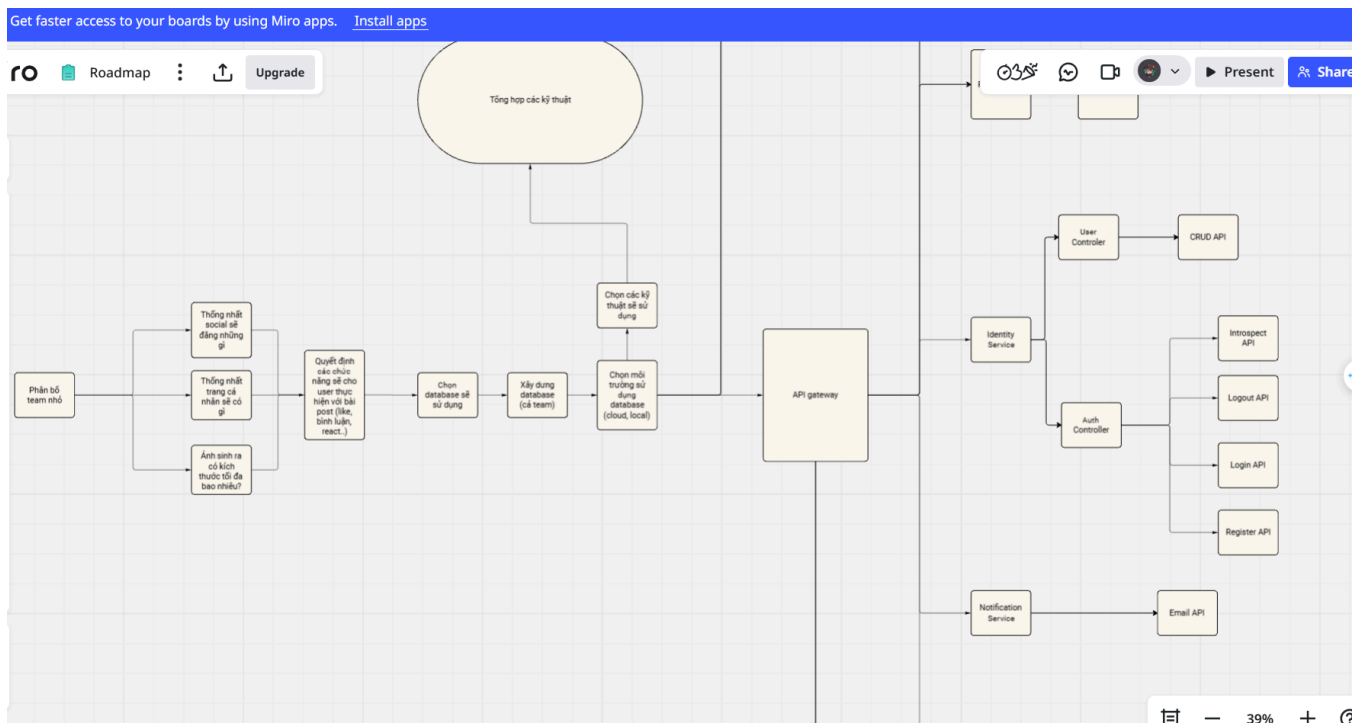
5. JIRA (hoặc Azure Boards – dự phòng)

Nếu Trello không đáp ứng đủ nhu cầu quản lý chi tiết, nhóm dự kiến chuyển sang **JIRA** để theo dõi backlog, sprint, và phân tích hiệu suất làm việc.

6. Miro

Sử dụng để có thể vẽ mô hình và diagram cho team thực hiện theo các tác vụ được ưu tiên và sẽ để thành viên vẽ thêm các chức năng đề xuất, giúp đồ án có một cái nhìn tổng quan hơn

Một phần hình ảnh đã được leader vẽ:



Link để có thể theo dõi: [Miro](#)

7. Tổng kết

Việc sử dụng kết hợp các công cụ trên giúp:

- Quản lý công việc rõ ràng, phân chia nhiệm vụ minh bạch.
- Dễ dàng trao đổi và hỗ trợ lẫn nhau qua Zalo.
- Đảm bảo lưu trữ, kiểm soát và cập nhật mã nguồn an toàn trên GitHub.
- Theo dõi tiến độ và báo cáo minh bạch qua Trello/JIRA.

Nhóm cam kết sử dụng các công cụ trên một cách thường xuyên và hiệu quả để đảm bảo tiến độ và chất lượng của đồ án.