Rapport Mobile : ACHIR CHAIMAA

EXERCICE1 : Application Machine Learning

1. Introduction

Ce rapport détaille l'implémentation d'une application Android qui affiche une interface utilisateur sur le thème du Machine Learning. L'application a été développée en utilisant Kotlin et Jetpack Compose, respectant scrupuleusement les spécifications fournies dans l'exercice.

2. Objectifs de l'exercice

Les objectifs principaux de cet exercice étaient :

- Créer une interface utilisateur avec Jetpack Compose
- Afficher une image en en-tête qui remplit toute la largeur de l'écran
- Configurer trois composables Text avec des spécifications précises de taille et de padding
- Appliquer un alignement justifié au texte pour une meilleure lisibilité
- Respecter les dimensions spécifiées (24sp, 16dp) pour la cohérence visuelle

3. Technologies utilisées

3.1 Kotlin

Kotlin est le langage de programmation officiel recommandé par Google pour le développement Android. Il offre une syntaxe concise, la sécurité des types null et une interopérabilité totale avec Java.

3.2 Jetpack Compose

Jetpack Compose est le toolkit moderne de Google pour construire des interfaces utilisateur natives Android. Il utilise une approche déclarative qui simplifie et accélère le développement d'interfaces utilisateur. Compose permet de créer des UI réactives avec moins de code et de manière plus intuitive.

3.3 Material Design 3

Material Design 3 est le système de design de Google qui fournit des composants et des styles cohérents pour créer des applications modernes et attrayantes.

4.2 Fichiers principaux

MainActivity.kt : Activité principale contenant l'interface utilisateur Compose

build.gradle : Configuration Gradle avec les dépendances Compose

AndroidManifest.xml : Manifeste de l'application avec la configuration

strings.xml : Ressources de chaînes de caractères

themes.xml : Thème de l'application

ml_header.png : Image d'en-tête pour l'interface

5. Implémentation détaillée

5.1 MainActivity.kt

Le fichier MainActivity.kt contient toute la logique de l'interface utilisateur. Voici les composants principaux :

5.1.1 Classe MainActivity

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MachineLearningTheme {
                Surface(
```

```
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MachineLearningScreen()
                }
            }
        }
    }
}
```

Cette classe hérite de ComponentActivity et initialise l'interface Compose dans la méthode onCreate().

5.1.2 Fonction MachineLearningScreen()

Cette fonction composable crée l'interface principale de l'application. Elle est divisée en plusieurs parties selon les spécifications :

a) Image en en-tête (Spécification 1)

```
Image(
    painter = painterResource(id = R.drawable.ml_header),
    contentDescription = "Machine Learning Header",
    modifier = Modifier.fillMaxWidth(),
    contentScale = ContentScale.FillWidth
)
```

Explications :

- fillMaxWidth() : Force l'image à occuper toute la largeur de l'écran
- ContentScale.FillWidth : Maintient le ratio d'aspect de l'image tout en remplissant la largeur
- painterResource() : Charge l'image depuis les ressources drawable

b) Premier Text - Titre (Spécification 2)

```
Text(
    text = "Machine Learning",
    fontSize = 24.sp,
    modifier = Modifier.padding(
        start = 16.dp,
        end = 16.dp,
        top = 16.dp,
        bottom = 16.dp
    )
)
```

Explications :

- fontSize = 24.sp : Taille de police de 24 scaled pixels comme spécifié
- padding(16.dp sur tous les côtés) : Espacement de 16 density-independent pixels
- start, end, top, bottom : Tous définis à 16.dp conformément aux spécifications

c) Deuxième Text - Définition (Spécification 3)

```
Text(
    text = "The use and development of computer systems...",
    textAlign = TextAlign.Justify,
    modifier = Modifier.padding(start = 16.dp, end = 16.dp)
)
```

Explications :

- fontSize : Utilise la taille par défaut (non spécifiée)
- textAlign = TextAlign.Justify : Justifie le texte sur toute la largeur
- padding(start = 16.dp, end = 16.dp) : Uniquement start et end comme spécifié

d) Troisième Text - Explication détaillée (Spécification 4)

```
Text(
    text = "A core objective of a learner is to generalise...",
    textAlign = TextAlign.Justify,
    modifier = Modifier.padding(
        start = 16.dp,
        end = 16.dp,
        top = 16.dp,
        bottom = 16.dp
    )
)
```

Explications :

- fontSize : Utilise la taille par défaut (non spécifiée)
- textAlign = TextAlign.Justify : Justifie le texte sur toute la largeur
- padding(16.dp sur tous les côtés) : Tous les côtés définis à 16.dp

6. Configuration Gradle

6.1 build.gradle (app)

Le fichier build.gradle de l'application contient :

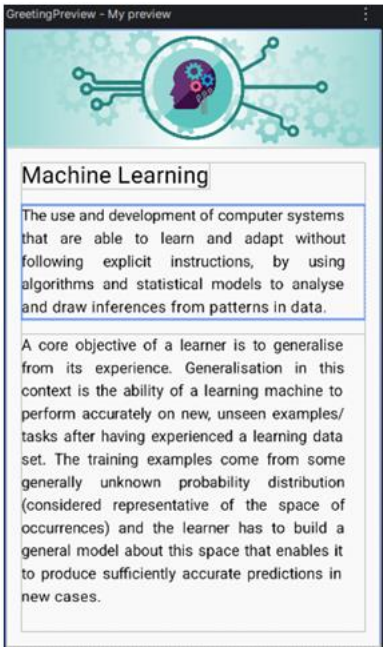
- Configuration du namespace et des versions SDK (compileSdk 34, minSdk 24)
- Activation de Jetpack Compose avec buildFeatures { compose true }
- Version du compilateur Compose : 1.5.1
- Dépendances Compose BOM (Bill of Materials) pour la gestion des versions

- Dépendances Material3 pour les composants UI modernes

6.2 Dépendances principales

- androidx.compose.ui:ui - Composants UI de base
- androidx.compose.material3:material3 - Composants Material Design 3
- androidx.activity:activity-compose - Support des activités Compose
- androidx.compose.ui:ui-tooling-preview - Prévisualisations dans Android Studio

9. Tests et validation



EXERCICE 2 : Écran de Complétion des Tâches

1. Description

L'exercice 2 consiste à créer un écran de félicitations qui s'affiche lorsque l'utilisateur termine toutes ses tâches. Cet écran doit présenter un message de réussite avec une icône visuelle centrée sur l'écran.

2. Objectifs

Les spécifications de cet exercice sont :

- Centrer tout le contenu verticalement et horizontalement sur l'écran
- Premier Text composable : police Bold, 24dp padding top, 8dp padding bottom
- Deuxième Text composable : taille de police de 16sp
- Afficher une icône de succès (cercle vert avec coche)
- Créer une expérience utilisateur positive et encourageante

3. Implémentation

3.1 CompletionActivity.kt

Une nouvelle activité a été créée spécifiquement pour cet écran de complétion. Cette séparation permet une meilleure organisation du code et facilite la navigation entre les différents écrans de l'application.

3.2 Fonction CompletionScreen()

Cette fonction composable construit l'interface de l'écran de complétion selon les spécifications détaillées ci-dessous :

a) Centrage du contenu (Spécification 1)

```
Column(
    modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center
) {
    // Contenu centré ici
}
```

Explications :

- fillMaxSize() : La colonne occupe tout l'espace disponible de l'écran
- horizontalAlignment = Alignment.CenterHorizontally : Centre le contenu horizontalement
- verticalArrangement = Arrangement.Center : Centre le contenu verticalement
- Ces trois propriétés combinées créent un centrage parfait sur l'écran

b) Premier Text - "All tasks completed" (Spécification 2)

```
Text(
    text = "All tasks completed",
    fontWeight = FontWeight.Bold,
    modifier = Modifier.padding(top = 24.dp, bottom = 8.dp)
)
```

Explications :

- fontWeight = FontWeight.Bold : Applique un style gras au texte
- padding(top = 24.dp) : Espacement de 24dp au-dessus du texte
- padding(bottom = 8.dp) : Espacement de 8dp en dessous du texte
- Le texte utilise la taille par défaut du thème

c) Deuxième Text - "Nice work!" (Spécification 3)

```
Text(
    text = "Nice work!",
    fontSize = 16.sp
)
```

Explications :

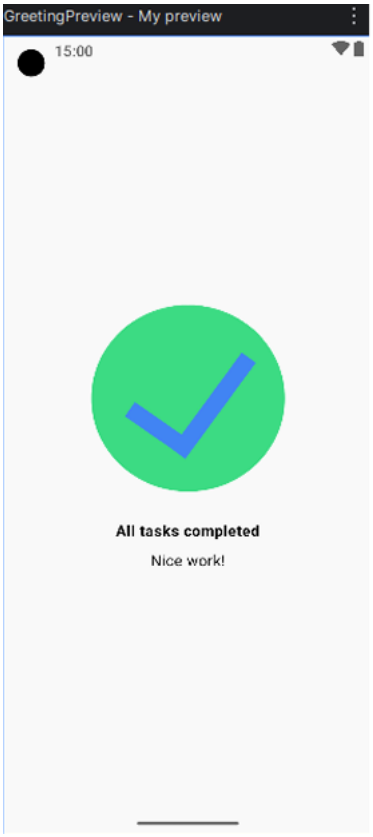
- fontSize = 16.sp : Taille de police définie à 16 scaled pixels
- Aucun padding spécifié, utilise les espacements par défaut
- Message encourageant pour féliciter l'utilisateur

7. Code source complet de CompletionActivity.kt

```
package com.example.machinelearning

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
```

```
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class CompletionActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MachineLearningTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    CompletionScreen()
                }
            }
        }
    }
}

@Composable
fun CompletionScreen() {
    // 1. Centrer tout le contenu verticalement et horizontalement
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        // Image de succès
        Image(
            painter = painterResource(id = R.drawable.ic_success),
            contentDescription = "Success Icon",
            modifier = Modifier.size(200.dp)
        )

        // 2. Premier Text - Bold, 24dp padding top, 8dp padding bottom
        Text(
            text = "All tasks completed",
            fontWeight = FontWeight.Bold,
            modifier = Modifier.padding(top = 24.dp, bottom = 8.dp)
        )

        // 3. Deuxième Text - 16sp font size
        Text(
            text = "Nice work!",
            fontSize = 16.sp
        )
    }
}

@Preview(showBackground = true)
@Composable
fun CompletionPreview() {
    MachineLearningTheme {
        CompletionScreen()
    }
}
```

8. Tests et validation



EXERCICE 3 : Écran avec Quadrants Composables

1. Description

L'exercice 3 consiste à créer un écran divisé en quatre quadrants égaux. Chaque quadrant présente des informations sur un composable différent de Jetpack Compose : Text, Image, Row, et Column. Cette interface permet d'apprendre visuellement les différents composables de base tout en créant une mise en page organisée et équilibrée.

2. Objectifs

Les spécifications de cet exercice sont :

- Diviser l'écran en quatre quadrants égaux
- Chaque quadrant contient une carte avec des informations sur un composable
- Appliquer 16dp de padding sur tous les côtés de chaque quadrant
- Centrer le contenu verticalement et horizontalement dans chaque quadrant
- Premier Text en gras avec 16dp de padding en bas
- Deuxième Text avec taille de police par défaut
- Utiliser des couleurs différentes pour distinguer les quadrants

Explications de la structure :

- Column principale avec fillMaxSize() occupe tout l'écran
- Deux Rows avec weight(1f) se partagent l'espace vertical à parts égales
- Dans chaque Row, deux ComposableInfoCard avec weight(1f) se partagent l'espace horizontal
- Résultat : quatre quadrants parfaitement égaux

4. Implémentation détaillée

4.1 Fonction QuadrantsScreen()

Cette fonction composable crée la structure principale de l'écran :

```
@Composable
fun QuadrantsScreen() {
    Column(modifier = Modifier.fillMaxSize()) {
        // Première ligne (2 quadrants en haut)
        Row(modifier = Modifier.weight(1f)) {
            ComposableInfoCard(
                title = "Text composable",
                description = "Displays text and follows the recommended...",
                backgroundColor = Color(0xFFEADDFF),
                modifier = Modifier.weight(1f)
            )
            ComposableInfoCard(
                title = "Image composable",
                description = "Creates a composable that lays out...",
                backgroundColor = Color(0xFFD0BCFF),
                modifier = Modifier.weight(1f)
            )
        }

        // Deuxième ligne (2 quadrants en bas)
        Row(modifier = Modifier.weight(1f)) {
            ComposableInfoCard(
                title = "Row composable",
                description = "A layout composable that places...",
                backgroundColor = Color(0xFFB69DF8),
                modifier = Modifier.weight(1f)
            )
            ComposableInfoCard(
                title = "Column composable",
                description = "A layout composable that places...",
                backgroundColor = Color(0xFFF6EDFF),
                modifier = Modifier.weight(1f)
            )
        }
    }
}
```

4.2 Fonction ComposableInfoCard()

Cette fonction composable réutilisable crée chaque quadrant avec toutes les spécifications :

```
@Composable
fun ComposableInfoCard(
    title: String,
    description: String,
    backgroundColor: Color,
    modifier: Modifier = Modifier
) {
    Column(
        modifier = modifier
            .fillMaxSize()
            .background(backgroundColor)
            // 1. 16dp padding sur tous les côtés
            .padding(16.dp),
        // 2. Centrer le contenu verticalement et horizontalement
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        // 3. Premier Text en gras avec 16dp padding bottom
        Text(
            text = title,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.padding(bottom = 16.dp),
            textAlign = TextAlign.Center
        )

        // 4. Deuxième Text avec taille par défaut
        Text(
            text = description,
            textAlign = TextAlign.Justify
        )
    }
}
```

7. Concepts clés de Compose utilisés

7.1 Weight Modifier

Le modifier weight() est essentiel pour créer des layouts proportionnels :

```
// La Column principale contient deux Rows avec weight(1f)
Row(modifier = Modifier.weight(1f)) { ... }  // Prend 50% de la hauteur
Row(modifier = Modifier.weight(1f)) { ... }  // Prend 50% de la hauteur

// Chaque Row contient deux cartes avec weight(1f)
ComposableInfoCard(..., modifier = Modifier.weight(1f))  // Prend 50% de la
largeur
ComposableInfoCard(..., modifier = Modifier.weight(1f))  // Prend 50% de la
largeur
```

Comment weight() fonctionne :

- weight(1f) signifie "prendre 1 part de l'espace disponible"
- Si deux éléments ont weight(1f), ils se partagent l'espace à 50/50
- weight(2f) prendrait deux fois plus d'espace que weight(1f)
- Permet de créer des layouts responsive qui s'adaptent à toutes les tailles d'écran

7.2 Background Modifier

Le modifier background() applique une couleur de fond à un composable :

```
.background(backgroundColor)
```

Points importants :

- Appliqué avant .padding() pour que le padding soit visible sur le fond coloré
- Accepte des couleurs Color() avec notation hexadécimale (0xFFRRGGBB)
- Peut aussi accepter Brush pour des dégradés
- Remplit tout l'espace du composable

7.3 Réutilisabilité avec ComposableInfoCard

La fonction ComposableInfoCard() est un excellent exemple de composable réutilisable :

- Paramètres personnalisables : title, description, backgroundColor
- Logique d'affichage encapsulée et centralisée
- Facilite la maintenance : un changement affecte tous les quadrants
- Évite la duplication de code (DRY - Don't Repeat Yourself)
- Peut être facilement testée et prévisualisée indépendamment

8. Description des composables présentés

8.1 Text composable

Description : "Displays text and follows the recommended Material Design guidelines."

Le composable Text est le plus utilisé dans Compose :

- Affiche du texte simple ou stylisé
- Support complet de l'accessibilité
- Peut utiliser des annotations pour du texte cliquable
- Suit les guidelines Material Design pour la typographie
- Supporte le texte multi-ligne et l'ellipsis

8.2 Image composable

Description : "Creates a composable that lays out and draws a given Painter class object."

Le composable Image affiche des ressources graphiques :

- Affiche des images depuis les ressources (drawable)
- Support des images depuis des URLs avec Coil
- Gère différents modes de mise à l'échelle (ContentScale)
- Peut afficher des Vector Drawables
- Support de la description pour l'accessibilité

8.3 Row composable

Description : "A layout composable that places its children in a horizontal sequence."

Le composable Row organise le contenu horizontalement :

- Place les enfants côte à côte horizontalement
- Supporte l'alignement vertical (Alignment.Top, Center, Bottom)
- Peut gérer l'arrangement horizontal (Arrangement.Start, Center, End, etc.)
- Utilisé pour les barres d'outils, boutons alignés, etc.
- Peut faire défiler horizontalement avec horizontalScroll()

8.4 Column composable

Description : "A layout composable that places its children in a vertical sequence."

Le composable Column organise le contenu verticalement :

- Place les enfants l'un en dessous de l'autre verticalement
- Supporte l'alignement horizontal (Alignment.Start, CenterHorizontally, End)
- Peut gérer l'arrangement vertical (Arrangement.Top, Center, Bottom, etc.)
- Utilisé pour la plupart des layouts de formulaires et listes
- Peut faire défiler verticalement avec verticalScroll()

11. Tests et validation