



UNIVERSITI MALAYSIA TERENGGANU

FACULTY OF OCEAN ENGINEERING TECHNOLOGY & INFORMATICS

[CSM3123]

NATIVE MOBILE PROGRAMING

(GROUP 1)

LAB REPORT 4

PREPARED BY:

NUR ELYA FARHANA BINTI ZAINORDIN (S63723)

PREPARED FOR:

DR. RABIEI BIN MAMAT

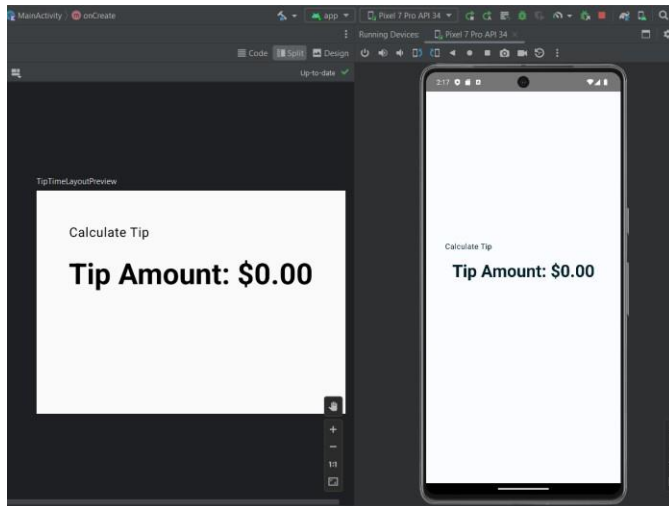
[BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONOURS]

SEMESTER I 2023/2024

Task 1

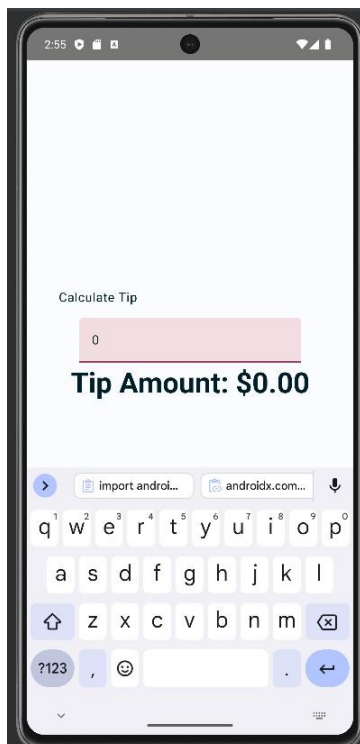
Follow this link for the instructions:

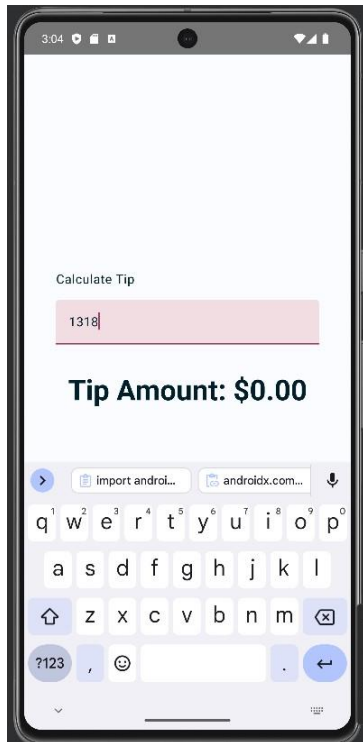
<https://developer.android.com/codelabs/basic-android-kotlin-compose-using-state>



Tip Time

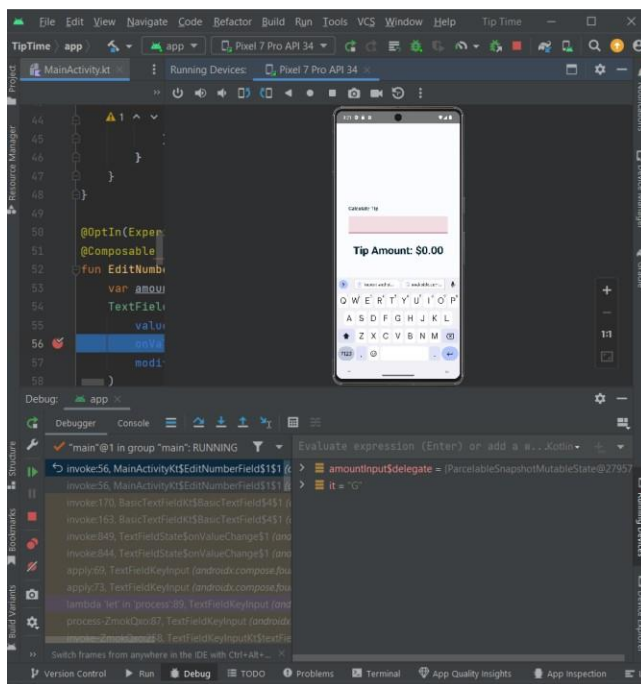
The composition



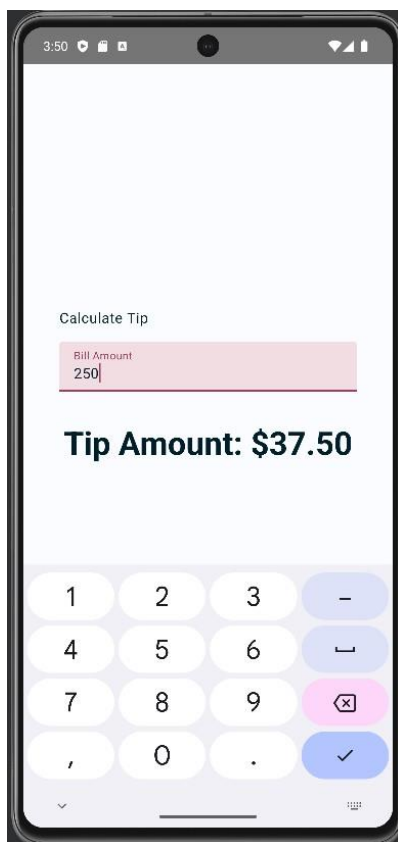
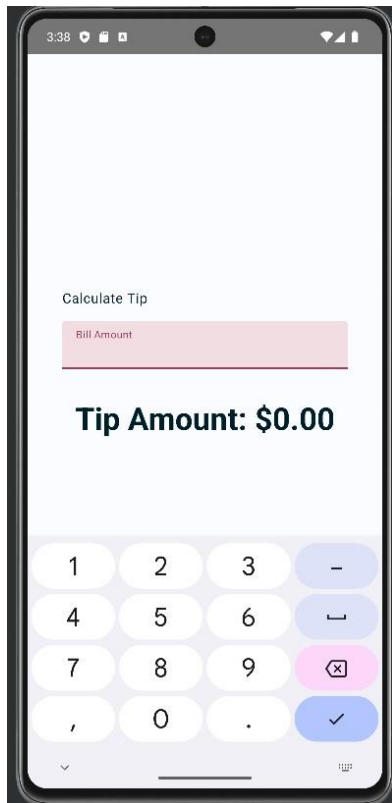


Use remember function to save state

State and recomposition in action



Modify the appearance



State hoisting

Task 2

Follow this link for the instructions:

<https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-1>

Generics, objects and extensions Use an enum constant

```
enum class Difficulty {
    EASY, MEDIUM, HARD
}

class Question<T> {
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
}

fun main() {
    val question1 = Question<String>("Quoth the raven __", "nevermore", Difficulty.MEDIUM)
    val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    println(question1.toString())
}
```

Question@1b2c6ec2

```
enum class Difficulty {
    EASY, MEDIUM, HARD
}

data class Question<T> {
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
}

fun main() {
    val question1 = Question<String>("Quoth the raven __", "nevermore", Difficulty.MEDIUM)
    val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    println(question1.toString())
}
```

Question(questionText=Quoth the raven __, answer=nevermore, difficulty=MEDIUM)

```
enum class Difficulty {
    EASY, MEDIUM, HARD
}

class Question<T> {
    val questionText: String,
    val answer: T,
    val difficulty: Difficulty
}

class Quiz {
    val question1 = Question<String>("Quoth the raven __", "nevermore", Difficulty.MEDIUM)
    val question2 = Question<Boolean>("The sky is green. True or false", false, Difficulty.EASY)
    val question3 = Question<Int>("How many days are there between full moons?", 28, Difficulty.HARD)
    fun printQuiz() {
        question1.let {
            println(it.questionText)
            println(it.answer)
            println(it.difficulty)
        }
        println()
        question2.let {
            println(it.questionText)
            println(it.answer)
            println(it.difficulty)
        }
        println()
        question3.let {
            println(it.questionText)
            println(it.answer)
            println(it.difficulty)
        }
        println()
    }
}

fun main() {
    val quiz = Quiz()
    quiz.printQuiz()
}
```

Quoth the raven __
nevermore
MEDIUM

The sky is green. True or false
false
EASY

How many days are there between full moons?
28
HARD

Convert question to a data class

```

class Cookie{
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
}

val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)

fun main() {
    cookies.forEach {
        println("Menu item: $it")
    }
}

```

```

Menu item: Cookie@4769b07b
Menu item: Cookie@6f539caf
Menu item: Cookie@79fc0f2f
Menu item: Cookie@50040f0c
Menu item: Cookie@2dda6444
Menu item: Cookie@5e9f23b4
Menu item: Cookie@4783da3f

```

Target platform: JVM -- Running on JDK

forEach()

```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

val cookies = listOf(
    Cookie(
        name = "Chocolate Chip",
        softBaked = false,
        hasFilling = false,
        price = 1.69
    ),
    Cookie(
        name = "Banana Walnut",
        softBaked = true,
        hasFilling = false,
        price = 1.49
    ),
    Cookie(
        name = "Vanilla Creme",
        softBaked = false,
        hasFilling = true,
        price = 1.59
    ),
    Cookie(
        name = "Chocolate Peanut Butter",
        softBaked = false,
        hasFilling = true,
        price = 1.49
    ),
    Cookie(
        name = "Snickerdoodle",
        softBaked = true,
        hasFilling = false,
        price = 1.39
    ),
    Cookie(
        name = "Blueberry Tart",
        softBaked = true,
        hasFilling = true,
        price = 1.79
    ),
    Cookie(
        name = "Sugar and Sprinkles",
        softBaked = false,
        hasFilling = false,
        price = 1.39
    )
)

fun main() {
    cookies.forEach {
        println("Menu item: ${it.name}")
    }
}

```

```

Menu item: Chocolate Chip
Menu item: Banana Walnut
Menu item: Vanilla Creme
Menu item: Chocolate Peanut Butter
Menu item: Snickerdoodle
Menu item: Blueberry Tart
Menu item: Sugar and Sprinkles

```

Target platform: JVM - Running on IntelliJ

map()

```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

fun main() {
    val cookies = listOf(
        Cookie(
            name = "Chocolate Chip",
            softBaked = false,
            hasFilling = false,
            price = 1.69
        ),
        Cookie(
            name = "Banana Walnut",
            softBaked = true,
            hasFilling = false,
            price = 1.49
        ),
        Cookie(
            name = "Vanilla Creme",
            softBaked = false,
            hasFilling = true,
            price = 1.59
        ),
        Cookie(
            name = "Chocolate Peanut Butter",
            softBaked = false,
            hasFilling = true,
            price = 1.49
        ),
        Cookie(
            name = "Snickerdoodle",
            softBaked = true,
            hasFilling = false,
            price = 1.39
        ),
        Cookie(
            name = "Blueberry Tart",
            softBaked = true,
            hasFilling = true,
            price = 1.79
        ),
        Cookie(
            name = "Sugar and Sprinkles",
            softBaked = false,
            hasFilling = false,
            price = 1.39
        )
    )

    val fullMenu = cookies.map {
        "${it.name} - ${it.price}"
    }

    println("Full menu:")
    fullMenu.forEach {
        println(it)
    }
}

```

```

Full menu:
Chocolate Chip - $1.69
Banana Walnut - $1.49
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Sugar and Sprinkles - $1.39

```



```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

fun main() {
    val cookies = listOf(
        Cookie(
            name = "Chocolate Chip",
            softBaked = false,
            hasFilling = false,
            price = 1.69
        ),
        Cookie(
            name = "Banana Walnut",
            softBaked = true,
            hasFilling = false,
            price = 1.49
        ),
        Cookie(
            name = "Vanilla Creme",
            softBaked = false,
            hasFilling = true,
            price = 1.59
        ),
        Cookie(
            name = "Chocolate Peanut Butter",
            softBaked = false,
            hasFilling = true,
            price = 1.49
        ),
        Cookie(
            name = "Snickerdoodle",
            softBaked = true,
            hasFilling = false,
            price = 1.39
        ),
        Cookie(
            name = "Blueberry Tart",
            softBaked = true,
            hasFilling = true,
            price = 1.79
        ),
        Cookie(
            name = "Sugar and Sprinkles",
            softBaked = false,
            hasFilling = false,
            price = 1.39
        )
    )

    val softBakedMenu = cookies.filter {
        it.softBaked
    }

    println("Soft cookies:")
    softBakedMenu.forEach {
        println("${it.name} - ${it.price}")
    }
}

```

```

Soft cookies:
Banana Walnut - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79

```

Chapter 10: Functional Programming

filter()

groupBy()

```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

fun main() {
    val cookies = listOf(
        Cookie(
            name = "Chocolate Chip",
            softBaked = false,
            hasFilling = false,
            price = 1.69
        ),
        Cookie(
            name = "Banana Walnut",
            softBaked = true,
            hasFilling = false,
            price = 1.49
        ),
        Cookie(
            name = "Vanilla Creme",
            softBaked = false,
            hasFilling = true,
            price = 1.59
        ),
        Cookie(
            name = "Chocolate Peanut Butter",
            softBaked = false,
            hasFilling = true,
            price = 1.49
        ),
        Cookie(
            name = "Snickerdoodle",
            softBaked = true,
            hasFilling = false,
            price = 1.39
        ),
        Cookie(
            name = "Blueberry Tart",
            softBaked = true,
            hasFilling = true,
            price = 1.79
        ),
        ..
        Cookie(
            name = "Sugar and Sprinkles",
            softBaked = false,
            hasFilling = false,
            price = 1.39
        )
    )

    val groupedMenu = cookies.groupBy { it.softBaked }
    val softBakedMenu = groupedMenu[true] ?: listOf()
    val crunchyMenu = groupedMenu[false] ?: listOf()

    println("Soft cookies:")
    softBakedMenu.forEach {
        println("${it.name} - ${it.price}")
    }
    println("Crunchy cookies:")
    crunchyMenu.forEach {
        println("${it.name} - ${it.price}")
    }
}

```

```

Soft cookies:
Banana Walnut - $1.49
Snickerdoodle - $1.39
Blueberry Tart - $1.79
Crunchy cookies:
Chocolate Chip - $1.69
Vanilla Creme - $1.59
Chocolate Peanut Butter - $1.49
Sugar and Sprinkles - $1.39

```

Target platform: JVM - Running on Kotlin v.

fold()

```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

fun main() {
    val cookies = listOf(
        Cookie(
            name = "Chocolate Chip",
            softBaked = false,
            hasFilling = false,
            price = 1.69
        ),
        Cookie(
            name = "Banana Walnut",
            softBaked = true,
            hasFilling = false,
            price = 1.49
        ),
        Cookie(
            name = "Vanilla Creme",
            softBaked = false,
            hasFilling = true,
            price = 1.59
        ),
        Cookie(
            name = "Chocolate Peanut Butter",
            softBaked = false,
            hasFilling = true,
            price = 1.49
        ),
        Cookie(
            name = "Snickerdoodle",
            softBaked = true,
            hasFilling = false,
            price = 1.39
        ),
        Cookie(
            name = "Blueberry Tart",
            softBaked = true,
            hasFilling = true,
            price = 1.79
        ),
        Cookie(
            name = "Sugar and Sprinkles",
            softBaked = false,
            hasFilling = false,
            price = 1.39
        )
    )
    val totalPrice = cookies.fold(0.0) { total, cookie ->
        total + cookie.price
    }
    println("Total price: $$${totalPrice}")
}

```

Total price: \$10.83

Target platform: JVM - Running on Kotlin: 1.9.3

sortedBy()

```

class Cookie(
    val name: String,
    val softBaked: Boolean,
    val hasFilling: Boolean,
    val price: Double
)

fun main() {
    val cookies = listOf(
        Cookie(
            name = "Chocolate Chip",
            softBaked = false,
            hasFilling = false,
            price = 1.69
        ),
        Cookie(
            name = "Banana Walnut",
            softBaked = true,
            hasFilling = false,
            price = 1.49
        ),
        Cookie(
            name = "Vanilla Creme",
            softBaked = false,
            hasFilling = true,
            price = 1.59
        ),
        Cookie(
            name = "Chocolate Peanut Butter",
            softBaked = false,
            hasFilling = true,
            price = 1.49
        ),
        Cookie(
            name = "Snickerdoodle",
            softBaked = true,
            hasFilling = false,
            price = 1.39
        ),
        Cookie(
            name = "Blueberry Tart",
            softBaked = true,
            hasFilling = true,
            price = 1.79
        ),
        Cookie(
            name = "Sugar and Sprinkles",
            softBaked = false,
            hasFilling = false,
            price = 1.39
        )
    )
    val alphabeticalMenu = cookies.sortedBy {
        it.name
    }
    println("Alphabetical menu:")
    alphabeticalMenu.forEach {
        println(it.name)
    }
}

```

```

Alphabetical menu:
Banana Walnut
Blueberry Tart
Chocolate Chip
Chocolate Peanut Butter
Snickerdoodle
Sugar and Sprinkles
Vanilla Creme

```

```

data class Event(
    val title: String,
    val description: String? = null,
    val daypart: String,
    val durationInMinutes: Int
)

fun main() {
    val studyEvent = Event(
        title = "Study Kotlin",
        description = "Commit to studying Kotlin at least 15 minutes per day.",
        daypart = "Evening",
        durationInMinutes = 15
    )
    println(studyEvent)
}

```

```
Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes p
```

```
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING,
}

data class Event(
    val title: String,
    val description: String? = null,
    val daypart: Daypart,
    val durationInMinutes: Int,
)

fun main() {
    val studyEvent = Event(
        title = "Study Kotlin",
        description = "Commit to studying Kotlin at least 15 minutes per day.",
        daypart = Daypart.EVENING,
        durationInMinutes = 15
    )

    println(studyEvent)
}
```

Event(title=Study Kotlin, description=Commit to studying Kotlin at least 15 minutes per

```
enum class Daypart {
    MORNING,
    AFTERNOON,
    EVENING,
}


data class Event(
    val title: String,
    val description: String? = null,
    val daypart: Daypart,
    val durationInMinutes: Int
)

fun main() {
    val event1 = Event(title = "Wake up", description = "Time to get up", daypart = Daypart.MORNING, durationInMinutes = 10)
    val event2 = Event(title = "Eat breakfast", daypart = Daypart.MORNING, durationInMinutes = 10)
    val event3 = Event(title = "Learn about Kotlin", daypart = Daypart.AFTERNOON, durationInMinutes = 10)
    val event4 = Event(title = "Practice Compose", daypart = Daypart.AFTERNOON, durationInMinutes = 10)
    val event5 = Event(title = "Watch latest DevBytes video", daypart = Daypart.AFTERNOON, durationInMinutes = 10)
    val event6 = Event(title = "Check out latest Android Jetpack library", daypart = Daypart.EVENING, durationInMinutes = 10)

    val events = mutableListOf(event1, event2, event3, event4, event5, event6)

    events.forEachIndexed { index, event ->
        println("Event ${index + 1}:")
        println("Title: ${event.title}")
        println("Description: ${event.description ?: "No description"}")
        println("Daypart: ${event.daypart}")
        println("Duration: ${event.durationInMinutes} minutes")
        println()
    }
}
```

Quiz


Essentials ▾ Design & Plan ▾ Develop ▾ Google Play Community
Search
English ▾ Android Studio
⋮ a

- Which of the following is a valid way to define a data class in Kotlin?

☒ `data class Person(val name: String, val age: Int)` Correct!
☐ `class Person(val name: String, val age: Int): data`
☐ `class Person(val name: String, val age: Int)`
☐ `data class Person(val name: String, val age: Int)`
- When using a sealed class, all direct subclasses must be in the same package.

☒ true Correct!
☐ false
- When using generics, the generic data type goes inside _____.

☐ {}
☐ {}
☐ ()
☒ <> Correct!

Developers

Essentials ▾

Design & Plan ▾

Develop ▾

Google Play

Community

Search

English ▾

Android Studio

⋮

8

6. Which of the following are higher-order functions?

Choose as many answers as you see fit.

☒ `map` Correct!

☐ `arrangeBy`

☒ `filter` Correct!

☒ `forEach` Correct!

7. Given the following line of code, which of the following commands will print `Blue` ?
(Hint: If you are not sure, try running the code in the Kotlin Playground.)

```
val colors = listOf("Red", "Green", "Blue")
```

Choose as many answers as you see fit.

☒ `println(colors[2])` Correct!

☒ `println(colors.get(2))` Correct!

☐ `println(colors.contains(2))`

☐ `println(colors.getOrElse(index = 2, defaultValue = 10))`

Developers

Essentials ▾

Design & Plan ▾

Develop ▾

Google Play

Community

Search

English ▾

Android Studio

⋮

8

☐ `println(colors.getOrElse(index = 2, defaultValue = 10))`

8. The programming concept of a class that has only one instance is called a ____.

☐ Uniqueness

☒ Singleton Correct!

☐ Mono-object

☐ Lambda

9. Which of the following statements is true regarding sets and maps?

☒ A set must contain distinct values and a map's keys must be distinct. Correct!

☐ A set must contain distinct values and a map's keys can contain duplicates.

☐ A set can contain duplicate values and a map's values can contain duplicate values.

☐ A set can contain duplicate values and a map's keys must be distinct.

Developers

Essentials ▾

Design & Plan ▾

Develop ▾

Google Play

Community

Search

English ▾

Android Studio

⋮

8

☒ `enum` Correct!

☐ sealed

☐ data

☐ inherited

4. A(n) ____ class is useful when you have a fixed set of values.

☒ `enum` Correct!

☐ sealed

☐ data

☐ inherited

5. To create a list object that has the ability to change its size, you would call ____.

☐ `modifiableList()`

☐ `immutableList()`

☐ `listOf()`

☒ `mutableList()` Correct!

10. If you have a variable named `records`, which is a collection, to determine the number of items it contains, you can call ____.

☐ `records.length`

☐ `records.quantity`

☐ `len(records)`

☒ `records.size`

✓ Correct!

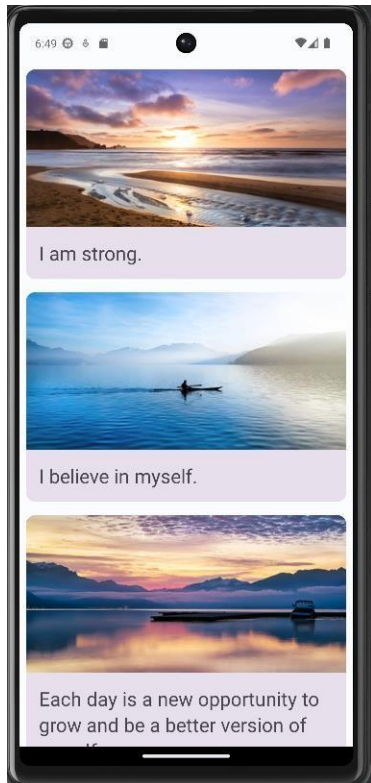
Results

You scored **10 out of 10**. Congratulations! You have passed this quiz.

Task 3

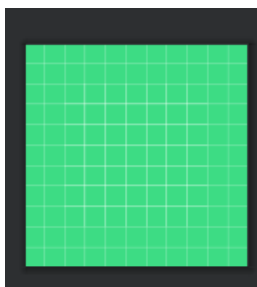
Follow this link for the instructions:

<https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-2>



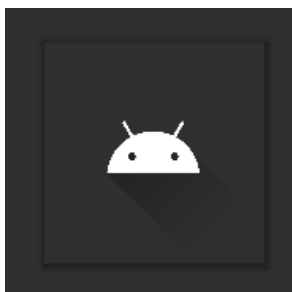
Activity 2

Activity 3



Background:

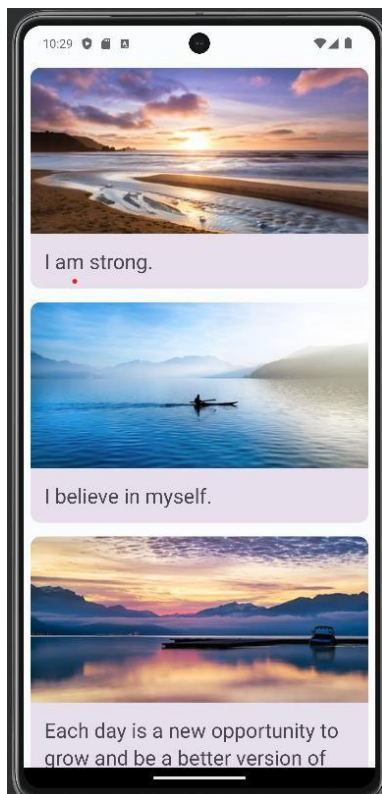
Foreground:



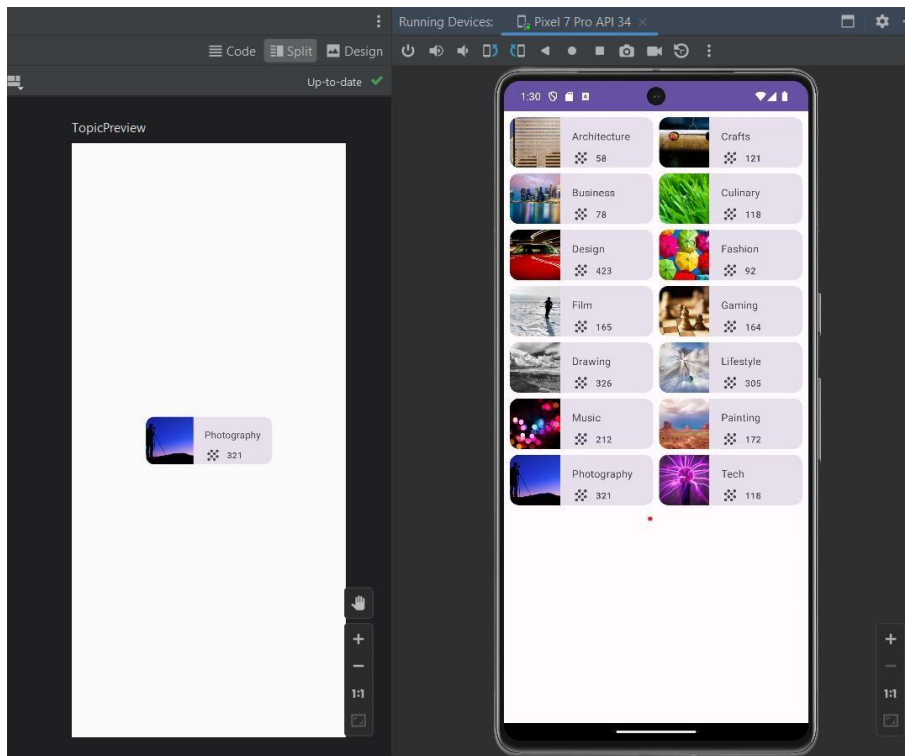


Test app on device(emulator)

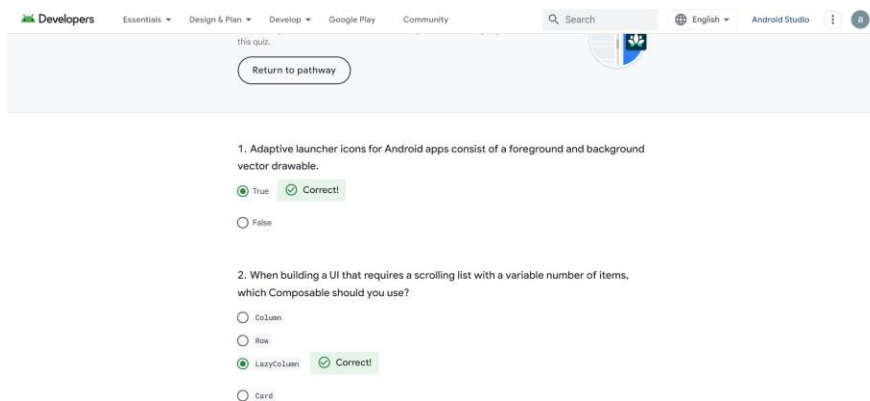
Final result



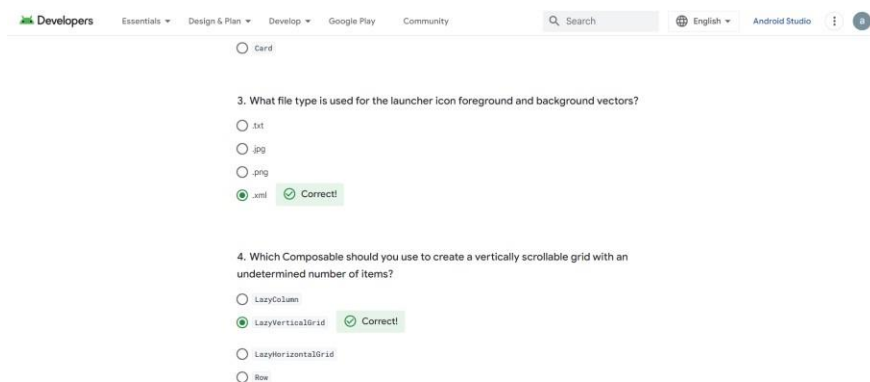
Activity 4

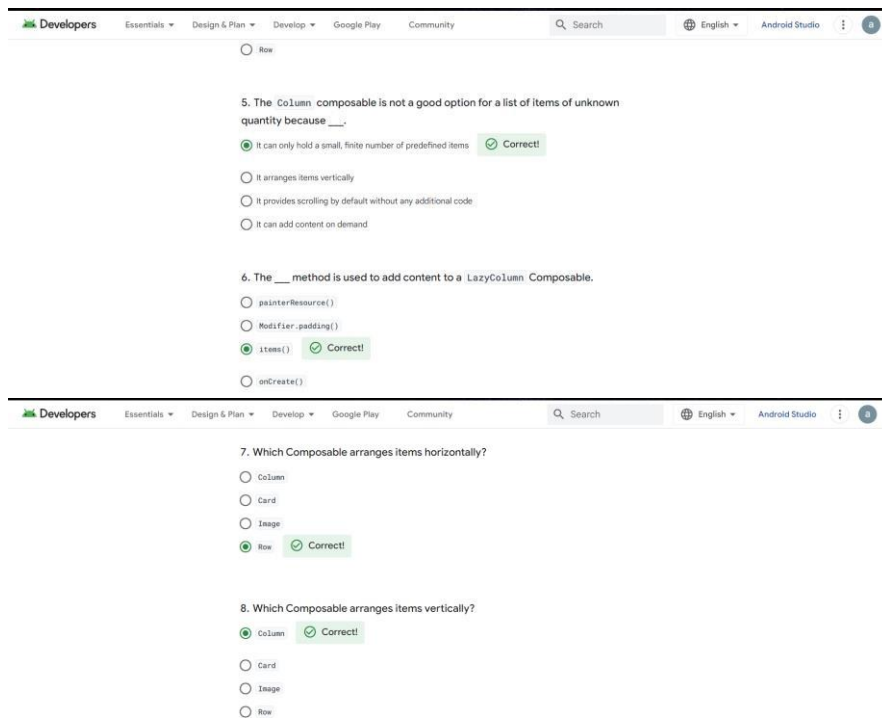


Final result



Quiz



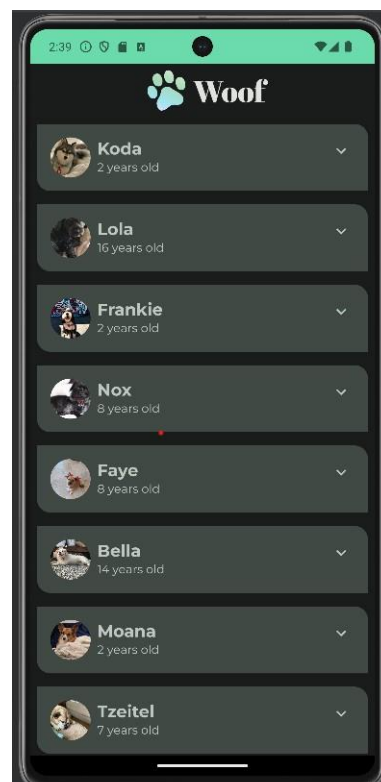
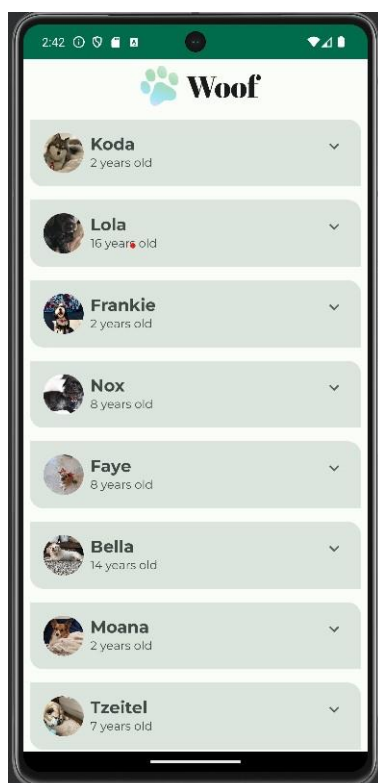


Task 4

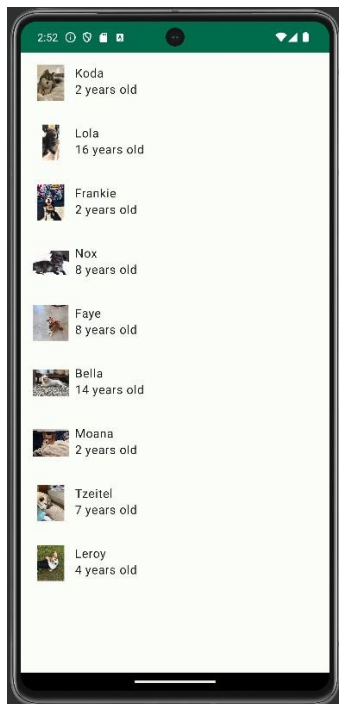
Follow this link for the instructions:

<https://developer.android.com/courses/pathways/android-basics-compose-unit-3-pathway-3>

Activity 2

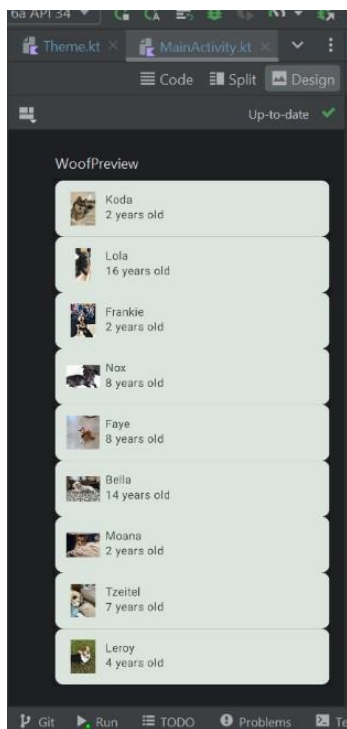


Light theme and Dark theme

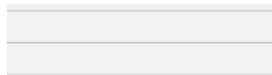
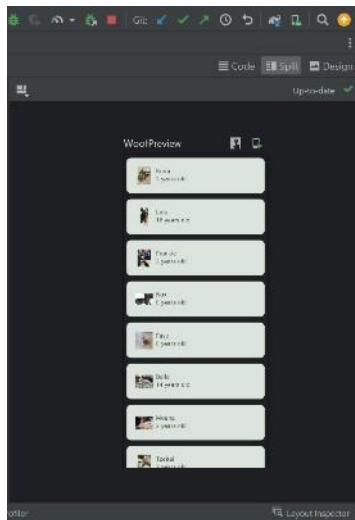


Color changed

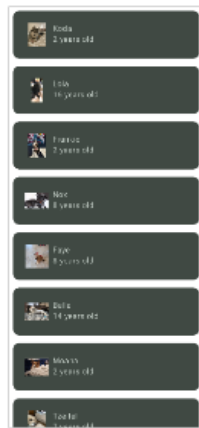
Add shape



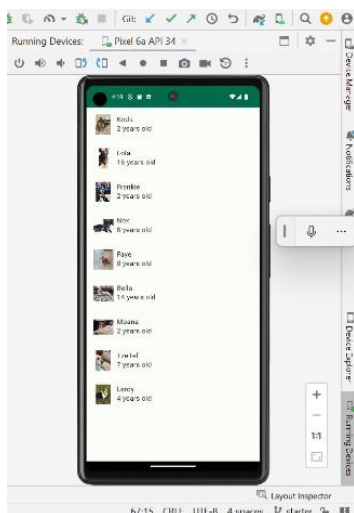
Add typography

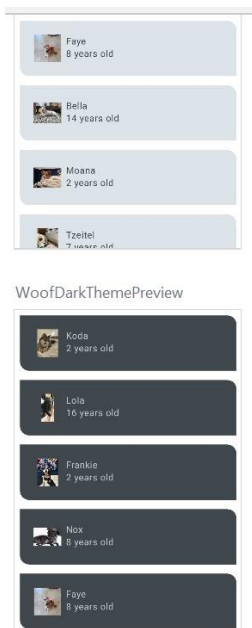
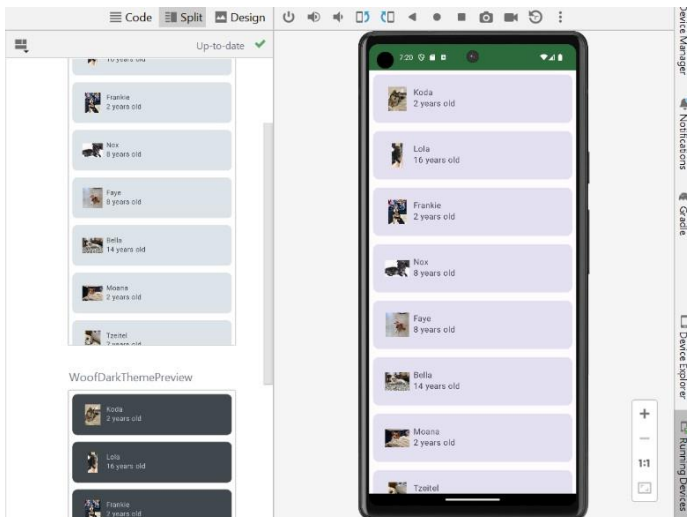
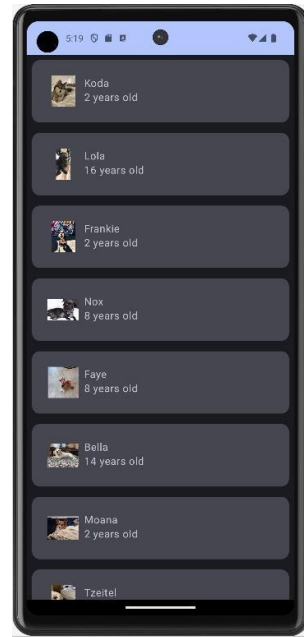
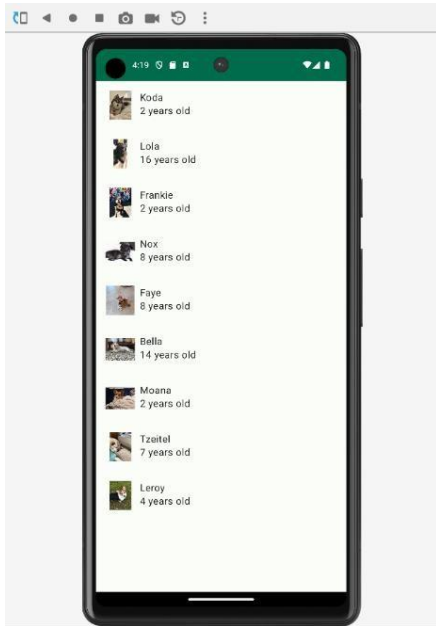


WoofDarkThemePreview

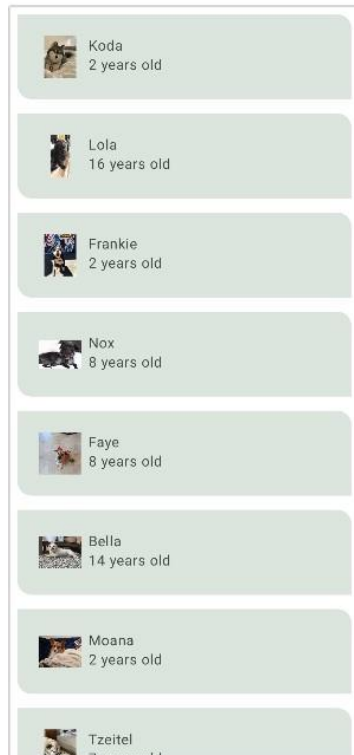


WoofPreview

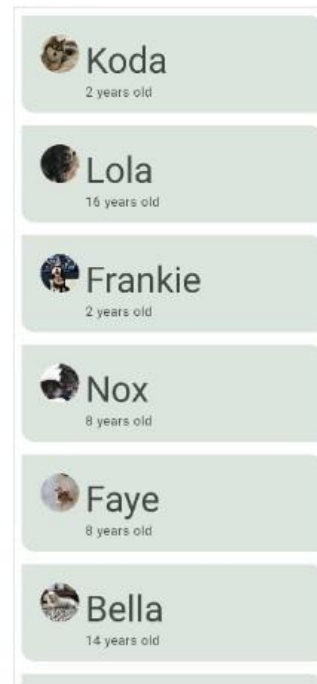




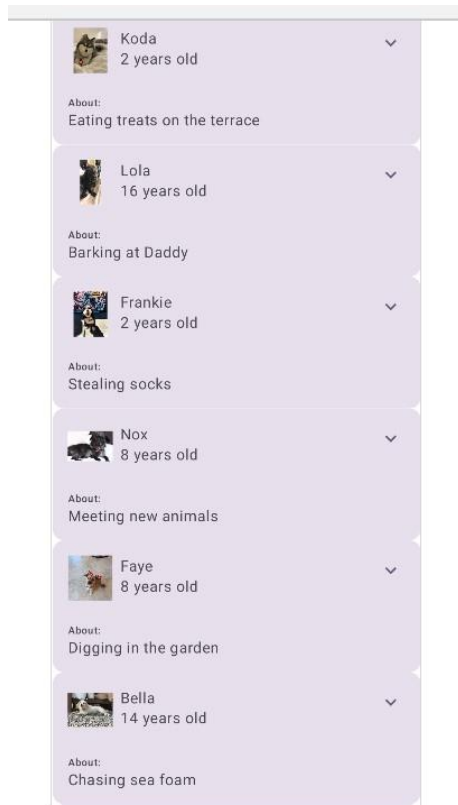
WoofPreview

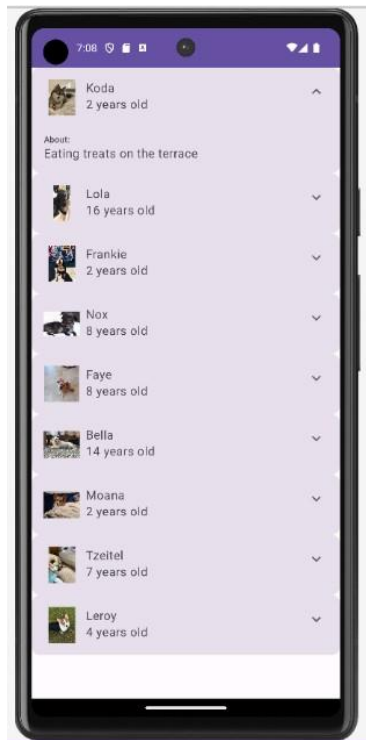
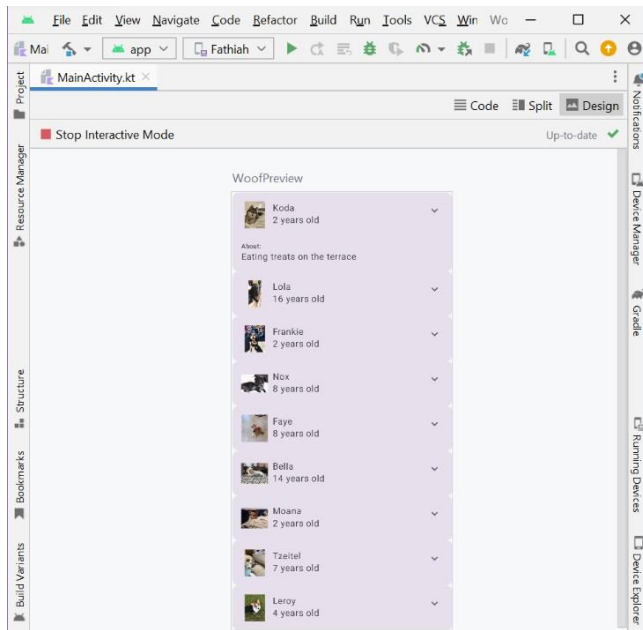


WoofPreview

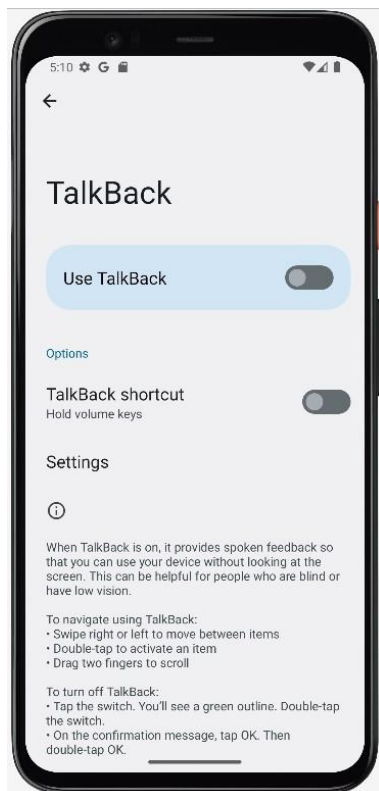
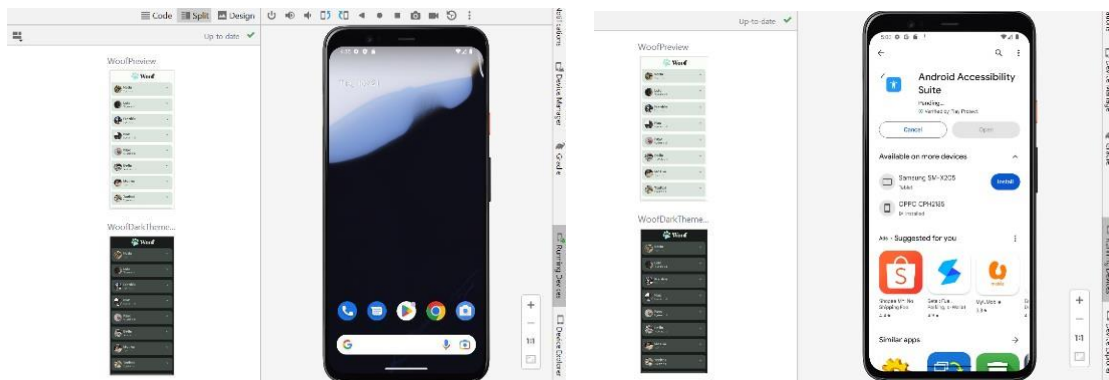


Simple animation





Testing for Accessibility



Quiz

1. Animations in your Android app can:

- ☐ Add visual cues about what's going on in your app.
- ☐ Add a polished look to your app.
- ☐ Help the user see what changed.
- ☒ All of the above. Correct!

2. Spring animation is based on:

- ☐ Start and end values over the specified duration.
- ☒ Damping ratio and stiffness. Correct!
- ☐ Snapshot values specified at different timestamps.
- ☐ Interpolation between two keyframe values.

3. Spring animation is a physics-based animation driven by spring force.

- ☒ True Correct!

Developers
Essentials
Design & Plan
Develop
Google Play
Community
Search
English
Android Studio
8

☐ False

4. In the Material theming, the ___ color is the color displayed most frequently across your app's screens and components.

☒ primary Correct!

☐ secondary

☐ surface

☐ background

5. The following file is used to define shapes of components in Compose.

☐ Theme.kt

☐ Color.kt

☒ Shape.kt Correct!

☐ Colors.kt

Developers
Essentials
Design & Plan
Develop
Google Play
Community
Search
English
Android Studio
8

6. You can only have one @Preview composable

☐ True

☒ False Correct!

7. A hex color code starts with a pound (#) character, and is followed by six letters and/or numbers that represent the red, green, and blue (RGB) components of that color.

☒ True Correct!

☐ False

8. The ___ file is the file that holds all the information about the theme of the app which is defined through color, shape, and typography.

☒ Theme.kt Correct!

☐ Color.kt

☐ Shape.kt

Developers
Essentials
Design & Plan
Develop
Google Play
Community
Search
English
Android Studio
8

☐ Colors.kt

9. ___ creates contrast between the Card and the background by adding a shadow to make the app look more realistic and visually interesting?

☒ Elevation Correct!

☐ Shape

☐ Color

☐ Theme

10. What are reasons that someone may use Dark Theme on their device?

☐ It can reduce power usage by a significant amount (depending on the device's screen technology).

☐ It improves visibility for users with low vision and those who are sensitive to bright light.

☐ It makes it easier for anyone to use a device in a low-light environment.

☒ All of the above. Correct!

Developers
Essentials
Design & Plan
Develop
Google Play
Community
Search
English
Android Studio
8

11. TalkBack allows a user to navigate an app using switches instead of the touchscreen.

☐ True

☒ False Correct!

12. Which attribute allows TalkBack to speak a meaningful representation of an image or icon?

☐ elevation

☐ shape

☒ contentDescription Correct!

☐ style

Results

You scored 12 out of 12. Congratulations! You have passed this quiz.

Link github

<https://github.com/Ainin24/CSM31>