Nama : M Rezky Raya Kilwouw

NIM : 222313190

Kelas : 3SI1

PRAKTIKUM 12

PEMROGRAMAN PLATFORM KHUSUS

1. Sebelum memulai

Dalam codelab ini, Saya akan mempelajari cara membuat daftar yang dapat di-scroll di aplikasi menggunakan Jetpack Compose.

Saya akan mengerjakan aplikasi Affirmations, yang menampilkan daftar afirmasi yang dipasangkan dengan gambar indah untuk membawa hal positif ke hari Saya.

Data sudah ada, Saya hanya perlu mengambil data tersebut dan menampilkannya di UI.

Prasyarat

- Pemahaman tentang daftar di Kotlin
- Pengalaman membuat tata letak dengan Jetpack Compose
- Pengalaman menjalankan aplikasi di perangkat atau emulator

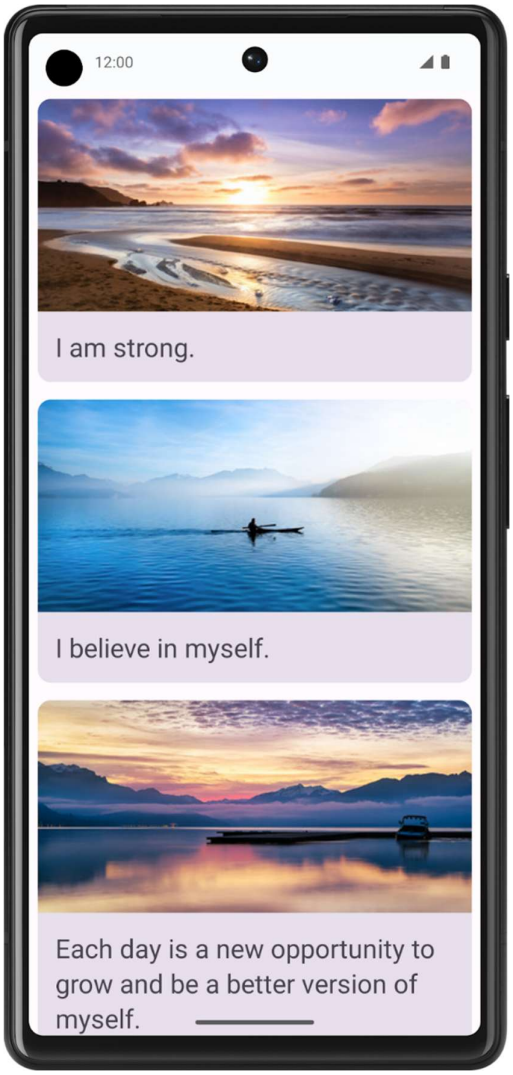
Yang akan Saya pelajari

- Cara membuat kartu Desain Material menggunakan Jetpack Compose
- Cara membuat daftar yang dapat di-scroll menggunakan Jetpack Compose

Yang akan Saya bangun

- Saya akan mengambil aplikasi yang sudah ada dan menambahkan daftar yang dapat di-scroll ke UI

Produk jadi akan terlihat seperti ini:



Yang akan Saya butuhkan

- Komputer dengan akses internet, browser web, dan Android Studio
- Akses ke GitHub

Mendownload kode awal

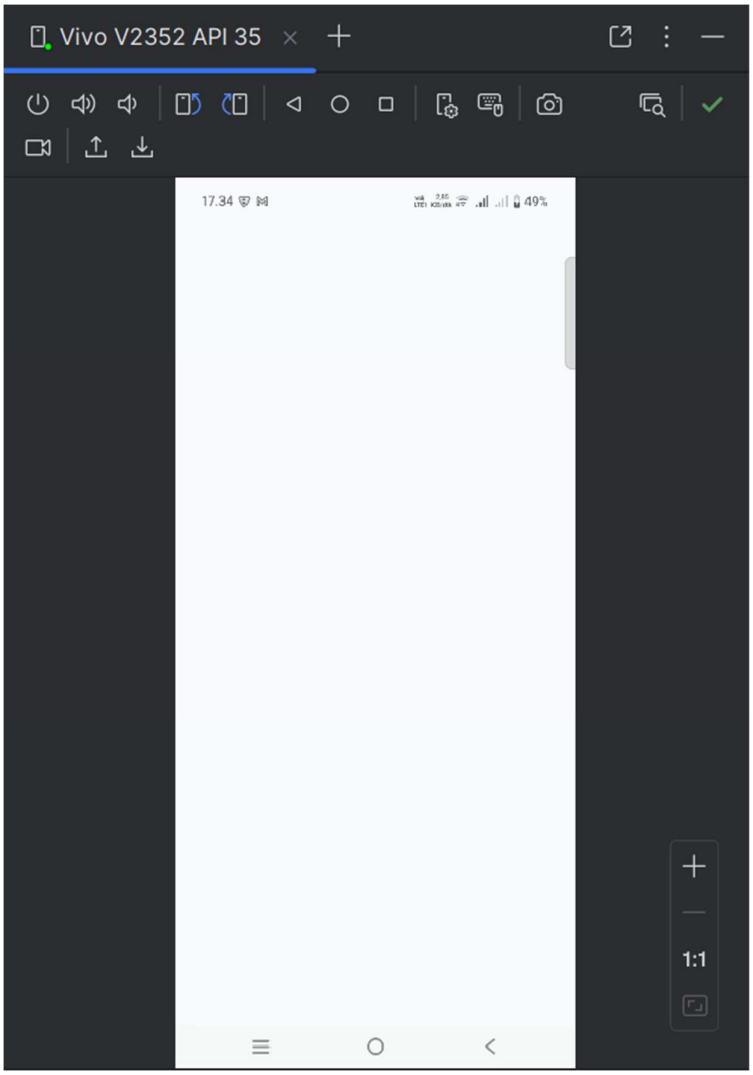
Di Android Studio, buka folder basic-android-kotlin-compose-training-affirmations.

URL kode awal:

https://github.com/google-developer-training/basic-android-kotlin-compose-training-affirmations

Nama cabang dengan kode awal: starter

Aplikasi diharapkan menampilkan layar kosong saat dibangun dari kode cabang starter.

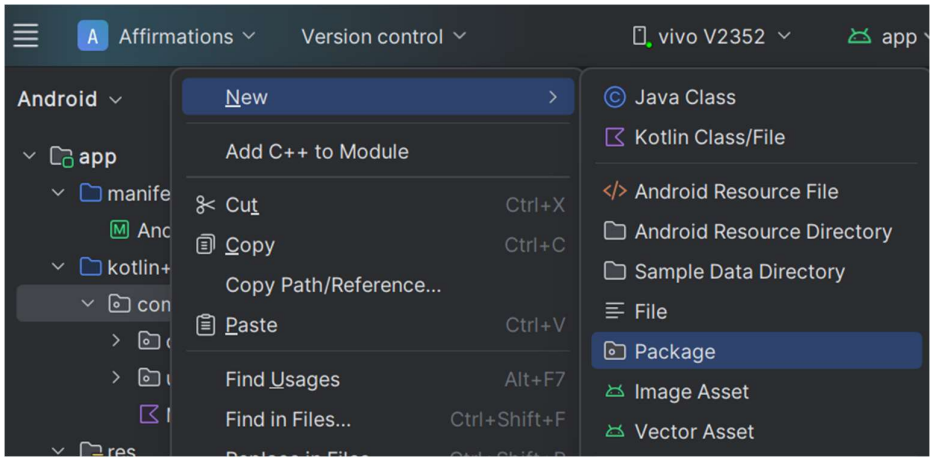


2. Membuat class data item daftar

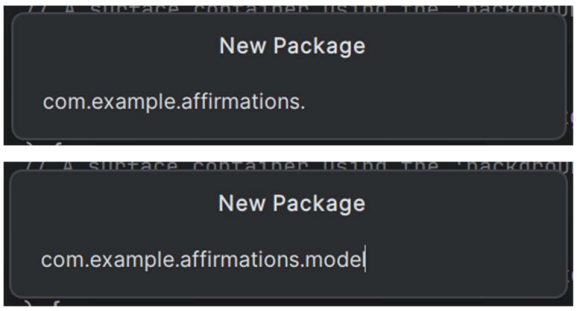
Membuat class data untuk Affirmation

Di aplikasi Android, daftar terdiri dari item daftar. Untuk data tunggal, ini bisa berupa hal sederhana seperti string atau bilangan bulat. Untuk item daftar yang memiliki beberapa data, seperti gambar dan teks, Saya memerlukan class yang berisi semua properti ini. Class data adalah jenis class yang hanya berisi properti. Class tersebut dapat menyediakan beberapa metode utilitas agar berfungsi dengan properti tersebut.

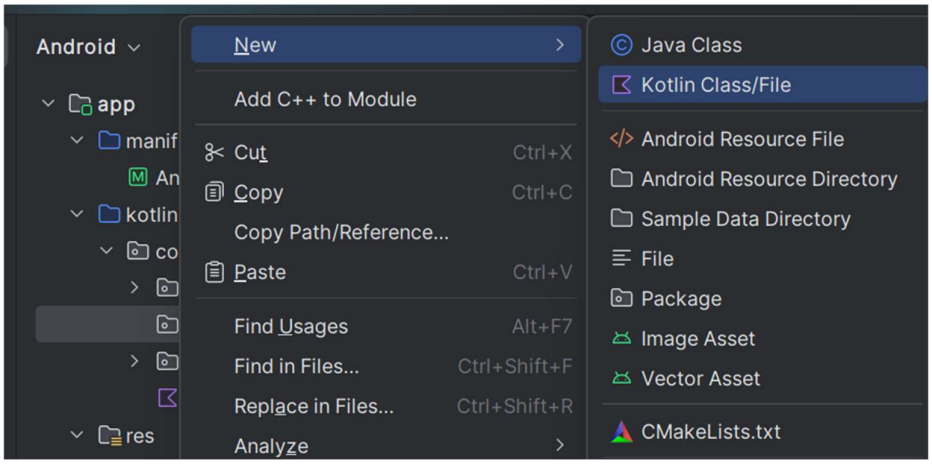
1. Buat paket baru di bagian com.example.affirmations.



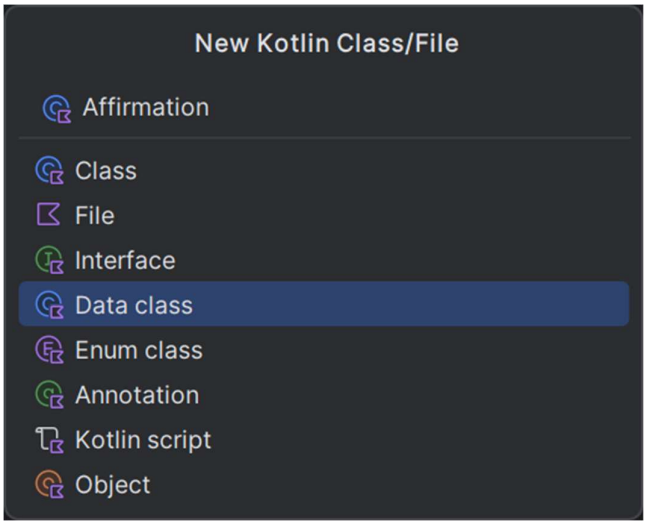
Beri nama paket baru tersebut dengan **model.** Paket model akan berisi model data yang akan direpresentasikan oleh class data. Class data akan terdiri dari properti yang mewakili informasi yang relevan dengan yang akan disebut "Affirmation", yang akan terdiri dari resource string dan resource gambar. Paket adalah direktori yang berisi beberapa class dan bahkan direktori lainnya.



2. Buat class baru di paket com.example.affirmations.model.



Beri nama class baru tersebut dengan Affirmation dan jadikan Data class.



3. Setiap Affirmation terdiri dari satu gambar dan satu string. Buat dua properti val di class data Affirmation. Salah satunya harus disebut stringResourceId dan yang lainnya imageResourceId. Keduanya harus berupa bilangan bulat.

Affirmation.kt

```
data class Affirmation(
    val stringResourceId: Int,
    val imageResourceId: Int
)
```

4. Anotasikan properti stringResourceId dengan anotasi @StringRes dan anotasikan imageResourceId dengan anotasi @DrawableRes. stringResourceId mewakili ID untuk teks afirmasi yang disimpan di resource string. imageResourceId mewakili ID untuk gambar afirmasi yang disimpan di resource drawable.

Affirmation.kt

```
import androidx.annotation.DrawableRes
import androidx.annotation.StringRes

data class Affirmation(
    @StringRes val stringResourceId: Int,
    @DrawableRes val imageResourceId: Int
)
```

5. Dalam paket **com.example.affirmations.data**, buka file **Datasource.kt** dan hapus tSaya komentar pada dua pernyataan impor serta konten class Datasource.

Datasource.kt

```
import com.example.affirmations.R
import com.example.affirmations.model.Affirmation

class Datasource() {
    fun loadAffirmations(): List<Affirmation> {
```

```
        return listOf<Affirmation>(
            Affirmation(R.string.affirmation1, R.drawable.image1),
            Affirmation(R.string.affirmation2, R.drawable.image2),
            Affirmation(R.string.affirmation3, R.drawable.image3),
            Affirmation(R.string.affirmation4, R.drawable.image4),
            Affirmation(R.string.affirmation5, R.drawable.image5),
            Affirmation(R.string.affirmation6, R.drawable.image6),
            Affirmation(R.string.affirmation7, R.drawable.image7),
            Affirmation(R.string.affirmation8, R.drawable.image8),
            Affirmation(R.string.affirmation9, R.drawable.image9),
            Affirmation(R.string.affirmation10, R.drawable.image10))
    }
}
```

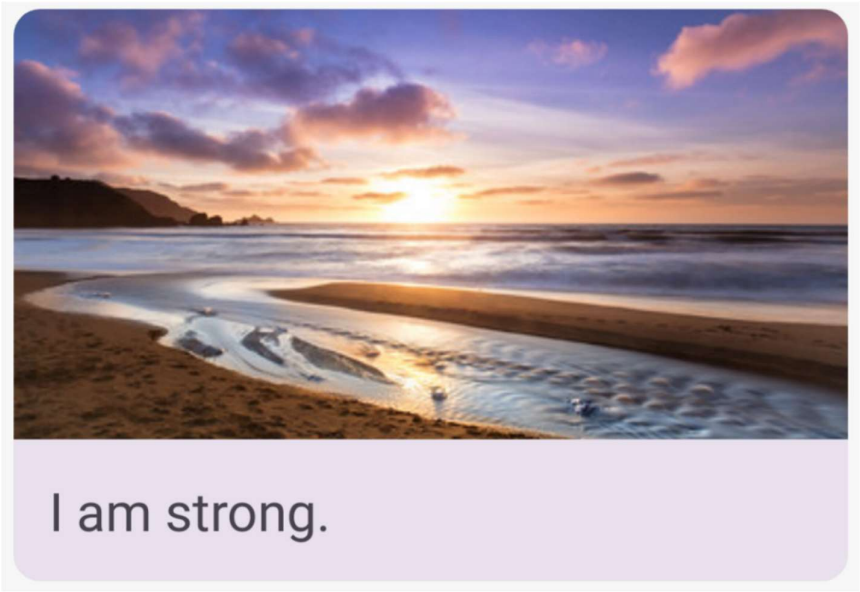
Catatan: Metode loadAffirmations() mengumpulkan semua data yang diberikan dalam kode awal dan menampilkannya sebagai daftar. Saya akan menggunakannya nanti untuk membuat daftar yang dapat di-scroll.

3. Menambahkan daftar ke aplikasi

Membuat kartu item daftar

Aplikasi ini dimaksudkan untuk menampilkan daftar afirmasi. Langkah pertama dalam mengonfigurasi UI untuk menampilkan daftar adalah membuat item daftar. Setiap item afirmasi terdiri dari gambar dan string. Data untuk setiap item ini dilengkapi dengan kode awal, dan Saya akan membuat komponen UI untuk menampilkan item tersebut.

Item akan terdiri dari composable Card, yang berisi Image dan composable Text. Di Compose, Card adalah platform yang menampilkan konten dan tindakan dalam satu penampung. Kartu Affirmation akan terlihat seperti ini di pratinjau:



Kartu ini menampilkan gambar dengan teks di bawahnya. Tata letak vertikal ini dapat dibuat menggunakan composable Column yang digabungkan dalam composable Card. Saya dapat mencobanya sendiri, atau ikuti langkah-langkah di bawah untuk melakukannya.

1. Buka file MainActivity.kt.
2. Buat metode baru di bawah metode AffirmationsApp(), yang disebut AffirmationCard(), dan anotasikan dengan anotasi @Composable.

MainActivity.kt

```
@Composable
fun AffirmationsApp() {
}

@Composable
fun AffirmationCard() {

}
```

3. Edit tSaya tangan metode untuk mengambil objek Affirmation sebagai parameter. Objek Affirmation berasal dari paket model.

MainActivity.kt

```
import com.example.affirmations.model.Affirmation

@Composable
fun AffirmationCard(affirmation: Affirmation) {

}
```

4. Tambahkan parameter modifier ke tSaya tangan. Setel nilai default Modifier untuk parameter.

MainActivity.kt

```
@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {

}
```

Catatan: Sebaiknya teruskan pengubah ke setiap composable dan setel ke nilai default.

5. Di dalam metode AffirmationCard, panggil composable Card. Teruskan parameter modifier.

MainActivity.kt

```
import androidx.compose.material3.Card

@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {
```

```
    Card(modifier = modifier) {

    }
}
```

6. Tambahkan composable Column di dalam composable Card. Item dalam composable Column menyusun dirinya sendiri secara vertikal di UI. Hal ini memungkinkan Saya menempatkan gambar di atas teks terkait. Sebaliknya, composable Row mengatur item yang ditampung secara horizontal.

MainActivity.kt

```
import androidx.compose.foundation.layout.Column

@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {
    Card(modifier = modifier) {
        Column {

        }
    }

}
```

7. Tambahkan composable Image di dalam isi lambda dari composable Column. Ingat kembali bahwa composable Image selalu memerlukan resource untuk ditampilkan, dan contentDescription. Resource ini harus berupa painterResource yang diteruskan ke parameter painter. Metode painterResource akan memuat vektor drawable atau format aset raster seperti PNG. Selain itu, teruskan stringResource untuk parameter contentDescription.

MainActivity.kt

```
import androidx.compose.foundation.Image
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource

@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {
    Card(modifier = modifier) {
        Column {
            Image(
                painter = painterResource(affirmation.imageResourceId),
                contentDescription =
stringResource(affirmation.stringResourceId),
            )
        }
    }
}
```

8. Selain parameter painter dan contentDescription, teruskan modifier dan contentScale. contentScale menentukan cara gambar harus diskalakan dan ditampilkan. Objek Modifier harus memiliki atribut fillMaxWidth yang disetel dan tinggi 194.dp. contentScale harus ContentScale.Crop.

MainActivity.kt

```
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.ui.unit.dp
import androidx.compose.ui.layout.ContentScale

@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {
    Card(modifier = modifier) {
        Column {
            Image(
                painter = painterResource(affirmation.imageResourceId),
                contentDescription =
stringResource(affirmation.stringResourceId),
                modifier = Modifier
                    .fillMaxWidth()
                    .height(194.dp),
                contentScale = ContentScale.Crop
            )
        }
    }
}
```

9. Di dalam Column, buat composable Text setelah composable Image. Teruskan stringResource dari affirmation.stringResourceId ke parameter text, teruskan objek Modifier dengan atribut padding yang disetel ke 16.dp, dan setel tema teks dengan meneruskan MaterialTheme.typography.headlineSmall ke parameter style.

MainActivity.kt

```
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.ui.unit.dp
import androidx.compose.ui.layout.ContentScale

@Composable
fun AffirmationCard(affirmation: Affirmation, modifier: Modifier =
Modifier) {
    Card(modifier = modifier) {
        Column {
            Image(
                painter = painterResource(affirmation.imageResourceId),
                contentDescription =
stringResource(affirmation.stringResourceId),
                modifier = Modifier
```

```
                    .fillMaxWidth()
                    .height(194.dp),
                contentScale = ContentScale.Crop
            )
        }
    }
}
```

Pratinjau composable AffirmationCard

Kartu ini adalah inti dari UI untuk aplikasi **Affirmations**, dan Saya telah bekerja keras untuk membuatnya. Untuk memeriksa apakah kartu sudah benar, Saya dapat membuat composable yang dapat dilihat pratinjaunya tanpa meluncurkan seluruh aplikasi.

1. Buat metode pribadi bernama AffirmationCardPreview(). Anotasikan metode dengan @Preview dan @Composable.

MainActivity.kt

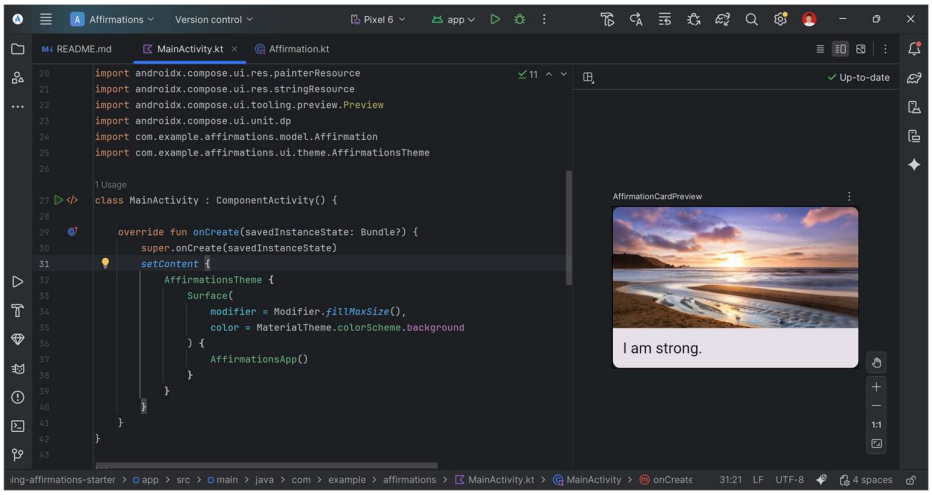
```
import androidx.compose.ui.tooling.preview.Preview

@Preview
@Composable
private fun AffirmationCardPreview() {

}
```

2. Di dalam metode, panggil composable AffirmationCard, dan teruskan objek Affirmation baru dengan resource string R.string.affirmation1 dan resource drawable R.drawable.image1 yang diteruskan ke konstruktornya.

MainActivity.kt

```
@Preview
@Composable
private fun AffirmationCardPreview() {
    AffirmationCard(Affirmation(R.string.affirmation1,
R.drawable.image1))
}
```

3. Buka tab **Split** dan Saya akan melihat pratinjau AffirmationCard. Jika perlu, klik **Build & Refresh** di panel **Design** untuk menampilkan pratinjau.



Membuat daftar

Komponen item daftar adalah elemen penyusun daftar. Setelah item daftar dibuat, Saya dapat memanfaatkannya untuk membuat komponen daftar itu sendiri.

1. Buat fungsi yang disebut AffirmationList(), anotasikan dengan anotasi @Composable, dan deklarasikan List objek Affirmation sebagai parameter di tSaya tangan metode.

MainActivity.kt

```
@Composable
fun AffirmationList(affirmationList: List<Affirmation>) {

}
```

2. Deklarasikan objek modifier sebagai parameter dalam tSaya tangan metode dengan nilai default Modifier.

MainActivity.kt

```
@Composable
fun AffirmationList(affirmationList: List<Affirmation>, modifier:
Modifier = Modifier) {

}
```

3. Di Jetpack Compose, daftar yang dapat di-scroll dapat dibuat menggunakan composable LazyColumn. Perbedaan antara LazyColumn dan Column adalah bahwa Column harus digunakan saat Saya memiliki sedikit item untuk ditampilkan, karena Compose memuat semuanya sekaligus. Column hanya dapat menyimpan composable dengan jumlah yang tetap atau telah ditentukan. LazyColumn dapat menambahkan konten sesuai keperluan, yang menjadikannya cocok untuk daftar panjang, terutama jika panjang daftar tidak diketahui. LazyColumn juga menyediakan scroll secara default, tanpa kode tambahan. Deklarasikan composable LazyColumn di

dalam fungsi AffirmationList(). Teruskan objek modifier sebagai argumen ke LazyColumn.

MainActivity.kt

```
import androidx.compose.foundation.lazy.LazyColumn

@Composable
fun AffirmationList(affirmationList: List<Affirmation>, modifier:
Modifier = Modifier) {
    LazyColumn(modifier = modifier) {

    }
}
```

4. Dalam isi lambda LazyColumn, panggil metode items() dan teruskan affirmationList. Metode items() adalah cara Saya menambahkan item ke LazyColumn. Metode ini agak unik untuk composable ini, dan bukan praktik umum untuk sebagian besar composable.

MainActivity.kt

```
import androidx.compose.foundation.lazy.items

@Composable
fun AffirmationList(affirmationList: List<Affirmation>, modifier:
Modifier = Modifier) {
    LazyColumn(modifier = modifier) {
        items(affirmationList) {

        }
    }
}
```

5. Panggilan ke metode items() memerlukan fungsi lambda. Dalam fungsi tersebut, tetapkan parameter affirmation yang mewakili satu item afirmasi dari affirmationList.

MainActivity.kt

```
@Composable
fun AffirmationList(affirmationList: List<Affirmation>, modifier:
Modifier = Modifier) {
    LazyColumn(modifier = modifier) {
        items(affirmationList) { affirmation ->

        }
    }
}
```

6. Untuk setiap afirmasi dalam daftar, panggil composable AffirmationCard(). Teruskan affirmation dan objek Modifier dengan atribut padding yang disetel ke 8.dp.

MainActivity.kt

```
@Composable
fun AffirmationList(affirmationList: List<Affirmation>, modifier:
Modifier = Modifier) {
```

```
    LazyColumn(modifier = modifier) {
        items(affirmationList) { affirmation ->
            AffirmationCard(
                affirmation = affirmation,
                modifier = Modifier.padding(8.dp)
            )
        }
    }
}
```

Menampilkan daftar

1. Pada composable AffirmationsApp, ambil rute tata letak saat ini lalu simpan dalam variabel. Rute ini akan digunakan untuk mengonfigurasi padding nanti.

MainActivity.kt

```
import com.example.affirmations.data.Datasource

@Composable
fun AffirmationsApp() {
    val layoutDirection = LocalLayoutDirection.current
}
```

2. Sekarang, buat composable Surface. Composable ini akan menetapkan padding untuk composable AffirmationsList.

MainActivity.kt

```
import com.example.affirmations.data.Datasource

@Composable
fun AffirmationsApp() {
    val layoutDirection = LocalLayoutDirection.current
    Surface() {
    }
}
```

3. Teruskan Modifier ke composable Surface yang mengisi lebar dan tinggi maksimum induknya, menetapkan padding status bar, serta menyetel padding awal dan akhir ke layoutDirection. Berikut contoh cara menerjemahkan objek LayoutDirection menjadi padding: WindowInsets.safeDrawing.asPaddingValues().calculateStartPadding(layoutDirection).

MainActivity.kt

```
import com.example.affirmations.data.Datasource

@Composable
fun AffirmationsApp() {
    val layoutDirection = LocalLayoutDirection.current
    Surface(
        Modifier = Modifier
        .fillMaxSize()
        .statusBarsPadding()
```

```
        .padding(
            start = WindowInsets.safeDrawing.asPaddingValues()
                    .calculateStartPadding(layoutDirection),
            end = WindowInsets.safeDrawing.asPaddingValues()
                    .calculateEndPadding(layoutDirection),
        ),
    ) {
    }
}
```

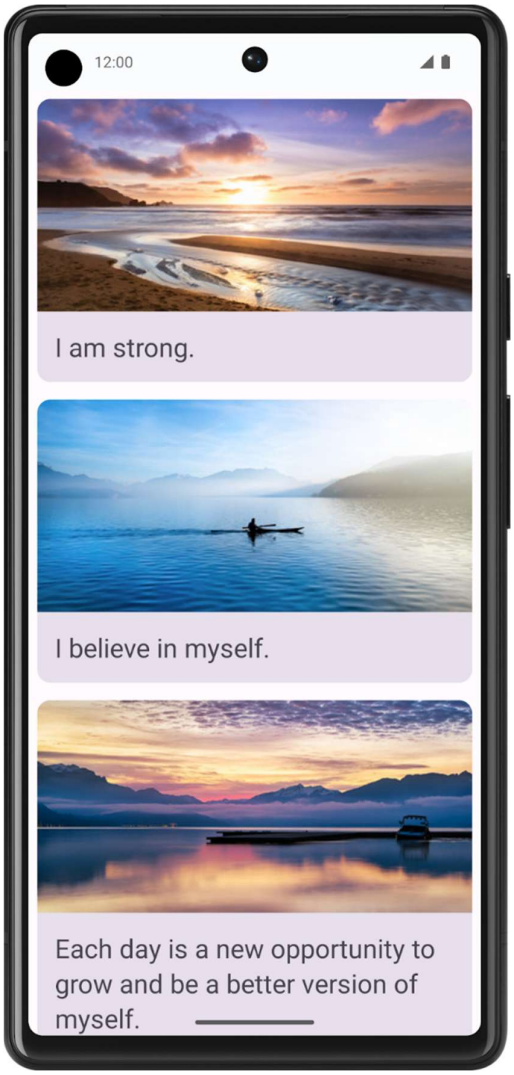
4. Di lambda untuk composable Surface, panggil composable AffirmationList, lalu teruskan DataSource().loadAffirmations() ke parameter affirmationList.

Catatan: Class DataSource ditemukan dalam paket data.

MainActivity.kt

```
import com.example.affirmations.data.Datasource

@Composable
fun AffirmationsApp() {
    val layoutDirection = LocalLayoutDirection.current
    Surface(
        Modifier = Modifier
        .fillMaxSize()
        .statusBarsPadding()
        .padding(
            start = WindowInsets.safeDrawing.asPaddingValues()
                    .calculateStartPadding(layoutDirection),
            end = WindowInsets.safeDrawing.asPaddingValues()
                    .calculateEndPadding(layoutDirection),
        ),
    ) {
        AffirmationsList(
            affirmationList = Datasource().loadAffirmations(),
        )
    }
}
```

Jalankan aplikasi Affirmations di perangkat atau emulator dan lihat produk yang sudah selesai.



4. Mendapatkan kode solusi

Untuk mendownload kode codelab yang sudah selesai, Saya dapat menggunakan perintah git berikut:

```
$ git clone https://github.com/google-developer-training/basic-android-kotlin-compose-training-affirmations.git
$ cd basic-android-kotlin-compose-training-affirmations
$ git checkout intermediate
```

Atau, Saya dapat mendownload repositori sebagai file ZIP, lalu mengekstraknya, dan membukanya di Android Studio.

file_downloadDownload zip

Catatan: Kode solusi ada di cabang intermediate dari repositori yang didownload.

Jika Saya ingin melihat kode solusi, lihat di GitHub.

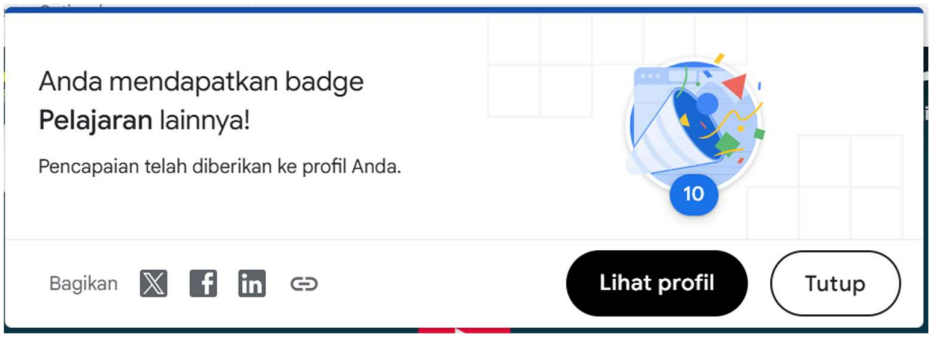
5. Kesimpulan

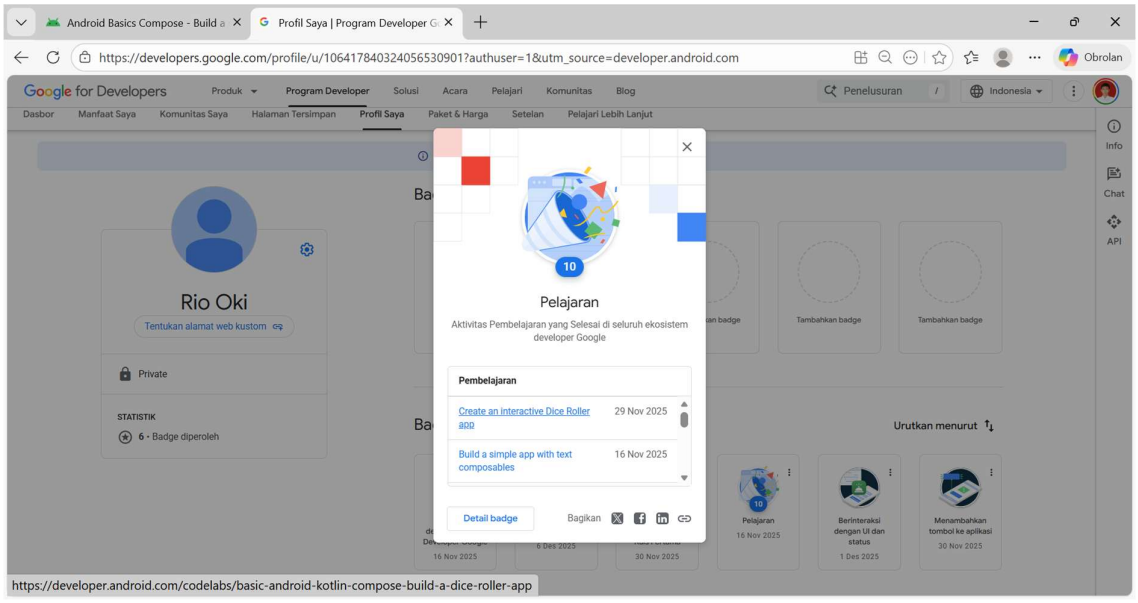
Saya sekarang tahu cara membuat kartu, item daftar, dan daftar yang dapat di-scroll menggunakan Jetpack Compose. Ingatlah bahwa ini hanyalah alat dasar untuk membuat daftar. Saya dapat menyalurkan kreativitas dan menyesuaikan item daftar sesuka hati.

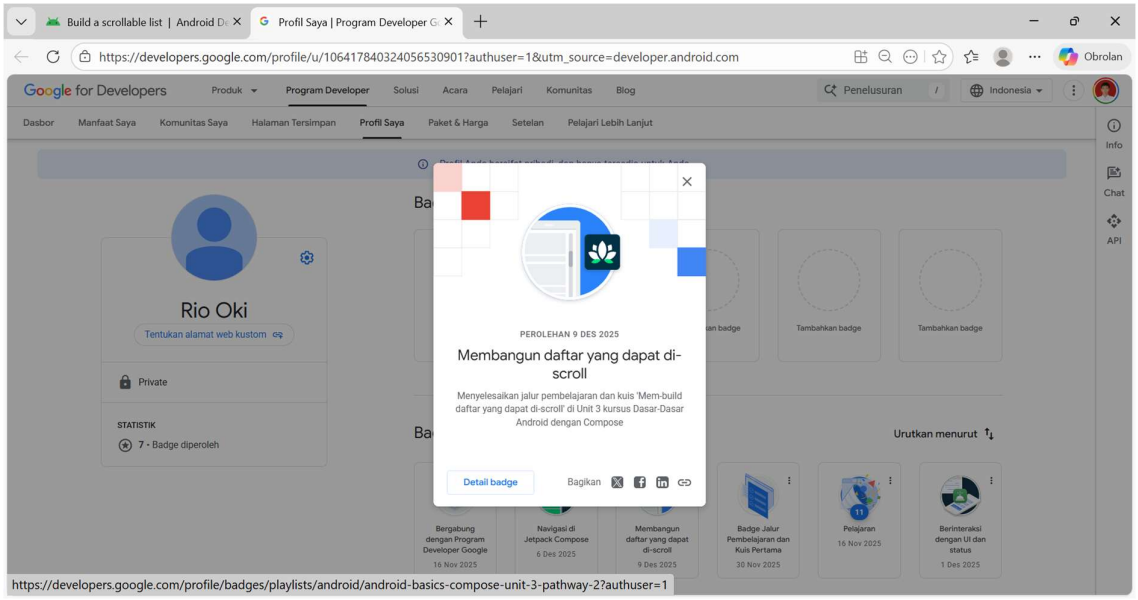
Ringkasan

- Gunakan composable Card untuk membuat item daftar.
- Ubah UI yang ada dalam composable Card.
- Buat daftar yang dapat di-scroll menggunakan composable LazyColumn.
- Buat daftar menggunakan item daftar kustom.

6. Learning Badge







Latihan:Membuat Petak

1. Sebelum Mulai

Selamat! Saya telah membangun aplikasi pertama dengan daftar yang dapat di-scroll. Sekarang Saya siap untuk mempraktikkan semua yang telah Saya pelajari.

Latihan ini berfokus pada pembuatan komponen yang diperlukan untuk membangun daftar yang dapat di-scroll. Materi ini dikembangkan dari materi yang Saya pelajari di codelab Menambahkan daftar yang dapat di-scroll, dan memungkinkan Saya menerapkan pengetahuan tersebut untuk membuat petak yang dapat di-scroll.

Beberapa bagian mungkin mengharuskan Saya menggunakan composable atau pengubah, yang mungkin belum pernah Saya lihat sebelumnya. Jika demikian, lihat Referensi yang tersedia untuk setiap soal. Saya dapat menemukan link ke dokumentasi yang terkait dengan pengubah, properti, atau composable yang tidak Saya ketahui. Saya dapat membaca dokumentasi dan mempelajari cara menerapkan konsep-konsep yang ada ke dalam aplikasi. Kemampuan untuk memahami dokumentasi adalah keterampilan penting yang harus Saya kembangkan untuk meningkatkan pengetahuan.

Kode solusi tersedia di bagian akhir, tetapi cobalah untuk menyelesaikan latihan sebelum Saya memeriksa jawabannya. Pertimbangkan solusi tersebut sebagai salah satu cara untuk menerapkan aplikasi.

Prasyarat

- Selesaikan kursus Dasar-Dasar Android di Compose melalui codelab Menambahkan daftar yang dapat di-scroll.

Yang akan Saya butuhkan

- Komputer yang dilengkapi akses internet dan Android Studio.

Aset

Saya akan memerlukan resource berikut guna menyelesaikan kode untuk soal latihan ini

- Gambar topik. Gambar ini mewakili setiap topik dalam daftar.
- ic_grain.xml. Ini adalah ikon dekoratif yang muncul di samping jumlah kursus dalam topik.

Yang akan Saya bangun

Dalam soal latihan ini, Saya akan membangun aplikasi Kursus dari awal. Aplikasi Kursus akan menampilkan daftar topik kursus.

Soal latihan dibagi menjadi beberapa bagian, yang meminta Saya membangun:

- Class data topik kursus:

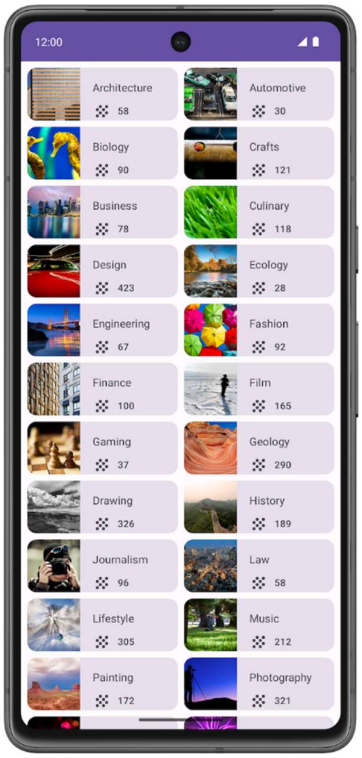
Data topik akan memiliki gambar, nama, dan jumlah kursus terkait dalam topik tersebut.

- Composable untuk mewakili item petak topik kursus:

Setiap item topik akan menampilkan gambar, nama, jumlah kursus terkait, dan ikon dekoratif.

- Composable untuk menampilkan petak item topik kursus.

Aplikasi final akan terlihat seperti ini:



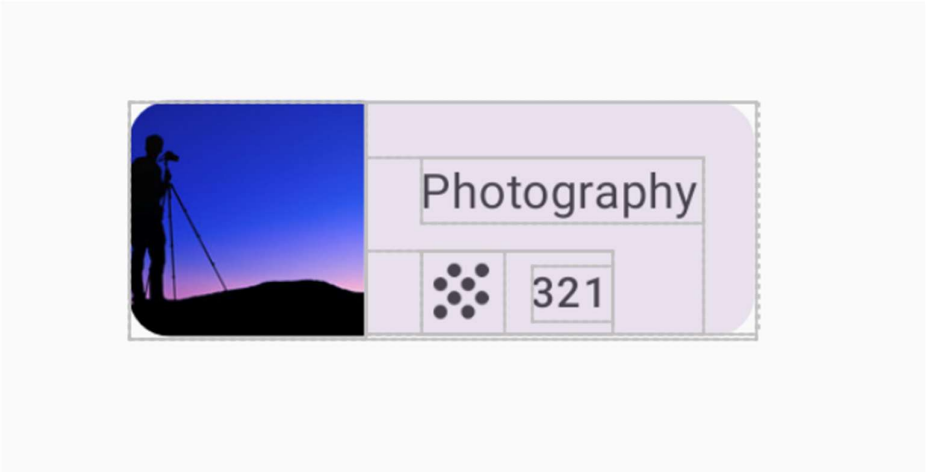
2. Memulai

Buat Project Baru dengan template Empty Activity dan SDK minimum 24.

3. Class data topik

Di bagian ini, Saya akan membangun class untuk menyimpan data setiap topik kursus.

Lihat item dari aplikasi akhir.



Setiap topik kursus memiliki tiga informasi unik. Dengan menggunakan konten unik setiap item sebagai referensi, buat class untuk menyimpan data ini.

4. Sumber data

Di bagian ini, Saya akan membuat set data untuk petak kursus.

Salin item berikut ke app/src/main/res/values/strings.xml:

```
<string name="architecture">Architecture</string>
<string name="crafts">Crafts</string>
<string name="business">Business</string>
<string name="culinary">Culinary</string>
<string name="design">Design</string>
<string name="fashion">Fashion</string>
<string name="film">Film</string>
<string name="gaming">Gaming</string>
<string name="drawing">Drawing</string>
<string name="lifestyle">Lifestyle</string>
<string name="music">Music</string>
<string name="painting">Painting</string>
<string name="photography">Photography</string>
<string name="tech">Tech</string>
```

Buat file kosong bernama DataSource.kt. Salin kode berikut ke dalam file:

```
object DataSource {
    val topics = listOf(
```

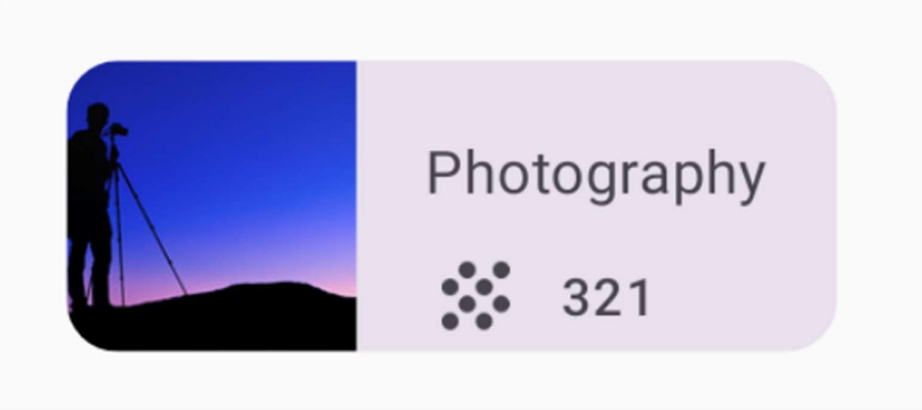
```
        Topic(R.string.architecture, 58, R.drawable.architecture),
        Topic(R.string.crafts, 121, R.drawable.crafts),
        Topic(R.string.business, 78, R.drawable.business),
        Topic(R.string.culinary, 118, R.drawable.culinary),
        Topic(R.string.design, 423, R.drawable.design),
        Topic(R.string.fashion, 92, R.drawable.fashion),
        Topic(R.string.film, 165, R.drawable.film),
        Topic(R.string.gaming, 164, R.drawable.gaming),
        Topic(R.string.drawing, 326, R.drawable.drawing),
        Topic(R.string.lifestyle, 305, R.drawable.lifestyle),
        Topic(R.string.music, 212, R.drawable.music),
        Topic(R.string.painting, 172, R.drawable.painting),
        Topic(R.string.photography, 321, R.drawable.photography),
        Topic(R.string.tech, 118, R.drawable.tech)
    )
}
```

5. Item petak topik

Buat composable untuk mewakili item petak topik.

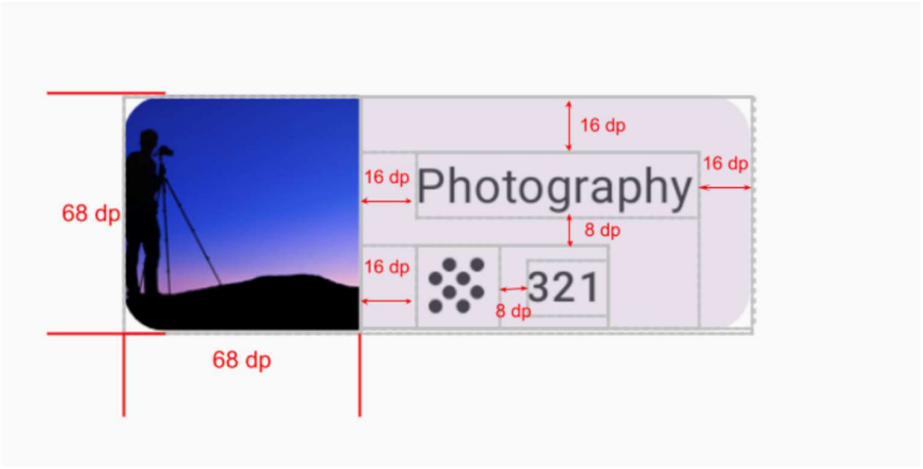
Screenshot final

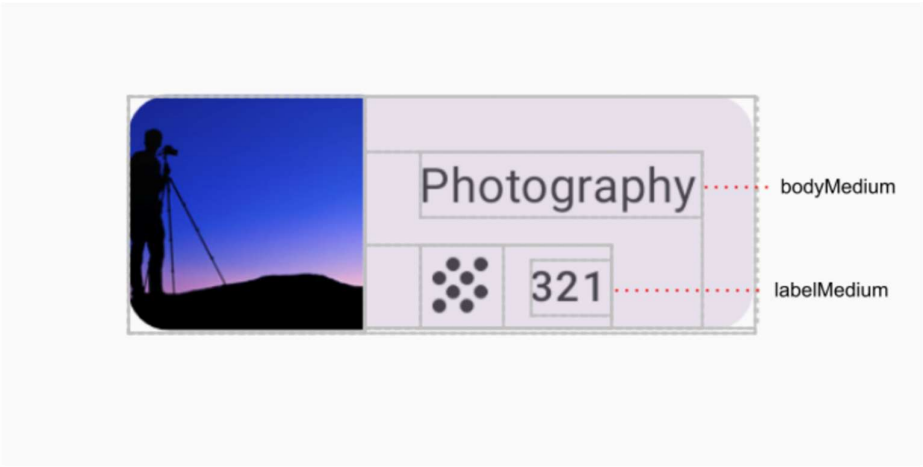
Setelah menyelesaikan implementasi, tata letak item topik Saya akan seperti screenshot di bawah:



Spesifikasi UI

Gunakan spesifikasi UI berikut:





Gaya Teks

Petunjuk: Composable mana yang mengatur turunannya secara vertikal dan mana yang mengatur turunannya secara horizontal?

Referensi

- Tipografi
- Komponen tata letak stSayar
- Tata letak Box
- Tata letak Column
- Tata letak Row
- aspectRatio
- painterResource

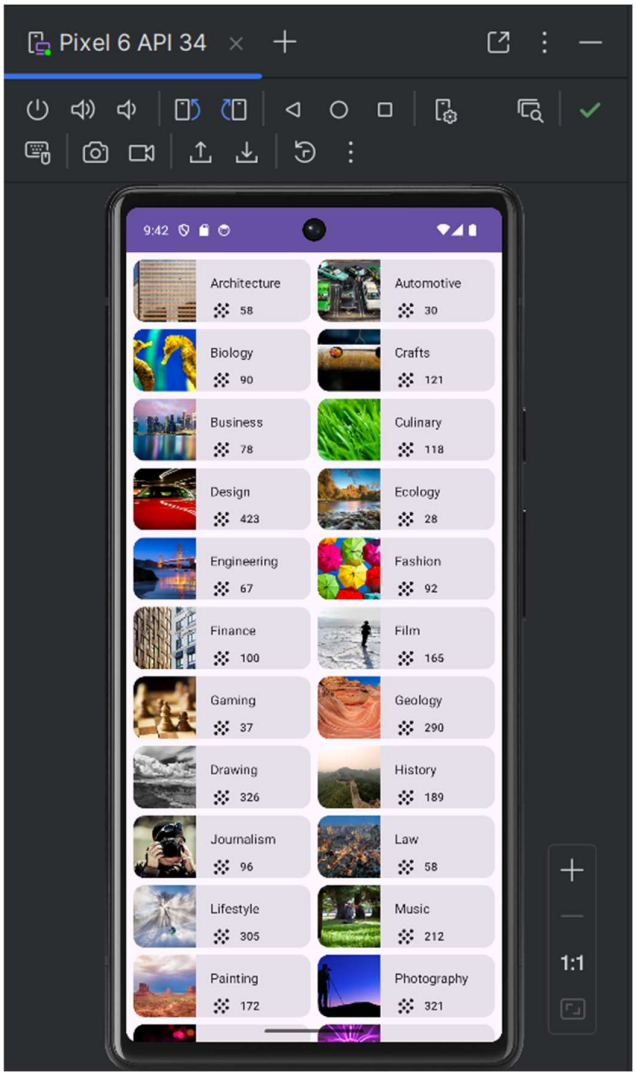
6. Petak kursus

Setelah dibuat, item petak topik dapat digunakan untuk membuat petak topik kursus.

Dalam latihan ini, Saya menggunakan composable item petak untuk membuat petak dengan dua kolom.

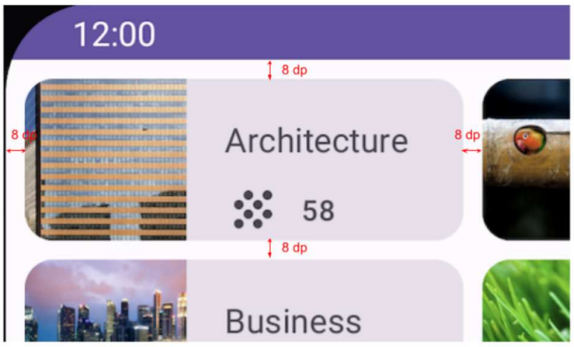
Screenshot final

Setelah menyelesaikan implementasi, desain Saya akan seperti screenshot di bawah:



Spesifikasi UI

Gunakan spesifikasi UI berikut:



Referensi

- Daftar dan petak
- Sel petak tetap
- Daftar: Jarak konten
- Codelab: Menambahkan daftar yang dapat di-scroll

7. Mendapatkan kode solusi

Guna mendownload kode untuk codelab yang sudah selesai, Saya dapat menggunakan perintah git ini:

```
$ git clone https://github.com/google-developer-training/basic-android-kotlin-compose-training-courses.git
```

Atau, Saya dapat mendownload repositori sebagai file ZIP, lalu mengekstraknya, dan membukanya di Android Studio.

file_downloadDownload zip

Jika Saya ingin melihat kode solusi, lihat di GitHub.

Aplikasi yang Saya bangun dalam latihan ini adalah versi modifikasi dari layar kursus aplikasi Owl. Aplikasi Owl adalah aplikasi contoh komprehensif yang menunjukkan kemampuan Compose. Aplikasi contoh Compose lainnya dapat ditemukan di repositori GitHub compose-samples.

8. Learning Badge