ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:

Sinh viên thực hiện: Hoàng Anh Tuấn
Lớp ĐTVT 03 - K63
Giảng viên hướng dẫn: TS.

Hà Nội 3/2021

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

Đề tài:

Sinh viên thực hiện: Hoàng Anh Tuấn
Lớp ĐTVT 03 - K63
Giảng viên hướng dẫn: TS.
Cán bộ phản biện :

Hà Nội 3/2021

LỜI NÓI ĐẦU

Phần này trình bày một cách rất khái quát (khoảng 1 đến 2 trang) với bối cảnh hình thành và mục đích đồ án. Lời cảm ơn với những tổ chức và cá nhân góp phần trong việc hoàn thiện đồ án(nếu có) nên đặt cuối mục này

LỜI CAM ĐOAN

Tôi tên là HOÀNG ANH TUẤN, mã sô sinh viên xxx, sinh viên lớp yyy, khóa zzz, người hướng dẫn là TS.. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án ..... là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, ngày tháng năm

Người cam đoan

HOÀNG ANH TUẤN

MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN

Phần này tóm tắt những mục đích và các kết luận quan trọng của đồ án bằng cả tiếng việt và cả tiếng Anh

PHẦN MỞ ĐẦU

Đặt vấn đề

Lĩnh vực y tế đang có những bước chuyển mình lớn trong cuộc cách mạng công nghiệp lần thứ tư (hay còn được gọi là cuộc cách mạng công nghiệp 4.0). Đại dịch COVID-19 đã chứng minh được tầm quan trọng của việc áp dụng khoa học kỹ thuật vào những sản phẩm y tế giúp đẩy lùi dịch bệnh, có thể kể đến như máy rửa tay tự động do TS.Hàn Huy Dũng (đang công tác tại Trường Điện - Điện tử, thuộc Đại học Bách khoa Hà Nội) (thêm reference) cùng các cộng sự sáng chế, và một số ứng dụng di động nổi bật được hầu hết người dân Việt Nam sử dụng trong đại dịch COVID-19 như Bluezone - ứng dụng cảnh báo tiếp xúc gần với những người nhiễm COVID qua Bluetooth low energy, ứng dụng NCOVI, theo dõi các ca nhiễm và thực hiện khai báo y tế, ứng dụng PC-Covid để cập nhật các thông tin tiêm vắc xin, thông tin xét nghiệm. Đây là một tín hiệu cho thấy nước ta đang áp dụng công nghệ 4.0 vào trong ngành y tế một cách chủ động. Hiện nay, việc chăm sóc sức khoẻ đang được chú trọng, đặc biệt là đối với những mẹ bầu, những người cần theo dõi sức khoẻ định kỳ liên tục, việc di chuyển đến bệnh viện đông đúc để thăm khám rất khó khăn, cộng với chi phí không hề rẻ, và tỉ lệ sẩy thai, thai lưu khi phát hiện không kịp thời là khá cao. câu hỏi đặt ra là có cách nào có thể giúp các mẹ bầu không cần di chuyển

Đề xuất hệ thống

(Nếu có) [1]

Cấu trúc đồ án

(nếu có)

CHƯƠNG 1. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

Trong chương này, chúng em sẽ tiến hành phân tích hệ thống cho dự án đề tài "Hệ thống quản lý ECG". Đây là một hệ thống được thiết kế để quản lý và xử lý dữ liệu điện tâm đồ (ECG) của người dùng. Hệ thống cung cấp khả năng ghi lại và hiển thị dữ liệu điện tim, cho phép người dùng theo dõi và đánh giá sự hoạt động của tim. Người dụng cũng có thể thực hiện trao đổi dựa trên các kết quả điện tim đo được. Những thông tin này có thể hữu ích trong việc theo dõi sức khỏe tim mạch, theo dõi hiệu quả của liệu pháp và hỗ trợ quyết định của người dùng.

1.1 Yêu cầu hệ thống

1.1.1 Yêu cầu về người dùng hệ thống

Hệ thống được thiết kế để phục vụ các đối tượng sau:

- Bệnh nhân: Người sử dụng hệ thống để thực hiện kiểm tra ECG thông qua Bluetooth và theo dõi sức khỏe của mình. Bệnh nhân có quyền truy cập vào kết quả ECG của mình, được một bác sĩ theo dõi và có thể theo dõi các thông tin liên quan đến điện tim và sức khoẻ.
- Bác sĩ: Người sử dụng hệ thống để xem và đánh giá kết quả ECG của bệnh nhân, đưa ra nhận xét và đề xuất điều trị. Bác sĩ có thể trao đổi với bệnh nhân và gửi thông báo quan trọng liên quan đến chăm sóc sức khỏe.
- Quản trị viên: Người sử dụng hệ thống để quản lý các tài khoản người dùng, phân công bệnh nhân cho bác sĩ và quản lý mối quan hệ giữa bác sĩ và bệnh nhân.

1.1.2 Yêu cầu chức năng

Các chức năng chính của hệ thống bao gồm:

- Ghi lại dữ liệu điện tim: Hệ thống cho phép ghi lại tín hiệu điện tim từ máy đo ECG (Electrocardiogram) hay thiết bị đo điện tim khác. Dữ liệu được chuyển tới ứng dụng của người dùng thông qua Bluetooth để lưu trữ, phân tích và có thể xem lại sau này.
- Hiển thị và phân tích dữ liệu: Hệ thống hiển thị dữ liệu điện tim theo dạng đồ thị. Hệ thống cũng hỗ trợ xuất ra các tệp đã được chuẩn hoá cho các dữ liệu chuỗi thời gian (time-series database) để phục vụ mục đích phân tích và nghiên cứu sâu hơn.

- Lưu trữ: Hệ thống hỗ trợ lưu dữ liệu mà người dùng đo được từ thiết bị trên cả ứng dụng và trên server của hệ thống. Dữ liệu điện tim cũng được đồng bộ hóa và lưu trữ trên máy chủ của hệ thống. Qua quá trình đồng bộ hóa, dữ liệu từ ứng dụng được truyền đến máy chủ và được lưu trữ an toàn và bảo mật trên hệ thống. Việc lưu trữ dữ liệu điện tim trên cả ứng dụng và máy chủ giúp đảm bảo rằng dữ liệu quan trọng này được lưu trữ một cách đáng tin cậy và có sẵn cho phân tích hoặc sử dụng tương lai.
- Trao đổi và chia sẻ thông tin về dữ liệu điện tim: Hệ thống giúp người dùng có thể trao đổi trực tiếp với nhau, chia sẻ kết quả đo điện tim, hỏi đáp về các vấn đề sức khỏe hoặc thảo luận về các quyết định. Điều này mang lại sự tiện lợi và hỗ trợ đáng kể cho người dùng trong việc xác định về tình trạng sức khoẻ hiện tại của bản thân.

Hệ thống hỗ trợ các chức năng cơ bản sau đối với người dùng:

Đối với người dùng là bệnh nhân:

- Đăng nhập và đăng ký tài khoản bằng thông tin cá nhân, bao gồm tên, địa chỉ email, ngày sinh, số điện thoại và mật khẩu.
- Cập nhật các thông tin cá nhân.
- Xem kết quả ECG của mình, bao gồm biểu đồ và các thông số liên quan.
- Theo dõi các tin tức liên quan đến sức khoẻ và tim mạch.
- Nhận thông báo và có thể trao đổi trực tiếp với bác sĩ về tình hình sức khoẻ và các kết quả đo được từ thiết bị.

Đối với người dùng là bác sĩ:

- Được cấp tài khoản để sử dụng hệ thống.
- Cập nhật các thông tin cá nhân.
- Xem danh sách bệnh nhân được phân công cho mình và xem kết quả ECG của từng bệnh nhân.
- Đánh giá và đưa ra nhận xét về kết quả ECG của bệnh nhân.
- Trao đổi các thông tin liên quan đến tình hình sức khoẻ và kết quả đo của bệnh nhân.

Đối với người dùng là quản trị viên:

- Đăng nhập và đăng ký tài khoản bằng thông tin cá nhân, bao gồm tên, địa chỉ email, số điện thoại và mật khẩu.
- Cập nhật thông tin cá nhân.
- Quản lý danh sách người dùng trong hệ thống, bao gồm bệnh nhân và bác sĩ.
- Phân công bệnh nhân cho các bác sĩ và quản lý mối quan hệ giữa bác sĩ và bệnh nhân.
- Quản lý các tin tức được đăng trên ứng dụng của người dùng.

1.1.3 Yêu cầu phi chức năng

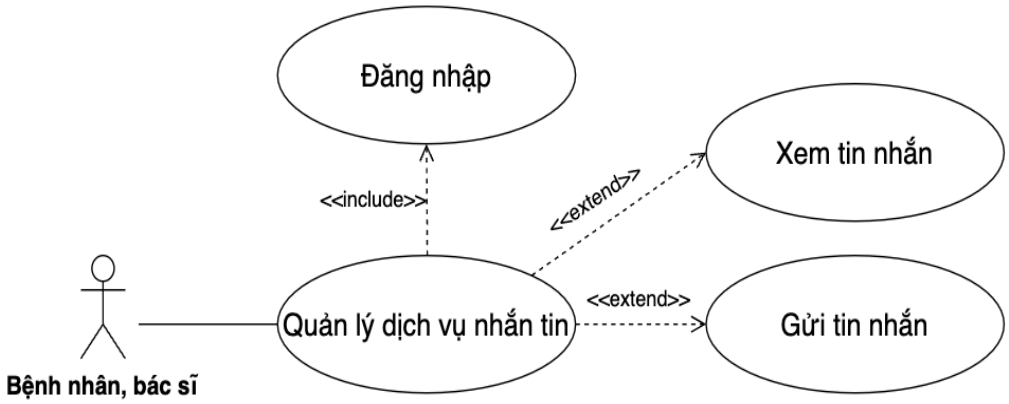
- Hệ thống hỗ trợ ngôn ngữ Tiếng Việt và Tiếng Anh.
- Hệ thống cần đảm bảo tính bảo mật và quyền riêng tư thông tin của người dùng.
- Hệ thống phải có giao diện người dùng thân thiện, dễ sử dụng và có thể tương tác trên các thiết bị di động.
- Thời gian phản hồi của hệ thống phải nhanh chóng và ổn định.
- Hệ thống cần sao lưu dữ liệu định kỳ để đảm bảo tính an toàn và khả năng khôi phục dữ liệu khi cần thiết.

Thông qua việc phân tích yêu cầu hệ thống, chúng ta có cái nhìn tổng quan về các chức năng, yêu cầu phi chức năng và các đối tượng người dùng mà hệ thống phải hỗ trợ. Phần phân tích này sẽ cung cấp cơ sở cho việc thiết kế và phát triển hệ thống quản lý ECG, đáp ứng đầy đủ các yêu cầu của người dùng và đảm bảo hiệu suất, bảo mật và tính khả dụng của hệ thống.

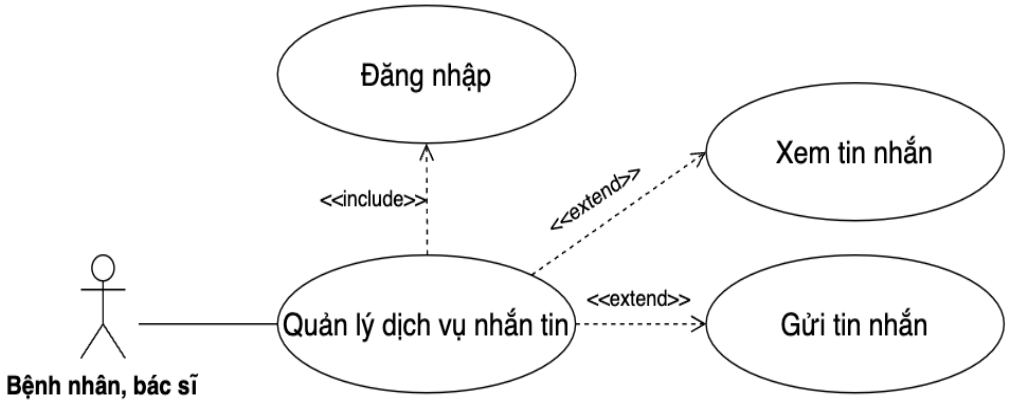
1.2 Phân tích tổng quan hệ thống

1.2.1 Sơ đồ use case

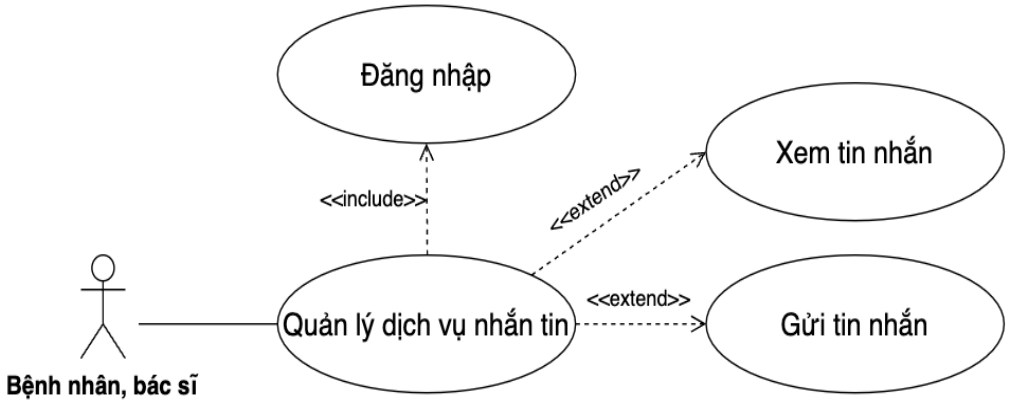
1.2.1.1 Use case xem/nhận tin nhắn



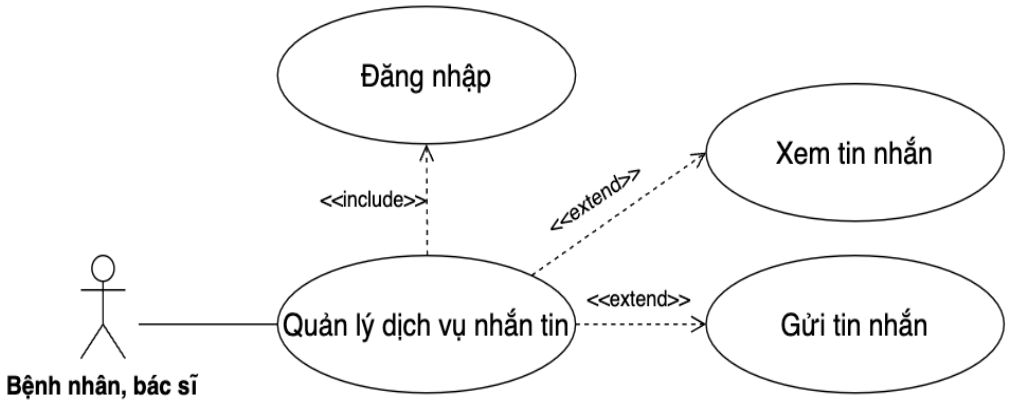
Hình 1.1 Sơ đồ tuần tự chức năng đăng ký trên App



Hình 1.2 Sơ đồ tuần tự chức năng đăng ký trên App

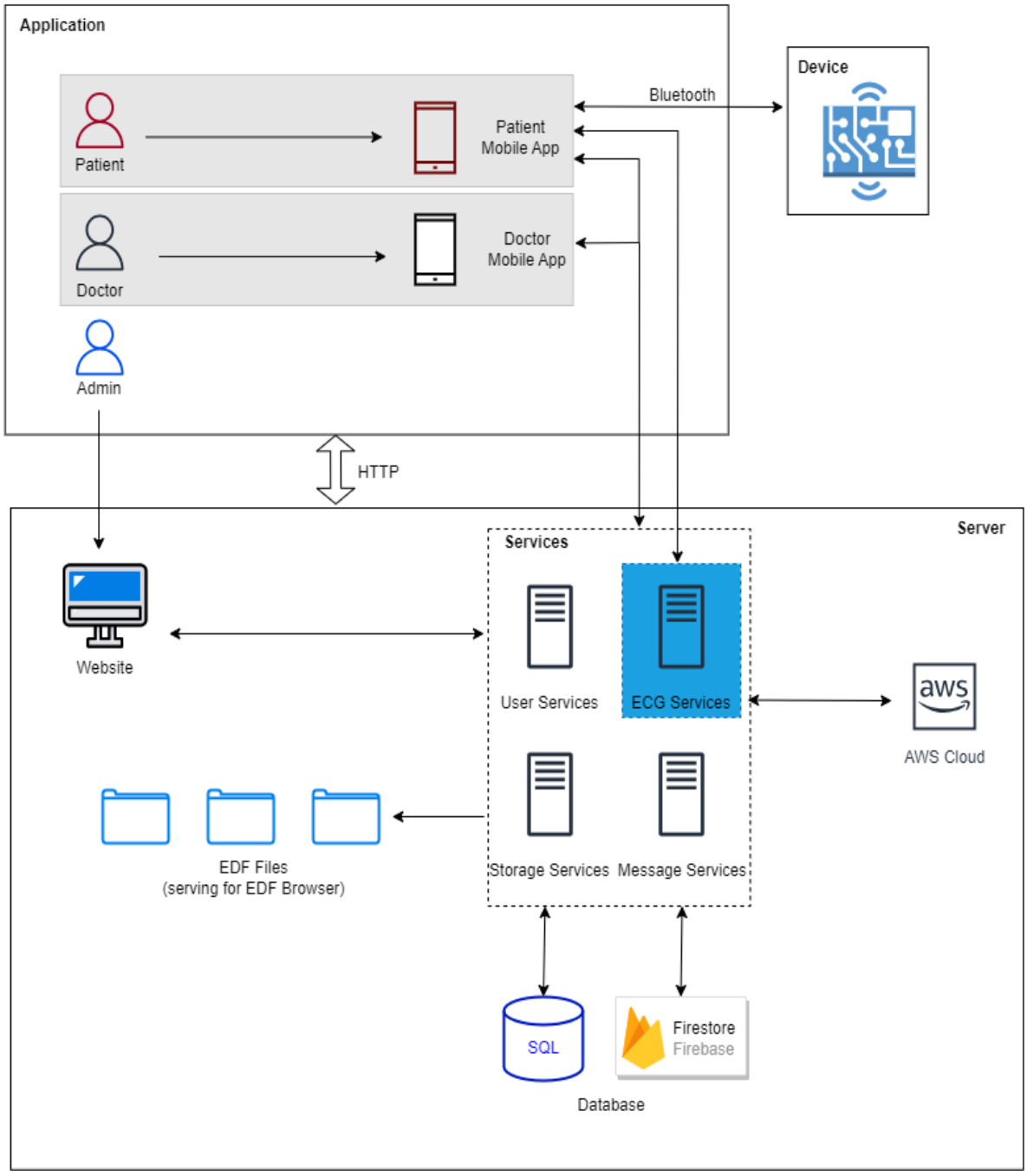


Hình 1.3 Sơ đồ tuần tự chức năng đăng ký trên App



Hình 1.4 Sơ đồ tuần tự chức năng đăng ký trên App

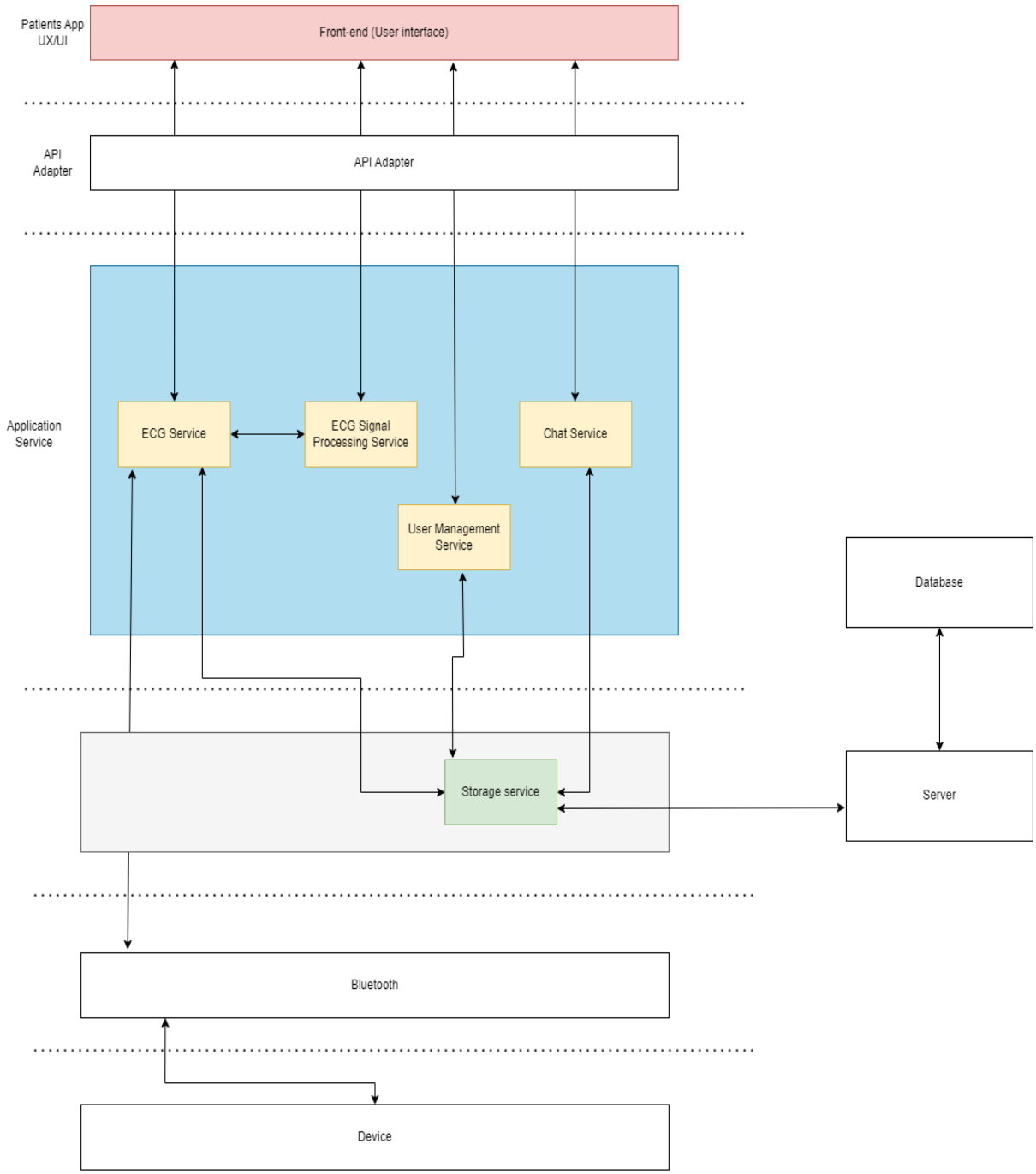
1.2.2 Sơ đồ kiến trúc hệ thống



Hình 1.5 Kiến trúc hệ thống

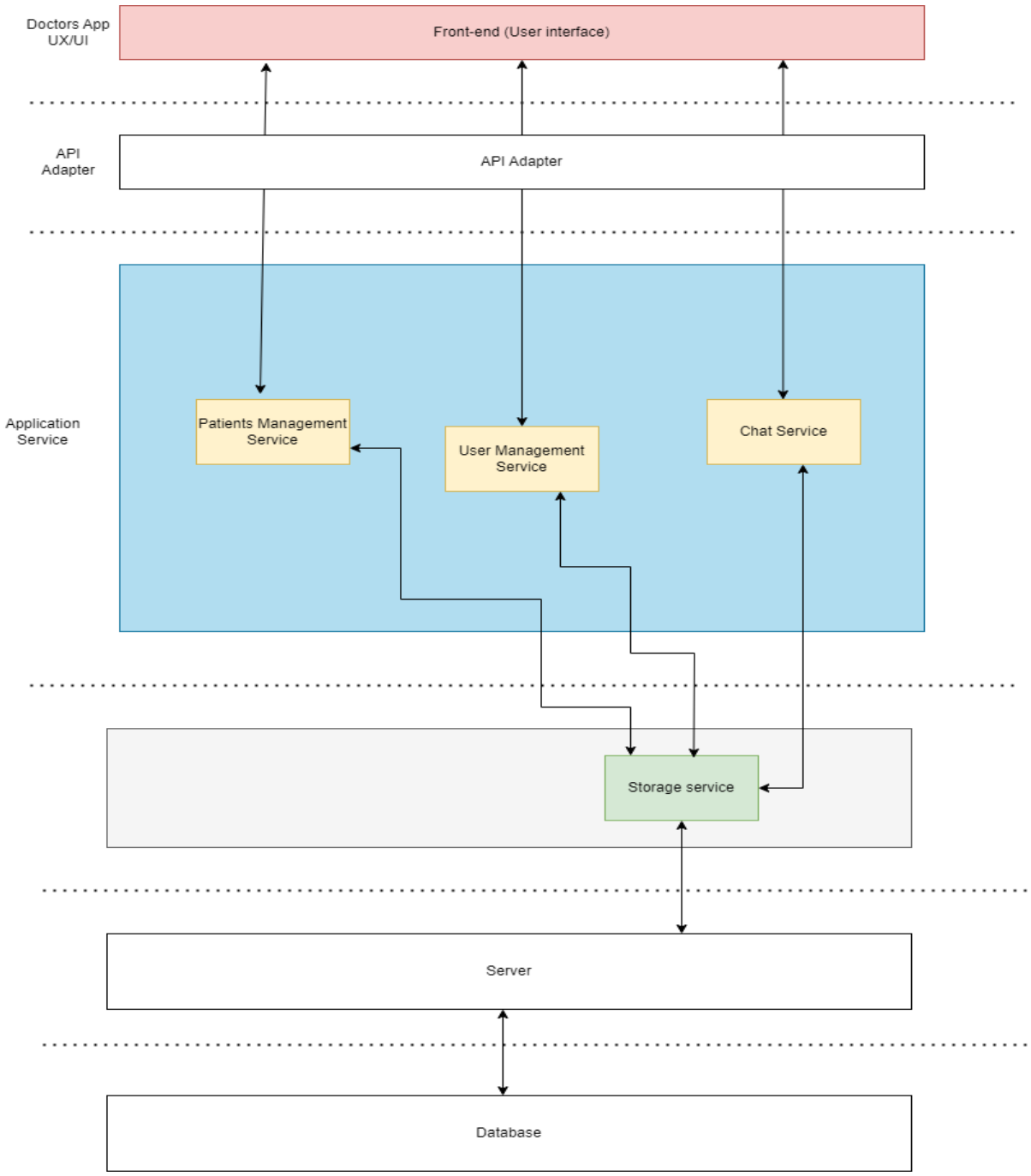
1.2.3 Sơ đồ khối phần mềm

1.2.3.1 Ứng dụng cho bệnh nhân



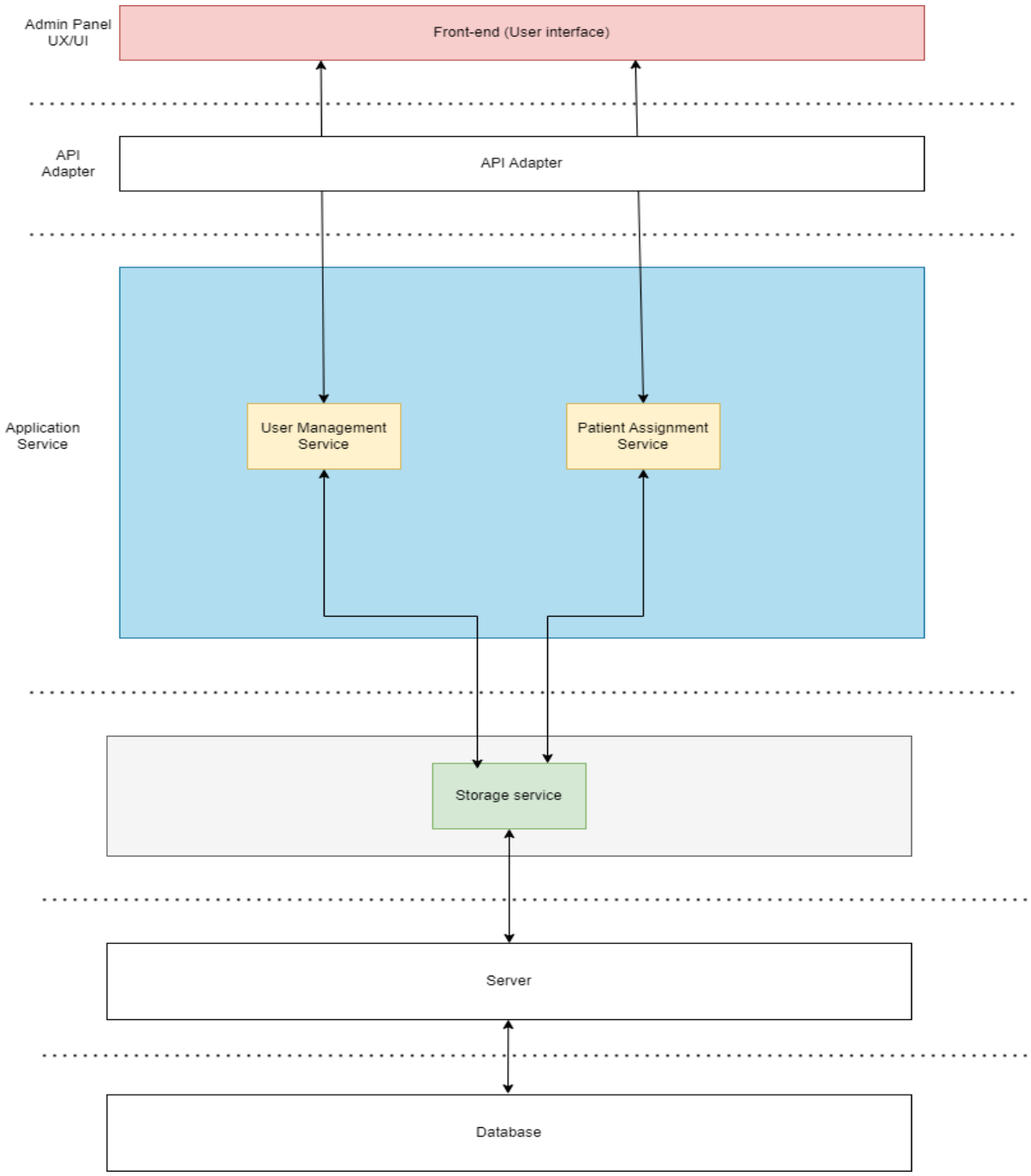
Hình 1.6 Sơ đồ khối phần mềm của ứng dụng cho bệnh nhân

1.2.3.2 Ứng dụng cho bác sỹ



Hình 1.7 Sơ đồ khối phần mềm của ứng dụng cho bác sĩ

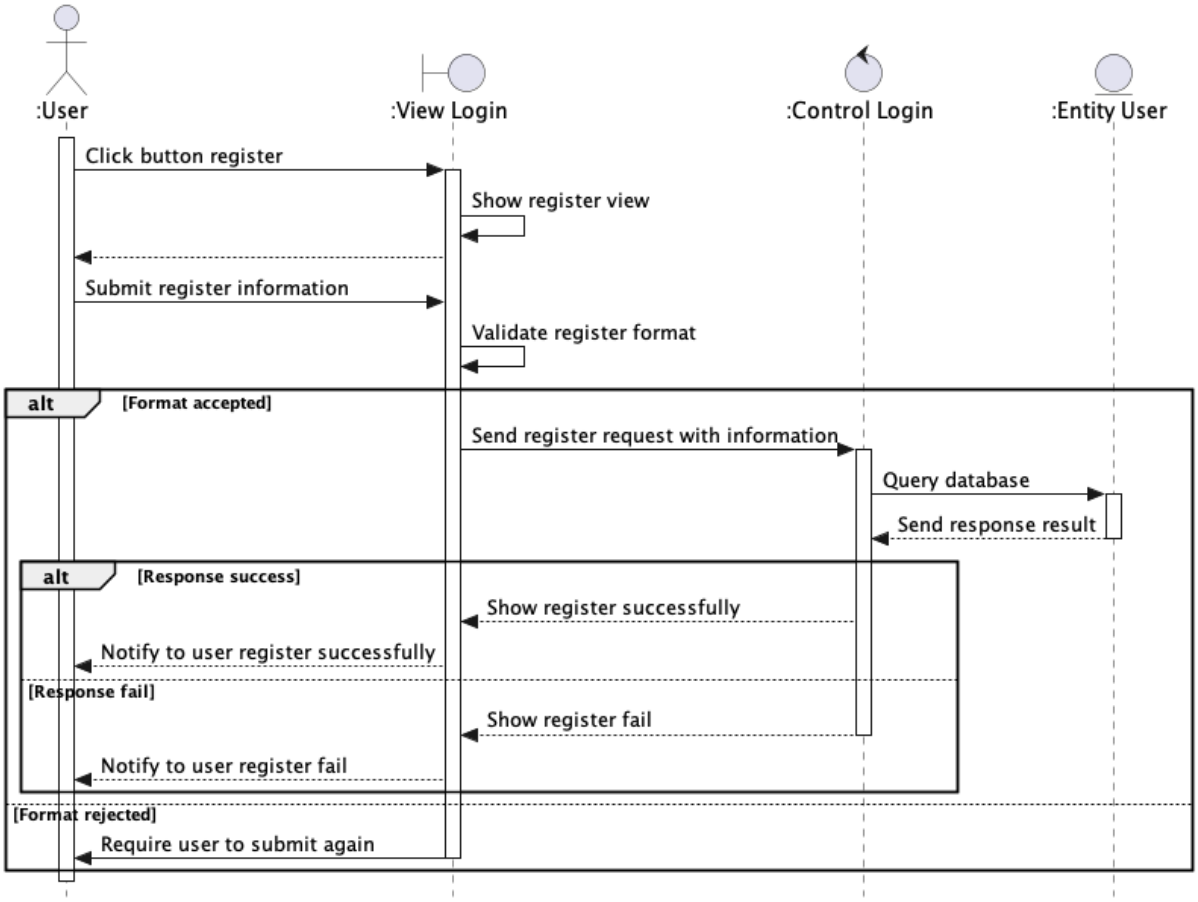
1.2.3.3 Website cho admin



Hình 1.8 Sơ đồ khối phần mềm của website quản trị

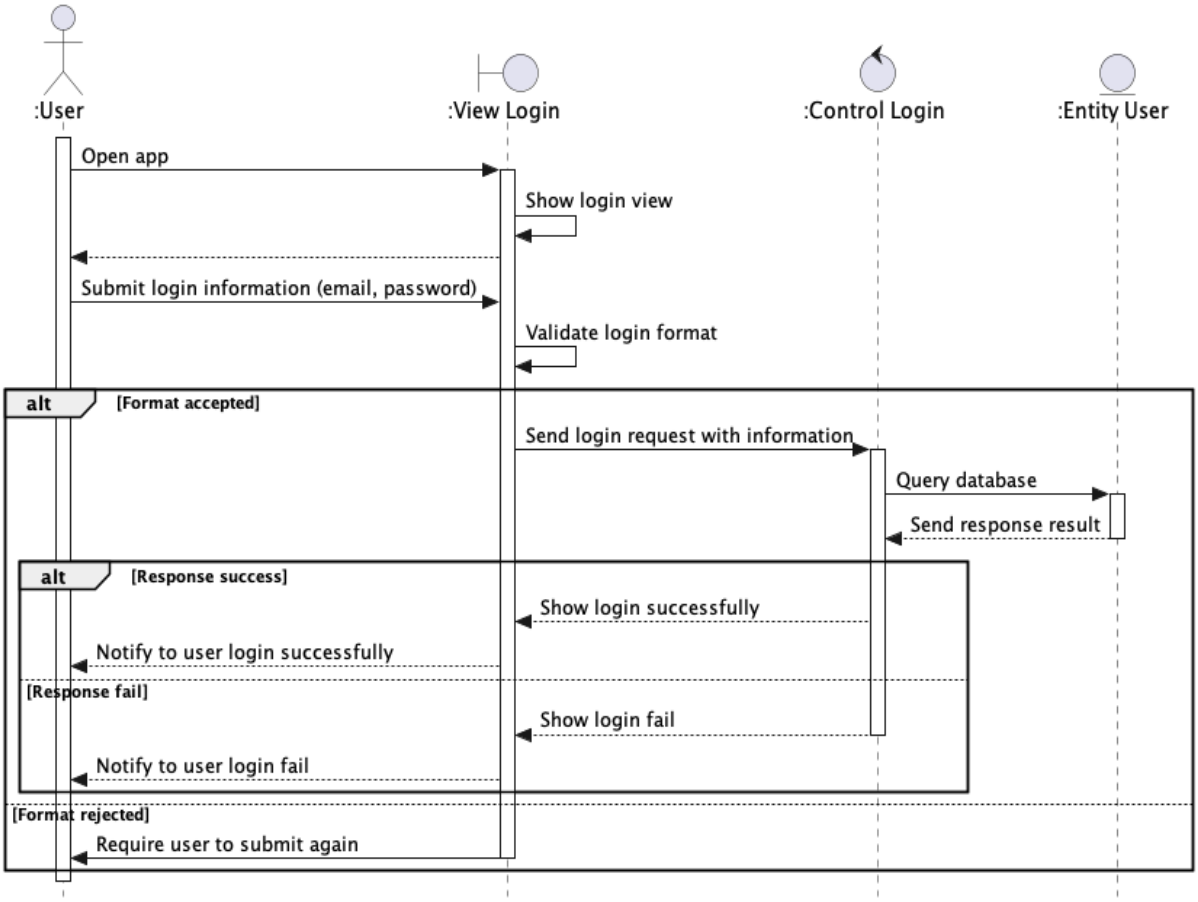
1.2.4 Sơ đồ tuần tự

1.2.4.1 Sơ đồ tuần tự chức năng đăng ký



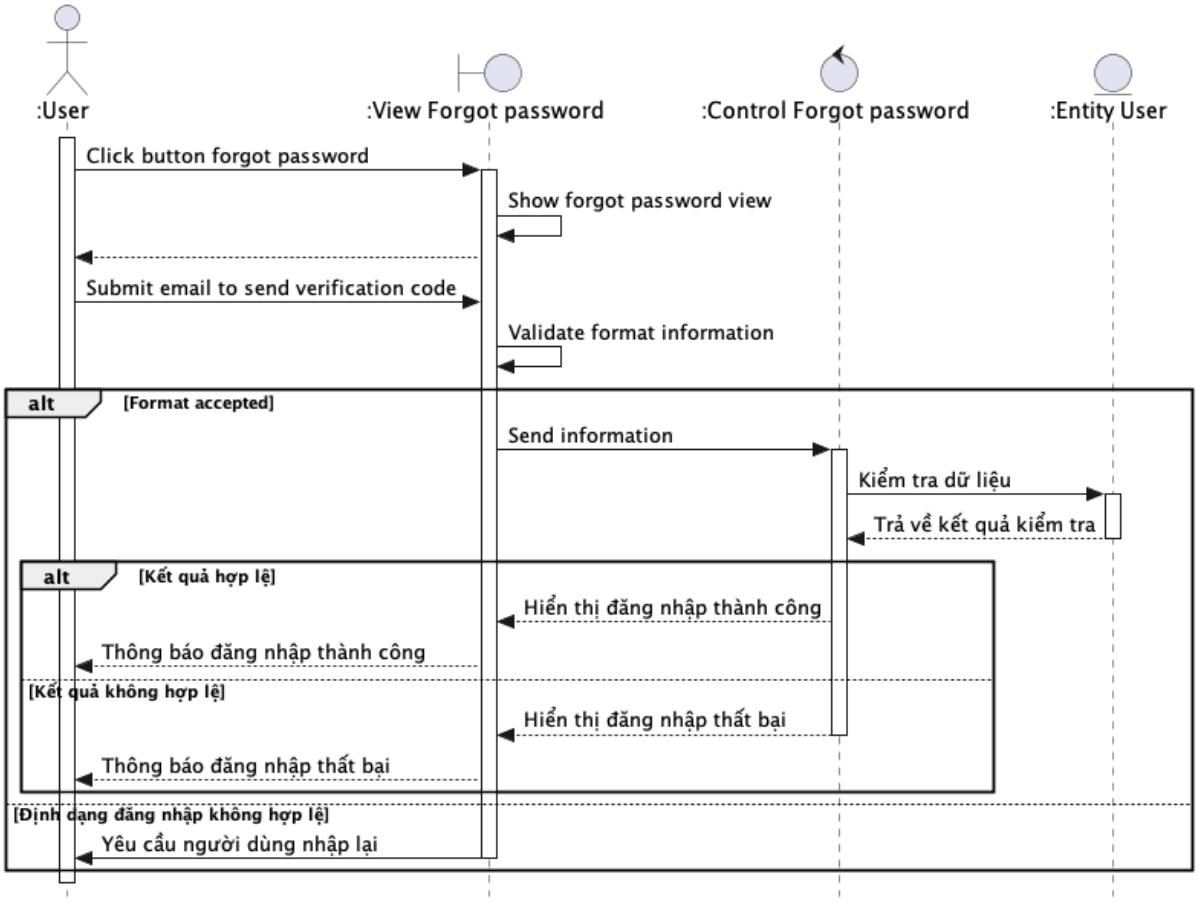
Hình 1.9 Sơ đồ tuần tự chức năng đăng ký trên App

1.2.4.2 Sơ đồ tuần tự chức năng đăng nhập



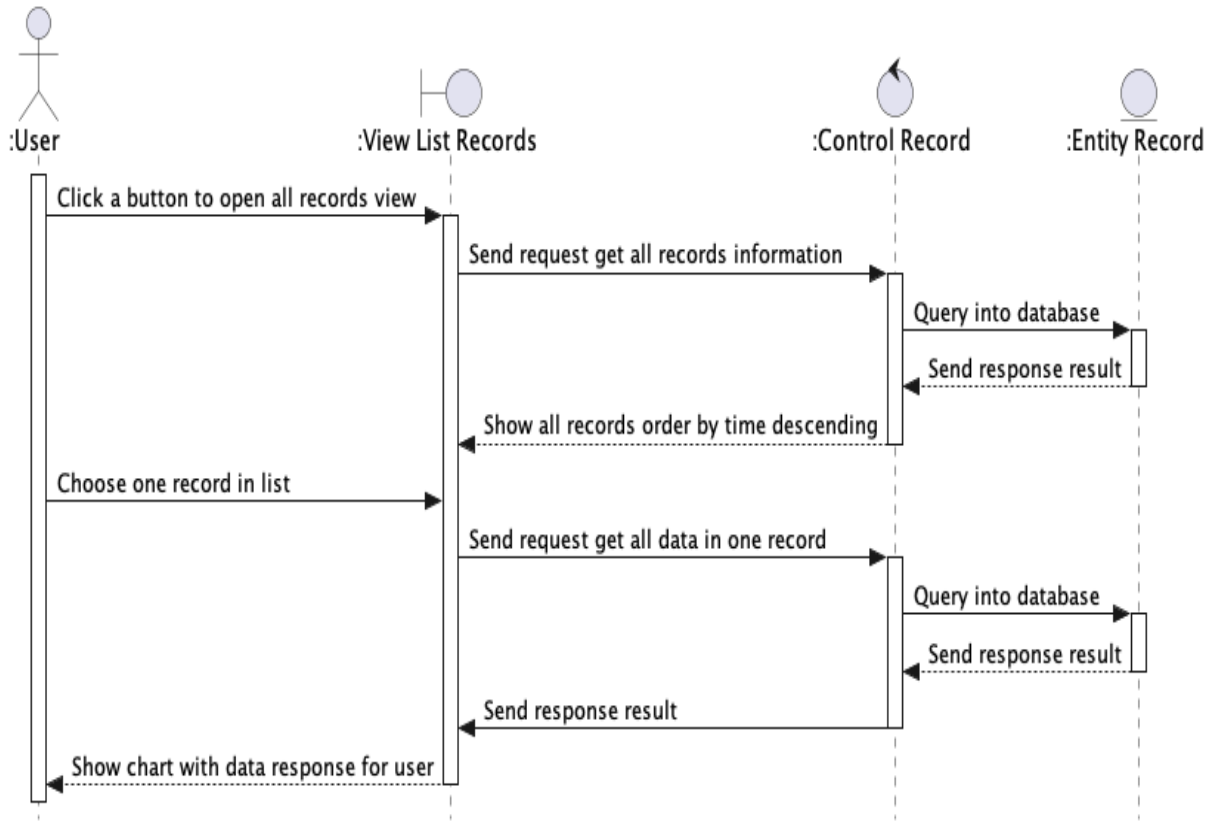
Hình 1.10 Sơ đồ tuần tự chức năng đăng nhập trên App

1.2.4.3 Sơ đồ tuần tự chức năng quên mật khẩu



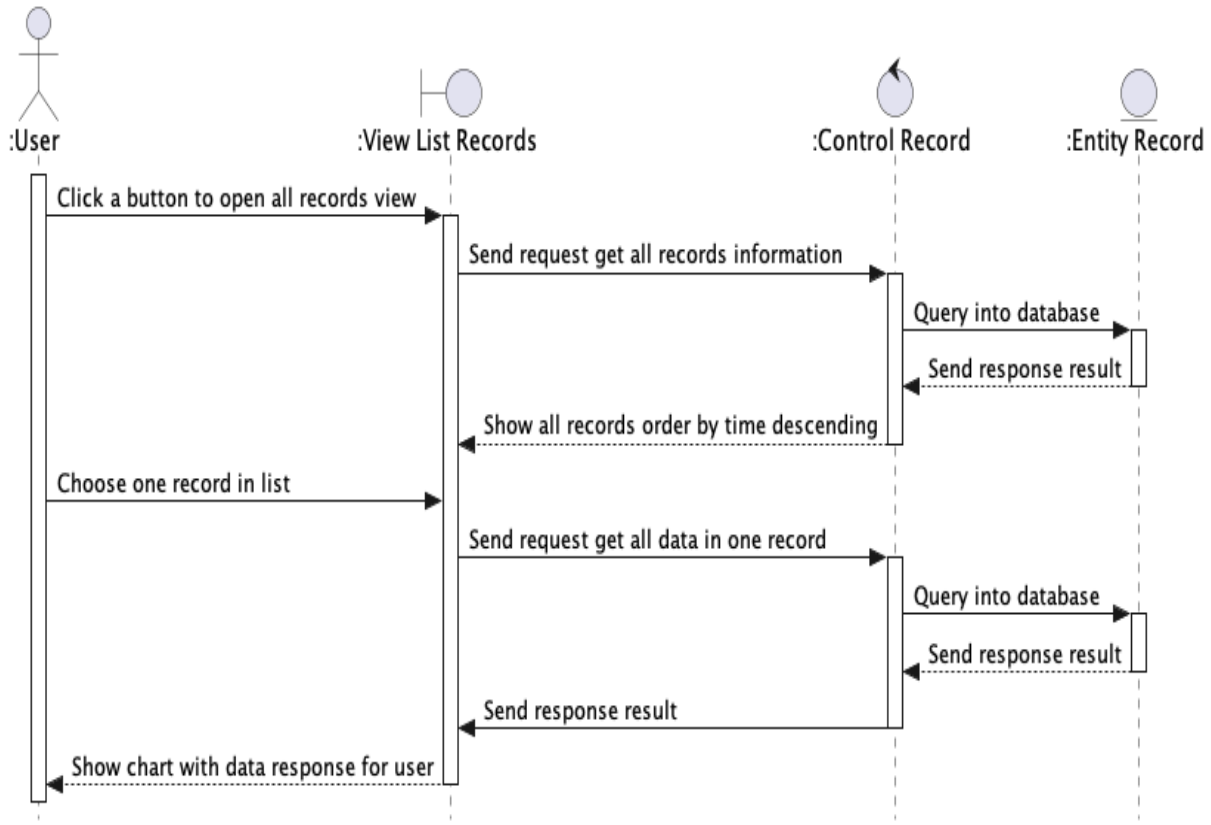
Hình 1.11 Sơ đồ tuần tự chức năng quên mật khẩu trên App

1.2.4.4 Sơ đồ tuần tự chức năng xem lịch sử các lần đo



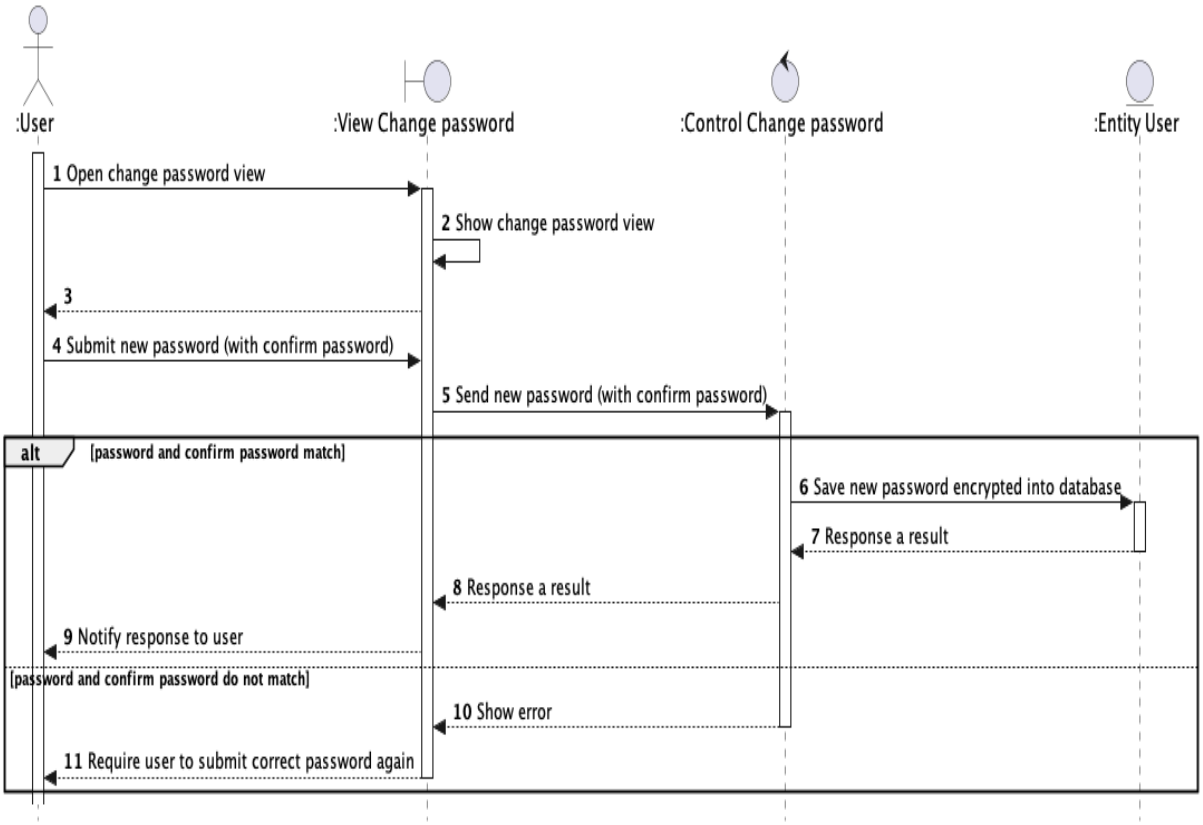
Hình 1.12 Sơ đồ tuần tự chức năng xem lịch sử các lần đo trên App

1.2.4.5 Sơ đồ tuần tự chức năng xem thay đổi thông tin cá nhân



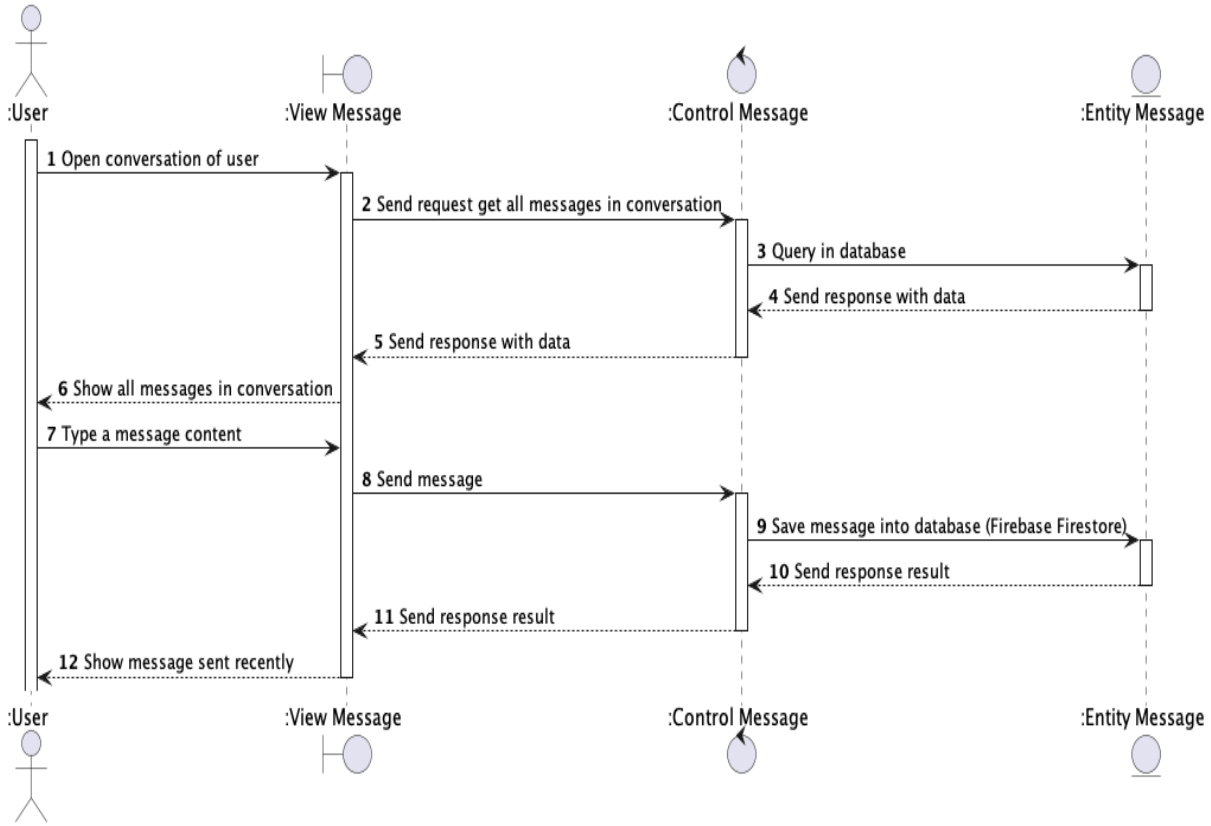
Hình 1.13 Sơ đồ tuần tự chức năng xem thay đổi thông tin cá nhân trên App

1.2.4.6 Sơ đồ tuần tự chức năng đổi mật khẩu



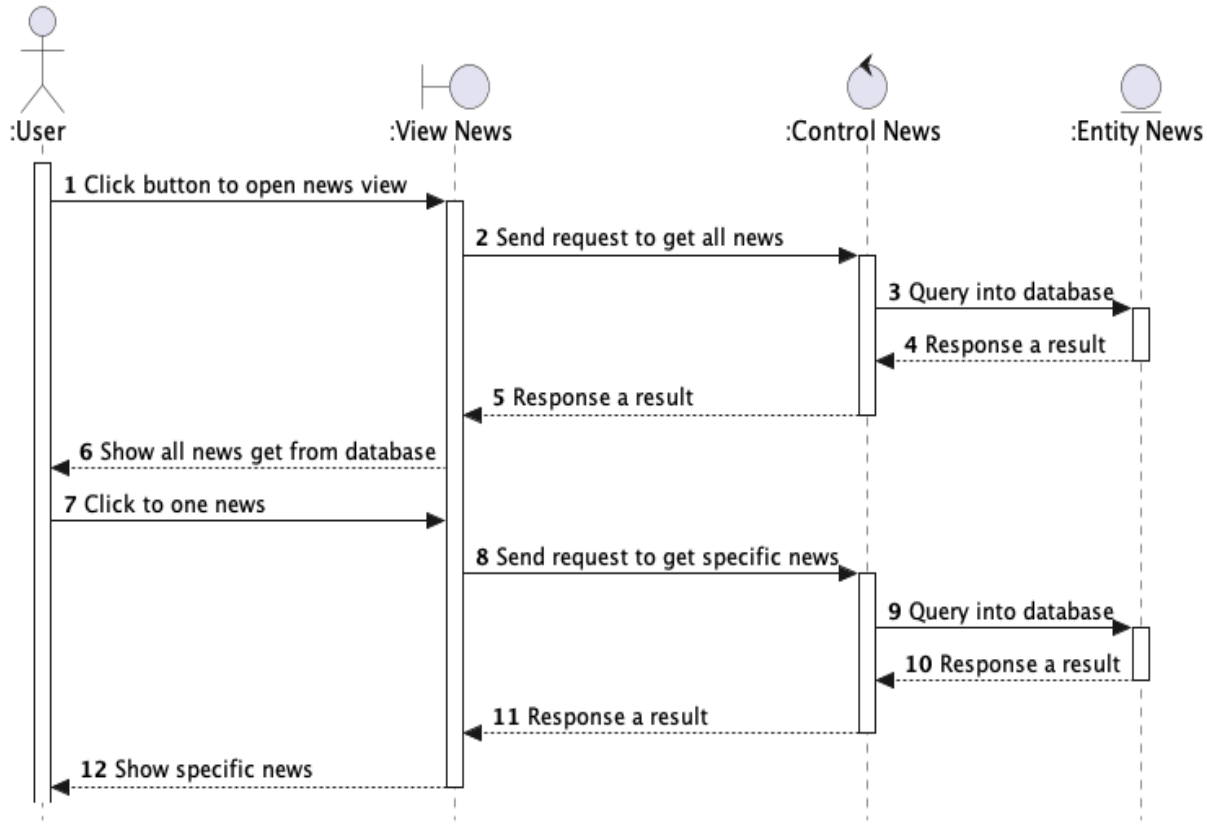
Hình 1.14 Sơ đồ tuần tự chức năng đổi mật khẩu trên App

1.2.4.7 Sơ đồ tuần tự chức năng xem/gửi tin nhắn



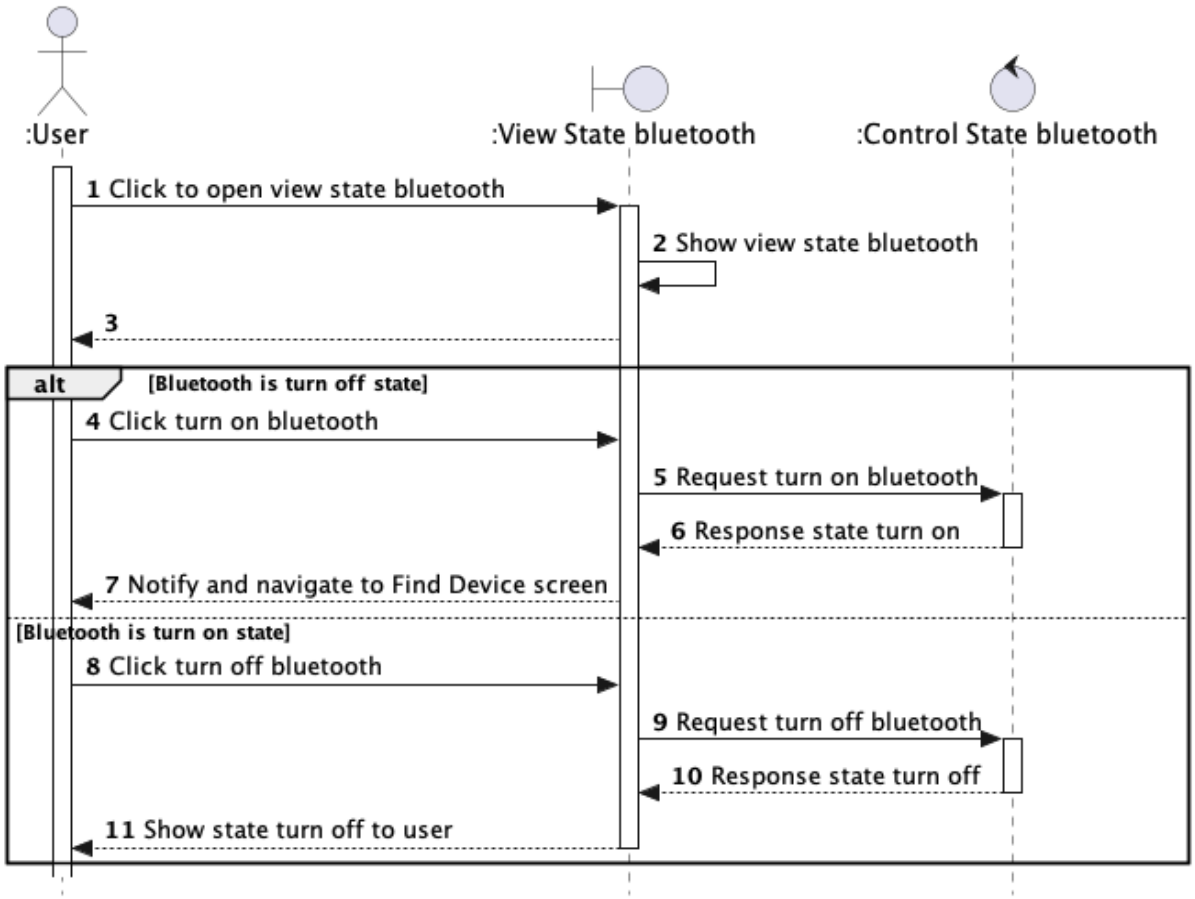
Hình 1.15 Sơ đồ tuần tự chức năng xem/gửi tin nhắn trên App

1.2.4.8 Sơ đồ tuần tự chức năng xem bài đăng tin tức



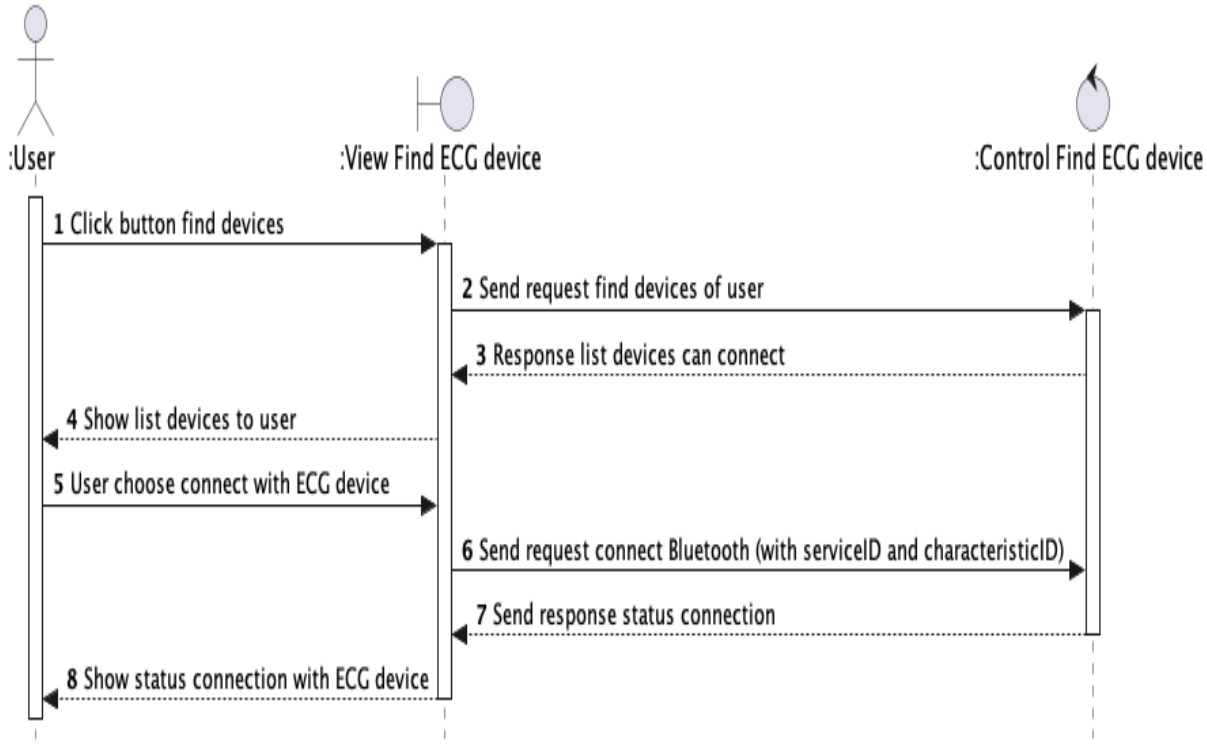
Hình 1.16 Sơ đồ tuần tự chức năng xem bài đăng tin tức trên App

1.2.4.9 Sơ đồ tuần tự chức năng bật/tắt Bluetooth



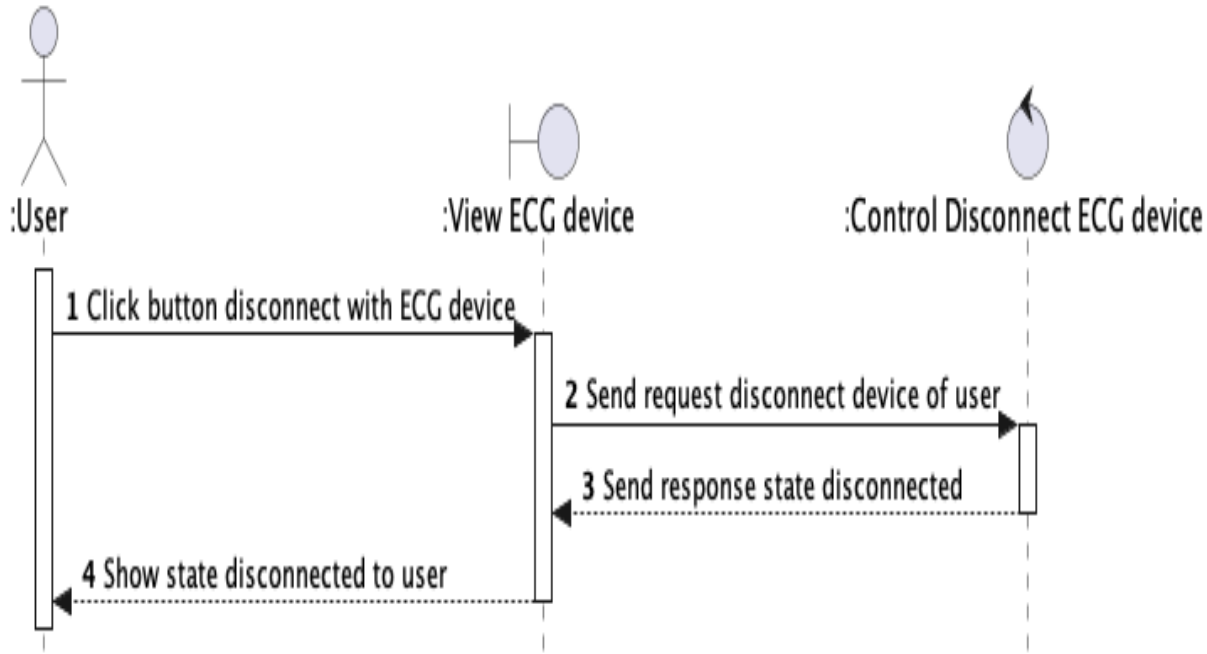
Hình 1.17 Sơ đồ tuần tự chức năng bật/tắt Bluetooth trên App

1.2.4.10 Sơ đồ tuần tự chức năng kết nối Bluetooth với thiết bị đo điện tim



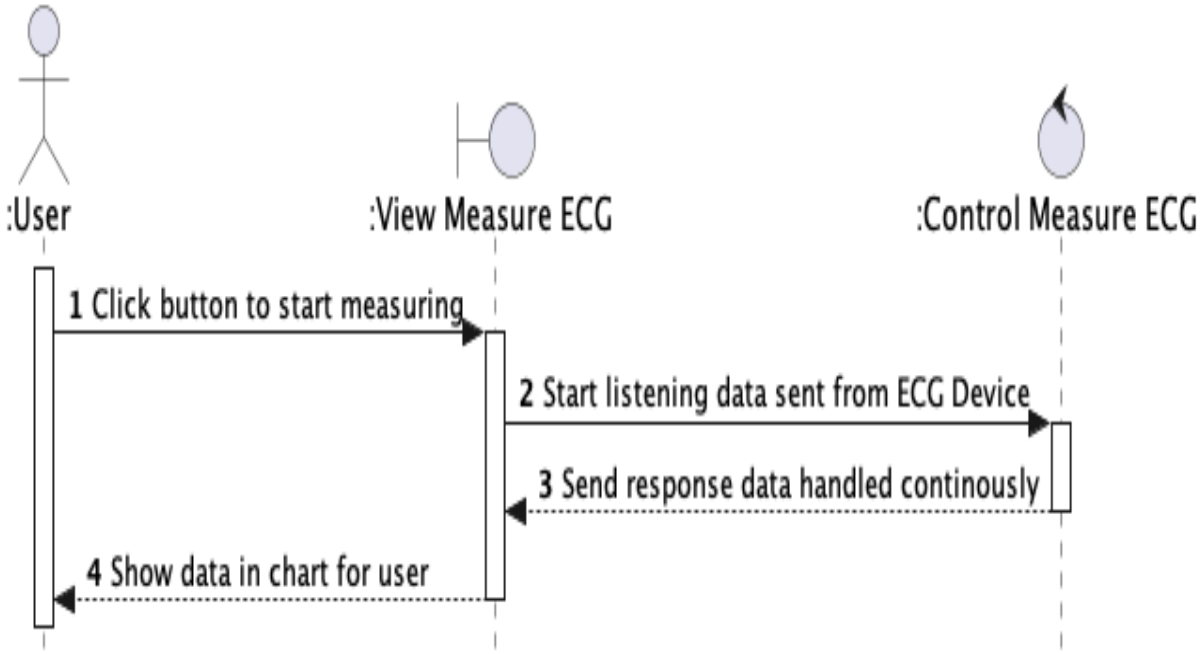
Hình 1.18 Sơ đồ tuần tự chức năng kết nối Bluetooth với thiết bị đo điện tim trên App

1.2.4.11 Sơ đồ tuần tự chức năng ngắt kết nối Bluetooth với thiết bị đo điện tim



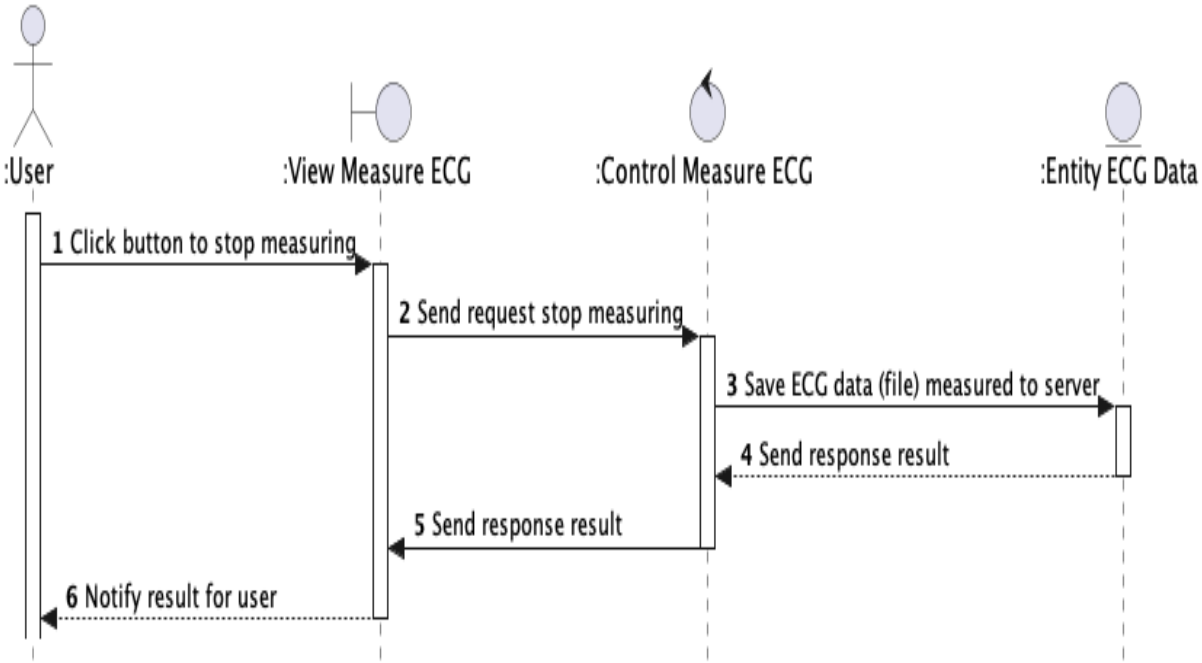
Hình 1.19 Sơ đồ tuần tự chức năng ngắt kết nối Bluetooth với thiết bị đo điện tim trên App

1.2.4.12 Sơ đồ tuần tự chức năng tiến hành đo điện tim



Hình 1.20 Sơ đồ tuần tự chức năng tiến hành đo điện tim trên App

1.2.4.13 Sơ đồ tuần tự chức năng kết thúc quá trình đo điện tim



Hình 1.21 Sơ đồ tuần tự chức năng kết thúc quá trình đo điện tim trên App

CHƯƠNG 2. THIẾT KẾ CHI TIẾT HỆ THỐNG

2.1 Công nghệ sử dụng

2.1.1 Ứng dụng

2.1.2 Server và Website

2.1.2.1 NodeJs

Node.js là một nền tảng phát triển ứng dụng mã nguồn mở cực kỳ mạnh mẽ và đa dạng, giúp các nhà phát triển xây dựng các ứng dụng web và dịch vụ server-side với hiệu suất cao và khả năng mở rộng linh hoạt. Được ra đời vào năm 2009, Node.js đã nhanh chóng trở thành một trong những công nghệ phổ biến nhất trong cộng đồng lập trình viên và được ứng dụng rộng rãi trong các dự án phức tạp cũng như các ứng dụng thời gian thực.

Một trong những điểm mạnh nổi bật của Node.js là sự hỗ trợ tốt cho kiến trúc không đồng bộ (asynchronous architecture). Bản chất của kiến trúc này là khả năng xử lý đa luồng (multithreading) không đồng bộ, giúp ứng dụng có thể xử lý nhiều yêu cầu cùng một lúc mà không phải chờ đợi hoàn thành từng yêu cầu trước. Điều này cho phép Node.js xử lý các yêu cầu I/O (Input/Output) nặng như truy vấn cơ sở dữ liệu, đọc/ghi dữ liệu từ tập tin, hay gửi và nhận các yêu cầu HTTP một cách hiệu quả và nhanh chóng. Sự không đồng bộ cũng giúp tăng hiệu suất cho ứng dụng, đồng thời tiết kiệm tài nguyên hệ thống.

Node.js được xây dựng dựa trên JavaScript, ngôn ngữ lập trình phổ biến và có tính linh hoạt cao. Sự hòa quyện giữa Node.js và JavaScript cho phép phát triển viên sử dụng cùng một ngôn ngữ cho cả phía client và phía server, giúp đơn giản hóa việc phát triển và duy trì mã nguồn. Ngoài ra, cộng đồng phát triển JavaScript rất lớn và đầy đủ tài nguyên học tập và hỗ trợ trực tuyến, điều này làm cho việc học và sử dụng Node.js trở nên dễ dàng và thuận tiện.

Vì Node.js là mã nguồn mở, nên nó rất linh hoạt và có khả năng tùy chỉnh cao. Phát triển viên có thể sửa đổi và mở rộng các tính năng của Node.js tùy theo nhu cầu của dự án một cách dễ dàng. Hơn nữa, Node.js có cộng đồng lớn và năng động, luôn cập nhật và cải tiến mã nguồn để đáp ứng yêu cầu của các dự án phức tạp và đa dạng. Cộng đồng Node.js cũng cung cấp rất nhiều các thư viện và module mở rộng, giúp tăng tính năng và hiệu suất của ứng dụng.

Một trong những ưu điểm nổi bật của Node.js là hỗ trợ đa nền tảng (cross-platform), cho phép chạy ứng dụng trên nhiều hệ điều hành khác nhau như Windows, macOS,

Linux, và nhiều nền tảng khác. Điều này giúp đơn giản hóa quá trình triển khai và triển khai ứng dụng trên nhiều môi trường mà không cần thay đổi mã nguồn.

Một ưu điểm khác của Node.js là hỗ trợ tốt cho việc xây dựng các ứng dụng thời gian thực (real-time applications). Điều này rất hữu ích trong việc xây dựng các ứng dụng trò chơi trực tuyến, ứng dụng trò chuyện (chat applications), hay ứng dụng giao tiếp thời gian thực. Cơ chế WebSocket được tích hợp sẵn trong Node.js giúp kết nối và truyền thông dữ liệu giữa server và client một cách liên tục và nhanh chóng.

Bên cạnh đó, Node.js cũng hỗ trợ các thư viện và giao thức mạng như HTTP, HTTPS, TCP, và UDP, cho phép phát triển viên xây dựng các ứng dụng mạng phức tạp và mạnh mẽ. Node.js cũng cung cấp khả năng tạo máy chủ đơn giản và hiệu quả, giúp xây dựng các ứng dụng web phức tạp với khả năng mở rộng linh hoạt.

Trong việc lưu trữ dữ liệu, Node.js hỗ trợ cơ chế xử lý dữ liệu không đồng bộ, giúp tối ưu hóa hiệu suất của ứng dụng trong việc truy xuất và lưu trữ dữ liệu vào cơ sở dữ liệu. Node.js cũng hỗ trợ các cơ sở dữ liệu phổ biến như MySQL, PostgreSQL, MongoDB, và Redis, cho phép phát triển viên lựa chọn cơ sở dữ liệu phù hợp với yêu cầu của dự án.

Ngoài ra, Node.js còn hỗ trợ các tiện ích cho việc phát triển ứng dụng như gỡ lỗi (debugging), kiểm thử (testing), và quản lý gói (package management) thông qua npm (Node Package Manager). Việc sử dụng các công cụ và thư viện này giúp đơn giản hóa việc phát triển và nâng cao chất lượng của mã nguồn.

Trong quá trình triển khai ứng dụng, Node.js hỗ trợ dễ dàng tích hợp với các công cụ triển khai (deployment tools) như Docker và Kubernetes, giúp đơn giản hóa quá trình triển khai và vận hành ứng dụng trên các môi trường khác nhau.

Tổng kết lại, Node.js là một nền tảng phát triển ứng dụng đa dạng và mạnh mẽ, với nhiều ưu điểm nổi trội như kiến trúc không đồng bộ, tích hợp với JavaScript, đa nền tảng, hỗ trợ ứng dụng thời gian thực, và cộng đồng lớn hỗ trợ phong phú. Với những lợi ích đáng kể này, Node.js đã trở thành một lựa chọn ưu việt cho việc xây dựng các ứng dụng web và dịch vụ server-side hiệu quả và mạnh mẽ.

2.1.2.2 Javascript

JavaScript là một ngôn ngữ lập trình thông dịch, đa mục đích và được sử dụng phổ biến trên web. Nó ra đời vào năm 1995 bởi Brendan Eich, khi còn là một ngôn ngữ lập trình đơn giản để cung cấp tính năng tương tác và hiệu ứng động cho trang web. Tuy nhiên, từ đó đến nay, JavaScript đã trở thành một trong những ngôn ngữ lập trình quan trọng nhất trên thế giới, vượt qua giới hạn trang web và mở rộng sang nhiều lĩnh vực phát triển phần mềm khác nhau.

Một trong những yếu tố quan trọng đóng góp vào sự phổ biến của JavaScript là việc nó được tích hợp một cách tự nhiên và mạnh mẽ với các trình duyệt web. Trước khi JavaScript xuất hiện, việc tạo ra hiệu ứng động và tương tác trên trang web đòi hỏi sự hỗ trợ của các ngôn ngữ lập trình phía server như PHP hoặc các công nghệ khác. JavaScript đã định hình lại cách các ứng dụng web tương tác và hiển thị nội dung, giúp chúng trở nên đa dạng và mạnh mẽ hơn.

JavaScript là một ngôn ngữ linh hoạt và dễ tiếp cận, cho phép người phát triển xây dựng các ứng dụng phong phú và phức tạp với giao diện tương tác, kiểu dáng đẹp mắt và tính năng đa dạng. Ngôn ngữ này hỗ trợ một loạt các kiểu dữ liệu như số, chuỗi, mảng, đối tượng và hỗ trợ xử lý chuỗi, số học, ngày tháng, và các phép tính logic phức tạp.

Một điểm mạnh của JavaScript là khả năng xử lý các sự kiện (event) và gắn kết chúng vào các thành phần trên trang web. Sự kiện là những hành động như nhấp chuột, nhập liệu từ bàn phím, hoặc thay đổi trạng thái của trang web. JavaScript cho phép người phát triển xử lý các sự kiện này và thay đổi nội dung hoặc hành vi của trang web một cách linh hoạt, đáp ứng tương tác của người dùng.

Ngoài ra, JavaScript còn hỗ trợ các khái niệm cấu trúc dữ liệu và kiểu dữ liệu linh hoạt như đối tượng (object), mảng (array), và chuỗi (string). Điều này cho phép người phát triển xây dựng các ứng dụng phức tạp và có tính mô đun cao, giúp quản lý mã nguồn dễ dàng và giảm thiểu lỗi trong quá trình phát triển.

Một trong những tính năng đáng chú ý của JavaScript là khả năng thực hiện các yêu cầu mạng bất đồng bộ (asynchronous) thông qua việc sử dụng các API như XML-HttpRequest (XHR) hay Fetch API. Điều này cho phép trang web tương tác với các dịch vụ web và cơ sở dữ liệu một cách hiệu quả mà không làm trì hoãn trình duyệt.

JavaScript cũng hỗ trợ việc tạo ra các ứng dụng web động, đặc biệt là những ứng dụng đơn trang (single-page applications - SPA). SPA là mô hình phát triển ứng dụng web mà trang web chỉ tải một lần duy nhất khi người dùng truy cập, và sau đó các tương tác tiếp theo được thực hiện một cách bất đồng bộ thông qua JavaScript để cập nhật nội dung và thay đổi giao diện mà không cần tải lại trang. Điều này giúp tăng tốc độ và trải nghiệm người dùng.

Một yếu tố quan trọng khác là cộng đồng JavaScript rất đông đảo và nhiệt tình. Có rất nhiều người dùng và nhà phát triển trên toàn thế giới đóng góp và chia sẻ mã nguồn, thư viện, framework và các tài liệu học tập trực tuyến. Điều này làm cho việc học và sử dụng JavaScript trở nên dễ dàng và thuận tiện, đồng thời giúp phát triển viên giải quyết các vấn đề và thách thức trong quá trình xây dựng ứng dụng.

Một trong những hướng phát triển quan trọng của JavaScript là sự phát triển của các framework và thư viện mã nguồn mở như React, Angular, và Vue.js. Các framework

này giúp người phát triển xây dựng các ứng dụng web phức tạp và mạnh mẽ một cách dễ dàng và hiệu quả hơn, đồng thời đảm bảo tính tương thích và bảo mật.

Tuy nhiên, JavaScript cũng có một số hạn chế như việc thực hiện mã nguồn phức tạp có thể dẫn đến các vấn đề bảo mật và hiệu suất. Để giải quyết vấn đề này, người phát triển cần phải tuân thủ các nguyên tắc phát triển an toàn và tối ưu hóa mã nguồn.

Tổng kết lại, JavaScript là một ngôn ngữ lập trình linh hoạt, mạnh mẽ và phổ biến, có nhiều ưu điểm vượt trội như tích hợp tốt với trình duyệt, đa dạng kiểu dữ liệu, khả năng thực hiện bất đồng bộ, và cộng đồng phát triển mạnh mẽ. JavaScript đã định hình lại cách các ứng dụng web tương tác và được sử dụng rộng rãi trong nhiều lĩnh vực phát triển phần mềm. Sự phát triển của các framework và thư viện JavaScript giúp người phát triển xây dựng các ứng dụng web phức tạp và mạnh mẽ một cách dễ dàng và hiệu quả hơn.

2.1.2.3 AdminJS

AdminJS là một framework mã nguồn mở được thiết kế để hỗ trợ phát triển các trang quản lý (admin dashboard) dễ dàng và nhanh chóng. Với AdminJS, người phát triển có thể xây dựng các giao diện quản lý phức tạp và mạnh mẽ một cách dễ dàng và hiệu quả, giúp quản lý và điều hành hệ thống, ứng dụng hoặc dịch vụ một cách thuận tiện và hiệu quả hơn.

Một trong những ưu điểm nổi bật của AdminJS là tích hợp các tính năng và chức năng phong phú sẵn có, giúp giảm thiểu thời gian và công sức trong việc xây dựng giao diện quản lý. AdminJS cung cấp các thành phần giao diện (UI components) sẵn có cho việc quản lý các bảng dữ liệu (data tables), hiển thị các biểu đồ thống kê, tạo các biểu mẫu nhập liệu (form fields) và nhiều tính năng khác. Nhờ vậy, người phát triển không cần phải viết mã nguồn từ đầu mà vẫn có thể xây dựng các trang quản lý đẹp mắt và chuyên nghiệp.

AdminJS hỗ trợ tích hợp với nhiều loại cơ sở dữ liệu phổ biến như MongoDB, PostgreSQL, MySQL, và SQLite, giúp tương tác và quản lý dữ liệu một cách dễ dàng. Ngoài ra, nó cũng hỗ trợ tích hợp với các framework và thư viện phổ biến như Express.js, Nest.js, và các thư viện mã nguồn mở khác. Tích hợp với các công nghệ này giúp AdminJS trở nên linh hoạt và dễ dàng tích hợp vào các dự án sẵn có.

Một tính năng quan trọng của AdminJS là khả năng tùy chỉnh và mở rộng. Người phát triển có thể tùy chỉnh các thành phần giao diện sẵn có để phù hợp với yêu cầu và phong cách của dự án. Điều này giúp tạo ra các trang quản lý có giao diện độc đáo và phù hợp với thương hiệu của dự án. Ngoài ra, người phát triển cũng có thể mở rộng và thêm các tính năng tùy chỉnh theo yêu cầu của dự án một cách dễ dàng.

AdminJS cũng hỗ trợ tích hợp các tính năng bảo mật và xác thực. Người phát triển có thể dễ dàng thiết lập và quản lý các quyền truy cập và vai trò người dùng trên các trang quản lý. Điều này giúp bảo mật và bảo vệ dữ liệu quan trọng khỏi việc truy cập trái phép. AdminJS cũng hỗ trợ tích hợp với các công nghệ bảo mật như JSON Web Token (JWT), Passport.js và các cơ chế xác thực khác.

Một điểm mạnh nữa của AdminJS là sự hỗ trợ mạnh mẽ từ cộng đồng. AdminJS có một cộng đồng đông đảo và năng động, luôn cập nhật và đóng góp cho mã nguồn để nâng cao tính năng và hiệu suất của framework. Cộng đồng cũng cung cấp rất nhiều tài liệu học tập và hỗ trợ trực tuyến, giúp người phát triển học hỏi và giải quyết các vấn đề trong quá trình xây dựng giao diện quản lý.

Ngoài ra, AdminJS còn hỗ trợ tích hợp với các công cụ phát triển phổ biến như VSCode, Sublime Text, và Atom, giúp tăng cường trải nghiệm phát triển và giảm thiểu lỗi trong quá trình phát triển. Nó cũng hỗ trợ tích hợp với các công cụ quản lý mã nguồn như Git và GitHub, giúp quản lý và kiểm soát phiên bản mã nguồn một cách hiệu quả.

Tóm lại, AdminJS là một framework mạnh mẽ và linh hoạt để xây dựng các trang quản lý (admin dashboard) dễ dàng và nhanh chóng. Với các tính năng và chức năng phong phú, tích hợp dễ dàng với nhiều loại cơ sở dữ liệu và khả năng tùy chỉnh mạnh mẽ, AdminJS giúp người phát triển xây dựng các giao diện quản lý chuyên nghiệp và mạnh mẽ một cách dễ dàng và hiệu quả. Sự hỗ trợ mạnh mẽ từ cộng đồng và tích hợp với các công cụ phát triển giúp tăng cường trải nghiệm phát triển và đảm bảo chất lượng của mã nguồn.

2.1.2.4 MySQL

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến và mạnh mẽ, được sử dụng rộng rãi trong nhiều ứng dụng và dự án phát triển phần mềm. MySQL ra đời vào năm 1995, do công ty TcX phát triển và sau đó được mua lại bởi công ty MySQL AB. Hiện nay, MySQL thuộc sở hữu của Oracle Corporation sau khi Oracle mua lại Sun Microsystems - công ty mẹ của MySQL AB vào năm 2010.

MySQL là hệ quản trị cơ sở dữ liệu (DBMS - Database Management System) dựa trên mô hình quan hệ (relational database) và sử dụng ngôn ngữ truy vấn cấu trúc (Structured Query Language - SQL) để thực hiện các thao tác truy vấn dữ liệu. Mô hình quan hệ của MySQL cho phép lưu trữ và quản lý dữ liệu trong các bảng có mối quan hệ với nhau, tạo điều kiện để thực hiện các truy vấn phức tạp và hiệu quả.

MySQL hỗ trợ nhiều loại dữ liệu phổ biến như số, chuỗi, ngày tháng, boolean, và các kiểu dữ liệu đặc biệt khác như JSON và các kiểu dữ liệu không gian (spatial data types). Điều này giúp người dùng lưu trữ và xử lý các loại dữ liệu khác nhau một cách

linh hoạt và hiệu quả.

Một trong những ưu điểm quan trọng của MySQL là hiệu suất cao và khả năng mở rộng linh hoạt. MySQL được tối ưu hóa để xử lý các yêu cầu truy vấn nhanh chóng và hiệu quả, cho phép xử lý hàng triệu hoặc thậm chí hàng tỷ bản ghi một cách mượt mà. Ngoài ra, MySQL cũng hỗ trợ tính năng phân chia dữ liệu (sharding) và nhân bản dữ liệu (replication), giúp tăng khả năng chịu tải và đảm bảo tính sẵn sàng và tin cậy của hệ thống.

MySQL cung cấp các tính năng bảo mật và quản lý người dùng mạnh mẽ. Người quản trị có thể xác định và quản lý các quyền truy cập của người dùng và vai trò của họ trên các bảng và cơ sở dữ liệu. Điều này giúp bảo vệ dữ liệu quan trọng và ngăn chặn truy cập trái phép.

MySQL hỗ trợ các công cụ và giao thức mạng như JDBC, ODBC, và các API kết nối khác, cho phép ứng dụng kết nối và tương tác với cơ sở dữ liệu MySQL từ các ứng dụng phần mềm và trang web khác nhau. MySQL cũng hỗ trợ giao thức kết nối và truy vấn an toàn, giúp đảm bảo tính bảo mật và chống lại các cuộc tấn công từ xa.

MySQL có sự hỗ trợ mạnh mẽ từ cộng đồng và cung cấp rất nhiều tài liệu học tập và hỗ trợ trực tuyến. Cộng đồng MySQL rất đông đảo và năng động, luôn cập nhật và đóng góp cho mã nguồn để nâng cao tính năng và hiệu suất của hệ quản trị cơ sở dữ liệu này.

MySQL cũng hỗ trợ tích hợp với các công cụ phát triển và quản lý mã nguồn như MySQL Workbench, phpMyAdmin và các công cụ khác, giúp quản lý và xem xét dữ liệu một cách thuận tiện và hiệu quả.

Một điểm đáng chú ý nữa của MySQL là tính tương thích với nhiều hệ điều hành và nền tảng khác nhau. MySQL có phiên bản dành cho Windows, macOS và các hệ điều hành Linux khác nhau, giúp người dùng lựa chọn và triển khai phù hợp với môi trường họ sử dụng.

Tóm lại, MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ và phổ biến, được sử dụng rộng rãi trong nhiều ứng dụng và dự án phát triển phần mềm. Với tính năng hiệu suất cao, khả năng mở rộng linh hoạt, tính bảo mật và quản lý người dùng, sự hỗ trợ từ cộng đồng và tích hợp với nhiều công cụ phát triển, MySQL đã trở thành một lựa chọn ưu việt cho việc quản lý và xử lý cơ sở dữ liệu.

2.1.2.5 Postman

Postman là một ứng dụng máy tính được sử dụng phổ biến trong quá trình phát triển và kiểm thử các dịch vụ web (web service) và các API (Application Programming Interface). Được phát triển bởi công ty Postman Inc., ứng dụng này cung cấp một giao

diện đơn giản và dễ sử dụng để gửi các yêu cầu HTTP (HTTP request) và nhận các phản hồi (response) từ các API. Postman giúp người phát triển dễ dàng thử nghiệm và kiểm tra tính năng, hiệu năng và bảo mật của các dịch vụ web và API, từ đó đảm bảo chất lượng và độ tin cậy của ứng dụng.

Một trong những ưu điểm nổi bật của Postman là khả năng gửi các yêu cầu HTTP một cách linh hoạt và dễ dàng. Người dùng có thể tùy chỉnh các yêu cầu HTTP bằng cách thêm các thông số và tham số, cũng như xác định các phương thức HTTP như GET, POST, PUT, DELETE, và nhiều phương thức khác. Điều này giúp người phát triển kiểm tra các chức năng khác nhau của API và dịch vụ web một cách hiệu quả.

Postman cũng hỗ trợ quản lý các môi trường (environments) khác nhau, cho phép người dùng cấu hình các giá trị môi trường như URL, token xác thực, hay các giá trị động khác. Nhờ vậy, người dùng có thể thử nghiệm các yêu cầu API trên nhiều môi trường khác nhau một cách dễ dàng, từ đó giả lập các trường hợp sử dụng và kiểm tra tính ổn định của API.

Một tính năng mạnh mẽ khác của Postman là khả năng tự động hóa các bước kiểm thử và thực hiện các bộ kiểm thử (test suite). Người dùng có thể viết các mã kiểm thử (test script) bằng ngôn ngữ JavaScript để kiểm tra phản hồi từ API và đánh giá tính đúng đắn của dữ liệu trả về. Điều này giúp tăng cường tính tin cậy của các bài kiểm thử và giảm thiểu nguy cơ sai sót từ phía người dùng.

Postman hỗ trợ tích hợp với các công cụ và dịch vụ phổ biến khác như GitHub, Jenkins, và các công cụ kiểm thử liên quan khác. Điều này giúp người phát triển tích hợp Postman vào quy trình phát triển phần mềm tự động một cách dễ dàng, từ việc kiểm tra mã nguồn đến việc chạy các bài kiểm thử tự động.

Postman cung cấp giao diện người dùng thân thiện và trực quan, giúp người dùng dễ dàng thao tác và nắm bắt các tính năng của ứng dụng. Nó cũng hỗ trợ nhiều loại dữ liệu và định dạng như JSON, XML, và form data, giúp người dùng kiểm thử các yêu cầu API đa dạng và phong phú.

Ngoài ra, Postman còn hỗ trợ tích hợp với Postman Cloud, một dịch vụ lưu trữ dữ liệu trực tuyến cho phép người dùng chia sẻ và quản lý các bài kiểm thử và dữ liệu API trực tuyến. Điều này giúp tăng cường sự cộng tác và quản lý dự án trong quá trình phát triển.

Postman đã trở thành một công cụ quan trọng và không thể thiếu trong quá trình phát triển và kiểm thử các ứng dụng và dịch vụ web. Sự dễ dàng sử dụng, tích hợp và tự động hóa giúp tăng cường năng suất và chất lượng của quá trình phát triển, đồng thời giảm thiểu nguy cơ lỗi và tối ưu hóa hiệu suất của ứng dụng.

2.2 Thiết kế cơ sở dữ liệu

2.2.1 Xây dựng mô hình thực thể liên kết

Xác định các thực thể và thuộc tính:

Bảng 2.1 Bảng thực thể và thuộc tính

Thực thể	Thuộc tính
Người dùng	ID người dùng, Mật khẩu, Email, Tên, Ngày sinh, Số điện thoại, Quyền
ECG Record	ID bản ghi ECG, ID người dùng, ID thiết bị, Đường dẫn lưu trữ dữ liệu, Thời gian bắt đầu, Thời gian kết thúc, Loại cảm biến
Danh mục tin tức	ID danh mục tin tức, Tên danh mục tin tức, Mô tả danh mục tin tức
Tin tức	ID tin tức, Tiêu đề, Nội dung, ID danh mục tin tức, Tác giả, Đường dẫn, Đường dẫn hình ảnh
Phân công bệnh nhân - bác sĩ	ID phân công, ID bệnh nhân, ID bác sĩ, Ngày bắt đầu
Mã thông báo đặt lại	ID mã thông báo, ID người dùng, Mã thông báo, Thời gian hết hạn
Phiên đăng nhập	ID phiên đăng nhập, ID người dùng, Mã phiên đăng nhập, Thời gian hết hạn
Thiết bị	ID thiết bị, Tên thiết bị

2.2.2 Chuyển mô hình thực thể liên kết sang mô hình quan hệ

Các bảng trong mô hình thực thể liên kết đã được chuyển thành các bảng trong mô hình quan hệ, với mỗi bảng đại diện cho một thực thể và các mối quan hệ được biểu diễn bằng các khóa ngoại.

2.2.3 Mối quan hệ dữ liệu

- Bảng `ecg_record` có mối quan hệ 1-n với bảng `user` thông qua khóa ngoại `user_id`.
- Bảng `news` có mối quan hệ n-1 với bảng `news_category` thông qua khóa ngoại `category_id`.

- Bảng `patient_doctor_assignment` có mối quan hệ n-1 với bảng `user` thông qua khóa ngoại `patient_id` và `doctor_id`.
- Bảng `reset_token` có mối quan hệ n-1 với bảng `user` thông qua khóa ngoại `user_id`.

2.2.4 Chuẩn hoá 3NF

Các bảng đã được thiết kế theo nguyên tắc chuẩn hoá 3NF, vì không có thuộc tính lặp lại và các thuộc tính không phụ thuộc vào một tập hợp con của khóa chính.

2.2.5 Từ điển dữ liệu

Bảng 2.2 Bảng user

Thuộc tính	Kiểu dữ liệu	Mô tả
user_id	INTEGER	Khóa chính của bảng, đại diện cho ID người dùng.
password	STRING	Mật khẩu của người dùng.
email	STRING	Địa chỉ email của người dùng.
name	STRING	Tên của người dùng.
doB	DATE	Ngày sinh của người dùng.
phone_number	STRING	Số điện thoại của người dùng.
role	INTEGER	Quyền của người dùng (0-patient, 1-doctor, 2-admin).
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.3 Bảng ecg_record

Thuộc tính	Kiểu dữ liệu	Mô tả
record_id	INTEGER	Khóa chính của bảng, đại diện cho ID bản ghi ECG.
user_id	INTEGER	Khóa ngoại tham chiếu đến `user_id` trong bảng `user`.
device_id	STRING	ID thiết bị.
data_directory	STRING	Đường dẫn lưu trữ dữ liệu.
start_time	DATE	Thời gian bắt đầu ghi lại ECG.
stop_time	DATE	Thời gian kết thúc ghi lại ECG.
sensor_type	STRING	Loại cảm biến.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.4 Bảng news_category

Thuộc tính	Kiểu dữ liệu	Mô tả
category_id	INTEGER	Khóa chính của bảng, đại diện cho ID danh mục tin tức.
category_name	STRING	Tên danh mục tin tức.
category_description	STRING	Mô tả danh mục tin tức.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.5 Bảng news

Thuộc tính	Kiểu dữ liệu	Mô tả
news_id	INTEGER	Khóa chính của bảng, đại diện cho ID tin tức.
title	STRING	Tiêu đề tin tức.
content	TEXT	Nội dung tin tức.
category_id	INTEGER	Khóa ngoại tham chiếu đến `category_id` trong bảng `news_category`.
author	STRING	Tác giả tin tức.
url	STRING	Đường dẫn tin tức.
image	STRING	Đường dẫn hình ảnh tin tức (có thể là null).
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.6 Bảng patient_doctor_assignment

Thuộc tính	Kiểu dữ liệu	Mô tả
assign_id	INTEGER	Khóa chính của bảng, đại diện cho ID phân công bệnh nhân - bác sĩ.
patient_id	INTEGER	Khóa ngoại tham chiếu đến `user_id` trong bảng `user` (với quyền là bệnh nhân).
doctor_id	INTEGER	Khóa ngoại tham chiếu đến `user_id` trong bảng `user` (với quyền là bác sĩ).
start_date	DATE	Ngày bắt đầu phân công.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.7 Bảng reset_token

Thuộc tính	Kiểu dữ liệu	Mô tả
id	INTEGER	Khóa chính của bảng, đại diện cho ID mã thông báo đặt lại.
user_id	INTEGER	Khóa ngoại tham chiếu đến `user_id` trong bảng `user`.
token	STRING	Mã thông báo đặt lại.
expiration	DATE	Thời gian hết hạn của mã thông báo đặt lại.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

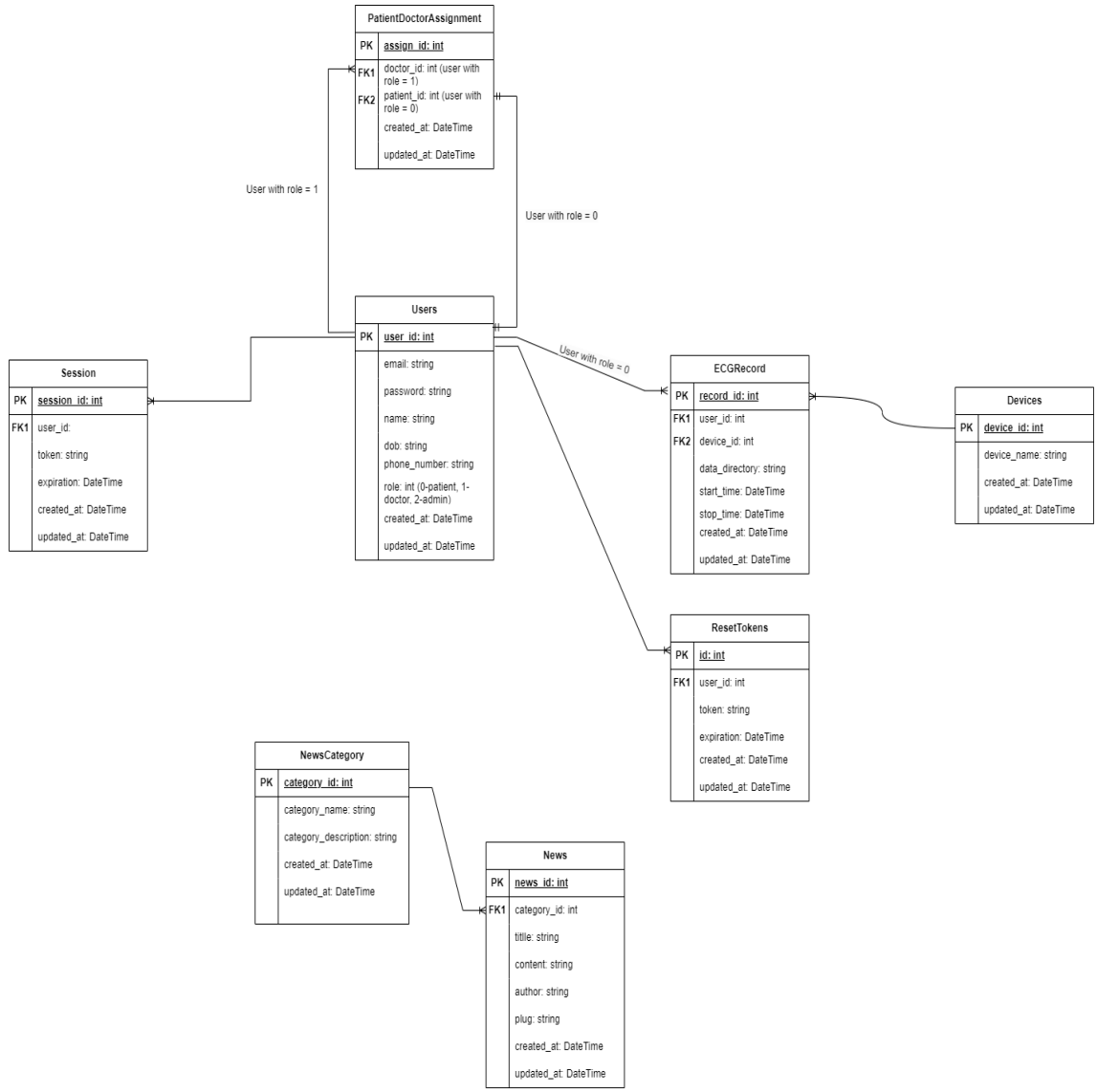
Bảng 2.8 Bảng session

Thuộc tính	Kiểu dữ liệu	Mô tả
session_id	INTEGER	Khóa chính của bảng, đại diện cho ID phiên đăng nhập.
user_id	INTEGER	Khóa ngoại tham chiếu đến `user_id` trong bảng `user`.
token	STRING	Mã phiên đăng nhập.
expiration	DATE	Thời gian hết hạn của phiên đăng nhập.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

Bảng 2.9 Bảng device

Thuộc tính	Kiểu dữ liệu	Mô tả
device_id	INTEGER	Khóa chính của bảng, đại diện cho ID thiết bị.
device_name	STRING	Tên thiết bị.
created_at	DATE	Thời điểm thêm mới dữ liệu vào database.
updated_at	DATE	Thời điểm cập nhật dữ liệu vào database.

2.2.6 Sơ đồ ERD



Hình 2.1 Sơ đồ ERD

2.3 Phân tích chi tiết hệ thống

2.3.1 Thiết kế giao diện

2.3.1.1 Ứng dụng

2.3.1.2 Website

2.3.2 Thiết kế API

2.3.2.1 API liên quan đến thông tin người dùng

Bảng 2.10 Kết quả thí nghiệm

Đường dẫn	**Phương thức**	**Mô tả**
api/users/profile	GET	Lấy thông tin của người dùng với đầu vào là JWT token khi người dùng đăng nhập thành công vào hệ thống
api/users/profile	PUT	Cập nhật thông tin người dùng
api/users/change-password	PUT	Thay đổi mật khẩu của người dùng
api/users	GET	Lấy toàn bộ danh sách thông tin của người dùng
api/users/{:userId}	GET	Lấy thông tin của người dùng cụ thể theo user id

2.3.2.2 API liên quan đến việc xác thực người dùng

Bảng 2.11 Kết quả thí nghiệm

Đường dẫn	**Phương thức**	**Mô tả**
api/register	POST	Đăng ký tài khoản
api/login	POST	Đăng nhập vào hệ thống
api/logout	GET	Đăng xuất khỏi hệ thống
api/reset-password	POST	Gửi reset token đến email của người dùng để reset mật khẩu
api/reset-password/reset	POST	Giúp reset lại mật khẩu mới với verify token được nhận từ api: api/reset-password

2.3.2.3 API liên quan đến tin tức

Bảng 2.12 Kết quả thí nghiệm

Đường dẫn	**Phương thức**	**Mô tả**
api/news/{:newsId}	GET	Lấy nội của tin tức tương ứng với id dưới dạng HTML
api/news	GET	Lấy toàn bộ danh sách thông tin của tin tức
api/categories	GET	Lấy toàn bộ danh sách thông tin của tin tức
api/category/{:categoryId}	GET	Lấy thông tin của loại tin tức theo id tương ứng
api/ news/category/{:categoryId}	GET	Lấy toàn bộ thông tin của tin tức theo id của loại tin

2.3.2.4 API liên quan đến bản ghi ECG

Bảng 2.13 Kết quả thí nghiệm

Đường dẫn	**Phương thức**	**Mô tả**
api/ecg-records/upload	POST	Tải dữ liệu của phiên đo ECG lên server
api/ecg-records/patient/{:patientId}	GET	Lấy danh sách thông tin các phiên đo ECG của bệnh nhân
api/ecg-records/doctor/{:doctorId}	GET	Lấy danh sách thông tin các phiên đo ECG của các bệnh nhân được quản lý bởi bác sỹ
api/ecg-records/record-data/{:recordId}	GET	Lấy dữ liệu một phiên đo của bệnh nhân

2.3.2.5 API liên quan liên quan đến việc phân công bệnh nhân cho bác sỹ

Bảng 2.14 Kết quả thí nghiệm

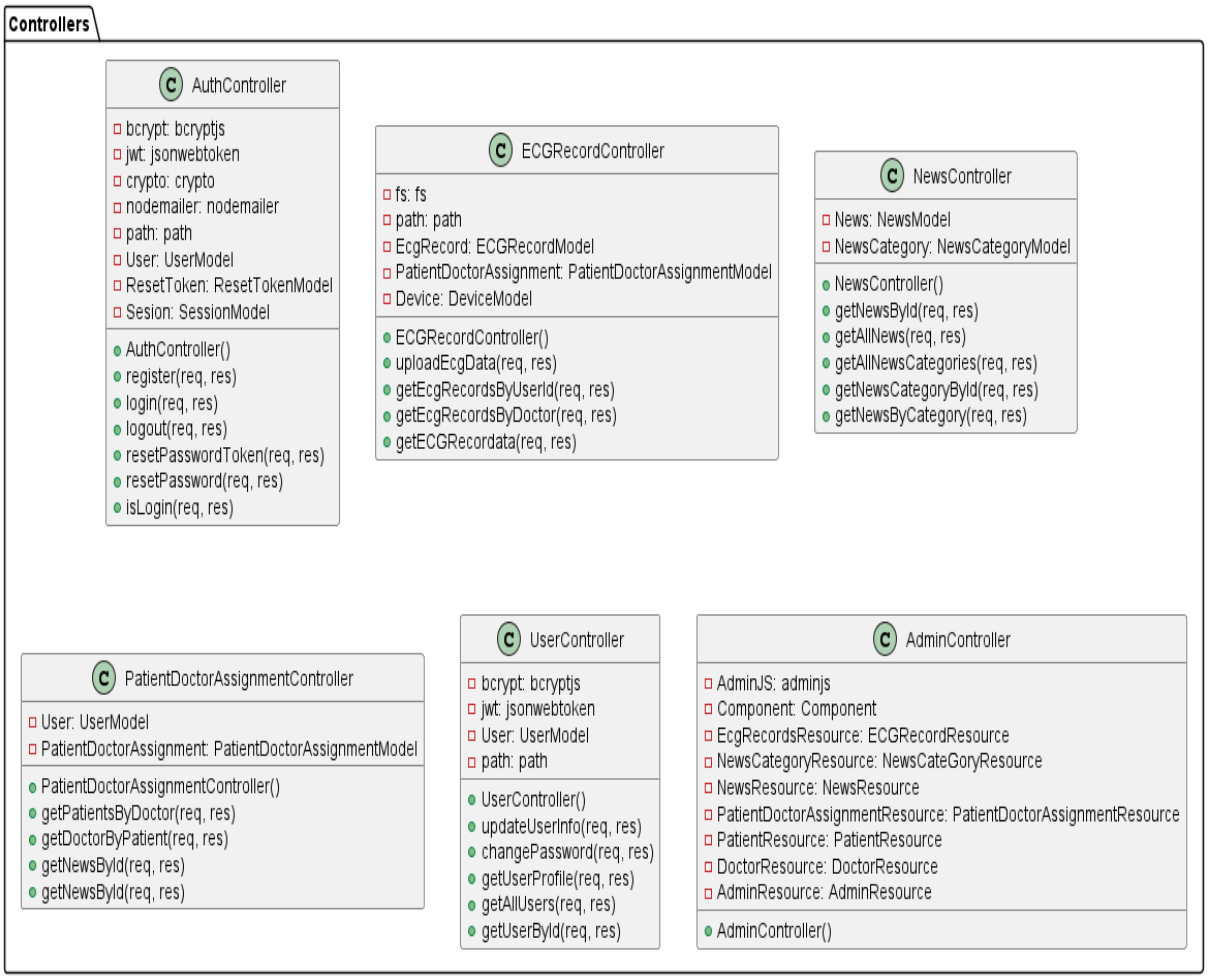
Đường dẫn	Phương thức	Mô tả
api/doctor/{:doctorId}/patients	GET	Lấy danh sách thông tin bệnh nhân được phân công cho bác sĩ
api/patient/{:patientId}/doctor	GET	Lấy thông tin bác sỹ được phân công cho bệnh nhân

2.3.3 Sơ đồ lớp

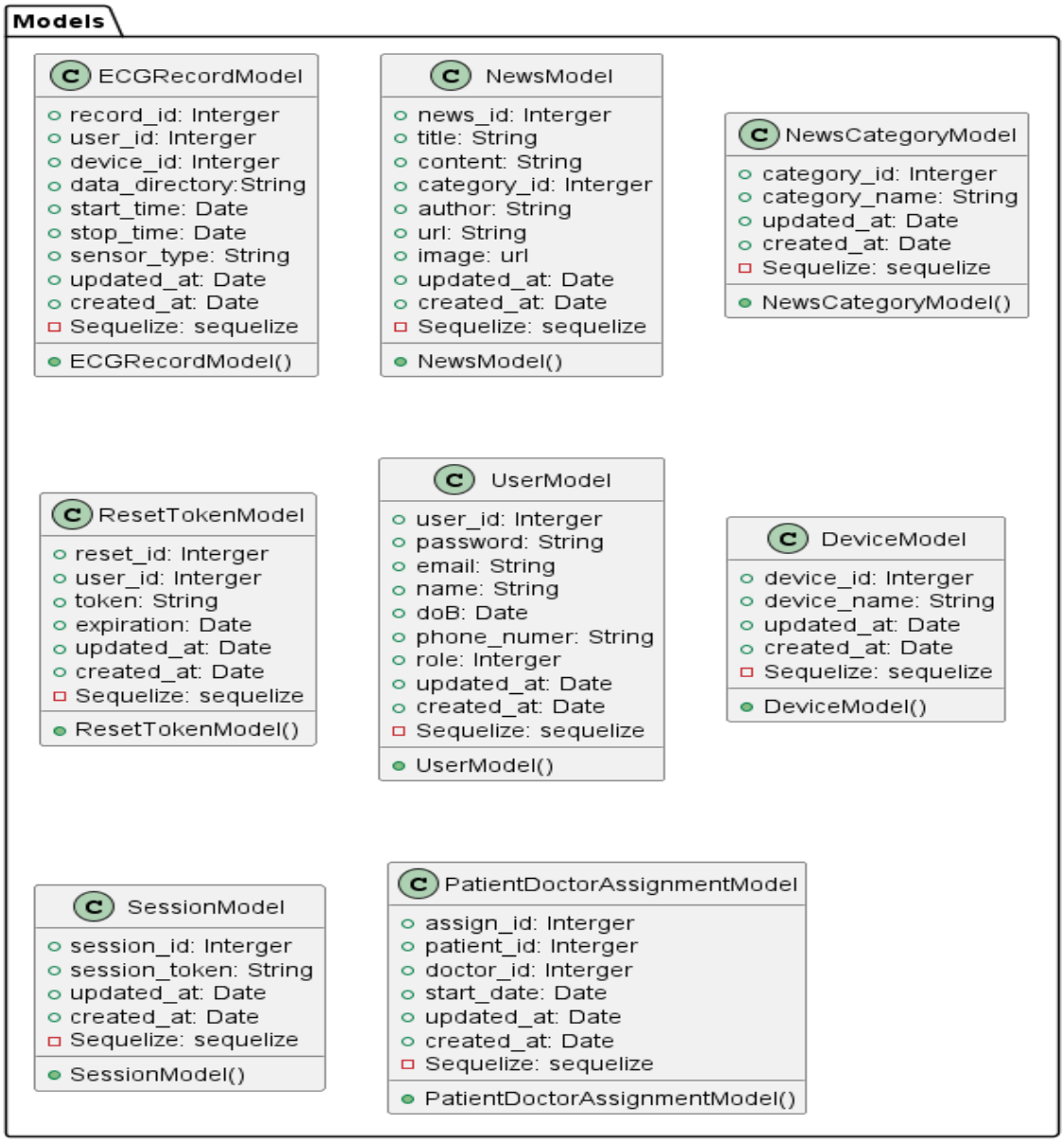
2.3.3.1 Ứng dụng

2.3.3.2 Server và Website quản trị hệ thống

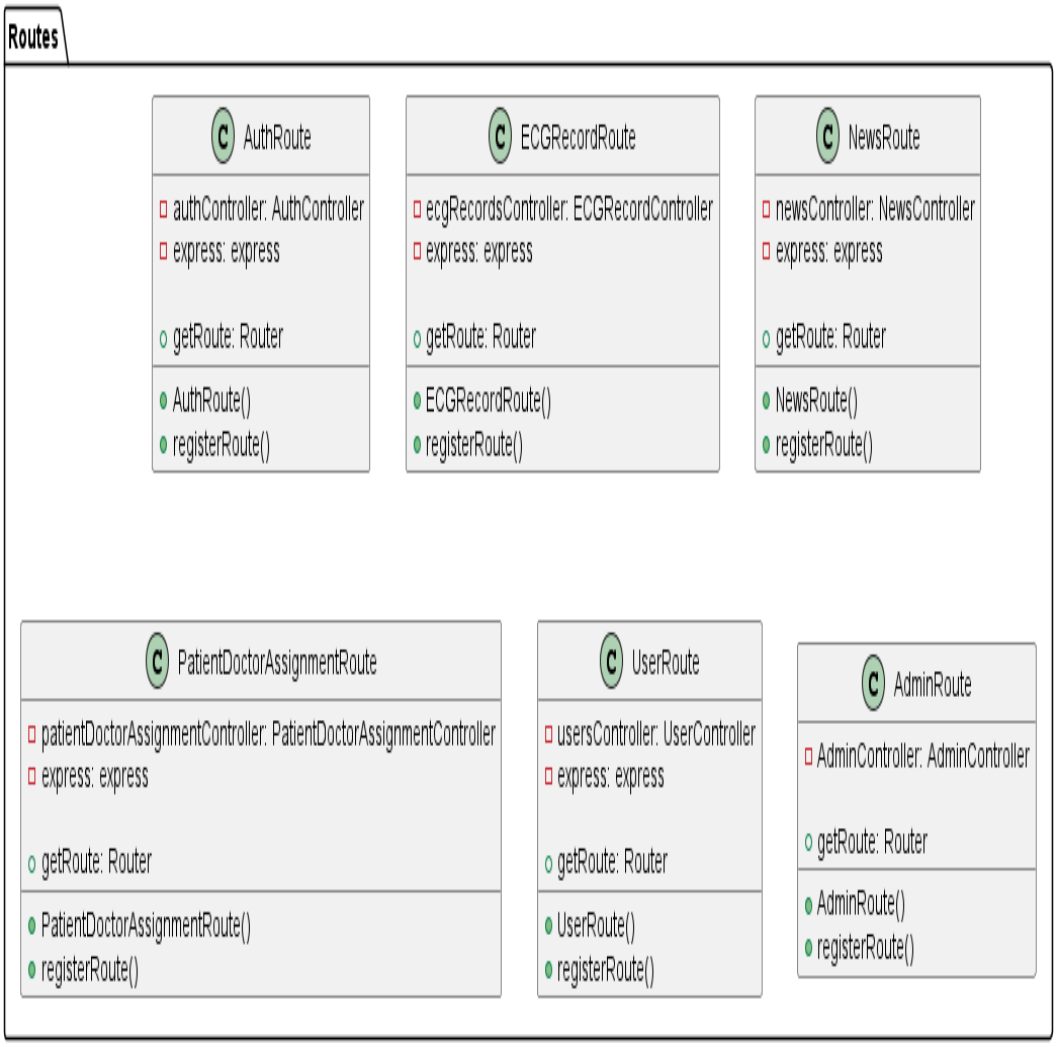
a) Danh sách các class diagram



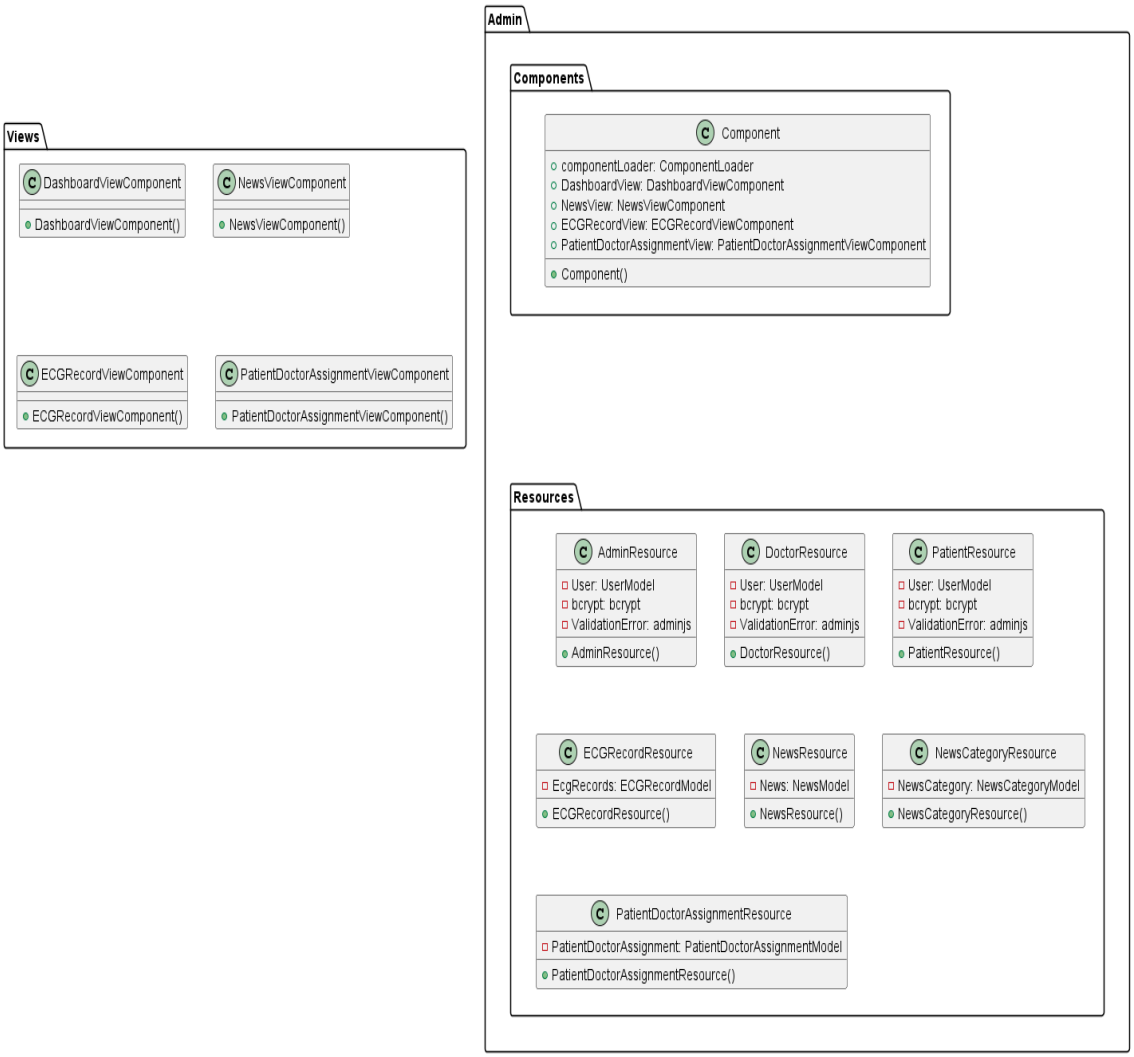
Hình 2.2 Sơ đồ lớp của package Controllers



Hình 2.3 Sơ đồ lớp của package Models



Hình 2.4 Sơ đồ lớp của package Routes



Hình 2.5 Sơ đồ lớp của package View và Admin

- Package "Controllers":

  Chứa các controllers xử lý logic và điều khiển các yêu cầu từ người dùng.

- Package "Models":

  Chứa các lớp đại diện cho các đối tượng của cơ sở dữ liệu, định nghĩa các thuộc tính và phương thức để làm việc với dữ liệu.

- Package "Routes":

  Chứa các lớp route, xác định các endpoint và xử lý các yêu cầu HTTP từ người dùng bằng cách gọi tới các controllers tương ứng.

- Package "Views":

  Chứa các view components đại diện cho giao diện người dùng.

- Package "Admin":

  Chứa các thành phần liên quan đến trang Admin dashboard.

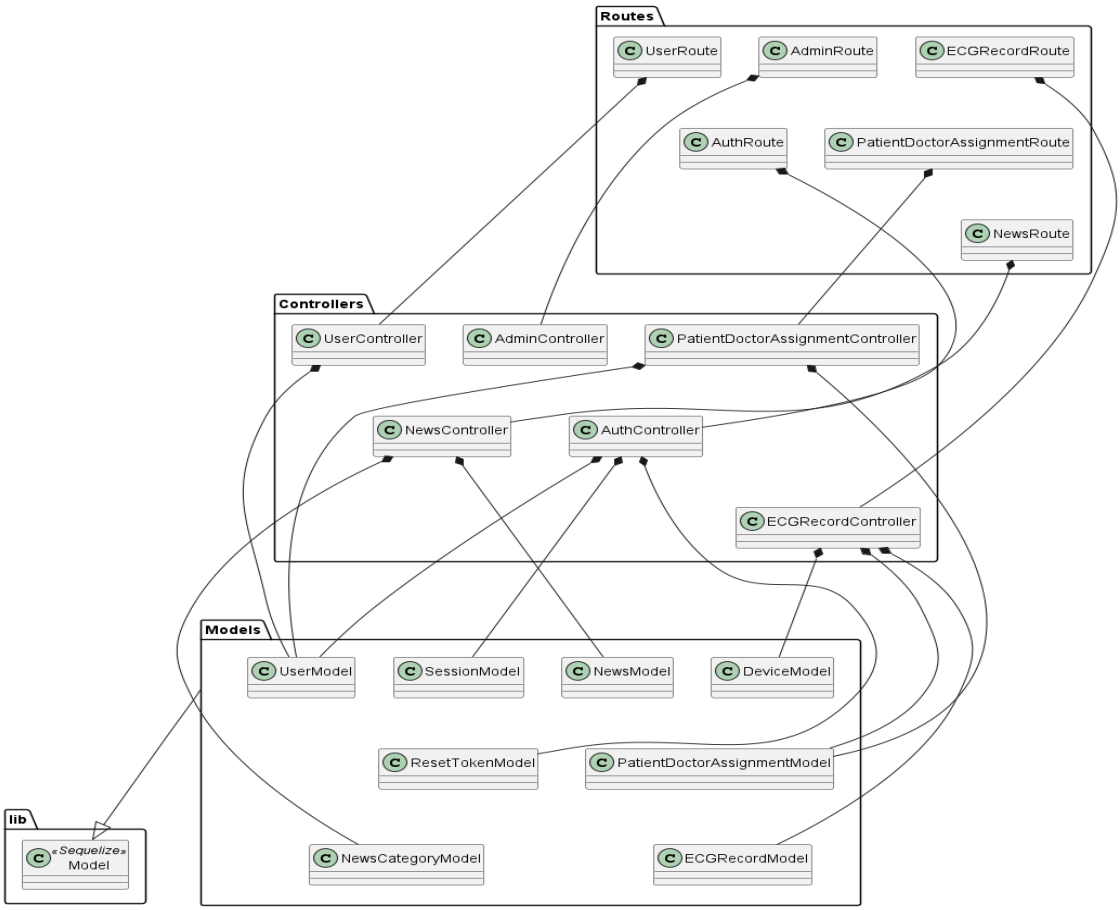
- Package "Resources" (Trong "Admin"):

Chứa các lớp Resource (nguồn tài nguyên) đại diện cho các tài nguyên của trang Admin dashboard, bao gồm: AdminResource, DoctorResource, PatientResource, ECGRecordResource, NewsResource, NewsCategoryResource và PatientDoctor-AssignmentResource.

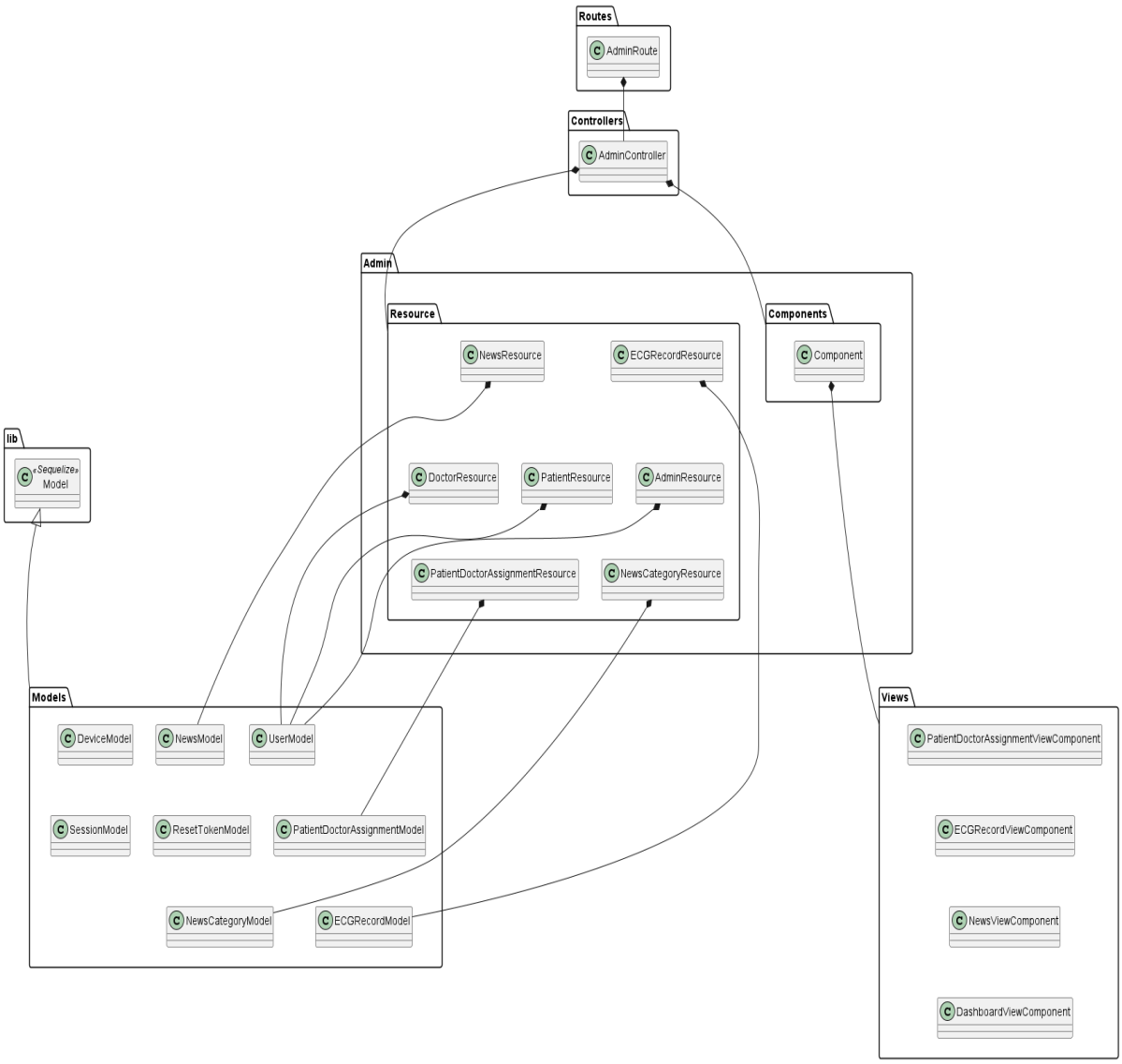
- Package "Components" (Trong "Admin"):

  Chứa các lớp Components đại diện cho các thành phần giao diện của trang Admin dashboard, bao gồm: ComponentLoader, DashboardViewComponent, NewsViewComponent, ECGRecordViewComponent và PatientDoctorAssignmentView-Component.

b) Mối quan hệ



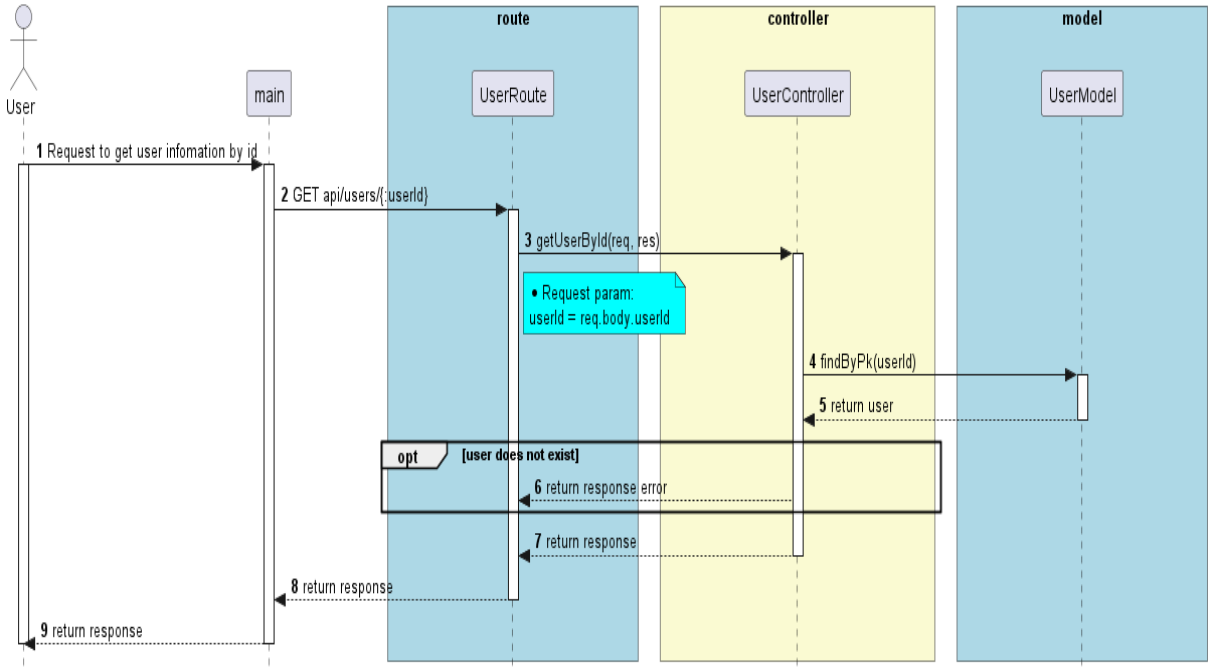
Hình 2.6 Mối quan hệ giữa các class ở phía server



Hình 2.7 Mối quan hệ giữa các class ở phía website quản trị

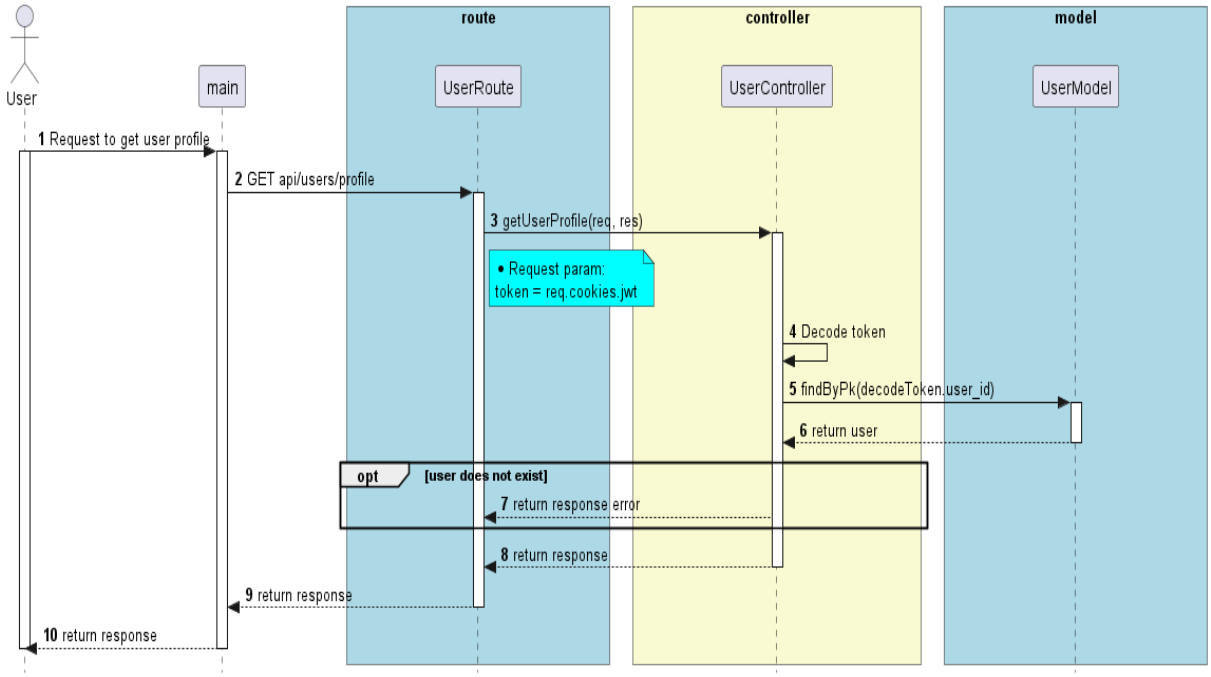
2.3.4 Sơ đồ tuần tự

2.3.4.1 API liên quan đến thông tin người dùng



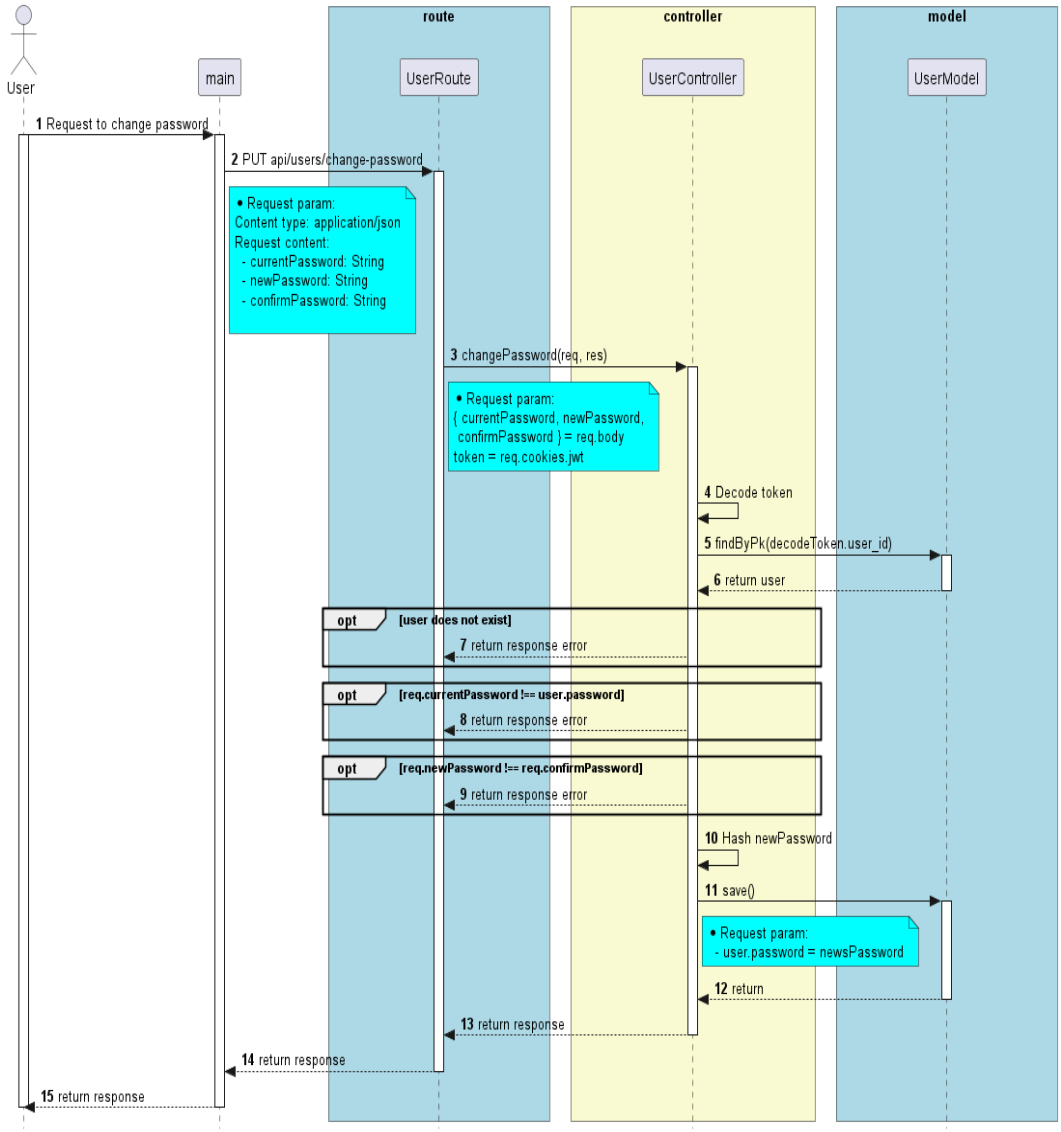
Hình 2.8 Sơ đồ tuần tự cho API lấy thông tin của người dùng dựa trên ID

Hình 2.8 mô tả quá trình lấy thông tin người dùng dựa trên ID trong ứng dụng. Người dùng gửi yêu cầu lấy thông tin người dùng theo ID, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi UserController. UserController kiểm tra thông tin và truy vấn UserModel để lấy thông tin người dùng. Nếu người dùng không tồn tại, hệ thống trả về response lỗi, ngược lại, response chứa thông tin người dùng được gửi lại từ UserController tới người dùng.



Hình 2.9 Sơ đồ tuần tự cho API lấy thông tin của người dùng dựa trên JWT token

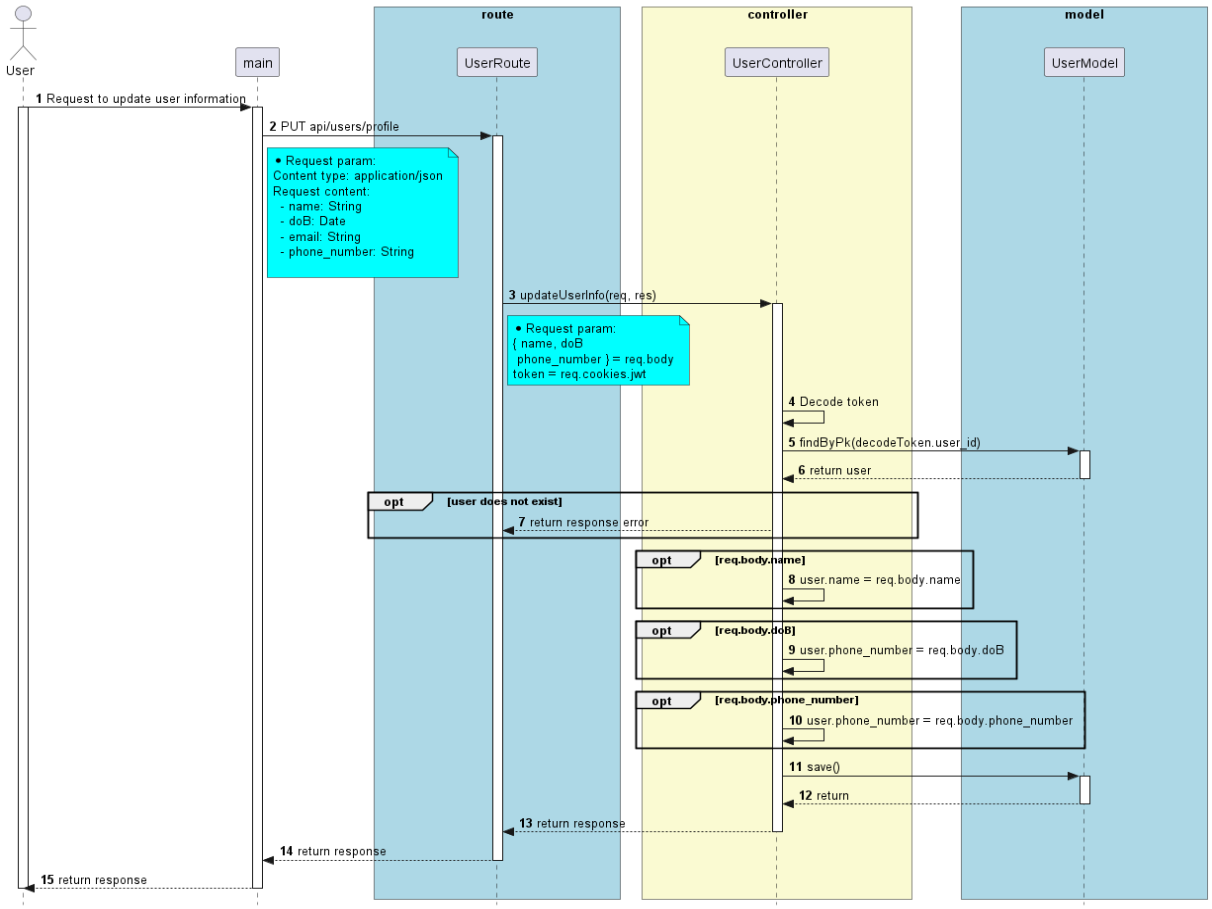
Hình 2.9 mô tả quá trình lấy thông tin hồ sơ người dùng trong ứng dụng. Người dùng gửi yêu cầu lấy thông tin hồ sơ, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi UserController. UserController giải mã mã thông báo (token) để xác định người dùng, sau đó truy vấn UserModel để lấy thông tin hồ sơ của người dùng dựa trên user_id từ mã giải mã. Nếu người dùng không tồn tại, hệ thống trả về response lỗi, ngược lại, response chứa thông tin hồ sơ của người dùng được gửi lại từ UserController tới người dùng.



Hình 2.10 Sơ đồ tuần tự cho API thay đổi mật khẩu người dùng

Hình 2.10 mô tả quá trình thay đổi mật khẩu người dùng trong ứng dụng. Người dùng gửi yêu cầu thay đổi mật khẩu, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi UserController. Đầu tiên, hệ thống sẽ kiểm tra và giải mã mã thông báo (token) để xác định người dùng. Sau đó, UserController truy vấn UserModel để tìm người dùng dựa trên user_id từ mã giải mã. Nếu người dùng không tồn tại hoặc mật khẩu hiện tại

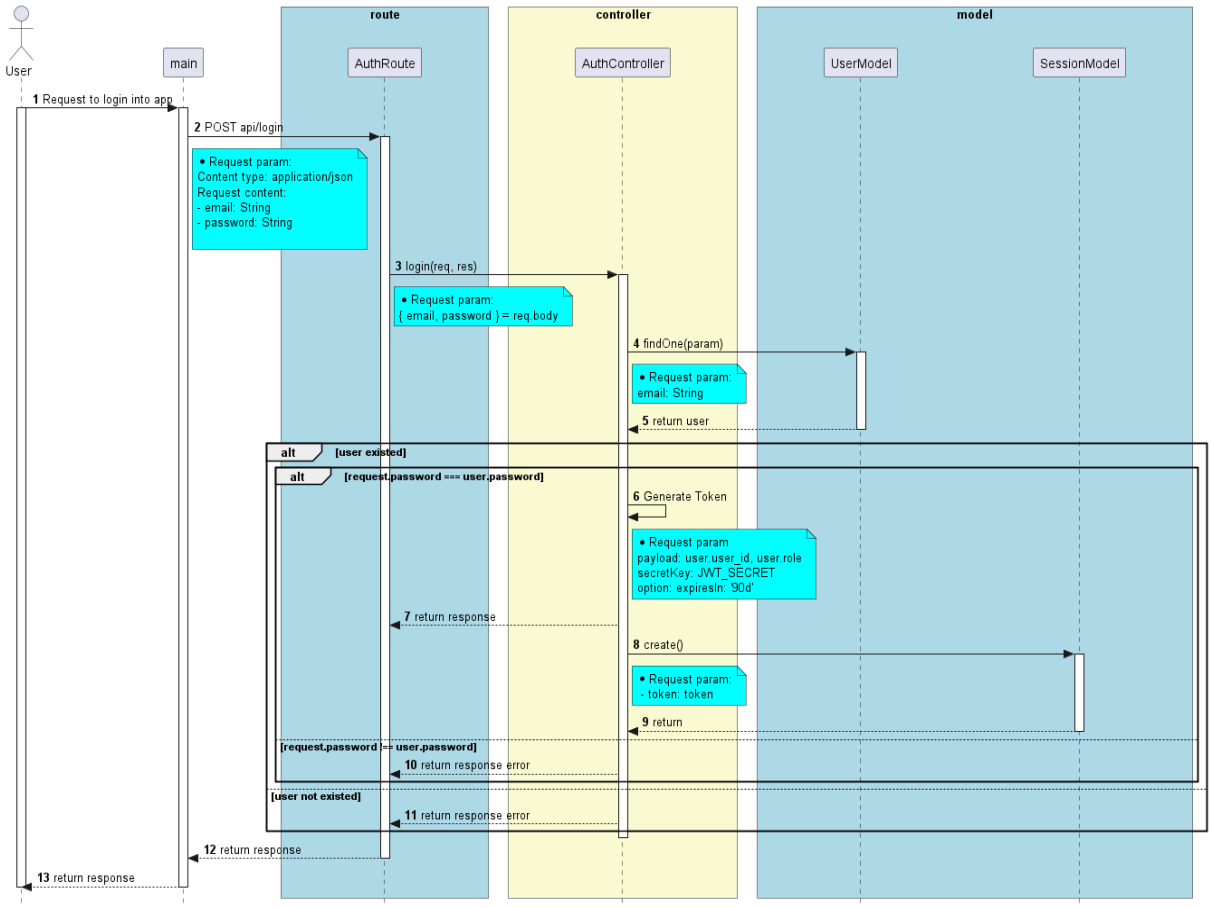
không khớp với mật khẩu trong hệ thống, hoặc mật khẩu mới không khớp với xác nhận mật khẩu, hệ thống sẽ trả về response lỗi tương ứng. Ngược lại, mật khẩu mới sẽ được mã hóa và lưu vào UserModel, sau đó hệ thống trả về response thành công tới người dùng.



Hình 2.11 Sơ đồ tuần tự cho API cập nhật thông tin người dùng

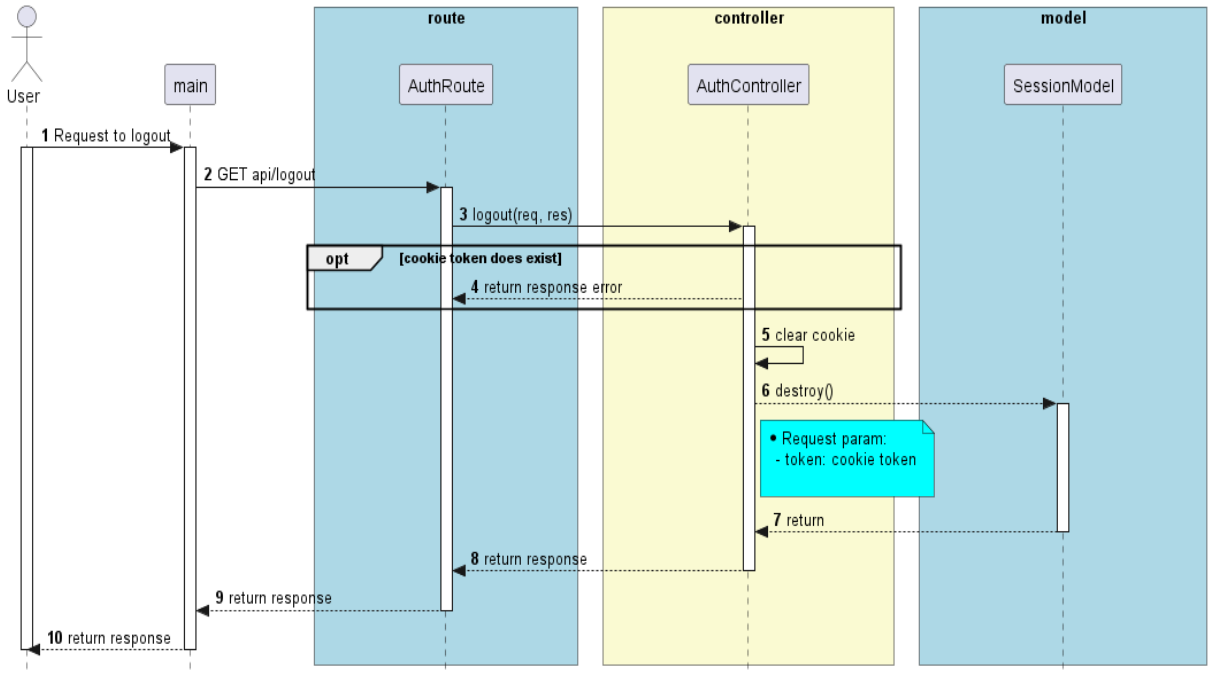
Hình 2.11 mô tả quá trình cập nhật thông tin người dùng trong ứng dụng. Người dùng gửi yêu cầu cập nhật thông tin, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi UserController. Đầu tiên, hệ thống sẽ kiểm tra và giải mã mã thông báo (token) để xác định người dùng. Sau đó, UserController truy vấn UserModel để tìm người dùng dựa trên user_id từ mã giải mã. Nếu người dùng không tồn tại, hệ thống sẽ trả về response lỗi. Ngược lại, thông tin cập nhật như tên, ngày sinh (dob), và số điện thoại (phone_number) sẽ được cập nhật vào UserModel. Sau đó, hệ thống trả về response thành công tới người dùng.

2.3.4.2 API liên quan đến việc xác thực người dùng



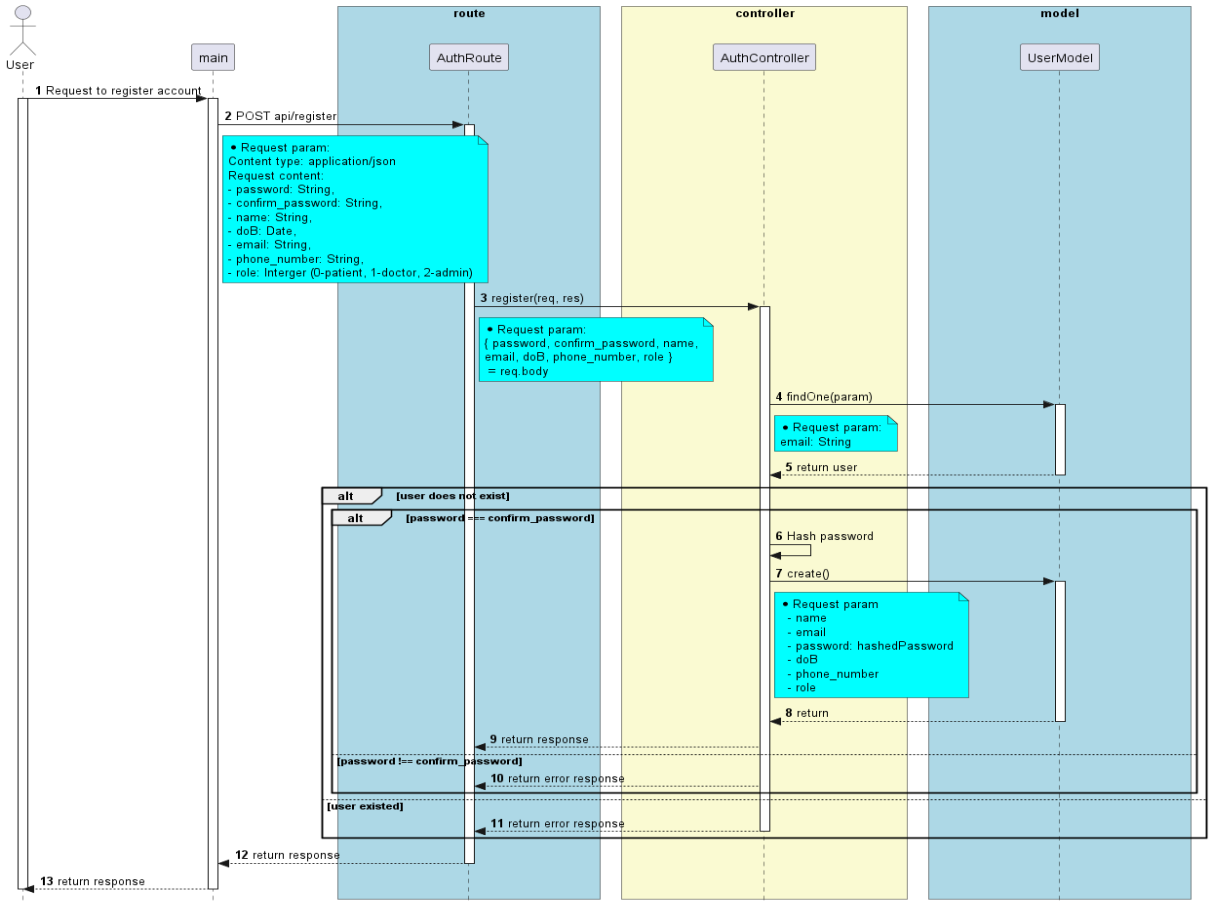
Hình 2.12 Sơ đồ tuần tự cho API đăng nhập vào hệ thống

Hình 2.12 mô tả quá trình xác thực người dùng đăng nhập vào ứng dụng. Người dùng gửi yêu cầu đăng nhập, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi AuthController. AuthController kiểm tra thông tin người dùng, tạo token nếu đúng thông tin đăng nhập và trả về response cho người dùng.



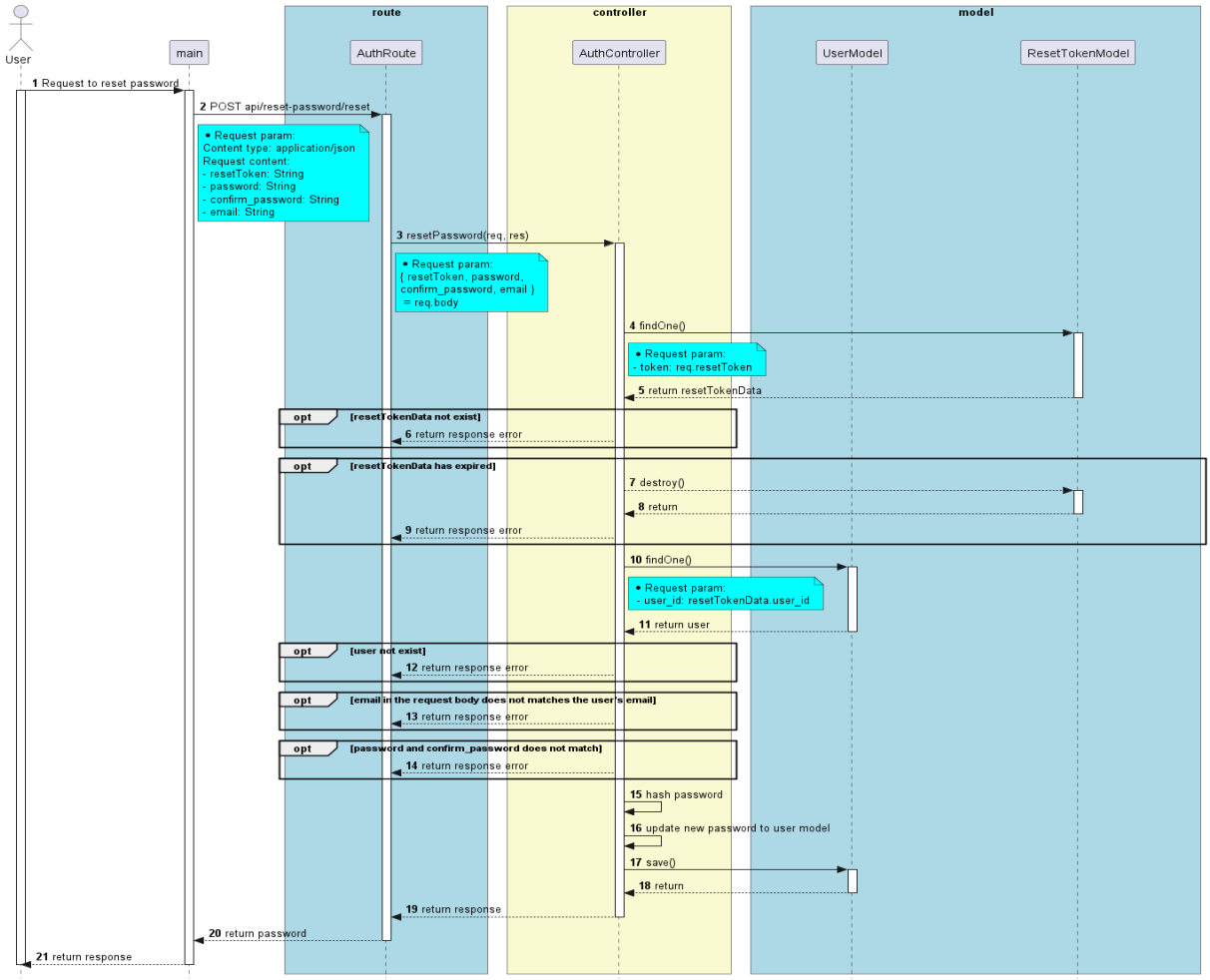
Hình 2.13 Sơ đồ tuần tự cho API đăng xuất khỏi hệ thống

Hình 2.13 mô tả quá trình đăng xuất (logout) người dùng khỏi ứng dụng. Người dùng gửi yêu cầu đăng xuất, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi AuthController. Đầu tiên, hệ thống kiểm tra xem cookie token có tồn tại hay không. Nếu không tồn tại, hệ thống trả về response lỗi. Ngược lại, AuthController xóa cookie và gọi tới SessionModel để hủy phiên đăng nhập của người dùng dựa trên token trong cookie. Sau khi xử lý, hệ thống trả về response thành công tới người dùng.



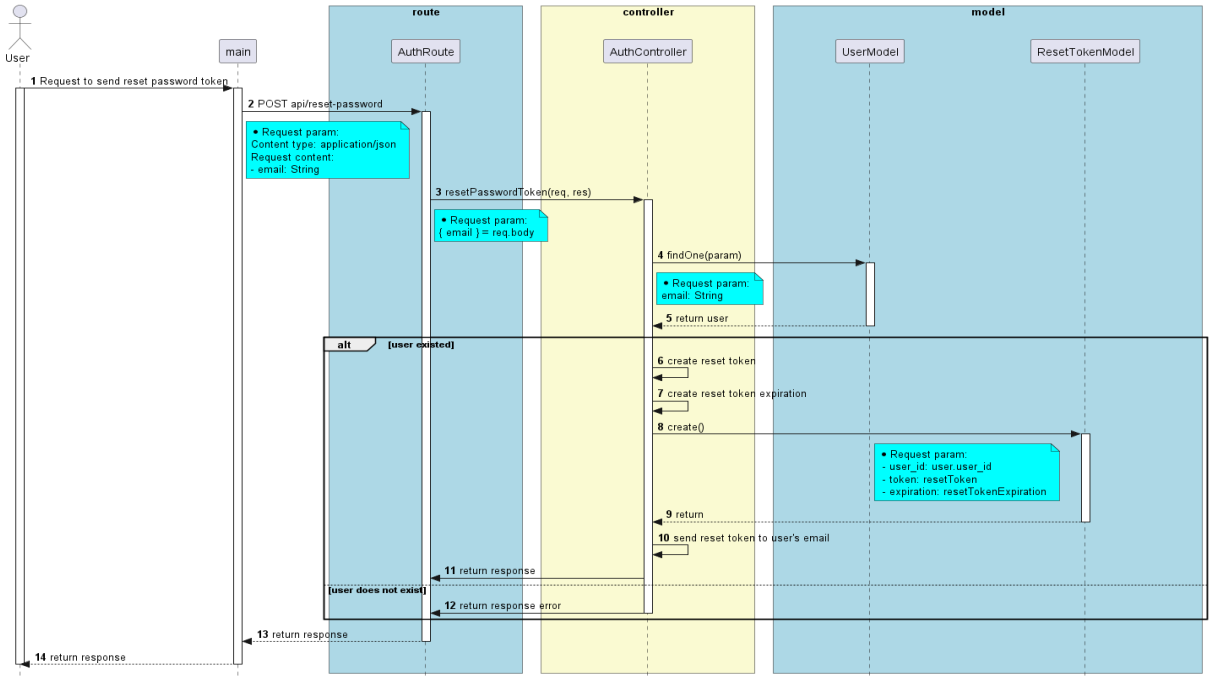
Hình 2.14 Sơ đồ tuần tự cho API đăng ký tài khoản

Hình 2.14 mô tả quá trình đăng ký tài khoản trong ứng dụng. Người dùng gửi yêu cầu đăng ký, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi AuthController. Đầu tiên, hệ thống kiểm tra thông tin yêu cầu đăng ký, bao gồm việc kiểm tra mật khẩu và xác nhận mật khẩu có khớp hay không. Sau đó, AuthController kiểm tra xem người dùng đã tồn tại trong hệ thống hay chưa. Nếu người dùng chưa tồn tại và mật khẩu khớp, AuthController sẽ mã hóa mật khẩu và tạo một bản ghi mới trong UserModel để lưu thông tin đăng ký. Nếu người dùng đã tồn tại hoặc mật khẩu không khớp, hệ thống sẽ trả về response lỗi tương ứng. Sau khi xử lý, hệ thống trả về response kết quả cho người dùng.



Hình 2.15 Sơ đồ tuần tự cho API đặt lại mật khẩu

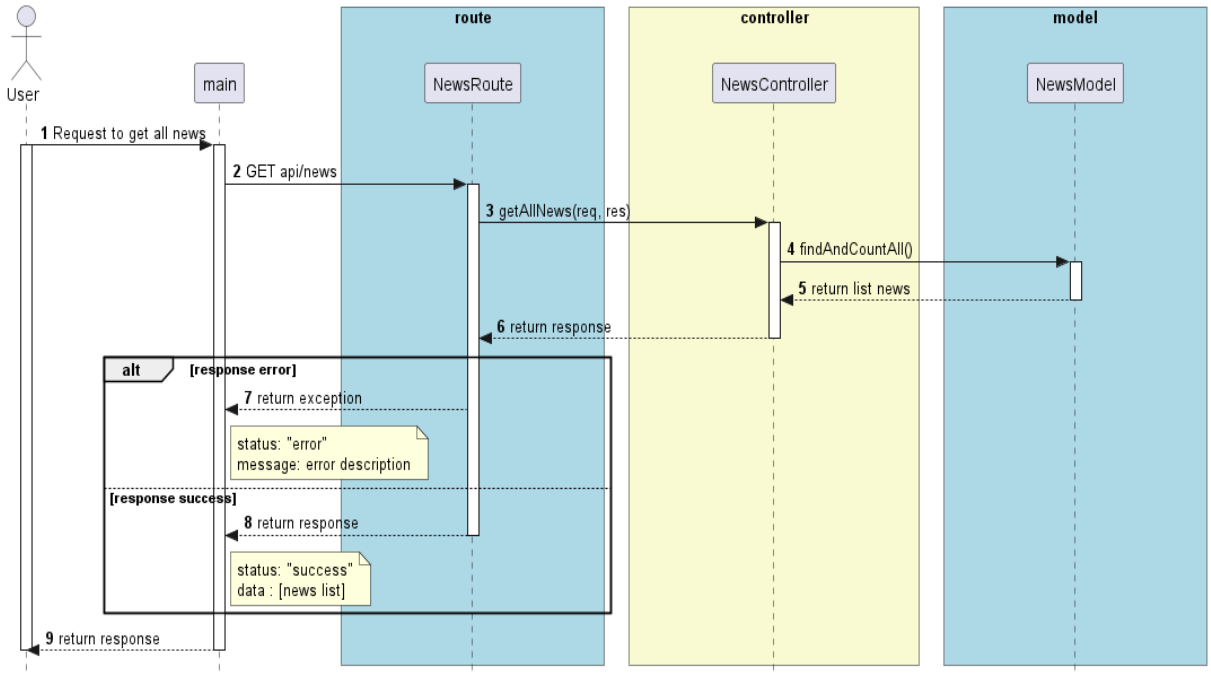
Hình 2.15 mô tả quá trình đặt lại mật khẩu (reset password) trong ứng dụng. Người dùng gửi yêu cầu đặt lại mật khẩu, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi AuthController. Đầu tiên, hệ thống kiểm tra thông tin yêu cầu đặt lại mật khẩu, bao gồm resetToken, mật khẩu mới và xác nhận mật khẩu. Sau đó, AuthController kiểm tra xem resetToken có hợp lệ hay không, bằng cách tìm kiếm trong ResetTokenModel. Nếu resetToken không hợp lệ hoặc đã hết hạn, hệ thống sẽ trả về response lỗi tương ứng. Nếu resetToken hợp lệ, AuthController tiếp tục kiểm tra xem người dùng có tồn tại và email trong yêu cầu có khớp với email của người dùng không. Nếu không hợp lệ, hệ thống sẽ trả về response lỗi tương ứng. Nếu mọi thông tin đều hợp lệ, AuthController sẽ mã hóa mật khẩu mới, cập nhật mật khẩu trong UserModel và trả về response thành công tới người dùng.



Hình 2.16 Sơ đồ tuần tự cho API gửi token đặt lại mật khẩu

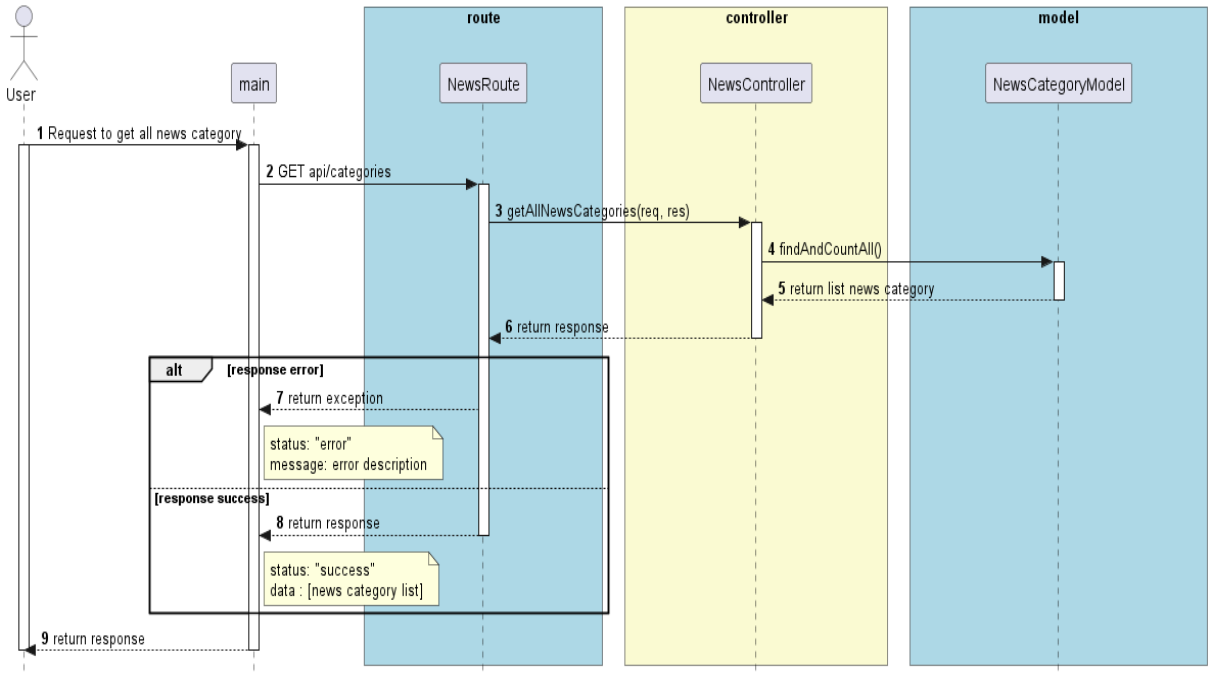
Hình 2.16 mô tả quá trình yêu cầu gửi mã thông báo đặt lại mật khẩu (reset password token) trong ứng dụng. Người dùng gửi yêu cầu đặt lại mật khẩu, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi AuthController. Đầu tiên, hệ thống kiểm tra thông tin yêu cầu đặt lại mật khẩu, bao gồm email người dùng. Sau đó, AuthController kiểm tra xem người dùng đã tồn tại trong hệ thống hay chưa. Nếu người dùng tồn tại, AuthController sẽ tạo một mã thông báo đặt lại mật khẩu (reset token) cùng với thời gian hết hạn cho mã (reset token expiration) và lưu thông tin này vào ResetTokenModel. Sau đó, hệ thống sẽ gửi mã thông báo đặt lại mật khẩu tới email của người dùng. Nếu người dùng không tồn tại, hệ thống sẽ trả về response lỗi tương ứng. Sau khi xử lý, hệ thống trả về response kết quả cho người dùng.

2.3.4.3 API liên quan đến tin tức



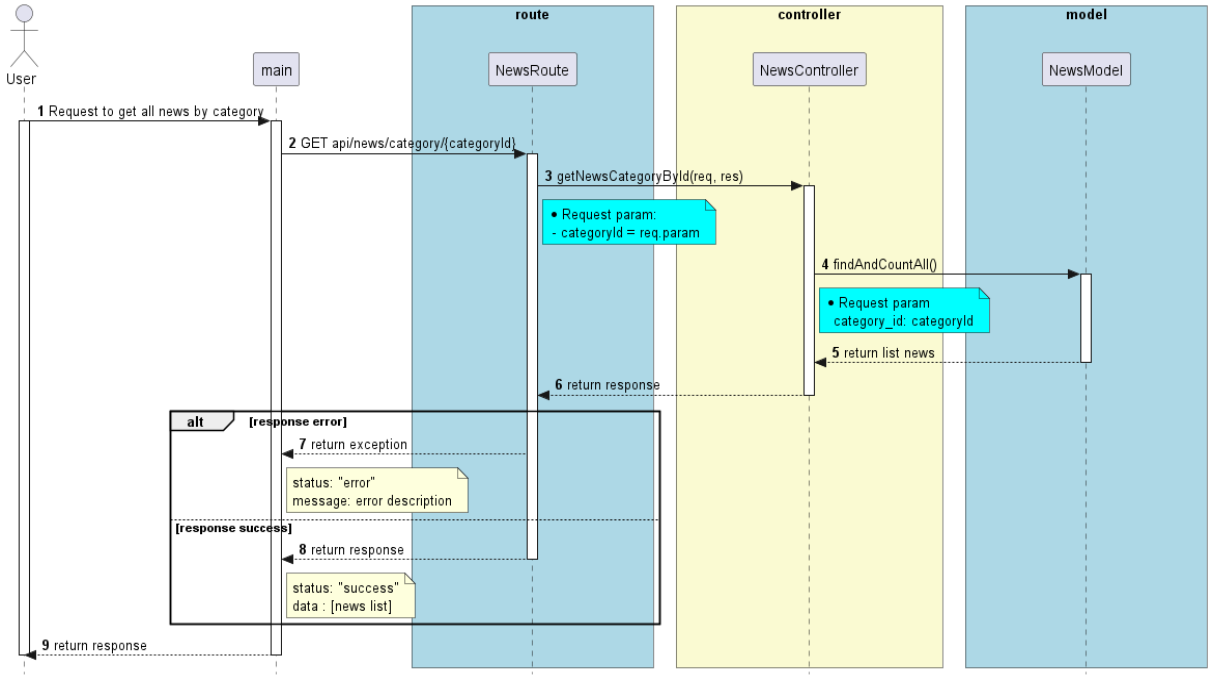
Hình 2.17 Sơ đồ tuần tự cho API lấy tất cả thông tin của tin tức

Hình 2.17 mô tả quá trình lấy tất cả tin tức (news) trong ứng dụng. Người dùng gửi yêu cầu lấy tất cả tin tức, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi NewsController. NewsController tiếp tục truy vấn NewsModel để lấy danh sách tất cả các tin tức từ cơ sở dữ liệu. Sau đó, danh sách tin tức được trả về từ NewsModel và được gửi trở lại NewsRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa danh sách tin tức cho người dùng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi với mô tả lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



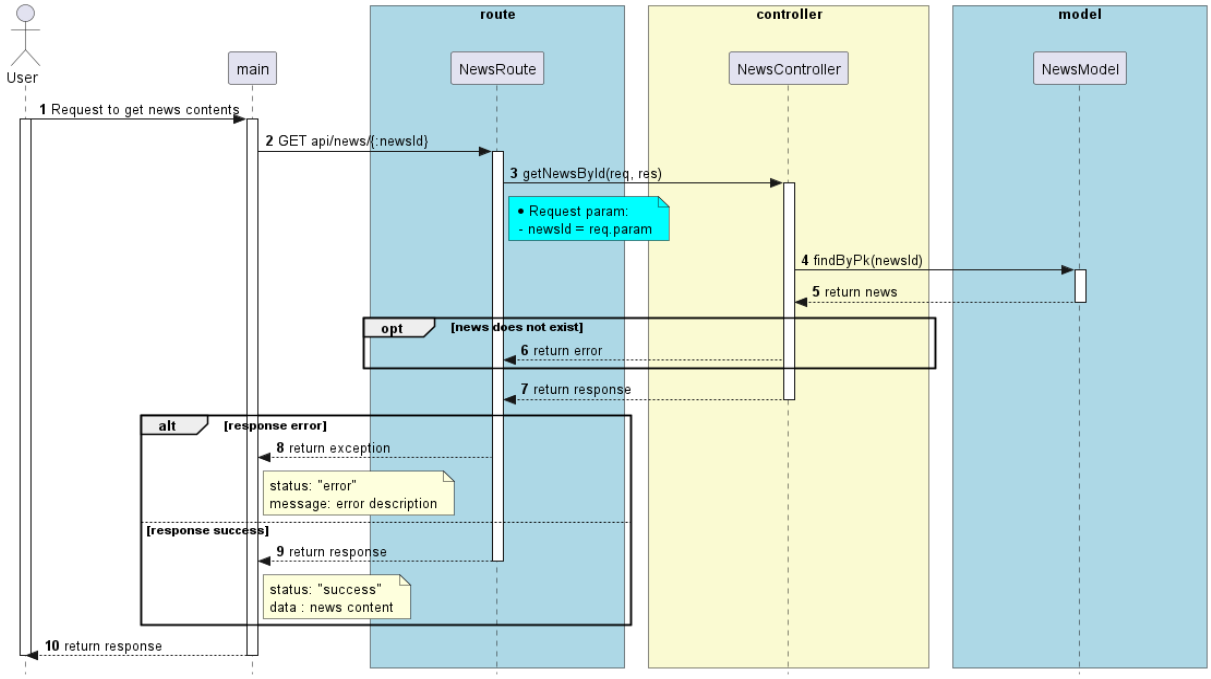
Hình 2.18 Sơ đồ tuần tự cho API lấy tất cả thông tin của danh mục tin tức

Hình 2.18 mô tả quá trình lấy tất cả danh mục tin tức (news category) trong ứng dụng. Người dùng gửi yêu cầu lấy tất cả danh mục tin tức, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi NewsController. NewsController tiếp tục truy vấn NewsCategoryModel để lấy danh sách tất cả các danh mục tin tức từ cơ sở dữ liệu. Sau đó, danh sách danh mục tin tức được trả về từ NewsCategoryModel và được gửi trở lại NewsRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa danh sách danh mục tin tức cho người dùng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi với mô tả lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



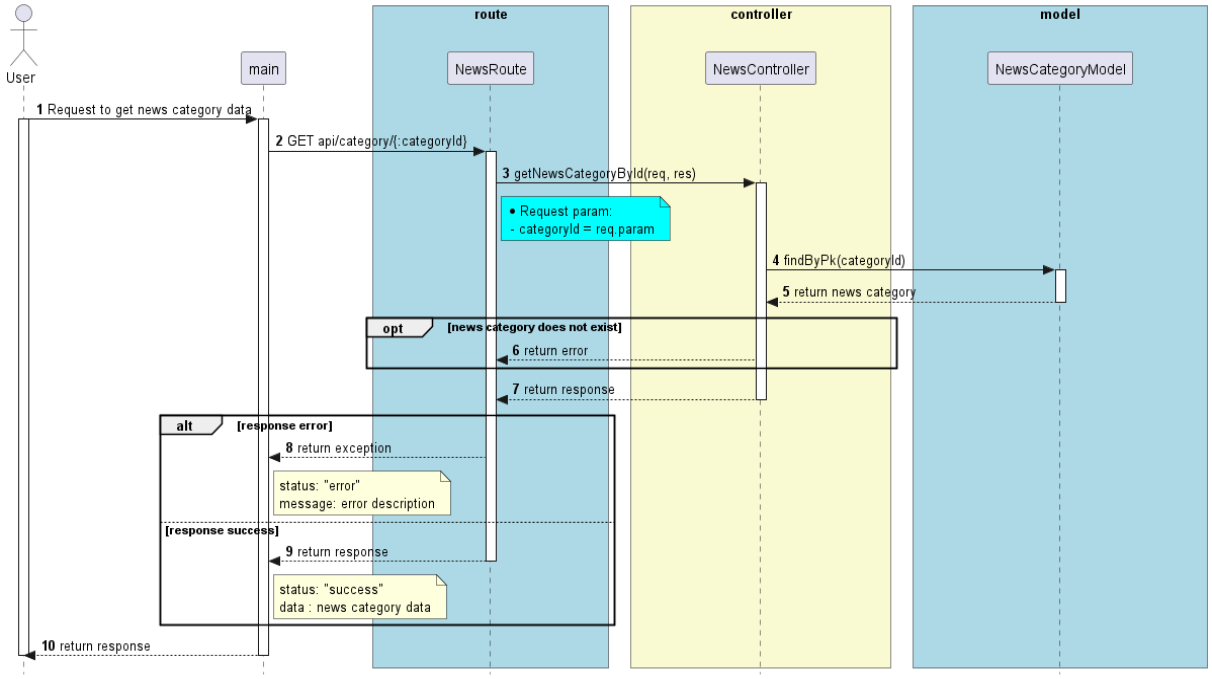
Hình 2.19 Sơ đồ tuần tự cho API lấy tất cả tin tức theo danh mục tin tức

Hình 2.19 mô tả quá trình lấy danh sách tin tức (news list) theo danh mục (category) trong ứng dụng. Người dùng gửi yêu cầu lấy danh sách tin tức theo danh mục, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi NewsController. NewsController tiếp tục truy vấn NewsModel để lấy danh sách tin tức dựa trên categoryId được truyền vào từ yêu cầu. Sau đó, danh sách tin tức được trả về từ NewsModel và được gửi trở lại NewsRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa danh sách tin tức cho người dùng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



Hình 2.20 Sơ đồ tuần tự cho API lấy nội dung của một tin tức

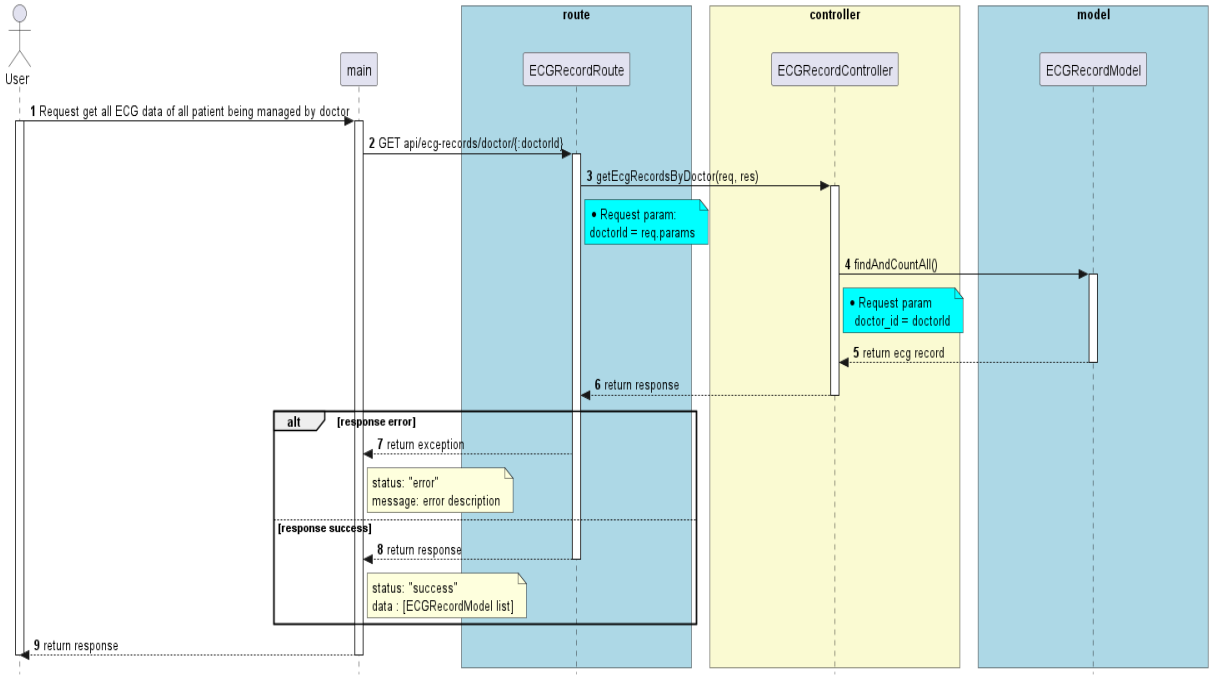
Hình 2.20 mô tả quá trình lấy nội dung tin tức (news content) trong ứng dụng. Người dùng gửi yêu cầu lấy nội dung tin tức, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi NewsController. NewsController tiếp tục truy vấn NewsModel để lấy nội dung tin tức dựa trên newsId được truyền vào từ yêu cầu. Sau đó, nội dung tin tức được trả về từ NewsModel và được gửi trở lại NewsRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa nội dung tin tức cho người dùng. Nếu tin tức không tồn tại, hệ thống trả về response lỗi tương ứng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi với mô tả lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



Hình 2.21 Sơ đồ tuần tự cho API lấy thông tin của một danh mục tin tức

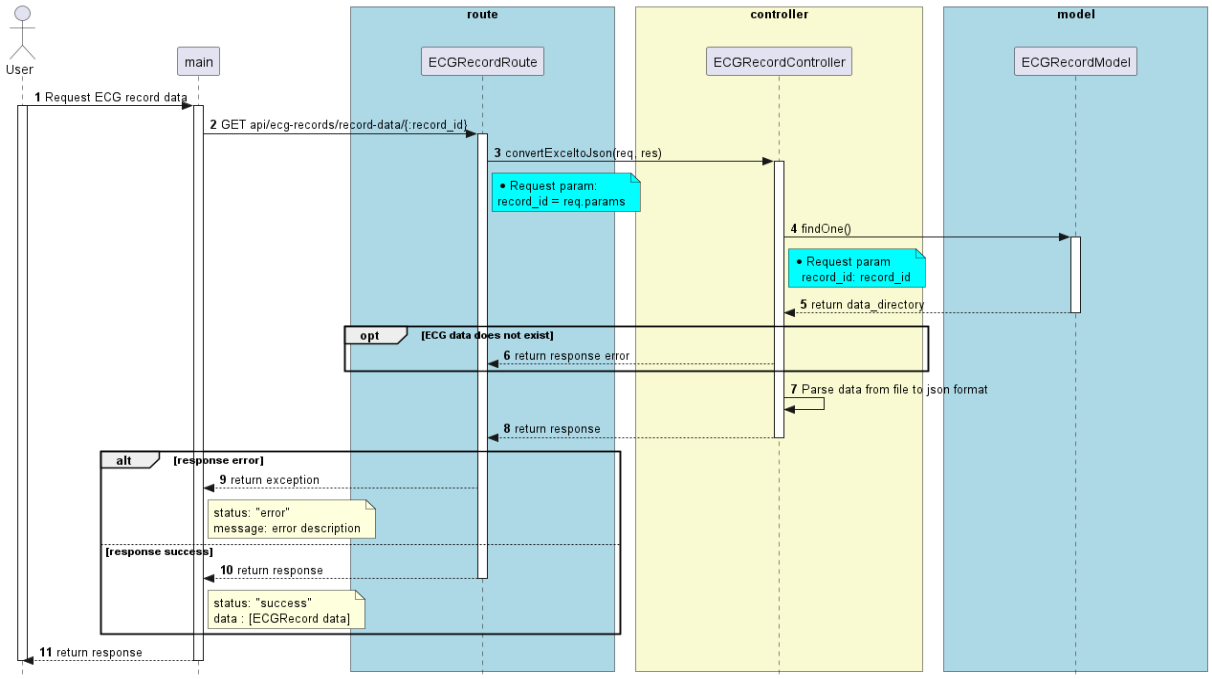
Hình 2.21 mô tả quá trình lấy dữ liệu danh mục tin tức (news category data) trong ứng dụng. Người dùng gửi yêu cầu lấy dữ liệu danh mục tin tức, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi NewsController. NewsController tiếp tục truy vấn NewsCategoryModel để lấy dữ liệu danh mục tin tức dựa trên categoryId được truyền vào từ yêu cầu. Sau đó, dữ liệu danh mục tin tức được trả về từ NewsCategoryModel và được gửi trở lại NewsRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa dữ liệu danh mục tin tức cho người dùng. Nếu danh mục tin tức không tồn tại, hệ thống trả về response lỗi tương ứng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi với mô tả lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.

2.3.4.4 API liên quan đến bản ghi ECG



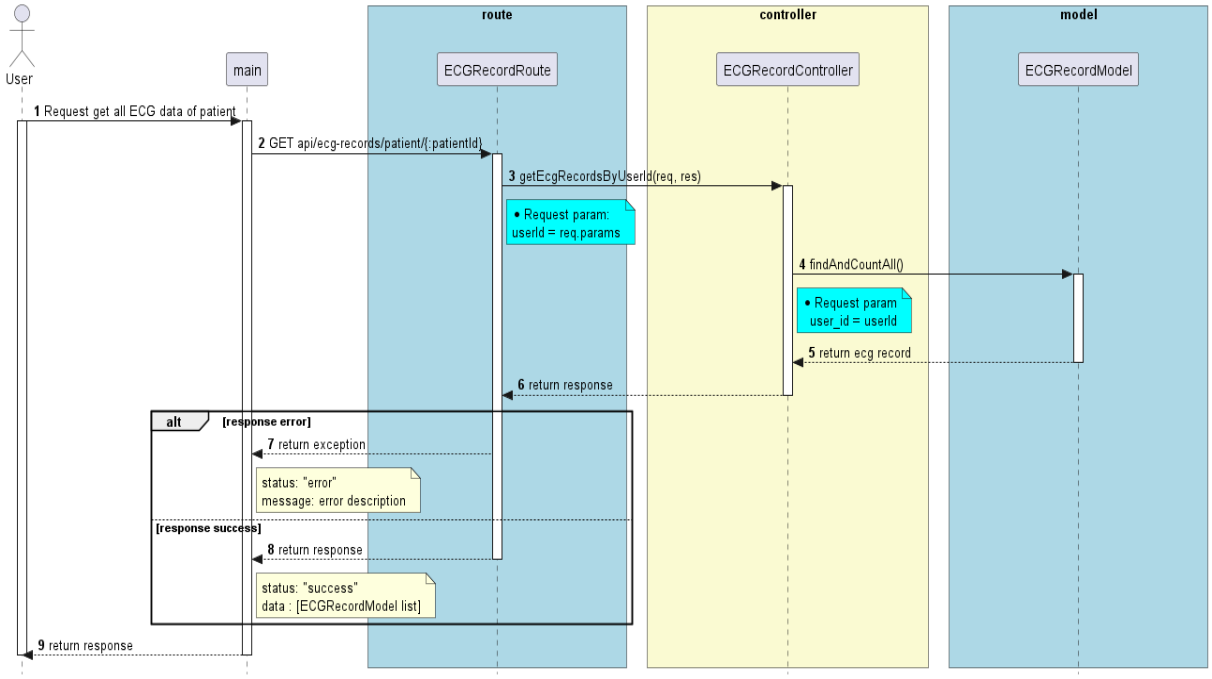
Hình 2.22 Sơ đồ tuần tự cho API lấy thông tin phiên đo ECG của các bệnh nhân được quản lý bởi một bác sĩ

Hình 2.22 mô tả quá trình lấy dữ liệu ECG (Electrocardiogram) của tất cả bệnh nhân được quản lý bởi một bác sĩ trong ứng dụng. Người dùng (bác sĩ) gửi yêu cầu lấy dữ liệu ECG của tất cả bệnh nhân mà họ quản lý, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi ECGRecordController. ECGRecordController tiếp tục truy vấn ECGRecordModel để lấy dữ liệu ECG dựa trên doctorId (ID của bác sĩ) được truyền vào từ yêu cầu. Sau đó, danh sách dữ liệu ECG của các bệnh nhân được trả về từ ECGRecordModel và được gửi trở lại ECGRecordRoute để trả về response cho người dùng. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa danh sách dữ liệu ECG cho bác sĩ. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng (bác sĩ).



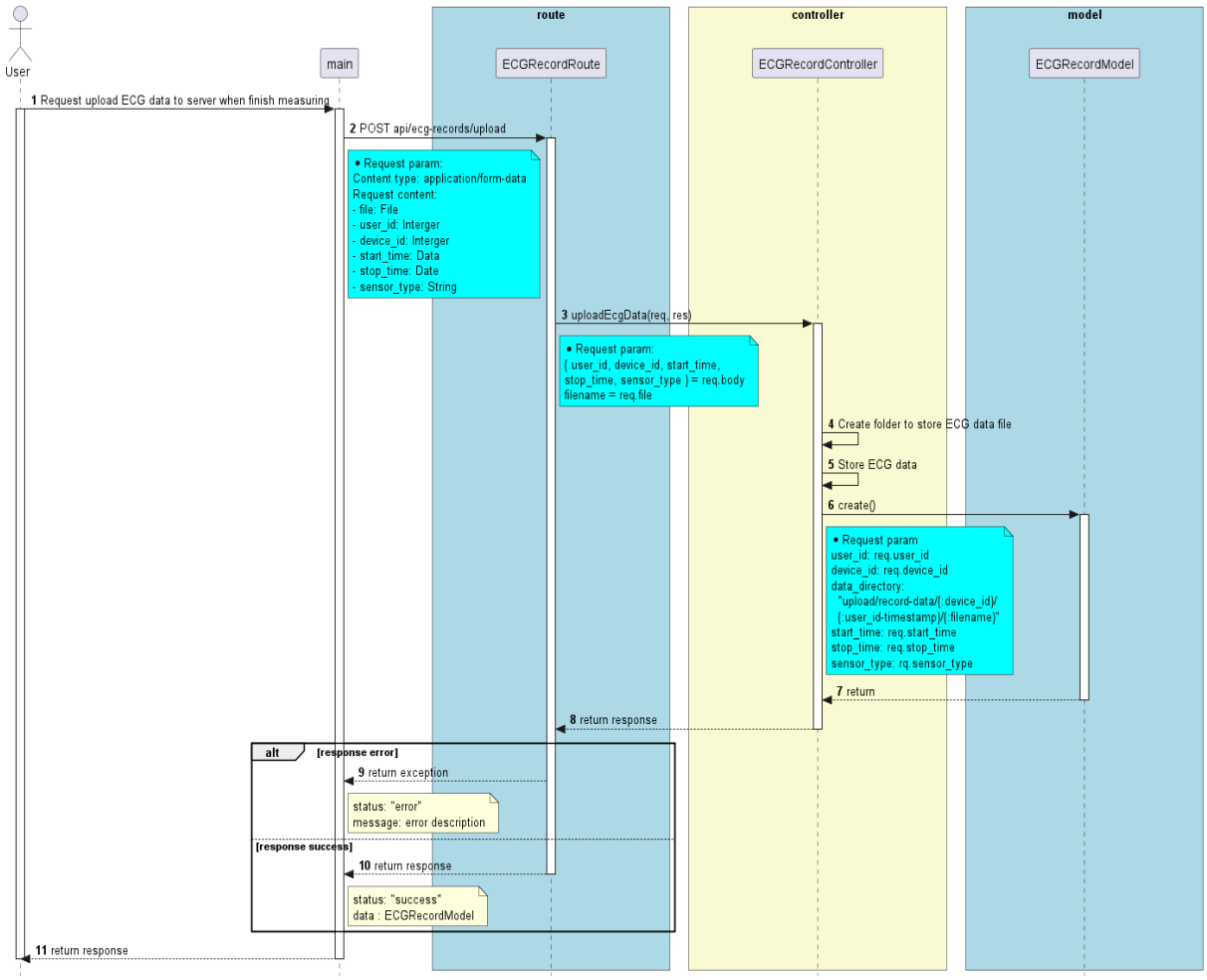
Hình 2.23 Sơ đồ tuần tự cho API lấy dữ liệu của một phiên đo ECG

Hình 2.23 mô tả quá trình yêu cầu lấy dữ liệu ECG (Electrocardiogram) từ một bản ghi cụ thể trong ứng dụng. Người dùng (User) gửi yêu cầu lấy dữ liệu ECG cho một bản ghi có record_id nhất định, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi ECGRecordController. ECGRecordController tiếp tục truy vấn ECGRecordModel để tìm kiếm bản ghi ECG dựa trên record_id được truyền vào từ yêu cầu. Sau đó, data_directory của bản ghi ECG được trả về từ ECGRecordModel và được gửi trở lại ECGRecordRoute để xử lý việc chuyển đổi dữ liệu từ định dạng file Excel sang định dạng JSON. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa dữ liệu ECG của bản ghi tương ứng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



Hình 2.24 Sơ đồ tuần tự cho API lấy thông tin các phiên đo ECG của một bệnh nhân

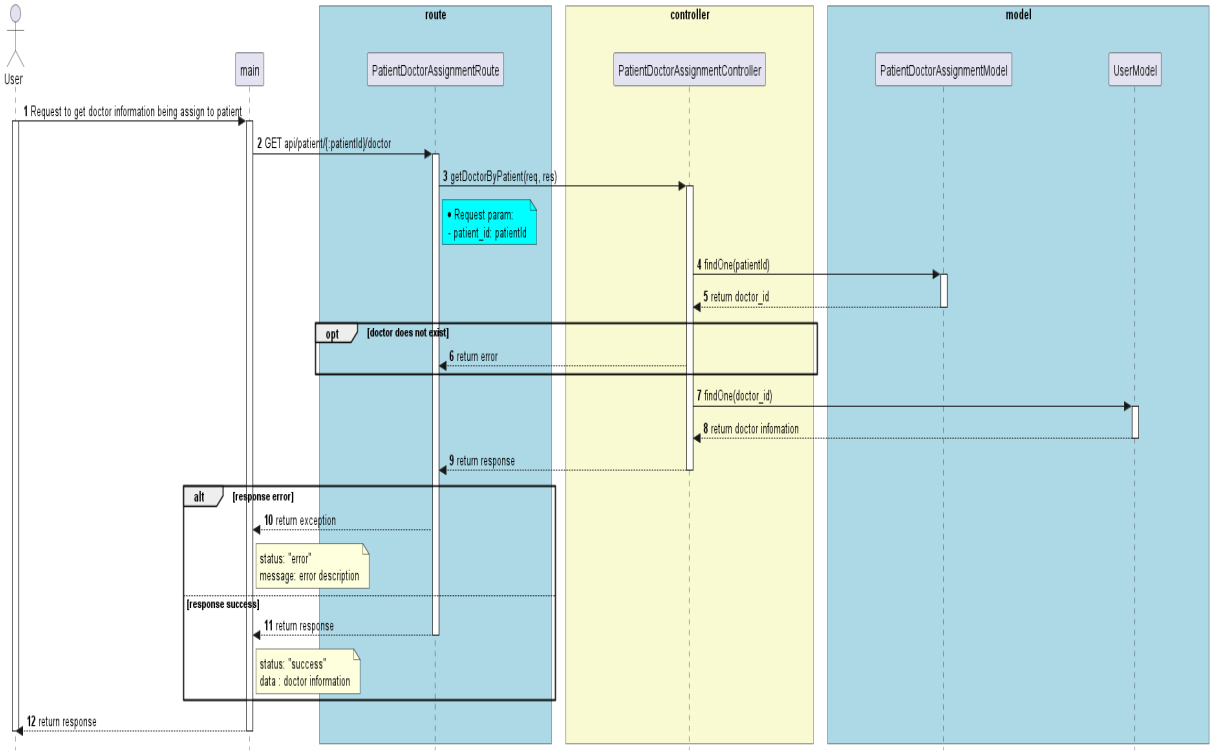
Hình 2.24 mô tả quá trình yêu cầu dữ liệu ECG (Electrocardiogram) của một bệnh nhân cụ thể từ ứng dụng. Người dùng (User) gửi yêu cầu lấy dữ liệu ECG cho một bệnh nhân với patientId nhất định, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi ECGRecordController. ECGRecordController tiếp tục truy vấn ECGRecordModel để tìm kiếm các bản ghi ECG của bệnh nhân dựa trên patientId được truyền vào từ yêu cầu. Sau đó, danh sách các bản ghi ECG của bệnh nhân được trả về từ ECGRecordModel và được gửi trở lại ECGRecordRoute để tạo response chứa danh sách này. Nếu quá trình thực hiện thành công, hệ thống trả về response chứa danh sách các bản ghi ECG của bệnh nhân tương ứng. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



Hình 2.25 Sơ đồ tuần tự cho API tải dữ liệu ECG lên server

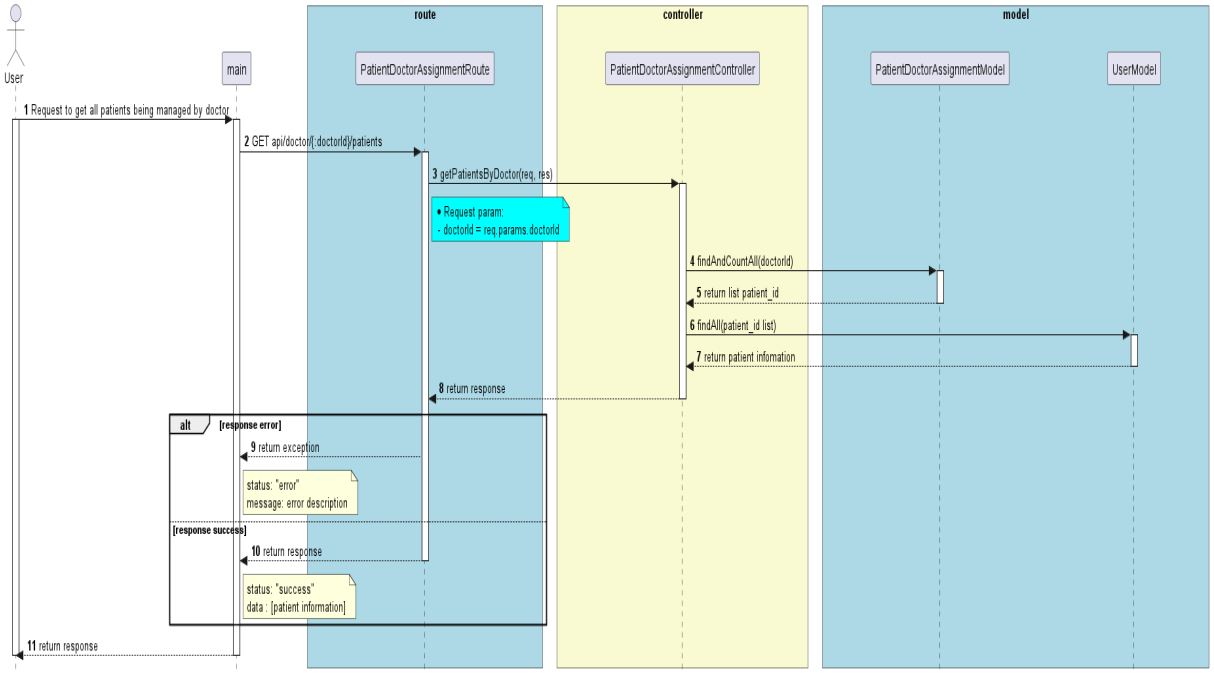
Hình 2.25 mô tả quá trình tải lên dữ liệu ECG (Electrocardiogram) sau khi hoàn thành việc đo đạc từ thiết bị đo ECG. Người dùng (User) gửi yêu cầu tải lên dữ liệu ECG cho một bệnh nhân cụ thể, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi ECGRecordController. ECGRecordController tiếp tục tạo folder mới để lưu trữ dữ liệu ECG, sau đó lưu dữ liệu ECG được tải lên vào folder vừa tạo. Tiếp theo, ECGRecordController tạo một bản ghi mới trong ECGRecordModel, lưu trữ thông tin về người dùng (user_id), thiết bị đo (device_id), thời gian bắt đầu đo (start_time), thời gian kết thúc đo (stop_time), loại cảm biến (sensor_type) và đường dẫn của dữ liệu ECG được lưu trữ. Sau đó, ECGRecordModel trả về thông tin vừa lưu vào ECGRecordController. Nếu quá trình tải lên thành công, hệ thống trả về response thành công chứa thông tin về bản ghi ECG vừa tạo. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.

2.3.4.5 API liên quan liên quan đến việc phân công bệnh nhân cho bác sỹ



Hình 2.26 Sơ đồ tuần tự cho API lấy thông tin bác sĩ được phân công cho một bệnh nhân

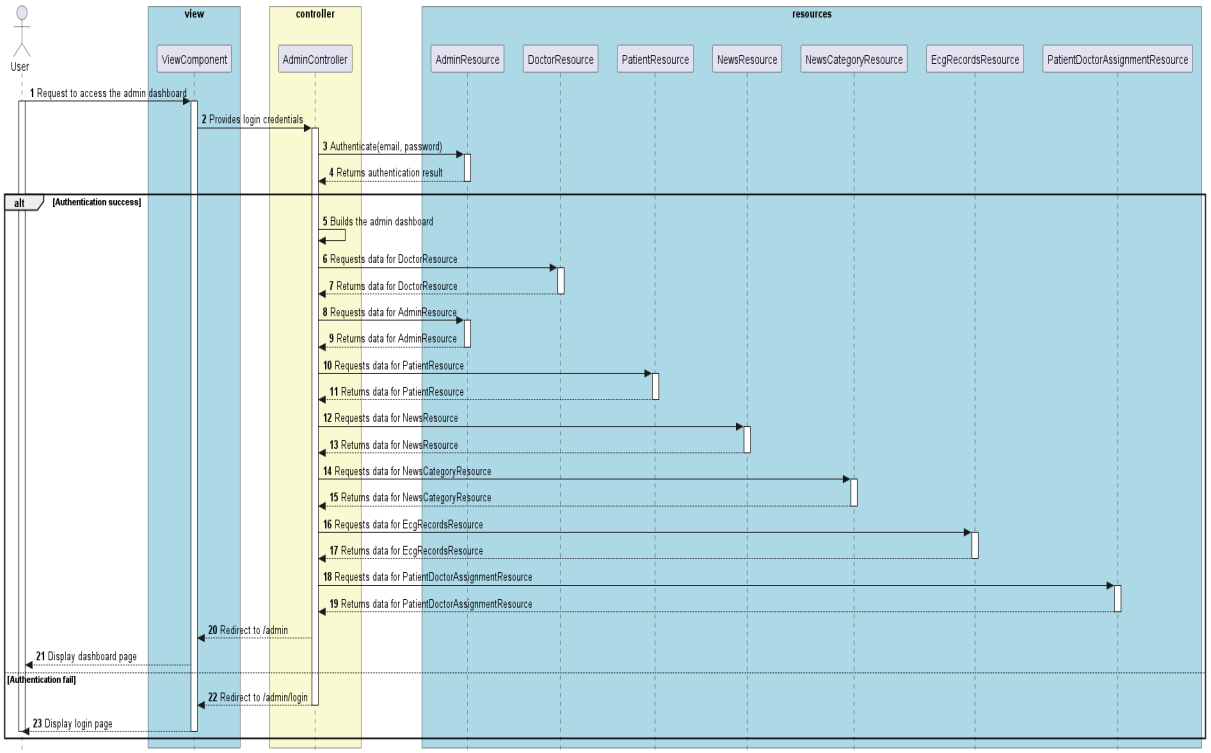
Hình 2.26 mô tả quá trình lấy thông tin bác sĩ được phân công cho một bệnh nhân cụ thể. Người dùng (User) gửi yêu cầu lấy thông tin bác sĩ của một bệnh nhân cụ thể, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi PatientDoctorAssignmentController. PatientDoctorAssignmentController tìm kiếm thông tin liên kết giữa bệnh nhân và bác sĩ trong PatientDoctorAssignmentModel bằng cách tìm bản ghi với patient_id trùng khớp với bệnh nhân được chỉ định. Nếu không tìm thấy thông tin liên kết, hệ thống trả về response lỗi tương ứng. Nếu tìm thấy thông tin liên kết, PatientDoctorAssignmentController sẽ tiếp tục tìm kiếm thông tin của bác sĩ trong UserModel bằng cách sử dụng doctor_id lấy từ bản ghi liên kết. Sau khi tìm thấy thông tin của bác sĩ, hệ thống trả về response thành công chứa thông tin về bác sĩ đó. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.



Hình 2.27 Sơ đồ tuần tự cho API lấy thông tin tất cả bệnh nhân được quản lý bởi một bác sĩ

Hình 2.27 mô tả quá trình lấy thông tin tất cả bệnh nhân được quản lý bởi một bác sĩ cụ thể. Người dùng (User) gửi yêu cầu lấy thông tin tất cả bệnh nhân của một bác sĩ cụ thể, thông qua các tầng của hệ thống, yêu cầu này được xử lý bởi PatientDoctorAssignmentController. PatientDoctorAssignmentController tìm kiếm thông tin liên kết giữa bác sĩ và bệnh nhân trong PatientDoctorAssignmentModel bằng cách tìm bản ghi với doctorId trùng khớp với bác sĩ được chỉ định. Sau đó, PatientDoctorAssignmentController tìm kiếm thông tin của tất cả bệnh nhân có trong danh sách patient_id tìm thấy, thông qua UserModel. Mỗi bệnh nhân sẽ có thông tin riêng bao gồm tên, tuổi, địa chỉ, v.v. Sau khi tìm thấy thông tin của tất cả bệnh nhân, hệ thống trả về response thành công chứa danh sách thông tin của các bệnh nhân đó. Nếu có lỗi xảy ra trong quá trình này, hệ thống trả về response lỗi tương ứng. Sau khi xử lý, response cuối cùng được trả về tới người dùng.

2.3.4.6 Web



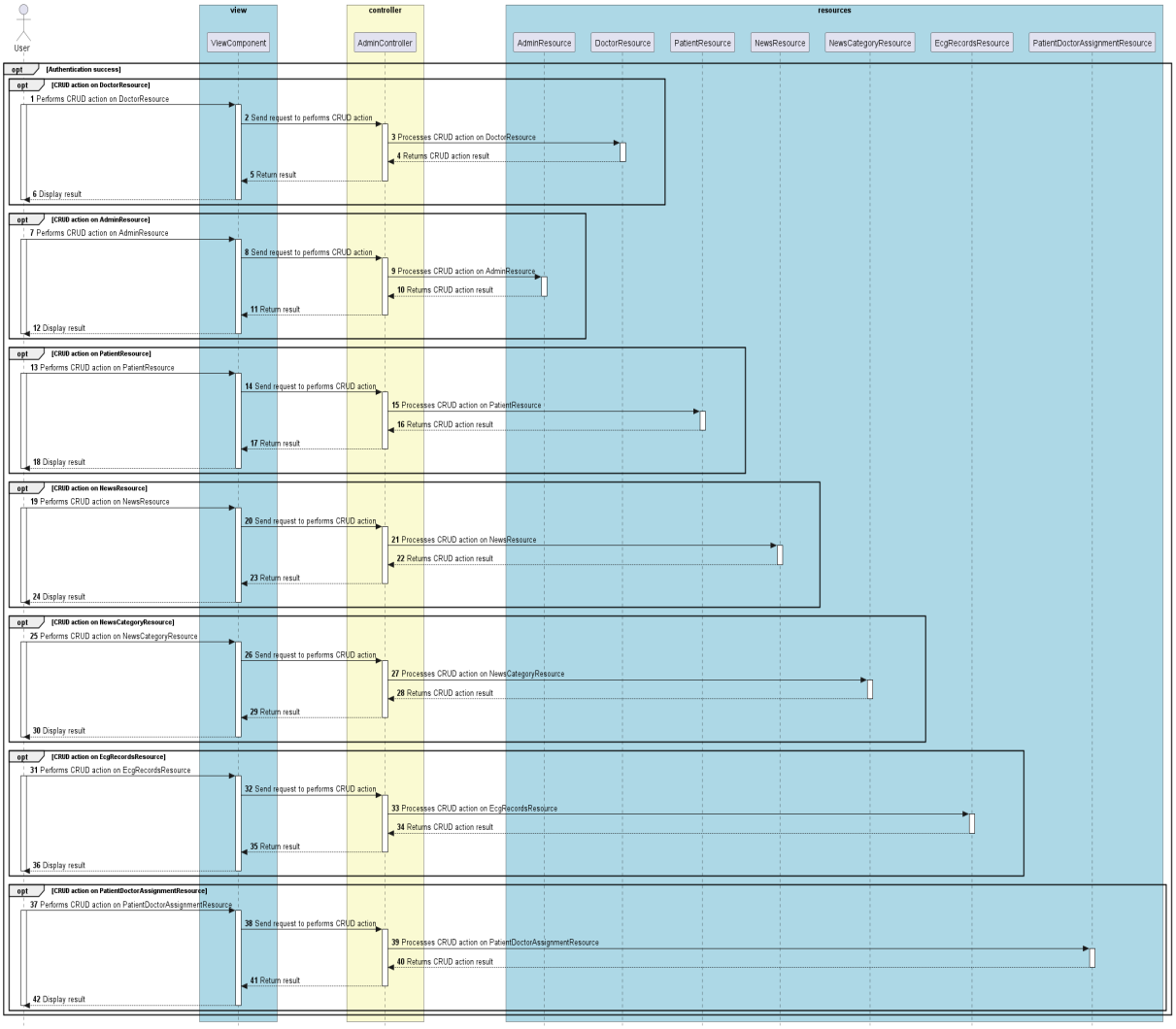
Hình 2.28 Sơ đồ tuần tự cho quá trình truy cập vào trang quản trị (admin dashboard)

Hình 2.28 mô tả quá trình truy cập vào bảng điều khiển quản trị (admin dashboard) của người dùng (User). Khi người dùng yêu cầu truy cập vào admin dashboard, hệ thống sẽ yêu cầu xác thực thông qua ViewComponent. Nếu xác thực thành công, AdminJS sẽ xây dựng admin dashboard và yêu cầu dữ liệu từ các tài nguyên (resources) khác nhau để hiển thị thông tin.

Nếu xác thực thành công, AdminJS sẽ yêu cầu dữ liệu từ các tài nguyên khác nhau, bao gồm DoctorResource, AdminResource, PatientResource, NewsResource, NewsCategoryResource, EcgRecordsResource và PatientDoctorAssignmentResource. Mỗi tài nguyên sẽ trả về dữ liệu tương ứng và AdminJS sẽ sử dụng các dữ liệu này để hiển thị trên admin dashboard.

Sau khi AdminJS đã thu thập đủ dữ liệu từ các tài nguyên, nó sẽ chuyển hướng ViewComponent đến trang /admin để hiển thị admin dashboard cho người dùng. Nếu xác thực không thành công, AdminJS sẽ chuyển hướng ViewComponent đến trang /admin/login để hiển thị trang đăng nhập cho người dùng.

Sau khi hoàn tất, hệ thống sẽ trả về response cuối cùng và hiển thị trang tương ứng cho người dùng.



Hình 2.29 Sơ đồ tuần tự cho quá trình trình thực hiện các thao tác CRUD trên các tài nguyên (resources)

Hình 2.29 mô tả quá trình thực hiện các thao tác CRUD trên các tài nguyên (resources) khác nhau của admin dashboard khi người dùng (User) thực hiện các thao tác trong ViewComponent.

Khi người dùng thực hiện các thao tác CRUD trên các tài nguyên, hệ thống sẽ xác thực người dùng trước đó. Sau khi xác thực thành công, người dùng sẽ thực hiện các thao tác CRUD trên ViewComponent, và ViewComponent sẽ gửi yêu cầu thực hiện thao tác tương ứng tới AdminJS.

Sau đó, AdminJS sẽ thực hiện thao tác CRUD tương ứng trên các tài nguyên, bao gồm DoctorResource, AdminResource, PatientResource, NewsResource, NewsCategoryResource, EcgRecordsResource và PatientDoctorAssignmentResource. Mỗi tài nguyên sẽ xử lý yêu cầu và trả về kết quả tương ứng cho AdminJS.

Sau khi AdminJS đã nhận được kết quả từ các tài nguyên, nó sẽ trả về kết quả đó

cho ViewComponent. ViewComponent sẽ hiển thị kết quả tương ứng cho người dùng. Quá trình này được thực hiện đối với từng tài nguyên mà người dùng yêu cầu thực hiện thao tác CRUD.

CHƯƠNG 3. TRIỂN KHAI VÀ KIỂM THỬ

3.1 Triển khai ứng dụng

Trong quá trình triển khai ứng dụng, chúng em sử dụng dịch vụ Elastic Compute Cloud (EC2) của AWS để chạy ứng dụng và sử dụng dịch vụ Relational Database Service (RDS) để lưu trữ cơ sở dữ liệu của ứng dụng. Việc sử dụng EC2 và RDS giúp chúng em tối ưu hóa việc quản lý hệ thống, đảm bảo tính sẵn sàng và mở rộng khả năng chịu tải cho ứng dụng.

3.1.1 Triển khai ứng dụng trên AWS EC2

- Tạo máy ảo EC2: Chúng em đã tạo một EC2 instance với loại instance t2.micro. Đây là loại instance nhỏ, phù hợp với các ứng dụng có lưu lượng truy cập thấp hoặc giai đoạn phát triển. Instance này sử dụng kiến trúc 64-bit, cho phép chạy các ứng dụng trên cả hệ thống 32-bit và 64-bit. Bộ nhớ của instance là 1 GiB, đủ để chạy ứng dụng Node.js cùng với các dependencies. Dung lượng lưu trữ 8GB, chúng tôi đã cài đặt hệ điều hành Linux/UNIX để chạy mã nguồn của ứng dụng.
- Cài đặt các phần mềm và dependencies: Sau khi triển khai máy ảo EC2, chúng em đã cài đặt các phần mềm và dependencies cần thiết để chạy ứng dụng, bao gồm Node.js và các thư viện hỗ trợ và các gói npm cần thiết.
- Tạo và cấu hình môi trường ứng dụng:Chúng em đã tạo môi trường ứng dụng, bao gồm việc cấu hình các biến môi trường, thiết lập các file cấu hình, và chạy các lệnh khởi tạo ban đầu cho ứng dụng.
- Triển khai ứng dụng: Tiếp theo, chúng em tải lên mã nguồn của ứng dụng lên máy ảo EC2 thông qua git
- Mở cổng cho ứng dụng: Chúng em đã mở cổng mạng trên máy ảo EC2 để cho phép ứng dụng lắng nghe các yêu cầu từ internet.
- Khởi động ứng dụng:Để khởi động ứng dụng, chúng em chạy các lệnh quản lý quá trình như npm start

3.1.2 Triển khai cơ sở dữ liệu trên RDS

- Tạo cơ sở dữ liệu RDS: Chúng em đã triển khai một cơ sở dữ liệu MySQL trên dịch vụ RDS của AWS. Cơ sở dữ liệu này được đặt tên là "fm_ecg" và có mật

khẩu "fmecgdatabase" để bảo mật. Địa chỉ host của cơ sở dữ liệu là "fm-ecg-database.cx3akkmg3hid.ap-southeast-1.rds.amazonaws.com" cho phép EC2 instance kết nối và truy vấn dữ liệu từ cơ sở dữ liệu. Để truy cập và quản lý cơ sở dữ liệu, chúng tôi sử dụng tài khoản "admin" với các quyền tương ứng.

- Kết nối giữa EC2 và RDS: Chúng em đã cấu hình ứng dụng Node.js chạy trên EC2 instance để kết nối đến cơ sở dữ liệu RDS thông qua thông tin host, tên người dùng và mật khẩu đã cung cấp. Khi ứng dụng gửi các truy vấn SQL đến cơ sở dữ liệu, RDS sẽ xử lý các truy vấn này và trả về kết quả cho ứng dụng.
- Backup và giám sát cơ sở dữ liệu: Chúng em đã thiết lập các chính sách sao lưu định kỳ cho cơ sở dữ liệu RDS để đảm bảo an toàn cho dữ liệu và khả năng khôi phục trong trường hợp xảy ra sự cố.

3.2 Kiểm thử

3.2.1 Kiểm thử hoạt động của các API

3.2.2 Kiểm thử ứng dụng web

KẾT LUẬN

Kết luận chung

Kết luận chung cho các chương trong đồ án. Mục này cần nhấn mạnh những vấn đề đã giải quyết và vấn đề chưa giải quyết để đưa ra các đánh giá về mức độ hoàn thành cô việc. Đánh giá này thường so sánh kết quả thu được với mục tiêu đề ra ban đầu

Hướng phát triển

(Nếu có) [1]

Kiến nghị và đề xuất

(nếu có)

TÀI LIỆU THAM KHẢO

[1] D. T. Nhu Y, N. T. Hoang, P. K. Lieu, H. Harada, N. Brion, D. V. Hieu, N. V. Hop, and H. Olde Venterink, "Effects of nutrient supply and nutrient ratio on diversity–productivity relationships of phytoplankton in the cau hai lagoon, vietnam," *Ecology and evolution*, vol. 9, no. 10, pp. 5950–5962, 2019.

PHỤ LỤC

Mã nguồn chương trình nếu có được đưa và đây sử dụng font Courier New, cỡ 10pt.

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho giản viên hướng dẫn)

Tên giảng viên đánh giá:

Họ và tên Sinh Viên: MSSV:

Tên đồ án:

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết va thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các cấn đề và các giả thuyết (bao gồm mục đíc và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đo được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng.	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai.	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy địnhg	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.).	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế	1	2	3	4	5
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.).	1	2	3	4	5
10c	Không có thành tích về nghiên cứu khoa học	1	2	3	4	5
Điểm tổng		**/50**				
Điểm tổng quy đổi về thang 10						

Nhận xét khác (về thái độ và tinh thần làm việc của sinh viên)

Hà Nội, ngày tháng năm

Người nhận xét

(Ký và ghi rõ họ tên)

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho cán bộ phản biện)

Tên giảng viên đánh giá:

Họ và tên Sinh Viên: MSSV:

Tên đồ án:

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:

Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết va thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các cấn đề và các giả thuyết (bao gồm mục đíc và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đo được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng.	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai.	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có căn lề, dấu cách sau dấu chấm, dấu phẩy v.v), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy địnhg	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.).	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) (chọn 1 trong 3 trường hợp)						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/đạt giải SVNC khoa học giải 3 cấp Viện trở lên/các giải thưởng khoa học (quốc tế/trong nước) từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế	1	2	3	4	5
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị sinh viên nghiên cứu khoa học nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành như TI contest.).	1	2	3	4	5
10c	Không có thành tích về nghiên cứu khoa học	1	2	3	4	5
Điểm tổng					**/50**	
Điểm tổng quy đổi về thang 10						

Nhận xét khác của cán bộ phản biện

Hà Nội, ngày tháng năm

Người nhận xét

(Ký và ghi rõ họ tên)