**ĐẠI HỌC PHENIKAA**

**KHOA HỆ THỐNG THÔNG TIN**

**⸎⸎⸎⸎⸎**

A logo for a university

AI-generated content may be incorrect.

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN PHÂN TÍCH VÀ THIẾT KẾ PHẦN MỀM**

**Đề tài: Xây dựng phần mềm Heart Beat trên nền tảng Mobile**

**Nhóm thực hành:** Nhóm 8

**Thành viên:** Hà Mạnh Long – 23010390

Bùi Anh Quốc – 23010328

**Giảng viên hướng dẫn:** Vũ Quang Dũng

Nguyễn Xuân Quế

**Lớp tín chỉ:** PT&TKPM(N05)

**Hà Nội, Tháng 10/2025**

# **Thông tin**

## **- Thành viên:**

|  |  |  |
| --- | --- | --- |
| Thành viên | Hà Mạnh Long | Bùi Anh Quốc |
| Mã sinh viên | 23010390 | 23010328 |
| Vai trò | - Thiết kế dự án (70%)  - Lập trình triển khai (30%) | - Thiết kế dự án (30%)  - Lập trình triển khai (70%) |

## **- Github:**

[*https://github.com/AnhQuocs/PTTKPM25-26\_ClassN05\_Nhom\_9*](https://github.com/AnhQuocs/PTTKPM25-26_ClassN05_Nhom_9)

## - **Link thiết kế giao diện Figma:**

[*https://www.figma.com/design/nlb29AHZX4q6bgxKT03d6j/Heart-Beat-Figma?node-id=0-1&m=dev&t=zltqMDcrCJK8UzzN-1*](https://www.figma.com/design/nlb29AHZX4q6bgxKT03d6j/Heart-Beat-Figma?node-id=0-1&m=dev&t=zltqMDcrCJK8UzzN-1)*.*

**Mục lục**

[**Thông tin** 1](#_Toc211875992)

[**- Thành viên:** 1](#_Toc211875993)

[**- Github:** 1](#_Toc211875994)

[- **Link thiết kế giao diện Figma:** 1](#_Toc211875995)

[**Công việc** 3](#_Toc211875996)

[**Phần 1: Giới thiệu & Lập kế hoạch:** 4](#_Toc211875997)

[**I. Giới thiệu chung:** 4](#_Toc211875998)

[**II. Lập kế hoạch:** 4](#_Toc211875999)

[**Phần 2: Phân tích yêu cầu:** 5](#_Toc211876000)

[**I. Actors (Vai trò):** 5](#_Toc211876001)

[**II. Tổng quan hệ thống:** 6](#_Toc211876002)

[**III. Yêu cầu chức năng:** 7](#_Toc211876003)

[**IV. Yêu cầu phi chức năng:** 7](#_Toc211876004)

[**V. Mô hình hóa Use Case & kịch bản:** 7](#_Toc211876005)

[**Phần 3: Thiết kế hệ thống:** 13](#_Toc211876006)

[**I. Thiết kế kiến trúc hệ thống:** 13](#_Toc211876007)

[**II. Thiết kế sơ đồ lớp & Tạo cơ sở code:** 15](#_Toc211876008)

[**III. Thiết kế tương tác:** 18](#_Toc211876009)

[**IV. Thiết kế hành vi và trạng thái:** 26](#_Toc211876010)

[**V. Áp dụng mẫu thiết kế:** 29](#_Toc211876011)

[**Phần 4: Triển khai hệ thống:** 34](#_Toc211876012)

[**I. Công nghệ sử dụng:** 34](#_Toc211876013)

[**Phần 5: Tổng kết & Hướng phát triển:** 36](#_Toc211876014)

[**I. Đánh giá những điểm đã làm được và chưa làm được:** 36](#_Toc211876015)

[**II. Hướng phát triển tiếp theo của sản phẩm:** 37](#_Toc211876016)

[**Lời cảm ơn** 39](#_Toc211876017)

# **Công việc**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nội dung | Quốc | Long |
| Tuần 1 | Phân tích yêu cầu | - Phân tích các Actors.  - Mô tả các yêu cầu chức năng/ phi chức năng. | - Viết tài liệu mô tả các yêu cầu chức năng, hệ thống và danh sách Actor/Use Case. |
| Tuần 2 | UseCase & Activity | - Tổng hợp thông tin Actors, Use Case.  - Kiểm tra các logic hoạt động. | - Vẽ Biểu đồ Use Case, Biểu đồ hoạt động.  - Tài liệu kịch bản cho các Use Case chính. |
| Tuần 3 | Kiến trúc hệ thống | - Xây dựng và triển khai code theo cấu trúc hệ thống  Clean Architecture. | - Mô tả mối quan hệ giữa các tầng của hệ thống.  - Vẽ Biểu đồ gói. |
| Tuần 4 | Thiết kế Lớp | - Xác định các thuộc tính, phương thức của đối tượng. | - Vẽ Biểu đồ lớp.  - Mô tả các mối quan hệ của các đối tượng. |
| Tuần 5 | Thiết kế tương tác & Giao diện | - Lên ý tưởng và lập trình cho giao diện. | - Vẽ biểu đồ trình tự.  - Thiết kế giao diện (Figma). |
| Tuần 6 | Hành vi & Trạng thái | - Mô phỏng các luồng trạng thái, logic trong code. | - Vẽ biểu đồ trạng thái, phân tích luồng trạng thái. |
| Tuần 7 | Design Patterns | - Lập trình triển khai áp dụng các mẫu thiết kế vào trong code. | - Phân tích và vẽ biểu đồ lớp cho các mẫu thiết kế |
| Tuần 8 | Lập trình chức năng lõi | - Lập trình các chức năng chính, quan trọng. | - Lập trình các chức năng cơ bản. |
| Tuần 9 | Lập trình giao diện & tích hợp | - Lập trình tích hợp giao diện. | - Thiết kế, hoàn thiện phắc thảo giao diện người dùng. |
| Tuần 10 | Báo cáo | - Kiểm tra toàn bộ mã nguồn | - Viết tài liệu báo cáo cuối kỳ |
|  | Tổng kết | 50% | 50% |

# **Phần 1: Giới thiệu & Lập kế hoạch:**

## **I. Giới thiệu chung:**

Trong bối cảnh nhu cầu chăm sóc sức khỏe cộng đồng ngày càng tăng cao, việc ứng dụng công nghệ số để quản lý và kết nối các hoạt động y tế trở thành xu hướng tất yếu. Nhóm chúng em định hướng phát triển một ứng dụng chăm sóc sức khỏe tổng thể trong tương lai, hỗ trợ người dùng theo dõi và quản lý nhiều hoạt động liên quan đến sức khỏe.

Trong giai đoạn hiện tại, nhóm lựa chọn chức năng hiến máu nhân đạo làm phần khởi đầu. Đây là hoạt động mang tính nhân văn sâu sắc, góp phần cứu sống hàng triệu sinh mạng mỗi năm, nhưng công tác tổ chức và quản lý vẫn còn nhiều hạn chế.

Nhận thấy nhu cầu thực tế này, nhóm chúng em đã quyết định thực hiện đề tài: “***Xây dựng phần mềm Heart Beat trên nền tảng Mobile***”.

## **II. Lập kế hoạch:**

**1. Mục tiêu dự án:**

* Xây dựng ứng dụng hiến máu giúp kết nối người hiến máu – bệnh viện – ban tổ chức sự kiện.
* Hỗ trợ người hiến máu trong các chức năng chính:
  + Quản lý sức khỏe cá nhân.
  + Đăng ký vào các hoạt động hiến máu.
  + Quản lý lịch hẹn và lịch sử hiến máu.
  + Nhận thông báo về các sự kiện hiến máu.
* Cung cấp công cụ cho nhân viên y tế và quản trị viên để quản lý sự kiện, xác nhận việc đăng ký hiến máu, thống kê và lưu trữ dữ liệu về sức khỏe.

**2. Phạm vi dự án:**

* Đối tượng phục vụ: Cộng đồng
* Nền tảng triển khai: Ứng dụng di động (mobile-first).
* Giới hạn chức năng: Trong giai đoạn đầu dự án tập trung vào việc phát triển một ứng dụng mà mọi người có thể đăng ký hiến máu và quản lý dữ liệu về sức khỏe.

**3. Mô hình quy trình Agile (Scrum):**

* Lý do lựa chọn:
  + Linh hoạt: Dễ dàng điều chỉnh chức năng theo phản hồi thực tế.
  + Chia nhỏ công việc: Các sprint ngắn giúp kiểm soát được tiến độ và chất lượng công việc.
  + Phù hợp học tập: Tiện cho việc phát triển theo tuần, gắn với yêu cầu môn học.
* Các bước chính trong Agile/Scrum:
  + Lập kế hoạch: Xác định backlog các chức năng chính của dự án.
  + Thực hiện Sprint: Phát triển từng phần nhỏ (Usecase, Class, Sequence,...).
  + Kiểm thử: Đảm báo các tính đúng đắn và ổn định sau mỗi sprint.
  + Đánh giá & cải tiến: Tiếp nhận các phản hồi và phát triển tối ưu cho dự án.

# **Phần 2: Phân tích yêu cầu:**

## **I. Actors (Vai trò):**

**1. Người hiến máu (Donor):**

* Đăng nhập/ Đăng ký (Login/Register).
* Đăng ký/ Hủy đăng ký hiến máu (Register/Cancel Blood Donation).
* Xem lịch hẹn hiến máu (View Appointment).
* Xem lịch sử hiến máu (View Donation History).

**2. Nhân viên y tế (Staff):**

* Đăng nhập/ Đăng ký (có mã code riêng do QTV cấp).
* Quản lý lịch hiến máu (Manage Donation Schedule).
* Xác nhận đăng ký hiến máu(Confirm Donation Registration).
* Ghi nhận kết quả hiến máu (Record Results).

**3. Quản trị viên (Admin):**

* Đăng nhập (Login).
* Quản lý nhân sự (cấp mã code riêng cho NVYT).
* Quản lý người hiến máu (Manage Donors).
* Quản lý sự kiện hiến máu (Manage Donation Events).
* Quản lý địa điểm hiến máu (Manage Locations).

**4. Hệ thống (System):**

* Kiểm tra điều hiện hiến máu (độ tuổi, sức khỏe, khoảng cách ngày hiến máu trước đó,...).
* Gửi thông báo nhắc nhở lịch hẹn/ địa điểm hiến máu.
* Lưu trữ và backup dữ liệu.

## **II. Tổng quan hệ thống:**

**1. Mục tiêu hệ thống:**

* Kết nối người hiến máu, nhân viên y tế và ban tổ chức.
* Hỗ trợ quy trình đăng ký, quản lý, nhắc nhở và lưu trữ lịch sử.

**2. Phạm vi hệ thống:**

* Đối tượng phục vụ: người hiến máu, nhân viên y tế, quản trị viên.
* Nền tảng: Android/ Web/ Mobile-first.

**3. Các rằng buộc:**

* Cần Internet.
* Phải tuân thủ quy định y tế về bảo mật thông tin cá nhân.

**4. Giả định và phụ thuộc:**

* Người dùng có thiết bị kết nối Internet.
* Người hiến máu cung cấp thông tin chính xác khi đăng ký.

## **III. Yêu cầu chức năng:**

* Hệ thống cho phép người hiến máu tạo tài khoản và đăng ký hiến máu trong các sự kiện còn chỗ.
* Người hiến máu có thể hủy đăng ký trong thời gian cho phép.
* Nhân viên y tế có thể quản lý danh sách đăng ký, xác nhận và cập nhật trạng thái hiến máu.
* Quản trị viên có thể quản lý thông tin sự kiện, địa điểm và người hiến máu, nhân viên y tế.
* Hệ thống gửi thông báo nhắc nhở trước lịch hiến máu ít nhất 1 ngày.
* Hệ thống cho phép người hiến máu tra cứu lịch sử hiến máu của bản thân.

## **IV. Yêu cầu phi chức năng:**

* **Bảo mật**: Bảo vệ thông tin cá nhân và y tế, xác thực đăng nhập an toàn.
* **Hiệu năng**: Thời gian phản hồi tra cứu sự kiện ≤ 2 giây.
* **Sẵn sàng**: Hệ thống hoạt động ổn định, có thể phục vụ ≥ 500 người dùng cùng lúc.
* **Khả năng mở rộng**: Có thể mở rộng số lượng sự kiện và địa điểm dễ dàng.
* **Dễ dùng**: Giao diện rõ ràng, hỗ trợ mobile-first.
* **Bảo toàn dữ liệu**: Sao lưu định kỳ và khôi phục nhanh khi có sự cố.
* **Khả năng truy cập**: Hỗ trợ đa ngôn ngữ (Việt – Anh).

## **V. Mô hình hóa Use Case & kịch bản:**

**1. Biểu đồ Use Case:**

A diagram of a system

AI-generated content may be incorrect.

**2. Biểu đồ hoạt động:**

* Luồng hoạt động của Admin:

A diagram of a server

AI-generated content may be incorrect.

* Luồng hoạt động của nhân viên y tế (Staff):

A diagram of a flowchart

AI-generated content may be incorrect.

* Luồng hoạt động của người hiến máu (Donor):

**A diagram of a flowchart

AI-generated content may be incorrect.**

**3. Kịch bản Use Case chi tiết:**

1. **UseCase: Đăng ký hiến máu:**

* Actor chính: Người hiến máu.
* Mục tiêu: Người hiến máu đăng ký tham gia một đợt hiến máu thành công.
* Tiền điều kiện: Người hiến máu đã có tài khoản hoặc thực hiện đăng nhập.
* Luồng chính/ Thành công:

- Người hiến máu mở ứng dụng và chọn chức năng Đăng ký hiến máu.

- Ứng dụng yêu cầu nhập thông tin cá nhân (họ tên, ngày sinh, số CMND/CCCD, nhóm máu, chọn địa điểm, thời gian,...).

- Hệ thống sẽ kiểm tra điều kiện hiến máu (tuổi, tình trạng sức khỏe, khoảng cách lần hiến máu trước,...).

- Nếu hợp lệ, hệ thống lưu thông tin đăng ký.

- Hệ thống hiện thị thông báo: “Đăng ký hiến máu thành công”.

- Hệ thống gửi thông báo nhắc nhở trước lịch hẹn.

* Trường hợp đặc biệt/ Đăng ký không thành công:

- Người hiến máu không đủ điều kiện (Ví dụ: chưa đủ tuổi, khoảng cách hiến máu lần trước quá gần,...)

- Địa điểm/Thời gian hiến máu đã đầy chỗ

1. **UseCase: Xác nhận đăng ký hiến máu:**

* Actor chính: Nhân viên y tế.
* Mục tiêu: Xác nhận đăng ký của người hiến máu hợp lệ.
* Tiền điều kiện: Người hiến máu đã gửi yêu cầu đăng ký thành công.
* Luồng chính/ Thành công:

- Nhân viên y tế đăng nhập vào hệ thống.

- Nhân viên y tế chọn ***Danh sách đăng ký hiến máu.***

- Nhân viên y tế mở hồ sơ một người hiến máu.

- Nhân viên y tế ***xác nhận nếu đủ điều kiện***.

- Hệ thống cập nhật trạng thái: “Đã xác nhận”.

- Người hiến máu nhận được ***thông báo xác nhận***.

* Trường hợp đặc biệt/ Người hiến máu không đủ điều kiện:

- Người hiến máu không đủ điều kiện về sức khỏe (Ví dụ: huyết áp không phù hợp,...).

1. **UseCase: Ghi nhận kết quả hiến máu:**

* Actor chính: Nhân viên y tế.
* Mục tiêu: Lưu kết quả hiến máu sau khi người hiến máu hoàn tất.
* Luồng chính/ Thành công:

- Sau khi hiến máu, nhân viên y tế mở ứng dụng, chọn ***Ghi nhận kết quả hiến máu.***

- Nhập thông tin: số lương máu lấy được, nhóm máu, tình trạng sau khi hiến vào hồ sơ người hiến máu.

- Hệ thống lưu kết quả vào hồ sơ.

- Hệ thống ***cập nhật vào lịch sử*** hiến máu.

- Hệ thống tự động lưu trữ & backup dữ liệu.

* Trường hợp đặc biệt/ Lỗi lưu kết quả:

- Hệ thống gặp sự cố lỗi kết nối hoặc nhân viên nhập sai định dạng thông tin.

# **Phần 3: Thiết kế hệ thống:**

## **I. Thiết kế kiến trúc hệ thống:**

**1. Mối quan hệ giữa các tầng:**

* **UI → UseCase:**

**-** UI nhận thao tác từ người dùng (click, nhập dữ liệu,..).

**-** Sau đó gọi đến UseCase để xử lý logic.

* **UseCase → Repository:**

- UseCase không truy cập dữ liệu trực tiếp.

- Gọi Repository để lấy dữ liệu từ API, Firebase hoặc cache...(calls).

* **Repository → Domain Entities:**

**-** Repository chịu trách nhiệm ánh xạ (mapping) dữ liệu từ nguồn ngoài thành Domain Entity.

**-** Đồng thời lưu trữ (persist) các entity vào CSDL khi cần.

* **Repository** ..|> **UseCase:**

**-** Repository trả kết quả về cho UseCase(return).

**-** Đảm bảo dữ liệu mà UseCase nhận được luôn ở dạng entity chuẩn.

* **UseCase ..|> Domain Entities:**

- UseCase thao tác với entity (đọc, validate,..).

- Nhưng không thay đổi định nghĩa hay cấu trúc của entity(read-only/ DTO).

* **Quan hệ cụ thể (Ví dụ):**

- AuthUseCase → AuthRepository để xử lý đăng nhập/đăng ký.

- EventUseCase → EventRepository để lấy danh sách sự kiện.

- AdminUseCase → AdminRepository để quản lý người dùng.

**2. Biểu đồ gói:**

A diagram of a computer program

AI-generated content may be incorrect.

## **II. Thiết kế sơ đồ lớp & Tạo cơ sở code:**

**1. Các lớp chính:**

**- User (lớp cha trừu tượng):**

**+ Attribute (thuộc tính):** id, name, role

**+ Method (phương thức):** login(), signUp()

**+ Validate (xác thực):** id không được rỗng, name 2-50 ký tự, role thuộc {admin, staff, user}.

**- Admin ( kế thừa User):**

**+ Method (phương thức):** manageLocation(List<Hospital>), manageStaff(List<Staff>), viewDonationHistoryByEvent(eventId)

**+ Validate (xác thực):** Chỉ role = “admin” mới được truy cập.

**- Donor ( kế thừa User):**

**+ Attribute (thuộc tính):** dateOfBirth, age, bloodGroup, city, gender, phoneNumber, willingToDonate, about, avatar, donationHistiory

**+ Method (phương thức):** registerDonateBlood(), cancelRegistrationDonate(), viewHistory(), viewAppointment()

**+ Validate (xác thực):**

**- Staff ( kế thừa User):**

**+ Attribute (thuộc tính):** staffId, code

**+ Method (phương thức):** confirmRegistration(), manageEvent(Event), manageDonor(List<Donor>)

**+ Validate (xác thực):** mỗi 1 staff có 1 mã code riêng, chỉ có role = “staff” mới có thể truy cập.

**- Hospital:**

**+ Attribute (thuộc tính):** hospitalId, hospitalName, imgUrl, address, phone, province, district, createAt, updateAt, eventList.

**- Event:**

**+ Attribute (thuộc tính):** id, name, description, location, date, time, deadline, donorList, capacity, donorCount.

**+ Method (phương thức):** updateDonorCount()

**- DonateBlood:**

**+ Attribute (thuộc tính):** userId, donor, nationalIdCard(CCCD), status.

**+ Method (phương thức):** addToEvent(), deleteFromEvent().

**+ Validate (xác thực):** nationalIdCard đúng định dạng 12 số,

**- UserDonationHistory:**

**+ Attribute (thuộc tính):** id, donorId, donorFullName, quantityInMI, status, event.

**+ Method (phương thức):** getDonorById(donorId)

**- SystemService (Helper/ Service)**

**+ Attribute (thuộc tính):** eventList: List<Event>.

**+ Method (phương thức):** Validation(), Notification(), backUpData()

**2. Mối quan hệ giữa các lớp:**

**- Kế thừa:**

+ Admin, Donor, Staff kế thừa từ User.

**- Admin – Staff:**

+ 1 Admin có thể quản lý nhiều Staff (1-n).

**- Admin – Hospital:**

+ 1 Admin có thể quản lý nhiều Hospital (1-n).

**- Hospital – Event:**

+ 1 Hospital có thể quản lý nhiều Event (1-n).

+ 1 Event chỉ thuộc về 1 Hospital.

**- Event – DonateBlood – Donor:**

+ Mỗi quan hệ n – m giữa Donor và Event thông qua DonateBlood.

+ 1 Donor có thể tham gia nhiều Event.

+ 1 Event có thể có nhiều Donor.

**- Staff – Event:**

+ 1 Staff có thể quản lý nhiều Event (1-n).

**- Staff – Donateblood:**

+ 1 Staff có thể xác nhận nhiều đơn đăng ký hiến máu (1-n).

**- Donor – UserDonationHistory:**

+ 1 Donor có thể có nhiều lịch sử hiến máu (1-n).

**- Event – UserDonationHistory:**

+ 1 Event có thể có nhiều bản ghi lịch sử hiến máu (1-n).

**3. Biểu đồ lớp:**

A diagram of a computer

AI-generated content may be incorrect.

## **III. Thiết kế tương tác:**

**1. Biểu đồ trình tự:**

* **Login:**

A screenshot of a computer screen

AI-generated content may be incorrect.

* **Sign Up:**

A diagram of a sign up

AI-generated content may be incorrect.

* **Register/Cancel Donation Blood:**

A screenshot of a computer

AI-generated content may be incorrect.

* **Staff confirm registration:**

A diagram of a blood donation process

AI-generated content may be incorrect.

* **Staff manage event & donor:**

A screenshot of a diagram

AI-generated content may be incorrect.

* **Donor views history & appointment:**

A diagram of a service

AI-generated content may be incorrect.

* **System service:**

A diagram of a software project

AI-generated content may be incorrect.

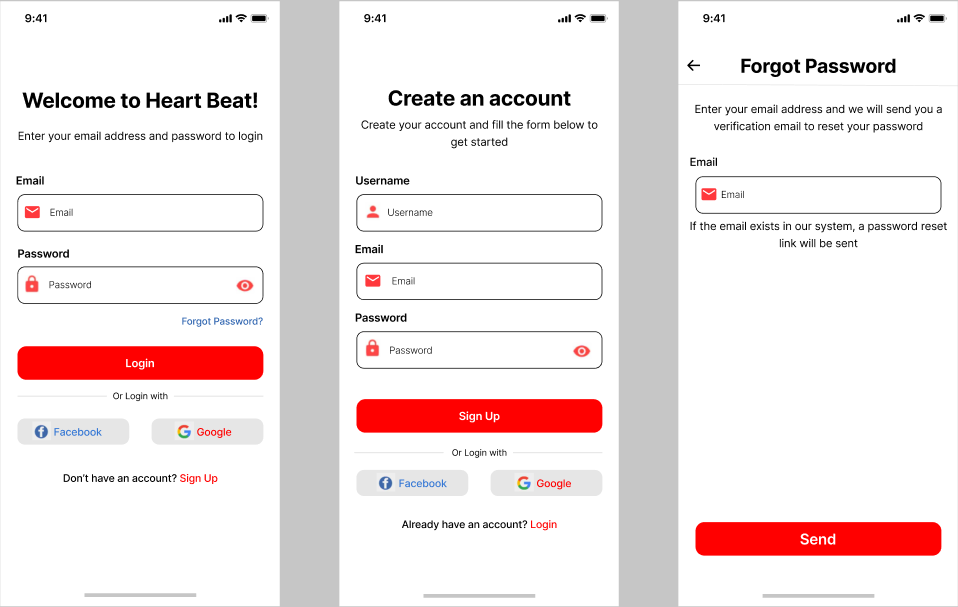
**2. Thiết kế giao diện: (Figma):**

* **Link thiết kế giao diện Figma (Phần Design trên Github):**

*https://www.figma.com/design/nlb29AHZX4q6bgxKT03d6j/Heart-Beat-Figma?node-id=0-1&m=dev&t=zltqMDcrCJK8UzzN-1.*

* **Một số hình ảnh giao diện tổng quát:**

- SignUp & SignIn:



- Profile setup:

Screens screenshot of a phone

AI-generated content may be incorrect.

- Blood donations:

Screens screenshot of a phone

AI-generated content may be incorrect.

- Event:

Screens screenshot of a phone

AI-generated content may be incorrect.

## **IV. Thiết kế hành vi và trạng thái:**

**1. Donate Blood Stage Machine(mô tả lifecycle của 1 lượt đăng ký của donor cho 1 event):**

* NotRegistered: Donor chưa đăng ký hiến máu.
* Pending: Donor vừa đăng ký, chờ Staff xác nhận.
* Confirmed: Staff duyệt đăng ký thành công.
* Cancelled: Donor tự hủy hoặc bị từ chối.
* Completed: Donor tham gia hiến máu thành công.
* Archived: backUp dữ liệu, lưu trữ vào lịch sử.

A diagram of a system

AI-generated content may be incorrect.

**2. User State Machine (Admin/Staff/Donor – mô tả trạng thái tài khoản):**

* NewAccount: tạo tài khoản mới (chưa verify hoặc active).
* Active: tài khoản hoạt động (có thể thực hiện các chức năng tùy thuộc vào role = {admin, staff, donor}).
* Suspended: tài khoản bị tạm khóa(admin suspend).
* Deactivated: tài khoản bị xóa/ vô hiệu hóa(user/admin delete).

A diagram of a computer flowchart

AI-generated content may be incorrect.

**3. Event State Machine(quản lý trạng thái của 1 event):**

* Scheduled: event được tạo bởi admin (chưa mở đăng ký).
* Open: đang mở cho phép đăng ký hiến máu.
* Full: đã đủ chỗ/ đủ số lượng không nhận thêm.
* Closed: đã đóng đăng ký hoặc sự kiện đã kết thúc.
* Archived: backUp/lưu trữ.

A diagram of a flowchart

AI-generated content may be incorrect.

## **V. Áp dụng mẫu thiết kế:**

- Trong dự án ***Heart Beat***, nhóm đã áp dụng nhiều mẫu Design Pattern khác nhau nhằm đảm bảo tính mở rộng, dễ bảo trì và tuân thủ nguyên tắc của mô hình ***Clean Architecture***. Các mẫu thiết kế được áp dụng như sau:

**1. Mô hình tổng quan (Clean Architecture):**

* Dữ liệu được luân chuyển từ Domain đến UI theo chuỗi.
* Phân tách rõ ràng giữa các tầng, dễ dàng mở rộng và bảo trì, có thể thay đổi tầng Data (Firebase, API,...) mà không ảnh hưởng đến logic nghiệp vụ hoặc giao diện người dùng.
* Cấu trúc:

UI Layer → ViewModel Layer → Domain Layer → Data Layer → Data Source Layer

**2. Dependency Injection (DI) – Hilt:**

* Dependency Injection (DI) được triển khai trong dự án thông qua Hilt, giúp quản lý toàn bộ các dependency một cách tự động và nhất quán.
* Mỗi Module trong Hilt tương ứng với một nhóm chức năng (như Auth, Event, Donation, Donor, Hospital, System, App, …).
* Mỗi module chịu trách nhiệm cung cấp (provide) các dependency cần thiết như Repository, UseCase, hoặc các Firebase instance (Firestore, Auth, Storage, …).
* Việc áp dụng Hilt giúp đơn giản hóa quá trình khởi tạo, tăng khả năng tái sử dụng, và tối ưu hóa quản lý vòng đời đối tượng trong toàn bộ
* Cấu trúc:

A diagram of a diagram

AI-generated content may be incorrect.

**3. Observer Pattern:**

* Được áp dụng giữa ViewModel và UI (Compose). UI đóng vai trò là Observer, lắng nghe sự thay đổi của ViewModel thông qua StateFlow hoặc LiveData.
* Khi dữ liệu trong ViewModel thay đổi, UI sẽ tự đông cập nhật giao diện mà không cần can thiệp thủ công.
* Pattern này chủ yếu áp dụng cho 1 đối tượng (Subject) thông báo tự động đến nhiều đối tượng khác (Observers) khi trạng thái của nó thay đổi.
* Class Diagram (Ví dụ: Event):
  + Subject (EventViewModel) quản lý dữ liệu, Observer (EventScreen) hiển thị giao diện.
  + Khi dữ liệu thay đổi (ví dụ 1 lượt đăng ký hiến máu thành công), EventViewModel cập nhật uiState và gọi notifyObservers() để báo cho tất cả màn hình đang quan sát.
  + EventScreen nhận được thông báo và tự động cập nhật giao diện hiện thị danh sách/ số lượng người đăng ký.

A diagram of a function

AI-generated content may be incorrect.

**4. Singleton Pattern:**

* Các đối tượng chỉ được khởi tạo duy nhất một lần và được Hilt quản lý vòng đời thông qua annotation @Singleton.
* Mục tiêu của việc áp dụng mẫu thiết kế này là tránh khởi tạo lặp lại, tối ưu hiệu năng bộ nhớ, và đảm bảo tính nhất quán của dữ liệu trong suốt vòng đời ứng dụng.
* Class Diagram:
* SingletonComponent thành phần quản lý Singleton:

+ Có thuộc tính instance(duy nhất trong suốt vòng đời ứng dụng).

+ Có phương thức getInstance() để lấy ra dữ liệu đối tượng.

* FirebaseAuth, FirebaseFirestore, RepositoryImpl, AppModule:

+ Đây là các class dùng chung trong toàn app.

+ Không tự khởi tạo nhiều lần, lấy dữ liệu thông qua SingletonComponent.

* Mối quan hệ: SingletonComponent cung cấp (provides) các dependency giúp đảm bảo chỉ có 1 instance duy nhất.

A screenshot of a diagram

AI-generated content may be incorrect.

**5. Factory Pattern:**

* Trong dự án, Hilt tự động sinh ra các Factory class để khởi tạo ViewMode thông qua ViewModelFactory cùng các UseCase bên trong.
* Pattern này giúp tách biệt logic khởi tạo ViewModel khỏi Activity/Fragment và đảm bảo các dependency được inject đầy đủ.
* Factory đóng vai trò trung gian giữa Hilt và ViewModel.
* Class Diagram:
* ViewModelFactory:

+ Vai trò: lớp Factory chịu trách nhiệm khởi tạo các ViewModel.

+ Có phương thức create(type: String) → trả về một đối tượng ViewModel tùy theo loại được yêu cầu.

* ViewModel:

+ Lớp cha của các ViewModel cụ thể (EventViewModel, DonorViewModel, …).

+ Giúp định nghĩa chung cho tất cả các ViewModel, dễ quản lý.

Mỗi ViewModel chứa logic để làm việc với UI tương ứng. Chúng phụ thuộc vào UseCase để xử lý nghiệp vụ (business logic).

* UseCase:

+ Chứa logic nghiệp vụ độc lập (ví dụ: kiểm tra thông tin người hiến, quản lý sự kiện, xác thực đăng nhập).

+ Được inject vào ViewModel nhờ Hilt/Dagger.

A diagram of a flowchart

AI-generated content may be incorrect.

**6. Repository Pattern:**

* Repository Pattern được áp dụng trong dự án để quản lý việc truy cập và xử lý dữ liệu một cách thống nhất:
* Áp dụng cho tất cả các đối tượng trong domain như Event, Province, Hospital…
* Repository là lớp trung gian giữa tầng Domain và tầng Data, chịu trách nhiệm lấy dữ liệu từ nhiều nguồn (Remote hoặc Local) và cung cấp cho UseCase.Ví dụ: EventRepository, ProvinceRepository, HospitalRepository…
* Class Diagram (Ví dụ: Event, Donor):
  + UseCase: Là nơi gọi đến Repository để thực hiện các hành động (CRUD).
  + Repository: Định nghĩa các phương thức chung để lấy hoặc xử lý dữ liệu.
  + RepositoryImpl: là lớp Implements thực hiện chi tiết cách lấy dữ liệu (Firestore, API,...).

A diagram of a program

AI-generated content may be incorrect.

# **Phần 4: Triển khai hệ thống:**

## **I. Công nghệ sử dụng:**

|  |  |  |
| --- | --- | --- |
| Công nghệ | Mục đích sử dụng | Vai trò trong dự án |
| Kotlin | Ngôn ngữ lập trình chính cho Android | Xây dựng toàn bộ logic ứng dụng và xử lý nghiệp vụ |
| Jetpack Compose | UI ToolKit hiện đại cho Android | Thiết kế giao diện động, trực quan dễ mở rộng |
| MVVM | Mẫu kiến trúc (Model-View-ViewModel) | Tách biệt giao diện, xử lý logic và dữ liệu |
| Clean Architecture | Mô hình kiến trúc phần mềm theo tầng (UI, UseCase, Repository, Domain) | Giúp hệ thống rõ ràng, đảm bảo phân tách trách nhiệm giữa các tầng |
| Firebase | Backend-as-a-Service (Authentication, Firestore, Storage, Notifications). | Xác thực tài khoản, lưu trữ dữ liệu sự kiện/đăng ký, gửi thông báo đẩy (push notification). |
| Cloudinary | Nền tảng quản lý media (ảnh, video). | Lưu trữ và phân phối hình ảnh (hình ảnh bệnh viện/sự kiện). |
| DataStore | Thư viện lưu trữ dữ liệu nhẹ, thay thế SharedPreferences. | Lưu trữ dữ liệu cục bộ (token, thông tin đăng nhập, cấu hình cá nhân). |
| Gradle Kotlin DSL | Công cụ quản lý build và dependency bằng Kotlin DSL. | Quản lý phụ thuộc, build pipeline của ứng dụng. |
| Git | Hệ thống quản lý phiên bản phân tán. | Quản lý mã nguồn, hỗ trợ teamwork, CI/CD. |

## **II. Lập trình:**

**1. Cấu trúc thư mục:**

|  |  |
| --- | --- |
| **Thư mục** | **Thư mục con** |
| data | model, repository |
| di | modules |
| domain | entity, usecase |
| presentation | components, features |
| ui | dimens, theme |

**2. Luồng hoạt động:**

* Mỗi màn hình (screen) có 1 ViewModel tương ứng.
* ViewModel gọi UseCase → Repository → Firebase.
* Repository quản lý dữ liệu từ Firestore/Storage.
* UI lắng nghe StateFlow để tự động cập nhật (Observer pattern).

**3. Mô tả lập trình một số chức năng chính:**

|  |  |
| --- | --- |
| **Chức năng** | **Mô tả** |
| Đăng nhập/ Đăng ký | Người dùng xác thực tài khoản bằng Firebase Authentication. |
| Đăng ký/hủy đăng ký hiến máu | Người dùng chọn sự kiện và gửi yêu cầu đăng ký. |
| Xác nhận đăng ký (Staff) | Nhân viên y tế duyệt đăng ký, cập nhật trạng thái. |
| Ghi nhận kết quả hiến máu | Nhân viên y tế lưu lại kết quả và lịch sử hiến máu. |
| Quản lý sự kiện/ địa điểm | Nhân viên y tế thêm, sửa, xóa sự kiện. |

# **Phần 5: Tổng kết & Hướng phát triển:**

## **I. Đánh giá những điểm đã làm được và chưa làm được:**

**1. Những điểm đã làm được trong dự án:**

* Phân tích yêu cầu chi tiết: Đã xác định được rõ ràng các vai trò (Actors), yêu cầu chức năng và phi chức năng, tạo tiền đề vững chắc cho việc thiết kế.
* Thiết kế hệ thống toàn diện: Đã xây dựng được một kiến trúc phần mềm rõ ràng theo mô hình Clean Architecture, giúp hệ thống dễ bảo trì và mở rộng.
* Mô hình hóa chi tiết: Hoàn thành các biểu đồ quan trọng như UseCase, Biểu đồ hoạt động, Biểu đồ lớp, Biểu đồ tuần tự và Biểu đồ trạng thái giúp trực quan hóa luồng hoạt động và cấu trúc của hệ thống.
* Áp dụng các mẫu thiết kế (Design Patterns) hiệu quả: Nghiên cứu và lên kế hoạch áp dụng các mẫu thiết kế phổ biến như Repository, Observer, Singleton, Factory giúp code có cấu trúc tốt và tuân thủ các quy tắc lập trình hiện đại.
* Xây dựng giao diện người dùng (UI) trên Figma: Thiết kế được giao diện mẫu, tạo ra góc nhìn trực quan về sản phẩm cuối cùng và định hướng cho quá trình lập trình giao diện.
* Xây dựng được nền tảng mã nguồn: Thiết lập cấu trúc dự án rõ ràng và tích hợp các công nghệ nền tảng như Firebase, Hilt.

**2. Những điểm còn hạn chế/chưa làm được:**

* Dự án chỉ mới tập trung vào chức năng lõi như đăng ký hiến máu và các chức năng quản lý cơ bản chưa đi sâu vào việc quản lý sức khỏe sức khỏe cho người dùng.
* Thiếu kiểm thử tự động: Chưa xây dựng các kịch bản Unit Test cho UseCase, ViewModel để đảm bảo tính đúng đắn và ổn định của code khi có sự thay đổi.
* Chưa triển khai được hỗ trợ đa nền tảng mới chỉ cài đặt được trên Mobile Android.
* Chưa triển khai được các tính năng nâng cao như tính năng cộng đồng, trí tuệ nhân tạo, đồng bộ với thiết bị đeo tay vẫn còn là kế hoạch cho tương lai.

## **II. Hướng phát triển tiếp theo của sản phẩm:**

Trong tương lai, nhóm mong muốn mở rộng ứng dụng Heart Beat không chỉ dừng lại ở việc quản lý và kết nối hiến máu, mà còn trở thành một nền tảng tổng thể về chăm sóc sức khỏe cộng đồng. Các hướng phát triển chính bao gồm:

**1. Mở rộng chức năng quản lý sức khỏe tổng thể:**

* Tích hợp chức năng theo dõi chỉ số sức khỏe cơ bản (huyết áp, nhịp tim, BMI).
* Gợi ý chế độ ăn uống và tập luyện phù hợp cho người hiến máu.
* Đồng bộ dữ liệu với các thiết bị đeo thông minh (smartwatch, fitness tracker).

**2. Hệ thống gợi ý thông minh (AI):**

* Dùng trí tuệ nhân tạo để gợi ý thời điểm hiến máu phù hợp.
* Dự đoán nhu cầu máu theo khu vực, hỗ trợ cơ quan y tế lên kế hoạch sự kiện.

**3. Phát triển tính năng cộng đồng:**

* Xây dựng diễn đàn/nhóm chia sẻ kinh nghiệm hiến máu.
* Tạo cơ chế “thành viên tích cực” để khuyến khích hiến máu thường xuyên.
* Gamification (tích điểm, huy hiệu, xếp hạng) để tăng động lực tham gia.

**4. Hỗ trợ đa nền tảng và mở rộng quốc tế:**

* Phát triển phiên bản Web App, iOS để tiếp cận nhiều đối tượng người dùng.
* Mở rộng mô hình sang các quốc gia khác trong khu vực Đông Nam Á.

**5. Bảo mật và ổn định hệ thống:**

* Ứng dụng các chuẩn bảo mật y tế quốc tế (HIPAA, HL7).
* Nâng cao khả năng chịu tải, hỗ trợ hàng chục nghìn người dùng đồng thời.

# **Lời cảm ơn**

Để hoàn thành bài tập lớn môn học Phân tích và Thiết kế Phần mềm với đề tài "Xây dựng phần mềm Heart Beat trên nền tảng Mobile" , trước hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy **Vũ Quang Dũng** và thầy **Nguyễn Xuân Quế**. Trong suốt quá trình thực hiện đề tài, thầy đã luôn tận tình chỉ bảo, đưa ra những góp ý chuyên môn quý báu và định hướng kịp thời, giúp chúng em tháo gỡ những khó khăn, vướng mắc để hoàn thiện dự án một cách tốt nhất.

Mặc dù đã rất cố gắng nhưng do kiến thức và kinh nghiệm thực tiễn còn hạn chế, bài báo cáo chắc chắn không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp từ các thầy để đề tài được hoàn thiện hơn nữa.

*Một lần nữa, chúng em xin chân thành cảm ơn!*