

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG



ĐỒ ÁN CUỐI KỲ
MÔN: KỸ THUẬT LẬP TRÌNH NÂNG CAO

**Đề tài: UNIVERSAL WINDOWS PLATFORM (UWP) –
ỨNG DỤNG TỔ HỢP TIỆN ÍCH CONVENIENCE APP**

Lớp: 21DienTu

GIẢNG VIÊN HƯỚNG DẪN: HUỲNH QUỐC THỊNH

NHÓM SINH VIÊN THỰC HIỆN: NHÓM 3KG

STT	HỌ VÀ TÊN	MSSV
1	DANH CHÍ HIỀN (NHÓM TRƯỞNG)	21200287
2	TRẦN HỮU HẠNH	21200286
3	LÊ QUANG HUY	21200293

TP. Hồ Chí Minh – Năm 2024

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN CUỐI KỲ

MÔN: KỸ THUẬT LẬP TRÌNH NÂNG CAO

Đề tài: UNIVERSAL WINDOWS PLATFORM (UWP) –
ỨNG DỤNG TỔ HỢP TIỆN ÍCH CONVENIENCE APP

Lớp: 21DienTu

GIẢNG VIÊN HƯỚNG DẪN: HUỲNH QUỐC THỊNH

NHÓM SINH VIÊN THỰC HIỆN: NHÓM 3KG

STT	HỌ VÀ TÊN	MSSV
1	DANH CHÍ HIỀN (NHÓM TRƯỞNG)	21200287
2	TRẦN HỮU HẠNH	21200286
3	LÊ QUANG HUY	21200293

TP. Hồ Chí Minh – Năm 2024

LỜI CẢM ƠN

Lời đầu tiên, nhóm 3KG xin gửi lời cảm ơn sâu sắc và chân thành nhất đến giảng viên hướng dẫn, Th.S Huỳnh Quốc Thịnh. Thầy đã hỗ trợ và hướng dẫn nhóm trong suốt quá trình thực hiện đồ án cuối kỳ môn Kỹ thuật lập trình nâng cao với đề tài "UNIVERSAL WINDOWS PLATFORM (UWP) – ỨNG DỤNG TỐ HỢP TIỆN ÍCH CONVENIENCE APP".

Thầy đã không ngần ngại chia sẻ kiến thức, kinh nghiệm và thời gian quý báu của mình để giúp nhóm chúng em hoàn thiện đồ án. Những lời khuyên và góp ý của thầy đã giúp nhóm chúng em rất nhiều trong việc nắm bắt và áp dụng lý thuyết vào thực tế. Chúng em cũng xin gửi lời cảm ơn đến các bạn trong nhóm đã cùng nhau nỗ lực, cống hiến và hợp tác để hoàn thành đồ án này. Sự cố gắng và đam mê của mỗi thành viên đã tạo nên sự thành công của đồ án. Mặc dù còn nhiều sai sót trong quá trình thực hiện, nhưng đó là kết quả của sự nỗ lực không ngừng từ các thành viên.

Nhóm rất mong nhận được những góp ý từ thầy, nhằm giúp chúng em hoàn thiện vốn kiến thức và rút ra những kinh nghiệm quý báu để tiếp tục hoàn thành tốt những dự án sắp tới.

Xin chân thành cảm ơn thầy!

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

....., ngày ... tháng ... năm 2024

Người nhận xét

(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN	3
NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN	1
MỤC LỤC.....	2
DANH MỤC CÁC HÌNH.....	4
DANH MỤC CÁC BẢNG.....	6
TÓM TẮT	7
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI	8
1.1. LÝ DO CHỌN ĐỀ TÀI	8
1.2. MỤC TIÊU ĐỀ TÀI	9
1.3. PHẠM VI ĐỀ TÀI.....	10
1.4. MÔ TẢ YÊU CẦU CHỨC NĂNG	11
CHƯƠNG 2: THỰC HIỆN ĐỀ TÀI	14
PHẦN I: THIẾT LẬP CHƯƠNG TRÌNH.....	14
2.1. CODE CHƯƠNG TRÌNH	14
2.1.1. Điều hướng và truyền tham số qua các trang	14
2.1.2. Home Page	20
2.1.3. Calendar Page	22
2.1.4. Play Page.....	35
2.1.5. Calculator Page	43
2.1.6. AddFriend Page	54
2.1.7. ContactInfo Page.....	57
2.2. CHẠY CODE CHƯƠNG TRÌNH.....	59
2.2.1. GIAO DIỆN CHƯƠNG TRÌNH	59
2.2.2. HƯỚNG DẪN SỬ DỤNG CHI TIẾT CÁC MỤC GIAO DIỆN.....	64
PHẦN II: ĐỘ HOÀN THIỆN CỦA CHƯƠNG TRÌNH	80
2.3. UƯ ĐIỂM	80
2.4. HẠN CHẾ	81
2.5. ĐỀ XUẤT HƯỚNG PHÁT TRIỂN	83
CHƯƠNG 3: TỔNG KẾT ĐỀ TÀI.....	85
3.1. ĐÁNH GIÁ ĐỀ TÀI.....	85

3.2. HOẠT ĐỘNG CỦA NHÓM VÀ CÁC THÀNH VIÊN.....	88
3.3. PHÂN CÔNG, ĐÁNH GIÁ CÁC THÀNH VIÊN	89
TÀI LIỆU THAM KHẢO.....	93

DANH MỤC CÁC HÌNH

<i>Hình 2. Giao diện chính Home (Trang chủ)</i>	60
<i>Hình 3. Giao diện tối ưu phần hiển thị nội dung</i>	61
<i>Hình 4. Giao diện Calendar (Lịch)</i>	61
<i>Hình 5. Giao diện Play (Trò chơi)</i>	62
<i>Hình 6. Giao diện Calculator (Máy tính)</i>	62
<i>Hình 7. Giao diện Add Friend (Thêm bạn bè)</i>	63
<i>Hình 8. Giao diện Contact Information (Thông tin liên lạc)</i>	63
<i>Hình 9. Khi nhấn vào "HOME" ở thanh điều hướng</i>	64
<i>Hình 10. Khi nhấn vào "Calendar" ở thanh điều hướng</i>	65
<i>Hình 11. Nhấn vào nút "..." và chọn “New” ở góc phải màn hình để "Thêm lịch hẹn"</i>	65
<i>Hình 12. Thêm lịch hẹn</i>	66
<i>Hình 13. Điều chỉnh ngày hẹn và “ấn mũi tên” để chọn</i>	66
<i>Hình 14. Điều chỉnh giờ hẹn và "ấn mũi tên" để chọn</i>	67
<i>Hình 15. Hoàn tất mô tả cuộc hẹn và ấn "Save" để lưu</i>	68
<i>Hình 16. Cuộc hẹn hoàn tất và bạn chỉ cần đợi thông báo nhắc nhở đến hẹn....</i>	68
<i>Hình 17. Bạn có thể thêm nhiều cuộc hẹn bằng các thao tác như trên</i>	69
<i>Hình 18. Nhấn vào "Edit" để chỉnh sửa lịch hẹn</i>	69
<i>Hình 19. Điều chỉnh nội dung cuộc hẹn, nhớ ấn “Save” để lưu lại</i>	70
<i>Hình 20. Cuộc hẹn được điều chỉnh đã được cập nhật.....</i>	70
<i>Hình 21. Nhấn "Delete" để xóa cuộc hẹn</i>	71
<i>Hình 22. Lịch hẹn đã được xóa</i>	71

<i>Hình 23. Khi nhấn vào "Play" ở thanh điều hướng và nhấn "Start" để bắt đầu chơi</i>	72
<i>Hình 24. Bấm "Start" và tiến hành chơi theo luật chơi, chương trình sẽ tính thời gian chơi của bạn</i>	73
<i>Hình 25. Kết thúc trò chơi khi bạn đã chọn đúng hết, quan sát thời gian hoàn thành, nhấn "Play again?" nếu muốn cải thiện thời gian chơi</i>	73
<i>Hình 26. Hoàn thành chơi lại trò chơi và thời gian tốt nhất được cập nhật</i>	74
<i>Hình 27. Tiến hành "Play again?" nhưng thời gian mới không tốt nên không được cập nhật</i>	74
<i>Hình 28. Khi nhấn "Calculator" ở thanh điều hướng</i>	75
<i>Hình 29. Nhập vào giá trị cần tính toán, chọn phép tính muốn thực hiện và nhấn "OK" xem kết quả</i>	75
<i>Hình 30. Kết quả khi nhấn "OK"</i>	76
<i>Hình 31. Chọn sang phép trừ nhấn "OK" để xem kết quả</i>	76
<i>Hình 32. Chọn sang phép nhân nhấn "OK" để xem kết quả</i>	77
<i>Hình 33. Chọn sang phép chia nhấn "OK" để xem kết quả</i>	77
<i>Hình 34. Nhấn "Add Friend" ở thanh điều hướng</i>	78
<i>Hình 35. Nhập thông tin vào các ô và nhấn "Add"</i>	78
<i>Hình 36. Thông tin sẽ được chuyển đến trang "Contact Information"</i>	79
<i>Hình 37. Khi nhấn "Contact Information" từ thanh điều hướng</i>	79
<i>Hình 38. Nhấn vào mũi tên góc phải màn hình để điều hướng quay lại</i>	80
<i>Hình 39. Hoạt động nhóm trên MS Teams</i>	88

DANH MỤC CÁC BẢNG

Table 2. Bảng phân công - đánh giá thành viên nhóm 3KG.....

Table 3. Rubric đánh giá

TÓM TẮT

Đồ án cuối kỳ môn học Kỹ thuật lập trình nâng cao của nhóm 3KG, dưới sự hướng dẫn của giảng viên Th.S Huỳnh Quốc Thịnh, tập trung vào việc thiết kế và phát triển một ứng dụng Universal Windows Platform (UWP) trên Visual Studio 2022. Mục tiêu của chúng em là tạo ra một ứng dụng đa chức năng, tiện ích và dễ sử dụng, phục vụ cho nhu cầu hàng ngày của người dùng - ConvenienceApp.

Chúng em đã thành công trong việc phát triển một ứng dụng với nhiều tính năng hữu ích. Ứng dụng này bao gồm các chức năng như Đặt lịch hẹn, giúp người dùng quản lý thời gian và sắp xếp các cuộc hẹn một cách hiệu quả. Trò chơi giải trí giúp người dùng thư giãn sau những giờ làm việc căng thẳng. Máy tính được tích hợp trong ứng dụng hỗ trợ thực hiện các phép tính toán học cơ bản, đáp ứng nhu cầu tính toán hàng ngày. Ngoài ra, tính năng Lưu trữ thông tin người dùng cho phép quản lý dữ liệu cá nhân một cách an toàn và tiện lợi.

Giao diện của ứng dụng được thiết kế trực quan và dễ sử dụng, với các trang chính như HomePage, CalendarPage, PlayPage, CalculatorPage, AddFriendPage và ContactInformationPage. Việc sử dụng NavigationView giúp người dùng dễ dàng điều hướng giữa các trang và trải nghiệm ứng dụng một cách mượt mà.

Đồ án này không chỉ giúp chúng em củng cố kiến thức đã học mà còn phát triển kỹ năng lập trình thực tiễn. Chúng em đã học được cách tích hợp nhiều chức năng vào một ứng dụng duy nhất, từ việc thiết kế giao diện đến lập trình các chức năng phức tạp. Qua quá trình này, chúng em hiểu rõ hơn về việc phát triển phần mềm và những thách thức thực tế trong lĩnh vực công nghệ thông tin.

Chúng em rất tự hào về sản phẩm cuối cùng và hy vọng rằng ứng dụng này sẽ mang lại nhiều tiện ích cho người dùng. Đồng thời, chúng em cũng hy vọng rằng đồ án này sẽ là nguồn cảm hứng cho các bạn sinh viên khác trong việc học tập và ứng dụng kiến thức vào thực tiễn.

Trân trọng, nhóm 3KG.

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. LÝ DO CHỌN ĐỀ TÀI

Xu hướng công nghệ hiện đại: Universal Windows Platform (UWP) là một nền tảng phát triển ứng dụng hiện đại, cho phép xây dựng các ứng dụng có thể chạy trên nhiều thiết bị Windows khác nhau từ PC, laptop đến các thiết bị di động. Việc nắm bắt và áp dụng UWP trong đồ án giúp chúng em cập nhật với xu hướng công nghệ hiện đại, nâng cao kỹ năng lập trình và khả năng thích ứng với các công nghệ mới.

Tính ứng dụng cao: Một ứng dụng tích hợp nhiều chức năng như đặt lịch hẹn, trò chơi, máy tính và lưu trữ thông tin người dùng đáp ứng nhu cầu thực tế của nhiều người dùng. Trong cuộc sống hiện đại, việc có một ứng dụng đa năng giúp quản lý thời gian, giải trí và thực hiện các tác vụ cơ bản hàng ngày trở nên rất cần thiết. Đề tài này không chỉ mang tính học thuật mà còn có giá trị thực tiễn cao, có thể ứng dụng rộng rãi trong đời sống.

Phát triển kỹ năng lập trình toàn diện: Phát triển một ứng dụng đa chức năng đòi hỏi sự kết hợp nhiều kỹ năng lập trình khác nhau, từ thiết kế giao diện người dùng đến xử lý logic và quản lý dữ liệu. Việc thực hiện đồ án này giúp chúng em rèn luyện và phát triển toàn diện các kỹ năng lập trình, đồng thời làm quen với quy trình phát triển phần mềm chuyên nghiệp.

Thúc đẩy sáng tạo và giải quyết vấn đề: Đề tài này tạo điều kiện cho chúng em thể hiện sự sáng tạo trong việc thiết kế giao diện và trải nghiệm người dùng. Đồng thời, chúng em cũng phải đổi mới với nhiều thách thức kỹ thuật và tìm cách giải quyết các vấn đề phức tạp, từ đó nâng cao khả năng tư duy logic và kỹ năng giải quyết vấn đề.

Định hướng nghề nghiệp: Việc hoàn thành đồ án này không chỉ giúp chúng em có một sản phẩm cụ thể để trình bày trong hồ sơ xin việc mà còn giúp định hướng nghề nghiệp trong lĩnh vực phát triển ứng dụng và phần mềm. Kinh nghiệm thực tiễn từ đồ án sẽ là nền tảng vững chắc cho sự nghiệp của chúng em sau này.

Với những lý do trên, chúng em tin rằng việc chọn đề tài phát triển ứng dụng UWP tích hợp nhiều chức năng không chỉ phù hợp với yêu cầu của môn học mà còn mang lại nhiều lợi ích về kiến thức, kỹ năng và định hướng nghề nghiệp trong tương lai.

1.2. MỤC TIÊU ĐỀ TÀI

Phát triển một ứng dụng đa chức năng: Xây dựng một ứng dụng tích hợp nhiều chức năng hữu ích, bao gồm đặt lịch hẹn, trò chơi giải trí, máy tính và lưu trữ thông tin người dùng, đáp ứng nhu cầu hàng ngày của người dùng.

Thiết kế giao diện người dùng trực quan và thân thiện: Tạo ra các giao diện người dùng trực quan, dễ sử dụng và hấp dẫn, giúp người dùng dễ dàng tương tác với ứng dụng. Sử dụng NavigationView để điều hướng mượt mà giữa các trang như HomePage, CalendarPage, PlayPage, CalculatorPage, AddFriendPage và ContactInfoPage.

Đảm bảo tính ổn định và hiệu suất của ứng dụng: Đảm bảo rằng ứng dụng hoạt động ổn định, không có lỗi và có hiệu suất tốt trên các thiết bị chạy Windows. Tối ưu hóa mã nguồn và quy trình xử lý để ứng dụng chạy mượt mà và đáp ứng nhanh chóng các yêu cầu từ người dùng.

Bảo mật thông tin người dùng: Thiết kế và triển khai các biện pháp bảo mật để bảo vệ thông tin cá nhân của người dùng. Đảm bảo rằng dữ liệu được lưu trữ và truyền tải một cách an toàn.

Học tập và áp dụng các công nghệ mới: Thông qua việc phát triển ứng dụng UWP, chúng em muốn nâng cao hiểu biết và kỹ năng sử dụng các công cụ và công nghệ hiện đại như Visual Studio 2022, XAML và các thư viện liên quan đến UWP.

Phát triển kỹ năng làm việc nhóm: Rèn luyện kỹ năng làm việc nhóm, phân chia công việc hợp lý, hợp tác hiệu quả và giải quyết các vấn đề phát sinh trong quá trình phát triển dự án.

Đáp ứng yêu cầu môn học và tạo ra sản phẩm có giá trị thực tiễn: Hoàn thành đồ án theo đúng yêu cầu của môn học Kỹ thuật lập trình nâng cao, đồng thời tạo ra một sản phẩm có giá trị thực tiễn, có thể được sử dụng và đánh giá cao trong cuộc sống hàng ngày.

Bằng cách đặt ra các mục tiêu này, chúng em hy vọng sẽ không chỉ đạt được kết quả cao trong môn học mà còn tạo ra một ứng dụng thực tiễn, hữu ích và có giá trị cho người dùng.

1.3. PHẠM VI ĐỀ TÀI

Phát triển các chức năng chính của ứng dụng:

- **Đặt lịch hẹn:** Tạo giao diện cho phép người dùng thêm, chỉnh sửa, xóa và xem các lịch hẹn. Cung cấp tính năng nhắc nhở để người dùng không bỏ lỡ các cuộc hẹn quan trọng.
- **Trò chơi giải trí:** Tích hợp một hoặc nhiều trò chơi đơn giản, giúp người dùng thư giãn và giải trí.
- **Máy tính:** Phát triển một công cụ máy tính hỗ trợ các phép tính toán học cơ bản như cộng, trừ, nhân, chia.
- **Lưu trữ thông tin người dùng:** Tạo giao diện và chức năng cho phép người dùng lưu trữ và quản lý thông tin cá nhân, bao gồm tên, địa chỉ, số điện thoại và các thông tin liên hệ khác.

Thiết kế giao diện người dùng (UI):

- Sử dụng XAML để thiết kế các giao diện người dùng trực quan, thân thiện và dễ sử dụng.

- Sử dụng NavigationView để điều hướng giữa các trang chính: HomePage, CalendarPage, PlayPage, CalculatorPage, AddFriendPage và ContactInfoPage.

Đảm bảo tính ổn định và hiệu suất: Tối ưu hóa mã nguồn để đảm bảo ứng dụng hoạt động mượt mà trên các thiết bị chạy Windows. Thực hiện kiểm thử để phát hiện và khắc phục các lỗi tiềm ẩn, đảm bảo ứng dụng hoạt động ổn định.

Bảo mật thông tin người dùng: Áp dụng các biện pháp bảo mật cơ bản để bảo vệ dữ liệu người dùng, đảm bảo rằng thông tin cá nhân được lưu trữ và truyền tải một cách an toàn.

Thời gian thực hiện: Đề tài được thực hiện trong khoảng thời gian từ ngày 16/05/2024 đến 28/06/2024, đảm bảo hoàn thành đúng tiến độ và đáp ứng yêu cầu của môn học.

Đối tượng sử dụng: Ứng dụng hướng đến đối tượng người dùng phổ thông, những người có nhu cầu quản lý thời gian, giải trí và thực hiện các phép tính toán học cơ bản hàng ngày.

Phạm vi đề tài được xác định rõ ràng nhằm đảm bảo rằng các chức năng chính của ứng dụng được phát triển và hoàn thiện đúng theo yêu cầu. Đồng thời, việc giới hạn phạm vi giúp chúng em tập trung vào việc tạo ra một sản phẩm chất lượng, hoàn thiện và có giá trị thực tiễn.

1.4. MÔ TẢ YÊU CẦU CHỨC NĂNG

1. Chức năng Đặt lịch hẹn

- Thêm lịch hẹn:

+ Người dùng có thể tạo mới một lịch hẹn bằng cách nhập thông tin chi tiết như tiêu đề, mô tả, ngày giờ bắt đầu và kết thúc.

+ Ứng dụng sẽ lưu trữ và hiển thị lịch hẹn trong một danh sách hoặc trên lịch.

- Chính sửa lịch hẹn:

+ Người dùng có thể chọn một lịch hẹn từ danh sách để chỉnh sửa thông tin.

+ Các thay đổi sẽ được lưu lại và cập nhật trong danh sách lịch hẹn.

- Xóa lịch hẹn: Người dùng có thể xóa một lịch hẹn đã chọn từ danh sách.

- Nhắc nhở lịch hẹn: Ứng dụng sẽ gửi thông báo nhắc nhở trước thời gian diễn ra lịch hẹn dựa trên thiết lập của người dùng.

2. Chức năng Trò chơi Match Game

- Thiết kế trò chơi: Trò chơi Match Game sẽ bao gồm một lưới gồm 16 ô (4 hàng x 4 cột) chứa các Emoiji.

- Luật chơi:

+ Người chơi click chuột vào 2 Emoiji.

+ Nếu 2 Emoiji được nhấp liên tục trùng nhau thì sẽ ẩn 2 TextBlock chứa hai hình đó.

+ Nếu không trùng nhau sẽ không được xóa bỏ..

- Bộ đếm thời gian và hiển thị điểm:

+ Trò chơi sẽ có một bộ đếm thời gian để người chơi theo dõi thời gian chơi.

+ Hiển thị thời gian tốt nhất (ngắn nhất) mà người chơi từng đạt được để tạo cảm hứng cạnh tranh.

3. Chức năng Máy tính cơ bản:

- Cung cấp giao diện máy tính hỗ trợ các phép tính toán học cơ bản như cộng, trừ, nhân, chia.
- Máy tính có giao diện trực quan, dễ sử dụng, tương tự như một máy tính cầm tay thông thường.

4. Chức năng Lưu trữ thông tin người dùng

- Thêm thông tin liên hệ: Người dùng có thể thêm mới thông tin liên hệ bao gồm tên, số điện thoại, email và địa chỉ.
- Chính sửa thông tin liên hệ: Người dùng có thể chọn một thông tin liên hệ để chỉnh sửa các chi tiết.
- Xóa thông tin liên hệ: Người dùng có thể xóa thông tin liên hệ đã chọn khỏi danh sách.
- Xem thông tin liên hệ: Thông tin liên hệ được hiển thị trong một danh sách, cho phép người dùng dễ dàng truy cập và quản lý.

5. Giao diện người dùng (UI)

- Thiết kế giao diện thân thiện.
- NavigationView:
 - + Sử dụng NavigationView để điều hướng giữa các trang chính của ứng dụng: HomePage, CalendarPage, PlayPage, CalculatorPage, AddFriendPage và ContactInfoPage.
 - + Giao diện người dùng thân thiện, dễ sử dụng và đồng nhất giữa các trang.
- Thiết kế giao diện: Các giao diện được thiết kế trực quan, dễ hiểu, giúp người dùng dễ dàng thực hiện các thao tác và trải nghiệm ứng dụng một cách tốt nhất.

6. Tính bảo mật

- Bảo mật dữ liệu.
- Áp dụng các biện pháp bảo mật để bảo vệ dữ liệu cá nhân của người dùng.
- Đảm bảo rằng thông tin liên hệ và lịch hẹn được lưu trữ một cách an toàn.

7. Tính ổn định và hiệu suất

- Thực hiện kiểm thử ứng dụng để đảm bảo rằng các chức năng hoạt động đúng, không có lỗi và hiệu suất tốt.

- Tối ưu hóa mã nguồn để đảm bảo ứng dụng chạy mượt mà trên các thiết bị Windows.

Với các yêu cầu chức năng trên, chúng em mong muốn phát triển một ứng dụng UWP toàn diện, hữu ích và đáp ứng tốt nhu cầu của người dùng.

CHƯƠNG 2: THỰC HIỆN ĐỀ TÀI

PHẦN I: THIẾT LẬP CHƯƠNG TRÌNH

2.1. CODE CHƯƠNG TRÌNH

2.1.1. Điều hướng và truyền tham số qua các trang

Đọc file MainPage.xaml và đoạn code XAML trong thẻ <Grid>

```
<Grid>
    <NavigationView x:Name="MyNavigationView"
ItemInvoked="MyNavigationView_ItemInvoked" IsBackEnabled="True"
BackRequested="MyNavigationView_BackRequested">
        <NavigationView.MenuItems>
            <NavigationViewItem Icon="Home" Content="Home"
x:Name="HomeItem"/>
            <NavigationViewItem Icon="Calendar" Content="Calendar"
x:Name="CalendarItem"/>
            <NavigationViewItem Icon="Play" Content="Play"
x:Name="PlayItem"/>
            <NavigationViewItem Icon="Calculator" Content="Calculator"
x:Name="CalculatorItem"/>
            <NavigationViewItem Icon="AddFriend" Content="Add Friend"
x:Name="AddFriendItem"/>
            <NavigationViewItem Icon="ContactInfo" Content="Contact
Information"
x:Name="ContactInfoItem"/>
        </NavigationView.MenuItems>
        <NavigationView.Content>
            <Frame x:Name="ContentFrame"/>
        </NavigationView.Content>
    </NavigationView>
</Grid>
```

Đoạn chương trình trên được viết bằng XAML code - sử dụng trong các ứng dụng Windows UWP (Universal Windows Platform). Chương trình này tạo ra một giao diện người dùng với điều hướng thông qua NavigationView. Dưới đây là cách hoạt động chi tiết của đoạn chương trình này:

Grid: Là phần tử bố trí cơ bản để chứa các thành phần khác. Trong trường hợp này, Grid chứa một NavigationView.

NavigationView: Là thành phần điều hướng chính của ứng dụng. Nó cung cấp giao diện người dùng để điều hướng giữa các trang khác nhau. Các thuộc tính và sự kiện chính được khai báo bao gồm:

- x:Name="MyNavigationView": Đặt tên cho NavigationView để có thể tham chiếu trong mã C#.

- ItemInvoked="MyNavigationView_ItemInvoked": Gắn sự kiện ItemInvoked, được gọi khi một mục trong danh sách điều hướng được chọn.
- IsBackEnabled="True": Cho phép hiển thị nút "Back" (quay lại).
- BackRequested="MyNavigationView_BackRequested": Gắn sự kiện BackRequested, được gọi khi người dùng nhấn nút "Back".

NavigationView.MenuItems: Chứa các mục điều hướng (menu items):

- NavigationViewItem: Đại diện cho từng mục trong menu. Các mục này có biểu tượng và nội dung được xác định qua thuộc tính Icon và Content.
- Các mục điều hướng bao gồm: Home, Calendar, Play, Calculator, Add Friend, Contact Information, mỗi mục được đặt tên qua thuộc tính x:Name.

NavigationView.Content: Chứa nội dung hiển thị trong NavigationView.

- Frame x:Name="ContentFrame": Là khung chứa để hiển thị nội dung tương ứng với mục điều hướng được chọn.

Hoạt động:

- Khi ứng dụng chạy, NavigationView sẽ hiển thị một danh sách các mục điều hướng ở phía bên trái (hoặc vị trí mặc định tùy thuộc vào cấu hình NavigationView).
- Khi người dùng chọn một mục điều hướng (ví dụ: Home, Calendar, Play, v.v.), sự kiện ItemInvoked sẽ được kích hoạt và gọi đến hàm xử lý sự kiện MyNavigationView_ItemInvoked trong mã C#.
- Khi người dùng nhấn nút "Back", sự kiện BackRequested sẽ được kích hoạt và gọi đến hàm xử lý sự kiện MyNavigationView_BackRequested.
- Nội dung tương ứng với mỗi mục điều hướng sẽ được hiển thị trong Frame (được tham chiếu bởi ContentFrame).

- Viết logic để xử lý các sự kiện ItemInvoked và BackRequested.
- Hàm MyNavigationView_ItemInvoked sẽ điều hướng đến các trang khác nhau tùy thuộc vào mục điều hướng được chọn.
- Hàm MyNavigationView_BackRequested sẽ điều hướng quay lại trang trước đó nếu có thể.

Thay đổi hàm dựng MainPage trong MainPage.xaml.cs:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// The Blank Page item template is documented at
// https://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace Convenience_App
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to
    /// within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
            ContentFrame.Navigate(typeof(HomePage)); //Điều hướng đến
Page mặc định của Frame
        }

        private void
MyNavigationView_ItemInvoked(Windows.UI.Xaml.Controls.NavigationView
sender, NavigationViewItemInvokedEventArgs args)
        {
            FrameNavigationOptions navigationOptions = new
FrameNavigationOptions();
            //Tạo đối tượng Frame để chuyển hướng
            navigationOptions.TransitionInfoOverride =
args.RecommendedNavigationTransitionInfo; //Chọn hiệu ứng chuyển trang
thích hợp
        }
    }
}

```

```

        var selectedItem = MyNavigationView.SelectedItem; //Kiểm
tra Item được chọn
        if (selectedItem == HomeItem)
        {
            ContentFrame.Navigate(typeof(HomePage)); //Chuyển
hướng sang HomePage
        }
        else if (selectedItem == CalendarItem)
        {
            ContentFrame.Navigate(typeof(CalendarPage));
        }
        else if (selectedItem == PlayItem)
        {
            ContentFrame.Navigate(typeof(PlayPage));
        }
        else if (selectedItem == CalculatorItem)
        {
            ContentFrame.Navigate(typeof(CalculatorPage));
        }
        else if (selectedItem == AddFriendItem)
        {
            ContentFrame.Navigate(typeof(AddFriendPage));
        }
        else if (selectedItem == ContactInfoItem)
        {
            ContentFrame.Navigate(typeof(ContactInfoPage));
        }
    }

    private void
MyNavigationView_BackRequested(Windows.UI.Xaml.Controls.NavigationView
sender, NavigationViewBackRequestedEventArgs args)
{
    if (ContentFrame.CanGoBack)
    {
        ContentFrame.GoBack();
    }
}
}

```

Đoạn chương trình này là phần mở rộng của code XAML đã được mô tả ở trên. Nó cung cấp các chức năng cần thiết để điều hướng giữa các trang trong ứng dụng UWP dựa trên sự tương tác của người dùng với các mục điều hướng trong NavigationView. Đoạn chương trình này sẽ giúp điều khiển việc điều hướng giữa các trang trong ứng dụng UWP một cách dễ dàng và hiệu quả. Dưới đây là giải thích chi tiết về cách đoạn mã vận hành:

Khởi tạo và cấu trúc chương trình

Namespace và thư viện:

- Đoạn mã bắt đầu bằng việc khai báo các `using` để bao gồm các thư viện cần thiết cho ứng dụng UWP, bao gồm các thư viện để quản lý giao diện người dùng, điều khiển và điều hướng.

Namespace Convenience_App:

- Đoạn mã thuộc về namespace Convenience_App, giúp phân biệt các lớp và thành phần khác nhau trong ứng dụng.

Class MainPage:

- Đây là lớp MainPage, được định nghĩa là `sealed`, nghĩa là không thể kế thừa lớp này.
- Lớp này kế thừa từ lớp Page, đại diện cho một trang trong ứng dụng.

Hàm khởi tạo MainPage:

```
public MainPage()
{
    this.InitializeComponent();
    ContentFrame.Navigate(typeof(HomePage)); //Điều hướng đến Page mặc định
    của Frame
}
```

- Hàm khởi tạo MainPage được gọi khi trang được tạo.
- `this.InitializeComponent()` ; khởi tạo các thành phần giao diện của trang.
- `ContentFrame.Navigate(typeof(HomePage))` ; điều hướng đến HomePage khi ứng dụng bắt đầu, đặt HomePage làm trang mặc định.

Phương thức xử lý sự kiện ItemInvoked:

```
private void
MyNavigationView_ItemInvoked(Windows.UI.Xaml.Controls.NavigationView sender,
NavigationViewItemInvokedEventArgs args)
{
    FrameNavigationOptions navigationOptions = new FrameNavigationOptions();
    //Tạo đối tượng Frame để chuyển hướng
    navigationOptions.TransitionInfoOverride =
    args.RecommendedNavigationTransitionInfo; //Chọn hiệu ứng chuyển trang thích hợp
```

```

    var selectedItem = MyNavigationView.SelectedItem; //Kiểm tra Item được
chọn
    if (selectedItem == HomeItem)
    {
        ContentFrame.Navigate(typeof(HomePage)); //Chuyển hướng sang HomePage
    }
    else if (selectedItem == CalendarItem)
    {
        ContentFrame.Navigate(typeof(CalendarPage));
    }
    else if (selectedItem == PlayItem)
    {
        ContentFrame.Navigate(typeof(PlayPage));
    }
    else if (selectedItem == CalculatorItem)
    {
        ContentFrame.Navigate(typeof(CalculatorPage));
    }
    else if (selectedItem == AddFriendItem)
    {
        ContentFrame.Navigate(typeof(AddFriendPage));
    }
    else if (selectedItem == ContactInfoItem)
    {
        ContentFrame.Navigate(typeof(ContactInfoPage));
    }
}

```

- Phương thức MyNavigationView_ItemInvoked xử lý sự kiện khi một mục điều hướng trong NavigationView được chọn.
- FrameNavigationOptions navigationOptions = new FrameNavigationOptions(); tạo một đối tượng FrameNavigationOptions để cấu hình các tùy chọn điều hướng, bao gồm hiệu ứng chuyển tiếp trang.
- var selectedItem = MyNavigationView.SelectedItem; lấy mục được chọn.
- Sử dụng cấu trúc if-else để kiểm tra mục được chọn và điều hướng đến trang tương ứng (HomePage, CalendarPage, PlayPage, CalculatorPage, AddFriendPage, ContactInfoPage).

Phương thức xử lý sự kiện BackRequested:

```

private void
MyNavigationView_BackRequested(Windows.UI.Xaml.Controls.NavigationView
sender, NavigationViewBackRequestedEventArgs args)
{
    if (ContentFrame.CanGoBack)
    {
        ContentFrame.GoBack();
    }
}

```

```
    }  
}
```

- Phương thức MyNavigationView_BackRequested xử lý sự kiện khi người dùng nhấn nút "Back".
- if (ContentFrame.CanGoBack) kiểm tra nếu ContentFrame có thể quay lại trang trước đó.
- Nếu có thể, gọi ContentFrame.GoBack(); để quay lại trang trước đó.

Tóm lại: Đoạn chương trình thiết lập một hệ thống điều hướng cơ bản cho ứng dụng UWP sử dụng NavigationView và Frame. Nó bao gồm:

- Khởi tạo trang: Đặt HomePage làm trang mặc định khi ứng dụng khởi động.
- Điều hướng giữa các trang: Xử lý sự kiện khi người dùng chọn một mục điều hướng trong NavigationView, điều hướng đến trang tương ứng.
- Quay lại trang trước đó: Xử lý sự kiện khi người dùng nhấn nút "Back", quay lại trang trước đó nếu có thể.

2.1.2. Home Page

Thực hiện điều hướng qua lại giữa các trang khi nhấn vào thanh điều hướng. Cài đặt HomePage là trang mặc định của Frame.

HomePage.xaml - Đoạn code XAML định nghĩa giao diện cho HomePage trong ứng dụng:

```
<Grid>  
    <Image Source="/Assets/NenHCMUSchung.png" />  
</Grid>
```

Trong Grid, có một thành phần Image:

- Source="/Assets/NenHCMUSchung.png": Đặt đường dẫn đến ảnh nền được hiển thị trong HomePage.
- Lưu ý: Ta add thêm 1 hình ảnh có tên là NenHCMUSchung.png vào thư mục Assets của project.

`HomePage.xaml.cs` - Đoạn code này là phần code phía sau (code-behind) liên kết với giao diện XAML của `HomePage`. Nó thực hiện việc khởi tạo và thiết lập ban đầu cho `HomePage`.

```
namespace Convenience_App
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a
    /// Frame.
    /// </summary>
    public sealed partial class HomePage : Page
    {
        public HomePage()
        {
            this.InitializeComponent();
        }
    }
}
```

Class `HomePage`:

- Lớp `HomePage` được định nghĩa là `sealed`, nghĩa là không thể kế thừa lớp này.
- Lớp này kế thừa từ lớp `Page`, đại diện cho một trang trong ứng dụng.

Constructor `HomePage`:

```
public HomePage()
{
    this.InitializeComponent();
}
```

- Hàm khởi tạo `HomePage` được gọi khi trang được tạo.
- `this.InitializeComponent()`; khởi tạo các thành phần giao diện của trang, liên kết mã phía sau với giao diện XAML.

Tóm lại: Đoạn code XAML và đoạn code C# này định nghĩa và triển khai `HomePage` trong ứng dụng UWP. Hai phần chương trình này kết hợp với nhau để tạo ra trang `HomePage` với giao diện đơn giản, hiển thị một ảnh nền. Giao diện và mã phía sau hoạt động cùng nhau để tạo ra trải nghiệm người dùng mượt mà và tương tác trong ứng dụng:

- `HomePage.xaml`: Định nghĩa giao diện của trang, bao gồm một `Grid` chứa một `Image`. Ảnh nền được đặt từ đường dẫn `/Assets/NenHCMUSchung.png` và nền của trang được thiết lập theo chủ đề của ứng dụng.

- `HomePage.xaml.cs`: Triển khai đoạn chương trình phía sau, liên kết với giao diện XAML, thực hiện việc khởi tạo và thiết lập ban đầu cho `HomePage`.

2.1.3. Calendar Page

Đoạn code XAML định nghĩa giao diện người dùng cho trang `CalendarPage`.

```
<Grid Margin="50">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <CalendarDatePicker Grid.Row="0" Name="Picker"
HorizontalAlignment="Stretch"
PlaceholderText="Display Appointments From"
DateChanged="Picker_DateChanged"/>
    <ListBox Grid.Row="1" Name="Display" Loaded="Display_Loaded">
        <ListBox.ItemContainerStyle>
            <Style TargetType="ListBoxItem">
                <Setter Property="HorizontalContentAlignment"
Value="Stretch"/>
            </Style>
        </ListBox.ItemContainerStyle>
        <ListBox.ItemTemplate>
            <DataTemplate>
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="40*"/>
                        <ColumnDefinition Width="40*"/>
                        <ColumnDefinition Width="Auto"/>
                    </Grid.ColumnDefinitions>
                    <Grid Padding="5" Grid.Column="0"
Background="{ThemeResource AccentButtonBackground}">
                        <TextBlock Text="{Binding When}">
                            <TextBlock.Foreground>
                                <TextBlockResource Key="AccentButtonForeground"/>
                            </TextBlock.Foreground>
                        </TextBlock>
                    <Grid Padding="5" Grid.Column="1"
Background="{ThemeResource AccentButtonForeground}">
                        <TextBlock Text="{Binding Subject}">
                            <TextBlock.Foreground>
                                <TextBlockResource Key="AccentButtonBackground"/>
                            </TextBlock.Foreground>
                        </TextBlock>
                    <StackPanel Grid.Column="3" Orientation="Horizontal">
                        <AppBarButton Name="Edit" Icon="Edit"
Label="Edit" Tag="{Binding}" Click="Edit_Click"/>
                        <AppBarButton Name="Delete" Icon="Delete"
Label="Delete" Tag="{Binding}" Click="Delete_Click"/>
                    </StackPanel>
                </Grid>
            </DataTemplate>
        </ListBox.ItemTemplate>
    </ListBox>
</Grid>
```

```

        </StackPanel>
    </Grid>
</DataTemplate>
<ListBox.ItemTemplate>
</ListBox>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Name="Add" Icon="Add" Label="New"
Click="Add_Click"/>
</CommandBar>
</Grid>

```

Thành phần <Grid>:

- Margin="50": Đặt lề cho Grid.
- Grid.RowDefinitions: Định nghĩa hai hàng cho Grid, với hàng đầu tiên có chiều cao tự động (Auto) và hàng thứ hai chiếm phần còn lại (*).

CalendarDatePicker:

- Grid.Row="0": Đặt trong hàng đầu tiên.
- Name="Picker": Đặt tên để tham chiếu trong mã phía sau.
- HorizontalAlignment="Stretch": Kéo dài theo chiều ngang.
- PlaceholderText="Display Appointments From": Văn bản hiển thị khi không có ngày được chọn.
- DateChanged="Picker_DateChanged": Sự kiện được gọi khi ngày được thay đổi.

ListBox:

- Grid.Row="1": Đặt trong hàng thứ hai.
- Name="Display": Đặt tên để tham chiếu trong mã phía sau.
- Loaded="Display_Loaded": Sự kiện được gọi khi ListBox được tải.

ListBox.ItemContainerStyle:

- Setter Property="HorizontalContentAlignment" Value="Stretch": Đặt căn chỉnh nội dung theo chiều ngang cho các mục trong ListBox.

ListBox.ItemTemplate:

- Xác định cách hiển thị từng mục trong ListBox.
- Gồm một Grid với ba cột:
 - Cột 1: Hiển thị thời gian của cuộc hẹn.
 - Cột 2: Hiển thị chủ đề của cuộc hẹn.
 - Cột 3: Chứa các nút Edit và Delete.

CommandBar:

- Chứa một nút Add để thêm cuộc hẹn mới.

Trong đoạn code CalendarPage.xaml.cs:

```
namespace Convenience_App
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a
    Frame.
    /// </summary>
    public sealed partial class CalendarPage : Page
    {
        Library library = new Library();

        public CalendarPage()
        {
            this.InitializeComponent();
        }

        private async void Display_Loaded(object sender, RoutedEventArgs e)
        {
            library.Start(ref Picker);
            Display.ItemsSource = await
library.ListAsync((DateTimeOffset)Picker.Date);
        }

        private async void Picker_ValueChanged(CalendarDatePicker sender,
CalendarDatePickerEventArgs args)
        {
            Display.ItemsSource = await
library.ListAsync((DateTimeOffset)Picker.Date);
        }

        private async void Add_Click(object sender, RoutedEventArgs e)
        {
            if (await library.AddAsync(Add, App.Current.Resources))
            {
                Display.ItemsSource = await
library.ListAsync((DateTimeOffset)Picker.Date);
            }
        }
    }
}
```

```

private async void Edit_Click(object sender, RoutedEventArgs e)
{
    if (await library.EditAsync((AppBarButton)sender,
App.Current.Resources))
    {
        Display.ItemsSource = await
library.ListAsync((DateTimeOffset)Picker.Date);
    }
}

private async void Delete_Click(object sender, RoutedEventArgs e)
{
    if (await library.DeleteAsync((AppBarButton)sender))
    {
        Display.ItemsSource = await
library.ListAsync((DateTimeOffset)Picker.Date);
    }
}
}

```

Class CalendarPage:

- Lớp CalendarPage kế thừa từ lớp Page.
- Khởi tạo một đối tượng Library để quản lý các cuộc hẹn.

Constructor CalendarPage:

- Gọi InitializeComponent () để khởi tạo các thành phần giao diện.

Display_Loaded:

- Được gọi khi ListBox được tải.
- library.Start (ref Picker): Khởi tạo Picker với ngày hiện tại nếu chưa được đặt.
- Display.ItemsSource = await
library.ListAsync ((DateTimeOffset) Picker.Date): Thiết lập nguồn dữ liệu cho ListBox với các cuộc hẹn từ ngày được chọn.

Picker_ValueChanged:

- Được gọi khi ngày trong Picker thay đổi.
- Cập nhật nguồn dữ liệu của ListBox dựa trên ngày mới được chọn.

Add_Click:

- Được gọi khi nút Add được nhấn.
- Mở hộp thoại thêm cuộc hẹn mới và cập nhật ListBox nếu thêm thành công.

Edit_Click:

- Được gọi khi nút Edit được nhấn.
- Mở hộp thoại chỉnh sửa cuộc hẹn và cập nhật ListBox nếu chỉnh sửa thành công.

Delete_Click:

- Được gọi khi nút Delete được nhấn.
- Xóa cuộc hẹn và cập nhật ListBox nếu xóa thành công.

Tóm lại: Hai phần chương trình trên kết hợp với nhau để tạo ra trang CalendarPage với các chức năng quản lý cuộc hẹn, cung cấp trải nghiệm người dùng mượt mà và dễ sử dụng:

CalendarPage.xaml:

- Định nghĩa giao diện người dùng bao gồm CalendarDatePicker để chọn ngày và ListBox để hiển thị các cuộc hẹn.
- Các nút Add, Edit, Delete để thêm, chỉnh sửa và xóa các cuộc hẹn.
- Giao diện trực quan và dễ sử dụng với các thành phần được căn chỉnh và bố trí hợp lý.

CalendarPage.xaml.cs:

- Triển khai logic phía sau để xử lý các sự kiện giao diện người dùng.
- Tương tác với lớp Library để quản lý dữ liệu cuộc hẹn.
- Đảm bảo cập nhật hiển thị khi có thay đổi (thêm, sửa, xóa cuộc hẹn).

Bổ sung thêm về Library Class: Chương trình sử dụng một lớp Library để quản lý các cuộc hẹn với các chức năng của Library:

- Library có thể chứa các phương thức như Start, ListAsync, AddAsync, EditAsync, DeleteAsync.
- Phương thức Start có thể khởi tạo hoặc đặt lại ngày cho CalendarDatePicker.
- Phương thức ListAsync lấy danh sách các cuộc hẹn dựa trên ngày.
- Phương thức AddAsync mở hộp thoại thêm cuộc hẹn và thêm vào danh sách.
- Phương thức EditAsync mở hộp thoại chỉnh sửa cuộc hẹn và cập nhật danh sách.
- Phương thức DeleteAsync xóa cuộc hẹn khỏi danh sách.

```
namespace Convenience_App
{
    public class Library
    {
        private const string app_title = "Calendar App";

        private async Task<AppointmentStore> Store()
        {
            return await
AppointmentManager.RequestStoreAsync(AppointmentStoreAccessType.AppCalendarsR
eadWrite);
        }

        private DateTime GetDateTime(DateTimeOffset date, TimeSpan time)
        {
            return new DateTime(date.Year, date.Month, date.Day, time.Hours,
time.Minutes, time.Seconds);
        }

        private async Task<Appointment> Dialog(Appointment appointment,
ResourceDictionary resources)
        {
            Thickness margin = new Thickness(5);
            DatePicker date = new DatePicker()
            {
                Date = appointment.StartTime,
                Margin = margin,
                HorizontalAlignment = HorizontalAlignment.Stretch
            };
            TimePicker time = new TimePicker()
            {
                Time = appointment.StartTime.TimeOfDay,
                Margin = margin,
                HorizontalAlignment = HorizontalAlignment.Stretch
            };
            TextBox subject = new TextBox()
            {
                Text = appointment.Subject,
            }
        }
    }
}
```

```

        Margin = margin,
        PlaceholderText = "Subject"
    };
    TextBox details = new TextBox()
    {
        Text = appointment.Details,
        Margin = margin,
        PlaceholderText = "Details",
        AcceptsReturn = true,
        TextWrapping = TextWrapping.Wrap,
        Height =
(double)resources[ "SearchBoxSuggestionPopupThemeMaxHeight" ]
    };
    StackPanel panel = new StackPanel()
    {
        Orientation = Orientation.Vertical
    };
    panel.Children.Add(date);
    panel.Children.Add(time);
    panel.Children.Add(subject);
    panel.Children.Add(details);
    ContentDialog dialog = new ContentDialog()
    {
        Title = "Appointment",
        PrimaryButtonText = "Save",
        CloseButtonText = "Cancel",
        Content = panel
    };
    ContentDialogResult result = await dialog.ShowAsync();
    if (result == ContentDialogResult.Primary)
    {
        appointment.StartTime = GetDateTime(date.Date, time.Time);
        appointment.Subject = subject.Text;
        appointment.Details = details.Text;
        return appointment;
    }
    return null;
}

private async Task<AppointmentCalendar> GetAsync()
{
    AppointmentCalendar result = null;
    AppointmentStore store = await Store();
    if (store != null)
    {
        IReadOnlyList<AppointmentCalendar> list = await
store.FindAppointmentCalendarsAsync();
        if (list.Count == 0)
        {
            result = await
store.CreateAppointmentCalendarAsync(app_title);
        }
        else
        {
            result = list.FirstOrDefault(s => s.DisplayName ==
app_title);
        }
    }
}

```

```

        }
    }
    return result;
}

private async Task<IReadOnlyList<Appointment>>
ListAppointmentsAsync(DateTimeOffset start, TimeSpan range)
{
    AppointmentStore store = await Store();
    if (store != null)
    {
        AppointmentCalendar calendar = await GetAsync();
        if (calendar != null)
        {
            return await calendar.FindAppointmentsAsync(start,
range);
        }
    }
    return null;
}

private async Task<Appointment> GetAppointmentAsync(string id)
{
    AppointmentCalendar calendar = await GetAsync();
    if (calendar != null)
    {
        return await calendar.GetAppointmentAsync(id);
    }
    return null;
}

public void Start(ref CalendarDatePicker picker)
{
    DateTime now = DateTime.Now;
    if (picker.Date == null)
    {
        picker.Date = GetDateTime(DateTime.Now, new TimeSpan(0, 0,
0));
    }
}

public async Task<List<Item>> ListAsync(DateTimeOffset start)
{
    start = start.AddDays(-1);
    DateTimeOffset finish = start.AddMonths(1);
    TimeSpan range = finish - start;
    List<Item> results = new List<Item>();
    IReadOnlyList<Appointment> appointments = await
ListAppointmentsAsync(start, range);
    foreach (Appointment appointment in appointments)
    {
        results.Add(new Item()
        {
            Id = appointment.LocalId,
            StartTime = appointment.StartTime,
            Subject = appointment.Subject,

```

```

        Details = appointment.Details
    });
}
return results;
}

public async Task<bool> AddAsync(AppBarButton button,
ResourceDictionary resources)
{
    Appointment appointment = await Dialog(new Appointment(),
resources);
    if (appointment != null)
    {
        AppointmentCalendar calendar = await GetAsync();
        if (calendar != null)
        {
            await calendar.SaveAppointmentAsync(appointment);
            return true;
        }
    }
    return false;
}

public async Task<bool> EditAsync(AppBarButton button,
ResourceDictionary resources)
{
    Item item = (Item)button.Tag;
    Appointment appointment = await GetAppointmentAsync(item.Id);
    if (appointment != null)
    {
        appointment = await Dialog(appointment, resources);
        if (appointment != null)
        {
            AppointmentCalendar calendar = await GetAsync();
            if (calendar != null)
            {
                await calendar.SaveAppointmentAsync(appointment);
                return true;
            }
        }
    }
    return false;
}

public async Task<bool> DeleteAsync(AppBarButton button)
{
    Item item = (Item)button.Tag;
    Appointment appointment = await GetAppointmentAsync(item.Id);
    if (appointment != null)
    {
        AppointmentCalendar calendar = await GetAsync();
        if (calendar != null)
        {
            await calendar.DeleteAppointmentAsync(item.Id);
            return true;
        }
    }
}

```

```
        }
    }
}
```

Lớp Library là trọng tâm để quản lý logic dữ liệu và giúp tách biệt giữa giao diện người dùng và logic nghiệp vụ. Lớp Library trong ứng dụng Convenience_App chịu trách nhiệm quản lý các cuộc hẹn. Dưới đây là cách lớp này hoạt động:

Khởi tạo và Lấy Store:

- `private async Task<AppointmentStore> Store():` Phương thức này yêu cầu quyền truy cập vào cửa hàng cuộc hẹn và trả về một đối tượng AppointmentStore.

Lấy Lịch:

- `private async Task<AppointmentCalendar> GetAsync():` Phương thức này lấy lịch cuộc hẹn của ứng dụng. Nếu lịch chưa tồn tại, nó sẽ tạo một lịch mới.

Danh sách cuộc hẹn:

- `private async Task<IReadOnlyList<Appointment>> ListAppointmentsAsync(DateTimeOffset start, TimeSpan range):` Phương thức này trả về danh sách các cuộc hẹn trong khoảng thời gian từ start đến range.

Khởi tạo Picker:

- `public void Start(ref CalendarDatePicker picker):` Đặt ngày hiện tại cho CalendarDatePicker nếu chưa được đặt.

Danh sách các mục:

- `public async Task<List<Item>> ListAsync(DateTimeOffset start):` Trả về danh sách

các đối tượng Item đại diện cho các cuộc hẹn bắt đầu từ start và kéo dài một tháng.

Thêm cuộc hẹn:

- public async Task<bool> AddAsync(AppBarButton button, ResourceDictionary resources): Mở hộp thoại để thêm cuộc hẹn mới và lưu nó vào lịch. Trả về true nếu thành công, false nếu không.

Chỉnh sửa cuộc hẹn:

- public async Task<bool> EditAsync(AppBarButton button, ResourceDictionary resources): Mở hộp thoại để chỉnh sửa cuộc hẹn hiện có và lưu các thay đổi. Trả về true nếu thành công, false nếu không.

Xóa cuộc hẹn:

- public async Task<bool> DeleteAsync(AppBarButton button): Xóa cuộc hẹn được chọn khỏi lịch. Trả về true nếu thành công, false nếu không.

Hộp thoại cuộc hẹn:

- private async Task<Appointment> Dialog(Appointment appointment, ResourceDictionary resources): Mở hộp thoại để nhập thông tin cuộc hẹn. Nếu người dùng lưu, cập nhật đối tượng Appointment và trả về. Nếu không, trả về null.

Tổng quan quá trình:

- Khi người dùng chọn một ngày, Library sẽ lấy danh sách các cuộc hẹn từ ngày đó.
- Người dùng có thể thêm, chỉnh sửa hoặc xóa cuộc hẹn thông qua các phương thức tương ứng (AddAsync, EditAsync, DeleteAsync).
- Mỗi hành động trên đều mở một hộp thoại (được triển khai trong phương thức Dialog) để nhập hoặc chỉnh sửa thông tin cuộc hẹn.

- Sau khi thêm, chỉnh sửa hoặc xóa, danh sách cuộc hẹn sẽ được cập nhật để hiển thị các thay đổi.

Tóm lược: Lớp Library cung cấp các phương thức để quản lý cuộc hẹn trong ứng dụng, bao gồm việc lấy lịch, danh sách cuộc hẹn, thêm, chỉnh sửa và xóa cuộc hẹn. Nó đảm bảo rằng dữ liệu cuộc hẹn được đồng bộ hóa với giao diện người dùng một cách hiệu quả.

Class Item – Item.cs

```
public class Item
{
    public string Id { get; set; }
    public DateTimeOffset StartTime { get; set; }
    public string When { get { return StartTime.ToString("dd MMMM yyyy
HH:mm:ss"); } }
    public string Subject { get; set; }
    public string Details { get; set; }
}
```

Lớp Item là một phần quan trọng của ứng dụng Calendar, đại diện cho một cuộc hẹn trong lịch. Dưới đây là giải thích chi tiết về lớp Item:

Định nghĩa lớp Item:

- `public class Item`: Khai báo lớp Item là public để có thể truy cập từ các lớp khác trong ứng dụng.

Thuộc tính Id:

- `public string Id { get; set; }`: Property Id đại diện cho mã định danh duy nhất của một cuộc hẹn.
- Sử dụng property này để đọc (get) hoặc gán (set) giá trị mã định danh cho cuộc hẹn.

Thuộc tính StartTime:

- `public DateTimeOffset StartTime { get; set; }`: Property StartTime lưu trữ thời gian bắt đầu của cuộc hẹn dưới dạng DateTimeOffset.
- DateTimeOffset là một cấu trúc bao gồm thông tin về ngày, giờ và múi giờ.

- Sử dụng property này để đọc (get) hoặc gán (set) giá trị thời gian bắt đầu của cuộc hẹn.

Thuộc tính When:

- `public string When { get { return StartTime.ToString("dd MMMM yyyy HH:mm:ss"); } }`: Property When chỉ có get accessor, tức là chỉ có thể đọc giá trị mà không thể gán giá trị mới.
- Property này trả về chuỗi định dạng ngày giờ từ StartTime dưới dạng dd MMMM yyyy HH:mm:ss (ví dụ: "25 June 2024 14:30:00").
- Sử dụng property này để lấy chuỗi hiển thị thời gian bắt đầu của cuộc hẹn.

Thuộc tính Subject:

- `public string Subject { get; set; }`: Property Subject đại diện cho tiêu đề hoặc tên của cuộc hẹn.
- Sử dụng property này để đọc (get) hoặc gán (set) giá trị tiêu đề cho cuộc hẹn.

Thuộc tính Details:

- `public string Details { get; set; }`: Property Details lưu trữ chi tiết hoặc mô tả của cuộc hẹn.
- Sử dụng property này để đọc (get) hoặc gán (set) giá trị mô tả cho cuộc hẹn.

Tổng quan về Properties trong Item Class

- **Properties:** Thuộc tính trong lớp Item giúp bảo vệ dữ liệu bằng cách ẩn việc truy cập trực tiếp đến các trường (fields) bên trong lớp. Thay vào đó, chúng cung cấp các phương thức truy cập có kiểm soát (getters và setters).
- **Encapsulation:** Các property cho phép đóng gói (encapsulation) dữ liệu và logic liên quan bên trong lớp. Điều này giúp duy trì tính nhất quán và bảo mật của dữ liệu.

- **Read-Only Property:** Property When là một ví dụ về property chỉ có getter, tức là giá trị của nó được tính toán từ dữ liệu hiện tại và không thể gán giá trị trực tiếp.

Tóm lại: Các property trong lớp Item cung cấp một cách tiếp cận linh hoạt và an toàn để truy cập và quản lý dữ liệu của các cuộc hẹn. Chúng đóng vai trò quan trọng trong việc duy trì tính toàn vẹn và bảo mật của dữ liệu, đồng thời cung cấp các phương thức thuận tiện để thao tác với các thuộc tính của đối tượng.

2.1.4. Play Page

PlayPage là một trang của ứng dụng Convenience App, cung cấp trò chơi ghép đôi các biểu tượng động vật (animal emojis). Mục tiêu của trò chơi là tìm tất cả các cặp biểu tượng giống nhau trong thời gian ngắn nhất. PlayPage có các tính năng chính sau:

- Thiết kế giao diện người dùng (UI) đơn giản và trực quan: Sử dụng XAML để định nghĩa giao diện với lưới chứa các TextBlock đại diện cho các ô trong trò chơi.
- Quản lý thời gian và số cặp biểu tượng: Sử dụng DispatcherTimer để đo thời gian trò chơi và cập nhật số cặp biểu tượng đã tìm thấy.
- Tương tác người dùng: Xử lý sự kiện Tapped trên các TextBlock để người chơi có thể lật các ô và kiểm tra sự trùng khớp.

Chi tiết nội dung của PlayPage.xaml

```
<Grid>
    <Grid x:Name="mainGrid">
        <Grid.RowDefinitions>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
            <RowDefinition Height="1*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
            <ColumnDefinition Width="1*"/>
        </Grid.ColumnDefinitions>
```

```

        <TextBlock Grid.Row="1" Grid.Column="0" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
            <TextBlock Grid.Row="1" Grid.Column="1" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                <TextBlock Grid.Row="1" Grid.Column="2" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                    <TextBlock Grid.Row="1" Grid.Column="3" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                        <TextBlock Grid.Row="0" Grid.Column="0" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                            <TextBlock Grid.Row="0" Grid.Column="1" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                <TextBlock Grid.Row="0" Grid.Column="2" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                    <TextBlock Grid.Row="0" Grid.Column="3" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                        <TextBlock Grid.Row="2" Grid.Column="0" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                            <TextBlock Grid.Row="2" Grid.Column="1" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                <TextBlock Grid.Row="2" Grid.Column="2" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                    <TextBlock Grid.Row="2" Grid.Column="3" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                        <TextBlock Grid.Row="3" Grid.Column="0" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                            <TextBlock Grid.Row="3" Grid.Column="1" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                                <TextBlock Grid.Row="3" Grid.Column="2" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                                    <TextBlock Grid.Row="3" Grid.Column="3" Text="?" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="40" Tapped="TextBlock_Tapped"/>
                                                                        <TextBlock Grid.Row="4" Grid.ColumnSpan="4" Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center" x:Name="bestTimeTextBlock" Text="Best Time: ---" />

```

```

        <TextBlock x:Name="timeTextBlock" Text="Start"
HorizontalAlignment="Center"
VerticalAlignment="Center" FontSize="40" Tapped="timeTextBlock_Tapped"/>
    </StackPanel>
</Grid>
</Grid>
```

- Grid Layout: Sử dụng Grid để tạo bố cục 4x5 chứa các TextBlock, mỗi ô chứa một biểu tượng động vật.
- TextBlocks: Các TextBlock ban đầu có nội dung là "?" và sẽ hiển thị biểu tượng động vật khi người dùng nhấn vào.
- StackPanel: Chứa các thông tin về thời gian tốt nhất (bestTimeTextBlock) và thời gian hiện tại (timeTextBlock), cùng với sự kiện nhấn để bắt đầu hoặc chơi lại.

Chi tiết nội dung của PlayPage.xaml.cs

```

namespace Convenience_App
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a
    Frame.
    /// </summary>
    public sealed partial class PlayPage : Page
    {
        float bestTime;

        TextBlock lastTextBlockClicked;
        bool findingMatch = false;

        DispatcherTimer timer = new DispatcherTimer(); //Tạo biến timer
        int tenthsOfSecondsElapsed; //Tạo biến đếm thời gian 1/10 giây
        int matchesFound; //Số cặp được trùng khớp

        public PlayPage()
        {
            this.InitializeComponent();
            timer.Interval = TimeSpan.FromSeconds(.1);
            timer.Tick += Timer_Tick; ;
            SetUpGame();
        }

        private void Timer_Tick(object sender, object e)
        {
            tenthsOfSecondsElapsed++;
            timeTextBlock.Text = (tenthsOfSecondsElapsed /
10F).ToString("0.0s");
            if (matchesFound == 8)
            {
                float currentTime = tenthsOfSecondsElapsed / 10F;
```

```

        if (bestTime == 0 || currentTime < bestTime)
        {
            bestTime = currentTime;
            bestTimeTextBlock.Text = "Best Time: " +
bestTime.ToString("0.0s");
        }

        timer.Stop();
        timeTextBlock.Text = timeTextBlock.Text + " - Play again?";
    }
}

private void SetUpGame()
{
    List<string> animalEmoji = new List<string>()
    {
        "🐯", "🐯",
        "🐶", "🐶",
        "🐱", "🐱",
        "🐵", "🐵",
        "🦉", "🦉",
        "🐺", "🐺",
        "🐱", "🐱",
        "🐹", "🐹",
        "🐦", "🐦",
    };
    Random random = new Random(); //Tạo biến random
    foreach (TextBlock textBlock in
mainGrid.Children.OfType<TextBlock>())
    {
        int index = random.Next(animalEmoji.Count); //Tạo giá trị
ngẫu nhiên 0 - kích thước của list
        string nextEmoji = animalEmoji[index]; //Tạo emoji ngẫu
nhiên từ list với index
        textBlock.Text = nextEmoji; //Gán Emoj cho textBlock
        animalEmoji.RemoveAt(index); //Xóa phần tử vừa gán
        textBlock.Visibility = Visibility.Visible; // Reset trạng
tháí của các TextBlock
    }
    tenthsOfSecondsElapsed = 0;
    matchesFound = 0;
}

private void TextBlock_Tapped(object sender, TappedRoutedEventArgs e)
{
    TextBlock textBlock = sender as TextBlock;
    if (findingMatch == false)
    {
        textBlock.Visibility = Visibility.Collapsed;
        lastTextBlockClicked = textBlock;
        findingMatch = true;
    }
    else if (textBlock.Text == lastTextBlockClicked.Text && textBlock
!= lastTextBlockClicked)
    {
        textBlock.Visibility = Visibility.Collapsed;
    }
}

```

```

thấy
        matchesFound++; // Tăng số lượng cặp hình giống nhau đã tìm
    }
    if (matchesFound == 8)
    {
        float currentTime = tenthsOfSecondsElapsed / 10F;
        if (bestTime == 0 || currentTime < bestTime)
        {
            bestTime = currentTime;
            bestTimeTextBlock.Text = "Best Time: " +
bestTime.ToString("0.0s");
        }
        timer.Stop(); // Dừng timer nếu tất cả cặp hình giống
nhau đã được tìm thấy
        timeTextBlock.Text = timeTextBlock.Text + " - Play
again?";
    }
    findingMatch = false;
}
else
{
    lastTextBlockClicked.Visibility = Visibility.Visible;
    findingMatch = false;
}
}

private void timeTextBlock_Tapped(object sender,
TappedRoutedEventArgs e)
{
    if (matchesFound == 8 || tenthsOfSecondsElapsed == 0)
    {
        SetUpGame();
        timer.Start(); // Khởi động timer khi bắt đầu hoặc chơi lại
    }
}
}

```

Khởi tạo và thiết lập:

- Constructor (PlayPage): Khởi tạo các biến, thiết lập khoảng thời gian cho timer và gọi phương thức SetUpGame () để khởi tạo trò chơi.
- SetUpGame () : Tạo danh sách các biểu tượng động vật, phân phối ngẫu nhiên các biểu tượng vào các ô trong lưới và đặt lại trạng thái ban đầu cho trò chơi.

Xử lý sự kiện:

- Timer_Tick: Cập nhật thời gian trôi qua, hiển thị thời gian và kiểm tra nếu tất cả các cặp đã được tìm thấy để dừng timer và cập nhật thời gian tốt nhất nếu cần.

- `TextBlock_Tapped`: Xử lý sự kiện nhấn vào `TextBlock`, kiểm tra và xử lý các cặp biểu tượng giống nhau, ẩn hoặc hiển thị các `TextBlock` phù hợp.
- `timeTextBlock_Tapped`: Khởi động lại trò chơi khi người dùng nhấn vào `timeTextBlock` nếu tất cả các cặp đã được tìm thấy hoặc trò chơi chưa bắt đầu.

Giải thích chi tiết từng phần:

Thuộc tính và Biến

- `float bestTime`: Biến lưu thời gian tốt nhất mà người chơi đạt được.
- `TextBlock lastTextBlockClicked`: Lưu trữ `TextBlock` cuối cùng mà người dùng đã nhấn.
- `bool findingMatch`: Biến cờ đánh dấu xem người dùng đang tìm cặp biểu tượng thứ hai.
- `DispatcherTimer timer`: Bộ đếm thời gian để theo dõi thời gian trò chơi.
- `int tenthsOfSecondsElapsed`: Đếm số phần mươi giây đã trôi qua kể từ khi bắt đầu trò chơi.
- `int matchesFound`: Số cặp biểu tượng mà người dùng đã tìm thấy.

Constructor và Phương thức Khởi tạo

```
public PlayPage()
{
    this.InitializeComponent();
    timer.Interval = TimeSpan.FromSeconds(.1);
    timer.Tick += Timer_Tick;
    SetUpGame();
}
```

- Constructor: Khởi tạo trang, đặt khoảng thời gian cho `timer` (0.1 giây), đăng ký sự kiện `Timer_Tick` và gọi `SetUpGame()` để khởi tạo trò chơi.

Timer_Tick Event Handler

```
private void Timer_Tick(object sender, object e)
```

```

tenthsOfSecondsElapsed++;
timeTextBlock.Text = (tenthsOfSecondsElapsed / 10F).ToString("0.0s");
if (matchesFound == 8)
{
    float currentTime = tenthsOfSecondsElapsed / 10F;
    if (bestTime == 0 || currentTime < bestTime)
    {
        bestTime = currentTime;
        bestTimeTextBlock.Text = "Best Time: " +
bestTime.ToString("0.0s");
    }

    timer.Stop();
    timeTextBlock.Text = timeTextBlock.Text + " - Play again?";
}
}

```

- Timer_Tick: Mỗi khi timer kích hoạt (mỗi 0.1 giây), tăng tenthsOfSecondsElapsed, cập nhật hiển thị thời gian và kiểm tra nếu tất cả các cặp đã được tìm thấy.
- Nếu tất cả các cặp được tìm thấy, cập nhật thời gian tốt nhất nếu cần thiết và dừng timer.

SetUpGame Method

```

private void SetUpGame()
{
    List<string> animalEmoji = new List<string>
    {
        "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe",
        "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe", "\ud83d\udcbb", "\ud83d\udcbe",
    };
    Random random = new Random();
    foreach (TextBlock textBlock in mainGrid.Children.OfType<TextBlock>())
    {
        int index = random.Next(animalEmoji.Count);
        string nextEmoji = animalEmoji[index];
        textBlock.Text = nextEmoji;
        animalEmoji.RemoveAt(index);
        textBlock.Visibility = Visibility.Visible;
    }
    tenthsOfSecondsElapsed = 0;
    matchesFound = 0;
}

```

- SetUpGame: Khởi tạo danh sách các biểu tượng động vật, phân phối ngẫu nhiên chúng vào các TextBlock trong mainGrid, và đặt lại các biến trạng thái trò chơi.

TextBlock_Tapped Event Handler

```

private void TextBlock_Tapped(object sender, TappedRoutedEventArgs e)
{
    TextBlock textBlock = sender as TextBlock;
    if (findingMatch == false)
    {
        textBlock.Visibility = Visibility.Collapsed;
        lastTextBlockClicked = textBlock;
        findingMatch = true;
    }
    else if (textBlock.Text == lastTextBlockClicked.Text && textBlock != lastTextBlockClicked)
    {
        textBlock.Visibility = Visibility.Collapsed;
        matchesFound++;
        if (matchesFound == 8)
        {
            float currentTime = tenthsOfSecondsElapsed / 10F;
            if (bestTime == 0 || currentTime < bestTime)
            {
                bestTime = currentTime;
                bestTimeTextBlock.Text = "Best Time: " +
bestTime.ToString("0.0s");
            }
            timer.Stop();
            timeTextBlock.Text = timeTextBlock.Text + " - Play again?";
        }
        findingMatch = false;
    }
    else
    {
        lastTextBlockClicked.Visibility = Visibility.Visible;
        findingMatch = false;
    }
}

```

- `TextBlock_Tapped`: Xử lý khi người dùng nhấn vào `TextBlock`.
 - Nếu đây là lần nhấn đầu tiên trong cặp, ẩn `TextBlock` và lưu lại nó.
 - Nếu đây là lần nhấn thứ hai và hai `TextBlock` khớp nhau, ẩn `TextBlock` và tăng số cặp tìm thấy.
 - Nếu không khớp, hiện lại `TextBlock` trước đó.
 - Nếu tất cả các cặp đã được tìm thấy, cập nhật thời gian tốt nhất và dừng `timer`.

timeTextBlock_Tapped Event Handler

```

private void timeTextBlock_Tapped(object sender, TappedRoutedEventArgs e)
{
    if (matchesFound == 8 || tenthsOfSecondsElapsed == 0)
}

```

```

    {
        SetUpGame();
        timer.Start();
    }
}

```

- timeTextBlock_Tapped: Xử lý khi người dùng nhấn vào timeTextBlock để bắt đầu hoặc chơi lại trò chơi.
- Nếu tất cả các cặp đã được tìm thấy hoặc trò chơi chưa bắt đầu, khởi tạo lại trò chơi và bắt đầu timer.

Tóm lại: Đoạn code PlayPage.xaml.cs điều khiển logic của trò chơi ghép đôi biểu tượng động vật. Nó quản lý giao diện người dùng, xử lý sự kiện nhấn, quản lý thời gian thực và kiểm tra các điều kiện để cập nhật trạng thái trò chơi. Các chức năng của từng phần mã được tổ chức rõ ràng và dễ hiểu, giúp duy trì và mở rộng trò chơi trong tương lai trở nên dễ dàng hơn.

2.1.5. Calculator Page

Ứng dụng Convenience_App chứa một trang tính toán (CalculatorPage) được xây dựng trên nền tảng XAML và C#. Trang này có chức năng chính là cho phép người dùng nhập hai giá trị, chọn một phép toán (cộng, trừ, nhân, chia) và hiển thị kết quả của phép toán đó. Chúng ta sẽ đi qua từng thành phần và giải thích chi tiết cách mà chúng hoạt động cùng nhau để cung cấp chức năng này.

Chi tiết đoạn code XAML CalculatorPage.xaml

```

<Page.DataContext>
    <local:MainViewModel/>
</Page.DataContext>

<Page.Resources>
    <local:BoolToVisibilityConverter x:Key="boolToVisibilityConverter"/>
</Page.Resources>

<Grid>
    <Grid.Background>
        <ImageBrush ImageSource="/Assets/Calculator.png"/>
    </Grid.Background>

    <StackPanel Orientation="Vertical" HorizontalAlignment="Center"
VerticalAlignment="Center">
        <TextBox Text="{Binding Value1, Mode=TwoWay}" Width="300"
Margin="0, 10"/>

```

```

        <TextBox Text="{Binding Value2, Mode=TwoWay}" Width="300"
Margin="0, 10"/>
        <StackPanel Width="400" Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0, 10">
            <RadioButton.IsChecked="{Binding IsAddChecked, Mode=TwoWay}"
Content="+"/>
            <RadioButton.IsChecked="{Binding IsSubtractChecked,
Mode=TwoWay}" Content="-"/>
            <RadioButton.IsChecked="{Binding IsMultiplyChecked,
Mode=TwoWay}" Content="*"/>
            <RadioButton.IsChecked="{Binding IsDivideChecked,
Mode=TwoWay}" Content="/"/>
        </StackPanel>
        <TextBlock Text="{Binding Result, Mode=TwoWay}" Width="300"
Margin="0, 10" Visibility="{Binding IsAnyRadioBtnChecked,
Converter={StaticResource boolToVisibilityConverter}}"/>
        <Button Command="{Binding OkButtonClicked}" IsEnabled="{Binding
IsAnyRadioBtnChecked}" Content="OK" HorizontalAlignment="Center"
VerticalAlignment="Center" Margin="0, 10" Width="100"
Background="#33000000"/>
    </StackPanel>
</Grid>

```

Chức năng tổng quan: CalculatorPage.xaml định nghĩa giao diện người dùng cho trang tính toán. Trang này chứa các thành phần giao diện như TextBox để nhập giá trị, RadioButton để chọn phép toán, TextBlock để hiển thị kết quả và Button để thực hiện phép toán.

Chi tiết từng thành phần:

- **Page.DataContext:** Liên kết trang này với MainViewModel để sử dụng dữ liệu và lệnh từ ViewModel.
- **Page.Resources:** Định nghĩa một BoolToVisibilityConverter để chuyển đổi giá trị boolean thành kiểu hiển thị (Visibility).
- **Grid:** Cấu trúc chính của trang, chứa một hình nền và các thành phần giao diện khác.
- **ImageBrush:** Đặt hình nền cho lưới là hình ảnh Calculator.png.
- **StackPanel:** Sắp xếp các phần tử con theo chiều dọc:
 - **TextBox:** Nhập Value1 và Value2, liên kết hai chiều với thuộc tính tương ứng trong MainViewModel.
 - **RadioButton:** Chọn phép toán (+, -, *, /), liên kết hai chiều với các thuộc tính Boolean trong MainViewModel.

- **TextBlock**: Hiển thị kết quả của phép toán, chỉ hiện khi có một **RadioButton** được chọn.
- **Button**: Thực hiện lệnh tính toán khi được nhấn. Kích hoạt lệnh **OkButtonClicked** khi được nhấn, chỉ khả dụng khi có một **RadioButton** được chọn.

Trong CalculatorPage.xaml.cs

```
namespace Convenience_App
{
    public sealed partial class CalculatorPage : Page
    {
        public CalculatorPage()
        {
            this.InitializeComponent();
        }
    }
}
```

- Đoạn chương trình chịu trách nhiệm khởi tạo trang. **CalculatorPage** kế thừa từ **Page** và gọi phương thức **InitializeComponent()** để khởi tạo các thành phần trong XAML.

Class Calculators: chứa các phương thức thực hiện các phép toán cơ bản như cộng, trừ, nhân, chia.

```
namespace Convenience_App
{
    public class Calculators
    {
        private int value1 = 0;
        private int value2 = 0;

        public Calculators(int val1, int val2)
        {
            value1 = val1;
            value2 = val2;
        }

        public int Add()
        {
            return value1 + value2;
        }

        public int Subtract()
        {
            return value1 - value2;
        }

        public int Multiply()
        {
```

```

        return value1 * value2;
    }

    public int Divide()
    {
        return value1 / value2;
    }
}

```

Chi tiết các thành phần:

- Namespace: Convenience_App là không gian tên chứa class Calculators.
- Class: Calculators là lớp đại diện cho mô hình thực hiện các phép toán số học.
- Fields: value1 và value2 là các trường dữ liệu (fields) lưu trữ hai giá trị số học đầu vào.
- Constructor: Calculators(int val1, int val2) là phương thức khởi tạo, nhận hai tham số đầu vào và gán chúng cho các trường value1 và value2.
- Các Methods:
 - Add(): Phương thức thực hiện phép cộng, trả về tổng của value1 và value2.
 - Subtract(): Phương thức thực hiện phép trừ, trả về hiệu của value1 và value2.
 - Multiply(): Phương thức thực hiện phép nhân, trả về tích của value1 và value2.
 - Divide(): Phương thức thực hiện phép chia, trả về thương của value1 và value2.

Class MainViewModelBase.cs - Lớp cơ sở cho ViewModel, triển khai giao diện INotifyPropertyChanged để hỗ trợ thông báo thay đổi dữ liệu.

```

namespace Convenience_App
{
    public abstract class MainViewModelBase : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;

        protected virtual void OnPropertyChanged(string propertyName)

```

```

    {
        PropertyChangedEventHandler handler = this.PropertyChanged;
        if (handler != null)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}

```

Chi tiết các thành phần:

- Namespace: Convenience_App là không gian tên chứa class MainViewModelBase.
- Class: MainViewModelBase là lớp trừu tượng (abstract) cung cấp chức năng cơ bản cho ViewModel.
- Event: PropertyChanged là sự kiện được khai báo từ giao diện INotifyPropertyChanged để thông báo khi một thuộc tính thay đổi.
- Method: OnPropertyChanged(string propertyName) là phương thức bảo vệ (protected) gọi sự kiện PropertyChanged với tên của thuộc tính đã thay đổi. Phương thức này giúp cập nhật giao diện người dùng khi dữ liệu thay đổi.

Class MainViewModel.cs - Lớp ViewModel chứa logic xử lý cho CalculatorPage, kế thừa từ MainViewModelBase.

```

namespace Convenience_App
{
    class MainViewModel : MainViewModelBase
    {
        private Calculators calculator_;
        private int result_;
        private int value1_;
        private int value2_;
        private bool isAddChecked_;
        private bool isSubtractChecked_;
        private bool isMultiplyChecked_;
        private bool isDivideChecked_;
        private bool isAnyRadioBtnChecked_;

        public bool IsAnyRadioBtnChecked
        {
            get { return isAnyRadioBtnChecked_; }
            set
            {
                isAnyRadioBtnChecked_ = value;
                OnPropertyChanged("IsAnyRadioBtnChecked");
            }
        }
    }
}

```

```

    }

    public int Value1
    {
        get { return value1_; }
        set
        {
            value1_ = value;
            OnPropertyChanged("Value1");
        }
    }

    public int Value2
    {
        get { return value2_; }
        set
        {
            value2_ = value;
            OnPropertyChanged("Value2");
        }
    }

    public int Result
    {
        get { return result_; }
        set
        {
            if (value != result_)
            {
                result_ = value;
                OnPropertyChanged("Result");
            }
        }
    }

    public bool IsAddChecked
    {
        get { return isAddChecked_; }
        set
        {
            if (value != isAddChecked_)
            {
                isAddChecked_ = value;
                IsAnyRadioBtnChecked = true;
                OnPropertyChanged("IsAddChecked");
            }
        }
    }

    public bool IsSubtractChecked
    {
        get { return isSubtractChecked_; }
        set
        {
            if (value != isSubtractChecked_)
            {

```

```

        isSubtractChecked_ = value;
        IsAnyRadioBtnChecked = true;
        OnPropertyChanged("IsSubtractChecked");
    }
}
}

public bool IsMultiplyChecked
{
    get { return isMultiplyChecked_; }
    set
    {
        if (value != isMultiplyChecked_)
        {
            isMultiplyChecked_ = value;
            IsAnyRadioBtnChecked = true;
            OnPropertyChanged("IsMultiplyChecked");
        }
    }
}

public bool IsDivideChecked
{
    get { return isDivideChecked_; }
    set
    {
        if (value != isDivideChecked_)
        {
            isDivideChecked_ = value;
            IsAnyRadioBtnChecked = true;
            OnPropertyChanged("IsDivideChecked");
        }
    }
}

public ICommand OkButtonClicked
{
    get
    {
        return new DelegateCommand(FindResult);
    }
}

public void FindResult()
{
    calcular_ = new Calculators(Value1, Value2);
    if (IsAddChecked)
    {
        Result = calcular_.Add();
    }
    else if (IsSubtractChecked)
    {
        Result = calcular_.Subtract();
    }
    else if (IsMultiplyChecked)
    {

```

```

        Result = calcular_.Multiply();
    }
    else if (IsDivideChecked)
    {
        Result = calcular_.Divide();
    }
}
}
}

```

Chi tiết các thành phần:

- Namespace: Convenience_App là không gian tên chứa class MainViewModel.
- Class: MainViewModel kế thừa từ MainViewModelBase, chứa logic xử lý và dữ liệu cho trang tính toán.
- Fields:
 - calcular_: Đối tượng của lớp Calculators, thực hiện các phép toán.
 - result_: Kết quả của phép toán.
 - value1_, value2_: Các giá trị số học đầu vào.
 - isAddChecked_, isSubtractChecked_,
isMultiplyChecked_, isDivideChecked_: Các trạng thái của RadioButton.
 - isAnyRadioBtnChecked_: Xác định xem có RadioButton nào được chọn hay không.
- Properties:
 - IsAnyRadioBtnChecked: Thuộc tính xác định xem có RadioButton nào được chọn, sử dụng để điều khiển hiển thị kết quả và nút OK.
 - Value1, Value2: Thuộc tính lưu trữ giá trị đầu vào, liên kết với TextBox.
 - Result: Thuộc tính lưu trữ kết quả của phép toán, liên kết với TextBlock.
 - IsAddChecked, IsSubtractChecked,
IsMultiplyChecked, IsDivideChecked: Thuộc tính lưu

trữ trạng thái của các RadioButton, xác định phép toán nào được chọn.

➤ Command:

- OkButtonClicked: Lệnh thực thi khi nút OK được nhấn, gọi phương thức FindResult.

➤ Method:

- FindResult(): Phương thức tạo đối tượng Calculators với các giá trị Value1 và Value2, sau đó thực hiện phép toán dựa trên lựa chọn của RadioButton và cập nhật kết quả.

Class DelegateCommand.cs - Lớp lệnh thực thi hành động khi một lệnh được gọi, triển khai giao diện ICommand

```
namespace Convenience_App
{
    class DelegateCommand : ICommand
    {
        private SimpleEventHandler handler;
        private bool isEnabled = true;
        public event EventHandler CanExecuteChanged;
        public delegate void SimpleEventHandler();
        public DelegateCommand(SimpleEventHandler handler)
        {
            this.handler = handler;
        }
        public bool IsEnabled
        {
            get { return this.isEnabled; }
        }
        void ICommand.Execute(object org)
        {
            this.handler();
        }

        bool ICommand.CanExecute(object org)
        {
            return this.IsEnabled;
        }
        public void OnCanExecuteChanged()
        {
            if (this.CanExecuteChanged != null)
            {
                this.CanExecuteChanged(this, EventArgs.Empty);
            }
        }
    }
}
```

| }

Chi tiết các thành phần:

- Namespace: Convenience_App là không gian tên chứa class DelegateCommand.
- Class: DelegateCommand triển khai giao diện ICommand, cho phép tạo các lệnh tùy chỉnh.
- Delegate: SimpleEventHandler là một delegate đại diện cho phương thức không có tham số và không trả về giá trị.
- Constructor: DelegateCommand(SimpleEventHandler handler) nhận một delegate SimpleEventHandler, phương thức này sẽ được gọi khi lệnh được thực thi.
- Fields:
 - handler: Delegate được gọi khi lệnh được thực thi.
 - isEnabled: Xác định xem lệnh có thể thực thi hay không.
- Properties:
 - IsEnabled: Thuộc tính trả về giá trị của isEnabled.
- Events:
 - CanExecuteChanged: Sự kiện được kích hoạt khi trạng thái có thể thực thi của lệnh thay đổi.
- Methods:
 - Execute(object org): Gọi phương thức handler khi lệnh được thực thi.
 - CanExecute(object org): Trả về giá trị của isEnabled, xác định xem lệnh có thể thực thi hay không.
 - OnCanExecuteChanged(): Kích hoạt sự kiện CanExecuteChanged, thông báo UI khi trạng thái thực thi lệnh thay đổi.

Class BoolToVisibilityConverter.cs - Lớp chuyển đổi giá trị Boolean thành Visibility, triển khai giao diện IValueConverter

```

namespace Convenience_App
{
    class BoolToVisibilityConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object parameter, string language)
        {
            if (!(value is bool))
            {
                return Visibility.Collapsed;
            }
            bool objValue = (bool)value;
            if (objValue)
            {
                return Visibility.Visible;
            }
            return Visibility.Collapsed;
        }

        public object ConvertBack(object value, Type targetType, object parameter, string language)
        {
            try
            {
                if ((bool)value)
                {
                    return Visibility.Visible;
                }
            }
            catch
            {
            }
            return Visibility.Collapsed;
        }
    }
}

```

Chi tiết các thành phần:

- Namespace: Convenience_App là không gian tên chứa class BoolToVisibilityConverter.
- Class: BoolToVisibilityConverter triển khai giao diện IValueConverter, cho phép chuyển đổi giá trị Boolean thành Visibility.
- Methods:
 - Convert (object value, Type targetType, object parameter, string language): Chuyển đổi giá trị Boolean thành Visibility.Visible hoặc Visibility.Collapsed dựa trên giá trị true hoặc false.

- ConvertBack(object value, Type targetType, object parameter, string language): Chuyển đổi ngược lại từ Visibility thành Boolean, nhưng trong trường hợp này luôn trả về Visibility.Collapsed nếu có lỗi.

Tổng kết:

Các lớp trên được thiết kế theo mô hình MVVM, tách biệt rõ ràng giữa giao diện người dùng (View), logic xử lý (ViewModel), và dữ liệu (Model). Điều này giúp cho ứng dụng dễ dàng bảo trì, mở rộng và kiểm thử, đồng thời tận dụng sức mạnh của data binding và command trong XAML để tạo ra một ứng dụng có giao diện người dùng linh hoạt và mạnh mẽ.

2.1.6. AddFriend Page

Trang AddFriendPage là một giao diện người dùng được thiết kế để thêm thông tin bạn bè, bao gồm tên, email và số điện thoại.

AddFriendPage.xaml - Giao diện này bao gồm các ô nhập liệu (TextBox) để người dùng nhập thông tin và một nút (Button) để xác nhận và lưu trữ thông tin đã nhập.

```
<Grid>
    <Grid.Background>
        <ImageBrush ImageSource="/Assets/AddFriend.png"/>
    </Grid.Background>
    <StackPanel Orientation="Vertical" HorizontalAlignment="Center"
    VerticalAlignment="Center">
        <TextBox x:Name="NameTextBox" PlaceholderText="Nhập tên" Width="200"
        Margin="5"/>
        <TextBox x:Name="EmailTextBox" PlaceholderText="Nhập Email"
        Width="200" Margin="5"/>
        <TextBox x:Name="PhoneNumberTextBox" PlaceholderText="Nhập số điện
        thoại" Width="200" Margin="5"/>
        <Button x:Name="AddButton" Content="Add" Width="200" Margin="5"
        Click="AddButton_Click"/>
    </StackPanel>
</Grid>
```

Chi tiết các thành phần:

- Grid.Background: chứa một ImageBrush với thuộc tính ImageSource trỏ đến tệp hình ảnh /Assets/AddFriend.png. Hình ảnh này được dùng làm nền cho trang.

➤ TextBox:

- Có ba TextBox trong StackPanel, mỗi TextBox có thuộc tính `x:Name` để đặt tên và thuộc tính `PlaceholderText` để hiển thị văn bản gợi ý cho người dùng biết cần nhập gì vào các ô.
- NameTextBox có PlaceholderText là "Nhập tên".
- EmailTextBox có PlaceholderText là "Nhập Email".
- PhoneNumberTextBox có PlaceholderText là "Nhập số điện thoại".
- Mỗi TextBox có Width được đặt là 200 và Margin là 5 để tạo khoảng cách giữa các ô nhập liệu.

➤ Button:

- Button có `x:Name` là AddButton và thuộc tính Content là "Add", điều này xác định văn bản hiển thị trên nút.
- Width của nút được đặt là 200 và Margin là 5 để tạo khoảng cách từ các thành phần khác.
- Sự kiện Click của nút được liên kết với phương thức `AddButton_Click`, sẽ được xử lý trong phần code phía sau (code-behind) của trang để lưu thông tin bạn mới khi nút được nhấn.

`AddFriendPage.xaml.cs` - là phần code phía sau (code-behind) của trang `AddFriendPage.xaml`, chứa logic xử lý sự kiện và điều hướng.

```
namespace Convenience_App
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class AddFriendPage : Page
    {
        public AddFriendPage()
        {
            this.InitializeComponent();
        }

        public class Information
        {
            public string Name { get; set; } = "ABC";
            public string Email { get; set; } = "abc@fetel.hcmus.edu.vn";
            public string PhoneNumber { get; set; } = "0123456789";
        }
    }
}
```

```

        }

        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            Information infoParameter = new Information();
            infoParameter.Name = NameTextBox.Text;
            infoParameter.Email = EmailTextBox.Text;
            infoParameter.PhoneNumber = PhoneNumberTextBox.Text;
            Frame.Navigate(typeof(ContactInfoPage), infoParameter);
        }
    }
}

```

Chi tiết các thành phần:

➤ Khởi tạo trang AddFriendPage

- Trong hàm khởi tạo AddFriendPage() , gọi InitializeComponent() để khởi tạo giao diện
- Hàm này sẽ thiết lập và hiển thị giao diện người dùng cho trang.

➤ Tạo cấu trúc thông tin (Information)

- Information là một lớp (class) để lưu trữ thông tin về bạn bè
- Lớp này được sử dụng để tổ chức và lưu trữ các thông tin chi tiết về bạn bè mà người dùng nhập vào.
- Có ba thuộc tính (properties): Name, Email và PhoneNumber
- Các thuộc tính này tương ứng với các ô nhập liệu trong giao diện.
- Giá trị mặc định của các thuộc tính được đặt là “ABC”, “abc@fetel.hcmus.edu.vn” và “0123456789”
- Các giá trị mặc định này được sử dụng khi đối tượng Information được khởi tạo mà không có giá trị cụ thể nào được gán.

➤ Xử lý sự kiện khi người dùng nhấn nút “Thêm”

- Trong phương thức AddButton_Click() , tạo một đối tượng Information mới - đối tượng này sẽ lưu trữ các giá trị mà người dùng nhập vào các ô nhập liệu.
- Gán giá trị từ các ô nhập liệu (NameTextBox, EmailTextBox, PhoneNumberTextBox) cho các thuộc tính của đối tượng

- Thông tin từ các TextBox được gán cho các thuộc tính tương ứng của đối tượng Information.
- Sử dụng Frame.Navigate() để chuyển hướng sang trang ContactInfoPage và truyền thông tin qua tham số
- Sau khi lưu trữ thông tin, người dùng sẽ được chuyển hướng sang trang ContactInfoPage, nơi thông tin đã nhập sẽ được hiển thị hoặc xử lý tiếp.

Tóm lại: Trang AddFriendPage trong ứng dụng Convenience_App có chức năng chính là thu thập thông tin về một người bạn mới từ người dùng và chuyển hướng đến một trang hiển thị thông tin chi tiết về người bạn đó - ContactInfoPage.

2.1.7. ContactInfo Page

ContactInfoPage là một trang đơn giản trong ứng dụng Convenience_App, được thiết kế để hiển thị thông tin chi tiết về một người bạn mới được thêm vào. Trang này nhận dữ liệu từ AddFriendPage và hiển thị thông tin như tên, email và số điện thoại của người bạn đó.

ContactInfoPage.xaml

```
<Grid>
    <Grid.Background>
        <ImageBrush ImageSource="/Assets/ContactInfor.png"/>
    </Grid.Background>
    <StackPanel Orientation="Vertical" HorizontalAlignment="Center"
    VerticalAlignment="Center">
        <TextBlock Text="Contact Information"/>
        <TextBlock x:Name="InforTextBlock" Text="Nơi hiển thị thông tin"/>
    </StackPanel>
</Grid>
```

Chi tiết các thành phần:

➤ Grid:

- Được sử dụng để tổ chức và bố trí các thành phần giao diện.
- Sử dụng ImageBrush để đặt hình nền từ đường dẫn /Assets/ContactInfor.png.

➤ StackPanel:

- Thành phần này sắp xếp các điều khiển con theo chiều dọc (Orientation="Vertical").
- Căn giữa nội dung theo chiều ngang và chiều dọc (HorizontalAlignment="Center", VerticalAlignment="Center").

➤ TextBlock:

- Hiển thị tiêu đề "Contact Information".
- Hiển thị thông tin chi tiết của người bạn trong InforTextBlock, nơi sẽ cập nhật nội dung dựa trên dữ liệu được truyền vào.

ContactInfoPage.xaml.cs

```
namespace Convenience_App
{
    public sealed partial class ContactInfoPage : Page
    {
        public ContactInfoPage()
        {
            this.InitializeComponent();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            var infoParameter = (Information)e.Parameter;
            if (infoParameter != null)
            {
                InforTextBlock.Text = $"Name: {infoParameter.Name}; Email: {infoParameter.Email}; Phone Number: {infoParameter.PhoneNumber}";
            }
            else
            {
                InforTextBlock.Text = "Dữ liệu bị trống, hãy nhập dữ liệu";
            }
        }
    }
}
```

Chi tiết các phương thức:

➤ Khởi tạo trang ContactInfoPage:

- Hàm khởi tạo ContactInfoPage ():

- Gọi phương thức `InitializeComponent()` để khởi tạo giao diện.

➤ Phương thức `OnNavigatedTo - Override`

`OnNavigatedTo(NavigationEventArgs e):`

- Gọi `base.OnNavigatedTo(e)` để đảm bảo rằng các xử lý điều hướng cơ bản được thực hiện.
- Lấy dữ liệu truyền vào: Sử dụng `e.Parameter` để lấy đối tượng `Information` được truyền từ `AddFriendPage`.
- Cập nhật `TextBlock`: Nếu dữ liệu không null, cập nhật `InforTextBlock.Text` với thông tin của người bạn (tên, email, số điện thoại). Nếu dữ liệu null, hiển thị thông báo "Dữ liệu bị trống, hãy nhập dữ liệu".

Tổng kết chức năng của `ContactInfoPage`:

`ContactInfoPage` có chức năng chính là hiển thị thông tin chi tiết về người bạn mới được thêm vào. Khi người dùng chuyển từ `AddFriendPage` sang `ContactInfoPage`, thông tin của người bạn sẽ được truyền qua và hiển thị trên trang này. Giao diện đơn giản và trực quan, dễ sử dụng và cho phép người dùng xem nhanh thông tin họ đã nhập.

2.2. CHẠY CODE CHƯƠNG TRÌNH

2.2.1. GIAO DIỆN CHƯƠNG TRÌNH

Chương trình gồm một giao diện chính:



Hình 1. Giao diện chính Home (Trang chủ)

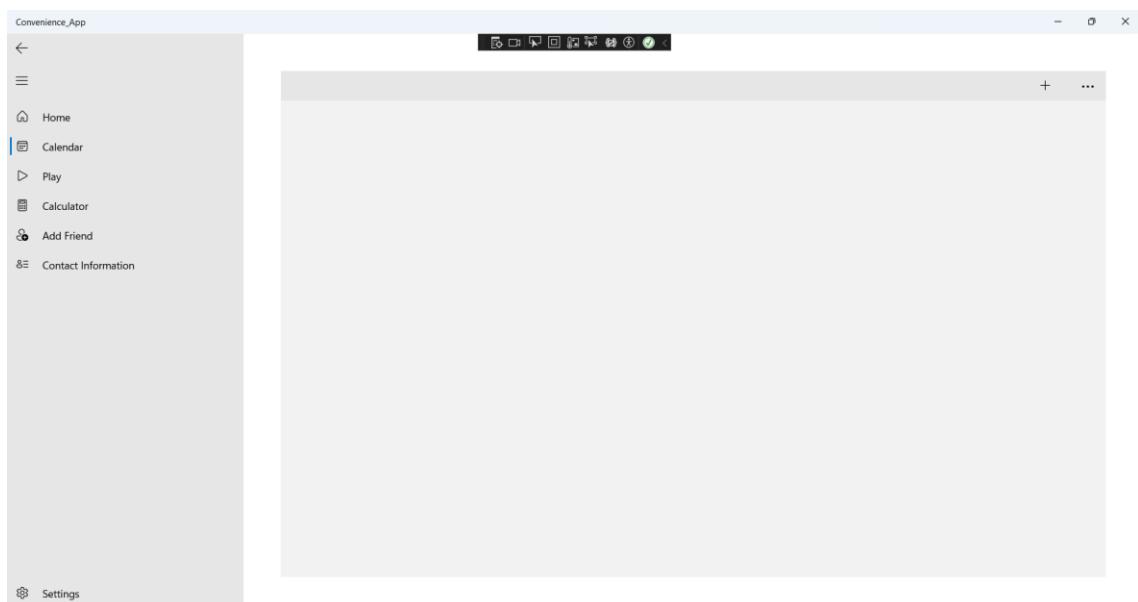
Khi mở ứng dụng Convenience App, bạn sẽ thấy một giao diện chính gồm hai phần chính: thanh điều hướng bên trái và khu vực hiển thị nội dung bên phải.

- ✚ Thanh điều hướng - Thanh điều hướng nằm ở bên trái màn hình, bao gồm các mục sau:
 - Home (Trang chủ): Trang mặc định hiển thị khi bạn khởi động ứng dụng.
 - Calendar (Lịch): Chuyển đến trang quản lý sự kiện và lịch của bạn.
 - Play (Trò chơi): Chuyển đến trang trò chơi Match Game.
 - Calculator (Máy tính): Chuyển đến trang máy tính đơn giản.
 - Add Friend (Thêm bạn bè): Chuyển đến trang thêm thông tin bạn bè.
 - Contact Information (Thông tin liên lạc): Chuyển đến trang hiển thị thông tin liên lạc bạn bè.
- ✚ Khu vực hiển thị nội dung: Phần này nằm ở bên phải và chiếm phần lớn diện tích màn hình, nơi nội dung của mỗi trang sẽ được hiển thị khi bạn chọn các mục từ thanh điều hướng.

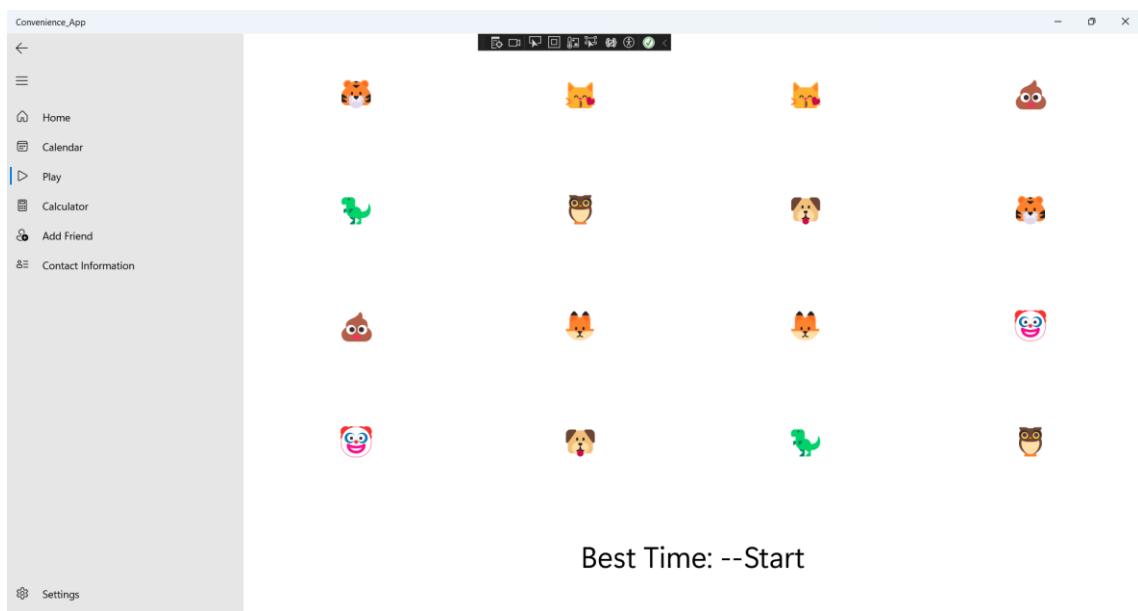
Hình ảnh chi tiết giao diện và các mục chính:



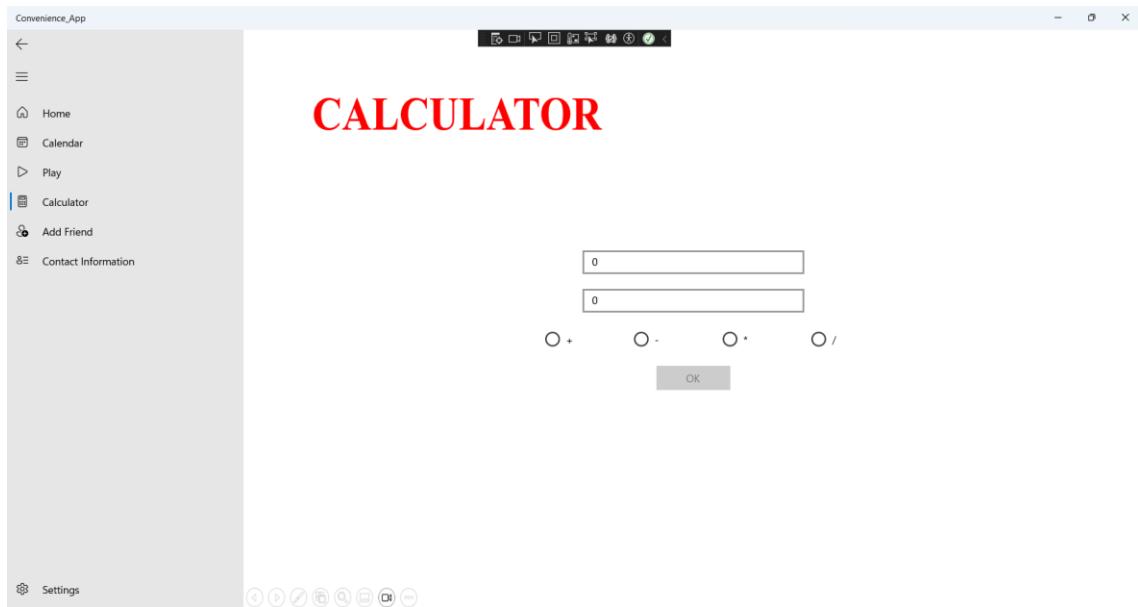
Hình 2. Giao diện tối ưu phần hiển thị nội dung



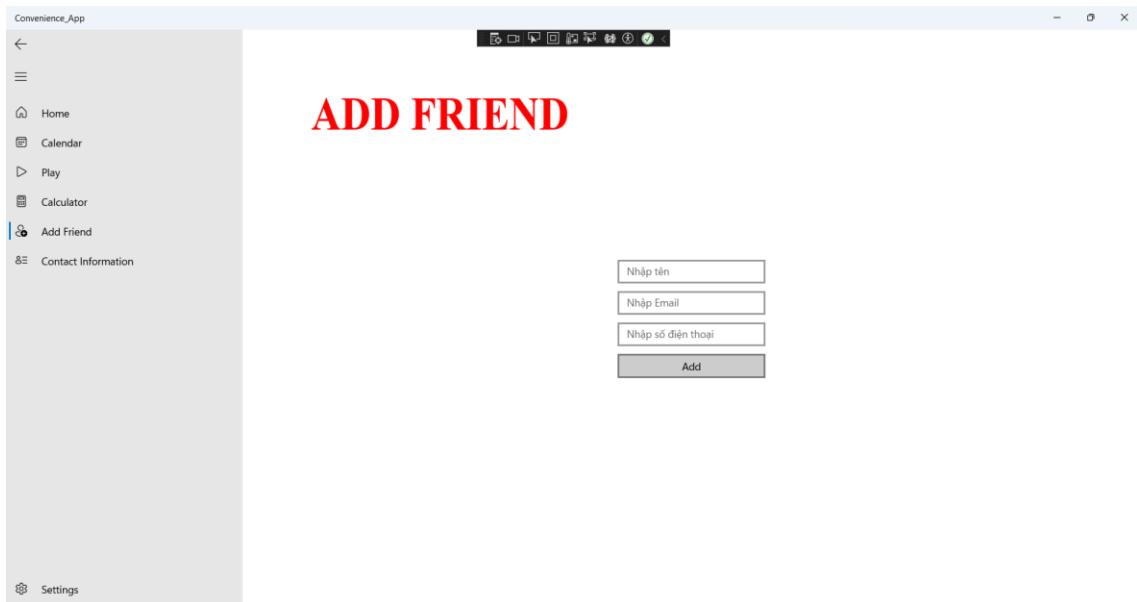
Hình 3. Giao diện Calendar (Lịch)



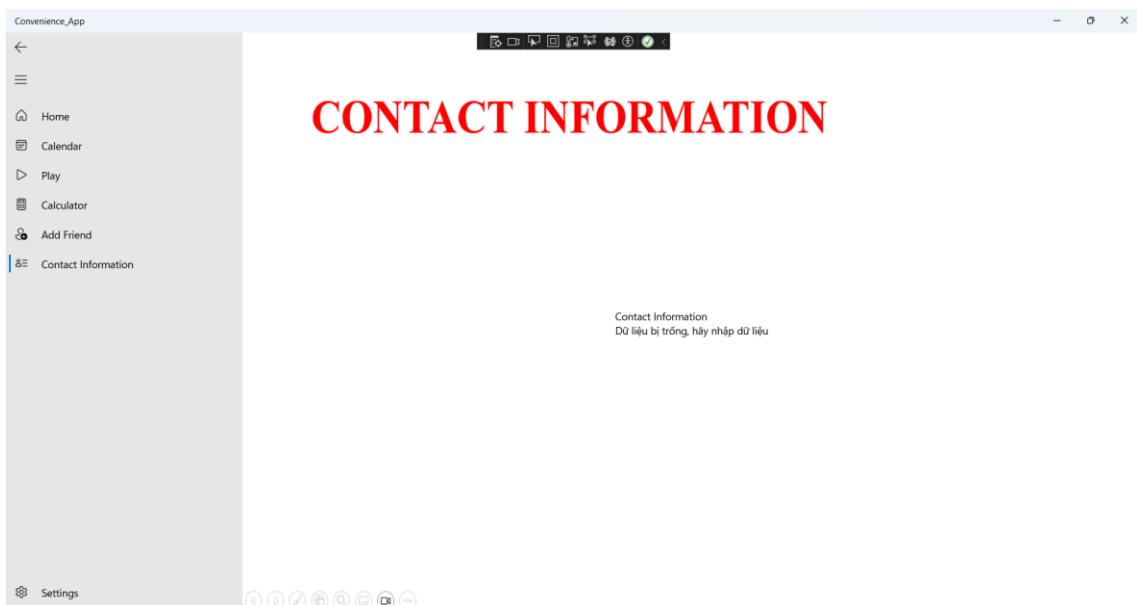
Hình 4. Giao diện Play (Trò chơi)



Hình 5. Giao diện Calculator (Máy tính)



Hình 6. Giao diện Add Friend (Thêm bạn bè)



Hình 7. Giao diện Contact Information (Thông tin liên lạc)

2.2.2. HƯỚNG DẪN SỬ DỤNG CHI TIẾT CÁC MỤC GIAO DIỆN

1. Home (Trang chủ):



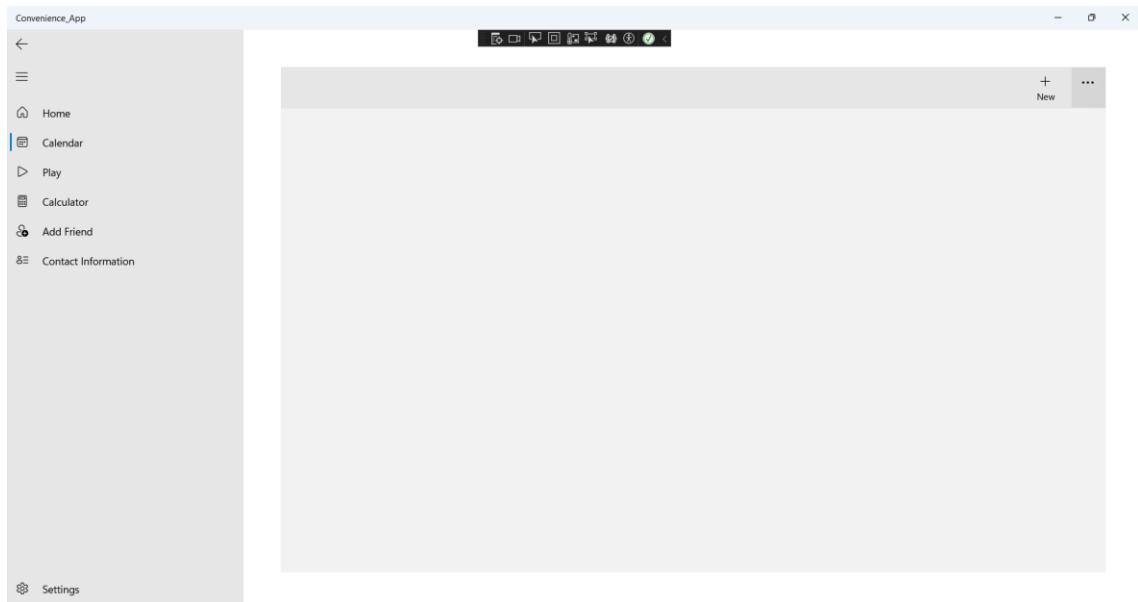
Hình 8. Khi nhấp vào "HOME" ở thanh điều hướng

Mục đích: Hiển thị thông tin tổng quan, chào mừng người dùng đến với ứng dụng Convenience App.

Cách sử dụng: Khi mở ứng dụng, bạn sẽ thấy trang chủ với các thông tin giới thiệu về ứng dụng.

2. Calendar (Lịch hẹn):

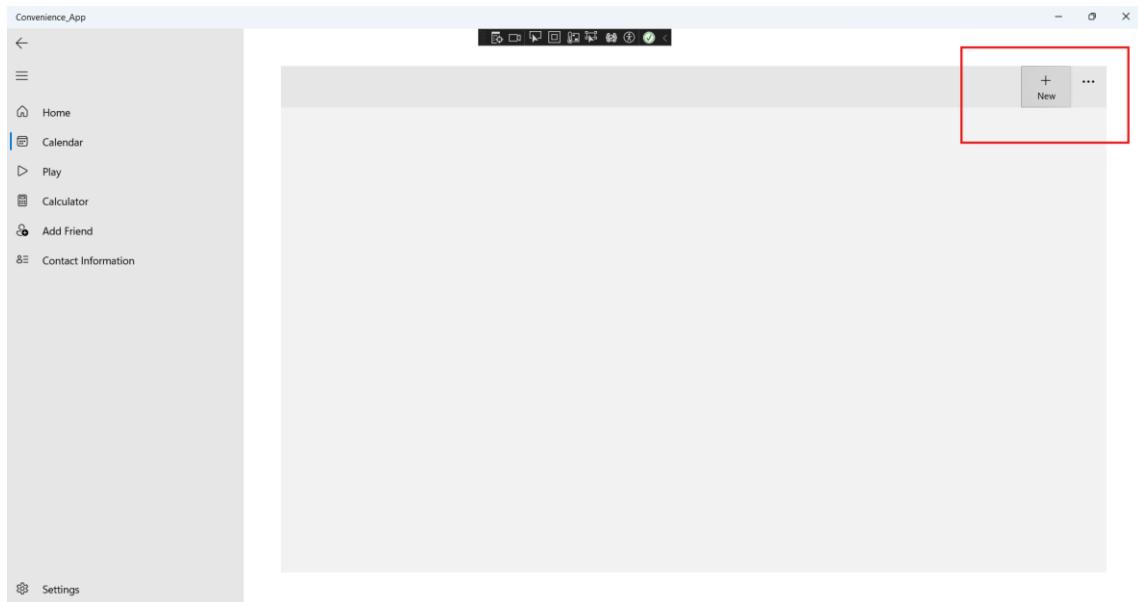
Chức năng Đặt lịch hẹn:



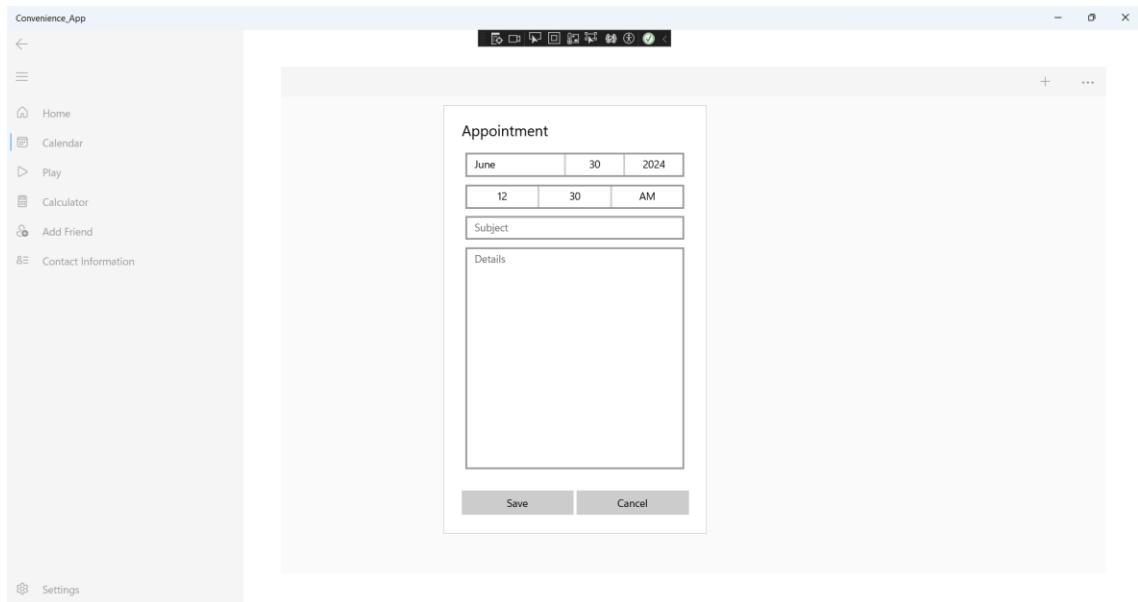
Hình 9. Khi nhấp vào "Calendar" ở thanh điều hướng

Thêm lịch hẹn:

- Người dùng có thể tạo mới một lịch hẹn bằng cách nhập thông tin chi tiết như tiêu đề, mô tả, ngày giờ bắt đầu và kết thúc.
- Ứng dụng sẽ lưu trữ và hiển thị lịch hẹn trong một danh sách hoặc trên lịch.

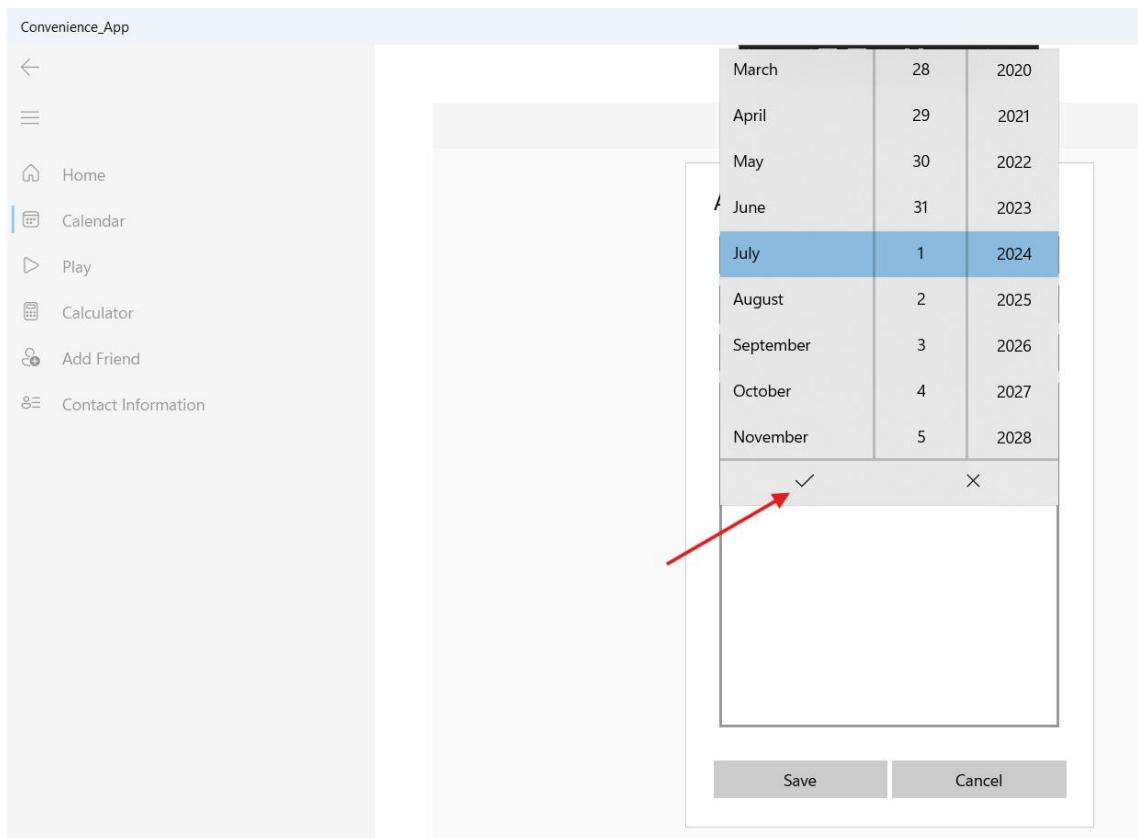


Hình 10. Nhấp vào nút "..." và chọn "New" ở góc phải màn hình để "Thêm lịch hẹn"

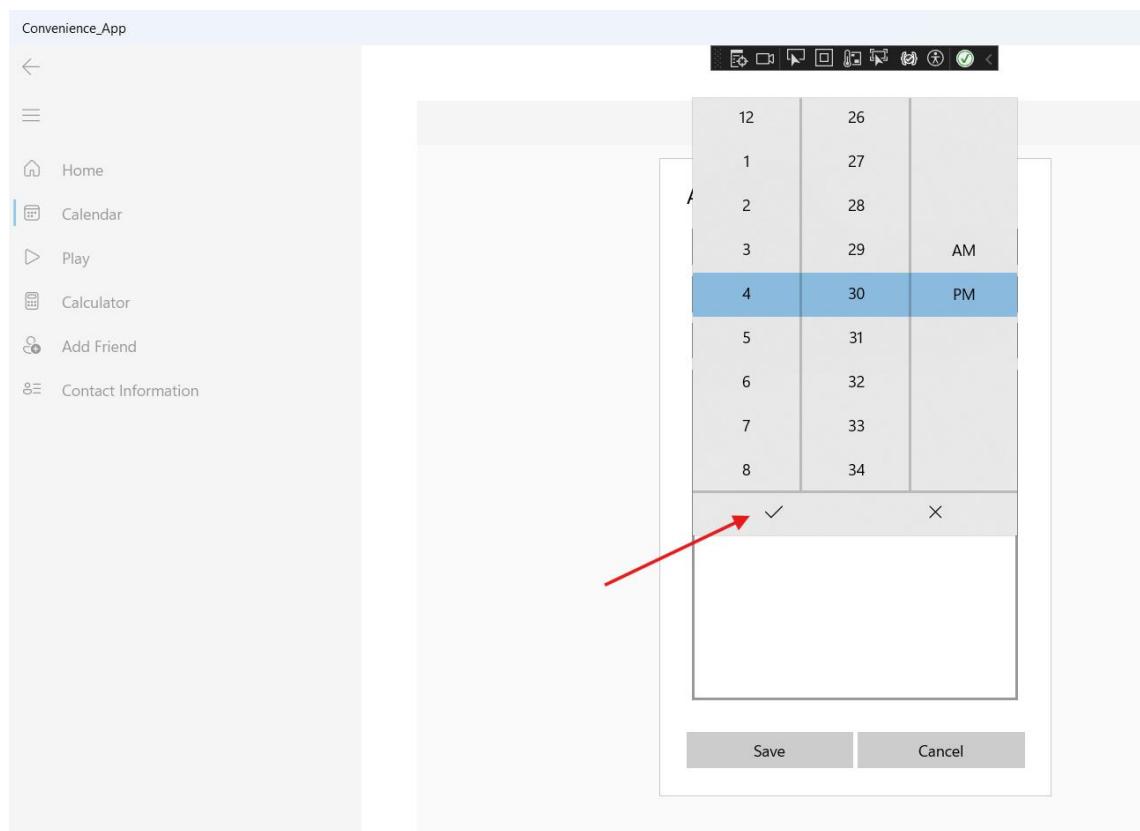


Hình 11. Thêm lịch hẹn

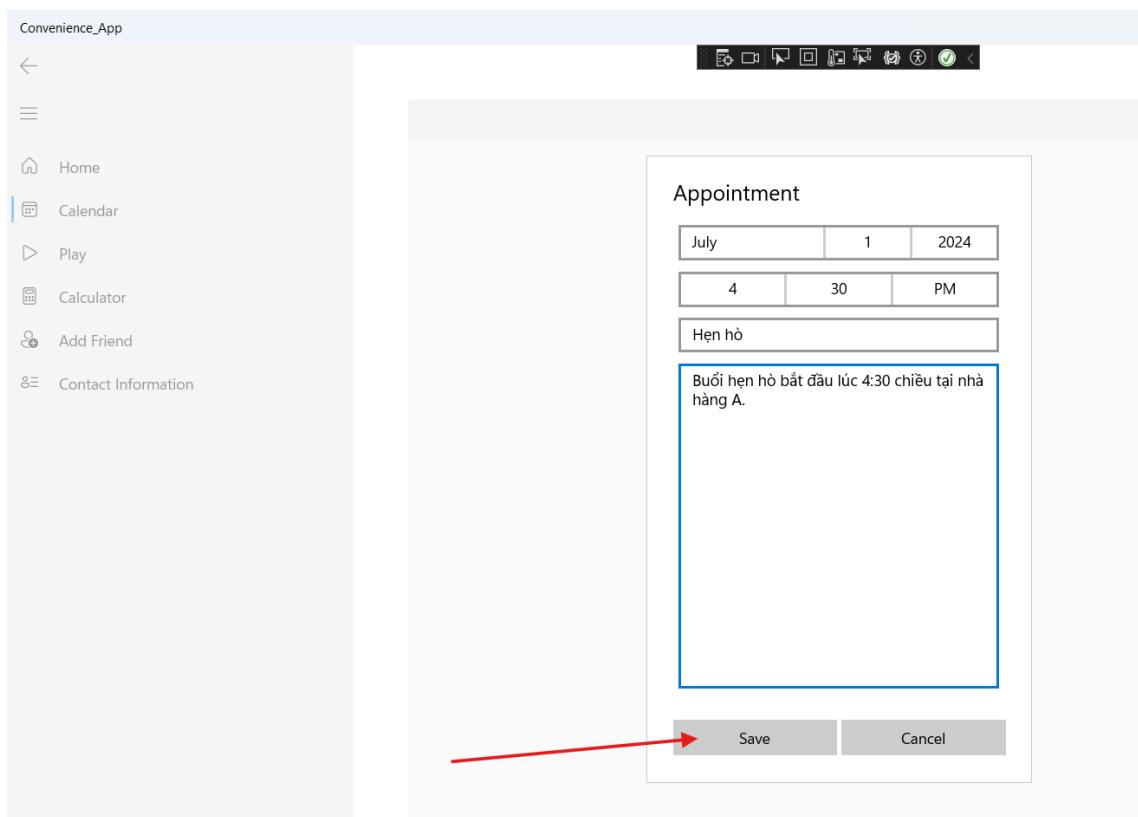
- Nhấn vào để điều chỉnh thời gian cuộc hẹn, ghi tên cuộc hẹn và mô tả chi tiết cuộc hẹn.



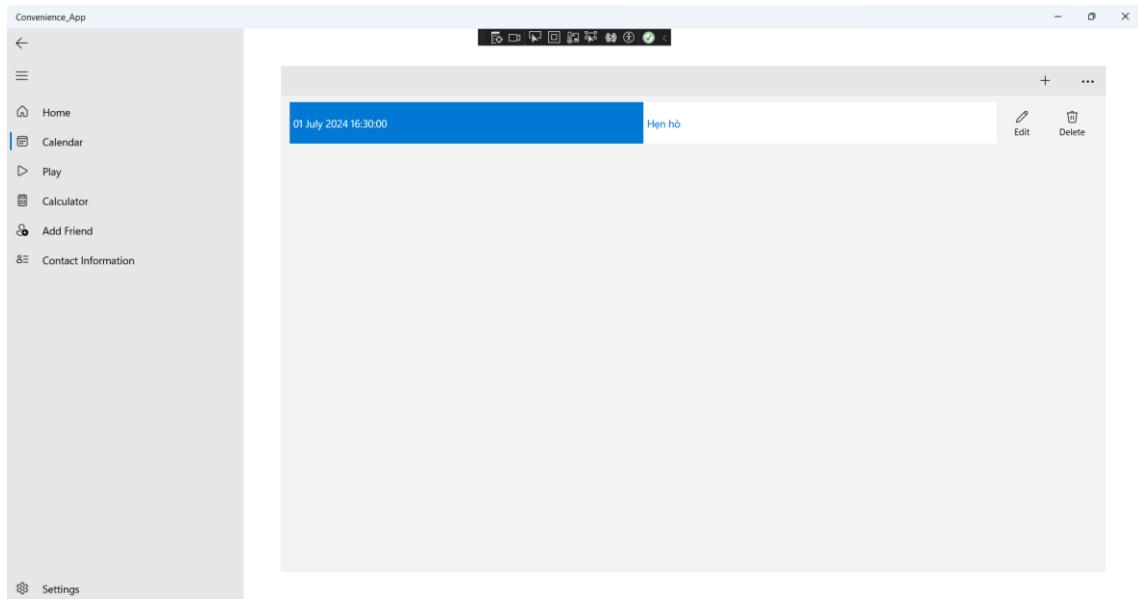
Hình 12. Điều chỉnh ngày hẹn và “ấn mũi tên” để chọn



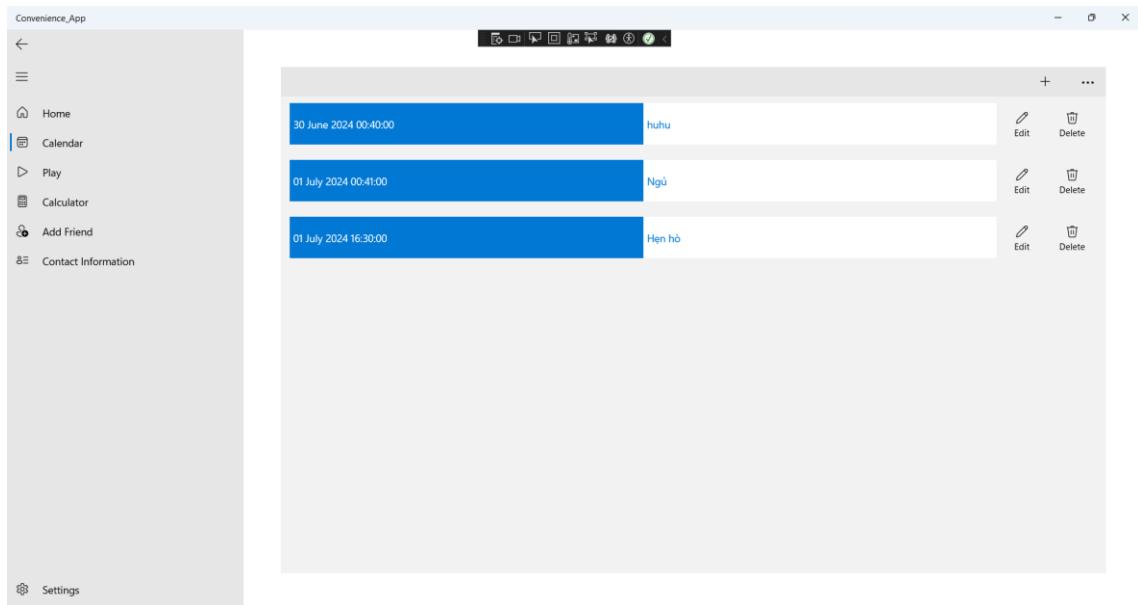
Hình 13. Điều chỉnh giờ hẹn và "ấn mũi tên" để chọn



Hình 14. Hoàn tất mô tả cuộc hẹn và ấn "Save" để lưu



Hình 15. Cuộc hẹn hoàn tất và bạn chỉ cần đợi thông báo nhắc nhở đến hẹn



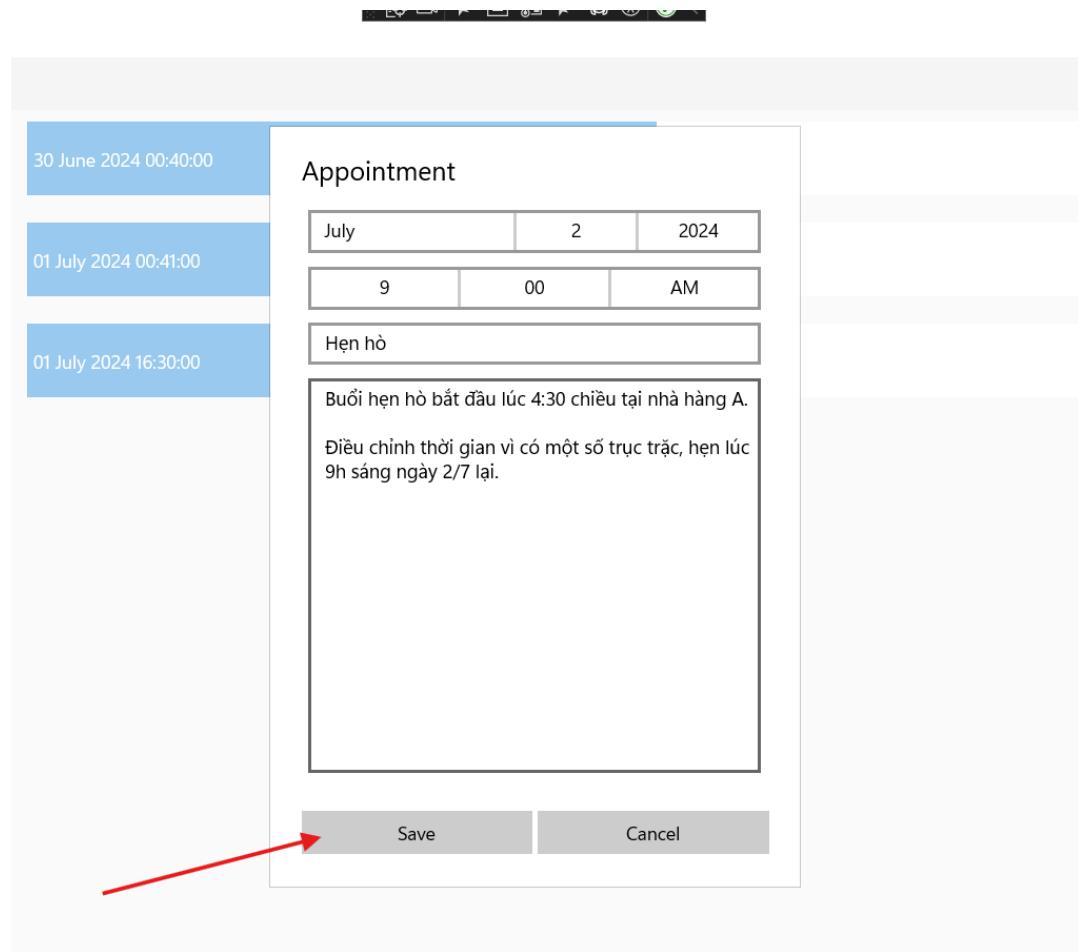
Hình 16. Bạn có thể thêm nhiều cuộc hẹn bằng các thao tác như trên

Chỉnh sửa lịch hẹn:

- Người dùng có thể chọn một lịch hẹn từ danh sách để chỉnh sửa thông tin.
- Các thay đổi sẽ được lưu lại và cập nhật trong danh sách lịch hẹn.



Hình 17. Nhấn vào "Edit" để chỉnh sửa lịch hẹn

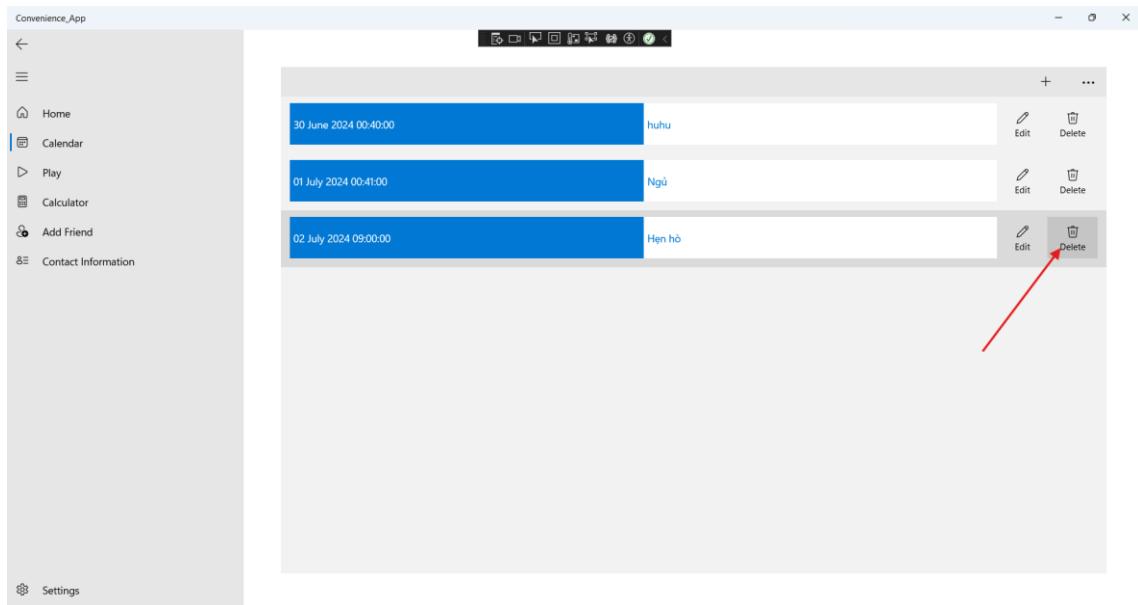


Hình 18. Điều chỉnh nội dung cuộc hẹn, nhớ ấn “Save” để lưu lại

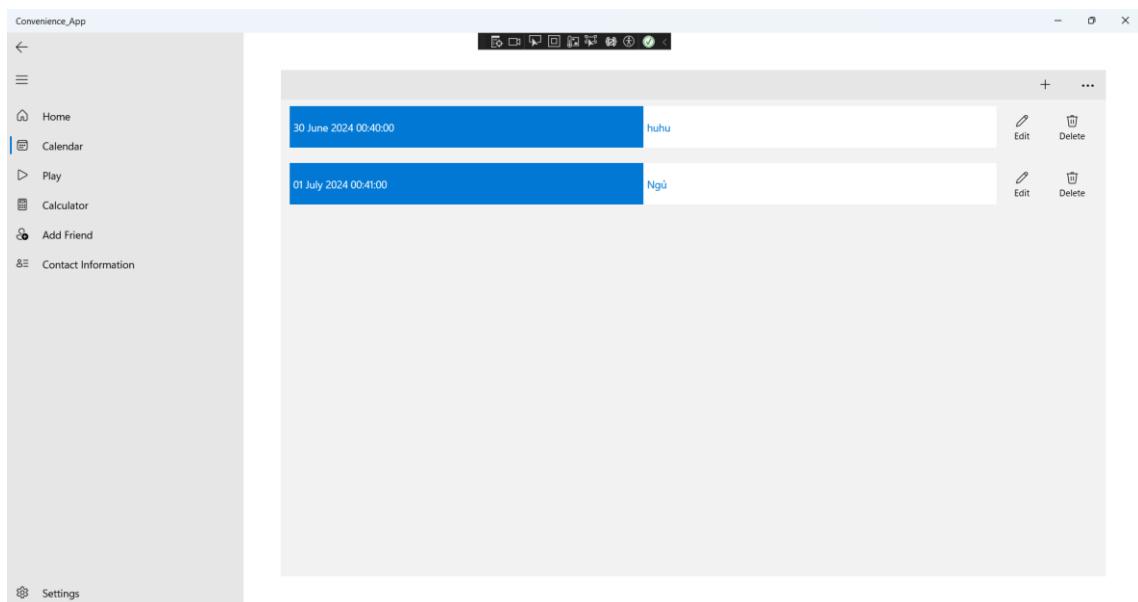


Hình 19. Cuộc hẹn được điều chỉnh đã được cập nhật

Xóa lịch hẹn: Người dùng có thể xóa một lịch hẹn đã chọn từ danh sách.



Hình 20. Nhấn "Delete" để xóa cuộc hẹn



Hình 21. Lịch hẹn đã được xóa

Nhắc nhở lịch hẹn: Ứng dụng sẽ gửi thông báo nhắc nhở trước thời gian diễn ra lịch hẹn dựa trên thiết lập của người dùng.

3. Play (Trò chơi Match Game):

Chức năng Trò chơi Match Game:

 Thiết kế trò chơi:

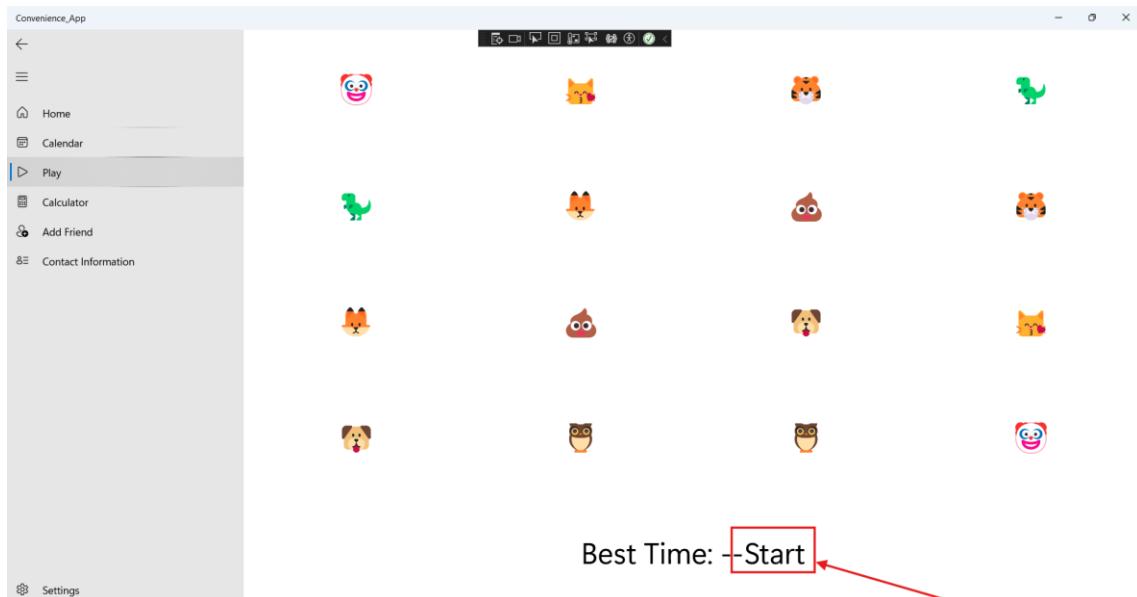
- Trò chơi Match Game sẽ bao gồm một lưới gồm 16 ô (4 hàng x 4 cột) chứa các emoji.

 Luật chơi:

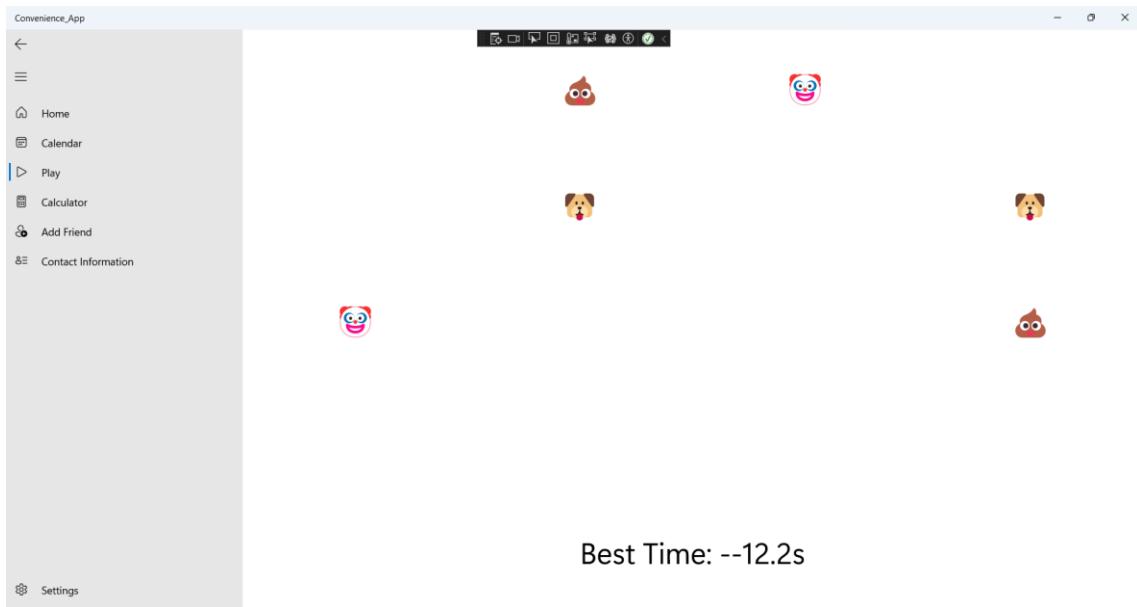
- Người chơi click chuột vào 2 emoji.
- Nếu 2 emoji được nhấn liên tục trùng nhau thì sẽ ẩn 2 TextBlock chứa hai hình đó.
- Nếu không trùng nhau sẽ không được xóa bỏ.

 Bộ đếm thời gian và hiển thị điểm:

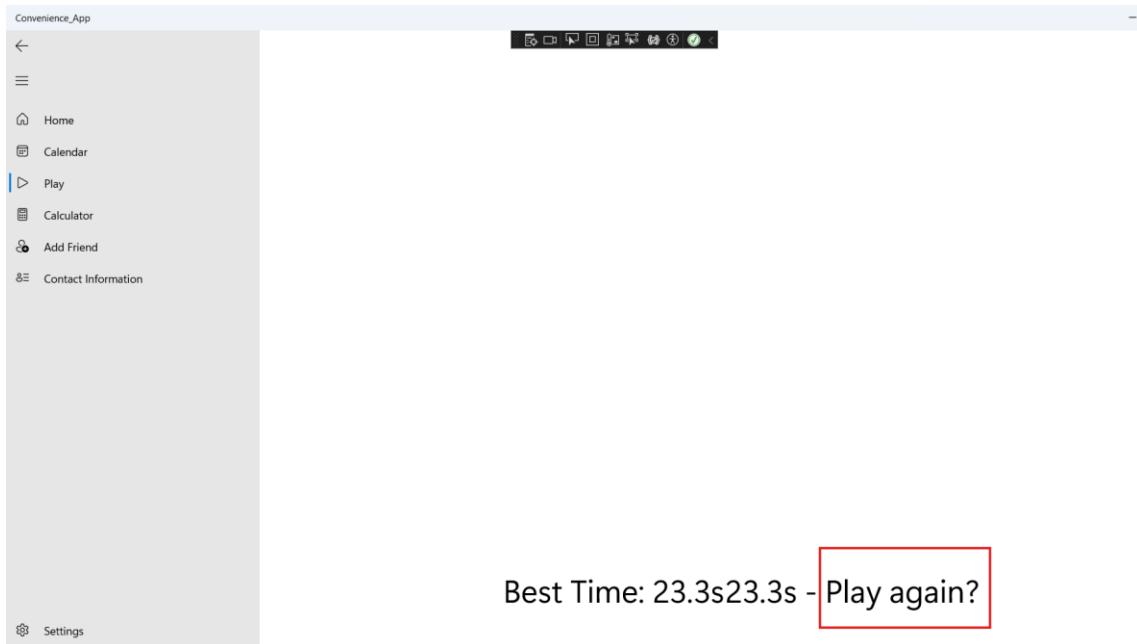
- Trò chơi sẽ có một bộ đếm thời gian để người chơi theo dõi thời gian chơi.
- Hiển thị thời gian tốt nhất (ngắn nhất) mà người chơi từng đạt được để tạo cảm hứng cạnh tranh.



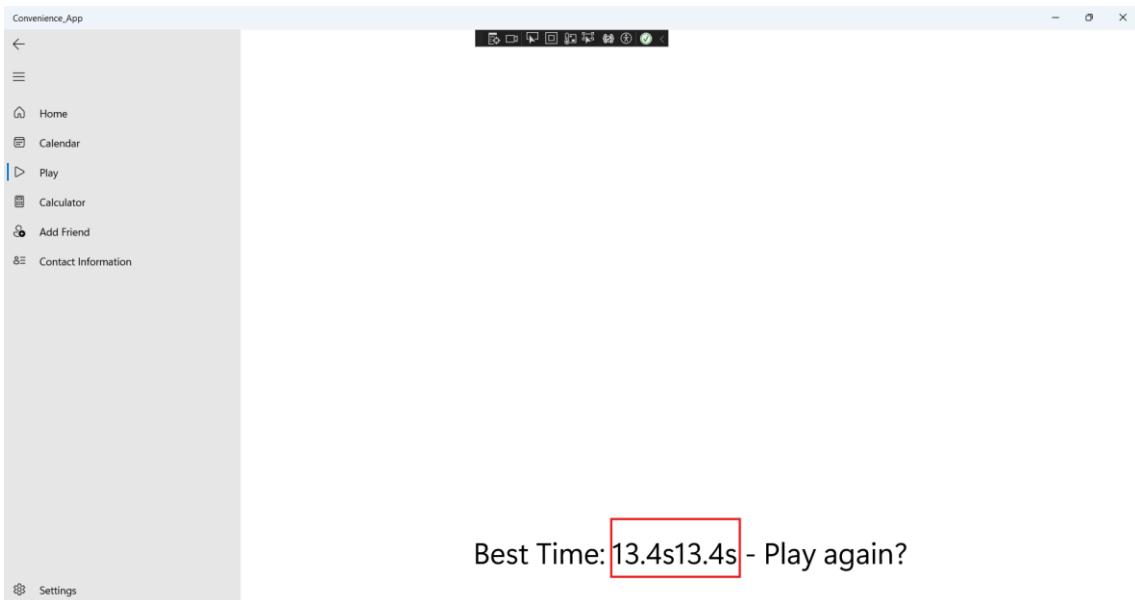
Hình 22. Khi nhấn vào "Play" ở thanh điều hướng và nhấn "Start" để bắt đầu chơi



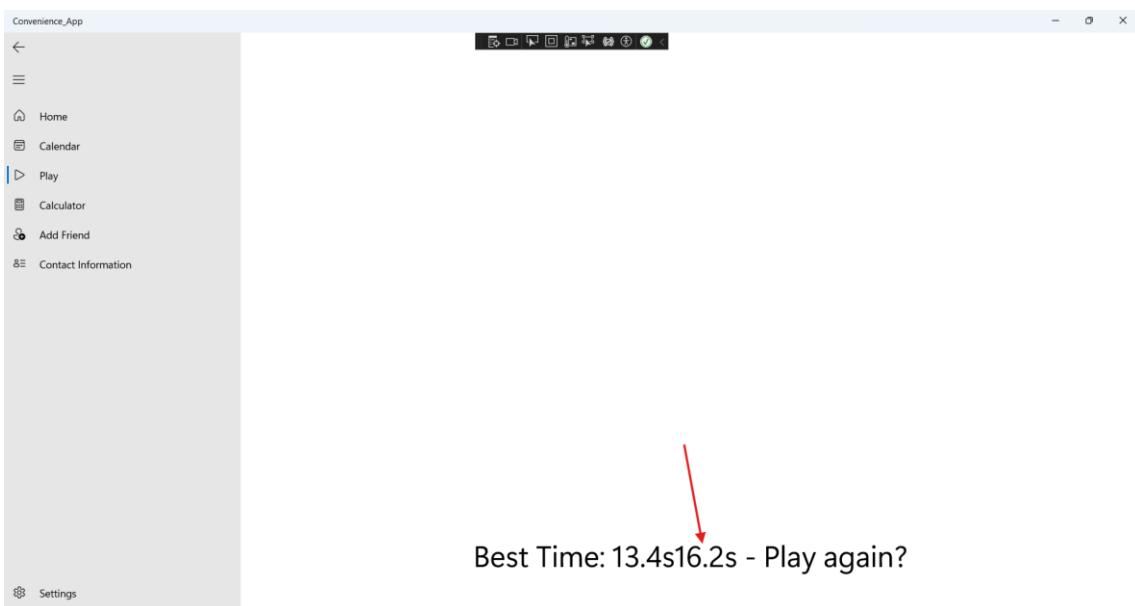
Hình 23. Bấm "Start" và tiến hành chơi theo luật chơi, chương trình sẽ tính thời gian chơi của bạn



Hình 24. Kết thúc trò chơi khi bạn đã chọn đúng hết, quan sát thời gian hoàn thành, nhấn "Play again?" nếu muốn cải thiện thời gian chơi



Hình 25. Hoàn thành chơi lại trò chơi và thời gian tốt nhất được cập nhật



Hình 26. Tiến hành "Play again?" nhưng thời gian mới không tốt nên không được cập nhật

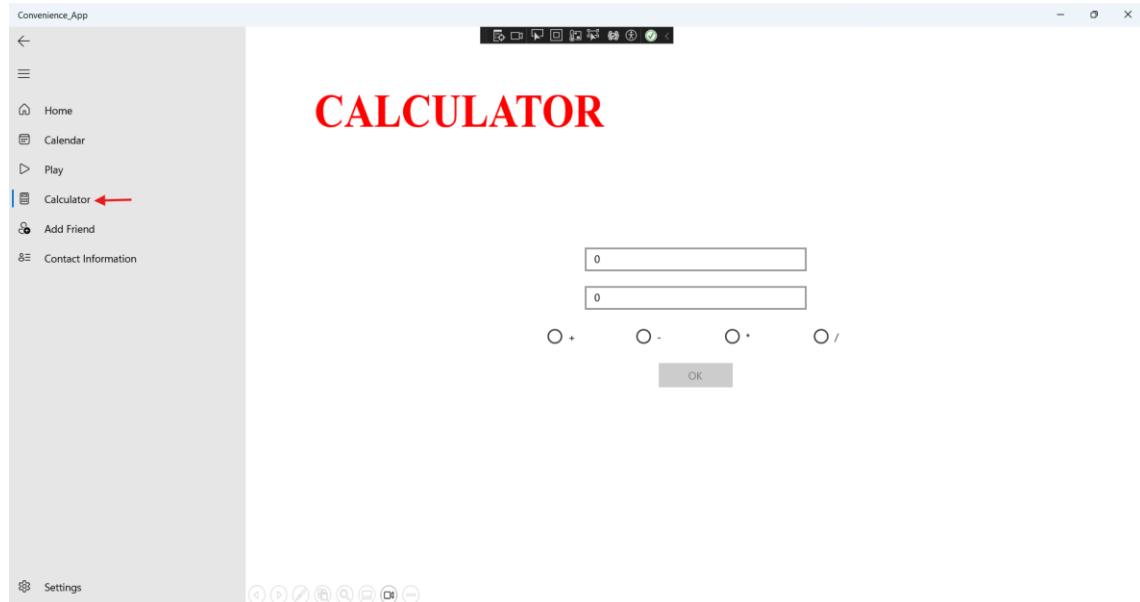
4. Calculator (Máy tính):

✚ Mục đích: Cung cấp công cụ tính toán đơn giản.

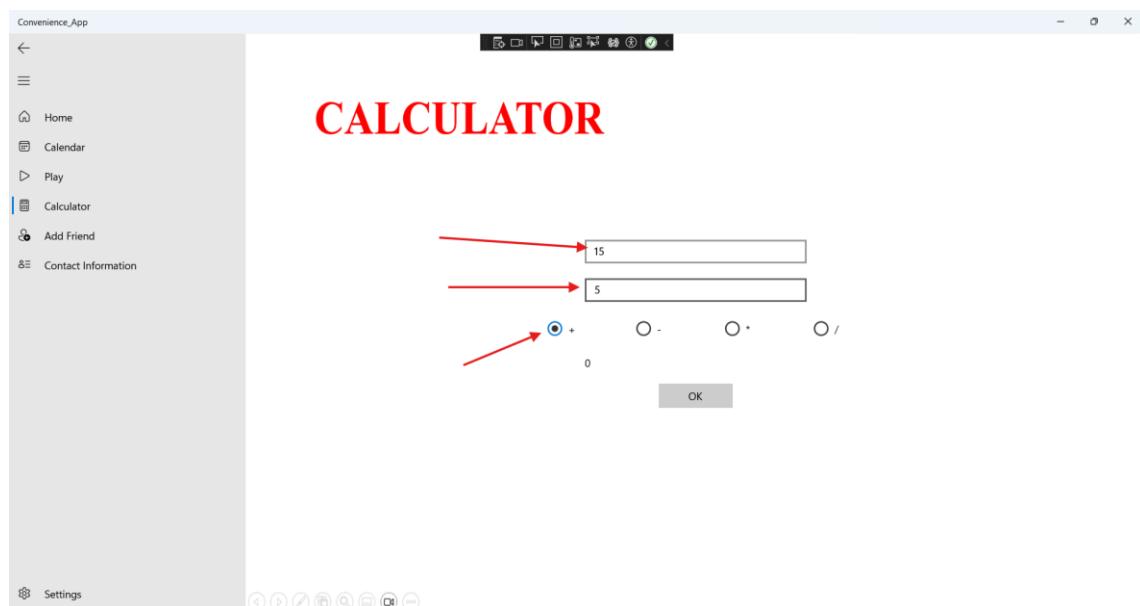
✚ Cách sử dụng:

- Chọn mục "Calculator" từ thanh điều hướng.

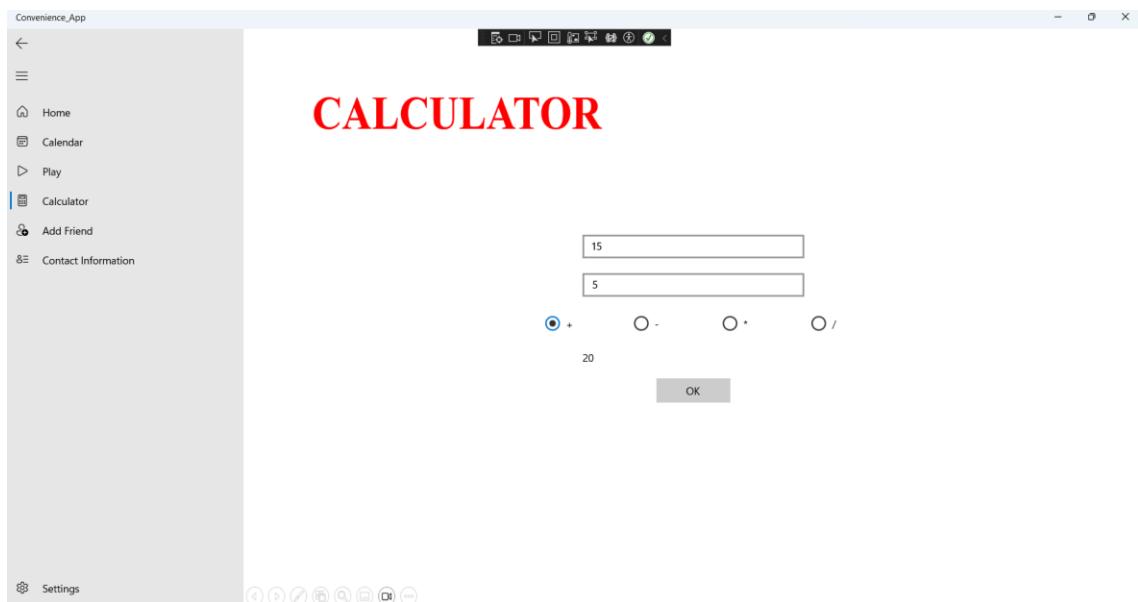
- Bạn sẽ thấy giao diện máy tính với các ô nhập số và các tùy chọn phép tính (+, -, *, /).
- Nhập số vào các ô và chọn phép tính, sau đó nhấn nút "OK" để xem kết quả.



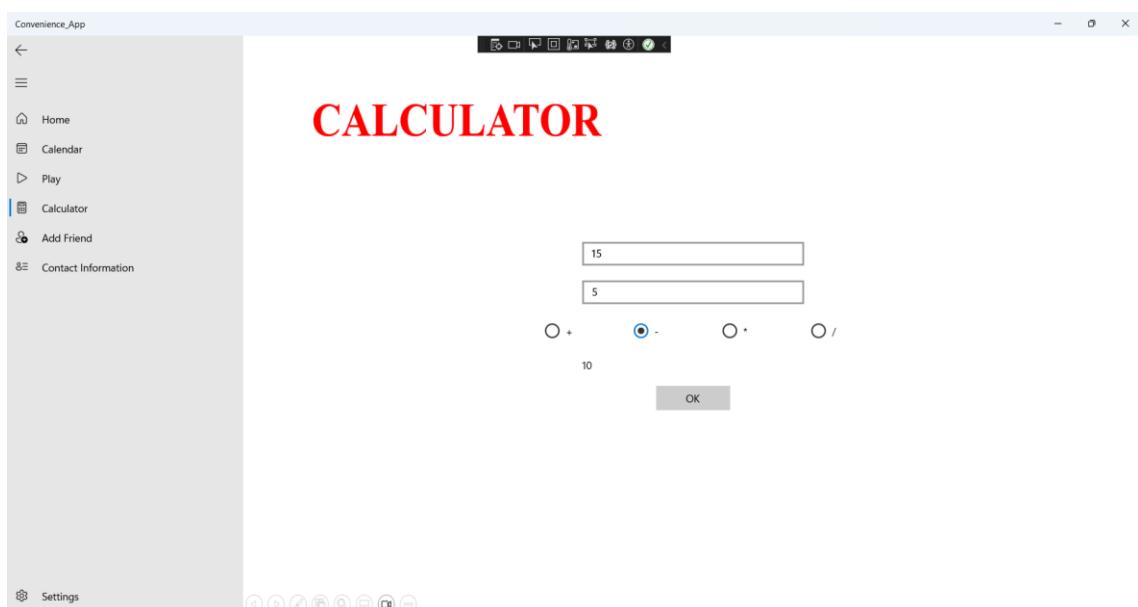
Hình 27. Khi nhấn "Calculator" ở thanh điều hướng



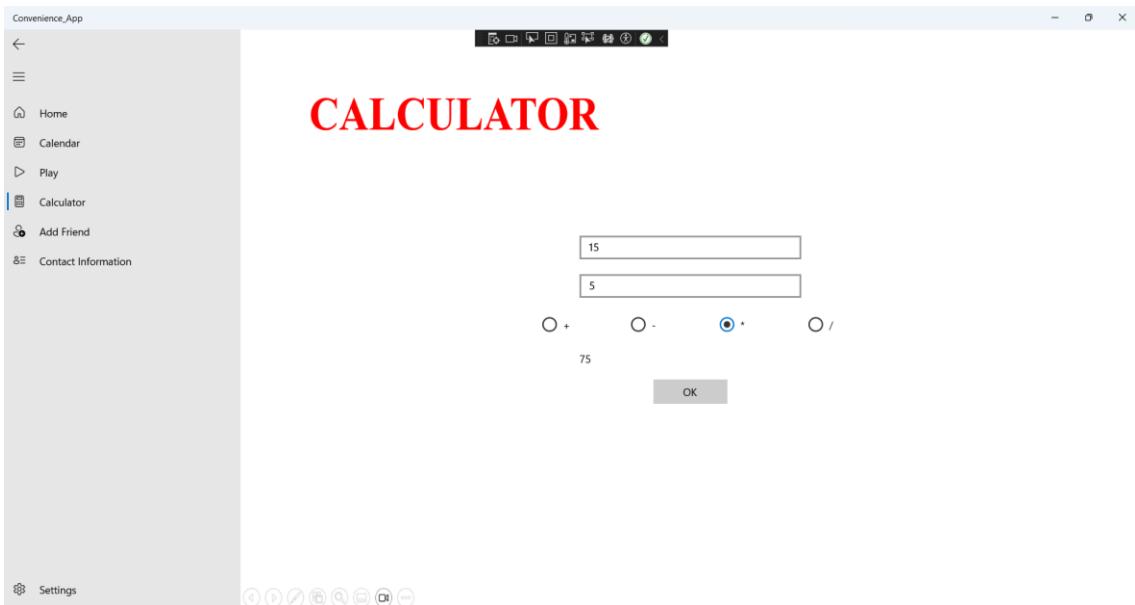
Hình 28. Nhập vào giá trị cần tính toán, chọn phép tính muốn thực hiện và nhấn "OK" xem kết quả



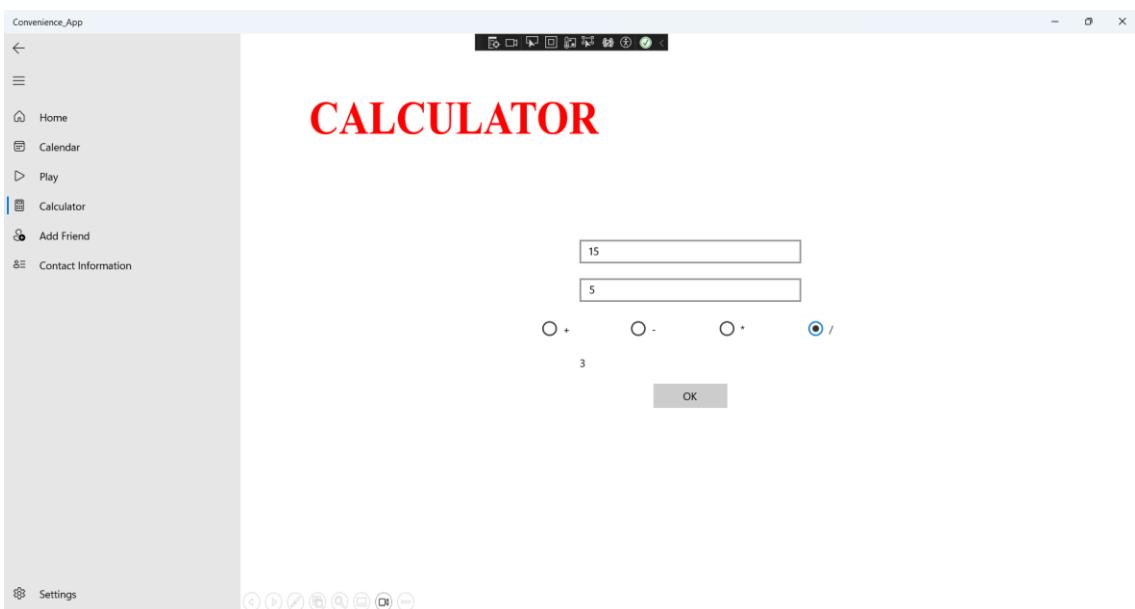
Hình 29. Kết quả khi nhấn "OK"



Hình 30. Chọn sang phép trừ nhấn "OK" để xem kết quả



Hình 31. Chọn sang phép nhân nháń "OK" để xem kết quả



Hình 32. Chọn sang phép chia nháń "OK" để xem kết quả

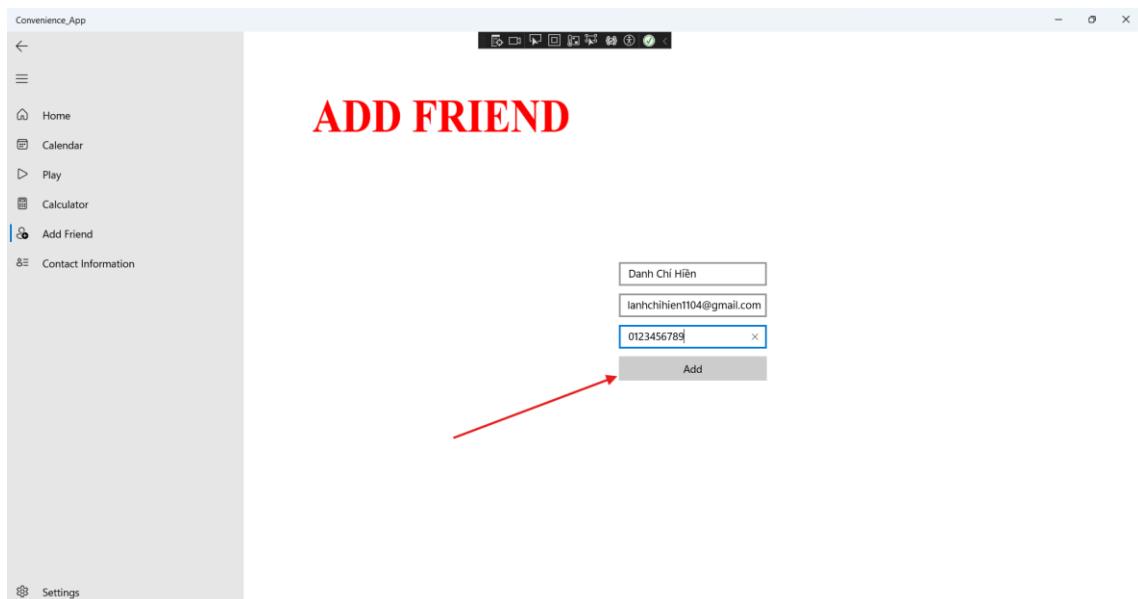
5. Add Friend (Thêm bạn bè):

- ➡ Mục đích: Nhập thông tin liên lạc của bạn bè.
- ➡ Cách sử dụng:
 - Chọn mục "Add Friend" từ thanh điều hướng.
 - Bạn sẽ thấy các ô nhập liệu cho tên, email và số điện thoại.

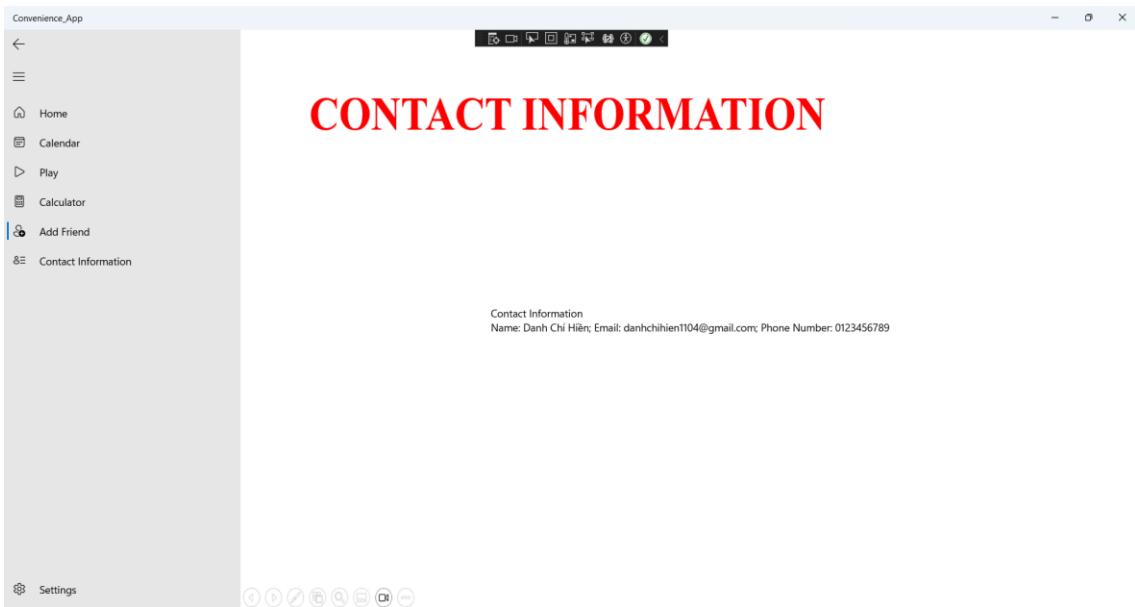
- Nhập thông tin vào các ô và nhấn nút "Add" để lưu thông tin.
- Thông tin sẽ được chuyển đến trang "Contact Information".



Hình 33. Nhấn "Add Friend" ở thanh điều hướng



Hình 34. Nhập thông tin vào các ô và nhấn "Add"

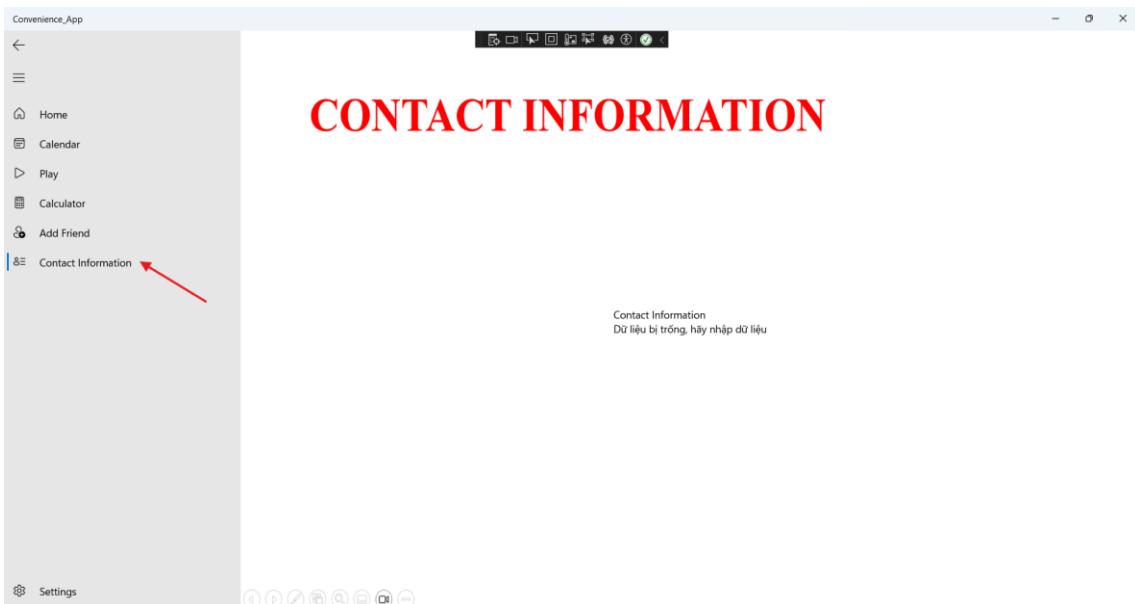


Hình 35. Thông tin sẽ được chuyển đến trang "Contact Information"

6. Contact Information (Thông tin liên lạc):

- ✚ Mục đích: Hiển thị thông tin liên lạc bạn bè đã thêm.
- ✚ Cách sử dụng:

- Chọn mục "Contact Information" từ thanh điều hướng.
- Thông tin liên lạc bạn bè đã thêm sẽ được hiển thị tại đây.



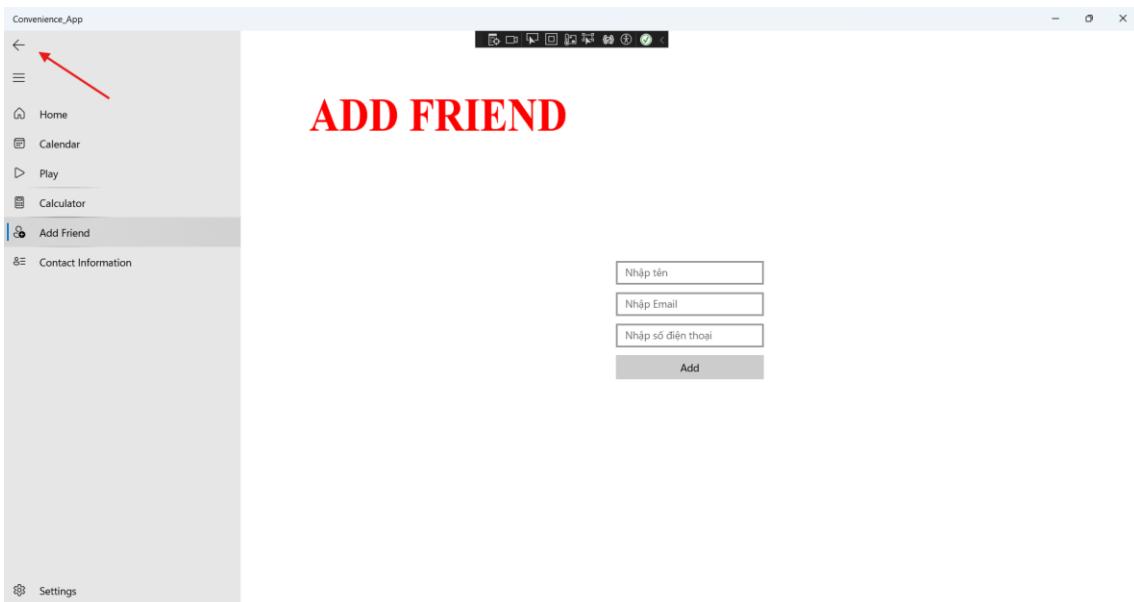
Hình 36. Khi nhấn "Contact Information" từ thanh điều hướng

7. Điều hướng quay lại:

⊕ Mục đích: Trở lại trang trước đó.

⊕ Cách sử dụng:

- Sử dụng nút quay lại trên thanh điều hướng hoặc hệ thống (nếu có).
- Nếu có trang trước đó, bạn sẽ được chuyển trở lại trang đó.



Hình 37. Nhấn vào mũi tên góc phải màn hình để điều hướng quay lại

PHẦN II: ĐỘ HOÀN THIỆN CỦA CHƯƠNG TRÌNH

2.3. ƯU ĐIỂM

Giao diện người dùng của Convenience App được thiết kế với sự đơn giản và tính trực quan cao, nhằm mang lại trải nghiệm sử dụng dễ dàng và tiện lợi cho người dùng. Dưới đây là các điểm nổi bật khi chạy giao diện của ứng dụng:

⊕ Tính nhất quán và dễ sử dụng:

- Thanh điều hướng: Nằm ở vị trí cố định bên trái, giúp người dùng dễ dàng truy cập vào các chức năng chính của ứng dụng.
- Các mục điều hướng rõ ràng: Mỗi mục trên thanh điều hướng đều được gắn nhãn và biểu tượng dễ nhận biết, giúp người dùng nhanh chóng hiểu được chức năng của từng mục.

 Tính trực quan và phản hồi nhanh:

- Khu vực hiển thị nội dung: Phần lớn màn hình được dành cho nội dung, thay đổi linh hoạt theo mục được chọn từ thanh điều hướng.
- Hiệu ứng chuyển trang: Các hiệu ứng chuyển trang mượt mà tạo cảm giác liền mạch và thân thiện khi người dùng điều hướng giữa các trang.

 Tính năng đa dạng và hữu ích:

- Trang Home: Cung cấp thông tin giới thiệu hoặc tổng quan về ứng dụng.
- Trang Calendar: Cho phép người dùng thêm, chỉnh sửa, xóa và nhận nhắc nhở về các lịch hẹn một cách dễ dàng.
- Trang Play (Trò chơi Match Game): Một trò chơi đơn giản nhưng thú vị, giúp giải trí và thử thách người chơi.
- Trang Calculator: Cung cấp công cụ tính toán cơ bản, hỗ trợ người dùng trong các phép toán hàng ngày.
- Trang Add Friend: Cho phép người dùng nhập và lưu thông tin liên lạc của bạn bè.
- Trang Contact Information: Hiển thị thông tin liên lạc đã lưu, giúp người dùng dễ dàng tra cứu và quản lý.

 Điều hướng quay lại:

- Nút quay lại: Giúp người dùng trở về trang trước đó một cách dễ dàng, tăng cường tính tiện dụng và linh hoạt.

 Tổng kết

- Giao diện người dùng của Convenience App được thiết kế một cách có hệ thống và trực quan, tạo điều kiện thuận lợi cho người dùng truy cập và sử dụng các tính năng của ứng dụng. Sự rõ ràng trong cấu trúc và điều hướng giúp người dùng dễ dàng làm quen và thao tác, từ đó nâng cao trải nghiệm tổng thể khi sử dụng ứng dụng.

2.4. HẠN CHẾ

 Tính năng hạn chế:

➤ Chức năng Calendar:

Thiếu tính năng đồng bộ: Hiện tại, ứng dụng không có chức năng đồng bộ lịch hẹn với các ứng dụng lịch khác (như Google Calendar, Outlook).

Giới hạn trong nhắc nhở: Chức năng nhắc nhở đơn giản, chưa cho phép tùy chỉnh linh hoạt như nhắc nhở theo khoảng thời gian trước khi sự kiện bắt đầu.

➤ Chức năng Trò chơi:

Số lượng trò chơi: Ứng dụng chỉ có một trò chơi duy nhất, giới hạn sự lựa chọn và trải nghiệm của người dùng.

Thiếu tính năng đa người chơi: Trò chơi Match Game không hỗ trợ chế độ đa người chơi, giảm tính cạnh tranh và tương tác xã hội.

➤ Chức năng Calculator:

Chỉ hỗ trợ các phép toán cơ bản: Máy tính chỉ hỗ trợ các phép toán cơ bản, thiếu các phép toán phức tạp như lũy thừa, căn bậc hai, và các hàm toán học khác.

✚ Giao diện người dùng:

Thiếu tùy chỉnh giao diện: Người dùng không thể tùy chỉnh giao diện theo sở thích cá nhân, làm giảm tính cá nhân hóa.

Thiết kế giao diện đơn giản: Giao diện hiện tại mặc dù dễ sử dụng nhưng khá đơn giản, chưa thực sự hấp dẫn và cuốn hút người dùng.

✚ Hiệu năng và trải nghiệm người dùng:

Hiệu năng trên thiết bị cấu hình thấp: Ứng dụng có thể gặp vấn đề về hiệu năng trên các thiết bị có cấu hình thấp, dẫn đến trải nghiệm người dùng không mượt mà.

Thiếu tối ưu cho các thiết bị khác nhau: Ứng dụng chưa được tối ưu hóa cho nhiều loại thiết bị khác nhau, đặc biệt là các thiết bị có kích thước màn hình khác nhau.

✚ Khả năng mở rộng và bảo trì:

Khả năng mở rộng hạn chế: Thiết kế hiện tại của ứng dụng có thể gặp khó khăn khi cần mở rộng tính năng hoặc thay đổi cấu trúc dữ liệu.

Khả năng bảo trì: Mã nguồn chưa được tổ chức và chú thích rõ ràng, có thể gây khó khăn cho việc bảo trì và phát triển sau này.

 **Bảo mật và quyền riêng tư:**

Bảo mật dữ liệu: Ứng dụng chưa có cơ chế bảo mật mạnh mẽ cho dữ liệu người dùng, như mã hóa thông tin liên lạc hoặc bảo vệ dữ liệu cá nhân.

Quyền riêng tư: Chưa có chính sách rõ ràng về quyền riêng tư và cách thức ứng dụng xử lý dữ liệu cá nhân của người dùng.

2.5. ĐỀ XUẤT HƯỚNG PHÁT TRIỂN

 **Tăng cường tính năng:**

➤ **Lịch hẹn:**

Đồng bộ hóa: Phát triển chức năng đồng bộ lịch hẹn với các ứng dụng lịch khác như Google Calendar, Outlook.

Nhắc nhở linh hoạt: Thêm tùy chọn nhắc nhở linh hoạt, cho phép người dùng thiết lập nhắc nhở theo khoảng thời gian trước khi sự kiện bắt đầu.

➤ **Trò chơi:**

Thêm trò chơi mới: Phát triển thêm nhiều trò chơi khác để tăng tính giải trí và thu hút người dùng.

Chế độ đa người chơi: Thêm chức năng đa người chơi cho trò chơi Match Game để tăng tính cạnh tranh và tương tác xã hội.

➤ **Máy tính:**

Phép toán nâng cao: Nâng cấp máy tính với các phép toán phức tạp hơn như lũy thừa, căn bậc hai, và các hàm toán học khác.

 **Cải thiện giao diện người dùng:**

Tùy chỉnh giao diện: Cho phép người dùng tùy chỉnh giao diện theo sở thích cá nhân.

Nâng cấp thiết kế giao diện: Làm mới giao diện với thiết kế hiện đại, hấp dẫn hơn để cải thiện trải nghiệm người dùng.

Tối ưu hiệu năng và trải nghiệm người dùng:

Tối ưu hiệu năng: Tối ưu hóa mã nguồn để cải thiện hiệu năng, đặc biệt trên các thiết bị có cấu hình thấp.

Đa dạng hóa thiết bị hỗ trợ: Tối ưu hóa ứng dụng để hoạt động tốt trên nhiều loại thiết bị khác nhau, bao gồm cả các thiết bị có kích thước màn hình khác nhau.

Khả năng mở rộng và bảo trì:

Cải thiện cấu trúc mã nguồn: Tổ chức và chú thích mã nguồn rõ ràng hơn để dễ dàng mở rộng và bảo trì.

Sử dụng các mẫu thiết kế: Áp dụng các mẫu thiết kế phần mềm (design patterns) để cải thiện khả năng mở rộng và bảo trì.

Bảo mật và quyền riêng tư:

Cơ chế bảo mật: Phát triển các cơ chế bảo mật mạnh mẽ như mã hóa dữ liệu và bảo vệ thông tin cá nhân của người dùng.

Chính sách quyền riêng tư: Xây dựng và công khai chính sách quyền riêng tư rõ ràng để người dùng hiểu cách thức ứng dụng xử lý dữ liệu cá nhân của họ.

Tổng kết:

Bằng cách khắc phục những hạn chế và phát triển thêm các tính năng mới, Convenience App sẽ trở nên toàn diện hơn, đáp ứng tốt hơn nhu cầu đa dạng của người dùng, và cung cấp trải nghiệm sử dụng tối ưu.

CHƯƠNG 3: TỔNG KẾT ĐỀ TÀI

3.1. ĐÁNH GIÁ ĐỀ TÀI

Tổng quan

Convenience App là một ứng dụng UWP (Universal Windows Platform) nhằm cung cấp một tập hợp các chức năng tiện ích đa dạng cho người dùng, bao gồm quản lý lịch hẹn, trò chơi giải trí, máy tính cơ bản và quản lý liên hệ. Đây là một dự án thể hiện sự tích hợp nhiều tính năng trong một ứng dụng duy nhất, giúp người dùng tiết kiệm thời gian và thuận tiện hơn trong việc quản lý các hoạt động hàng ngày.

Ưu điểm

➤ Tính đa năng và tích hợp:

Nhiều tính năng hữu ích: Ứng dụng cung cấp nhiều chức năng khác nhau như lịch hẹn, trò chơi, máy tính và quản lý liên hệ, giúp người dùng có thể sử dụng một ứng dụng duy nhất cho nhiều mục đích.

Giao diện đơn giản và dễ sử dụng: Giao diện của ứng dụng được thiết kế đơn giản, dễ dàng tiếp cận và sử dụng cho người dùng ở mọi độ tuổi.

➤ Tính mở rộng:

Khả năng mở rộng: Dễ dàng mở rộng thêm các tính năng mới trong tương lai, chẳng hạn như thêm các trò chơi mới hoặc các phép toán phức tạp hơn cho máy tính.

Khả năng điều hướng: Ứng dụng sử dụng NavigationView để điều hướng giữa các trang, giúp người dùng dễ dàng chuyển đổi giữa các chức năng.

➤ Cải thiện hiệu suất:

Hiệu năng: Ứng dụng hoạt động mượt mà trên hầu hết các thiết bị UWP, đảm bảo trải nghiệm người dùng không bị gián đoạn.

Hạn chế

➤ Giới hạn chức năng:

Chức năng lịch hẹn: Hiện tại, chức năng lịch hẹn còn hạn chế trong việc đồng bộ hóa với các ứng dụng lịch khác và chưa có nhắc nhở linh hoạt.

Chức năng trò chơi: Chỉ có một trò chơi Match Game, chưa có nhiều lựa chọn giải trí khác cho người dùng.

Chức năng máy tính: Chỉ hỗ trợ các phép toán cơ bản, thiếu các phép toán phức tạp như lũy thừa, căn bậc hai.

➤ Thiếu tính năng bảo mật:

Bảo mật dữ liệu: Chưa có các biện pháp bảo mật mạnh mẽ cho dữ liệu người dùng, như mã hóa thông tin liên lạc hoặc bảo vệ dữ liệu cá nhân.

Quyền riêng tư: Chưa có chính sách rõ ràng về quyền riêng tư và cách thức ứng dụng xử lý dữ liệu cá nhân của người dùng.

➤ Thiếu tính năng cá nhân hóa:

Tùy chỉnh giao diện: Người dùng không thể tùy chỉnh giao diện theo sở thích cá nhân, làm giảm tính cá nhân hóa.

Thiết kế giao diện đơn giản: Giao diện hiện tại khá đơn giản, chưa thực sự hấp dẫn và cuốn hút người dùng.

➤ So sánh với các ứng dụng khác

So với các ứng dụng chuyên dụng trong từng lĩnh vực, Convenience App có những ưu và nhược điểm nhất định:

➤ So với các ứng dụng lịch chuyên dụng (như Google Calendar):

Ưu điểm: Tích hợp nhiều chức năng khác nhau trong một ứng dụng duy nhất.

Nhược điểm: Thiếu tính năng đồng bộ và nhắc nhở linh hoạt, không mạnh mẽ bằng các ứng dụng lịch chuyên dụng.

➤ So với các ứng dụng trò chơi chuyên dụng:

Ưu điểm: Tích hợp trong một ứng dụng đa năng, giúp người dùng không cần tải nhiều ứng dụng khác nhau.

Nhược điểm: Thiếu đa dạng về trò chơi, không có các trò chơi phức tạp và hấp dẫn như các ứng dụng trò chơi chuyên dụng.

➤ So với các ứng dụng máy tính chuyên dụng:

Ưu điểm: Tiện lợi cho các phép toán cơ bản hàng ngày.

Nhược điểm: Không hỗ trợ các phép toán phức tạp, thiếu các chức năng nâng cao.

⊕ **Đề xuất hướng phát triển**

➤ **Tăng cường tính năng:**

Lịch hẹn: Đồng bộ hóa với các ứng dụng lịch khác, thêm nhắc nhở linh hoạt.

Trò chơi: Phát triển thêm nhiều trò chơi khác, hỗ trợ chế độ đa người chơi.

Máy tính: Nâng cấp để hỗ trợ các phép toán phức tạp hơn.

➤ **Cải thiện giao diện người dùng:**

Tùy chỉnh giao diện: Cho phép người dùng tùy chỉnh giao diện theo sở thích cá nhân.

Nâng cấp thiết kế giao diện: Làm mới giao diện với thiết kế hiện đại, hấp dẫn hơn.

➤ **Tối ưu hiệu năng và trải nghiệm người dùng:**

Tối ưu hiệu năng: Cải thiện hiệu năng trên các thiết bị cấu hình thấp.

Đa dạng hóa thiết bị hỗ trợ: Tối ưu hóa ứng dụng để hoạt động tốt trên nhiều loại thiết bị khác nhau.

➤ **Khả năng mở rộng và bảo trì:**

Cải thiện cấu trúc mã nguồn: Tổ chức và chú thích mã nguồn rõ ràng hơn để dễ dàng mở rộng và bảo trì.

Sử dụng các mẫu thiết kế: Áp dụng các mẫu thiết kế phần mềm để cải thiện khả năng mở rộng và bảo trì.

➤ **Bảo mật và quyền riêng tư:**

Cơ chế bảo mật: Phát triển các cơ chế bảo mật mạnh mẽ như mã hóa dữ liệu và bảo vệ thông tin cá nhân của người dùng.

Chính sách quyền riêng tư: Xây dựng và công khai chính sách quyền riêng tư rõ ràng.

✚ Kết luận

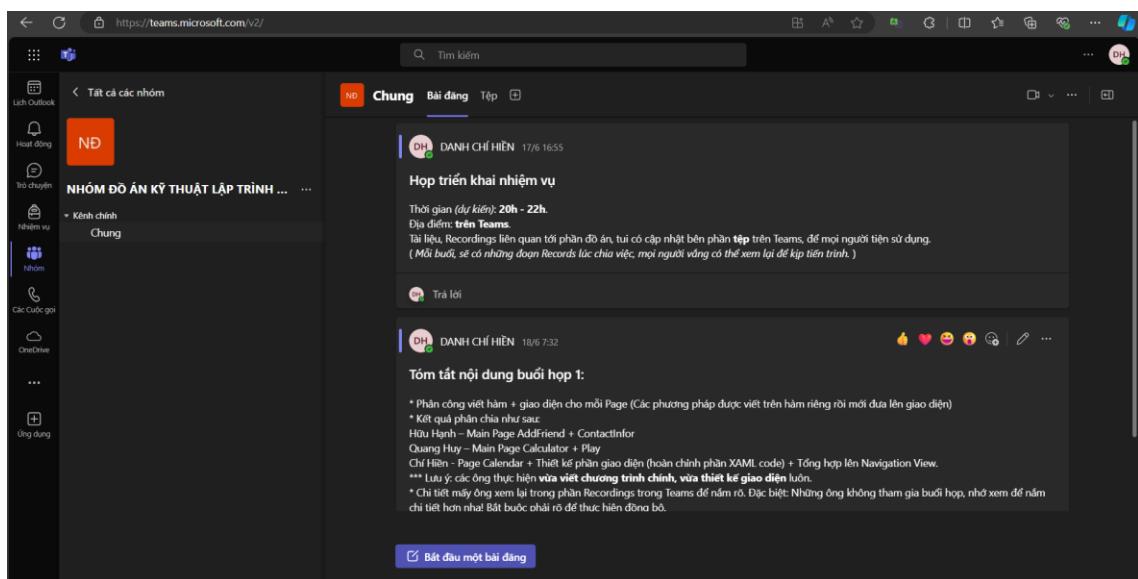
Convenience App là một ứng dụng đa năng và tiện ích, mang lại nhiều lợi ích cho người dùng trong việc quản lý các hoạt động hàng ngày. Tuy nhiên, ứng dụng vẫn còn một số hạn chế cần khắc phục và cải thiện để đáp ứng tốt hơn nhu cầu của người dùng. Bằng cách phát triển thêm các tính năng mới, cải thiện giao diện và tối ưu hóa hiệu năng, ứng dụng sẽ trở nên toàn diện và hấp dẫn hơn.

3.2. HOẠT ĐỘNG CỦA NHÓM VÀ CÁC THÀNH VIÊN

✚ Hoạt động nhóm:

Nhóm phân công hoạt động giữa các thành viên, triển khai, phân chia nhiệm vụ trên Microsoft Teams:

NHÓM ĐO ÁN KỸ THUẬT LẬP TRÌNH NÂNG CAO (thầy Thinh) | Chung | Microsoft Teams



Hình 38. Hoạt động nhóm trên MS Teams

Nhóm cũng triển khai những buổi họp offline trên trường (tại phòng tự học) để hoạt động triển khai viết code và thiết kế giao diện với các thành viên.

Trong các buổi họp online đều có Recording để những thành viên có thể xem lại lúc cần thiết.

- ✚ Nhóm có sử dụng phần mềm quản lý code để cộng tác (Github):

[danhchihien/UWP_ConvenienceApp: This is the final project for the Advanced Programming Techniques course, programming Universal Windows Platform \(UWP\) on Visual Studio 2022. \(github.com\)](https://github.com/danhchihien/UWP_ConvenienceApp)

3.3. PHÂN CÔNG, ĐÁNH GIÁ CÁC THÀNH VIÊN

Bảng phân công - đánh giá thành viên nhóm 3KG

TT	Tên thành viên	Nhiệm vụ	Dánh giá nhóm (tổng các thành viên = 100%)	Ghi chú
1	Danh Chí Hiền 21200287	Nhóm trưởng, quản lý nhóm, phân công nhiệm vụ các thành viên Khởi gợi yêu cầu (User-story) Đặc tả yêu cầu chức năng, Quản lý mã nguồn Viết các hàm, nêu được thuật toán của Page Calendar + Page Calculator + Thiết kế phần giao diện (hoàn chỉnh phần XAML code) + Tổng hợp lên Navigation View Cho được ví dụ test đúng chương trình. Hoàn thiện báo cáo	Hoàn thành tốt (50%)	

2	Trần Hữu Hạnh 21200286	Thành viên Viết các hàm, nêu được thuật toán của Main Page AddFriend + ContactInfor Cho được ví dụ test đúng chương trình.	Hoàn thành tốt (25%)	
3	Lê Quang Huy 21200293	Thành viên Viết các hàm, nêu được thuật toán của Play Page Cho được ví dụ test đúng chương trình.	Hoàn thành tốt (25%)	

Rubric đánh giá:

TT	Nội dung	Điểm tối ta	Nhóm đánh giá	GV đánh giá	Ghi chú
1	Dự án sử dụng C#/C++ để thiết kế ứng dụng hoặc game	0.5	0.5		
2	Dự án sử dụng một trong các nền tảng: WPF, Win Form, UWP, Unity 3D,... để thiết kế giao diện và chức năng	0.5	0.5		
3	Sử dụng các kỹ thuật lập trình: lớp, phương thức, field, properties	1	1		
4	Có sử dụng các kỹ thuật kế thừa và đa hình	1	1		
5	Có sử dụng interface hoặc abstract class	0.5	0.5		
6	Có xử lý ngoại lệ (Exception)	0.5	0.5		
7	Trình bày báo cáo trên file pdf (tối thiểu 12 trang A4)				
7.1	Có trình bày mục tiêu, chức năng của sản phẩm	0.5	0.5		
7.2	Có trình bày thuật toán rõ ràng	1	1		
7.3	Thiết kế hoàn chỉnh giao diện	0.5	1		
7.4	Có giải thích code	0.5	1		
7.5	Minh họa hoạt động của ứng dụng (game) bằng hình ảnh	1	1		

7.6	Có bảng phân công nhiệm vụ và đánh giá nhóm	0.5	0.5		
8	Có sử dụng nền tảng cộng tác github hoặc tương đương hoặc sử dụng nền tảng đám mây để xây dựng web app hoặc app liên quan đến giao tiếp internet, IoT	0.5	0.5		
9	Thể hiện sự hợp tác hiệu quả giữa các thành viên	0.5	0.5		
10	Code tự viết: trên 90% (1 điểm) trên 70% (0.75 điểm) trên 50% (0.5 điểm) trên 25% (0.25 điểm)	1	1		

TÀI LIỆU THAM KHẢO

- [1] Huỳnh Quốc Thịnh, "Slide Bài giảng Kỹ thuật lập trình nâng cao", Khoa Điện Tử - Viễn Thông, Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM.
- [2] Huỳnh Quốc Thịnh, "Tài liệu Thực hành Kỹ thuật lập trình nâng cao", Khoa Điện Tử - Viễn Thông, Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM.