**CÔNG TY CÔNG NGHỆ & KỸ THUẬT STECH**

**VIỆN NGHIÊN CỨU & SÁNG TẠO KHOA HỌC MÁY TÍNH – IRICS**

**A logo with orange letters

Description automatically generated**

**BÁO CÁO NGHIÊN CỨU FLUTTER**

**TASK 5 – NGHIÊN CỨU FLUTTER**

Người hướng dẫn**: MR. NGUYỄN TẤN HÙNG ANH**

**MR. VÕ THÀNH LUÂN**

Người thực hiện**: ĐINH NGỌC AN THƠ**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

**TÓM TẮT**

Ứng dụng Flutter đã xây dựng là một ví dụ tiêu biểu cho sức mạnh và linh hoạt của nền tảng phát triển ứng dụng di động Flutter. Mục tiêu của ứng dụng là tạo ra một giao diện người dùng đơn giản nhưng hữu ích, cho phép người dùng chọn và xem hình ảnh của các loài động vật.

Quy trình phát triển ứng dụng bao gồm các bước cơ bản từ việc tạo dự án mới đến thiết kế giao diện người dùng và triển khai logic ứng dụng. Bằng cách sử dụng các công cụ và tính năng của Flutter như widget, state management và event handling, chúng tôi đã xây dựng một ứng dụng hoàn chỉnh và chức năng.

Ứng dụng không chỉ cung cấp một trải nghiệm người dùng tốt mà còn mang lại lợi ích thực tế bằng cách giúp người dùng dễ dàng tương tác và tìm hiểu về thế giới tự nhiên xung quanh chúng. Đồng thời, việc xây dựng ứng dụng này đã mở ra một cánh cửa mới cho việc thử nghiệm và phát triển ứng dụng Flutter trong tương lai.

**MỤC LỤC**

[CHƯƠNG I: CODE FILE QUIZZLZER 6](#_Toc168406498)

[I. MAIN.DART 6](#_Toc168406499)

[II. QUESTION.DART 12](#_Toc168406500)

[III. QUIZ\_BRAIN.DART 12](#_Toc168406501)

[CHƯƠNG II: CODE FILE PROJECT\_TASK\_5 14](#_Toc168406502)

[I. MAIN.DART 14](#_Toc168406503)

[II. ON\_HOVER\_TEST.DART 18](#_Toc168406504)

[III. QUESTION.DART 19](#_Toc168406505)

[IV. QUIZ\_BRAIN.DART 19](#_Toc168406506)

[CHƯƠNG III: TỔNG HỢP 20](#_Toc168406507)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

[Hình 1: Package and Main trong file Quizzlzer 6](#_Toc168407011)

[Hình 2: class Quizzler của file Quizzler 7](#_Toc168407012)

[Hình 3: class QuizPage & class \_QuizPageState 8](#_Toc168407013)

[Hình 4: class QuizPage & class \_QuizPageState 9](#_Toc168407014)

[Hình 5: widget builld 10](#_Toc168407015)

[Hình 6: expanded widget build 11](#_Toc168407016)

[Hình 7: class Question in Question.dart 12](#_Toc168407017)

[Hình 8: QuizBrain 14](#_Toc168407018)

[Hình 9: Main in Main.dart file project task 5 15](#_Toc168407019)

[Hình 10 : class QuizPage & class \_QuizPageState 17](#_Toc168407020)

[Hình 11: widget build in project task 5 18](#_Toc168407021)

[Hình 12 : align 18](#_Toc168407022)

[Hình 13 : On\_Hover\_Test.dart 19](#_Toc168407023)

CHƯƠNG I: CODE FILE QUIZZLZER

1. MAIN.DART
2. **Package & Main**

**A computer screen shot of text

Description automatically generated**

Hình 1: Package and Main trong file Quizzlzer

* Import

/*material :* Thư viện Material cung cấp các Widget thiết kế để xây dựng giao diện người dùng của Flutter ( *Scaffold, AppBar, Text, Buttons* )

/*rflutter\_alrert:* Thư viện hiển thị cảnh báo có thể tùy chỉnh trong Flutter cho phép tạo các hộp thoại cảnh báo với nhiều kiểu và tùy chọn khác nhau để nâng cao sự tương tác với người dùng bằng các hiển thị các cảnh cáo, xác nhận hoặc thông tin bật lên

*/quiz\_brain.dart* : Nhập tập Dart được tùy chỉnh là ‘*quiz\_brain.dart’*  chứa class *QuizBrain*  quản lý logic của phần câu hỏi, theo dõi câu trả lời, cung cấp thông tin cho phần Quiz

* *QuizBrain quizBrain =QuizBrain() :*

ở đối tượng của class ‘QuizBrain’ được tạo và gán cho biến ‘*quizBrain’.* Đối tượng được sử dụng trong suốt ứng dụng để quản lý các hoạt động liên quan đên Quiz

* *Void main() => runApp(Quizzler());*

Void main () : Main được gọi khi ứng dụng khởi chạy. Chạy widget ‘*Quizzler’*  như gốc của ứng dụng

1. **Class Quizzler**

**A screen shot of a computer program

Description automatically generated**

Hình 2: class Quizzler của file Quizzler

* *‘Quizzer extends Statelesswidget’ : Quizzer* kế thừ từ *StatelessWidget ,*  là một widget tĩnh, không có trạng thái thay đổi sau khi build. Không thể thay đổi nội dung sau khi được hiển thị ( ngược với StatefullWidget)
* *‘Widget build (BuildContext context)* : Hàm *build*  là phương thức bắt buộc phải ghi đè khi tạo một widget. Xây dựng và trả về cấu trúc của widget dựa trên ‘BuildContext’
* *‘MaterialApp’ :* widget gốc của một app Flutter, cung cấp tính năng cơ bản như điều hướng, chủ đề, ngôn ngữ
* *‘home: Scaffold’ :* widget khung của Flutter, cung cấp cấu trúc cơ bản cho một trang như AppBar, Drawer, BottomNavigationBar
* *‘backgroundColor: Colors.grey.shade900’ :* thiết lập máu nền của Scaffold là màu xám
* *‘body: SafeArea’ :*  widget đảm bảo nội dung app không bị chèn
* *;child: Padding’ :* widget thêm các khoảng đệm xung quanh các widget con, thêm 10ps khoảng cách vào hai bên phải và trái
* *‘child : QuizPage() ‘ :* class khác trong app, hiển thị nội dung của trang quiz được đặt bên trong ‘*Padding’*

1. **Class QuizPage & \_QuizPageState**

**A computer screen shot of code

Description automatically generated**

Hình 3: class QuizPage & class \_QuizPageState

* *‘QuizPage extends StatefulWidget’*  : *QuizPage* kế thừa từ *StatefullWidget* trạng trái có thể thay đổi được
* *‘createState() => \_QuizPageState()’* : phương thức tạo ra một đối tượng là ‘*\_QuizPageState’*  là một trạng thái liên kết với widget ‘QuizPage’
* Biến ‘*scoreKeeper*’ :

*List<Icon> scoreKeeper = [] :* danh sách lưu trữ các biểu tượng icon đúng (icon check) và icon sai ( icon close) tương ứng với mỗi câu trả lời

* *Void checkAnswer (bool userPickedAnswer)* : phương thức nhận đầu vào là câu trả lời của người dùng (*userPickedAnswer),* để kiểm tra câu trả lời đó, cập nhật trạng thái và hiển thị thông báo khi Quiz kết thúc
* *Bool correctAnswer = quizBrain.getCorrectAnswer() :*  chỉ lấy câu trả lời đúng
* *‘setState’ :* đảm bảo giao diện người dụng được cập nhập khi thay đổi trong trạng thái
* *If(quizBrain.isFinished() ==true ){ …*

*scoreKeeper=[];*

*}*

Nếu Quiz kết thúc, hiển thị thông báo đặt lại Quiz cùng với ‘scoreKeeper’

A computer screen shot of a program code

Description automatically generated

Hình 4: class QuizPage & class \_QuizPageState

* *Else {*

*If (userPickedAnswer == correctAnwer) {…*

*} else {*

*scoreKeeper.add(Icon(…*

*)*

*}*

*quizBrain.nextQuestion();*

*}*

Nếu trả lời đúng thêm biểu tượng ‘*Icon.check’* màu xanh lá vào ‘*scoreKeeper’*

Nếu trả lời đúng thêm biểu tượng ‘*Icon.close’* màu đỏ vào ‘*scoreKeeper’*

*‘quizBrain.nextQuestion();’ :*Chuyển sang câu hỏi tiếp theo

1. **Widget build**

**A screen shot of a computer program

Description automatically generated**

Hình 5: widget builld

* *‘@override’ :* từ khóa cho biết phương thức ‘ *build’*  đang ghi đè phương thức ‘*build’*  là State
* *‘widget build(Buildcontext context)’* : phương thức ‘*build’*  xây dựng và trả về giao diện người dùng cho widget
* *‘column’ :* widget ‘*Column’* sắp xếp các widget con theo chiều dọc
* *‘mainAxisAlignment: MainAxisAlignment.spaceBetween’* : Sắp xếp các widget con trong ‘*column’*  với khoảng cách bằng nhau từ trên xuống dưới
* *‘crossAxisAlignment: CrossAxisAlignment.stretch’ :* kéo dài các widget con theo chiều ngang, lấp đầy toàn bộ chiều rộng của ‘Column’
* *‘children :<Widget> […]’ :* danh sách các widget con của Column
* *‘expanded’ :*  widget mở rộng widget con để lấp đầy không gian có sẵn, ‘*flex’* xác định tỉ lệ không gian mà widget con chiếm trong ‘*column’*  là 5
* *‘padding’ :* widget thêm khoảng đệm xung quanh các widget con của nó. Khoảng đệm là ‘*EdgeInsets.all(10.0)’* là thêm 10 khoảng cách xung quanh tất cả các cạnh
* *‘Center’ :* widget căn giữa các widget con theo chiều ngang và chiều dọc
* *‘Text’ :* widget hiển thị văn bản. văn bản được lấy từ phương thức ‘*quizBrain.getQuestionText()’*  trả về câu hỏi hiện tại trong bài kiểm tra
* *‘textAlign : TextAlign.center’ :*  Căn giữa văn bản theo chiều ngang

**A screen shot of a computer screen

Description automatically generated**

Hình 6: expanded widget build

* *‘style: ButtonStyle()’ :* Thiết lập kiểu dáng nút bấm, màu nền là màu xanh lá cây (true) và màu đỏ ( false)
* *‘onPressed()*

*{ checkAnswer (false);} :*  hàm gọi nút bấm được nhấn, truyển giá trị false/true cho phương thức ‘checkAnswer’

1. QUESTION.DART

**A screen shot of a computer code

Description automatically generated**

Hình 7: class Question in Question.dart

* Class ‘*Questio’n* đinh nghĩa một câu hỏi với văn bản câu hỏi và câu trả lời đúng hoặc sai. Sử dụng ‘*late’*  cho phép các thuộc tính được khởi tạo sau khi đối tượng được tạo ra. Sử dụng Constructor của lớp nhận các giá trị tham số để khởi tạo các thuộc tính này
* Class giúp tạo ra các đối tượng ‘*Question’*  với các thuộc tính ‘*questionText’* và ‘*questionAnswer’* là nền tảng để quản lý các câu hỏi trong Quiz

1. QUIZ\_BRAIN.DART

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**Hình 8: QuizBrain

* Class ‘QuizzBrain’ cung cấp một cơ chế quản lý danh sách câu hỏi . bằng cách sử dụng ‘*\_questionNumber’*  có thể xác định câu hỏi hiện tại để hiển thị. Danh sách ‘*questionBank*’ chứa tất cả các câu hỏi, cho phép ứng dụng truy cập và hiển thị hợp lý
* ‘*int \_questionNumber =0’ :* Biến được sử dụng để theo dõi chỉ số câu hỏi hiện tại trong danh sách ‘\_questionBank’ . được thiết lập từ đầu là ‘*0’ ( chỉ số của câu hỏi đầu tiên)*
* *‘List<Question>\_questionBank :*  danh sách chứa các đối tượng ‘Question’. Mỗi đối tượng bao gồm một câu hỏi và một câu trả lời tương ứng. Trong trường hợp này, danh sách chứa một số câu hỏi mẫu
* *Void nextQuestion() {…} :*  Kiểm tra câu hỏi tiếp theo trong danh sách(‘\_*questionNumber’ < số lượng câu hỏi trong danh sách -1)* thì tăng -*questionNumber* lên 1
* *String getQuestionText() {…} :*  trả về văn bản của câu hỏi hiện tại. (Truy cập vào danh sách ‘\_questionBank’ và lấy văn bản của câu hỏi hiện tại bằng ‘\_questionNumber’)
* *getCorrectAnswer :* Trả về câu trả lời đúng cho câu hỏi hiện tại. (Truy cập vào danh sách *\_questionBank* và lấy ra câu trả lời đúng của câu hỏi hiện tại bằng cách sử dụng *\_questionNumber*.)
* *isFinished :* Kiểm tra bài Quiz đã kết thúc chưa (Nếu *\_questionNumber* lớn hơn hoặc bằng số lượng câu hỏi trong danh sách trừ 1, tức là đã đến câu hỏi cuối cùng, phương thức trả về true, ngược lại trả về false.)
* *Reset :* đặt lại chỉ số của câu hỏi về 0, bắt đầu kiểm tra từ đầu (Gán \_questionNumber bằng 0)

CHƯƠNG II: CODE FILE PROJECT\_TASK\_5

1. MAIN.DART

**A screenshot of a computer program

Description automatically generated**

Hình 9: Main in Main.dart file project task 5

* Import

/*material :* Thư viện chứa widget và công cụ cần thiết để xây dựng giao diện người dùng trong Flutter

*/on\_hover\_test* : Đường dẫn tới tập tin liên quan đến tương tác của người dùng với widget khi di chuột

*/rflutter\_arlert :* Thư viện hiển thị các cảnh cáo (alerts) trong ứng dụng Flutter

*/quiz\_brain :* tập tin chứa logic và dữ liệu của Quiz

*/confetti :* Thư viện hiển thị hiệu ứng confetti khi người dùng hoàn thành bài kiểm traMain : Hàm khởi đầu của thư viện. Main được gọi khi ứng dụng khởi chạy.

* *Main* : Hàm khởi đầu của thư viện. Main được gọi khi ứng dụng khởi chạy
* *‘Quizzer extends Statelesswidget’ : Quizzer* kế thừ từ *StatelessWidget ,*  là một widget tĩnh, không có trạng thái thay đổi sau khi build. Không thể thay đổi nội dung sau khi được hiển thị ( ngược với StatefullWidget)
* *‘Widget build (BuildContext context)* : Hàm *build*  là phương thức bắt buộc phải ghi đè khi tạo một widget. Xây dựng và trả về cấu trúc của widget dựa trên ‘BuildContext’
* *‘MaterialApp’ :* widget gốc của một app Flutter, cung cấp tính năng cơ bản như điều hướng, chủ đề, ngôn ngữ
* *‘home: Scaffold’ :* widget khung của Flutter, cung cấp cấu trúc cơ bản cho một trang như AppBar, Drawer, BottomNavigationBar
* *‘backgroundColor: Colors.grey.shade900’ :* màu nền của Scaffold là màu xám
* *‘body: SafeArea’ :*  widget đảm bảo nội dung app không bị chèn
* *;child: Padding’ :* widget thêm các khoảng đệm xung quanh các widget con, thêm 10ps khoảng cách vào hai bên phải và trái
* *‘child : QuizPage() ‘ :* class khác trong app, hiển thị nội dung của trang quiz được đặt bên trong ‘*Padding’*

**2. Class \_QuizPageSate**

**A screen shot of a computer program

Description automatically generated**

A screen shot of a computer program

Description automatically generated

Hình 10 : class QuizPage & class \_QuizPageState

* ‘*checkAnswer’*  (bool userPickedAnswer)
* *setState(() :* Các lệnh xử lý sẽ được thực thi trong setState để cập nhật giao diện người dùng sau mỗi lần thay đổi trạng thái
* *if (quizBrain.isFinished()) {…}*

).show();

Kiểm tra nếu bài kiểm tra đã kết thúc

- if (correctAnswers > wrongAnswers) {

controller.play();

}

Đặt phát hiệu ứng pháo hoa nếu số câu đúng lớn hơn số câu sau

- ‘ quizBrain.reset();

scoreKeeper = [];’ : Đặt lại trạng thái bài kiểm tra

- ‘if (userPickedAnswer == correctAnswer)’: Nếu Quiz chưa kết thúc

Nếu câu trả lời của người dùng đúng

scoreKeeper.add(Icon(

Icons.check,

color: Colors.green,

));

correctAnswers++;

} else {

Nếu câu trả lời của người dùng sai

scoreKeeper.add(Icon(

Icons.close,

color: Colors.red,

));

wrongAnswers++;

}

// Di chuyển đến câu hỏi tiếp theo

- quizBrain.nextQuestion(); : Di chuyển đến câu hỏi tiếp theo

- Nếu trả lời đúng thêm biểu tượng ‘*Icon.check’* màu xanh lá vào ‘*scoreKeeper’*

- Nếu trả lời đúng thêm biểu tượng ‘*Icon.close’* màu đỏ vào ‘*scoreKeeper’*

**3. Widget build**

**A screen shot of a computer program

Description automatically generated**

Hình 11: widget build in project task 5

- Phương thức ‘build’ trong class ‘\_QuizPageState’ : xây dựng giao diện của Quiz.

- ‘bool is Hovering = false’ : Khai báo biến ‘isHovering’ để theo dõi xem chuột có đang hover qua widget không

- widget ‘stack’ : widget xếp chồng các widget lên nhau

- ‘expanded’ : hiển thị nội dung câu hỏi

- ‘quizBrain.getQuestionText(),’ :

A screen shot of a computer

Description automatically generated

Hình 12 : align

- ‘align’ : căn chỉnh thiết lập liên kết cho widget

- ‘confettiController : controller’ : tạo hiệu ứng pháo hoa giữa màn hình

- ‘colors: const [..] ‘ : random màu xanh,hồng,cam cho hiệu ứng confetti

1. ON\_HOVER\_TEST.DART

**A screen shot of a computer program

Description automatically generated**

Hình 13 : On\_Hover\_Test.dart

- import :

/material : thư ciện cung cấp các widget và hàm cần thiết cho ứng dụng

-‘class OnHover’ : onHover là một StatefulWidget là một trạng thái có thể thay đổi

-‘builder’ là hàm nhận tham số ‘isHovering’ kiểu bool trả về một Widget. Điều này cho phép người dùng cung cấp widget tùy chỉnh mà thay đổi dụa trên trạng thái hover

-‘createState’ : tạo ra một thể hiện của ‘\_OnHoverState’ là lớp giữ trạng thái của widget

-‘isHovered’ : biến isHovered kiểu bool để xác định xem chuột có đang ở trên widget hay không

-‘build’ :

+ ‘hovered’ : là một đối tượng mà ‘Matrix4’ sử dụng phép biến đổi dịch chuyển (translate) lên trên 10 đơn vị (0,-10,0)

+ ‘transform’ : quyết định xem sẽ sử dụng biến đổi ‘hovered’ hay ‘Matrix4.identity()’(không biến đổi) dựa vào trạng thái ‘isHovered’

+’MouseReigon’ : widget cho phép phát hiện sự di chuyển của chuột vào và ra khỏi vùng của nó. Nó có hai callback ‘onEnter’ và ‘onExit’ để cập nhật trạng thái hover.

+’AnimatedContainer’ : là một container có thể tự động animate các thay đổi về thời gian. Nó chứa widget được trả về từ hàm ‘builder’ với tham số ‘isHovered’

-‘onEntered’ : kiểu bool, hàm cập nhật trạng thái ‘isHovered’ và gọi ‘setState’ để rebuild widget với trạng thái mới

-‘onHover’ : cho phép người dùng truyền vào một hàm builder để tạo widget con. Widget này thay đổi hình dạng khi chuột di chuyển vào và ra khỏi vùng của nó, sử dụng một hiệu ứng animate để dịch chuyển widget lên trên khi hover

1. QUESTION.DART

- Class ‘Question’ đinh nghĩa một câu hỏi với văn bản câu hỏi và câu trả lời đúng hoặc sai. Sử dụng ‘late’ cho phép các thuộc tính được khởi tạo sau khi đối tượng được tạo ra. Sử dụng Constructor của lớp nhận các giá trị tham số để khởi tạo các thuộc tính này

- Class giúp tạo ra các đối tượng ‘Question’ với các thuộc tính ‘questionText’ và ‘questionAnswer’ là nền tảng để quản lý các câu hỏi trong Quiz

1. QUIZ\_BRAIN.DART

* Class ‘QuizzBrain’ cung cấp một cơ chế quản lý danh sách câu hỏi . bằng cách sử dụng ‘*\_questionNumber’*  có thể xác định câu hỏi hiện tại để hiển thị. Danh sách ‘*questionBank*’ chứa tất cả các câu hỏi, cho phép ứng dụng truy cập và hiển thị hợp lý
* ‘*int \_questionNumber =0’ :* Biến được sử dụng để theo dõi chỉ số câu hỏi hiện tại trong danh sách ‘\_questionBank’ . được thiết lập từ đầu là ‘*0’ ( chỉ số của câu hỏi đầu tiên)*
* *‘List<Question>\_questionBank :*  danh sách chứa các đối tượng ‘Question’. Mỗi đối tượng bao gồm một câu hỏi và một câu trả lời tương ứng. Trong trường hợp này, danh sách chứa một số câu hỏi mẫu
* *Void nextQuestion() {…} :*  Kiểm tra câu hỏi tiếp theo trong danh sách(‘\_*questionNumber’ < số lượng câu hỏi trong danh sách -1)* thì tăng -*questionNumber* lên 1
* *String getQuestionText() {…} :*  trả về văn bản của câu hỏi hiện tại. (Truy cập vào danh sách ‘\_questionBank’ và lấy văn bản của câu hỏi hiện tại bằng ‘\_questionNumber’)
* *getCorrectAnswer :* Trả về câu trả lời đúng cho câu hỏi hiện tại. (Truy cập vào danh sách *\_questionBank* và lấy ra câu trả lời đúng của câu hỏi hiện tại bằng cách sử dụng *\_questionNumber*.)
* *isFinished :* Kiểm tra bài Quiz đã kết thúc chưa (Nếu *\_questionNumber* lớn hơn hoặc bằng số lượng câu hỏi trong danh sách trừ 1, tức là đã đến câu hỏi cuối cùng, phương thức trả về true, ngược lại trả về false.)
* *Reset :* đặt lại chỉ số của câu hỏi về 0, bắt đầu kiểm tra từ đầu (Gán \_questionNumber bằng 0)

CHƯƠNG III: TỔNG HỢP

* Project mini trên giúp hiểu và sử dụng được package đúng cách, cách viết mã nguồn ứng dụng, sử dụng package, library
* Về phần logic : các câu hỏi True/ False, cơ chế hoạt động của quản lý danh sách câu hỏi, sắp xếp các widget phù hợp
* Khai báo package Confetti cho phép người dùng tạo hiệu ứng pháo bông ngay giữa màn hình.
* OnHover tạo hiệu ứng cho khi hover mouse vào các nút, cho phép người dùng truyền vào một hàm builder để tạo widget con. Widget này thay đổi hình dạng khi chuột di chuyển vào và ra khỏi vùng của nó, sử dụng một hiệu ứng animate để dịch chuyển widget lên trên khi hover