|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  **Đề tài:**  **XÂY DỰNG HỆ THỐNG THEO DÕI BỆNH NHÂN TỪ XA**  Sinh viên thực hiện: NGUYỄN THỊ HIỀN  Lớp ĐTVT.10-K62  Giảng viên hướng dẫn: ThS. VŨ SONG TÙNG  Hà Nội, 02-2022 |

|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  ĐỒ ÁN  **TỐT NGHIỆP ĐẠI HỌC**  **Đề tài:**  **XÂY DỰNG HỆ THỐNG THEO DÕI BỆNH NHÂN TỪ XA**  Sinh viên thực hiện: NGUYỄN THỊ HIỀN  Lớp ĐTVT.10-K62  Giảng viên hướng dẫn: ThS. VŨ SONG TÙNG  Cán bộ phản biện:  Hà Nội, 02-2022 |

**ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP**

(Dùng cho giảng viên hướng dẫn)

Tên giảng viên đánh giá: Ths Vũ Song Tùng

Họ và tên sinh viên: Nguyễn Thị Hiền MSSV: 20172538

Tên đồ án: Xây dựng hệ thống theo dõi bệnh nhân từ xa

**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 và thực hành (20)** | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích 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 đạt 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; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | 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 SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế | 5 | | | | |
| 10b | Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH 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 (VD: TI contest) | 2 | | | | |
| 10c | Không có thành tích về nghiên cứu khoa học | 0 | | | | |
| **Điểm tổng** | | **/50** | | | | |
| **Điểm tổng quy đổi về thang 10** | |  | | | | |

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

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Người nhận xét**  (Ký và ghi rõ họ tên)  Ths Vũ Song Tùng |

**ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP**

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá:

Họ và tên sinh viên: Nguyễn Thị Hiền MSSV: 20172538

Tên đồ án: Xây dựng hệ thống theo dõi bệnh nhân từ xa

**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 và thực hành (20)** | | | | | | |
| 1 | Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích 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 đạt 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; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định | 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 SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế | 5 | | | | |
| 10b | Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH 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 (VD: TI contest) | 2 | | | | |
| 10c | Không có thành tích về nghiên cứu khoa học | 0 | | | | |
| **Điểm tổng** | | **/50** | | | | |
| **Điểm tổng quy đổi về thang 10** | |  | | | | |

***Nhận xét khác của cán bộ phản biện***

|  |  |
| --- | --- |
|  | Ngày: … / … / 20…  **Người nhận xét**  (Ký và ghi rõ họ tên) |

**LỜI NÓI ĐẦU**

Ngành Công nghệ thông tin và hệ thống IoT – Internet of Things là vấn đề đang rất được quan tâm, chú trọng hiện nay và có thể phát triển rộng rãi hơn trong tương lai. Do tính tiện dụng nên hệ thống IoT được coi là xu hướng phát triển nổi bật trong thời kỳ công nghiệp 4.0. Đây là thời kỳ của trí tuệ nhân tạo, tự động hóa với các thiết bị thông minh được kết nối với nhau thông qua mạng internet (vạn vật kết nối). Các thiết bị trong hệ thống đều được lập trình để có thể làm việc tự động, giúp cuộc sống của con người hiện đại hơn, thông minh hơn, có thể tiết kiệm thời gian và chi phí.

Hiện nay trong tình hình dịch covid đang bùng phát mạnh mẽ ở rất nhiều nơi trên toàn thế giới trong đó có Việt Nam chúng ta, điều này khiến cho các bệnh viện và các y bác sĩ có rất nhiều gánh nặng đặc biệt khi số lượng bệnh nhân quá lớn. Vì vậy, việc loại bỏ bớt việc tiếp xúc với các bệnh nhân là điều cần thiết. Nhận ra những thiếu xót và bất cập như trên, hệ thống “Theo dõi bệnh nhân từ xa” được đưa ra phân tích và thiết kế, với mong muốn giúp giảm tải áp lực cho hệ thống y tế tại Việt Nam và giảm thiếu chi phí phía bệnh nhân, đáp ứng được yêu cầu chăm sóc sức khỏe của mọi người dân.

Trước nhu cầu của người dùng và hướng phát triển trên, tác giả đã quyết định chọn đề tài: “XÂY DỰNG HỆ THỐNG THEO DÕI BỆNH NHÂN TỪ XA” với mục đích nghiên cứu, tìm hiểu và xây dựng ứng dụng có đủ những chức năng cần thiết. Ứng dụng này cho phép bác sĩ có thể theo dõi sức khỏe cùng lúc nhiều bệnh nhân mà không mất quá nhiều thời gian thăm khám. Nội dung đồ án gồm có 3 chương:

*Chương 1: Tổng quan hệ thống theo dõi bệnh nhân từ xa*. Chương này sẽ trình bày về mô hình tổng quan, mô tả rõ hoạt động của hệ thống, giới thiệu một số giải pháp và công cụ được sử dụng trong đề tài.

*Chương 2: Phân tích thiết kế hệ thống*. Chương này trình bày những yêu cầu của hệ thống về chức năng và phi chức năng. Ngoài ra, còn tạo ra sơ đồ Use Case, tìm hiểu thiết kế biểu đồ lớp (Class Diagram) cho Models chung và cuối cùng là thiết kế luồng đi của hệ thống, luồng đi của dữ liệu thông qua biểu đồ hoạt động (Activity Diagram) và biểu đồ tuần tự (Sequence Diagram).

*Chương 3: Triển khai mô hình MVC trên Xamarin*. Chương này trình bày về quá trình tiến hành triển khai ứng dụng theo dõi bệnh nhân từ xa bằng mô hình MVC với công cụ Xamarin, Visual Studio.

Do kiến thức vẫn còn hạn chế, bài báo cáo này không tránh khỏi những sai sót. Tác giả rất mong nhận được những lời góp ý từ quý thầy cô để đồ án tốt nghiệp được hoàn thiện và giúp tác giả có thêm những kinh nghiệm quý báu.

**LỜI CẢM ƠN**

Trong suốt khoảng thời gian học tập tại Bách Khoa, tác giả đã có nhiều kỷ niệm vui buồn, và trải qua quá trình học tập kiến thức cũng như rèn luyện kỹ năng mềm cho bản thân. Tác giả xin dành lời cảm ơn sâu sắc đến các thầy cô đã tận tình dạy dỗ, truyền đạt kiến thức, kinh nghiệm và hỗ trợ sinh viên hết sức mình trong suốt những năm vừa qua.

Tác giả vô cùng biết ơn giảng viên hướng dẫn ThS. Vũ Song Tùng đã tận tình hướng dẫn, chỉ bảo, góp ý trong suốt quá trình thực hiện đề tài để tác giả có thể hoàn thành đồ án của mình. Mặc dù tình hình dịch căng thẳng, khó khăn khiến quá trình làm việc phải thực hiện online nhưng thầy đã chỉ dạy, giúp đỡ sinh viên hết sức tận tình khi gặp khó khăn trong lúc thực hiện đồ án. Tác giả cũng vô cùng biết ơn gia đình và bạn bè, những người đã luôn luôn ở bên cạnh, động viên, chia sẻ trong suốt thời gian thực hiện đề tài đồ án tốt nghiệp.

Cuối cùng, tác giả xin kính chúc các thầy cô trường Đại học Bách khoa Hà Nội nói chung, các thầy cô trường Điện – Điện tử Viện Điện tử - Viễn thông nói riêng sức khỏe dồi dào, thành công trong sự nghiệp giảng dạy cao quý của mình.

**LỜI CAM ĐOAN**

Tôi là Nguyễn Thị Hiền, mã số sinh viên là 20172538, sinh viên lớp ĐTVT-10-K62, khóa 62. Người hướng dẫn là Ths. Vũ Song Tùng. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án “*Xây dựng hệ thống theo dõi bệnh nhân từ xa*” là kết quả quá trình tìm hiểu và nghiên cứu của tôi. 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 2021

Người cam đoan

**Nguyễn Thị Hiền**

**MỤC LỤC**

[DANH MỤC HÌNH VẼ iii](#_Toc96992117)

[DANH MỤC BẢNG BIỂU v](#_Toc96992118)

[DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT vi](#_Toc96992119)

[TÓM TẮT ĐỒ ÁN vii](#_Toc96992120)

[CHƯƠNG 1. TỔNG QUAN HỆ THỐNG THEO DÕI BỆNH NHÂN TỪ XA 1](#_Toc96992121)

[1.1 Lý do chọn đề tài 1](#_Toc96992122)

[1.2 Giới thiệu tổng quan hệ thống 2](#_Toc96992123)

[1.2.1 Mô hình hệ thống 2](#_Toc96992124)

[1.2.2 Nguyên tắc hoạt động của hệ thống 3](#_Toc96992125)

[1.3 Phương thức kết nối 3](#_Toc96992126)

[1.3.1 Đánh giá khả năng kết nối 3](#_Toc96992127)

[1.3.2 Tổng quan về MQTT 4](#_Toc96992128)

[1.4 Cơ sở dữ liệu 5](#_Toc96992129)

[1.4.1 So sánh cơ sở dữ liệu SQL và NoSQL 5](#_Toc96992130)

[1.4.2 Tổng quan về MongoDB 6](#_Toc96992131)

[1.5 Tổng quan về mô hình Model-View-Controller (MVC) 6](#_Toc96992132)

[1.5.1 Khái niệm 6](#_Toc96992133)

[1.5.2 Đặc điểm của mô hình MVC 7](#_Toc96992134)

[1.6 Giới thiệu về công cụ và kỹ thuật được sử dụng 7](#_Toc96992135)

[1.6.1 Ngôn ngữ C# 7](#_Toc96992136)

[1.6.2 Visual Studio 8](#_Toc96992137)

[1.6.3 Xamarin 10](#_Toc96992138)

[1.6.4 Windows Presentation Foundation (WPF) 11](#_Toc96992139)

[1.7 Kết luận 12](#_Toc96992140)

[CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ HỆ THỐNG 13](#_Toc96992141)

[2.1 Yêu cầu hệ thống 13](#_Toc96992142)

[2.1.1 Yêu cầu chức năng 13](#_Toc96992143)

[2.1.2 Yêu cầu phi chức năng 13](#_Toc96992144)

[2.2 Xây dựng Use case Diagram 13](#_Toc96992145)

[2.2.1 Tìm hiểu biểu đồ Use case 13](#_Toc96992146)

[2.2.2 Bản vẽ Use case 15](#_Toc96992147)

[2.3 Xây dựng Class Diagram 19](#_Toc96992148)

[2.3.1 Tìm hiểu biểu đồ lớp (Class Diagram) 19](#_Toc96992149)

[2.3.2 Bản vẽ Class Diagram 21](#_Toc96992150)

[2.4 Xây dựng Activity Diagram 22](#_Toc96992151)

[2.4.1 Tìm hiểu biểu đồ hoạt động (Activity Diagram) 22](#_Toc96992152)

[2.4.2 Bản vẽ Activity Diagram 24](#_Toc96992153)

[2.5 Xây dựng Sequence Diagram 30](#_Toc96992154)

[2.5.1 Tìm hiểu biểu đồ tuần tự (Sequence Diagram) 30](#_Toc96992155)

[2.5.2 Bản vẽ Sequence Diagram 31](#_Toc96992156)

[2.6 Xây dựng giao thức liên lạc 33](#_Toc96992157)

[2.7 Kết luận 35](#_Toc96992158)

[CHƯƠNG 3. TRIỂN KHAI MÔ HÌNH MVC TRÊN XAMARIN 36](#_Toc96992159)

[3.1 Tìm hiểu về Xamarin và Xamarin Forms 36](#_Toc96992160)

[3.1.1 Xamarin 36](#_Toc96992161)

[3.1.2 Xamarin Forms 38](#_Toc96992162)

[3.2 Triển khai trên Xamarin 41](#_Toc96992163)

[3.2.1 Tạo project trên Xamarin (Windows) 41](#_Toc96992164)

[3.2.2 Tiến hành Code phần Base 43](#_Toc96992165)

[3.3 Giao diện của một số chức năng đã thực hiện 49](#_Toc96992166)

[3.3.1 Giao diện đăng nhập 49](#_Toc96992167)

[3.3.2 Giao diện danh sách bệnh nhân 50](#_Toc96992168)

[3.3.3 Giao diện cài đặt 51](#_Toc96992169)

[3.3.4 Giao diện đổi mật khẩu 52](#_Toc96992170)

[3.4 Kết luận 53](#_Toc96992171)

[KẾT LUẬN 54](#_Toc96992172)

[Kết luận chung 54](#_Toc96992173)

[Hướng phát triển 54](#_Toc96992174)

[TÀI LIỆU THAM KHẢO 55](#_Toc96992175)

DANH MỤC HÌNH VẼ

[Hình 1.1 Mô hình tổng quan của hệ thống 2](#_Toc96991506)

[Hình 1.2 Mô hình MVC 7](#_Toc96991507)

[Hình 2.1 Sơ đồ Use Case tổng quát của hệ thống 15](#_Toc96991508)

[Hình 2.2 Sơ đồ use case cho trung tâm Y tế 16](#_Toc96991509)

[Hình 2.3 Sơ đồ use case cho bác sĩ 17](#_Toc96991510)

[Hình 2.4 Sơ đồ use case cho bệnh nhân 17](#_Toc96991511)

[Hình 2.5 Usecase quản lý bệnh nhân 18](#_Toc96991512)

[Hình 2.6 Usecase quản lý bác sĩ 18](#_Toc96991513)

[Hình 2.7 Ví dụ về các tính chất cơ bản của class 19](#_Toc96991514)

[Hình 2.8 Ví dụ về các phạm vi truy cập của class 19](#_Toc96991515)

[Hình 2.9 Các loại quan hệ trong class diagram 20](#_Toc96991516)

[Hình 2.10 Bản vẽ class diagram cho Model 21](#_Toc96991517)

[Hình 2.11 Biểu đồ hoạt động chức năng đăng nhập 24](#_Toc96991518)

[Hình 2.12 Biểu đồ hoạt động chức năng đổi mật khẩu 25](#_Toc96991519)

[Hình 2.13 Biểu đồ hoạt động chức năng xem chỉ số bệnh nhân 26](#_Toc96991520)

[Hình 2.14 Biểu đồ hoạt động chức năng cảnh báo 27](#_Toc96991521)

[Hình 2.15 Biểu đồ hoạt động chức năng quản lý bác sĩ 28](#_Toc96991522)

[Hình 2.16 Biểu đồ hoạt động chức năng quản lý bệnh nhân 29](#_Toc96991523)

[Hình 2.17 Biểu đồ tuần tự chức năng đăng nhập 31](#_Toc96991524)

[Hình 2.18 Biểu đồ tuần tự chức năng đổi mật khẩu 32](#_Toc96991525)

[Hình 2.19 Biểu đồ tuần tự chức năng xem chỉ số bệnh nhân 32](#_Toc96991526)

[Hình 2.20 Biểu đồ tuần tự chức năng cảnh báo 33](#_Toc96991527)

[Hình 3.1 Xamarin chia sẻ code trên nhiều nền tảng 36](#_Toc96991528)

[Hình 3.2 Một page trong Xamarin Forms 38](#_Toc96991529)

[Hình 3.3 Các pages trong Xamarin Forms 39](#_Toc96991530)

[Hình 3.4 Các layouts trong Xamarin Forms 40](#_Toc96991531)

[Hình 3.5 Tạo một project mới 41](#_Toc96991532)

[Hình 3.6 Chọn loại project 42](#_Toc96991533)

[Hình 3.7 Target Platforms 42](#_Toc96991534)

[Hình 3.8 New Empty Project 43](#_Toc96991535)

[Hình 3.9 Tạo thư mục theo mô hình MVC trong Project 43](#_Toc96991536)

[Hình 3.10 Class BaseController 44](#_Toc96991537)

[Hình 3.11 Class ResponseController 45](#_Toc96991538)

[Hình 3.12 Class BaseView 46](#_Toc96991539)

[Hình 3.13 Class MyNavigationRootPage 46](#_Toc96991540)

[Hình 3.14 Class MyNavigationItemPage 47](#_Toc96991541)

[Hình 3.15 Class MyMenuView 47](#_Toc96991542)

[Hình 3.16 Class MyListView 48](#_Toc96991543)

[Hình 3.17 Code giao diện đăng nhập 49](#_Toc96991544)

[Hình 3.18 Giao diện đăng nhập 49](#_Toc96991545)

[Hình 3.19 Code giao diện danh sách bệnh nhân 50](#_Toc96991546)

[Hình 3.20 Giao diện danh sách bệnh nhân 50](#_Toc96991547)

[Hình 3.21 Code giao diện cài đặt 51](#_Toc96991548)

[Hình 3.22 Giao diện cài đặt 51](#_Toc96991549)

[Hình 3.23 Code giao diện đổi mật khẩu 52](#_Toc96991550)

[Hình 3.24 Giao diện đổi mật khẩu 52](#_Toc96991551)

DANH MỤC BẢNG BIỂU

[Bảng 1.1 So sánh MQTT protocols và HTTP protocols 3](#_Toc96991552)

[Bảng 1.2 So sánh cơ sở dữ liệu SQL và NoSQL 5](#_Toc96991553)

[Bảng 2.1 Các phần tử của biểu đồ Use Case 14](#_Toc96991554)

[Bảng 2.2 Các thành phần của biểu đồ tuần tự 30](#_Toc96991555)

[Bảng 2.3 Danh sách các request 34](#_Toc96991556)

[Bảng 3.1 Mô tả chi tiết các pages trong Xamarin Forms 39](#_Toc96991557)

[Bảng 3.2 Mô tả chi tiết các kiểu view trong Xamarin Forms 40](#_Toc96991558)

[Bảng 3.3 Mô tả chi tiết các layouts trong Xamarin Forms 41](#_Toc96991559)

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

|  |  |
| --- | --- |
| **Kí hiệu** | **Thuật ngữ đầy đủ** |
| IoT | Internet of Thing |
| MQTT | Message Queuing Telemetry Transport |
| MVC | Model-View-Controller |
| NoSQL | Non relational Structured Query Language |
| SQL | Structured Query Language |
| RDBMS | Relational Database Management System |
| WPF | Windows Presentation Foundation |
| HTTP | Hyper Text Transfer Protocol |
| XAML | eXtensible Application Markup Language |
| IDE | Integrated Developement Environment |

# TÓM TẮT ĐỒ ÁN

Hiện nay, với sự phát triển mạnh mẽ của Công nghiệp 4.0, các hệ thống IoT đang ngày càng phát triển để phục vụ đời sống con người hiện đại, thông minh hơn. Đến từ nhu cầu theo dõi sức khỏe bệnh nhân từ xa trong tình hình dịch covid phức tạp, và trong tương lai với mong muốn giảm áp lực cho bệnh viện và kinh tế của người bệnh việc phát triển hệ thống theo dõi bệnh nhân từ xa là thực sự cần thiết.

Đề tài “Xây dựng hệ thống theo dõi bệnh nhân” có nội dung chính là phân tích thiết kế các chức năng các luồng dữ liệu của hệ thống, tìm hiểu về Xamrin - xây dựng các ứng dụng dành cho SmartPhone, nắm được tổng thể các phần trong một ứng dụng và cách thiết kế nên ứng dụng trên Xamarin. Với mục đích trau dồi kỹ năng làm việc nhóm, phân tích thiết kế hệ thống và khả năng tự tìm hiểu, nâng cao kiến thức về các kỹ thuật công nghệ mới, tác giả đã thực hành và viết nên ứng dụng theo dõi bệnh nhân này này bằng phần mềm Microsoft Visual Studio, dựa trên ngôn ngữ C#. Đồ án này gồm có 3 chương, trong đó chương đầu tiên sẽ trình bày về mô hình tổng quan, mô tả rõ hoạt động của hệ thống, giới thiệu một số giải pháp và công cụ được sử dụng trong đề tài. Chương thứ hai trình bày những yêu cầu của hệ thống về chức năng và phi chức năng, thiết kế các biểu đồ chức năng, biểu đồ lớp, biểu đồ hoạt đồng và biểu đồ tuần tự. Chương cuối cùng trình bày về quá trình tiến hành triển khai ứng dụng theo dõi bệnh nhân từ xa bằng mô hình MVC với công cụ Xamarin, Visual Studio.

Trong gần 4 tháng thực hiện đề tài, ứng dụng đã đạt được một số chức năng cơ bản cho một ứng dụng theo dõi bệnh nhân từ xa và một số giao diện cơ sở để ứng dụng có thể dễ dàng phát triển hơn trong tương lai.

**ABSTRACT**

Nowadays, with the development of The Fourth Industrial Revolution, IoT systems are increasingly developing to serve modern and smarter human life. Coming from the need to monitor the patient's health remotely during the complicated covid epidemic situation, and in the future with the desire to reduce pressure on the hospital and the patient's economy, the development of a remote patient monitoring system is necessary.

The topic "Building a patient monitoring system" has the main content is analyzing and designing the functions of the system's data streams, learning about Xamrin - building applications for SmartPhone, understanding the overall parts of an app and how to design apps on Xamarin. With the purpose of cultivating teamwork skills, system analysis and design, and the ability to self-study and improve knowledge about new technology techniques, the author has practiced and written this patient monitoring application using Microsoft Visual Studio software, based on C# language. This project consists of 3 chapters, in which the first chapter will present the overview model, clearly describe the operation of the system, introduce some solutions and tools used in the project. The second chapter presents functional and non-functional system requirements, designing User case Diagram, Class Diagram, Activity Diagram, and Sequence Diagram. The last chapter presents the process of deploying remote patient monitoring application using MVC model with Xamarin tools, Visual Studio.

In nearly 4 months of implementing the project, the application has achieved some basic functions for a remote patient monitoring application and some basic interfaces so that the application can be more easily developed in the future.

# TỔNG QUAN HỆ THỐNG THEO DÕI BỆNH NHÂN TỪ XA

Sau khi cần phân tích, chúng ta đã có cho những lý do phù hợp để chọn đề tài thực hiện này. Từ đó, lập kế hoạch và đặt mục tiêu để hoàn thành dự án. Nội dung sau sẽ trình bày về mô hình tổng quan, mô tả rõ hoạt động của hệ thống, giới thiệu một số giải pháp và công cụ được sử dụng trong đề tài.

## Lý do chọn đề tài

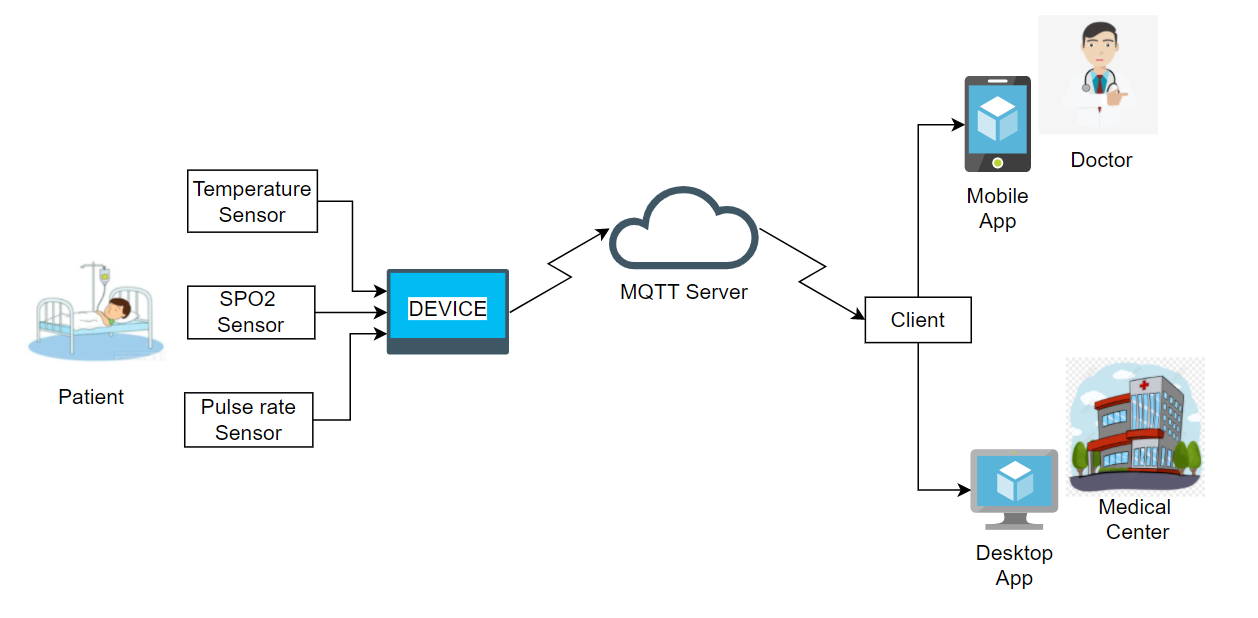
Y tế Việt Nam trong bối cảnh nền kinh tế thị trường đang phải đối mặt với nhiều vấn đề nghiêm trọng, đáng nói nhất là tình trạng quá tải bệnh viện. Hiện tượng nhức nhối này là mối quan tâm cấp bách cần giải quyết ngay của ngành Y tế và là câu hỏi chất vấn toàn xã hội đặt ra mà từ lâu đã không có lời giải. Đặc biệt là hiện nay trong tình hình dịch covid đang bùng phát mạnh mẽ ở rất nhiều nơi trên toàn thế giới trong đó có Việt Nam chúng ta, điều này khiến cho các bệnh viện và các y bác sĩ có rất nhiều gánh nặng đặc biệt khi số lượng bệnh nhân quá lớn. Vì vậy, việc loại bỏ bớt việc tiếp xúc với các bệnh nhân là điều cần thiết.

Trên thực tế, có một số bệnh mãn tính cần theo dõi dài ngày, như vậy bệnh nhân sẽ phải ở lại bệnh viện trong thời gian dài, chi phí sẽ rất tốn kém và như đã nói ở trên dễ dẫn đến tình trạng quá tải giường bệnh cũng như quá tải cho nhân viên chăm sóc. Trong những trường hợp như vậy, các giải pháp theo dõi bệnh nhân sức khỏe từ xa đóng một vai trò rất quan trọng. Nhận ra những thiếu xót và bất cập như trên, hệ thống “Theo dõi bệnh nhân từ xa” được đưa ra phân tích và thiết kế, với mong muốn giúp giảm tải áp lực cho hệ thống y tế tại Việt Nam và giảm thiếu chi phí phía bệnh nhân, đáp ứng được yêu cầu chăm sóc sức khỏe của mọi người dân.

Cụ thể, nếu áp dụng hệ thống này, bác sĩ có thể quản lý cùng lúc nhiều bệnh nhân mà không mất quá nhiều thời gian thăm khám. Hệ thống cũng có thể hỗ trợ bệnh nhân từ xa, điều này giúp người bệnh có thể chủ động nắm bắt tình trạng sức khỏe của bản thân mà không cần phải bỏ một số tiền lớn để nhận những dịch vụ khám bệnh riêng tư đắt đỏ. Nhờ đó, tình trạng quá tải bệnh viện cũng được khắc phục.

## Giới thiệu tổng quan hệ thống

### Mô hình hệ thống



Hình 1.1 Mô hình tổng quan của hệ thống

Mô tả hệ thống:

* Hệ thống gồm các khối device, server và client, và các đối tượng tham gia như bệnh nhân, bác sĩ, trung tâm y tế.
* Device bao gồm nhiều cảm biến đo các chỉ số sinh tồn của bệnh nhân, các chỉ số được cập nhật liên tục, nhiều lần trong ngày, mỗi khi phát hiện bất thường device sẽ có nhiệm vụ báo về server thông qua giao thức MQTT.
* Server nhận các bản tin và phân phối các bản tin cho client đã đăng kí thông qua giao thức MQTT.
* Client bao gồm các ứng dụng mobile và desktop được cung cấp cho các end-users để theo dõi thông tin và nhận cảnh báo mỗi khi xảy ra bất thường.
* Khi người dùng (App Android/IOS) đăng nhập vào tài khoản thì phía server sẽ gửi một token về phía người dùng, với token này app Android sẽ gửi lệnh cho MQTT broker server, server sẽ xác định xem đó là người dùng nào rồi chuyển cho thiết bị hoặc phía client khác, sau đó phản hồi lại kết quả cho người dùng thông qua MQTT

### Nguyên tắc hoạt động của hệ thống

Hệ thống được thiết lập để kết nối các thiết bị đo lường trong y tế với các thiết bị điện tử như là smart phone, laptop, ... nhằm giúp cho việc theo dõi bệnh nhân trở nên thuận tiện mà không mất nhiều công sức. Một hệ thống theo dõi bệnh nhân cơ bản sẽ bao gồm một máy chủ (server) nhằm kết nối các thiết bị với nhau và giúp chúng trao đổi thông tin với nhau.

Các thiết bị đầu cuối gọi chung là Client (laptop, smart phone, …) sẽ đượng kết nối với nhau trong hệ thống thông qua các phần mềm(applications) hoạt động trên nhiều nền tảng bao gồm Mobile app và Desktop app. Thông qua các ứng dụng thiết bị sẽ được kết nối đến Server và người dùng có thể theo dõi và nhận được thông tin của người bệnh được cung cấp qua server ở bất kì đâu, bất kì lúc nào.

## Phương thức kết nối

### Đánh giá khả năng kết nối

Về phần giao thức kết nối, có khá nhiều loại trong đó có HTTP, MQTT, ... Trong đó giao thức MQTT là giao thức được lựa chọn để sử dụng trong hệ thống vì đây là một giao thức nhẹ, lý tưởng cho hầu hết các kết nối IoT và M2M vì yêu cầu băng thông thấp và có độ tin cậy cao.

Sự so sánh về đặc điểm của hai giao thức kết nối MQTT và HTTP được thể hiện rõ trong bảng 1-1 [1]. Từ đó, chúng ta có thể thấy được ưu điểm của giao thức MQTT.

Bảng 1.1 So sánh MQTT protocols và HTTP protocols

|  |  |  |
| --- | --- | --- |
| Đặc điểm | MQTT PROTOCOL | HTTP PROTOCOL |
| Tên đầy đủ | Message Queuing Telemetry Transport | Hyper Text Transfer Protocol |
| Kích thước bản tin | nhỏ, là mã nhị phân với header là 2 byte | lớn, mã hóa bằng ASCII |
| Bảo mật dữ liệu | Có | Không, https mới cung cấp bảo mật dữ liệu |
| Độ phức tạp | đơn giản | phức tạp |

### Tổng quan về MQTT

#### Khái niệm

MQTT (Message Queuing Telemetry Transport) là một giao thức gửi dạng publish/subscribe sử dụng cho các thiết bị (IoT) với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Được thiết kế nhẹ và linh hoạt, do đó nó chỉ có một lớp bảo mật ở tầng ứng dụng: bảo mật bằng xác thực (xác thực các client được truy cập tới broker).

Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M [2].

#### Đặc điểm

Trong một hệ thống sử dụng giao thức MQTT, nhiều client kết nối tới một server (trong MQTT, server được gọi là MQTT Broker). Mỗi client sẽ đăng kí theo dõi các kênh thông tin (gọi là Topic) hoặc gửi dữ liệu lên kênh thông tin đó. Quá trình đăng ký này gọi là “Subscribe” và hành động một client gửi dữ liệu lên kênh thông tin được gọi là “Publish”. Mỗi khi thông tin đó được cập nhật dữ liệu (dữ liệu này có thể đến từ các client khác) thì những client nào đăng ký theo dõi kênh này sẽ nhận được dữ liệu cập nhật đó.

#### Ưu điểm

MQTT có đặc trưng riêng và rất lý tưởng để sử dụng trong các môi trường bị giới hạn tài nguyên như:

* Những nơi mà giá mạng quá đắt hoặc băng thông thấp, hoặc độ tin cậy thấp
* Khi chạy trên một thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ
* Việc truyền message là luôn không quan tâm đến nội dung truyền

MQTT được thiết kế nhằm phục vụ truyền thông machine – to – machine nhưng thực tế chứng minh nó lại linh hoạt hơn mong đợi. Nó hoàn toàn có thể áp dụng cho các kịch bản truyền thông khác nhau như: machine - to – cloud, cloud – to – machine, app – to – app. Chỉ cần có một broker phù hợp và MQTT client được cài đặt đúng cách, các thiết bị xây dựng trên nhiều nền tảng khác nhau có thể giao tiếp với nhau một cách dễ dàng.

## Cơ sở dữ liệu

### So sánh cơ sở dữ liệu SQL và NoSQL

Có khá nhiều cơ sở có thể sử dụng như là SQL server, MongoDB, … trong đó cơ sở dữ liệu NoSQL cụ thể là MongoDB sẽ được sử dụng trong hệ thống bởi nó có ưu điểm như là dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ. Đặc biệt dữ liệu sẽ được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao.

Sự so sánh về đặc điểm của hai cơ sở dữ liệu SQL và NoSQL được thể hiện rõ trong bảng 1-2 [3]. Từ đó, chúng ta có thể thấy được ưu điểm của cơ sở dữ liệu NoSQL.

Bảng 1.2 So sánh cơ sở dữ liệu SQL và NoSQL

|  |  |  |
| --- | --- | --- |
| **Tham số** | **SQL** | **NoSQL** |
| Định nghĩa | Cơ sở dữ liệu SQL chủ yếu được gọi là RDBMS hoặc Cơ sở dữ liệu quan hệ | Cơ sở dữ liệu NoSQL chủ yếu được gọi là cơ sở dữ liệu không liên quan hoặc phân tán |
| Ngôn ngữ Query | Structured query language (SQL) | Không có ngôn ngữ query |
| Loại hình | SQL databases là cơ sở dữ liệu dựa trên bảng | NoSQL databases có thể dựa trên tài liệu, cặp khóa-giá trị, cơ sở dữ liệu biểu đồ |
| Lược đồ | SQL databases có lược đồ được xác định trước | NoSQL databases sử dụng lược đồ động cho dữ liệu phi cấu trúc. |
| Khả năng mở rộng | SQL databases có thể mở rộng theo chiều dọc | NoSQL databases có thể mở rộng theo chiều ngang |
| Lưu trữ dữ liệu phân cấp | SQL databases không thích hợp cho việc lưu trữ dữ liệu phân cấp. | Phù hợp hơn cho kho lưu trữ dữ liệu phân cấp |
| Biến thể | Một loại có biến thể nhỏ | Nhiều loại khác nhau bao gồm các kho khóa-giá trị, cơ sở dữ liệu tài liệu và cơ sở dữ liệu đồ thị. |
| Loại lưu trữ | Highly Available Storage (SAN, RAID, etc.) | Commodity drives storage (standard HDDs, JBOD) |
| Tính năng nổi bật | Hỗ trợ đa nền tảng, Bảo mật và miễn phí | Dễ sử dụng, hiệu suất cao và công cụ linh hoạt. |

### Tổng quan về MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh [4].

Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng.

So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row, MongoDB sẽ dùng các document thay cho row trong RDBMS.

Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.

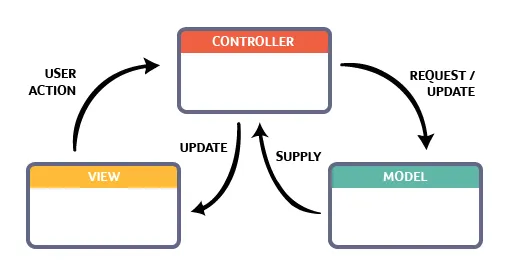
Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB.

## Tổng quan về mô hình Model-View-Controller (MVC)

### Khái niệm

MVC là viết tắt của cụm từ “Model-View-Controller”. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau được thể hiện trong hình 1-2. Theo hình 1-2, mô hình MVC bao gồm Model (dữ liệu), View (giao diện) và Controller (bộ điều khiển) [5]. Cụ thể về chức năng của từng phần như sau:

* Model: quản lý và xử lý cơ sở dữ liệu (database), logic và quy tắc (rule) cho hệ thống.
* View: View là phương tiện hiển thị các đối tượng trong một ứng dụng. Chẳng hạn như hiển thị một cửa sổ, nút hay văn bản trong một cửa sổ khác. Nó bao gồm bất cứ thứ gì mà người dùng có thể nhìn thấy được.
* Controller: là thành phần điều hướng, là chất kết dính giữa model và view, có nhiệm vụ nhận những request từ người dùng, tương tác với model để lấy thông tin và gửi cho view để hiển thị lại cho người dùng.



Hình 1.2 Mô hình MVC

Ở đây trong mô hình MVC, View sẽ không giao tiếp trực tiếp với Model mà sự tương tác giữa View và Model sẽ chỉ được xử lý bởi Controller.

### Đặc điểm của mô hình MVC

* Tiết kiệm băng thông vì không sử dụng viewstate.
* Kiểm tra lỗi đơn giản và dễ dàng.
* Các phần Model, Controller và View được tách biệt với nhau.
* Tối ưu trên các nền tảng khác nhau.
* Dễ dàng duy trì ứng dụng vì chúng được tách biệt với nhau.

## Giới thiệu về công cụ và kỹ thuật được sử dụng

### Ngôn ngữ C#

C# là một ngôn ngữ lập trình hướng đối tượng được phát triển bởi Tập đoàn Microsoft. Ngôn ngữ này được xem là ngôn ngữ lập trfinh hướng đối tượng trong sáng và thuần nhất. Nó hiện thực hầu hết các tính chất tốt của mô hình hướng đối tượng giống như ngôn ngữ lập trình Java. C# là ngôn ngữ được Microsoft phát triển dựa trên 2 ngôn ngữ huyền thoại đó là C++ và Java. Và nó cũng được miêu tả là loại ngôn ngữ có được sự cân bằng giữa C++, Visual Basic, Delphi và Java.

C# làm việc chủ yếu trên bộ khung .NET (.NET framework). Ngôn ngữ lập trình này có khả năng tạo ra nhiều ứng dụng mạnh mẽ và an toàn cho nền tảng Windows. Các thành phần máy chủ, dịch vụ web, ứng dụng di động và nhiều khả năng khác nữa.

C# là ngôn ngữ gần gũi với các ngôn ngữ lập trình thông dụng như C++, Java hay Pascal. Bởi nó được xây dưng trên nền tảng của những ngôn ngữ này. Chính bởi vậy mà nó sẽ được kế thừa những tinh hoa của những ngôn ngữ thông dụng trên. Khi C# ra đời nó đã giải quyết được các khiếm khuyết của C/C++ như con trỏ, các hiệu ứng phụ, … Vì vậy nó dễ tiếp cận và dễ được phát triển hơn.  Đây cũng là một lợi thế cho những ai đã học và quen thuộc với C++ hay Java khi mà học ngôn ngữ C#.

C # được hỗ trợ bởi một cộng đồng các nhà phát triển lớn. Chính cộng đồng sử dụng này đã khiến cho C# trở thành ngôn ngữ đáng tin cậy và hoạt động hiệu quả hơn.

C# có thể được sử dụng để tạo ra hầu hết mọi ứng dụng, Nhưng nó mạnh mẽ nhất là trong việc xây dựng ứng dụng và game trên desktop của Windows. C# cũng ngày càng trở nên phổ biến cho phát triển điện thoại di động và các ứng dụng web. Một số ví dụ về phát triển Game sử dụng bộ khung C# và [.NET](https://vi.wikipedia.org/wiki/.NET_Framework) là Unity, Mono Game và CryEngine.

### Visual Studio

#### Khái niệm Visual Studio

Visual Studio là một trong những công cụ hỗ trợ lập trình và thiết kế website rất nổi tiếng hiện nay của Microsoft và chưa có một phần mềm nào có thể thay thế được. Visual Studio được viết bằng 2 ngôn ngữ chính là C# và VB+. Đây là hai ngôn ngữ lập trình giúp người dùng có thể lập trình được hệ thống một các dễ dàng và nhanh chóng.

Visual Studio là một phần mềm lập trình hệ thống được sản xuất trực tiếp từ Microsoft. Từ khi ra đời đến nay, Visual Studio đã có rất nhiều các phiên bản sử dụng khác nhau. Điều đó, giúp cho người dùng có thể lựa chọn được phiên bản tương thích với dòng máy của mình cũng như cấu hình sử dụng phù hợp nhất [6].

#### Một số tính năng mềm Visual Studio

* Biên tập mã
* Giống như bất kỳ IDE khác, Visual Studio gồm có một trình soạn thạo mã hỗ trợ tô sáng cú pháp và hoàn thiện mã bằng cách sử dụng IntelliSense không chỉ cho các biến, hàm và các phương pháp mà còn các cấu trúc ngôn ngữ như vòng điều khiển hoặc truy vấn.
* Hỗ trợ cải tiến mã nguồn bao gồm tham số sắp xếp lại, biến và phương pháp đổi tên, khai thác và đóng gói giao diện các lớp thành viên bên trong những trạng thái giữa những thứ khác.
* Tính năng biên dịch nền (biên dịch gia tăng) Visual Studio biên dịch nó trong nền để cung cấp thông tin phản hồi về cú pháp và biên dịch lỗi, được đánh dấu bằng một gạch dưới gợn sóng màu đỏ.
* Trình gỡ lỗi
* Trình gỡ lỗi hoạt động vừa là một trình gỡ lỗi cấp mã nguồn và là một trình gỡ lỗi cấp máy. Hoạt động với cả hai mã quản lý cũng như ngôn ngữ được hỗ trợ bởi Visual Studio. Ngoài ra, cũng có thể đính kèm theo quy trình hoạt động và theo dõi và theo dõi những quy trình. Các chương trình đa luồng cao cấp cũng được hỗ trợ.
* Trình gỡ lỗi cho phép thiết lập các breakpoint và watch (giúp giám sát các giá trị của biến). Breakpoint có thể có điều kiện, nghĩa là chúng được kích hoạt khi điều kiện được đáp ứng. Khi gỡ lỗi, nếu con trỏ chuột di chuyển lên bất kì biến, giá trị hiện tại của nó được hiển thị trong phần chú giải. Trong quá trình viết mã, các trình gỡ lỗi của visual Studio cho phép một số chứng năng được gọi ra bằng tay từ cửa sổ cung cụ Immediate.

#### Tính năng thiết kế

* WindowsForm Designer: được sử dụng với mục đích xây dựng GUI sử dụng WindowsForms, được bố trí dùng để xây dựng các nút điều khiển bên trong hoặc cũng có thể khóa chúng vào bên cạnh mẫu. Điều khiển trình bày dữ liệu có thể được liên kết với các nguồn dữ liệu như: cơ sở dữ liệu hoặc truy vấn.
* WPF Desidner: tính năng này giống như WindowsForms Designer có công dụng hỗ trợ kéo và thả. Sử dụng tương tác giữa người và máy tính nhằm múc tiêu theo Windows Presentation Foundation. Hỗ trợ các chức năng WPF bao gồm kết nối dữ liệu và tự động hóa bố trí quản lý. Tao ra mã XAML cho giao diện người dùng. Các tập tin XAML được tạo ra là tương thích với Microsoft Expression Degin, sản phẩm thiết kế theo định hướng. Các mã XAML được liên kết với mã đang sử dụng một mô hình code – behind.
* Web designer/ development: Visual Studio cũng có một trình soạn thảo và thiết kế website cho phpes các trang web được thiết kế theo tính năng kéo và thả đối tượng
* Xamarin: công cụ được thêm vào để xây dựng các ứng dụng di dộng đa nền tảng.

#### Các công cụ khác

* Open Tabs Browser: được sử dụng để liệt kê tất cả thẻ đang mở và chuyển đổi giữa chúng.
* Properties Editor: được sử dụng để chỉnh sửa các thuộc tính trong một cửa sổ giao diện bên trong Visual Studio.
* Object Browser: là một không gian tên và trình duyệt lớp thư viện cho Microsoft NET. Có thể được sử dụng để duyệt các không gian tên (được sắp xếp theo thứ bậc) trong Assembly (CLI).
* Solution Explorer: theo cách nói trong Visual Studio, là một tập hợp các tập tin mã và các nguồn khác được sử dụng để xây dựng một ứng dụng
* Team Explorer: được sử dụng để tích hợp các khả năng của Team Foundation Server, Revision Control System và là cơ sở cho môi trường CodePlex đối với dự án mã nguồn mở.
* Visual Studio Tools for Office: là một SDK và một add – in cho Visual Studio bao gồm các công cụ để phát triển cho các bộ Microsoft Office

### Xamarin

Xamarin được thành lập vào tháng 5 năm 2011 bởi các kỹ sư đã tạo ra Mono, Mono cho Android và MonoTouch, là sự triển khai nhiều nền tảng của Common Language Infrastructure (CLI) và Common Language Specifications (Thường được gọi là Microsoft .NET). Với cơ sở mã nguồn mở của C#, các nhà phát triển có thể sử dụng các công cụ Xamarin để viết các ứng dụng Android, iOS và Windows với giao diện người dùng và chia sẻ code trên nhiều nền tảng, bao gồm Windows và macOS [7].

Xamarin có thể ứng dụng phát triển cho đa nền tảng khác nhau. Điều này có được là sự kết hợp với ngôn ngữ là tringh C# và nền tảng framework. Đặc biệt công nghệ xamarin rất thân thiện với các hệ điều hành. Nó có thể chạy được trên cả 2 hệ điều hành phổ biến nhất trên thể giới đó là Mac và windown. Mộ điều đặc biệt hơn nữa là các lập trình viên có thể tái sử dụng 90% code. Để có thể tái sử dụng nhằm gia tăng thêm engineering cycle cho các nền tảng làm lần sau.

Nếu như tiêu chuẩn của lập trình là Native code với các số liệu cực kì tốt. Thì công nghệ xamarin cũng làm được điều đó, minh chứng là các số liệu performances cũng tương đương với các ứng dụng được tạo bởi Native. Cùng vời đó là performances của công nghệ xamarin cũng liên tục được cập nhật. Nhắm nâng cao chất lượng và cũng đáp úng được các tiêu chuẩn mới của native code.

Với cùng một ngôn ngữ lập trình là C#, chung API và cấu trúc dữ liệu. Cho nên tất cả các ứng dụng được thiết kế bằng công nghệ xamarin đều có thể chia sẻ ở mọi nền tảng. Một con số đáng kinh ngạc này lên tới tận 75% code có thể chia sẻ được. Cũng chính vì lí do này mà tiết kiệm rất nhiều thời gian và chi phí cho việc thiết kế ra 1 app. Có thể dùng công nghệ xamarin để thiết kế app cho cả IOS và Android. Đây là một trong những ưu điểm nổi trội của công nghệ này mang lại cho các coder.

Như đã nói ở trên, xamarin được tạo ra bởi một trong những công ty công nghệ lớn nhất thế giới. Đó chính là microsoft, cho nên chất lượng mà công nghệ này không thể chê được. Và đây cũng là một trong những công nghệ thiết kế ứng dụng di động được biết tới nhiểu nhất. Các lập trình viên hoàn toàn có thể tự thiết kế riêng trải nghiệm người dùng cho từng nền tảng. Điều này cũng đem tới cho người dùng app trải nghiệm sử dụng tốt nhất. Với công cụ xamarin form, bạn hoàn toàn có thể tạo ra được những ứng dụng đơn giản trên các nền tảng ios, android, win. Mà không đòi hỏi bạn có quá nhiều các kiến thức chuyên nghành quá sâu.

### Windows Presentation Foundation (WPF)

Windows Presentation Foundation (viết tắt là WPF) do Microsoft phát triển, là công nghệ kế tiếp Windows Form dùng để xây dựng các ứng dụng dành cho máy trạm chạy hệ điều hành Windows. WPF được giới thiệu từ năm 2006 trong.NET Framework 3.0 (dưới tên gọi Avalon), công nghệ này nhận được sự quan tâm của cộng đồng lập trình viên bởi nhiều điểm đổi mới trong lập trình ứng dụng và khả năng xây dựng giao diện thân thiện, sinh động.

WPF hỗ trợ tốt hơn Winform trong việc xử lý hình ảnh, âm thanh, video, quản lý phông chữ, quản lý hiển thị và chỉnh sửa văn bản. Các control trong WPF có thể được lồng ghép linh động để tạo ra giao diện do được viết bằng XAML. Một ứng dụng WPF có thể được xây dựng để chạy độc lập dưới dạng mở rộng EXE hoặc đóng gói với phần mở rộng là XBAP để có thể tích hợp lên website.

## Kết luận

Như vậy, chương một đã trình bày về lý do lựa chọn đề tài, mô hình tổng quan của hệ thống theo dõi bệnh nhân từ xa, thể hiện qua sơ đồ và mô tả rõ hoạt động từng thành phần của hệ thống, sau đó giới thiệu một số giải pháp và công cụ được sử dụng trong đề tài này. Kết thúc chương 1 “Tổng quan hệ thống theo dõi bệnh nhân từ xa”, chúng ta đã có thể hiểu được sơ bộ về đề tài, và tiếp theo đây chương 2 sẽ đi vào phân tích thiết kế chi tiết cho hệ thống.

# PHÂN TÍCH THIẾT KẾ HỆ THỐNG

Sau khi đã xác định được mục đích xây dựng ứng dụng đã nêu ở chương 1, chúng ta tiếp tục phân tích các chức năng cần thiết của ứng dụng. Từ đó tạo ra sơ đồ Use Case, các quan hệ trong lược đồ và mô tả lại từng yêu cầu hệ thống theo một kịch bản. Sau đó tìm hiểu thiết kế biểu đồ lớp (Class Diagram) cho Models chung và cuối cùng là thiết kế luồng đi của hệ thống, luồng đi của dữ liệu thông qua biểu đồ hoạt động (Activity Diagram) và biểu đồ tuần tự (Sequence Diagram).

## Yêu cầu hệ thống

### Yêu cầu chức năng

Chức năng cơ bản của hệ thống:

* Về phía bác sĩ có thể thực hiện chức năng: tìm kiếm bệnh nhân, quản lý bệnh nhân, xem các chỉ số của bệnh nhân, nhận cảnh báo nguy hiểm về bệnh nhân.
* Về phía bệnh nhân: xem các chỉ số của bản thân, nhận cảnh báo nguy hiểm.
* Về phía trung tâm y tế (bệnh viện): tìm kiếm bệnh nhân, quản lý bệnh nhân, quản lý bác sĩ, xem các chỉ số của bệnh nhân, nhận cảnh báo nguy hiểm về bệnh nhân.

### Yêu cầu phi chức năng

* Các ứng dụng chạy ổn định
* Giao diện dễ nhìn, phông chữ thống nhất
* Dễ sử dụng
* Ít lỗi
* Dễ sửa chữa khi có lỗi

## Xây dựng Use case Diagram

### Tìm hiểu biểu đồ Use case

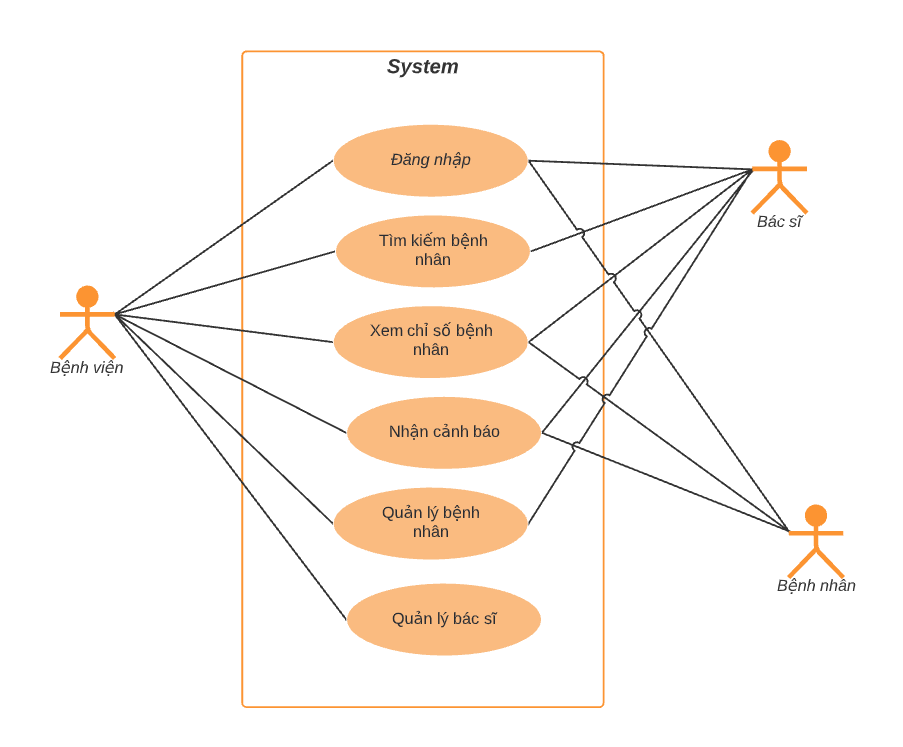
Biểu đồ Use Case biểu diễn sơ đồ chức năng của hệ thống. Từ tập yêu cầu của hệ thống, biểu đồ use case chỉ ra những điều hệ thống cần thực hiện để thỏa mãn các yêu cầu của người dùng hệ thống đó 🡪 Chỉ ra sự tương tác giữa tác nhân và hệ thống thông qua các Use Case. Các thành phần cơ bản của một biểu đồ Use case được thể hiện trong bảng 2-1.

Bảng 2.1 Các phần tử của biểu đồ Use Case

|  |  |  |  |
| --- | --- | --- | --- |
| **Phần tử mô hình** | **Ý nghĩa** | **Cách biểu diễn** | **Ký hiệu trong biểu đồ** |
| Use case | Biểu diễn một chức năng xác định của hệ thống | Hình Elip chứa tên của Use Case |  |
| Tác nhân | Là một đối tượng bên ngoài hệ thống tương tác trực tiếp với các use case. | Biểu diễn bởi một lớp kiểu Actor. | A picture containing chart  Description automatically generated |
| Mối quan hệ giữa các use case | Include: khi thực thi Use Case A sẽ luôn cần thực hiện Use Case B. | Extend và include có dạng các mũi tên đứt nét. | Text  Description automatically generated with low confidence  Text  Description automatically generated with low confidence |
| Extend: khi thực thi Use Case A, trong một số trường hợp sẽ thực hiện Use Case B. |
| Comunication Link: Kết nối giữa actor và use case, cho biết actor đó có những tương tác nào đến hệ thống | Comunication Link có dạng đường thẳng |
| Generalization: user case này được thiết kế các chức năng từ use case kia. | Generalization có dạng mũi tên tam giác. |
| Biên của hệ thống | Tách biệt phần bên trong và bên ngoài hệ thống. | Được biểu diễn bởi một hình chữ nhật rỗng. |  |

### Bản vẽ Use case

#### Use case tổng quát



Hình 2.1 Sơ đồ Use Case tổng quát của hệ thống

Mô tả:

Hệ thống gồm ba tác nhân chính: Bệnh viện, bác sĩ và bệnh nhân.

Bệnh viện là nơi quản lý bác sĩ và bệnh nhân, có quyền sử dụng toàn bộ chức năng của hệ thống.

Bác sĩ là người có nhiệm vụ theo dõi sức khỏe của bệnh nhân, có thể thao tác với bốn chức năng: tìm kiếm bệnh nhân mà bác sĩ điều trị, xem chỉ số của bệnh nhận đó, nhận về cảnh báo nếu các chỉ số bất thường, có thể quản lý bệnh nhân mà mình điều trị.

Bệnh nhân là người được bác sĩ theo dõi sức khỏe, có thể xem các chỉ số của bản thân và nhận cảnh báo nếu có bất thường với các chỉ số đó.

* Use case cho Actor “Bệnh viện”

Diagram

Description automatically generated

Hình 2.2 Sơ đồ use case cho trung tâm Y tế

Mô tả sơ đồ:

Trung tâm Y tế hay bệnh viện là tác nhân có thể sử dụng được toàn bộ chức năng của hệ thống, quản lý bác sĩ, bệnh nhân và có thể xem và nhận cảnh báo về các bất thường trong chỉ số của các bệnh nhân đang được điều trị tại bệnh viện.

* Use case cho Actor “Bác sĩ”

Diagram

Description automatically generated

Hình 2.3 Sơ đồ use case cho bác sĩ

* Use case cho Actor “Bệnh nhân”

Diagram

Description automatically generated

Hình 2.4 Sơ đồ use case cho bệnh nhân

#### Usecase quản lý bệnh nhân

Diagram

Description automatically generated

Hình 2.5 Usecase quản lý bệnh nhân

Mô tả: Chức năng quản lý bệnh nhân được sử dụng bởi hai actor trung tâm Y tế (bệnh viện) có thể quản lý toàn bộ danh sách bệnh nhân thuộc bệnh viện, hoặc bác sĩ có thể quản lý danh sách bệnh nhân mà mình theo dõi điều trị.

#### Usecase quản lý bác sĩ

Diagram

Description automatically generated

Hình 2.6 Usecase quản lý bác sĩ

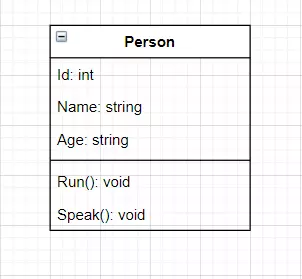
Mô tả: Chức năng quản lý bác sĩ bao gồm xem danh sách bác sĩ và có thể chính sửa danh sách đó

## Xây dựng Class Diagram

### Tìm hiểu biểu đồ lớp (Class Diagram)

Class diagram là một kỹ thuật mô hình hóa tồn tại ở tất cả các phương pháp phát triển hướng đối tượng mô tả kiểu của các đối tượng trong hệ thống và các loại quan hệ khác nhau tồn tại giữa chúng. Class diagram là biểu đồ hay dùng nhất trong UML và gần gũi nhất với các lập trình viên, giúp các lập trình viên trao đổi với nhau và hiểu rõ ý tưởng của nhau [8].

Các tính chất cơ bản của class diagram Ví dụ trong hình 2.7:



Hình 2.7 Ví dụ về các tính chất cơ bản của class

* Tên class: Person.
* Attribute (field, property): Id, Name, Age.
* Operation (method, function): Run(), Speak().

Access Modifier trong class diagram: Sử dụng để đặc tả phạm vi truy cập cho các Attribute và Operation của 1 class. Có 4 lựa chọn phạm vi truy cập được thể hiện trong hình 2.8, chi tiết như sau:

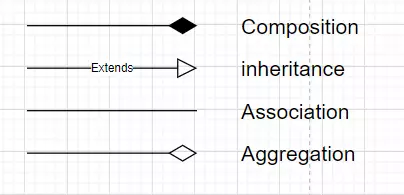
Table

Description automatically generated

Hình 2.8 Ví dụ về các phạm vi truy cập của class

* Private (-): Chỉ mình các đối tượng được tạo từ class này có thể sử dụng.
* Public (+): Mọi đối tượng đều có thể sử dụng.
* Protected (#): Chỉ các đối tượng được tạo từ class này và class kế thừa từ class này có thể sử dụng.
* Package/Default: Các đối tượng được tạo từ class trong lớp cùng gói có thể sử dụng.

Relationship trong class diagram: Sử dụng để thể hiện mỗi quan hệ giữa đối tượng được tạo từ 1 class với các đối tượng được tạo từ class khác trong class diagram. Có 4 loại relationship được thể hiện trong hình 2.9. Trong đó:



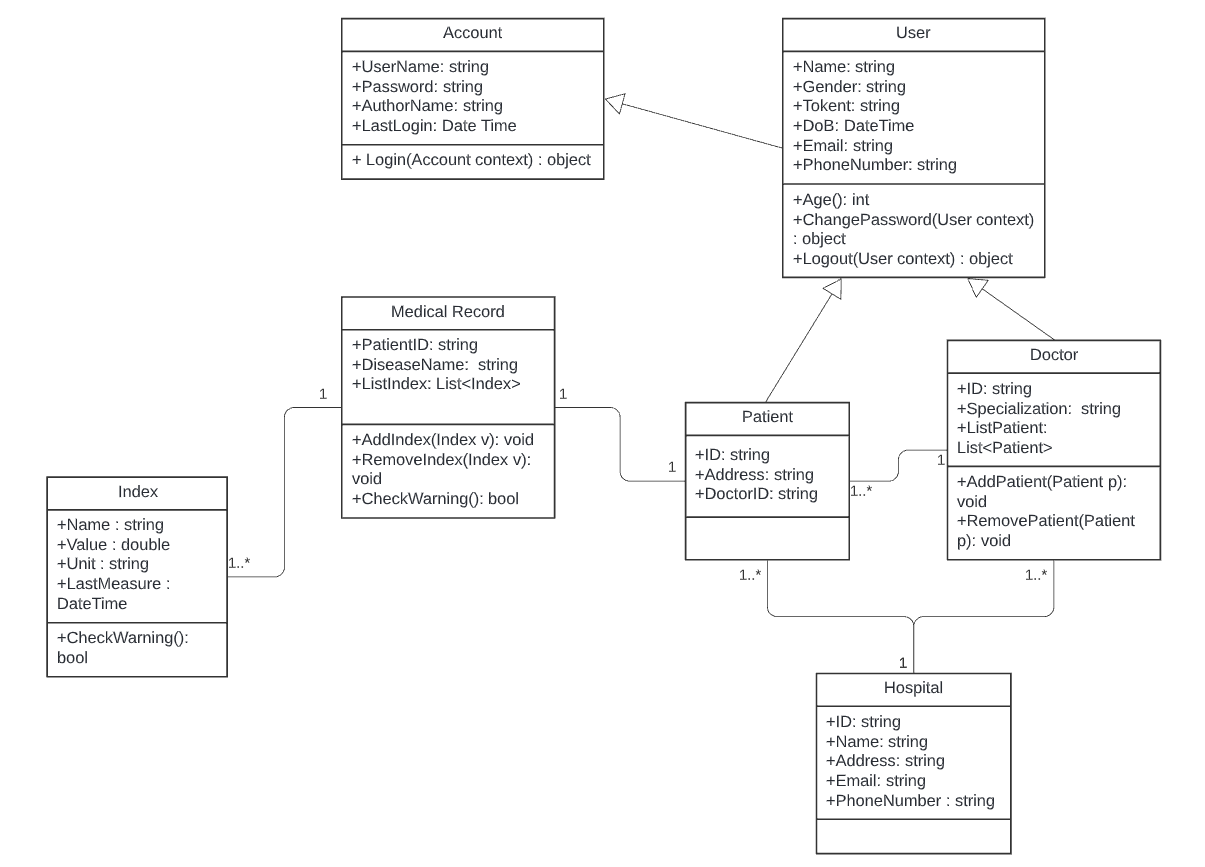
Hình 2.9 Các loại quan hệ trong class diagram

* Inheritance: 1 class kế thừa từ 1 class khác.
* Association: 2 class có liên hệ với nhau nhưng không chỉ rõ mối liên hệ.
* Composition: Đối tượng tạo từ class A mất thì đối tượng tạo từ class B sẽ mất.
* Agreegation: Đối tượng tạo từ class A mất thì đối tượng tạo từ class B vẫn tồn tại độc lập.

Multiplicity trong class diagram: Sử dụng để thể hiện quan hệ về số lượng giữa các đối tượng được tạo từ các class trong class diagram. Các kiểu multiplicity:

* 0...1: 0 hoặc 1.
* n: Bắt buộc có n.
* 0...\*: 0 hoặc nhiều.
* 1...\*: 1 hoặc nhiều.
* m... n: có tối thiểu là m và tối đa là n.

### Bản vẽ Class Diagram



Hình 2.10 Bản vẽ class diagram cho Model

## Xây dựng Activity Diagram

### Tìm hiểu biểu đồ hoạt động (Activity Diagram)

Activity diagram (biểu đồ hoạt động) là một mô hình logic được dùng để mô hình hoá cho các hoạt động trong một quy trình nghiệp vụ. Nó chỉ ra luồng đi từ hoạt động này sang hoạt động khác trong một hệ thống. Nó đặc biệt quan trọng trong việc xây dựng mô hình chức năng của hệ thống và nhấn mạnh tới việc chuyển đổi quyền kiểm soát giữa các đối tượng [9].

Để xây dựng được 1 bản vẽ activity diagram thì cần có những bước cơ bản như sau:

Bước 1: Xác định nghiệp vụ cần mô tả

Dựa vào các bản vẽ use case để xác định nghiệp vụ nào cần mô tả.

Bước 2: Xác định trạng thái đầu tiên và trạng thái kết thúc

Bước 3: Xác định các hoạt động tiếp theo

Xuất phát từ điểm bắt đầu, phân tích để xác định các hoạt động tiếp theo cho đến khi gặp điểm kết thúc để hoàn tất bản vẽ này.

Các thành phần của activity diagram

1. Start

* Ký hiệu:
* Đặc trưng:
  + - Khởi tạo một hoạt động
    - Một activity diagram có thể có nhiều trạng thái start
* Nên đặt tên là động từ. Và mô tả đủ ý nghĩa tổng thể của hoạt động có thể

1. Transition

* Ký hiệu:
* Đặc trưng: Mô tả sự chuyển đổi trạng thái của các hoạt động.

1. Decisition

* Ký hiệu:
* Đặc trưng:
  + Tập các điều kiện kích hoạt việc chuyển trạng thái
  + Branch
    - Mô tả điều kiện rẽ nhánh
    - Chỉ một dòng điều khiển đi vào
    - Hai hoặc nhiều dòng điều khiển ra
    - Chỉ một dòng điều khiển ra dẫn đến kết quả
    - Mỗi dòng chứa một điều kiện (guard), guard phải liên quan đến điều kiện và loại trừ nhau
  + Merge
    - Có hai hoặc nhiều dòng điều khiển đi vào
    - Chỉ một dòng điều khiển đi ra
* Có thể hiểu đây là ký hiệu biểu thị nút điều kiện chuyển hướng. Tùy theo trường hợp đúng sai của kết quả biểu thức logic bên trong ký hiệu mà có hướng di chuyển tiếp theo tương ứng.

1. Synchronization bar

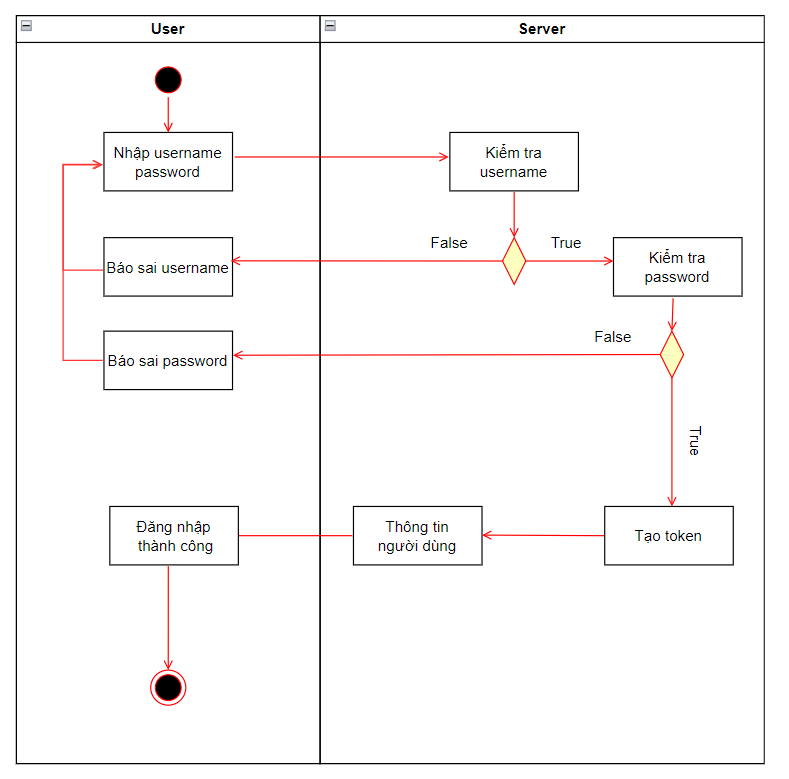
* Ký hiệu:
* Đặc trưng:
  + Mô tả dòng điều khiển thực hiện song song
  + Fork
    - Mô tả một dòng điều khiển được tách ra thực hiện song song
    - Chỉ một dòng điều khiển đi vào
    - Có hai hoặc nhiều dòng điều khiển đi ra
    - Dùng fork khi các hoạt động thực hiện không quan tâm thứ tự
  + Join
    - Kết hợp các dòng điều khiển song song
    - Có hai hoặc nhiều dòng điều khiển vào
    - Chỉ một dòng điều khiển ra
    - Dòng điều khiển ra được tạo khi tất cả các dòng cần thiết đã vào
* Chú ý: Fork và Join không cần nhãn

1. End

* Ký hiệu:
* Đặc trưng:
  + Mô tả trạng thái kết thúc quy trình
  + Một activity diagram có một hoặc nhiều trạng thái kết thúc
* Là điểm kết thúc luồng.

### Bản vẽ Activity Diagram

#### Biểu đồ hoạt động chức năng đăng nhập



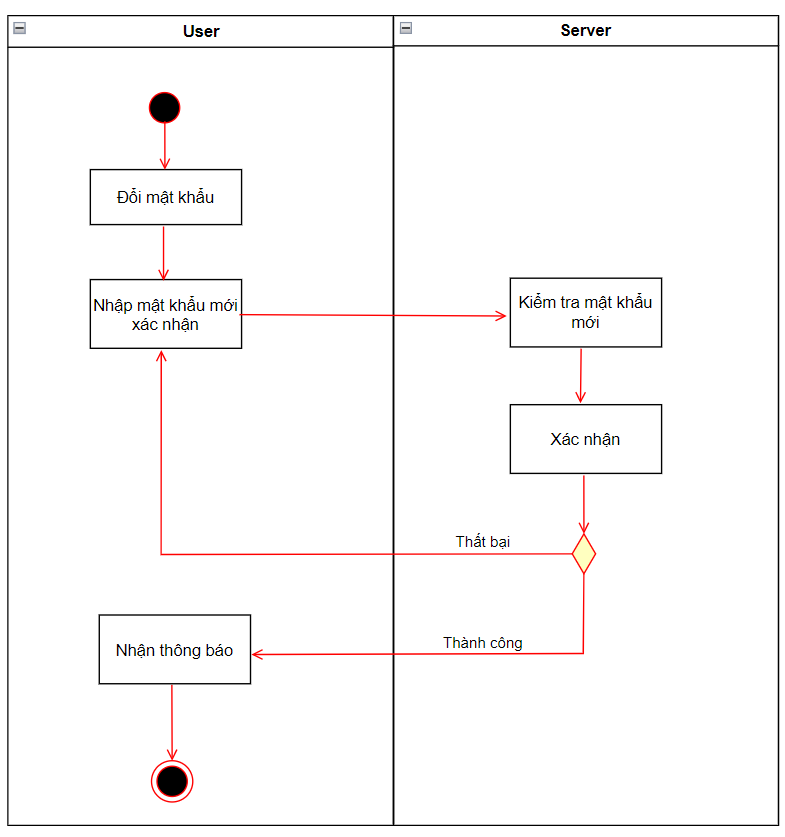
Hình 2.11 Biểu đồ hoạt động chức năng đăng nhập

Mô tả biểu đồ:

Để sử dụng hệ thống người dùng cần đăng nhập tài khoản mà đã được admin tạo sẵn. Hệ thống phải được kết nối internet. Khi có kết nối người dùng sẽ vào phần đăng nhập sau đó nhập 2 thành phần UserName và Password.

Khi Nhập UserName và Password, phía hệ thống sẽ kiếm tra có tài khoản này không, nếu không tồn tại sẽ có một thông báo “Wrong Username”, nếu có hệ thông sẽ tiếp tục kiểm tra mật khẩu, nếu không đúng sẽ thông báo “Wrong Password” và yêu cầu nhập lại, khi mật khẩu đúng thì người dùng đăng nhập thành công.

#### Biểu đồ hoạt động chức năng đổi mật khẩu



Hình 2.12 Biểu đồ hoạt động chức năng đổi mật khẩu

Mô tả biểu đồ:

Khi có nhu cầu sử dụng một mật khẩu mới thay vì mật khẩu do admin cung cấp, user được phép đổi mật khẩu theo mong muốn. User vào setting chọn phần đổi mật khẩu, điền password mới rồi ấn gửi và sẽ nhận được thông báo đổi mật khẩu thành công. Việc đổi mật khẩu sẽ giúp người dùng dễ nhớ và dễ sử dụng ứng dụng hơn.

#### Biểu đồ hoạt động chức năng xem chỉ số bệnh nhân

Diagram

Description automatically generated

Hình 2.13 Biểu đồ hoạt động chức năng xem chỉ số bệnh nhân

Mô tả biểu đồ:

Để sử dụng chức năng xem chỉ số, người dùng phải được kết nối với hệ thống thông qua chức năng đăng nhập. Hệ thống phải được duy trì kết nối internet. Khi đó người dùng sẽ vào phần theo dõi chỉ số trong Client để tiến hành xem chỉ số.

Khi vào phần theo dõi chỉ số, phía hệ thống sẽ truy cập vào cơ sở dữ liệu nơi mà các chỉ số được cập nhật liên tục bởi các thiết bị đo. Sau đó, hệ thống sẽ lấy các dữ liệu về chỉ số ra và gửi lại cho Client của người dùng.

#### Biểu đồ hoạt động chức năng cảnh báo

Diagram

Description automatically generated

Hình 2.14 Biểu đồ hoạt động chức năng cảnh báo

Mô tả biểu đồ:

Để sử dụng chức năng nhận cảnh báo, người dùng phải được kết nối với hệ thống thông qua chức năng đăng nhập. Hệ thống phải được duy trì kết nối internet. Khi đó người dùng sẽ nhận được cảnh báo nếu chỉ số có bất thường.

Thiết bị đo sẽ liên tục cập nhật dữ liệu về các chỉ số vào Database, bản thân thiết bị cũng có thể phát hiện ra các chỉ số bất thường và phát ra cảnh báo, đồng thời gửi một thông báo về bất thường đến cho Server, Server sẽ xử lý và gửi cảnh báo đến cho những User tương ứng.

#### Biểu đồ hoạt động chức năng quản lý bác sĩ

Diagram

Description automatically generated

Hình 2.15 Biểu đồ hoạt động chức năng quản lý bác sĩ

Mô tả biểu đồ:

Trung tâm y tế hoặc bệnh viện có thể sử dụng chức năng quản lý bác sĩ, thêm mới, chỉnh sửa hoặc xóa bác sĩ. Hơn nữa, bệnh viện cũng có thể chỉnh sửa danh sách bệnh nhân mà bác sĩ đang theo dõi điều trị.

Sau khi nhận được yêu cầu từ phía bệnh viện (User), Server sẽ xử lý rồi trả về danh sách bác sĩ sau khi cập nhật thay đổi.

#### Biểu đồ hoạt động chức năng quản lý bệnh nhân

Diagram

Description automatically generated

Hình 2.16 Biểu đồ hoạt động chức năng quản lý bệnh nhân

Mô tả biểu đồ:

Trung tâm y tế hoặc bệnh viện, bác sĩ đều có thể sử dụng chức năng quản lý bệnh nhân, thêm mới, chỉnh sửa hoặc xóa bệnh nhân. Cụ thể, bệnh viện có thể chỉnh sửa danh sách toàn bộ bệnh nhân của bệnh viện còn bác sĩ chỉ có thể chỉnh sửa danh sách bệnh nhân mà mình đang điều trị.

Sau khi nhận được yêu cầu từ phía bệnh viện hoặc bác sĩ (User), Server sẽ xử lý rồi trả về danh sách bệnh nhân sau khi cập nhật thay đổi.

## Xây dựng Sequence Diagram

### Tìm hiểu biểu đồ tuần tự (Sequence Diagram)

Biểu đồ tuần tự là biểu đồ dùng để xác định các trình tự diễn ra sự kiện của một nhóm đối tượng nào đó.

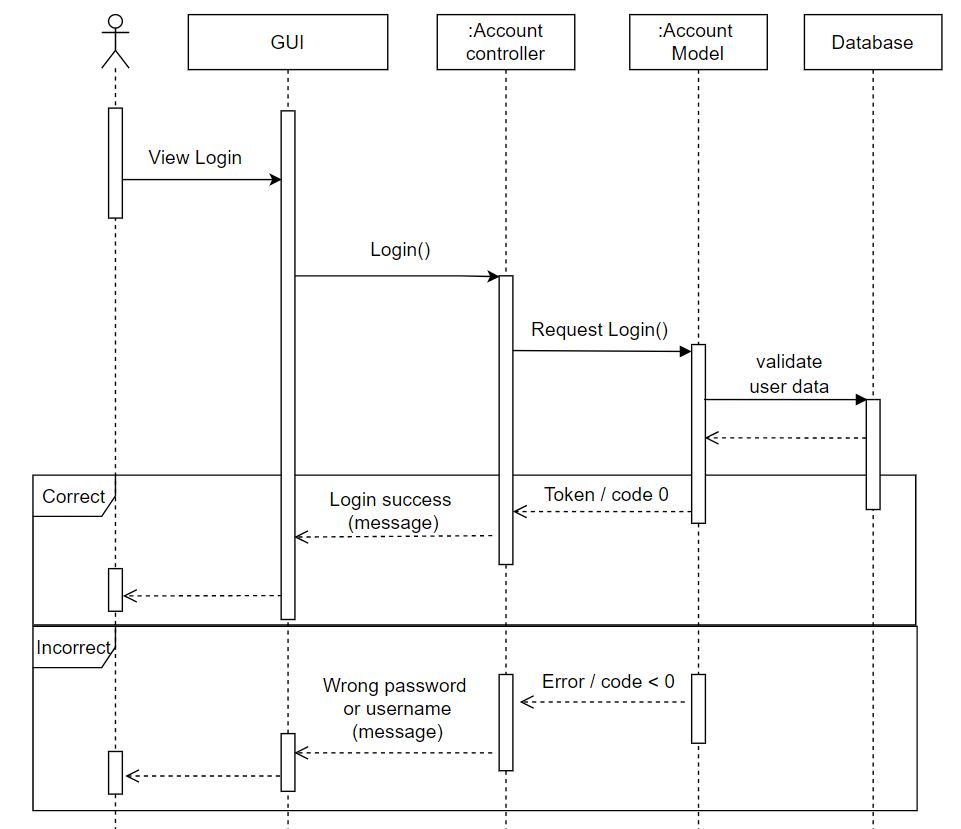
Nó mô tả chi tiết các thông điệp được gửi và nhận giữa các đối tượng đồng thời cũng chú trọng đến việc trình tự về mặt thời gian gửi và nhận các thông điệp đó. Các thành phần cơ bản của một biểu đồ tuần tự được thể hiện trong bảng 2-2.

Bảng 2.2 Các thành phần của biểu đồ tuần tự

|  |  |  |
| --- | --- | --- |
| **Tên thành phần** | **Ý nghĩa** | **Ký hiệu** |
| Đối tượng |  |  |
| Đường đời đối tượng | Biểu diễn bằng các đường gạch đứt thẳng đứng bên dưới đối tượng |  |
| Thông điệp đồng bộ | Thông điệp cần có một Request trước hành động tiếp theo | Text  Description automatically generated with low confidence |
| Thông điệp không đồng bộ | Thông điệp không cần có một Request trước hành động tiếp theo |  |
| Thông điệp chính mình | Thông điệp mà đối tượng gửi cho chính nó để thực hiện các hàm nội tại | Text  Description automatically generated with medium confidence |
| Thông điệp trả lời hoặc trả về | Thông điệp trả lời lại khi có Request hoặc sau khi kiểm tra tính đúng đắn của một điều kiện nào đó | Shape  Description automatically generated with medium confidence |

### Bản vẽ Sequence Diagram

#### Biểu đồ tuần tự chức năng đăng nhập



Hình 2.17 Biểu đồ tuần tự chức năng đăng nhập

Mô tả biểu đồ:

Khi người dùng tương tác với giao diện đăng nhập, nhập thông tin đăng nhập rồi ấn button đăng nhập, giao diện sẽ gọi đến hàm Login() trong AccountController. Controller này sẽ tạo ra một request về phía server để yêu cầu đăng nhập, server sẽ điều phối yêu cầu đó vào hàm Login() trong Account Model, nếu thông tin đăng nhập đúng, server sẽ trả về token và giá trị code 0 trong gói tin response, ngược lại nếu thông tin đăng nhập sai, server sẽ trả về các giá trị code < 0 cho client, từ đây client sẽ đưa ra thông báo cho người dùng nhận biết.

#### Biểu đồ tuần tự chức năng đổi mật khẩu

Diagram

Description automatically generated

Hình 2.18 Biểu đồ tuần tự chức năng đổi mật khẩu

Mô tả biểu đồ: Người dùng chọn giao diện đổi mật khẩu, thực hiện việc nhập mật khẩu mới, xác nhận, View sẽ gọi đến hàm ChangePassword() của SettingController, từ đây sẽ gửi một Request ChangePassword đến Server. Server điều phối cho User Model để thực hiện lưu mật khẩu mới.

#### Biểu đồ tuần tự chức năng xem chỉ số bệnh nhân

Diagram

Description automatically generated

Hình 2.19 Biểu đồ tuần tự chức năng xem chỉ số bệnh nhân

Mô tả biểu đồ: User chọn người dùng, View gọi đến hàm IndexList() của PatientController, sau đó Controller này sẽ gửi request IndexList đến Sever. Server điều phối đến hàm IndexList() trong MedicalRecord Model, sau đó trả về cho client.

#### Biểu đồ tuần tự chức năng cảnh báo

Diagram

Description automatically generated

Hình 2.20 Biểu đồ tuần tự chức năng cảnh báo

Mô tả biểu đồ: Thiết bị liên tục yêu cầu cập nhật chỉ số đến Server, nếu thiết bị phát hiện bất thường sẽ gửi một request Warning() đến cho Server và chính nó cũng phát ra cảnh báo. Server khi nhận được request Warning() sẽ tiến hành tìm các người dùng cần được cảnh báo và gửi đến WarningController phía client, Controller này sẽ đưa ra giao diện thông báo bất thường đến những người dùng cần thiết.

## Xây dựng giao thức liên lạc

Phía client sẽ liên lạc với server thông qua giao thức MQTT, khi client gửi request lên server sẽ gắn theo 4 thông tin chính: url, Value, client-id và token (nếu có). Trong đó:

* Url: có dạng Controller Name / Action Name, có chức năng để server hoặc client nhận biết request hoặc response đó thực hiện chức năng nào.
* Value: là giá trị đi kèm (nếu có) của từng request hoặc response.
* Client-Id: là Id của client request lên server. Khi server response sẽ gắn thêm ClientId vào topic trả về để chỉ client đó có thể nhận được gói tin.
* Token: Sau khi đăng nhập thành công, server sẽ trả về cho client giá trị Token, những request sau đó của client sẽ gắn thêm Token để nhận biết người dùng nào đang request.

Cụ thể, các request có thể được sử dụng ở trong hệ thống theo dõi bệnh nhân được thể hiện trong bảng 2.3 dưới đây:

Bảng 2.3 Danh sách các request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Request | | | |
| #url | Value | #client-id | #token |
| 1 | Account/Login | account | yes | no |
| 2 | User/changePassword | account | yes | yes |
| 3 | User/logout |  | yes | yes |
| 4 | Patient/PatientList |  | yes | yes |
| 5 | Patient/Add | Patient | yes | yes |
| 6 | Patient/Update | Patient | yes | yes |
| 7 | Patient/delete | PatientID | yes | yes |
| 8 | Patient/IndexList | PatientID | yes | yes |
| 9 | Patient/UpdateIndex | PatientID, Index | yes | yes |
| 10 | Doctor/DoctorList |  | yes | yes |
| 11 | Doctor/Add | Doctor | yes | yes |
| 12 | Doctor/Update | Doctor | yes | yes |
| 13 | Doctor/delete | DoctorId | yes | yes |
| 14 | Doctor/PatientList | DoctorId | yes | yes |
| 15 | Doctor/AddPatient | doctorId, Patient | yes | yes |
| 16 | Doctor/DeletePatient | doctorId, Patient | yes | yes |
| 17 | Warning | PatientID | yes | no |

## Kết luận

Như vậy, chương hai đã trình bày về việc phân tích yêu cầu của hệ thống, đưa ra những bản vẽ thiết kế cho toàn hệ thống nói chung và từng chức năng nói riêng, bao gồm các thiết kế UML như: sơ đồ Use case, Class Diagram, Activity Diagram, Sequence Diagram. Cuối cùng, chương 2 đưa ra thiết kế chung cho giao thức liên lạc giữa các bên Client và Server. Kết thúc chương 2 “Phân tích thiết kế hệ thống”, chúng ta đã có thể hiểu được chi tiết hơn về yêu cầu đề tài, làm rõ những chức năng cơ bản của hệ thống, và tiếp theo đây chương 3 sẽ đi vào triển khai mô hình MVC trên công cụ Xamarin, tiến hành thực hiện một số chức năng cơ bản của hệ thống theo dõi bệnh nhân từ xa.

# TRIỂN KHAI MÔ HÌNH MVC TRÊN XAMARIN

Sau khi đã phân tích yêu cầu của hệ thống, đưa ra những bản vẽ thiết kế cho toàn hệ thống nói chung và từng chức năng nói riêng ở chương 2, chúng ta sẽ tiến hành triển khai ứng dụng theo dõi bệnh nhân từ xa bằng mô hình MVC với công cụ Xamarin, Visual Studio. Mở đầu chương chúng ta sẽ tìm hiểu rõ hơn về Xamarin và Xamarin Forms, rồi sau đó tiến hành tạo project và triển khai một số mã nguồn cơ sở hỗ trợ cho việc tạo giao diện, cuối cùng là kết quả của một số chức năng đã thực hiện được.

## Tìm hiểu về Xamarin và Xamarin Forms

### Xamarin

#### Ưu điểm

Các công cụ của Xamarin về cơ bản sẽ cho phép bạn phát triển các ứng dụng Android hoặc iOS bằng ngôn ngữ C# và có thể chia sẻ rất nhiều phần code giữa các ứng dụng với nhau (được mình họa trong hình 3.1).

Khi viết một ứng dụng sử dụng bộ công cụ của hãng Xamarin thì về cơ bản là bạn đang sử dụng một lớp trừu tượng phía trên các SDK thực sự của iOS và Android. Điều này có nghĩa là bạn sẽ thu được kết quả là một ứng dụng native hoàn toàn cùng với giao diện người dùng native trên mỗi nền tảng [10].

Graphical user interface, application, website

Description automatically generated

Hình 3.1 Xamarin chia sẻ code trên nhiều nền tảng

Một số điểm cộng cho Xamarin framework:

* Có khả năng lựa chọn UI layout
* Tích hợp OAuth
* Tích hợp REST APIs từ xa
* Công nghệ dẫn đường và xử lý tín hiệu thời gian thực cho ứng dụng định vị
* Tích hợp mạng xã hội
* Có bộ cơ sở dữ liệu SqLite nhúng
* Thư viện XMPP cho phép xây dựng một loạt các ứng dụng

Ứng dụng Xamarin khá gần với giải pháp tự nhiên. Cảm giác nhìn chung là nền tảng cho phép thực hiện nhiều chức năng để tạo ra ứng dụng một cách rất native. Ngoài ra, Xamarin đang trên đà phát triển nên các thư viện component cũng được bổ sung nhanh chóng, từ UI control, UI theme đến các thư viện. Tất cả đều tiện dụng.

#### Nhược điểm

Tuy nhiên, Xamarin cũng tự tạo ra cho mình những khuyết điểm. AOT compiler. Nó không compile các đoạn code gọn gàng như Xcode, lấy ví dụ trên iOS. code C# sẽ không giống như đã được viết. Hạn chế chủ yếu của Xamarin là bạn không có full quyền điều khiển trên code sẽ chạy trên thiết bị.

Ứng dụng Xamarin lớn hơn, nặng hơn so với ứng dụng native. So sánh với ứng dụng native nó chiếm nhiều hơn vài Mb so với Java/Objective C tương ứng. kích thước của một ứng dụng code bằng xamarin là 3Mb, trong khi code bằng Objective C chỉ chiếm 172 Kb. Càng sử dụng nhiều API, càng nhiều lưu trữ bị chiếm trên thiết bị. Nghĩa là apps thực hiện chậm hơn và yêu cầu nhiều dung lượng hơn trên thiết bị.

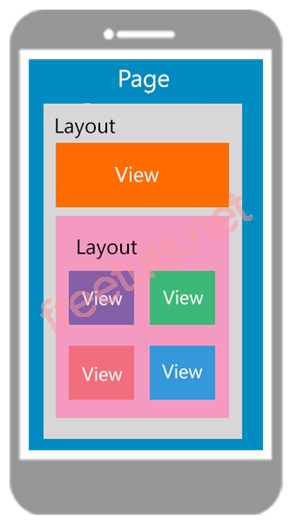
Cộng đồng hỗ trợ không nhiều, cộng đồng của Xamarin ít hơn so với cộng đồng iOS hay Android, vì vậy để tìm kiếm được một developer Xamarin có kinh nghiệm là chuyện không dễ dàng dù Xamarin là nền tảng được phát triển nhờ sự hỗ trợ của Microsoft. Theo nhiều nguồn thông tin, cộng đồng của Xamarin chỉ chiếm 10% cộng đồng lập trình mobile toàn cầu dựa trên thực tế là nền tảng này cung cấp tối da hỗ trợ cho các developer. Cụ thể, có một platform giáo dục chuyên biệt tại ĐH Xamarin, cung cấp kiến thức và khóa huấn luyện cho những người mới tiếp cận công nghệ này. Kết quả là learning curve của một engineer C#/.Net kinh nghiệm sẽ bị tối thiểu hóa.

### Xamarin Forms

Xamarin Forms cho phép các nhà phát triển phần mềm xây dựng các ứng dụng di động đa nền tảng sử dụng các thành phần giao diện cơ bản của hệ điều hành như: Pages, Layouts, Views, Controls. Khi chạy chương trình, mỗi thành phần giao diện của Xamarin Forms sẽ được chuyển hóa sang các thành phần giao diện tương ứng với mỗi nền tảng [11].

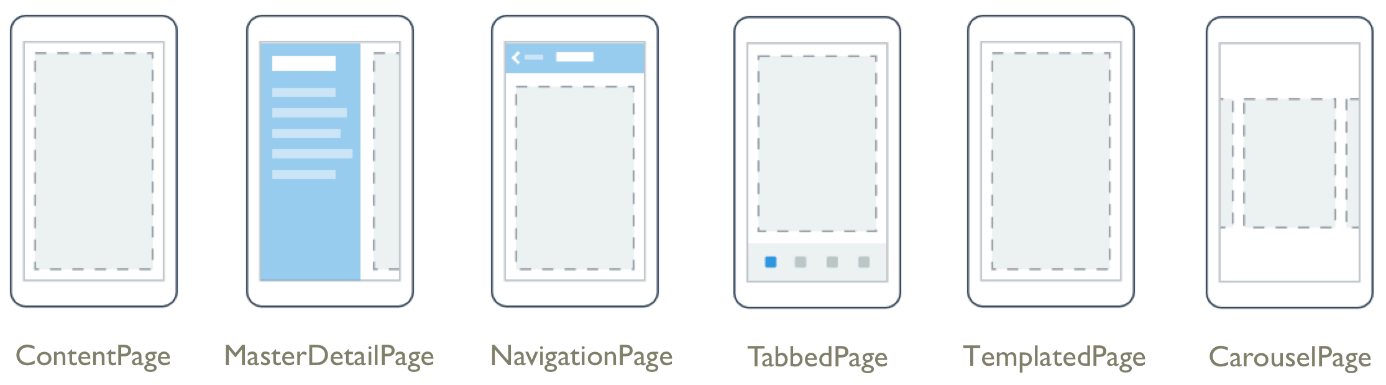
#### Page

Các pages trong Xamarin Forms đại diện cho các màn hình của ứng dụng, Một page chứa các layouts và views giúp hiển thị dữ liệu và nhận tương tác từ người dùng.



Hình 3.2 Một page trong Xamarin Forms

Một Page trong Xamarin Forms đại diện cho một View Controller trong iOS hay một Page trong Windows Phone/UWP hoặc một Activity trong Android. Xamarin có các kiểu Page như trong hình 3.3:



Hình 3.3 Các pages trong Xamarin Forms

Bảng 3.1 mô tả chi tiết về một số page thông dụng trong Xamarin Forms:

Bảng 3.1 Mô tả chi tiết các pages trong Xamarin Forms

|  |  |
| --- | --- |
| **Kiểu Pages** | **Mô tả chi tiết** |
| Content Page | Chứa một view đơn giản |
| MasterDetail Page | Một page mà chứa 2 thành phần. Thông thường, thành phần Master sẽ là menu và thành phần Detail là nội dung |
| Navigation Page | Một page mà chứa thanh điều hướng (Navigation bar). Page được giữ trong bộ nhớ ngăn xếp (stack) và có thể chuyển từ page này sang page khác. Thanh điều hướng có thể có nút điều hướng và tiêu đề của page. |
| Tabbed Page | Một trang mà chưa nhiều Content Page và được điều hướng bằng các nhấn vào các Tab. |
| Carousel Page | Một trang mà chưa nhiều Content Page và được điều hướng bằng cử chỉ vuốt sang ngang. |

#### Views

Xamarin View là một thành phần được hiển thị trên giao diện người dùng như là các nhãn (Labels), nút bấm (Buttons), ô nhập ký tự (Entry). Các thành phần Views cơ bản trong Xamarin Forms như bảng 3.2:

Xamarin Form Views có thể căn chỉnh chiều ngang và chiều dọc bằng cách sử dụng thuộc tính VerticalOptions và HorizontalOptions.

Bảng 3.2 Mô tả chi tiết các kiểu view trong Xamarin Forms

|  |  |
| --- | --- |
| **View** | **Mô tả chi tiết** |
| Entry | cho phép người dùng nhập một dòng văn bản |
| Lable | được sử dụng để hiển thị văn bản, cả một dòng và nhiều dòng. Nhãn có thể có trang trí văn bản, văn bản màu và sử dụng phông chữ tùy chỉnh (họ, kích thước và tùy chọn) |
| Button | hiển thị một chuỗi văn bản ngắn chỉ ra một lệnh, có thể hiển thị hình ảnh hoặc sự kết hợp giữa văn bản và hình ảnh. Người dùng nhấn nút bằng ngón tay hoặc nhấp vào nút bằng chuột để bắt đầu lệnh đó. |
| Image | hiển thị hình ảnh, icon để trang trí cho giao diện |
| BoxView | hiển thị một hình chữ nhật đơn giản có chiều rộng, chiều cao và màu sắc được chỉ định. Bạn có thể sử dụng BoxView để trang trí, đồ họa thô sơ và để tương tác với người dùng thông qua chạm |
| ListView | một dạng xem để trình bày danh sách dữ liệu, đặc biệt là các danh sách dài yêu cầu cuộn |

#### Layouts

Layouts là thành phần chứa các Views trong Xamarin, các View được đặt trong Layout một cách có cấu trúc. Có hai loại layout: “layout with single content” và “layout with multiple content”. Phần này sẽ trình bày về các loại “layout with multiple content”, có thể chứa và sắp xếp nhiều view bên trong như trong hình 3.4:

A picture containing text, screenshot

Description automatically generated

Hình 3.4 Các layouts trong Xamarin Forms

Sau đây là mô tả chi tiết về các layout được trình bày trong bảng 3.3:

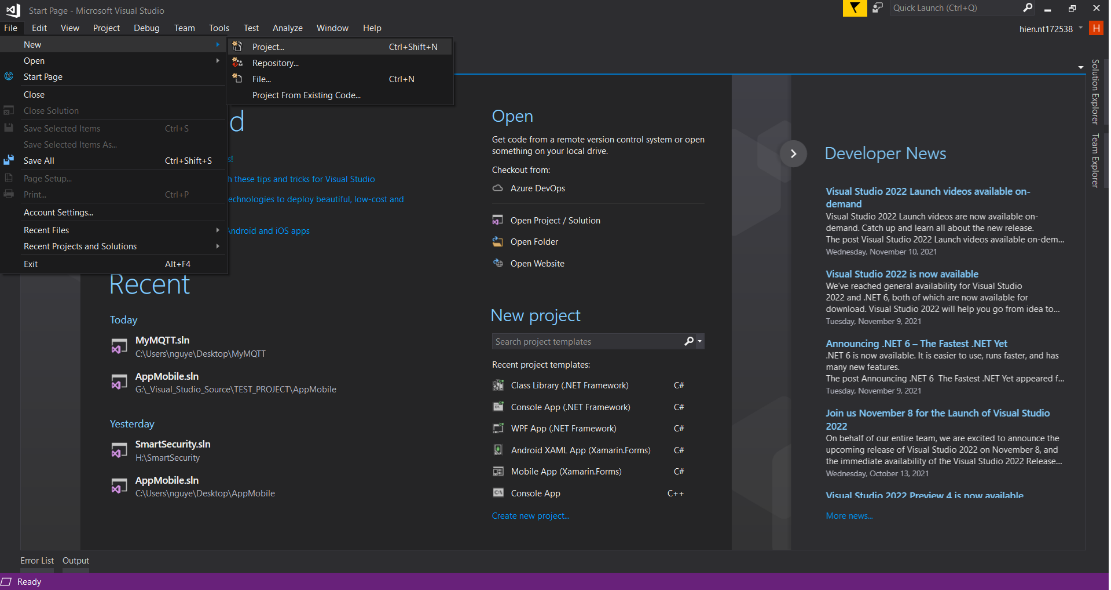
Bảng 3.3 Mô tả chi tiết các layouts trong Xamarin Forms

|  |  |
| --- | --- |
| **Kiểu Layouts** | **Mô tả chi tiết** |
| StackLayout | Nó tổ chức các Views theo cấu trúc ngăn xếp (stack), có thể theo chiều dọc hoặc ngang. Các Views trong StackLayout có thể thay đổi kích thước khoảng cách trong layout bằng cách sử dụng layout options |
| AbsoluteLayout | Kích thước và vị trí của các thành phần con trong AbsoluteLayout tỉ lệ với kích thước và vị trí của nó hoặc được đặt các giá trị cố định. Các View con có thể được đặt kích thước và vị trí sử dụng tỉ lệ hoặc các giá trị tĩnh, tỉ lệ và các giá trị tĩnh này có thể cùng được kết hợp. |
| RelativeLayout | Nó được sử dụng để đặt vị trí và kích thước các views có liên quan đến các thuộc tính của layout hoặc các view bên cạnh. |
| Grid | Gid hỗ trợ sắp xếp các View theo hàng và cột. Hàng và cột có thể được đặt kích thước theo tỉ lệ hoặc giá trị cố định. Grid không nên dùng để thay thế một bảng (table). |
| ScrollView | Chứa các Layouts và cho phép cuộn khi nội dung vượt quá kích thước màn hình, Nó cho phép các Views tự động di chuyển lên dùng nhìn thấy của màn hình khi bàn phím được hiển thị. |

## Triển khai trên Xamarin

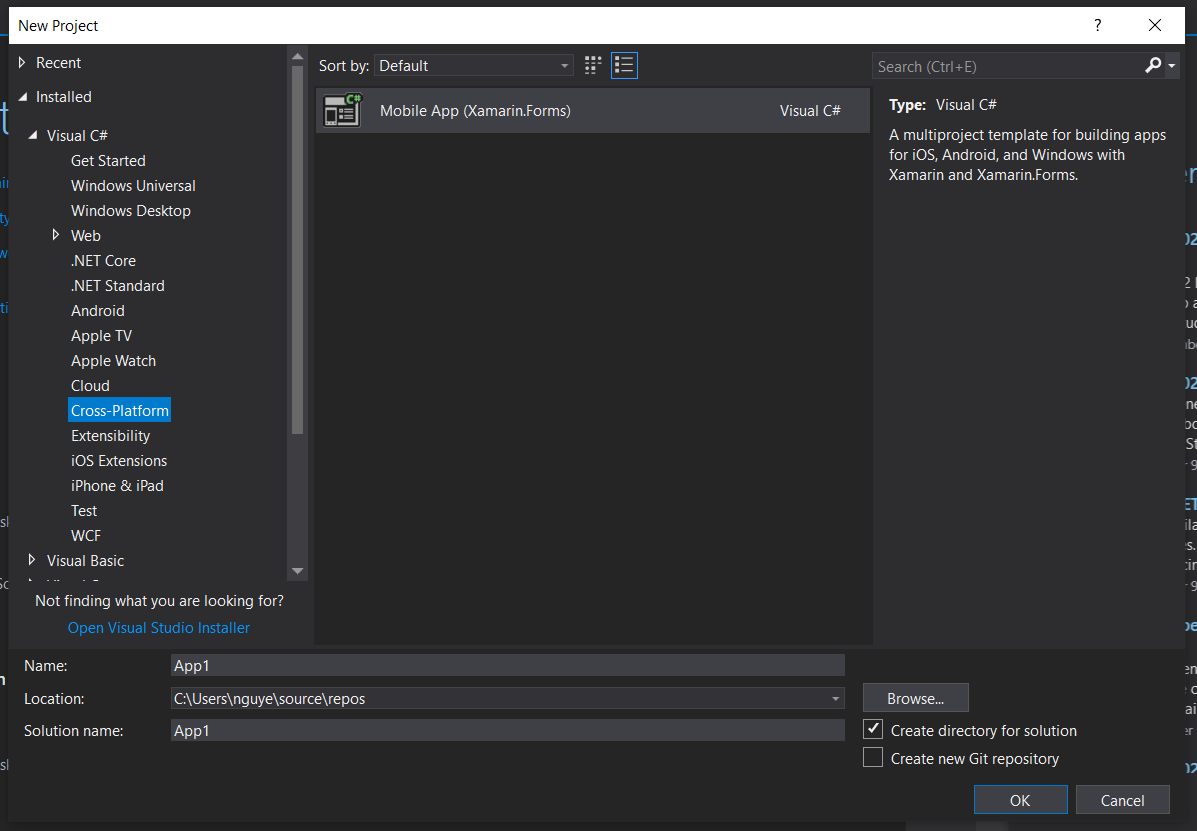
### Tạo project trên Xamarin (Windows)

Bước 1: Mở VisualStudio chọn vào File 🡪 New 🡪 Project.



Hình 3.5 Tạo một project mới

Bước 2: Vảo Visual C# chọn Cross-Platform và chọn Mobile App (Xamarin.Forms).



Hình 3.6 Chọn loại project

Bước 3: Điền tên ứng dụng, chọn thư mục lưu trong location và chọn button OK trong hình 3.6 để tiếp tục.

Bước 4: Tích vào cả 2 mục Android, iOS và bỏ tích UWP để tạo project cho cả 2 nền tảng, ở đây do thiếu trang thiết bị nên đồ án này chỉ thực hiện với nền tảng Android.

Graphical user interface, text, application

Description automatically generated

Hình 3.7 Target Platforms

Ta được New Project như hình 3.8

A screenshot of a computer

Description automatically generated

Hình 3.8 New Empty Project

Bước 5: Tạo thư mục mô hình MVC

Text

Description automatically generated

Hình 3.9 Tạo thư mục theo mô hình MVC trong Project

Như hình 3.9 đã thể hiện, project có các folder Controller Views và một project Models bên ngoài, đây là Models được thiết kế dùng chung cho cả hệ thống như bản vẽ Class Diagram hình 2.10.

### Tiến hành Code phần Base

#### Giao thức liên lạc

Để có thể kết nối với Sever thông qua giao thức MQTT, cần tạo ra một MqttClient để kết nối đến host của Sever. Trong báo cáo này, quá trình connect, subcribe, publish được định nghĩa trong BaseController như hình 3.10:

Text

Description automatically generated

Hình 3.10 Class BaseController

Trong hình 3.10 BaseController là Controller cơ sở để các Controller khác kế thừa, nó có chứa giá trị Response, là kết quả mà server trả về cho client sau mỗi request.

Nó gồm các thuộc tính như thuộc tính \_topic là topic để liên lạc với Server, client sử dụng topic này để publish lên server mà nó kết nối. Private topic là \_topic gắn thêm clientId, là topic mà phía client sẽ nghe để nhận response từ phía Server trả về.

Ngoài ra, cần quan tâm đến một số hàm quan trọng như: ConnectMqtt() là hàm thực hiện kết nối với Server, tham số truyền vào checkConnectionSeconds, là thời gian đặt lịch để kết nối lại; hàm Publish() tham số truyền vào url là url của request như trong bảng 3.2 và value đính kèm, hàm này gọi hàm Publish() trên nó có truyền vào tham số topic là thuộc tính \_topic của class; hàm Subscribe() truyền vào topic để đăng kí nghe PrivateTopic, hoặc các topic khác nếu phát triển thêm.

Hàm MqttMsgReceived() được gọi khi client nhận được một bản tin, hàm này sẽ đưa message nhận được vào thuộc tính Response của class và điều hướng đến ResponseController trong một luồng mới.

Sau khi request lên Sever, client sẽ nhận được response và được điều hướng đến ResponseController để xử lý chúng như hình 3.11:

Graphical user interface, application

Description automatically generated

Hình 3.11 Class ResponseController

Ban đầu client sẽ được điều hướng vào action Default() của class, tại đây sẽ lấy ra url trong Response để chuyển hướng xử lý xuống một trong những hàm được định nghĩa ở dưới, tên các hàm có dạng ControllerName\_Action như hình 3.11.

#### Các Page cơ sở

Class BaseView như hình 3.12 tạo ra một ContentPage, được sử dụng để tạo ra những page đơn lẻ, như page login. BaseView được truyền vào 2 tham số là TView và TModel để tiến hành render một view với model để binding. Đây là một class render cơ sở để các loại page khác kế thừa.

Graphical user interface, text, application

Description automatically generated

Hình 3.12 Class BaseView

Loại page thứ hai được sử dụng trong project là NavigationPage được mô tả trong bảng 3.1. class MyNavigationRootPage kế thừa từ BaseView và interface IRootPage, bằng cách viết lại hàm CreatePage() trong BaseView, nó tạo ra một NavigationPage với root là ContentPage mà BaseView trả về như trong hình 3.13.

Graphical user interface, text, application, email

Description automatically generated

Hình 3.13 Class MyNavigationRootPage

Tiếp theo, dưới đây là Class MyNavigationItemPage hình 3.14, giống như class MyNavigationRootPage, class này cũng kế thừa từ BaseView, trả về một ContentPage, là một page item trong một NavigationPage.

A picture containing text

Description automatically generated

Hình 3.14 Class MyNavigationItemPage

#### Các View cơ sở

Để thuận tiện trong quá trình viết giao diện, ta cần tạo ra một số form view cơ sở để các class khác kế thừa và sử dụng. Đầu tiên, trong đồ án này thiết kế class MyMenuView để tạo ra các view có dạng menu như view setting, việc sử dụng class này để viết view sẽ giúp các view có tính đồng bộ và thuận tiện trong quá trình chỉnh sửa thiết kế, thêm bớt các giao diện.

Graphical user interface, text, application

Description automatically generated

Hình 3.15 Class MyMenuView

Như hình 3.15, ta thấy class MyMenuView là một view kế thừa từ StackLayout được mô tả chi tiết trong bảng 3.3. Class này sử dụng 2 class khác là MyMenuItemInfo (mô tả các thuộc tính của một menu item), MyMenuItemView mô tả view của một item trong menu.

MyMenuView gồm các hàm chính Add, AddGroup lần lượt có chức năng là thêm một item vào menu bằng cách truyền vào MyMenuItemView hoặc một MyMenuItemInfo, và AddGroup() – thêm một list các item cùng nhóm vào menu bằng cách truyền vào List các MyMenuItemInfo.

Ngoài ra, class này còn cung cấp hàm Separate() để tạo ra phần view phân chia giữa các group trong menu, nó truyền vào các tham số, height – chiều cao của view, top – caption phía trên của view, bottom – caption phía dưới của view.

Tiếp theo, ta cũng cần xây dựng một class view cơ sở khác để có thể nhanh chóng tạo ra ListView cho một form xaml có sẵn như class MyListView trong hình 3.16. Class này có tác dụng render một TContent (một view truyền vào) thành một list các view đó với dữ liệu được binding từ model.

Graphical user interface, text, application

Description automatically generated

Hình 3.16 Class MyListView

MyListView cung cấp một hàm Binding truyền vào model và controller name, hàm này có tác dụng tạo ra các view item trong list view và binding data từ phía model vào view đó.

## Giao diện của một số chức năng đã thực hiện

### Giao diện đăng nhập

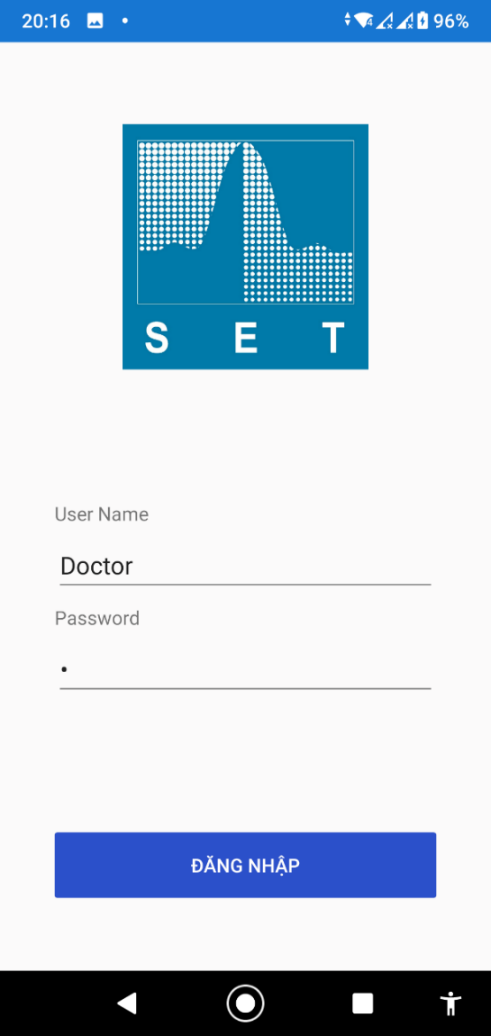
Như hình 3.17, giao diện đăng nhập được render từ LoginForm (là một view thiế kế bằng xaml) và model kiểu LoginInfo. Nó sử dụng ContentPage từ class BaseView vì giao diện đăng nhập sẽ chỉ xuất hiện một lần trong một phiên làm việc của ứng dụng.

A screenshot of a computer

Description automatically generated with medium confidence

Hình 3.17 Code giao diện đăng nhập

Kết quả giao diện như hình 3.18:



Hình 3.18 Giao diện đăng nhập

### Giao diện danh sách bệnh nhân

Như hình 3.19, giao diện danh sách bệnh nhân được render từ MyPatientListView (là một view list được render từ class MyListView) và model truyền vào từ controller. Nó là một root của NavigationPage, để chứa các page phía sau như detail của bệnh nhân.

Graphical user interface, text, application, email

Description automatically generated

Hình 3.19 Code giao diện danh sách bệnh nhân

Kết quả giao diện như hình 3.20:

Graphical user interface, application

Description automatically generated

Hình 3.20 Giao diện danh sách bệnh nhân

### Giao diện cài đặt

Như hình 3.21, giao diện cài đặt được render từ MySettingView (là một view menu kế thừa từ MyMenuView và sử dụng các hàm bên trong nó để tạo ra các view item) và model truyền vào từ controller. Nó là một root của NavigationPage, để chứa các page phía sau như đổi mật khẩu.

Graphical user interface, text, application

Description automatically generated

Hình 3.21 Code giao diện cài đặt

Kết quả giao diện như hình 3.22:

Graphical user interface, application, Teams

Description automatically generated

Hình 3.22 Giao diện cài đặt

### Giao diện đổi mật khẩu

Như hình 3.19, giao diện đổi mật khẩu được render từ AcountForm (là một view thiết kế bằng xaml) và model truyền vào từ controller. Nó là một item của NavigationPage, được thêm vào phía sau của setting view.

Graphical user interface, text, application

Description automatically generated

Hình 3.23 Code giao diện đổi mật khẩu

Kết quả giao diện như hình 3.24:

Graphical user interface, application

Description automatically generated

Hình 3.24 Giao diện đổi mật khẩu

## Kết luận

Như vậy, chương ba đã trình bày về việc triển khai ứng dụng mobile của hệ thống theo dõi bệnh nhân từ xa bằng mô hình MVC trên công cụ Xamarin, tiến hành thực hiện một số chức năng cơ bản của hệ thống theo dõi bệnh nhân từ xa. Việc sử dụng Xamarin để triển khai ứng dụng di động không hoàn toàn tối ưu, tuy nhiên trên cơ sở đã xây dựng mô hình MVC dùng chung cho các nền tảng bằng ngôn ngữ C#, thì việc sử dụng Xamarin lại là lựa chọn tối ưu để thống nhất một mô hình chung cho các nền tảng.

# KẾT LUẬN

## Kết luận chung

Về kiến thức bản thân:

* Hiểu biết nhiều hơn về ngôn ngữ lập trình C#. Nắm được các kỹ thuật trong lập trình giao thức MQTT.
* Bổ sung kiến thức về cơ sở dữ liệu, mạng máy tính, cách truyền và nhận dữ liệu từ Client đến Server và các công nghệ mới.
* Nâng cao khả năng tự học, tự tìm hiểu, và khả năng làm việc nhóm, sự liên kết giữa các thành viên trong đồ án.
* Việc triển khai đồ án tốt nghiệp giúp tác giả có cái nhìn tổng quan về quá trình xây dựng toàn bộ hệ thống, từ việc phân tích, thiết kế hệ thống, đến quá trình triển khai, xây dựng hệ thống, hiển rõ hơn về bản chất, cấu trúc làm việc của một hệ thống.

Về chương trình

* Chương trình đã đáp ứng được các chức năng cơ bản của một ứng dụng theo dõi bệnh nhân đơn giản, giúp đỡ bác sĩ có thể theo dõi sức khỏe cùng lúc nhiều bệnh nhân mà không mất quá nhiều thời gian thăm khám.

Tuy nhiên, do thời gian phát triển ứng dụng, cũng như những hạn chế về mặt kiến thức nên đồ án này vẫn còn nhiều thiếu sót, rất mong sự góp ý của thầy cô để đồ án hoàn thiện tốt hơn, đáp ứng nhu cầu của người dùng nhiều hơn.

## Hướng phát triển

Trong tương lai, việc hoàn thiện các chức năng cơ bản và phát triển thêm các chức năng khác của hệ thống là khá dễ dàng đó là nhờ vào việc thiết kế hệ thống theo hướng hướng đối tượng.

Ứng dụng này hiện tại cho Android. Việc phát triển ứng dụng này trên iOS hoặc các dự án khác là điều có thể làm được.

Cải thiện tư duy hệ thống hiệu quả hơn, kĩ năng code tối ưu hơn, khả năng lập trình hướng đối tượng nói riêng và các ngôn ngữ lập trình hướng đối tượng khác nói chung được nâng cao

# TÀI LIỆU THAM KHẢO

1. Borade Samar Sarjerao and Amara Prakasarao, “Smart Healthcare Monitoring System Using MQTTProtocol”, in *2018 3rd International Conference for Convergence in Technology (I2CT),* Apr 2018.
2. [Online]. Available: <https://viblo.asia/p/mqtt-la-gi-vai-tro-cua-mqtt-trong-iot-V3m5WL3bKO7>, truy cập cuối cùng ngày 10/12/2021
3. [Online]. Available: <https://viblo.asia/p/nhung-diem-khac-biet-giua-sql-va-nosql-gGJ59b4rKX2>, truy cập cuối cùng ngày 10/12/2021
4. [Online]. Available: https://viblo.asia/p/mongodb-la-gi-co-so-du-lieu-phi-quan-he-bJzKmgoPl9N, truy cập cuối cùng ngày 10/12/2021
5. [Online]. Available: <https://vietnix.vn/tim-hieu-mo-hinh-mvc-la-gi/>, truy cập cuối cùng ngày 14/12/2021
6. [Online]. Available: <http://bugnetproject.com/visual-studio-la-gi/>, truy cập cuối cùng ngày 10/12/2021
7. [Online]. Available: <https://viblo.asia/p/tim-hieu-ve-xamarin-trong-lap-trinh-mobile-ORNZqwXN50n>, truy cập cuối cùng ngày 10/12/2021
8. [Online]. Available: https://viblo.asia/p/tim-hieu-ve-cach-thiet-ke-class-diagram-L4x5xLyY5BM, truy cập cuối cùng ngày 29/12/2021
9. [Online]. Available: [Activity diagram là gì? Cách xây dựng Activity Diagram – Viết luận văn thuê (vietluanvanthue.com)](https://vietluanvanthue.com/activity-diagram-la-gi-cach-xay-dung-activity-diagram/), truy cập cuối cùng ngày 29/12/2021
10. [Online]. Available: https://viblo.asia/p/nhung-uu-khuyet-diem-cua-xamarin-DZrGNNlxGVB, truy cập cuối cùng ngày 24/02/2022
11. [Online]. Available: https://freetuts.net/tong-quan-ve-xamarin-form-1498.html#, truy cập cuối cùng ngày 24/02/2022