

TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT
KHOA KỸ THUẬT - CÔNG NGHỆ



BÁO CÁO

THỰC TẬP TỐT NGHIỆP

ĐỀ TÀI:

VoIP Call

Sinh viên thực hiện:	<i>Trần Phước Đức</i>
Mã số sinh viên:	<i>1624801040012</i>
Lớp:	<i>D16HT01</i>
Khoá:	<i>2016 - 2020</i>
Chuyên ngành:	<i>Hệ thống Thông Tin</i>
Giảng viên hướng dẫn:	<i>ThS. Dương Thị Kim Chi</i>



Bình Dương, 8/2019

TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT
KHOA KỸ THUẬT - CÔNG NGHỆ



BÁO CÁO

THỰC TẬP TỐT NGHIỆP

ĐỀ TÀI:

VoIP Call

Sinh viên thực hiện:	<i>Trần Phước Đức</i>
Mã số sinh viên:	<i>1624801040012</i>
Lớp:	<i>D16HT01</i>
Khoá:	<i>2016 - 2020</i>
Chuyên ngành:	<i>Hệ thống Thông Tin</i>
Giảng viên hướng dẫn:	<i>ThS. Dương Thị Kim Chi</i>



Bình Dương, 8/2019

-----🌀📖🌀-----

LỜI MỞ ĐẦU

Ngày nay, chúng ta đang được sống trong kỷ nguyên của tin học nhờ sự vượt bậc, sự bùng nổ mạnh mẽ của công nghệ thông tin. Công nghệ thông tin không chỉ dừng lại ở mục đích phục vụ cho khoa học kỹ thuật mà đi sâu vào đời sống, chính trị, kinh tế, xã hội, trở nên thân thiện, gần gũi, mang lại nhiều lợi ích cho con người. Công nghệ thông tin ngày càng khẳng định được tính hữu dụng và sức mạnh trong mọi phương diện, mọi ngành nghề của cuộc sống, nhất là trong thời đại kinh tế thị trường như bây giờ.

Đi kèm theo đó, nhu cầu về giải trí, học tập, liên lạc,... của con người cũng được nâng cao. Đời sống con người càng gắn bó hơn với công nghệ và internet thì đó cũng là thách thức lớn đối với đội ngũ kỹ sư Công nghệ Thông tin của nước nhà. Do đó, học phần Thực tập Tốt nghiệp rất quan trọng đối với sinh viên nói chung và cá nhân em nói riêng. Được sự giới thiệu và hướng dẫn thực tập tại Công ty TMA Solutions, đó là cơ hội cũng như là thách thức của bản thân em khi được tiếp xúc với môi trường doanh nghiệp ngay từ trên ghế nhà trường.

Trong thời gian thực tập tại doanh nghiệp, được sự giúp đỡ của ThS. Dương Thị Kim Chi và Anh Đàm Thuận Hoài Nam – Cán bộ hướng dẫn thực tập tại đơn vị, cùng với các anh chị trong Công ty TMA Solutions, em đã hoàn thành bài báo cáo thực tập của mình với đề tài “VoIP Call”. Tuy nhiên với kiến thức được học còn hạn chế và thời gian học công nghệ cũng như ngôn ngữ mới chưa nhiều nên không thể tránh khỏi những thiếu sót. Rất mong được sự góp ý của Quý Thầy Cô để đề tài và báo thực tập tốt nghiệp được tốt hơn.

Em xin gửi lời cảm ơn sâu sắc đến Quý Thầy Cô cùng các anh chị trong Công ty TMA Solutions đã giúp em hoàn thành học phần Thực tập Tốt nghiệp này.

Bình Dương, ngày 10 tháng 08 năm 2019

Sinh viên thực hiện

Trần Phước Đức

MỤC LỤC

LỜI MỞ ĐẦU	ii
DANH MỤC HÌNH	iv
DANH MỤC BẢNG	vii
CHƯƠNG 1. TỔNG QUAN VỀ CƠ SỞ THỰC TẬP	1
1.1. Tổ chức hành chính nhân sự của cơ quan:	1
1.1.1. Tổng quan về công ty.....	1
1.1.2. Tổ chức hành chính.....	2
1.1.3. Tổ chức nhân sự.....	3
1.2. Các hoạt động chuyên ngành và môi trường làm việc của công ty	7
1.2.1. Hoạt động chuyên ngành.....	7
1.2.2. Các trung tâm.....	8
1.2.3. Môi trường làm việc của cơ quan.....	8
1.2.4. Các bằng khen.....	8
1.2.5. Thế mạnh công ty.....	9
CHƯƠNG 2. NỘI DUNG THỰC TẬP TẠI DOANH NGHIỆP	10
2.1. Mô tả công việc	10
2.2. Phần lý thuyết	11
2.2.1. Windows Presentation Foundation (WPF).....	11
2.2.2. Mô hình MVVM.....	12
2.2.3. Material Design In XAML Toolkit.....	13
2.2.4. MVVM Light Toolkit.....	18
2.2.5. TCP/IP.....	22
2.3. Phần thực hành	24
2.3.1. Thiết lập Server/Client.....	24
2.3.2. Truyền nhận Audio thông qua Socket.....	33
2.3.3. Ứng dụng VoIP Call.....	41
CHƯƠNG 3. ĐỀ XUẤT CÁC GIẢI PHÁP CẢI THIỆN CHẤT LƯỢNG	
GIẢNG DẠY Ở NHÀ TRƯỜNG	47
CHƯƠNG 4. KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN	48
TÀI LIỆU THAM KHẢO	50

DANH MỤC HÌNH

Hình 1. Sơ đồ tổ chức hành chính.....	2
Hình 2. Sơ đồ tổ chức nhân sự.....	3
Hình 3. Sơ đồ tổ chức nhân sự DG2	4
Hình 4. Sơ đồ tổ chức nhân sự DG1	4
Hình 5. Sơ đồ tổ chức nhân sự DG3	5
Hình 6. Sơ đồ tổ chức nhân sự DG4	5
Hình 7. Sơ đồ tổ chức TMA Overseas	6
Hình 8. Sơ đồ tổ chức Business/IT.....	6
Hình 9. Sơ đồ tổ chức nhân sự HR/Admin Support.....	6
Hình 10. Sơ đồ tổ chức QMS/PMO	7
Hình 11. Sơ đồ tổ chức Finance/Legal	7
Hình 12. Sơ đồ tổ chức DCF.....	7
Hình 13. Sơ đồ mô hình MVVM	12
Hình 14. Giao diện chính của Material Design In XAML Toolkit.....	15
Hình 15. Menu với nhiều tùy chọn	16
Hình 16. Menu phối màu	16
Hình 17. Menu cho phép chọn và sao chép buttons và toggles được thiết kế sẵn ...	17
Hình 18. Menu thư viện icons được tích hợp sẵn	17
Hình 19. Source code mẫu để sử dụng.....	18
Hình 20. Cấu trúc MVVM Light	18
Hình 21. Cấu trúc chi tiết của MVVM Light.....	18
Hình 22. Hàm DataItem chứa các dữ liệu sử dụng toàn bộ cục phần mềm	19
Hình 23. Interface của phần mềm	19
Hình 24. Hàm gọi giao diện thiết kế mặc định	19
Hình 25. Hàm gọi giao diện người dùng	19
Hình 26. Hàm khởi tạo và gọi Data và ViewModel	20
Hình 27. Hàm ViewModel	21
Hình 28. Mô hình TCP/IP và các tầng dữ liệu.....	22
Hình 29. Hàm Run trong Class Server	24
Hình 30. Hàm Start Server.....	25
Hình 31. Hàm Stop trong Server.Class	25
Hình 32. Hàm Send trong Class Server.....	26

Hình 33. Hàm Receive trong Class Server	27
Hình 34. Hàm RelayCommand của Button Server	28
Hình 35. Hàm StartServer của Button Server	28
Hình 36. Hàm StopServer cho Button Server	29
Hình 37. Hàm Connect trong Class Client	29
Hình 38. Hàm Disconnect trong Class Client	30
Hình 39. Hàm Send trong Class Client	30
Hình 40. Hàm OnDataReceive trong Class Client	31
Hình 41. Hàm RelayCommand trong MainViewModel	32
Hình 42. Hàm ConnectClient trong MainViewModel	32
Hình 43. Hàm DisconnectClient trong MainViewModel	33
Hình 44. Hàm đóng gói âm thanh	33
Hình 45. Hiển thị các thiết bị đầu ra/đầu vào	34
Hình 46. Bộ đệm âm thanh của Client	34
Hình 47. Thiết lập bộ đệm phát âm thanh	35
Hình 48. Thiết lập bộ đệm ghi âm thanh	35
Hình 49. Thực hiện ghi âm thanh từ thiết bị đầu vào	36
Hình 50. Thực hiện dừng ghi âm thanh	36
Hình 51. Truyền âm thanh đã được mã hoá đến các Client	37
Hình 52. Dừng truyền âm thanh đến các Client	37
Hình 53. Hàm SendConfiguarationToClient	37
Hình 54. Hàm OnClientConfigReceived	38
Hình 55. Thực hiện phát âm thanh qua thiết bị đầu ra	39
Hình 56. Dừng phát âm thanh qua thiết bị đầu ra đến Server	39
Hình 57. Thực hiện ghi âm thanh từ thiết bị đầu vào của Client	40
Hình 58. Thực hiện việc dừng ghi âm thanh từ thiết bị đầu vào của Client	40
Hình 59. Giao diện chính	42
Hình 60. Giao diện Server	42
Hình 61. Giao diện Client	43
Hình 62. Multitab trong ứng dụng	43
Hình 63. Server khi được khởi tạo	44
Hình 64. Server hiển thị users sau khi được kết nối	44
Hình 65. Client sau khi kết nối đến Server	45

Hình 66. Giao diện cho phép chọn OutPut và InPut từ thiết bị.....	45
Hình 67. Lịch sử cuộc gọi.....	46
Hình 68. Thông báo khi có cuộc gọi đến.....	46

DANH MỤC BẢNG

Bảng 1. Mô tả công việc	11
-------------------------------	----

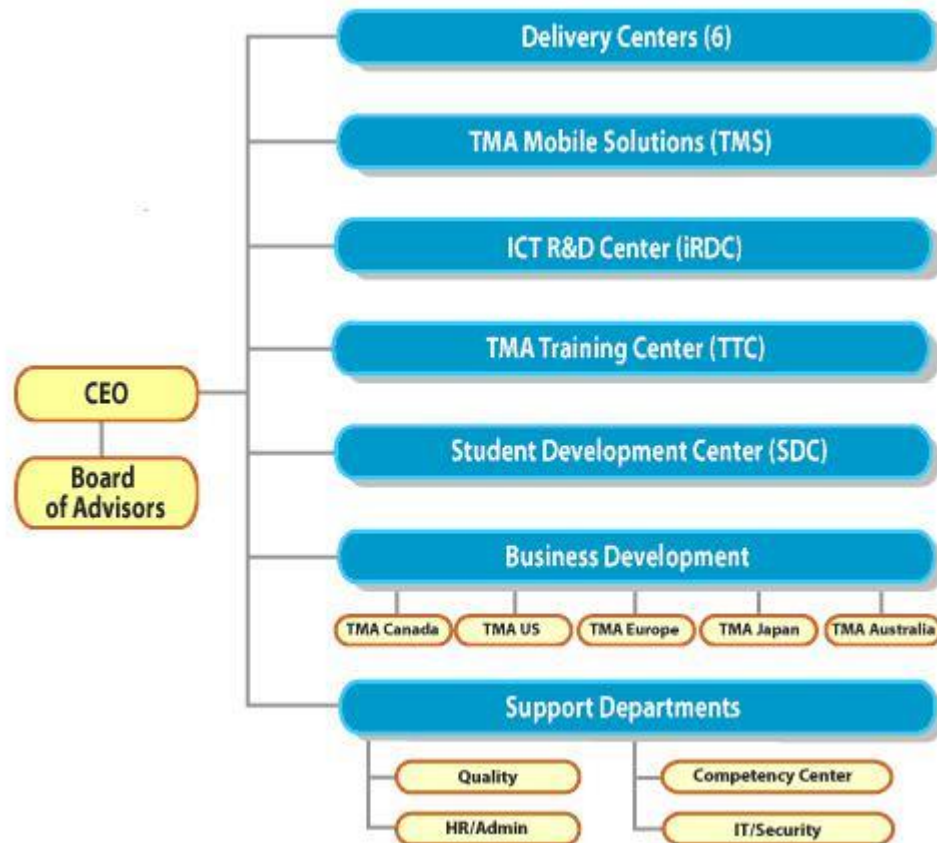
CHƯƠNG 1. TỔNG QUAN VỀ CƠ SỞ THỰC TẬP

1.1. Tổ chức hành chính nhân sự của cơ quan:

1.1.1. Tổng quan về công ty

- Doanh nghiệp tư nhân và dịch vụ Tường Minh (TMA Solutions) được thành lập vào tháng 3 năm 1997 để cung cấp chất lượng dịch vụ gia công phần mềm cho các công ty hàng đầu trên toàn thế giới (Nortel, IBM, Alcatel-Lucent, Juniper Networks, Flextronics, Genband, NTT, Toshiba, Samsung, NEC). Đây là công ty gia công phần mềm lớn tại thành phố Hồ Chí Minh với khoảng 1,800 kỹ sư (thống kê năm 2015). Đội ngũ kỹ thuật của TMA đã được lựa chọn từ một hồ bơi lớn của nguồn lực CNTT Việt Nam và khả năng của nó đã được chứng minh trong nhiều dự án thành công. TMA cung cấp đầy đủ các dịch vụ phần mềm, từ kiểm tra và bảo dưỡng để phát triển chu kỳ đầy đủ và giải pháp kết thúc. Sức mạnh chính của TMA là khả năng xử lý các dự án lớn và phức tạp với chất lượng phù hợp
- Hiện tại, TMA có 12 trụ sở ở các thành phố lớn trên thế giới trong đó có 6 trụ sở tại thành phố Hồ Chí Minh (trụ sở chính đặt tại địa chỉ 111 Nguyễn Đình Chính, Quận Phú Nhuận, TP. Hồ Chí Minh) và các chi nhánh tại các nước như Canada, Mỹ, Nhật Bản, Úc,...
- Quy trình quản lý chất lượng đã đạt được các chuẩn: ISO 9000 (năm 2005), TL 9000, CMMI-Level 3 (năm 2006), CMMI-Level 5 (năm 2011) và Agile (năm 2012).
- Công ty DNTN Dịch vụ Tường Minh (TMA solution) thành lập vào tháng 3 năm 1997 tại phòng khách nhà bà Bùi Ngọc Anh với 6 kỹ sư.
- Đến nay công ty gồm 10 trụ sở, trong đó trụ sở chính tại 111, Đường Nguyễn Đình Chính, Quận Phú Nhuận, Thành Phố Hồ Chí Minh, Việt Nam.
- Tháng 3/2015 công ty có trên 1800 nhân viên.

1.1.2. Tổ chức hành chính

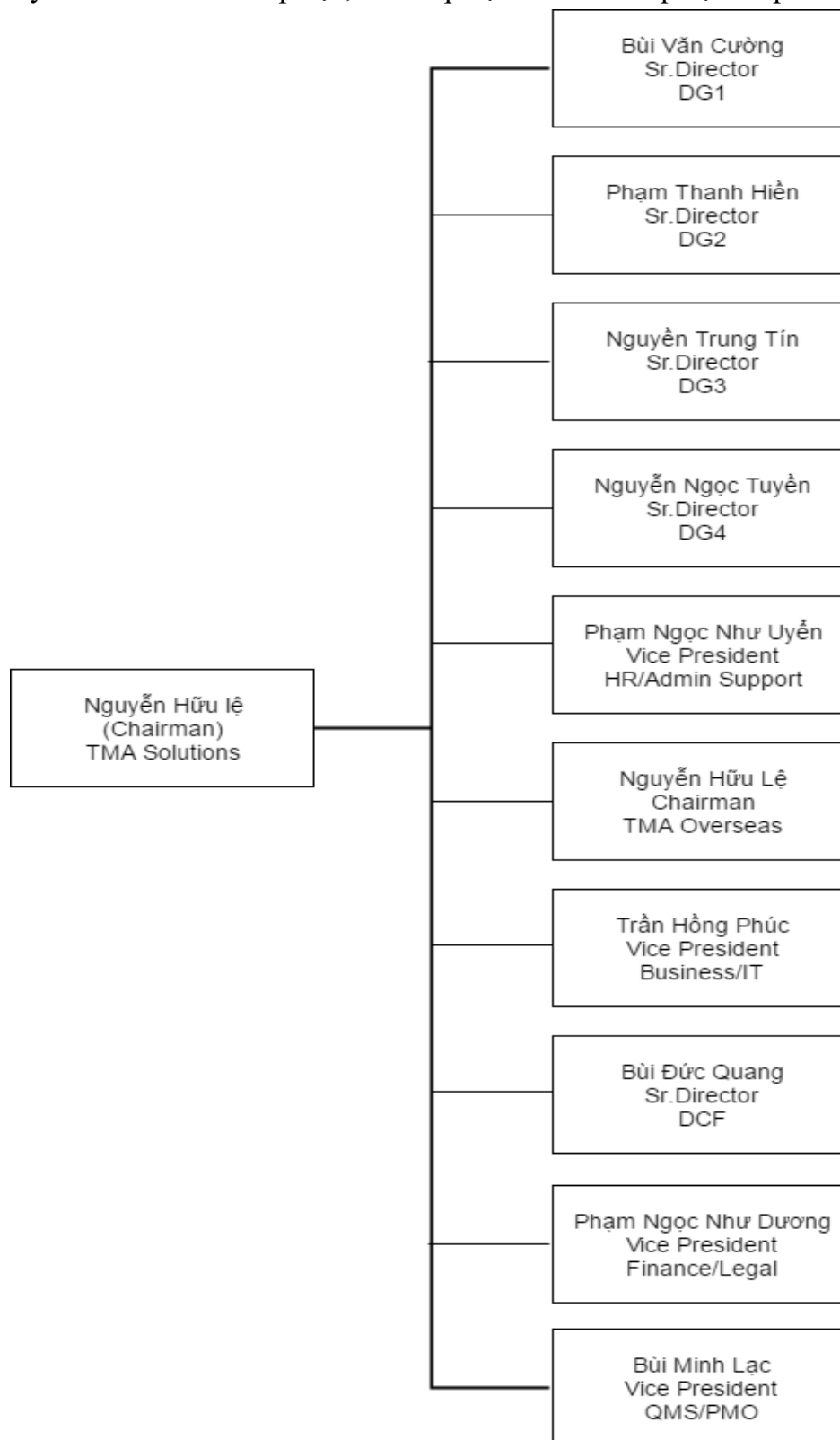


Hình 1. Sơ đồ tổ chức hành chính

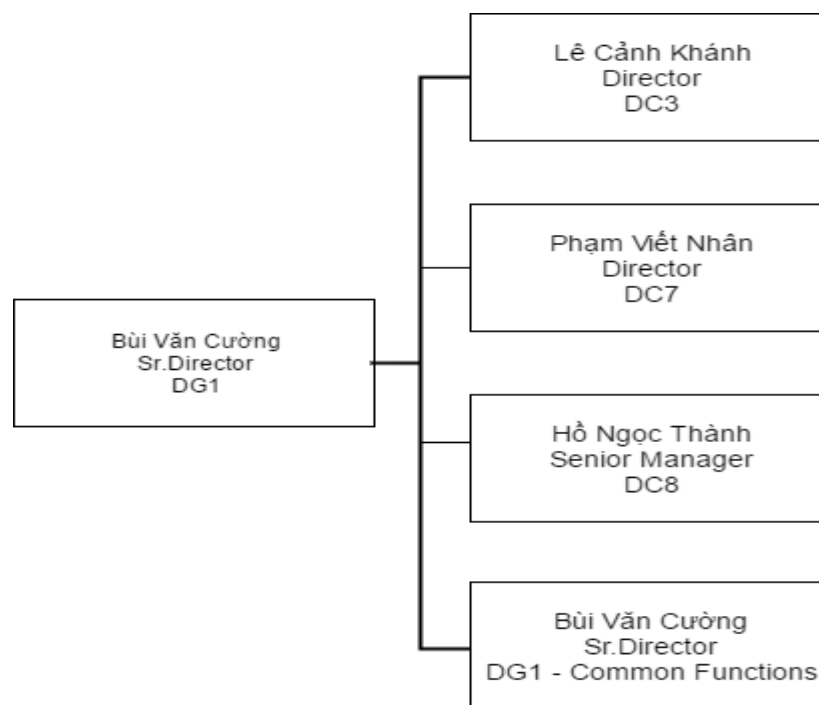
- **Board of Advisors:** Cố vấn cho công ty.
- **CEO:** Giám đốc điều hành của công ty
- **Delivery Centers:** Trung tâm chuyển giao sản phẩm cho khách hàng.
- **TMA Mobile Solutions (TMS):** Phát triển ứng dụng di động, thử nghiệm cũng như tạo ra các dịch vụ điện thoại di động, giá trị gia tăng và các ứng dụng cho các nhà khai thác di động, cung cấp nội dung và doanh nghiệp.
- **ICT R&D Center (iRDC - Information and Communications Technology Research & Development Center):** Trung tâm nghiên cứu và phát triển công nghệ thông tin và truyền thông.
- **TMA Training Center (TTC):** Trung tâm đào tạo nhân lực.
- **Student Development Center (SDC):** Có nhiệm vụ liên kết với sinh viên, nhận sinh viên thực tập tại công ty nhằm thu hút và bồi dưỡng nhân tài.
- **Business Development:** Tăng cường quan hệ với khách hàng hiện tại cũng như tạo thêm các mối quan hệ với các khách hàng khác.

1.1.3. Tổ chức nhân sự

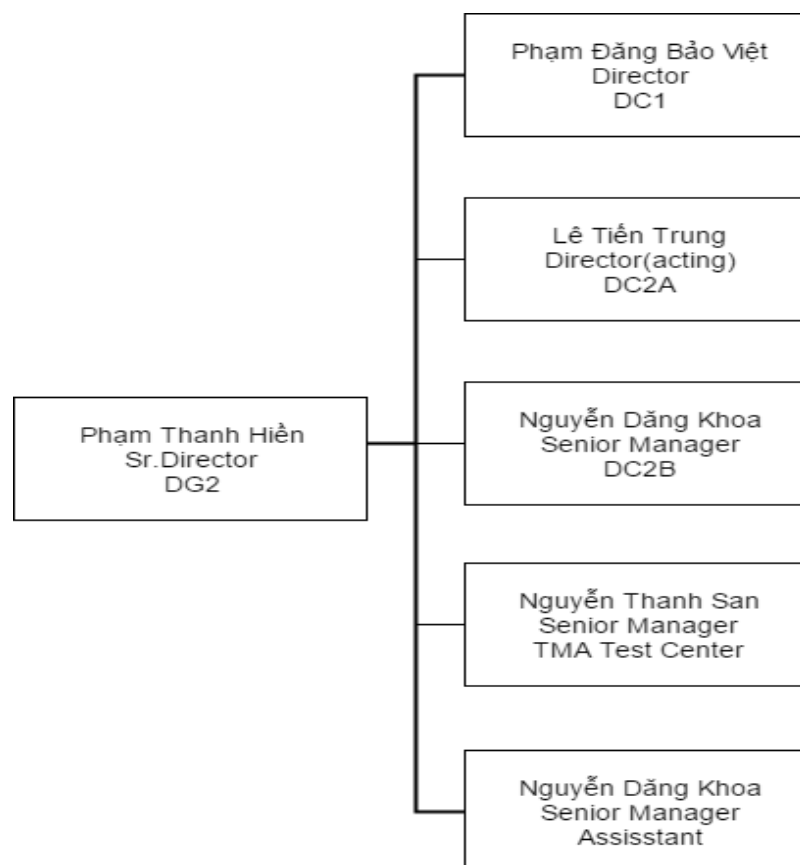
- Tính đến năm 2018 TMA Solutions có số lượng nhân viên là 2400 kỹ sư. Công ty chia theo nhiều cấp bậc, mỗi cấp bậc có nhiều cấp bậc thấp hơn.



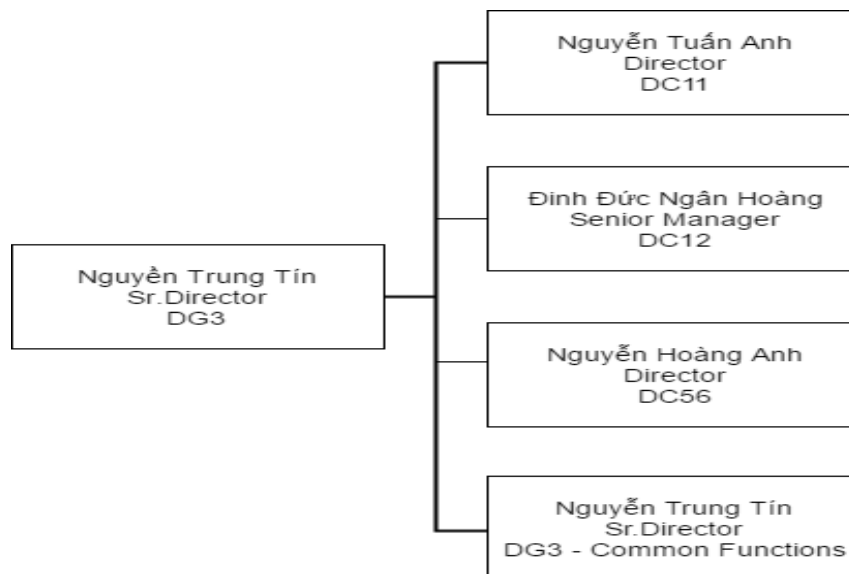
Hình 2. Sơ đồ tổ chức nhân sự



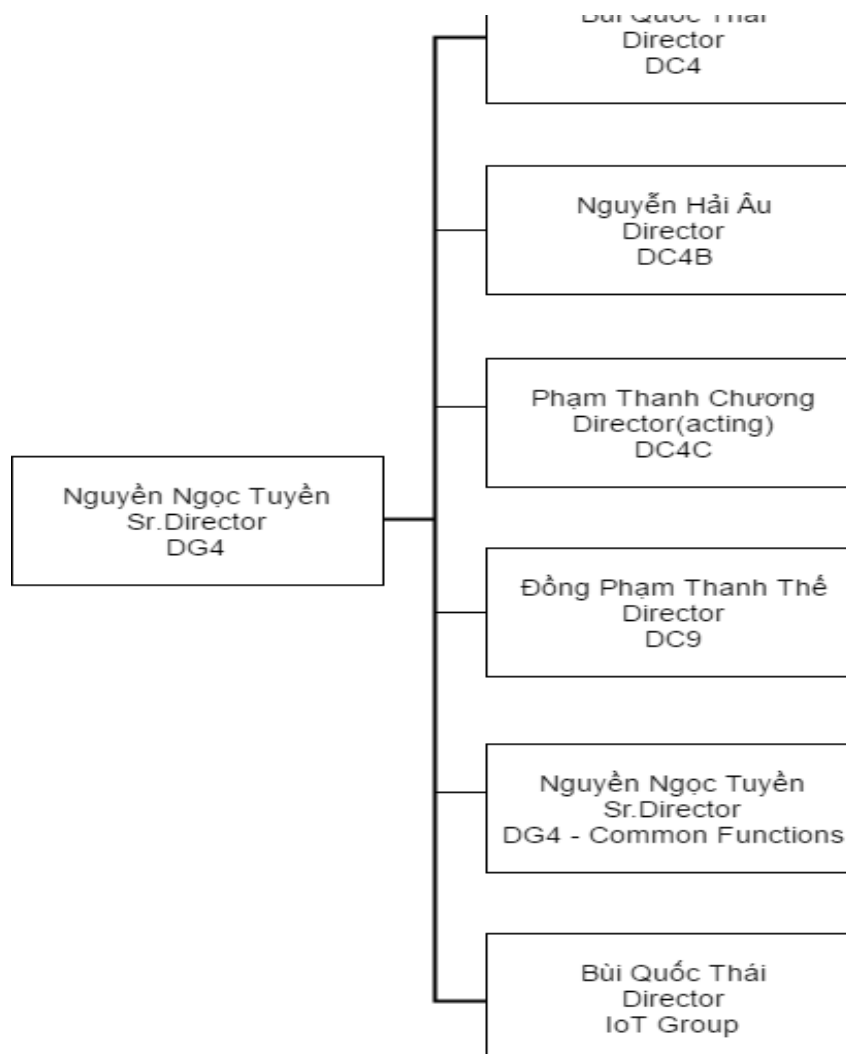
Hình 4. Sơ đồ tổ chức nhân sự DG1



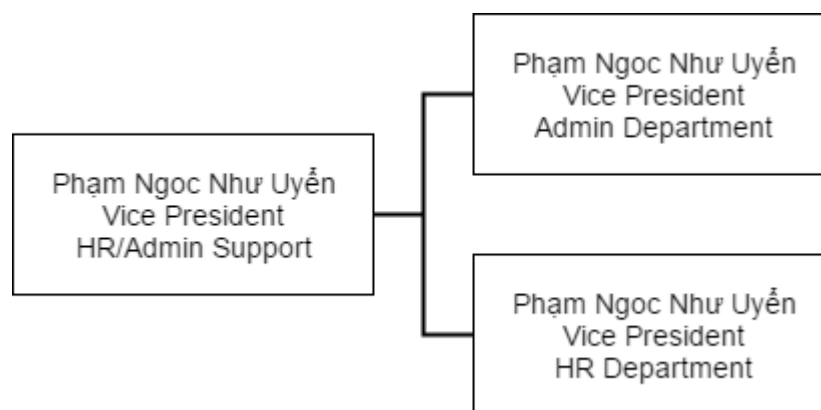
Hình 3. Sơ đồ tổ chức nhân sự DG2



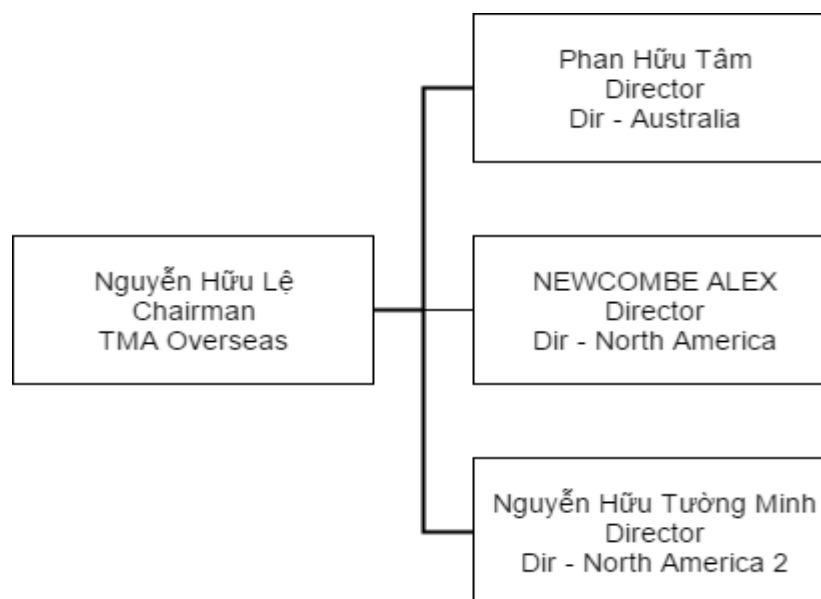
Hình 5. Sơ đồ tổ chức nhân sự DG3



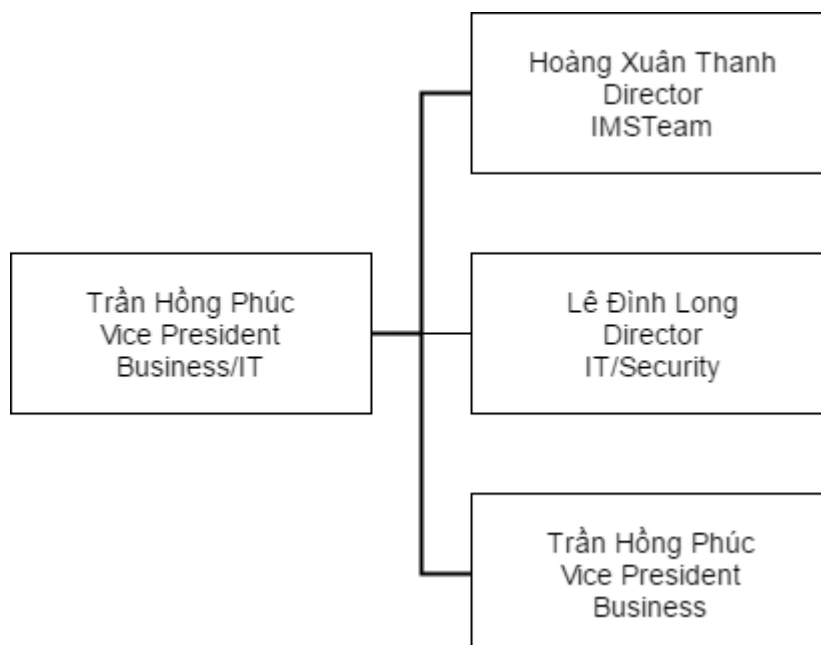
Hình 6. Sơ đồ tổ chức nhân sự DG4



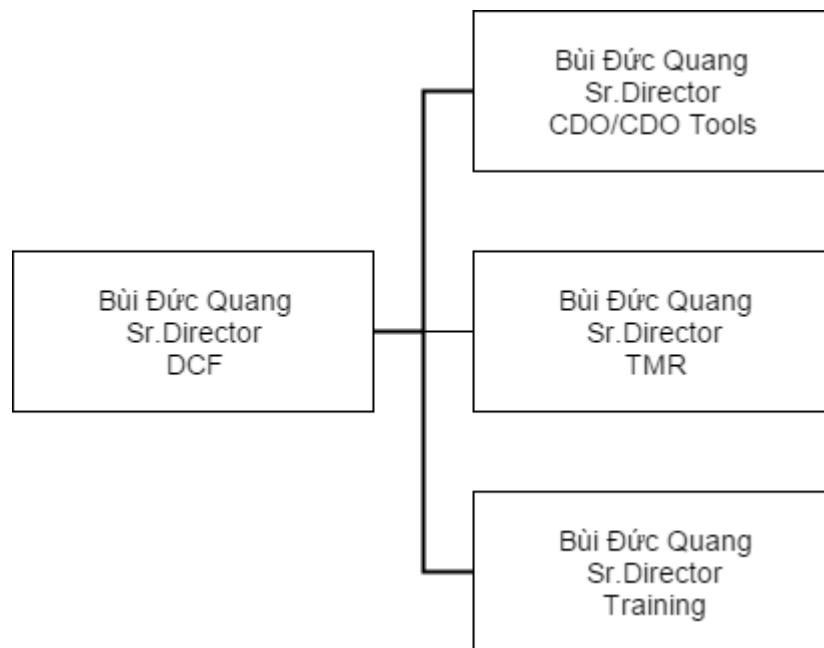
Hình 9. Sơ đồ tổ chức nhân sự HR/Admin Support



Hình 7. Sơ đồ tổ chức TMA Overseas



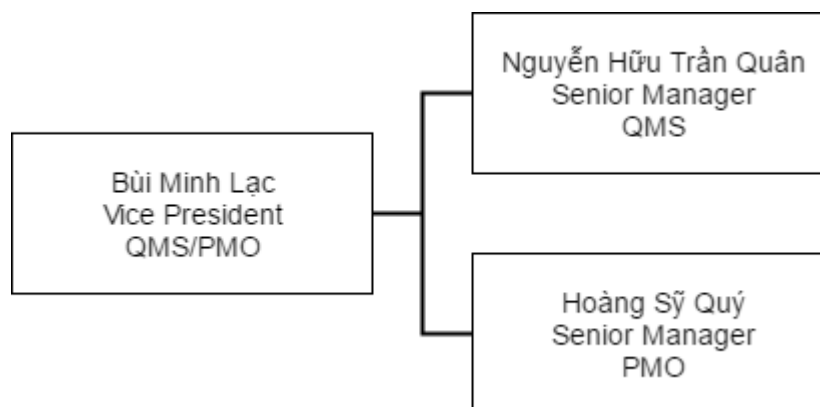
Hình 8. Sơ đồ tổ chức Business/IT



Hình 12. Sơ đồ tổ chức DCF



Hình 11. Sơ đồ tổ chức Finance/Legal



Hình 10. Sơ đồ tổ chức QMS/PMO

1.2. Các hoạt động chuyên ngành và môi trường làm việc của công ty

1.2.1. Hoạt động chuyên ngành

- Tích hợp hệ thống.
- Xuất khẩu phần mềm.
- Giải pháp phần mềm.
- Cung cấp các giải pháp, dịch vụ viễn thông

- Đào tạo sinh viên và nhân viên về kiến thức phần mềm và kỹ năng mềm

1.2.2. Các trung tâm

- Phòng Trung tâm Đào Tạo TMA
- Phòng Trung tâm Phát triển Sinh Viên
- Phòng Trung tâm Nghiên cứu và Phát triển (R&D)

1.2.3. Môi trường làm việc của cơ quan

- Môi trường làm việc chuyên nghiệp, thân thiện.
- Làm việc theo nhóm. Mỗi thành viên trong nhóm làm theo đúng công việc được phân công.
- Mỗi nhân viên được trang bị 1 máy tính kết nối mạng, 1 hay 2 màn hình tùy dự án.
- Đối với những án cần thiết bị để kiểm thử, mỗi nhân viên đều được trang bị đầy đủ.
- Phòng làm việc được trang bị đầy đủ thiết bị phục vụ cho công việc: máy tính, máy lạnh, máy in.

1.2.4. Các bằng khen

- Bằng khen của Ủy ban Nhân dân Thành phố Hồ Chí Minh: Có thành tích xuất sắc trong lĩnh vực CNTT - TT, góp phần tích cực vào sự phát triển của CNTT-TT của thành phố.
- Chứng nhận và cúp của Hội Tin học Thành Phố Hồ Chí Minh (HCA).
- Huy chương vàng xuất khẩu phần mềm (13 năm liền từ 2004 đến năm 2016).
- Top 5 đơn vị gia công phần mềm từ 2009 đến năm 2016.
- Bằng khen của VINASA: Có thành tích xuất sắc, đóng góp cho các hoạt động của Hiệp hội và cho sự phát triển của ngành phần mềm, dịch vụ công nghệ thông tin của Việt Nam.
- Là 1 trong 15 công ty hàng đầu thế giới trong việc áp dụng hiệu quả quy trình gia công phần mềm (Báo cáo của công ty tư vấn Aberdeen, 09/2002).
- Đối tác chính thức của Microsoft từ 2007 đến 2016.
- Đạt nhiều chứng chỉ quốc tế cao cấp (CMMI-L5, TL 9000, ISO 9001:2000, ISO 27001:2013).

1.2.5. Thế mạnh công ty

- Gia công phần mềm.
- Phát triển phần mềm.
- Kiểm thử phần mềm.
- Chuyển đổi công nghệ.
- Hỗ trợ sản xuất.
- Dịch vụ quản lý CNTT.
- Thiết kế giao diện người dùng.

CHƯƠNG 2. NỘI DUNG THỰC TẬP TẠI DOANH NGHIỆP**2.1. Mô tả công việc**

STT	Thời gian	Nội dung	Ghi chú
1	Tuần 1 (10 – 14/06/2019)	<ul style="list-style-type: none">- Tìm hiểu về quy định tại công ty- Nhận máy tính- Nhận thẻ nhân viên- Cài đặt các phần mềm cần thiết- Tham dự “Interview skill and teamwork”- Tham dự “Toastmaster #2”- Tìm hiểu về WPF- Tìm hiểu về mô hình MVVM	
2	Tuần 2 (17 – 21/06/2019)	<ul style="list-style-type: none">- Tìm hiểu về MVVM Light Toolkit- Tham dự “Toastmaster #3”- Nhận project và bảng phân công công việc	
3	Tuần 3 (24 – 28/06/2019)	<ul style="list-style-type: none">- Tìm hiểu về TCP/IP- Tham gia “Feature Development Process”	
4	Tuần 4 (01 – 05/07/2019)	<ul style="list-style-type: none">- Tham dự “Toastmaster #5”- Thiết kế các chức năng cho View-Model	
5	Tuần 5 (08 – 12/07/2019)	<ul style="list-style-type: none">- Tham dự “Toastmaster #6”- Tham dự “Communication skill”- Tìm hiểu về NAudio	
6	Tuần 6 (15 – 19/07/2019)	<ul style="list-style-type: none">- Tham dự “Toastmaster #7”	

		- Hoàn thành các chức năng, yêu cầu	
7	Tuần 7 (22 – 26/07/2019)	- Tham dự “Toastmaster #8” - Hoàn thành các chức năng, yêu cầu (tt)	
8	Tuần 8 (29/07 – 02/08/2019)	- Tham dự “Toastmaster #9” - Trả máy tính, các thiết bị, dụng cụ và thẻ nhân viên	

Bảng 1. Mô tả công việc

2.2. Phần lý thuyết

2.2.1. Windows Presentation Foundation (WPF)

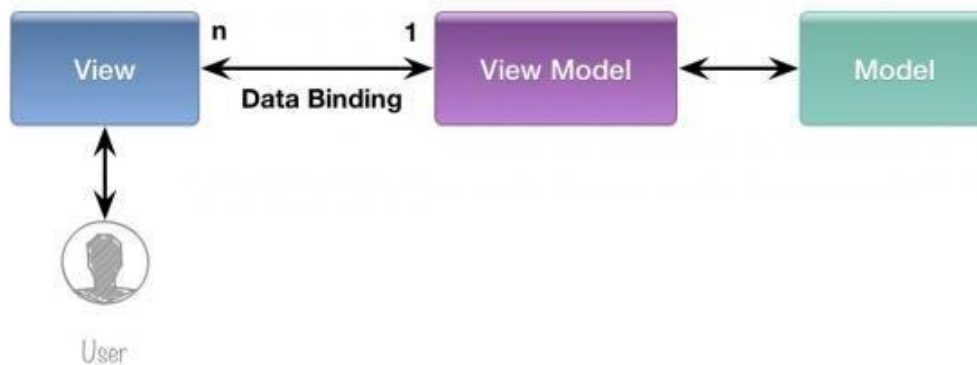
- Windows Presentation Foundation (viết tắt là WPF) do Microsoft phát triển, là công nghệ kế tiếp Windows Form dùng để xây dựng các ứng dụng dành cho máy trạm chạy hệ điều hành Windows. WPF được giới thiệu từ năm 2006 trong .NET Framework 3.0 (dưới tên gọi Avalon), công nghệ này nhận được sự quan tâm 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. Tại Việt Nam, WPF thực sự chưa phát triển so với nhánh khác là Silverlight (WPF/E).
- WPF sử dụng 2 thư viện lõi là PresentationCore và PresentationFramework để xử lý các điều hướng, ràng buộc dữ liệu, sự kiện và quản lý giao diện. WPF dựa trên nền tảng đồ họa là DirectX, xử lý vector, hỗ trợ gam màu rộng, cho phép tùy biến giá trị opacity hay tạo gradient một cách dễ dàng, cho phép tạo ảnh không gian 2 chiều hoặc 3 chiều. Thư viện thực thi của WPF tự động tính toán và tận dụng tài nguyên của hệ thống một cách tối ưu để giảm tải cho CPU.
- Ngoài ra, 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.

- Thư viện thực thi WPF được tích hợp trong tất cả các hệ điều hành kể từ Windows Vista và Windows Server 2008.
- Cho đến thời điểm hiện tại, WPF có 8 phiên bản: WPF 3.0 (11/2006), WPF 3.5 (11/2007), WPF 3.5sp1 (8/2008), WPF 4 (4/2010), WPF 4.5 (8/2012), WPF 4.5.1 (10/2013), WPF 4.5.2 (5/2014) và WPF 4.6 (7/2015).

2.2.2. Mô hình MVVM

2.2.2.1. Tổng quan về MVVM



Hình 13. Sơ đồ mô hình MVVM

- Kể từ khi Microsoft giới thiệu hai nền tảng phát triển ứng dụng mới là WPF và Silverlight, đã có nhiều thay đổi trong việc xử lý sự kiện và binding dữ liệu, giữa các tầng của ứng dụng với nhau. Qua đó, hầu hết các công việc của tầng kết hợp với lớp presentation. Điều này làm nảy sinh ra nhu cầu phải có một mô hình phát triển ứng dụng mới phù hợp hơn. Và do đó, Model – View – ViewModel (MVVM) pattern ra đời và ngày càng trở nên phổ biến.
- Đa số các ứng dụng thuộc bất kỳ nền tảng nào cũng có thể chia thành hai phần: giao diện (View) và dữ liệu (Model). Vì việc tách riêng các phần này, cần phải có một phần trung gian nào đó nối kết hai phần này lại, và chúng tạo nên một mô hình (pattern).
- Quen thuộc và phổ biến nhất với chúng ta là mô hình MVC (Model – View – Controller) . Có thể nói MVC là một mô hình tiêu chuẩn bởi sự

logic và hợp lý của nó. Điều này làm cho việc xuất hiện một mô hình phát triển ứng dụng mới có thể khiến bạn ngỡ ngàng.

2.2.2.2. Mô hình MVVM

- **View:**

- + View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, command.

- **View - Model:**

- + Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.

- **Model:**

- + Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.

2.2.3. Material Design In XAML Toolkit

2.2.3.1. Tổng quan về Material Design In XAML Toolkit

- Material Design là một phong cách thiết kế được Google phát triển và giới thiệu đến người dùng và lập trình viên cùng lúc với phiên bản Android 5.0 Lollipop.
- Phong cách thiết kế của Material Design nhắm đến những đường nét đơn giản, bằng cách sử dụng nhiều mảng màu đậm nổi bật và các đối tượng đồ họa xếp chồng lên nhau, Material Design mang đến cho người dùng cảm giác đối tượng thực tế hơn, ấn tượng hơn. Ngoài ra, Material Design bao gồm những hiệu ứng chuyển động đầy lý thú khi các nút, menu, nội dung,... hiển thị lên màn hình. Tất cả mọi thứ Material Design được thiết kế và phát triển nhằm mang lại trải nghiệm người dùng thêm mới mẻ, thú vị và gần với thực tế hơn.
- Material Design In XAML Toolkit là bộ công cụ mã nguồn mở giúp cho lập trình viên có thể thiết kế các ứng dụng Windows theo phong cách của Material Design và những bộ mã màu, cách chuyển động hiệu ứng mà Google đã quy định. Ngoài ra, nó còn là công cụ mạnh mẽ để giúp thiết

kế UI/UX chuyên nghiệp cho WPF và các ứng dụng chạy trên Windows Phone.

2.2.3.2. Cách cài đặt

- Để tải về bộ công cụ và mã nguồn của Material Design In XAML Toolkit, người dùng có thể truy cập tại đại chỉ:

<https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit>

- Ngoài ra, ta cài đặt *NuGet* từ *Package Manager Console* trên *Visual Studio* thông qua mã lệnh:

```
PM> Install-Package MaterialDesignThemes
```

- Trong file *App.xaml* ta cần bổ sung:

```
<?xml version="1.0" encoding="UTF-8"?>
<Application . . .>
  <Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/
MaterialDesignTheme.Light.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/
MaterialDesignTheme.Defaults.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/Reco
mmended/Primary/MaterialDesignColor.DeepPurple.xaml" />
        <ResourceDictionary
Source="pack://application:,,,/MaterialDesignColors;component/Themes/Reco
mmended/Accent/MaterialDesignColor.Lime.xaml" />
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

- Trong file *MainWindow.xaml* cần bổ sung:

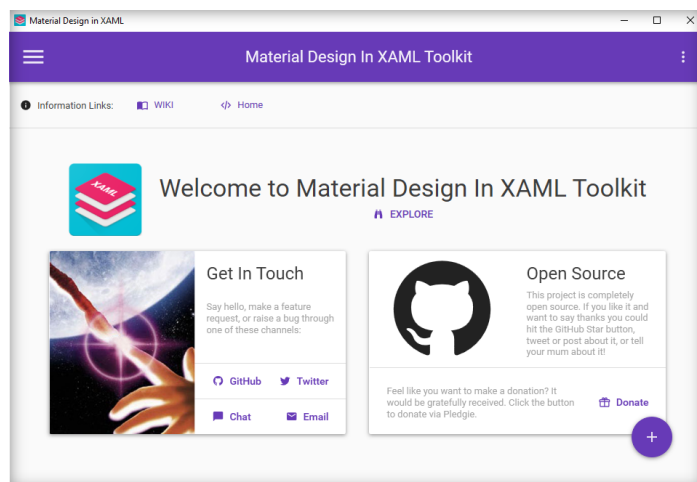
```
<Window . . .
  xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/the
mes"
  TextElement.Foreground="{DynamicResource MaterialDesignBody}"
  TextElement.FontWeight="Regular"
```

```
TextElement.FontSize="13"
TextOptions.TextFormattingMode="Ideal"
TextOptions.TextRenderingMode="Auto"
Background="{DynamicResource MaterialDesignPaper}"
FontFamily="{DynamicResource MaterialDesignFont}">
<Grid>
    <materialDesign:Card Padding="32" Margin="16">
        <TextBlock Style="{DynamicResource
MaterialDesignTitleTextBlock}">My First Material Design App</TextBlock>
    </materialDesign:Card>
</Grid>
</Window>
```

- Như vậy, ta đã hoàn thành việc cài đặt và thêm bộ thư viện Material Design In XAML Toolkit vào Visual Studio.

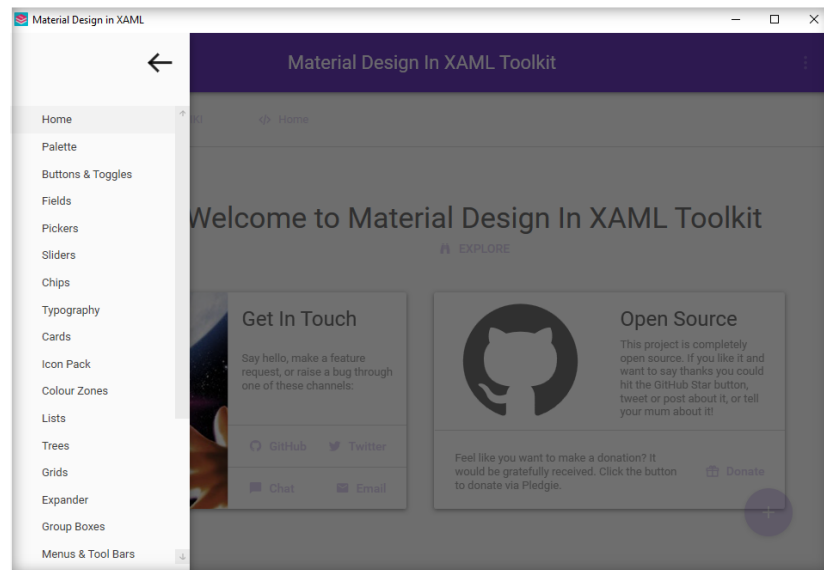
2.2.3.3. Các chức năng chính của bộ công cụ

- Bộ công cụ có giao diện đơn giản, cung cấp nhiều mã nguồn thiết kế như animation, button, slider, image, icon,... với đầy đủ màu sắc và dễ dàng lựa chọn cũng như thêm vào dự án của mình.



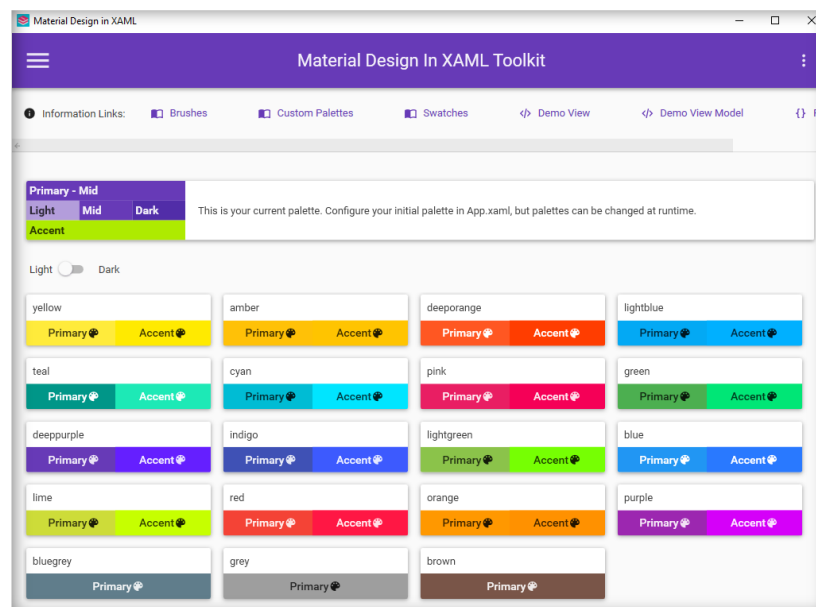
Hình 14. Giao diện chính của Material Design In XAML Toolkit

- Menu chính bao gồm rất nhiều tùy chọn với những thiết kế có sẵn, người dùng dễ dàng chọn lựa và dùng chúng ngay trên ứng dụng WPF của mình.



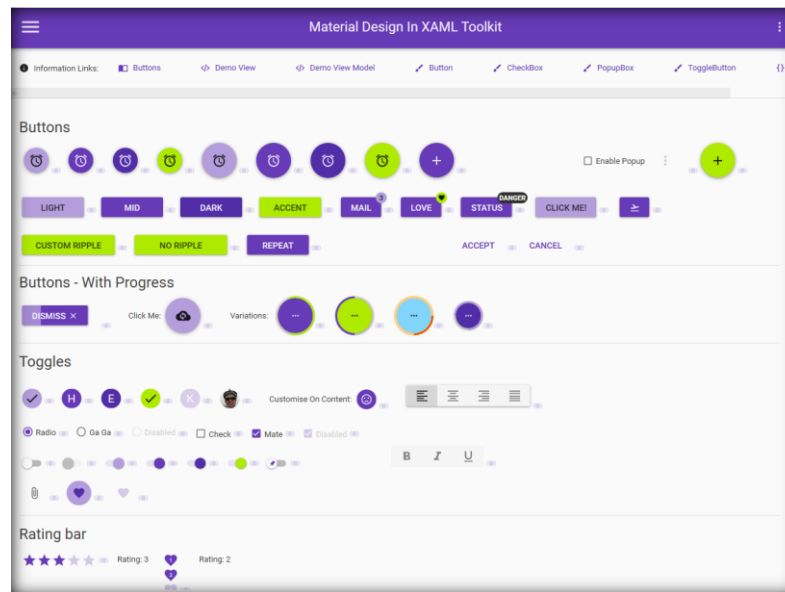
Hình 15. Menu với nhiều tùy chọn

- Ngoài ra, ta còn dễ dàng thay đổi màu sắc hoặc phối màu cho từng button hay các thuộc tính màu sắc cho từng đối tượng cụ thể.



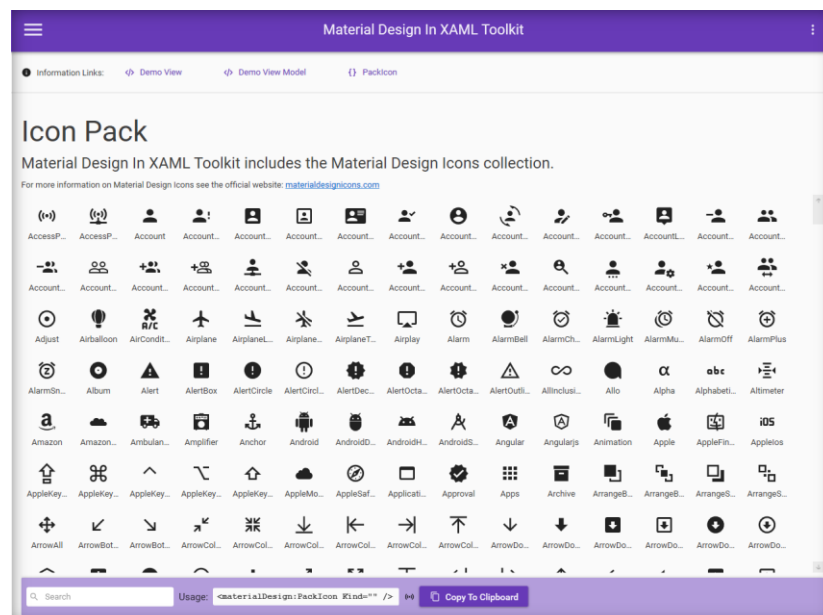
Hình 16. Menu phối màu

- Một số mẫu buttons và toggles được thiết kế sẵn theo phong cách Material Design mà Google quy định và dễ dàng sử dụng chúng.



Hình 17. Menu cho phép chọn và sao chép buttons và toggles được thiết kế sẵn

- Icons cũng được tích hợp sẵn giúp cho thao tác thiết kế đơn giản hơn rất nhiều, thay vì phải tải bộ icons từ trên internet về, người dùng có thể khai báo và sử dụng ngay khi thiết kế.



Hình 18. Menu thư viện icons được tích hợp sẵn

- Kèm theo từng mẫu được thiết kế, ta có thể dễ dàng xem source code và sử dụng ngay.

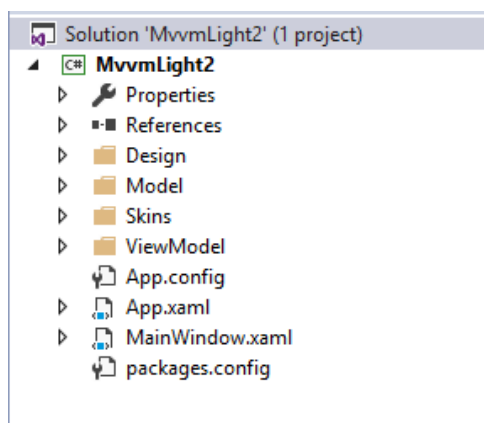


Hình 19. Source code mẫu để sử dụng

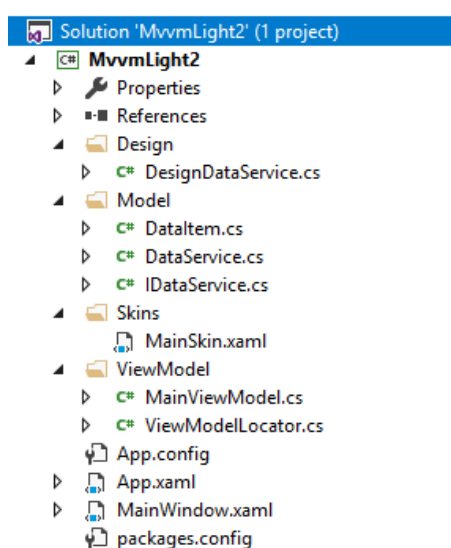
2.2.4. MVVM Light Toolkit

MVVM Light Toolkit là bộ công cụ hỗ trợ người dùng tăng tốc độ tạo và phát triển ứng dụng dựa theo mô hình MVVM trong WPF, Silverlight và Windows Phone.

MVVM Light Toolkit có cấu trúc sau khi được cài đặt như sau:



Hình 20. Cấu trúc MVVM Light



Hình 21. Cấu trúc chi tiết của MVVM Light

```
public class DataItem
{
    public string Title
    {
        get;
        private set;
    }

    public DataItem(string title)
    {
        Title = title;
    }
}
```

Hình 22. Hàm DataItem chứa các dữ liệu sử dụng toàn bộ cục phần mềm

```
public interface IDataService
{
    void GetData(Action<DataItem, Exception> callback);
}
```

Hình 23. Interface của phần mềm

```
public class DesignDataService : IDataService
{
    public void GetData(Action<DataItem, Exception> callback)
    {
        // Use this to create design time data

        var item = new DataItem("Welcome to MVVM Light [design]");
        callback(item, null);
    }
}
```

Hình 24. Hàm gọi giao diện thiết kế mặc định

```
public class DataService : IDataService
{
    public void GetData(Action<DataItem, Exception> callback)
    {
        // Use this to connect to the actual data service

        var item = new DataItem("Welcome to MVVM Light");
        callback(item, null);
    }
}
```

Hình 25. Hàm gọi giao diện người dùng

```
public class ViewModelLocator
{
    static ViewModelLocator ()
    {
        ServiceLocator.SetLocatorProvider (() => SimpleIoc.Default);

        if (ViewModelBase.IsInDesignModeStatic) {
            SimpleIoc.Default.Register<IDataService, Design.DesignDataService> ();
        } else {
            SimpleIoc.Default.Register<IDataService, DataService> ();
        }

        SimpleIoc.Default.Register<MainViewModel> ();
    }

    /// <summary>
    /// Gets the Main property.
    /// </summary>
    [System.Diagnostics.CodeAnalysis.SuppressMessage ("Microsoft.Performance",
        "CA1822:MarkMembersAsStatic",
        Justification = "This non-static member is needed for data binding purposes.")]
    public MainViewModel Main
    {
        get {
            return ServiceLocator.Current.GetInstance<MainViewModel> ();
        }
    }

    /// <summary>
    /// Cleans up all the resources.
    /// </summary>
    public static void Cleanup () { }
}
```

Hình 26. Hàm khởi tạo và gọi Data và ViewModel

```
public class MainViewModel : ViewModelBase
{
    private readonly IDataService _dataService;

    /// <summary>
    /// The <see cref="WelcomeTitle" /> property's name.
    /// </summary>
    public const string WelcomeTitlePropertyName = "WelcomeTitle";
    private string _welcomeTitle = string.Empty;
    /// <summary>
    /// Gets the WelcomeTitle property.
    /// Changes to that property's value raise the PropertyChanged event.
    /// </summary>
    public string WelcomeTitle
    {
        get {
            return _welcomeTitle;
        }
        set {
            Set (ref _welcomeTitle, value);
        }
    }

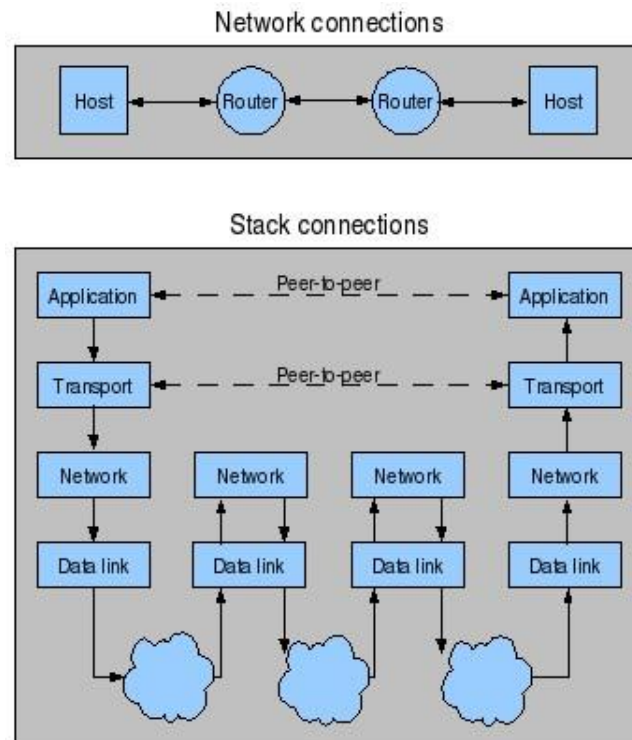
    /// <summary>
    /// Initializes a new instance of the MainViewModel class.
    /// </summary>
    public MainViewModel (IDataService dataService)
    {
        _dataService = dataService;
        _dataService.GetData (
            (item, error) => {
                if (error != null) {
                    // Report error here
                    return;
                }
                WelcomeTitle = item.Title;
            });
    }

    public override void Cleanup()
    {
        // Clean up if needed

        base.Cleanup();
    }
}
```

Hình 27. Hàm ViewModel

2.2.5. TCP/IP



Hình 28. Mô hình TCP/IP và các tầng dữ liệu

- TCP/IP là bộ giao thức cho phép kết nối các hệ thống mạng không đồng nhất với nhau. Ngày nay, TCP/IP được sử dụng rộng rãi trong mạng cục bộ cũng như mạng toàn cầu. TCP/IP được xem như giản lược của mô hình tham chiếu OSI với 4 tầng như sau:
 - + Tầng Liên kết (Datalink Layer):
Tầng liên kết (còn được gọi là tầng liên kết dữ liệu hay tầng giao tiếp mạng) là tầng thấp nhất trong mô hình TCP/IP, bao gồm các thiết bị giao tiếp mạng và các chương trình cung cấp các thông tin cần thiết để có thể hoạt động, truy nhập đường truyền vật lý thông qua các thiết bị giao tiếp mạng đó.
 - + Tầng Mạng (Internet Layer):
Tầng Internet (hay còn gọi là tầng mạng) xử lý quá trình truyền gói tin trên mạng, các giao thức của tầng này bao gồm: IP (Internet Protocol), ICMP (Internet Control Message Protocol), IGMP (Internet Group Message Protocol).
 - + Tầng Giao vận (Transport Layer):

Tầng giao vận phụ trách luồng dữ liệu giữa 2 trạm thực hiện các ứng dụng của tầng trên, tầng này có 2 giao thức chính là TCP và UDP.

TCP cung cấp luồng dữ liệu tin cậy giữa 2 trạm, nó sử dụng các cơ chế như chia nhỏ các gói tin ở tầng trên thành các gói tin có kích thước thích hợp cho tầng mạng bên dưới, báo nhận gói tin, đặt hạn chế thời gian timeout để đảm bảo bên nhận biết được các gói tin đã gửi đi. Do tầng này đã đảm bảo tính tin cậy nên tầng trên sẽ không cần quan tâm đến nữa.

UDP cung cấp một dịch vụ rất đơn giản hơn cho tầng ứng dụng. Nó chỉ gửi dữ liệu từ trạm này tới trạm kia mà không đảm bảo các gói tin đến được đích. Các cơ chế đảm bảo độ tin cậy được thực hiện bởi trên tầng ứng dụng.

+ Tầng Ứng dụng (Application Layer):

Là tầng trên của mô hình TCP/IP bao gồm các tiến trình và các ứng dụng cung cấp cho người sử dụng để truy cập mạng. Có rất nhiều ứng dụng được cung cấp trong tầng này, mà phổ biến là Telnet: sử dụng trong việc truy cập mạng từ xa, FTP, Email,...

2.3. Phần thực hành

2.3.1. Thiết lập Server/Client

2.3.1.1. Thiết lập Server

➤ Class Server.cs

Tạo hàm Run trong class Server được dùng để chờ đợi lắng nghe các Client kết nối vào Server và được chạy theo Multi thread lắng nghe liên tục việc Client kết nối vào Server.

```
private void Run()
{
    while (true)
    {
        try
        {
            //Waiting for incoming connection request
            TcpClient client = _tcpip.AcceptTcpClient();
            //Initializes and starts a TCPServer thread
            //and add it to the list of TCPServer threads
            ServerThread st = new ServerThread(client);

            //Add events
            st.DataReceived += new ServerThread.DelegateDataReceived(OnDataReceived);
            st.ClientDisconnected += new ServerThread.DelegateClientDisconnected(OnClientDisconnected);

            //Further work
            OnClientConnected(st);
        }
        catch (Exception ex)
        {
            //Starting to read
            client.Client.BeginReceive(st.ReadBuffer,
                0, st.ReadBuffer.Length, SocketFlags.None, st.Receive, client.Client);
        }
        catch (Exception ex)
        {
            //Connection faulty
            Console.WriteLine(ex.Message);
        }
        catch (Exception)
        {
            break;
        }
    }
}
```

Hình 29. Hàm Run trong Class Server

- Tạo hàm Start thông qua thư viện Sockets dùng để bắt đầu tạo một Server và thêm vào hàm Run dùng để tạo và bắt đầu luồng khi mà có Client kết nối vào.

```

public void Start(string strIPAddress, int Port)
{
    //Determine endpoint and listener
    _endpoint = new IPEndPoint(IPAddress.Parse(strIPAddress), Port);
    _tcpip = new TcpListener(_endpoint);
    if (_tcpip == null) return;
    try
    {
        _tcpip.Start();
        // Initialize and start the main TCP Server thread
        _ThreadMainServer = new Thread(new ThreadStart(Run));
        _ThreadMainServer.IsBackground = true;
        _ThreadMainServer.Start();
        //State
        this._State = ListenerState.Started;
    }
    catch (Exception ex)
    {
        //break up
        _tcpip.Stop();
        this._State = ListenerState.Error;

        //Throw an exception
        throw ex;
    }
}

```

Hình 30. Hàm Start Server

- Ngoài việc xây dựng hàm Start ngoài ra còn xây dựng hàm Stop hàm được dùng để thực hiện việc dừng server được tạo.

```

public void Stop () {
    try {
        // Stop all TCP Server threads
        for (IEnumerator en = _threads.GetEnumerator (); en.MoveNext ();) {
            //Get next TCP Server thread
            ServerThread st = (ServerThread) en.Current;
            //and stop
            st.Stop ();

            //Send event
            if (ClientDisconnected != null) {
                ClientDisconnected (st, "Connection has ended");
            }
        }

        if (_tcpip != null) {
            //Stop Listener
            _tcpip.Stop ();
            _tcpip.Server.Close ();
        }

        //Note status
        this._State = ListenerState.Stopped;
    } catch (Exception) {
        this._State = ListenerState.Error;
    }
}

```

Hình 31. Hàm Stop trong Server.Class

- Tạo hàm Send và hàm Receive được sử dụng để gửi và nhận các thông tin từ Client.

+ Hàm Send:

```
public void Send (Byte[] data) {  
    try {  
        //If the connection still exists  
        if (this._IsStopped == false) {  
            //Get the stream for writing  
            NetworkStream ns = this._Connection.GetStream ();  
  
            lock (ns) {  
                // Send the coded string to the TCP Server  
                ns.Write (data, 0, data.Length);  
            }  
        }  
    } catch (Exception ex) {  
        //Close connection  
        this._Connection.Close ();  
        //End connection  
        this._IsStopped = true;  
  
        //Send event  
        if (ClientDisconnected != null) {  
            ClientDisconnected (this, ex.Message);  
        }  
  
        //Forward Exception  
        throw ex;  
    }  
}
```

Hình 32. Hàm Send trong Class Server

+ Hàm Receive:

```
public void Receive (IAsyncResult ar) {
    try {
        //If not connected anymore
        if (this._Connection.Client.Connected == false) {
            return;
        }

        if (ar.IsCompleted) {
            //Read
            int bytesRead = _Connection.Client.EndReceive (ar);

            //If data exists
            if (bytesRead > 0) {
                //Detect only read bytes
                Byte[] data = new byte[bytesRead];
                System.Array.Copy (ReadBuffer, 0, data, 0, bytesRead);

                //Send event
                DataReceived (this, data);
                //Continue reading
                _Connection.Client.BeginReceive (ReadBuffer, 0,
                    ReadBuffer.Length, SocketFlags.None,
                    Receive, _Connection.Client);
            } else {
                //connection lost
                HandleDisconnection ("Connection has ended");
            }
        }
    } catch (Exception ex) {
        //connection lost
        HandleDisconnection (ex.Message);
    }
}
```

Hình 33. Hàm Receive trong Class Server

➤ Thiết lập Server Class MainViewModel.cs

- Trong View-Model xây dựng làm ServerStart với kiểu RelayCommand dùng để binding tới nút Button trên View thực hiện việc bắt đầu một Server hoặc Dừng Server lại. Ngoài ra dùng để chạy các hàm khác liên quan đến việc bắt đầu một Server hoặc dừng nó.

```

public RelayCommand ServerStart {
    get {
        return _ServerStart ??
            (_ServerStart = new RelayCommand (
                () => {
                    try {
                        if (IsServerRunning) {
                            StopServer ();
                            StopRecordingFromSounddevice_Server ();
                            StopTimerMixed ();
                        } else {
                            FormToConfig ();
                            StartServer ();
                            StartRecordingFromSounddevice_Server ();
                            StartTimerMixed ();
                        }
                    } catch (Exception ex) {
                        ShowError (Servernotification, ColorServerNotification, ex.Message);
                    }
                }
            ));
    }
}

```

Hình 34. Hàm RelayCommand của Button Server

- Hàm StartServer được xây dựng trong View-Model được thực hiện việc bắt đầu một Server khi người dùng nhấn vào nút button và Server ấy được kiểm tra là chưa được tạo sẽ được thực hiện.

```

private void StartServer () {
    try {
        if (IsServerRunning == false) {
            if (_Config.IPAddressServer.Length > 0 && _Config.PortServer > 0) {
                _Server = new Server ();
                _Server.ClientConnected += new Server.DelegateClientConnected (OnServerClientConnected);
                _Server.ClientDisconnected += new Server.DelegateClientDisconnected (OnServerClientDisconnected);
                _Server.DataReceived += new Server.DelegateDataReceived (OnServerDataReceived);
                _Server.Start (_Config.IPAddressServer, _Config.PortServer);

                //Depending on server status
                if (_Server.State == Server.ListenerState.Started) {
                    ShowStart ();
                } else {
                    ShowStop ();
                }
            }
        }
    } catch (Exception ex) {
        ShowError (Clientnotification, ColorClientnotification, ex.Message);
    }
}

```

Hình 35. Hàm StartServer của Button Server

- Hàm StopServer xây dựng trong View-Model thực hiện chức năng dừng Server lại khi người dùng nhấn vào Button và kiểm được Server đang chạy thì Server sẽ được dừng lại.

```
3 private void StopServer () {  
3     try {  
3         if (IsServerRunning == true) {  
3             DeleteAllServerThreadDatas ();  
3             ListClientIP.Clear ();  
3             _Server.Stop ();  
3             _Server.ClientConnected -= new Server.DelegateClientConnected (OnServerClientConnected);  
3             _Server.ClientDisconnected -= new Server.DelegateClientDisconnected (OnServerClientDisconnected);  
3             _Server.DataReceived -= new Server.DelegateDataReceived (OnServerDataReceived);  
3         }  
3  
3         if (_Server != null) {  
3             if (_Server.State == Server.ListenerState.Started) {  
3                 ShowStart ();  
3             } else {  
3                 ShowStop ();  
3             }  
3         }  
3  
3         _Server = null;  
3     } catch (Exception ex) {  
3         ShowError (Servernotification, ColorServerNotification, ex.Message);  
3     }  
3 }
```

Hình 36. Hàm StopServer cho Button Server

2.3.1.2. Thiết Lập Client

➤ Class Client.cs

- Cũng như thiết lập Server, Client có các hàm như là Connect được dùng để thực hiện việc thiết lập kết nối tới Server.

```
public void Connect () {  
    try {  
        //Possibly. Enable AutoConnect  
        InitTimerAutoConnect ();  
  
        // Create new socket bound to the _Server and _Port  
        Client = new TcpClient (this._Server, this._Port);  
        _NetStream = Client.GetStream ();  
  
        //Beginning to read  
        this.StartReading ();  
  
        //Send event  
        ClientConnected (this, String.Format ("server: {0} port: {1}", this._Server, this._Port));  
    } catch (Exception ex) {  
        //Hand off  
        throw ex;  
    }  
}
```

Hình 37. Hàm Connect trong Class Client

- Hàm Disconnect được dùng để dừng việc kết nối giữa Client và Server.

```
private void disconnect_intern () {
    if (Client != null) {
        Client.Close ();
    }
    if (_NetStream != null) {
        _NetStream.Close ();
    }
}

public void Disconnect () {
    //End connection
    disconnect_intern ();

    //If not already finished
    if (_TimerAutoConnect != null) {
        //Do not reconnect
        _TimerAutoConnect.Dispose ();
        _TimerAutoConnect = null;
    }

    //Send event
    if (this.ClientDisconnected != null) {
        this.ClientDisconnected (this, "Connection terminated");
    }
}
```

Hình 38. Hàm Disconnect trong Class Client

- Tiếp đó là hàm Send được dùng để gửi các tin dưới dạng Byte cho Server.

```
public void Send (Byte[] data) {
    try {
        // Send the coded string to the _server
        _NetStream.Write (data, 0, data.Length);

        //Send event
        if (this.DataSend != null) {
            this.DataSend (this, data);
        }
    } catch (Exception ex) {
        //Exception Send event
        ExceptionAppeared (this, ex);
    }
}
```

Hình 39. Hàm Send trong Class Client

- Và hàm nhận tin nhắn và động tin nhắn và đọc tin nhắn được nhận từ Server gửi trả về Client.

```
private void OnDataReceived (IAsyncResult ar) {
    try {
        NetworkStream myNetworkStream = (NetworkStream) ar.AsyncState;
        if (myNetworkStream.CanRead) {
            int numberOfBytesRead = myNetworkStream.EndRead (ar);
            if (numberOfBytesRead > 0) {
                //Send event
                if (this.DataReceived != null) {
                    Byte[] data = new byte[numberOfBytesRead];
                    System.Array.Copy (_ByteBuffer, 0, data, 0, numberOfBytesRead);
                    this.DataReceived (this, data);
                }
            } else {
                if (this.ClientDisconnected != null) {
                    this.ClientDisconnected (this, "FIN");
                }
                if (_AutoConnect == false) {
                    this.disconnect_intern ();
                } else {
                    this.Disconnect_ButAutoConnect ();
                }
                return;
            }
            myNetworkStream.BeginRead (_ByteBuffer, 0, _ByteBuffer.Length,
                new AsyncCallback (OnDataReceived), myNetworkStream);
        }
    } catch (Exception ex) {
        ExceptionAppeared (this, ex);
    }
}

private void StartReading () {
    try {
        _ByteBuffer = new byte[1024];
        _NetStream.BeginRead (_ByteBuffer, 0, _ByteBuffer.Length,
            new AsyncCallback (OnDataReceived), _NetStream);
    } catch (Exception ex) {
        ExceptionAppeared (this, ex);
    }
}
```

Hình 40. Hàm OnDataReceive trong Class Client

➤ Thiết lập Client trong Class MainViewModel.cs

- Tạo hàm StartClient trong View-Model với kiểu RelayCommand được dùng cho Button trên View thực hiện việc kết nối Client với Server cũng như là ngắt kết nối.

```
public RelayCommand StartClient {  
    get {  
        return _StartClient ??  
            (_StartClient = new RelayCommand (  
                () => {  
                    try {  
                        if (IsClientConnected) {  
                            DisconnectClient ();  
                            StopRecordingFromSounddevice_Client ();  
                        } else {  
                            FormToConfig ();  
                            ConnectClient ();  
                            string Name = Dns.GetHostName ();  
                            _Client.Send (binaryFormat.Serialize (Name));  
                        }  
                    }  
                    //Wait a minute  
                    System.Threading.Thread.Sleep (100);  
                } catch (Exception ex) {  
                    ShowError (Clientnotification, ColorClientnotification, ex.Message);  
                }  
            }));  
    }  
}
```

Hình 41. Hàm RelayCommand trong MainViewModel

- Trong đó hàm ConnectClient được xây dựng khi Button được nhấn và được kiểm tra Client chưa được kết nối thì sẽ thực việc kết nối.

```
private void ConnectClient () {  
    try {  
        if (IsClientConnected == false) {  
            //If input exists  
            if (_Config.IpAddressClient.Length > 0 && _Config.PortClient > 0) {  
                _Client = new Clients (_Config.IpAddressClient, _Config.PortClient);  
                _Client.ClientConnected += new Clients.DelegateConnection (OnClientConnected);  
                _Client.ClientDisconnected += new Clients.DelegateConnection (OnClientDisconnected);  
                _Client.ExceptionAppeared += new Clients.DelegateException (OnClientExceptionAppeared);  
                _Client.DataReceived += new Clients.DelegateDataReceived (OnClientDataReceived);  
                _Client.Connect ();  
            }  
        }  
    } catch (Exception ex) {  
        _Client = null;  
        ShowError (Clientnotification, ColorClientnotification, ex.Message);  
    }  
}
```

Hình 42. Hàm ConnnectClient trong MainViewModel

- Hàm DisconnectClient được dùng khi Button được nhấn và được kiểm tra Client với Server đang được kết nối sẽ thực hiện việc ngắt.

```
private void DisconnectClient () {  
    try {  
        StopRecordingFromSounddevice_Client ();  
  
        if (_Client != null) {  
            //Client break up  
            _Client.Disconnect ();  
            _Client.ClientConnected -= new Clients.DelegateConnection (OnClientConnected);  
            _Client.ClientDisconnected -= new Clients.DelegateConnection (OnClientDisconnected);  
            _Client.ExceptionAppeared -= new Clients.DelegateException (OnClientExceptionAppeared);  
            _Client.DataReceived -= new Clients.DelegateDataReceived (OnClientDataReceived);  
            _Client = null;  
        }  
    } catch (Exception ex) {  
        ShowError (Clientnotification, ColorClientnotification, ex.Message);  
    }  
}
```

Hình 43. Hàm DisconnectClient trong MainViewModel

2.3.2. Truyền nhận Audio thông qua Socket

- Khi bắt đầu việc truyền nhận Audio các việc cần làm cho cả chương trình thiết lập các việc cần thiết:
 - + Xây dựng hàm đóng gói âm thanh dưới dạng Byte để có thể Truyền nhận âm thanh giữa Client Và Server.

```
private WinSound.RTPPacket ToRTPPacket (Byte[] linearData, int bitsPerSample, int channels) {  
    Byte[] data = WinSound.Utills.LinearToMulaw (linearData, bitsPerSample, channels);  
  
    //New Create RTP Packet  
    WinSound.RTPPacket rtp = new WinSound.RTPPacket ();  
  
    //take values  
    rtp.Data = data;  
    rtp.HeaderLength = WinSound.RTPPacket.MinHeaderLength;  
    //Finished  
    return rtp;  
}  
  
private Byte[] ToRTPData (Byte[] data, int bitsPerSample, int channels) {  
    //New Create RTP Packet  
    WinSound.RTPPacket rtp = ToRTPPacket (data, bitsPerSample, channels);  
    //Create RTPHeader in Bytes  
    Byte[] rtpBytes = rtp.ToBytes ();  
    //Finished  
    return rtpBytes;  
}
```

Hình 44. Hàm đóng gói âm thanh

- + Hàm `SetUpComboxes` được xây dựng trong View-Model được dùng để thực hiện việc tìm kiếm thiết bị âm thanh cho Server và Client

```
private void SetUpComboboxesClient () {
    ListOutputSoundDevice.Clear ();
    ListInputSoundDevice.Clear ();
    List<String> playbackNames = WinSound.WinSound.GetPlaybackNames ();
    List<String> recordingNames = WinSound.WinSound.GetRecordingNames ();

    //Output
    foreach (String name in playbackNames.Where (x => x != null)) {
        ListOutputSoundDevice.Add (name);
    }
    //Input
    foreach (String name in recordingNames.Where (x => x != null)) {
        ListInputSoundDevice.Add (name);
    }

    //Output
    if (ListOutputSoundDevice.Count > 0) {
        IndexOutput = 0;
    }
    //Input
    if (ListInputSoundDevice.Count > 0) {
        IndexInput = 0;
    }
}
```

Hình 45. Hiển thị các thiết bị đầu ra/đầu vào

- + Hàm `SetUpJitterBufferClientRecording` được sử dụng cho việc thiết lập nhận bộ đệm cho việc ghi âm của Client.

```
private void SetUpJitterBufferClientRecording () {
    //If available
    if (_JitterBufferClientRecording != null) {
        _JitterBufferClientRecording.DataAvailable -=
            new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferClientDataAvailableRecording);
    }

    //Create new
    _JitterBufferClientRecording = new WinSound.JitterBuffer (null, RecordingJitterBufferCount, 20);
    _JitterBufferClientRecording.DataAvailable +=
        new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferClientDataAvailableRecording);
}
```

Hình 46. Bộ đệm âm thanh của Client

- + Hàm `SetUpJitterBufferClientPlaying` được dùng cho việc thiết lập bộ đệm cho việc bắt đầu phát đoạn âm thanh được truyền đi.

```
private void SetUpJitterBufferClientPlaying () {  
    //If available  
    if (_JitterBufferClientPlaying != null) {  
        _JitterBufferClientPlaying.DataAvailable -=  
            new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferClientDataAvailablePlaying);  
    }  
  
    //Create new  
    _JitterBufferClientPlaying = new WinSound.JitterBuffer (null, _Config.JitterBufferCountClient, 20);  
    _JitterBufferClientPlaying.DataAvailable +=  
        new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferClientDataAvailablePlaying);  
}
```

Hình 47. Thiết lập bộ đệm phát âm thanh

- + Hàm `SetUpJitterBufferServerRecording` cũng có tác dụng làm nhận bộ đệm cho việc ghi âm nhưng bộ đệm này được sử dụng cho Server

```
private void SetUpJitterBufferServerRecording () {  
    //If available  
    if (_JitterBufferServerRecording != null) {  
        _JitterBufferServerRecording.DataAvailable -=  
            new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferServerDataAvailable);  
    }  
  
    //Create new  
    _JitterBufferServerRecording = new WinSound.JitterBuffer (null, RecordingJitterBufferCount, 20);  
    _JitterBufferServerRecording.DataAvailable +=  
        new WinSound.JitterBuffer.DelegateDataAvailable (OnJitterBufferServerDataAvailable);  
}
```

Hình 48. Thiết lập bộ đệm ghi âm thanh

- Đối với Server trong View-Model

- + Tạo hàm `StartRecordingFromSounddevice_Server()` thực hiện việc ghi âm từ Server. Hàm được đặt trong `RelayCommand` được khởi tạo cùng lúc với việc bắt đầu khởi tạo Server

```
private void StartRecordingFromSounddevice_Server () {
    try {
        if (IsRecorderFromSounddeviceStarted_Server == false) {
            int bufferSize = 0;
            if (UseJitterBufferServerRecording) {
                bufferSize = WinSound.Utils.GetBytesPerInterval
                ((uint) _Config.SamplesPerSecondServer, _Config.BitsPerSampleServer,
                _Config.ChannelsServer) * (int) _RecorderFactor;
            } else {
                bufferSize = WinSound.Utils.GetBytesPerInterval
                ((uint) _Config.SamplesPerSecondServer,
                _Config.BitsPerSampleServer, _Config.ChannelsServer);
            }
            //If buffer is correct
            if (bufferSize > 0) {
                //Create a recorder
                _Recorder_Server = new WinSound.Recorder ();
                //Add events
                _Recorder_Server.DataRecorded +=
                new WinSound.Recorder.DelegateDataRecorded (OnDataReceivedFromSoundcard_Server);
                //Start recorder
                if (_Recorder_Server.Start
                (_Config.SoundInputDeviceNameServer,
                _Config.SamplesPerSecondServer, _Config.BitsPerSampleServer,
                _Config.ChannelsServer, _SoundBufferCount, bufferSize)) {

                    MainViewModel.DictionaryMixed[this] = new Queue<List<byte>> ();

                    _JitterBufferServerRecording.Start ();
                }
            }
        }
    } catch (Exception ex) {
        ShowError (Clientnotification, ColorClientnotification, ex.Message);
    }
}
```

Hình 49. Thực hiện ghi âm thanh từ thiết bị đầu vào

- + Hàm StopRecordingFromSounddevice_Server() sẽ thực hiện việc dừng việc ghi âm thanh của Server lại khi mà Server dừng được đóng lại.

```
private void StopRecordingFromSounddevice_Server () {
    try {
        if (IsRecorderFromSounddeviceStarted_Server) {
            //To stop
            _Recorder_Server.Stop ();

            //Remove events
            _Recorder_Server.DataRecorded -= new WinSound.Recorder.DelegateDataRecorded (OnDataReceivedFromSoundcard_Server);
            _Recorder_Server = null;

            //JitterBuffer break up
            _JitterBufferServerRecording.Stop ();
        }
    } catch (Exception ex) {
        ShowError (Clientnotification, ColorClientnotification, ex.Message);
    }
}
```

Hình 50. Thực hiện dừng ghi âm thanh

- + Hàm StartTimerMixed được sử dụng như bộ đếm thời gian gửi các đoạn ghi âm đi tới các Client.

```
private void StartTimerMixed () {  
    if (_TimerMixed == null) {  
        _TimerMixed = new WinSound.EventTimer ();  
        _TimerMixed.TimerTick += new WinSound.EventTimer.DelegateTimerTick (OnTimerSendMixedDataToAllClients);  
        _TimerMixed.Start (20, 0);  
    }  
}
```

Hình 51. Truyền âm thanh đã được mã hoá đến các Client

- + Hàm StopTimerMixed được dùng để ngừng việc đếm thời gian thực hiện khi mà Server được dùng.

```
private void StopTimerMixed () {  
    if (_TimerMixed != null) {  
        _TimerMixed.Stop ();  
        _TimerMixed.TimerTick -= new WinSound.EventTimer.DelegateTimerTick (OnTimerSendMixedDataToAllClients);  
        _TimerMixed = null;  
    }  
}
```

Hình 52. Dừng truyền âm thanh đến các Client

- + Hàm SendConfigurationToClient được dùng để thực hiện việc có cho phép Client kết nối vào Server hay không để bắt đầu việc Kết nối cũng như truyền nhận âm thanh của Client kết nối vào Server.

```
private void SendConfigurationToClient (ThreadData data) {  
    if (MessageBox.Show ("Do you accept the call's : " + ListClientIP[ListClientIP.Count - 1].NameIPConnected + "?",  
        "Call from IP: " + ListClientIP[ListClientIP.Count - 1].IDAddress, MessageBoxButtons.YesNo) == DialogResult.Yes) {  
        // You are Accept  
        Byte[] bytesConfig = _Encoding.GetBytes (String.Format ("Accept:{0}", _Config.SamplesPerSecondServer));  
        data.ServerThread.Send (_PrototolClient.ToBytes (bytesConfig));  
    } else {  
        // You aren't Accept  
        Byte[] bytesConfig = _Encoding.GetBytes (String.Format ("NotAccept:{0}", _Config.SamplesPerSecondServer));  
        data.ServerThread.Send (_PrototolClient.ToBytes (bytesConfig));  
    }  
}
```

Hình 53. Hàm SendConfigurationToClient

- Đối với Client trong View-Model

- + Hàm nhận tin nhắn chấp nhận hay từ chối việc được kết nối tới Server hay không để bắt đầu việc Truyền nhận âm thanh của Client. If từ chối Client sẽ tự ngắt kết nối với Server.

```
private void OnClientConfigReceived (Object sender, Byte[] data) {  
    try {  
        String msg = _Encoding.GetString (data);  
        if (msg.Length > 0) {  
            //parsing  
            String[] values = msg.Split (':');  
            String cmd = values[0];  
  
            //Depending on the command  
            switch (cmd.ToUpper ()) {  
                case "ACCEPT":  
                    int samplePerSecond = Convert.ToInt32 (values[1]);  
                    _Config.SamplesPerSecondClient = samplePerSecond;  
                    StartPlayingToSounddevice_Client ();  
                    StartRecordingFromSounddevice_Client ();  
                    break;  
                case "NOTACCEPT":  
                    DisconnectClient ();  
                    break;  
            }  
        }  
    } catch (Exception ex) {  
        Console.WriteLine (ex.Message);  
    }  
}
```

Hình 54. Hàm OnClientConfigReceived

- + Hàm StartPlayingToSounddevice_Client có chức năng bắt đầu việc phát âm thanh được gửi từ Server đến cho Client thông qua Thiết bị âm thanh. Và chức được thực hiện khi Server chấp nhận việc kết nối của Client và Client nhận được gói dữ liệu âm thanh.

```
private void StartPlayingToSounddevice_Client () {  
    //If desired  
    if (IsPlayingToSoundDeviceWanted) {  
        //Start JitterBuffer  
        if (_JitterBufferClientPlaying != null) {  
            SetUpJitterBufferClientPlaying ();  
            _JitterBufferClientPlaying.Start ();  
        }  
  
        if (_PlayerClient == null) {  
            _PlayerClient = new WinSound.Player ();  
            _PlayerClient.Open (_Config.SoundOutputDeviceNameClient,  
                                _Config.SamplesPerSecondClient, _Config.BitsPerSampleClient,  
                                _Config.ChannelsClient, (int) _Config.JitterBufferCountClient);  
        }  
    }  
}
```

Hình 55. Thực hiện phát âm thanh qua thiết bị đầu ra

- + Hàm StopPlayingToSounddevice_Client có chức năng dừng việc phát âm thanh nhận được ghi Client gắn kết nối với Server.

```
private void StopPlayingToSounddevice_Client () {  
    if (_PlayerClient != null) {  
        _PlayerClient.Close ();  
        _PlayerClient = null;  
    }  
  
    //JitterBuffer break up  
    if (_JitterBufferClientPlaying != null) {  
        _JitterBufferClientPlaying.Stop ();  
    }  
}
```

Hình 56. Dừng phát âm thanh qua thiết bị đầu ra đến Server

- + Hàm StartRecordingFromSounddevice_Client cũng tương tự như Server hàm có chức năng thực hiện việc bắt đầu ghi âm từ thiết bị âm thanh của Client. Chức năng được thực hiện khi Server chấp nhận việc kết nối của Client.


```
private void StartRecordingFromSounddevice_Client () {
    try {
        if (IsRecorderFromSounddeviceStarted_Client == false) {
            int bufferSize = 0;
            if (UseJitterBufferClientRecording) {
                bufferSize = WinSound.Utils.GetBytesPerInterval
                ((uint) _Config.SamplesPerSecondClient, _Config.BitsPerSampleClient,
                _Config.ChannelsClient) * (int) _RecorderFactor;
            } else {
                bufferSize = WinSound.Utils.GetBytesPerInterval
                ((uint) _Config.SamplesPerSecondClient, _Config.BitsPerSampleClient, _Config.ChannelsClient);
            }

            if (bufferSize > 0) {
                //Create a recorder
                _Recorder_Client = new WinSound.Recorder ();

                //Add events
                _Recorder_Client.DataRecorded += new WinSound.Recorder.DelegateDataRecorded (OnDataReceivedFromSoundcard_Client);

                //Start recorder
                if (_Recorder_Client.Start (_Config.SoundInputDeviceNameClient,
                _Config.SamplesPerSecondClient, _Config.BitsPerSampleClient,
                _Config.ChannelsClient, _SoundBufferCount, bufferSize)) {
                    if (UseJitterBufferClientRecording) {
                        _JitterBufferClientRecording.Start ();
                    }
                }
            }
        }
    } catch (Exception ex) {
        ShowError (Clientnotification, ColorClientnotification, ex.Message);
    }
}
```

Hình 57. Thực hiện ghi âm thanh từ thiết bị đầu vào của Client

- + Hàm StopRecordingFromSounddevice_Client hàm có chức năng thực hiện việc dừng ghi âm từ thiết bị âm thanh của Client. Chức năng được thực hiện khi việc kết nối của Client được ngừng.

```
private void StopRecordingFromSounddevice_Client () {
    try {
        if (IsRecorderFromSounddeviceStarted_Client) {
            //To stop
            _Recorder_Client.Stop ();

            //Remove events
            _Recorder_Client.DataRecorded -= new WinSound.Recorder.DelegateDataRecorded (OnDataReceivedFromSoundcard_Client);
            _Recorder_Client = null;

            //If jitterbuffer
            if (UseJitterBufferClientRecording) {
                _JitterBufferClientRecording.Stop ();
            }
        }
    } catch (Exception ex) {
        ShowError (Clientnotification, ColorClientnotification, ex.Message);
    }
}
```

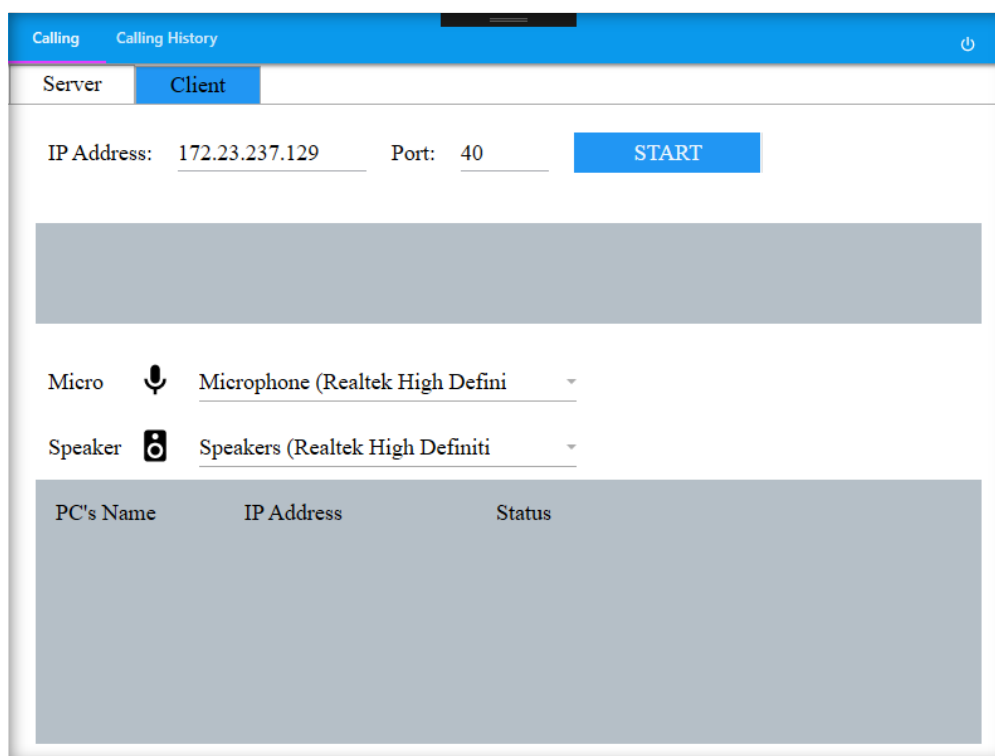
Hình 58. Thực hiện việc dừng ghi âm thanh từ thiết bị đầu vào của Client

2.3.3. Ứng dụng VoIP Call**2.3.2.1. Tổng quan về VoIP Call**

- VoIP (Voice over Internet Protocol) là một công nghệ cho phép truyền thoại sử dụng giao thức mạng IP, trên cơ sở hạ tầng sẵn có của Internet và có thể sử dụng nhiều loại mạng khác nhau: LAN, WAN,... Với chi phí thấp hơn nhiều so với các phương thức truyền thông truyền thống, VoIP đang được các doanh nghiệp đầu tư nhiều và được ứng dụng rộng rãi trong môi trường doanh nghiệp và cá nhân.
- Về dự án VoIP Call, đây là ứng dụng cho phép thực hiện truyền thoại giữa các máy tính sử dụng hệ điều hành Windows trong mạng LAN. Thông qua giao thức UDP và Socket. Các máy tính trong mạng LAN có thể giao tiếp, truyền và nhận âm thanh với nhau theo thời gian thực.
- Ứng dụng được viết bằng WPF theo mô hình MVVM giúp tối ưu hoá code và có thể tái sử dụng code, ngoài ra việc bổ sung hoặc chỉnh sửa chức năng đơn giản hơn so với Winform.
- Tuy nhiên, ứng dụng chỉ có thể chạy trên các máy tính sử dụng hệ điều hành Windows Vista hoặc Windows Server 2008 trở lên.

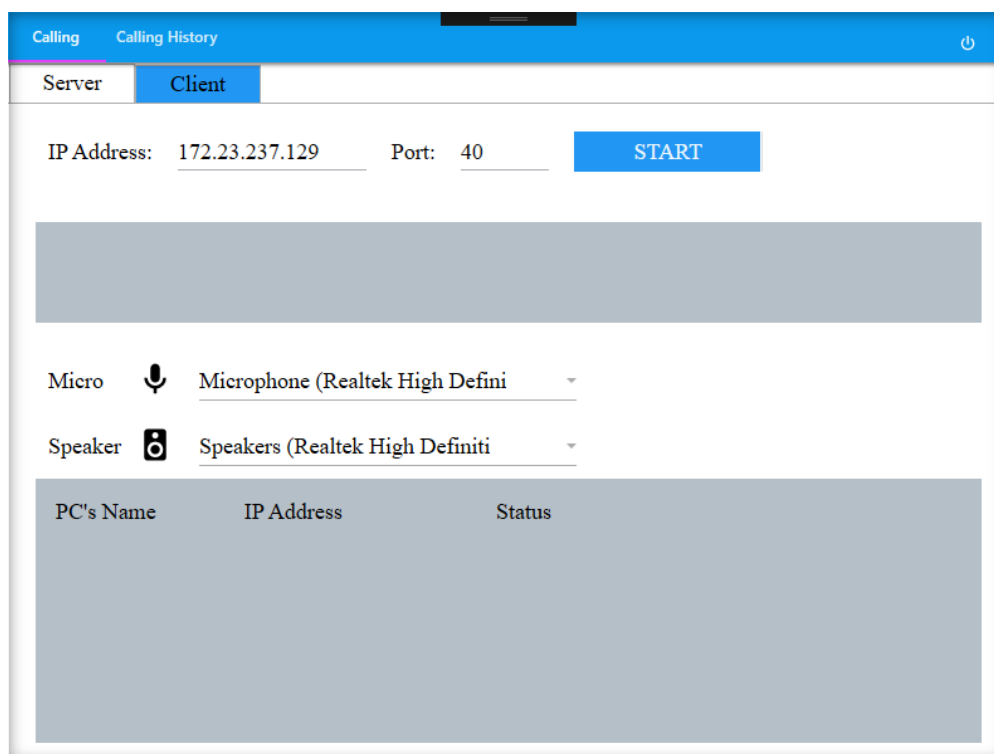
2.3.2.2. Từng chức năng của VoIP Call

- Giao diện chính của ứng dụng gồm 3 phần chính: Main Menu, Sub-Menu và nội dung. Trong mỗi phần để có phần nội dung riêng và từng chức năng riêng để đáp ứng đủ



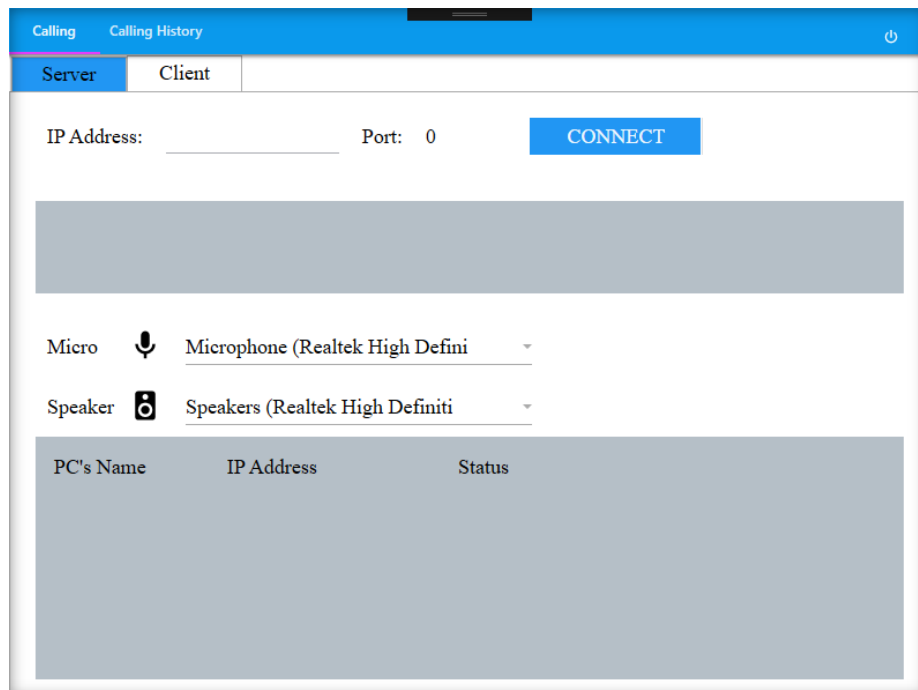
Hình 59. Giao diện chính

- Giao diện Server cho phép người dùng thao tác thiết lập server để client có thể kết nối đến, bao gồm địa chỉ IP, cổng port và chọn các thiết bị đầu vào/đầu ra.



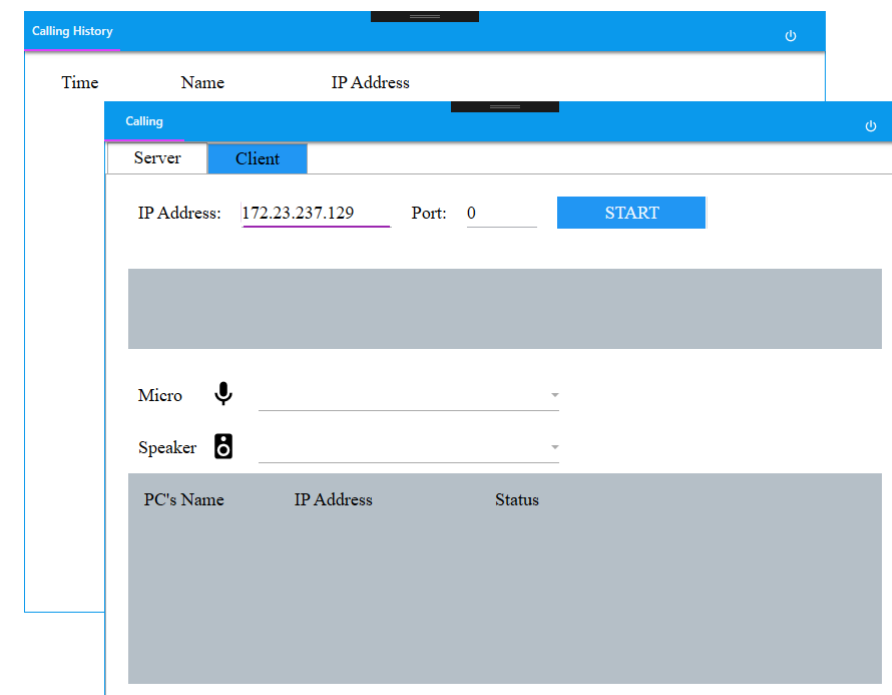
Hình 60. Giao diện Server

- Giao diện Client cho phép user kết nối đến Server thông qua địa chỉ IP và cổng Port của Server



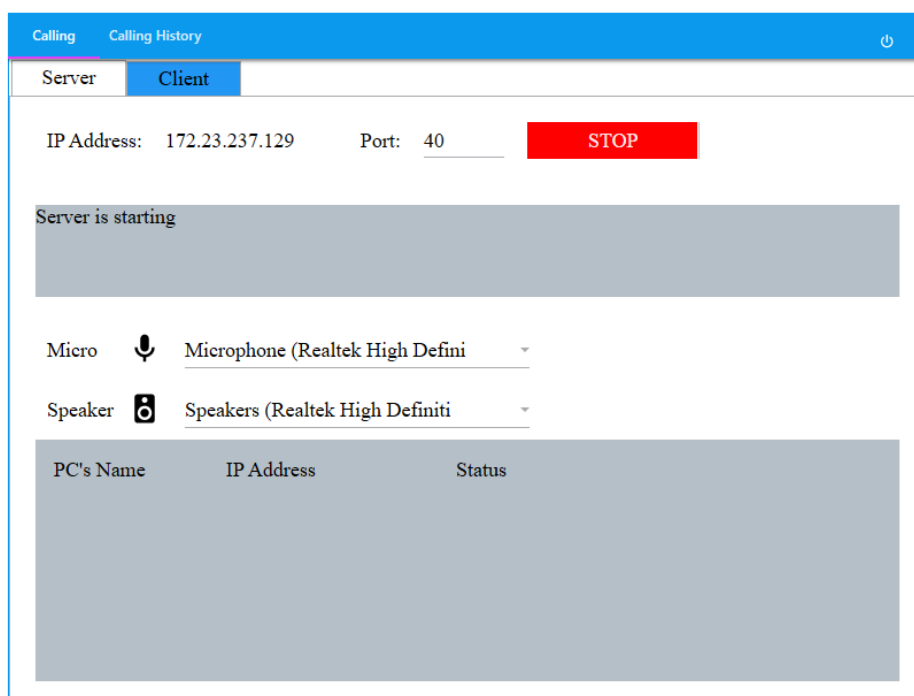
Hình 61. Giao diện Client

- Giao diện multitab cho phép kéo thả các tab để dàng kéo thả từng tab linh động



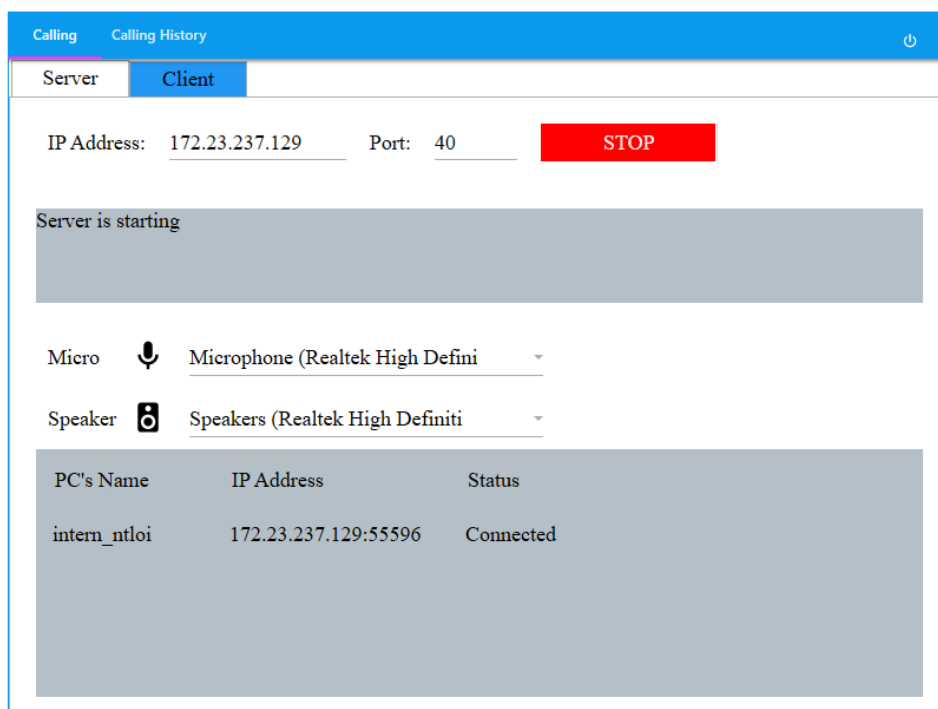
Hình 62. Multitab trong ứng dụng

- Giao diện khi khởi tạo Server.



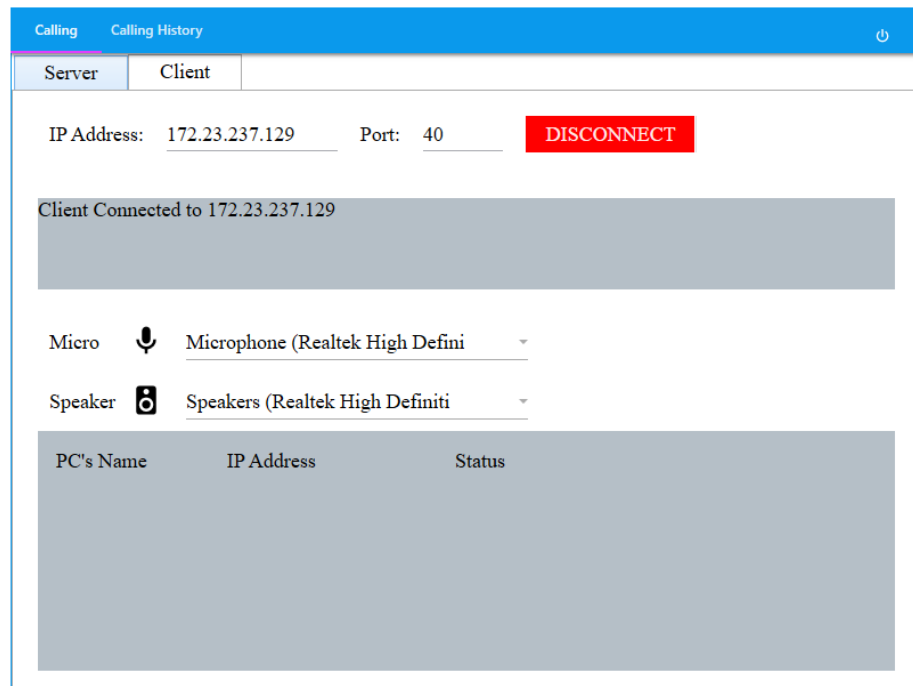
Hình 63. Server khi được khởi tạo

- Giao diện sau khi Client kết nối đến Server, thông tin từng users đã kết nối sẽ được hiển thị trên ListView



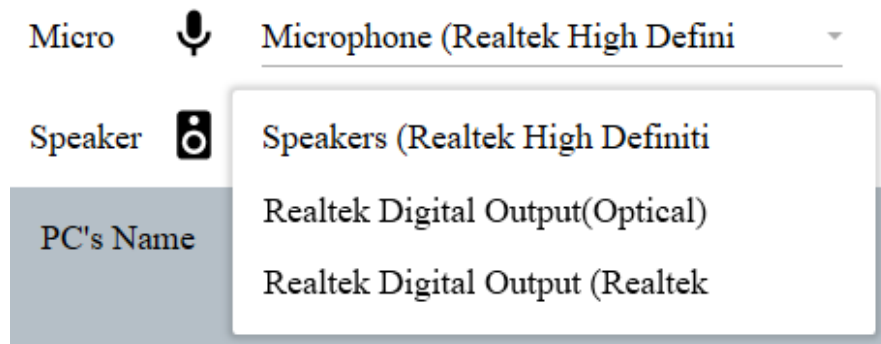
Hình 64. Server hiển thị users sau khi được kết nối

- Giao diện Client sau khi kết nối đến Server, thông báo sẽ được hiển thị trên TextBlock



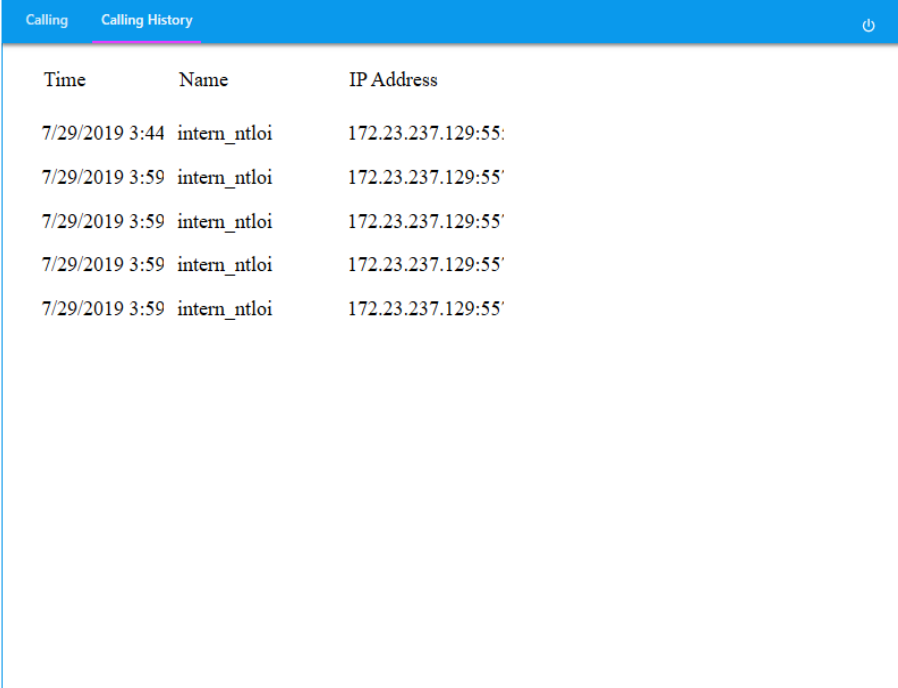
Hình 65. Client sau khi kết nối đến Server

- Tại mục Micro và Speaker, người dùng có thể dễ dàng chọn các thiết bị đầu ra/đầu vào thông qua combobox



Hình 66. Giao diện cho phép chọn OutPut và InPut từ thiết bị

- Calling History sẽ hiển thị các cuộc gọi gần đây nhất, dữ liệu được hiển thị trên ListView



Time	Name	IP Address
7/29/2019 3:44	intern_ntloi	172.23.237.129:55
7/29/2019 3:59	intern_ntloi	172.23.237.129:55
7/29/2019 3:59	intern_ntloi	172.23.237.129:55
7/29/2019 3:59	intern_ntloi	172.23.237.129:55
7/29/2019 3:59	intern_ntloi	172.23.237.129:55

Hình 67. Lịch sử cuộc gọi

- Khi có cuộc gọi từ user bất kỳ, Server sẽ hiển thị bảng thông báo cuộc gọi đến, người dùng hoàn toàn có thể từ chối hoặc chấp nhận cuộc gọi



Hình 68. Thông báo khi có cuộc gọi đến

- Ứng dụng VoIP đã có thể thực hiện được việc khởi tạo Server, Client và thực hiện được các chức năng cơ bản như nghe, gọi và đàm thoại theo thời gian thực, đồng thời cũng có thể lưu trữ được thông tin kết nối và lịch sử cuộc gọi.

CHƯƠNG 3. ĐỀ XUẤT CÁC GIẢI PHÁP CẢI THIỆN CHẤT LƯỢNG GIẢNG DẠY Ở NHÀ TRƯỜNG

Môi trường thực tế tại công ty là môi trường hoàn toàn khác so với môi trường giao dục tại nhà trường, do đó, sinh viên khi thực tập sẽ ngỡ ngàng và sẽ gặp một số khó khăn nhất định trong thực tập. Em xin được đề xuất một số giải pháp để cải thiện chất lượng giảng dạy ở nhà trường như sau:

- **Về kỹ năng mềm:**
 - Thường xuyên tổ chức các lớp rèn luyện kỹ năng làm việc nhóm;
 - Trang bị cho sinh viên kỹ năng tìm đọc tài liệu bằng Tiếng Anh;
 - Nên đưa Tiếng Anh vào việc giảng dạy và học tập ở một số môn học nhất định;
 - Tổ chức các buổi sinh hoạt Tiếng Anh.
- **Về kiến thức chuyên ngành:**
 - Chú trọng hơn về hướng đối tượng trong việc giảng dạy;
 - Đưa công nghệ mới vào chương trình đào tạo;
 - Đưa các công cụ lưu trữ mã nguồn như: Git, Github, Gitlab, Bitbucket,... vào giảng dạy cũng như điều kiện bắt buộc để lưu trữ mã nguồn khi thực hiện đồ án môn học;
 - Các môn học như thiết kế web, lập trình web,... cần tạo điều kiện cho Sinh viên deploy website thực tế lên host và trở tên miền thực tế;
 - Thường xuyên tổ chức các buổi seminar, workshop về các công nghệ, công cụ,... cần thiết cho môi trường doanh nghiệp như: Linux, Docker, Angular, React, React Native,... và các ngôn ngữ, công cụ mã nguồn mở khác;
 - Định hướng cho sinh viên nắm bắt hiểu rõ front-end, back-end, devOps,... từ đó giúp sinh viên có hướng đi tốt hơn trong suốt thời gian học tập tại trường.

CHƯƠNG 4. KẾT LUẬT VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

4.1. Kết luận

Qua thời gian thực tập thực tế tại Công ty TNHH Giải pháp Phần mềm Tường Minh, em đã học hỏi và trao đổi thêm nhiều kỹ năng, kiến thức thực tế; đồng thời được củng cố kiến thức đã học trên ghế nhà trường. Qua dự án thực tế, em đã thực hiện được một số yêu cầu nhất định; tuy nhiên, thời gian thực tập chưa được nhiều nên dự án còn gặp phải nhiều khó khăn cần khắc phục.

➤ Về dự án VoIP Call:

- Ưu điểm:

- + Ứng dụng VoIP có thể nghe gọi theo thời gian thực thông qua TCP/IP;
- + Giao diện dễ tương tác, sử dụng;
- + Thiết kế dạng multitab, dễ dàng kéo thả như giao diện metro;

- Nhược điểm:

- + Giao diện ứng dụng còn đơn giản, chưa có nhiều tính năng;
- + Độ trễ trong truyền âm thanh còn cao;
- + Độ nhiễu âm thanh còn cao;

➤ Về bản thân:

- Ưu điểm:

- + Củng cố thêm kiến thức về C#, Network,...
- + Học được thêm công nghệ và công cụ mới: WPF, mô hình MVVM, MVVM Light Toolkit, thư viện NAudio,...
- + Củng cố khả năng giao tiếp trong công sở;
- + Được đào tạo về các quy trình phát triển phần mềm đang được vận dụng tại công ty;
- + Củng cố và rèn luyện thêm kỹ năng nghe, nói, đọc, viết Tiếng Anh;
- + Khả năng tự học, tự nghiên cứu qua tài liệu được nâng cao.

- Nhược điểm:

- + Khả năng tìm lỗi và sửa lỗi còn nhiều khó khăn dẫn đến mất nhiều thời gian;
- + Khả năng vận dụng Tiếng Anh trong giao tiếp còn chưa cao.

4.2. Định hướng phát triển

Để làm cho dự án có tính thiết thực hơn đối với nhu cầu đại đa số người dùng, ứng dụng cần được bổ sung, phát triển thêm một số tính năng:

- Bổ sung và chỉnh sửa giao diện thân thiện hơn cho người dùng;
- Bổ sung tính năng SIP Call để thực hiện cuộc gọi ra bên ngoài mạng cục bộ;
- Chuyển đổi sang UWP thiết kế giao diện chuyên nghiệp và đa nền tảng.

TÀI LIỆU THAM KHẢO

- **Tài liệu Tiếng Anh:**

1. Richard Blum (2002). *C# Network Programming*. Sybex

- **Tài liệu từ Internet:**

1. Ebook – Lập trình mạng với C#

<https://tinhte.vn/threads/ebook-lap-trinh-mang-voi-c-full-117-trang-tieng-viet.2157487/> - 26/07/2019

2. Khoá học “Lập trình WPF cơ bản”

<https://howkteam.vn/course/lap-trinh-wpf-co-ban-30> - 28/06/2019

3. Khoá học “Lập trình phần mềm Quản lý kho WPF – MVVM”

<https://www.howkteam.vn/course/lap-trinh-phan-mem-quan-ly-kho-wpf--mvvm-42> - 28/06/2019

4. Khoá học “MVVM Light Toolkit Fundamentals”

<https://app.pluralsight.com/library/courses/mvvm-light-toolkit-fundamentals/> - 03/07/2019

5. Khoá học “Audio Programming with NAudio”

<https://app.pluralsight.com/library/courses/audio-programming-naudio/> - 10/07/2019

- **Phần mềm và thư viện được sử dụng:**

1. Microsoft Visual Studio 2017 Community
2. Microsoft Visual Code
3. MVVM Light Toolkit
4. Material Design In XAML Toolkit
5. NAudio Library