TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ MÔN

PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

TÌM HIỂU JETPACK COMPOSE VÀ XÂY DỰNG ỨNG DỤNG MINH HOẠ

Người hướng dẫn: **GV. LÊ VĂN VANG**

Người thực hiện: **LÝ TUẤN AN – 52000620**

Lớp : **20050201**

Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN GIỮA KÌ MÔN

PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

TÌM HIỂU JETPACK COMPOSE VÀ XÂY DỰNG ỨNG DỤNG MINH HOẠ

Người hướng dẫn: **GV. LÊ VĂN VANG**
Người thực hiện: **LÝ TUẤN AN**
Lớp **: 20050201**
Khoá **: 24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những "lời cảm ơn" đã có.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của GV. Lê Văn Vang. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Lý Tuấn An

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm
(kí và ghi họ tên)

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

MỤC LỤC

LỜI CẢM ƠN i
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii
TÓM TẮT iv
MỤC LỤC 1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4
CHƯƠNG 1 – GIỚI THIỆU 5
1.1 Mô tả vấn đề cần giải quyết 5
1.2 Mục tiêu và phạm vi đề tài 5
1.3 Định nghĩa cơ bản về Jetpack Compose 5
CHƯƠNG 2 – CỞ SỞ LÝ THUYẾT 6
2.1 Giới thiệu về Jetpack Compose 6
2.2 Kiến trúc của Jetpack Compose 6
2.3 Cấu trúc của Jetpack Compose 7
2.4 Các thành phần chính của Jetpack Compose 7
2.4.1 Composable Functions 7
2.4.2 Layout 7
2.4.3 Material Design 8
2.4.4 Animations 8
2.4.5 State Management 8
2.4.6 Navigation 9
CHƯƠNG 3 – XÂY DỰNG VÀ PHÂN TÍCH CÁCH SỬ DỤNG CỦA CÁC THÀNH PHẦN GIAO DIỆN JETPACK COMPOSE 10
3.1 Composable functions 10
3.2 Layout (Box, Row, Column) 10
3.3 Text 11
3.4 Image 11

3.5 Button 11
3.6 TextField 12
3.7 Radiobutton & Checkbox 12
3.8 LazyColumn & RowColmn 12
3.9 Navigation 12
CHƯƠNG 4 – ĐÁNH GIÁ VÀ SO SÁNH 14
4.1 Composable functions 14
4.2 Layout (Box, Row, Column) 14
CHƯƠNG 5 – KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 15
5.1 Tổng kết kết quả đạt được trong đề tài 15
5.2 Những hạn chế và thách thức trong quá trình nghiên cứu và xây dựng đề tài 15
5.3 Đề xuất các hướng phát triển tiếp theo ..... **Error! Bookmark not defined.**

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU

f *Tần số của dòng điện và điện áp (Hz)*

p *Mật độ điện tích khối (C/m3)*

CÁC CHỮ VIẾT TẮT

CSTD	Công suất tác dụng
MF	Máy phát điện
BER	Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 3.1: Chu trình của Composable (LifeCirc.....**Error! Bookmark not defined.**

DANH MỤC BẢNG

Bảng 3.1 Ví dụ cho chèn bảng .....**Error! Bookmark not defined.**

CHƯƠNG 1 – GIỚI THIỆU

1.1 Mô tả vấn đề cần giải quyết

Hiện nay, việc phát triển giao diện người dùng trên Android đòi hỏi phải sử dụng nhiều file XML và code để tạo ra giao diện phức tạp, dẫn đến việc làm việc tốn thời gian và khó bảo trì. Vấn đề này có thể được giải quyết bằng cách sử dụng Jetpack Compose để xây dựng giao diện người dùng bằng cách viết code theo cách khác, giúp đơn giản hóa quá trình phát triển và bảo trì ứng dụng.

Việc tạo ra các màn hình phức tạp và tùy chỉnh trong các ứng dụng Android truyền thống đòi hỏi phải sử dụng nhiều file XML và có khả năng gây ra sự cố khi tích hợp giữa các thành phần với nhau. Jetpack Compose cung cấp các thành phần tái sử dụng, giúp cho việc tạo ra các giao diện phức tạp trở nên dễ dàng hơn.

1.2 Mục tiêu và phạm vi đề tài

Tìm hiểu về Jetpack Compose: Nghiên cứu cách sử dụng Jetpack Compose để xây dựng giao diện người dùng trong ứng dụng Android. Bao gồm các thành phần cơ bản như Layout, Text, Image, Button, TextField, LazyColumn, RowColumn, … cách sử dụng, cú pháp và các tính năng cơ bản của Jetpack Compose.

Xây dựng các ứng dụng minh hoạ đơn giản sử dụng Jetpack Compose để hiển thị các giao diện người dùng và tính năng của nó.

Nhận xét, đánh giá và so sánh giữa xây dựng giao diện bằng Jetpack Compose và cách xây dựng giao diện truyền thống (sử dụng XML và code).

1.3 Định nghĩa cơ bản về Jetpack Compose

Jetpack Compose là một framework để xây dựng giao diện người dùng trong ứng dụng Android. Nó cho phép các nhà phát triển sử dụng một ngôn ngữ lập trình khác với XML và có thể tạo ra các giao diện người dùng động và phức tạp dễ dàng hơn.

CHƯƠNG 2 – CỞ SỞ LÝ THUYẾT

2.1 Giới thiệu về Jetpack Compose

Jetpack Compose là một framework mới của Google giúp các nhà phát triển Android xây dựng giao diện người dùng (UI) trong ứng dụng của mình. Nó là một công nghệ mới và thay thế cho việc sử dụng XML để thiết kế giao diện trong Android. Jetpack Compose cho phép các nhà phát triển tạo giao diện người dùng bằng cách sử dụng mã Kotlin thay vì XML, giúp tăng khả năng tái sử dụng mã và giảm thiểu số lượng mã phức tạp.

Bên cạnh đó, JetBrains là công ty phát triển Kotlin - ngôn ngữ lập trình được sử dụng để phát triển Jetpack Compose. Cả hai đều đóng góp quan trọng cho việc phát triển Jetpack Compose.

Jetpack Compose cũng cung cấp các thành phần UI động và phức tạp, cho phép các nhà phát triển tạo ra các hiệu ứng động, hoạt hình và tương tác phức tạp một cách dễ dàng. Nó cũng tích hợp tốt với các thành phần khác của Jetpack và cung cấp tính năng kiểm tra tự động và hỗ trợ bằng cách sử dụng Android Studio, giúp tăng năng suất và tiết kiệm thời gian trong quá trình phát triển ứng dụng.

Jetpack Compose được giới thiệu vào năm 2019 và hiện đang được phát triển và cải tiến liên tục bởi Google. Nó là một công nghệ đầy tiềm năng và được kỳ vọng sẽ là một trong những công nghệ quan trọng trong việc phát triển ứng dụng Android trong tương lai.

2.2 Kiến trúc của Jetpack Compose

Kiến trúc của Jetpack Compose được thiết kế để đơn giản hóa quá trình xây dựng và bảo trì giao diện người dùng trong ứng dụng Android. Nó gồm hai phần chính là:

+ Composable functions: Là các hàm có thể tạo ra các thành phần giao diện người dùng, chẳng hạn như các nút, trường văn bản và hình ảnh. Mỗi hàm Composable là một

phần của một giao diện người dùng lớn hơn, được gọi là Composable tree. Các hàm Composable được thiết kế để có thể tái sử dụng và kết hợp với nhau để tạo ra các giao diện người dùng phức tạp hơn.

+ Runtime: Là một thư viện động được tích hợp sẵn vào ứng dụng Android. Nó giúp tạo và quản lý các thành phần giao diện người dùng trong ứng dụng, đồng thời cũng đảm bảo rằng các thành phần này được hiển thị chính xác trên các thiết bị Android khác nhau.

2.3 Cấu trúc của Jetpack Compose

Cấu trúc của Jetpack Compose được xây dựng dựa trên cách tiếp cận "Declarative Programming", tức là các thành phần giao diện người dùng được mô tả bằng các khai báo, thay vì được xác định thông qua các thao tác trực tiếp. Các khai báo này được sử dụng để mô tả các giao diện người dùng trong Composable functions. Khi một Composable function được gọi, nó sẽ trả về một khai báo cho các thành phần giao diện người dùng của nó. Sau đó, Runtime sẽ sắp xếp các khai báo này để hiển thị các thành phần tương ứng trên giao diện người dùng.

Việc sử dụng Declarative Programming và Composable functions giúp Jetpack Compose dễ dàng hơn trong việc tạo, kiểm tra và bảo trì các giao diện người dùng phức tạp.

2.4 Các thành phần chính của Jetpack Compose

2.4.1 Composable Functions

Đây là thành phần cơ bản nhất của Jetpack Compose. Composable function là một hàm bình thường được đánh dấu bằng annotation @Composable. Nó cho phép xây dựng các thành phần giao diện người dùng, như các nút, trường văn bản, hình ảnh, …

2.4.2 Layout

Jetpack Compose cung cấp các thành phần lớp layout, ví dụ như Column, Row, Box,... để giúp cho việc bố trí các Composable Functions trên màn hình trở nên linh hoạt hơn.

2.4.3 Material Design

Được triển khai dưới dạng một bộ thư viện các Composable function có sẵn, được gọi là "Material Components for Android". Nó giúp tạo ra các giao diện đẹp mắt và đồng nhất với các ứng dụng Android khác. Ví dụ như Button, Text, TextField và nhiều hơn nữa.

2.4.4 Animations

Cung cấp một số tính năng tạo hiệu ứng chuyển động trơn tru trên giao diện người dùng, chẳng hạn như tạo fade-in hoặc scale-in effect cho các thành phần giao diện người dùng.

2.4.5 State Management

Cung cấp các cơ chế quản lý trạng thái ứng dụng, giúp tăng tính tương tác của ứng dụng. Trong Jetpack Compose, State Management được thực hiện thông qua việc sử dụng các Composable Function kết hợp với các loại state khác nhau

Các loại state trong Jetpack Compose bao gồm:

+ MutableState:

State có thể thay đổi được trong một Composable Function và sẽ gây ra việc vẽ lại UI khi giá trị của nó thay đổi.

+ ViewModel:

State được giữ bởi một ViewModel, giúp cho các thành phần của ứng dụng có thể truy cập và sử dụng trạng thái của ứng dụng một cách chung và đồng bộ.

ViewModel là một lớp được thiết kế để lưu trữ và quản lý dữ liệu của ứng dụng, giúp cho dữ liệu có thể tồn tại trong suốt vòng đời của các thành phần của ứng dụng. ViewModel có thể cung cấp các phương thức để tương tác với dữ liệu và đưa ra các sự kiện cập nhật khi dữ liệu thay đổi.

+ LiveData:

State được quản lý bởi LiveData, giúp cho các thành phần của ứng dụng có thể theo dõi và cập nhật trạng thái một cách tự động.

LiveData là một lớp được thiết kế để quản lý dữ liệu, nó cung cấp một cơ chế để gửi thông báo về sự thay đổi dữ liệu tới các thành phần khác của ứng dụng. Các thành phần khác có thể đăng ký theo dõi dữ liệu từ LiveData, và sẽ nhận được các thông báo khi dữ liệu thay đổi.

ViewModel và LiveData thường được sử dụng để giải quyết các vấn đề liên quan đến quản lý dữ liệu và cập nhật giao diện. Khi một thành phần của ứng dụng (ví dụ như Activity) cần truy cập hoặc sửa đổi dữ liệu, nó sẽ yêu cầu ViewModel cung cấp dữ liệu đó. ViewModel sẽ sử dụng các tài nguyên như cơ sở dữ liệu hoặc các API mạng để lấy dữ liệu mới nhất, sau đó cập nhật LiveData với dữ liệu này. Các thành phần khác của ứng dụng (ví dụ như giao diện) có thể đăng ký theo dõi LiveData để nhận thông báo về các thay đổi của dữ liệu và cập nhật giao diện tương ứng.

Tóm lại, sử dụng ViewModel và LiveData phù hợp trong các trường hợp cần lưu trữ và quản lý dữ liệu trong suốt vòng đời của ứng dụng, và đảm bảo rằng dữ liệu được cập nhật đồng bộ với giao diện người dùng.

2.4.6 Navigation

Là một cách để quản lý việc điều hướng giữa các màn hình trong ứng dụng. Navigation giúp cho việc điều hướng trở nên dễ dàng hơn, cho phép người phát triển quản lý và xác định các màn hình, tương tác giữa các màn hình, cũng như cách thức chuyển đổi giữa chúng.

CHƯƠNG 3 – XÂY DỰNG VÀ PHÂN TÍCH CÁCH SỬ DỤNG CỦA CÁC THÀNH PHẦN GIAO DIỆN JETPACK COMPOSE

3.1 Composable functions

Composable function trong Jetpack Compose là một loại hàm đặc biệt được sử dụng để xây dựng các thành phần giao diện người dùng. Composable function có thể nhận đầu vào và trả về một hoặc nhiều thành phần composable khác, và được sử dụng để định nghĩa các thành phần giao diện như buttons, text fields, images, layouts,…

Việc sử dụng Composable function trong Jetpack Compose cũng giúp giảm thiểu sự phức tạp của việc xử lý và quản lý các thành phần giao diện trong các ứng dụng Android truyền thống, đồng thời giúp tăng tính tương tác và hiệu suất của ứng dụng.

Demo: https://youtu.be/q7TbtXdAy7I

Tính tái sử dụng: https://youtu.be/yCZqlyrL8ec

3.2 Layout (Box, Row, Column)

Column và Row có thể được lồng vào nhau, tạo thành các thành phần UI phức tạp hơn.

Column và Row có nhiều thuộc tính hơn so với LinearLayout, cho phép chúng ta kiểm soát hình dáng, vị trí và kích thước của các thành phần UI một cách chính xác hơn

Demo:

Box Layout: https://youtu.be/iGwWbsKX05k

Column Layout: https://youtu.be/D-0upVwnXnQ

Row Layout: https://youtu.be/Fs9pMIajo0g

Weight: https://youtu.be/LigM4nKURAg

Liên quan: Modifier là một lớp dùng để cấu hình cho một Composable trong Jetpack Compose. Nó cho phép bạn thực hiện các thay đổi về kích thước, vị trí, màu sắc, viền, độ mờ và nhiều thuộc tính khác cho các Composable. Các Modifier có thể được kết hợp để thực hiện các thay đổi phức tạp cho Composable.

3.3 Text

Demo: https://youtu.be/XGFA2JsFHiQ

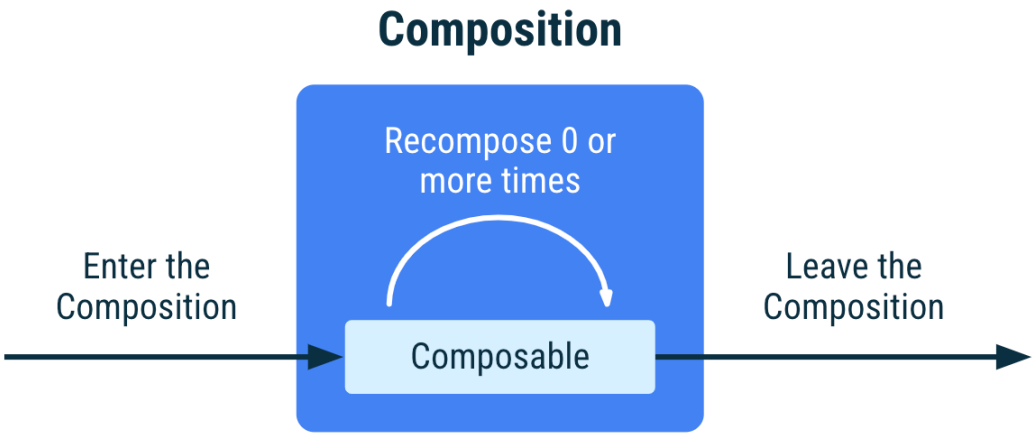
3.4 Image

Demo: https://youtu.be/nnidlwSbBo8

3.5 TextField & State

Hàm remember sẽ tạo ra một object để lưu trữ lại dữ liệu. Trong đó, mutableStateOf() là một hàm trong Compose cung cấp một State object có thể thay đổi được. State object chứa một giá trị và có thể được cập nhật bởi Composable function.

Vì vậy, để lưu trữ các giá trị dùng để render các Composable, chúng ta cần sử dụng State object. State object cung cấp một phương thức để Compose biết rằng Composable cần được render lại khi giá trị của State object thay đổi. State object cũng cho phép chúng ta cập nhật giá trị của nó bên trong Composable. Khi giá trị của State object thay đổi, Compose sẽ tự động cập nhật lại Composable tương ứng để hiển thị giá trị mới.



Hình 3.1 Chu trình của composition (LifeCircle)

Demo:

TextField: https://youtu.be/1CUMtzUKkBo

State: https://youtu.be/5ZTa_gX21hg

Keyboard Action: https://youtu.be/4ReuZ4n_QxM

3.6 Button

Demo: https://youtu.be/J5fXp-WI1D8

3.7 Radiobutton & Checkbox

Demo: https://youtu.be/oLv3MEWkNvA

3.8 LazyColumn & RowColmn

Trong lập trình truyền thống, RecyclerView sử dụng một Adapter để cung cấp dữ liệu và tạo View cho từng item trong danh sách. Mỗi khi danh sách thay đổi, Adapter phải cập nhật lại giao diện hiển thị của RecyclerView.

Trong Compose, LazyColumn (tương đương với RecyclerView) có thể sử dụng một danh sách đơn giản như một nguồn dữ liệu và tự động tạo ra các thành phần giao diện của từng phần tử trong danh sách. Khi danh sách thay đổi, Compose sẽ tự động cập nhật lại giao diện mà không cần sự can thiệp của người lập trình.3.9 Navigation

Demo: https://youtu.be/UjEEAEP_hjg

3.9 Navigation

NavHost là một Composable, là nơi chứa các màn hình con (composable screens) và quản lý các hoạt động điều hướng giữa chúng. NavHost chính là vùng hiển thị các màn hình của ứng dụng.

navController là đối tượng điều khiển hoạt động điều hướng trong NavHost. Nó giúp quản lý các màn hình, hoạt động điều hướng, truyền dữ liệu giữa các màn hình, v.v.

rememberNavController là một hàm Composable, Nó trả về một đối tượng NavHostController, được sử dụng để điều khiển việc điều hướng giữa các màn hình trong ứng dụng. rememberNavController() sử dụng một cơ chế gọi là "remember" để giữ giá trị của NavHostController trong quá trình render lại giao diện Composable. Điều này đảm bảo rằng khi màn hình được render lại, NavHostController sẽ không bị khởi tạo lại từ đầu, giúp tránh các vấn đề liên quan đến việc mất dữ liệu hoặc trạng thái của ứng dụng.

startDestination là tên của màn hình đầu tiên sẽ được hiển thị.

Để chuyển đổi giữa các màn hình, chúng ta sử dụng navController.navigate(route) với route là chuỗi định danh cho màn hình mà chúng ta muốn chuyển đến. Chúng ta cũng có thể sử dụng các tham số để truyền dữ liệu giữa các màn hình.

Demo: https://youtu.be/UjEEAEP_hjg

Trong demo, chúng ta sử dụng state để truyền dữ liệu giữa các màn hình. Tuy nhiên, nếu có nhiều dữ liệu và phức tạp hơn, việc sử dụng state để truyền dữ liệu có thể trở nên khó khăn và dễ gây lỗi. Trong trường hợp đó, sử dụng ViewModel và LiveData có thể là một giải pháp tốt hơn để quản lý dữ liệu.

CHƯƠNG 4 – ĐÁNH GIÁ VÀ SO SÁNH

4.1 Đánh giá

Jetpack Compose là một cách tiếp cận mới để phát triển giao diện người dùng trong Android, thay thế cho phương pháp truyền thống sử dụng XML để thiết kế giao diện. So sánh giữa Jetpack Compose và phương pháp truyền thống.

Đơn giản hóa cú pháp và tăng hiệu suất: Trong Jetpack Compose, UI được tạo ra bằng cách sử dụng một ngôn ngữ đơn giản và bị giới hạn, cho phép tạo ra các thành phần UI động và linh hoạt hơn. So với phương pháp truyền thống sử dụng XML, Jetpack Compose có cú pháp đơn giản hơn và cho phép định nghĩa UI trong cùng một tệp mã nguồn, giúp giảm bớt thời gian phát triển và tăng hiệu suất ứng dụng.

Cấu trúc UI động: Jetpack Compose cho phép tạo ra các thành phần UI động một cách dễ dàng và linh hoạt hơn. Điều này cho phép tạo ra các thành phần UI phức tạp hơn, cung cấp trải nghiệm người dùng tốt hơn.

Khả năng tái sử dụng: Jetpack Compose cho phép tạo ra các thành phần UI có thể tái sử dụng, giúp giảm thiểu việc lặp lại mã và tăng tính mở rộng của ứng dụng.

4.2 So sánh với lập trình truyền thống

Điểm mạnh của Jetpack Compose so với lập trình truyền thống:

+ Code ngắn gọn hơn

+ Dễ dàng tái sử dụng code

Một số điểm yếu của Jetpack Compose so với lập trình truyền thống:

+ Đang trong quá trình phát triển: Jetpack Compose vẫn đang trong giai đoạn phát triển và còn nhiều tính năng chưa được hoàn thiện hoặc được thêm vào.

+ Khó khăn trong việc tìm tài liệu và hỗ trợ: Vì Jetpack Compose vẫn còn mới, nên khó khăn trong việc tìm tài liệu và hỗ trợ.

CHƯƠNG 5 – KẾT LUẬN

5.1 Tổng kết kết quả đạt được trong đề tài

Hiểu đươc cơ bản các thành phần UI trong Jetpack Compose, cách các thành phần tương tác với nhau

Nắm được cách thức phát triển ứng dụng bằng Jetpack Compose, một công nghệ mới trong lĩnh vực phát triển ứng dụng Android, giúp chúng ta có thể tạo ra các giao diện người dùng đẹp mắt và linh hoạt hơn.

Đánh giá được ưu điểm và nhược điểm của Jetpack Compose so với lập trình truyền thống trong việc phát triển giao diện người dùng.

5.2 Những hạn chế và thách thức trong quá trình nghiên cứu và xây dựng đề tài

Có quá nhiều deadline trong cùng một thời điểm khiến cho việc tìm hiểu nghiên cứu bị chậm trễ.

Tài liệu tiếng anh nhiều hơn nên gây khó khăn trong việc tiếp cận ban đầu

TÀI LIỆU THAM KHẢO

https://developer.android.com/jetpack/compose/tutorial?hl=vi

https://developer.android.com/docs?hl=vi

https://viblo.asia/p/jetpack-compose-cho-nguoi-moi-bat-dau-phan-1-RnB5ppp75PG

https://www.jetpackcompose.net/

PHỤ LỤC

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh. . . . nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn.