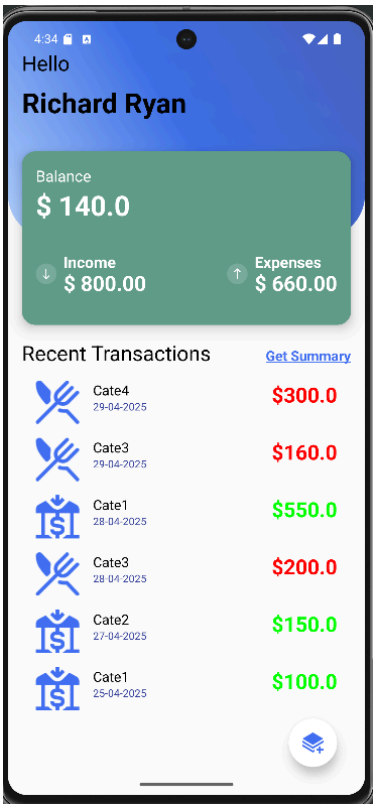
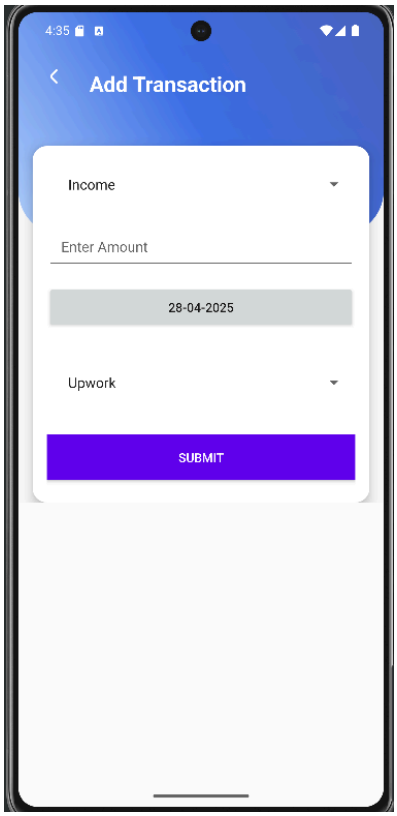
Project ETS
PBKK G

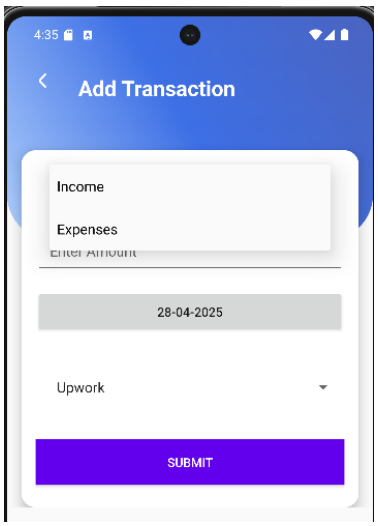
Richard Ryan - 5025211141

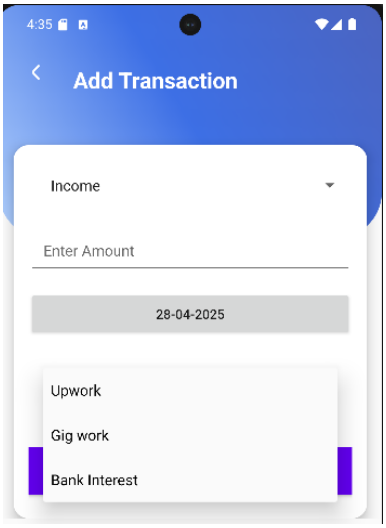
Tema aplikasi
Aplikasi Manajemen Finansial

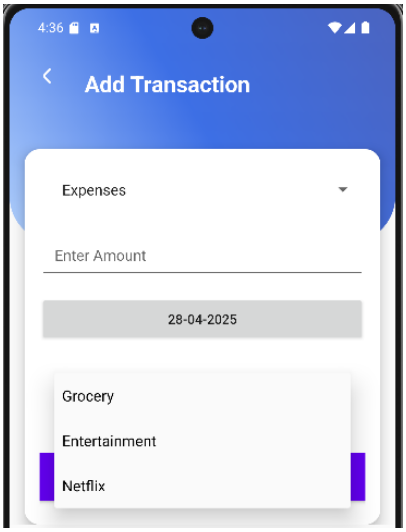
UI Aplikasi

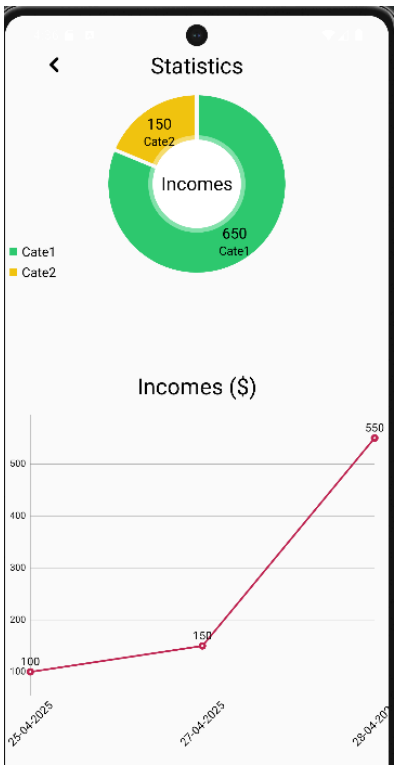


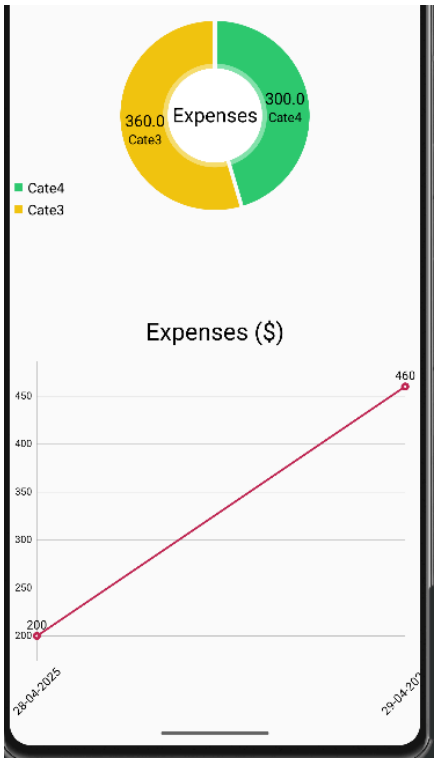



4:35
Add Transaction
Income
Expenses
28-04-2025
Upwork
SUBMIT


4:35
Add Transaction
Income
Enter Amount
28-04-2025
Upwork
Gig work
Bank Interest


4:36
Add Transaction
Expenses
Enter Amount
28-04-2025
Grocery
Entertainment
Netflix


Statistics
150
Cate2
Incomes
650
Cate1
Cate1
Cate2
Incomes ($)
550
100
150
25-04-2025
27-04-2025


300.0
Cate4
360.0
Cate3
Expenses
Cate4
Cate3
Expenses ($)
460
450
400
350
300
250
200
200
28-04-2025

Tahap Pengerjaan

1) Setup Aplikasi
   a) res / values / dimensions.xml
      File ini digunakan untuk menyimpan variabel variabel dimensi / resolusi untuk digunakan di seluruh aplikasi

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="intro_button_width">135dp</dimen>
    <dimen name="intro_button_height">50dp</dimen>
    <dimen name="dp_400">400dp</dimen>
    <dimen name="dp_300">300dp</dimen>
    <dimen name="dp_200">200dp</dimen>
    <dimen name="dp_128">128dp</dimen>
    <dimen name="dp_100">100dp</dimen>
    <dimen name="dp_80">80dp</dimen>
    <dimen name="dp_64">64dp</dimen>
    <dimen name="dp_50">50dp</dimen>
    <dimen name="dp_32">32dp</dimen>
    <dimen name="dp_24">24dp</dimen>
    <dimen name="dp_16">16dp</dimen>
    <dimen name="dp_8">8dp</dimen>
    <dimen name="dp_4">4dp</dimen>
    <dimen name="top_gradient_bg_height">250dp</dimen>
    <dimen name="balance_card_height">200dp</dimen>
</resources>
```

   b) res / values / colors.xml
      File ini digunakan untuk menyimpan variabel hex code warna

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="zinc">#BAC4C8</color>
    <color name="light_blue">#EAEEFC</color>
    <color name="blue">#4272F4</color>
    <color name="darkblue">#37449E</color>
    <color name="gray">#8E8E8E</color>
    <color name="lightgray">#DEDEDE</color>
    <color name="pink">#DBB7F1</color>
    <color name="green">#00FF00</color>
    <color name="darkgreen">#629C89</color>
    <color name="red">#FF00</color>
</resources>
```

c) res / values / string.xml

File ini digunakan untuk menyimpan string yang dapat digunakan di aplikasi

```xml
<resources>
    <string name="app_name">FinanceAppETS</string>
    <string name="Intro_bg_description">Intro Background Image</string>
    <string name="Intro_GetStarted">Get Started</string>
    <string name="test">Test ABC</string>
    <string name="Intro_app_desc">Finsera\n\nAplikasi Manajemen\nKeuangan\nRichard Ryan\n5025211141\nPPB G</string>
    <string name="username">Richard Ryan</string>
</resources>
```

d) app / manifests / AndroidManifests.xml

Tambahkan string berikut ke dalam tag <application>

```xml
        <activity
            android:name=".Home"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.FinanceAppETS2">
            <intent-filter...>
        </activity>
        <activity
            android:name=".AddTransaction"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.FinanceAppETS">
            <intent-filter...>
        </activity>
        <activity
            android:name=".Summary"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.FinanceAppETS">
            <intent-filter...>
        </activity>
```

2) Data, Tabel & Database

Akan digunakan database Room untuk menyimpan data

a) Database

```
package com.example.financeappets.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.example.financeappets.data.dao.TransactionEntityDao
import com.example.financeappets.data.model.TransactionEntity

@Database(entities = [TransactionEntity::class], version=1)
abstract class TransactionDatabase: RoomDatabase() {
    abstract fun transactionDao(): TransactionEntityDao

    companion object {
        @Volatile
        private var INSTANCE: TransactionDatabase? = null

        fun getDatabase(context: Context): TransactionDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    TransactionDatabase::class.java,
                    "finance_app_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

b) Entity untuk Transaksi

```
package com.example.financeappets.data.model

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "transaction_table")
data class TransactionEntity (
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val transactionType: String,
    val transactionCategory: String,
    val amount: Double,
    val date: String
)
```

c) Entity utilitas untuk perhitungan statistik harian

```
package com.example.financeappets.data.model

data class DailyTransaction(
    val date: String,
    val total: Double
)
```

d) DAO untuk manajemen tabel

File ini digunakan untuk menghubungkan fungsi dengan perintah SQL untuk dijalankan

- Query 1 : SELECT * FROM transaction_table ORDER BY date DESC
  Query ini digunakan untuk mengambil data transaksi terbaru untuk ditampilkan di halaman utama
- Query 2 : DELETE * FROM transaction_table
  Query ini opsional, digunakan untuk mereset tabel yang digunakan
- Query 3: SELECT date, SUM(amount) as total FROM transaction_table WHERE transactionType = 'Income' GROUP BY date ORDER BY date ASC
  Query ini digunakan untuk mendapatkan jumlah pendapatan yang didapatkan setiap harinya dengan mengagregasi jumlah nominal pemasukan berdasarkan tanggal. Query ini akan digunakan untuk membentuk grafik garis di halaman ringkasan transaksi
- Query 4: SELECT date, SUM(amount) as total FROM transaction_table WHERE transactionType = 'Expenses' GROUP BY date ORDER BY date ASC
  Query ini digunakan untuk mendapatkan jumlah pengeluaran yang didapatkan setiap harinya dengan mengagregasi jumlah nominal pengeluaran berdasarkan tanggal. Query ini akan digunakan untuk membentuk grafik garis di halaman ringkasan transaksi

```kotlin
package com.example.financeappets.data.dao

import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query
import com.example.financeappets.data.model.DailyTransaction
import com.example.financeappets.data.model.TransactionEntity

@Dao
interface TransactionEntityDao {
    @Query("SELECT * FROM transaction_table ORDER BY date DESC")
    suspend fun getAllTransactions(): List<TransactionEntity>

    @Insert
    suspend fun createTransaction(transaction: TransactionEntity)

    @Delete
    suspend fun deleteTransaction(transaction: TransactionEntity)

    @Query("DELETE FROM transaction_table")
    suspend fun deleteAllTransactions()

    @Query("SELECT date, SUM(amount) as total FROM transaction_table WHERE transactionType = 'Income' GROUP BY date ORDER BY date ASC")
    suspend fun getDailyIncome(): List<DailyTransaction>

    @Query("SELECT date, SUM(amount) as total FROM transaction_table WHERE transactionType = 'Expenses' GROUP BY date ORDER BY date ASC")
    suspend fun getDailyExpenses(): List<DailyTransaction>

}
```

e) Seeder

Apabila tabel kosong, maka isikan tabel dengan data yang diberikan

```kotlin
package com.example.financeappets.data.seeder

import com.example.financeappets.data.dao.TransactionEntityDao
import com.example.financeappets.data.model.TransactionEntity

class DatabaseSeeder(private val transactionDao: TransactionEntityDao) {

    suspend fun seed() {
        val transactions = transactionDao.getAllTransactions()
        if (transactions.isEmpty()) {
            val sampleTransactions = listOf(
                TransactionEntity(transactionType = "Income", transactionCategory = "Cate1", amount = 100.0, date = "25-04-2025"),
                TransactionEntity(transactionType = "Income", transactionCategory = "Cate2", amount = 150.0, date = "27-04-2025"),
                TransactionEntity(transactionType = "Income", transactionCategory = "Cate1", amount = 550.0, date = "28-04-2025"),
                TransactionEntity(transactionType = "Expenses", transactionCategory = "Cate3", amount = 200.0, date = "28-04-2025"),
                TransactionEntity(transactionType = "Expenses", transactionCategory = "Cate4", amount = 300.0, date = "29-04-2025"),
                TransactionEntity(transactionType = "Expenses", transactionCategory = "Cate3", amount = 160.0, date = "29-04-2025")
            )
            sampleTransactions.forEach {
                transactionDao.createTransaction(it)
            }
        }
    }
}
```

3) Halaman Awal
   a) Deklarasi variabel

```
private val myDatabase: TransactionDatabase by lazy {
    TransactionDatabase.getDatabase(applicationContext)
}

private val databaseSeeder: DatabaseSeeder by lazy {
    DatabaseSeeder(myDatabase.transactionDao())
}
```

   b) Inisialisasi tabel
   Opsional, hapus seluruh isi tabel kemudian seed ulang tabel

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    lifecycleScope.launch {
        val db = TransactionDatabase.getDatabase(this@MainActivity)
        val transactionDao = db.transactionDao()
        transactionDao.deleteAllTransactions()
        databaseSeeder.seed()
    }
```

   c) Layout

```
val layout = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
}
```

   d) Background

```
val imageView = ImageView(this).apply {
    id = ViewGroup.generateViewId()
    setImageResource(R.drawable.intro_pic)
    scaleType = ImageView.ScaleType.CENTER_CROP
    contentDescription = getString(R.string.Intro_bg_description)
}
```

e) Teks deskripsi

```
val descText = TextView(this).apply {
    id = ViewGroup.generateViewId()
    text = getString(R.string.Intro_app_desc)
    textSize = 30f
    setTextColor(ContextCompat.getColor(context, R.color.white))
    setTypeface(typeface, android.graphics.Typeface.BOLD)
}
```

f) Button

```
val startBtn = TextView(this).apply {
    id = ViewGroup.generateViewId()
    text = getString(R.string.Intro_GetStarted)
    textSize = 20f
    setTextColor(ContextCompat.getColor(context, R.color.white))
    gravity = android.view.Gravity.CENTER
    background = ContextCompat.getDrawable(context, R.drawable.home_textbox_bg)
    width = resources.getDimensionPixelSize(R.dimen.intro_button_width)
    height = resources.getDimensionPixelSize(R.dimen.intro_button_height)
    setOnClickListener {
        startActivity(Intent(this@MainActivity, Home::class.java))
    }
}
```

g) Penataan layout & penyusunan margin

```
layout.addView(imageView)
layout.addView(startBtn)
layout.addView(descText)

val set = ConstraintSet()
set.clone(layout)

set.connect(imageView.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
set.connect(imageView.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM)
set.connect(imageView.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
set.connect(imageView.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
set.constrainWidth(imageView.id, ConstraintSet.MATCH_CONSTRAINT)
set.constrainHeight(imageView.id, ConstraintSet.MATCH_CONSTRAINT)

set.connect(startBtn.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
set.connect(startBtn.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_32))

set.connect(descText.id, ConstraintSet.BOTTOM, startBtn.id, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_50))
set.connect(descText.id, ConstraintSet.START, startBtn.id, ConstraintSet.START)

set.applyTo(layout)

setContentView(layout)
```

4) Halaman Utama
   a) Layout root

```
val rootConstraint = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
}
```

   b) Scroll layout
   Layout ini digunakan sehingga halaman bisa discroll

```
val scrollView = NestedScrollView(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
}
```

   c) Layout Linear 1
   Layout ini digunakan untuk mengatur isi konten secara vertikal dan karena scroll layout hanya dapat memiliki 1 child layout

```
val linearLayout = LinearLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    orientation = LinearLayout.VERTICAL
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
}
```

d) Header

Bagian ini digunakan untuk UI bagian atas (lingkaran biru, dan teks nama)

```
val headerLayout = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.top_gradient_bg_height)
    )
}

val headerBackground = ImageView(this).apply {
    id = ConstraintLayout.generateViewId()
    setImageResource(R.drawable.top_gradient_display_bg)
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
    scaleType = ImageView.ScaleType.CENTER_CROP
}

val greetingText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()
    text = "Hello"
    textSize = 24f
    setTextColor(ContextCompat.getColor(context, R.color.black))
}
```

```
val greetingText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()
    text = "Hello"
    textSize = 24f
    setTextColor(ContextCompat.getColor(context, R.color.black))
}

val usernameText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()
    text = getString(R.string.username)
    textSize = 32f
    setTextColor(ContextCompat.getColor(context, R.color.black))
    setTypeface(typeface, android.graphics.Typeface.BOLD)
}
```

e) Kartu Nominal

Bagian ini digunakan untuk membuat UI kartu berisi teks nominal, total pendapatan dan pengeluaran

Pertama perlu dihitung terlebih dahulu total pendapatan dan pengeluaran, hal ini dilakukan dengan mengambil seluruh transaksi dan menjumlahkan nominal mereka berdasarkan jenis transaksi (pemasukan atau pengeluaran). Nominal akhir adalah selisih antara total pendapatan dan total pengeluaran

```
var totalIncome = 0.0
var totalExpense = 0.0
var balance = 0.0

lifecycleScope.launch {
    val db = TransactionDatabase.getDatabase(this@Home)
    val transactionDao = db.transactionDao()
    val transactionList = transactionDao.getAllTransactions()

    transactionList.forEach { transaction ->
        if (transaction.transactionType == "Expenses") {
            totalExpense = totalExpense + transaction.amount
        } else {
            totalIncome = totalIncome + transaction.amount
        }
    }
    balance = totalIncome - totalExpense
```

```
val greenCard = CardView(this@Home).apply {
    id = ConstraintLayout.generateViewId()
    radius = resources.getDimension(R.dimen.dp_16)
    cardElevation = resources.getDimension(R.dimen.dp_8)
    setCardBackgroundColor(ContextCompat.getColor(context, R.color.darkgreen))
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_200)
    ).apply {
        setMargins(
            resources.getDimensionPixelSize(R.dimen.dp_16),
            -resources.getDimensionPixelSize(R.dimen.dp_100),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            0
        )
    }
```

```
 val cardContentLayout = LinearLayout(context).apply {
     orientation = LinearLayout.VERTICAL
     layoutParams = LinearLayout.LayoutParams(
         ViewGroup.LayoutParams.MATCH_PARENT,
         ViewGroup.LayoutParams.MATCH_PARENT
     )
     setPadding(
         resources.getDimensionPixelSize(R.dimen.dp_16),
         resources.getDimensionPixelSize(R.dimen.dp_16),
         resources.getDimensionPixelSize(R.dimen.dp_16),
         resources.getDimensionPixelSize(R.dimen.dp_16)
     )
 }

 val balanceTitle = TextView(context).apply {
     text = "Balance"
     textSize = 18f
     setTextColor(ContextCompat.getColor(context, android.R.color.white))
 }
```

```
val balanceAmount = TextView(context).apply {
    text = "\$ ${balance}"
    textSize = 32f
    setTextColor(ContextCompat.getColor(context, android.R.color.white))
    setTypeface(typeface, android.graphics.Typeface.BOLD)
}

cardContentLayout.addView(balanceTitle)
cardContentLayout.addView(balanceAmount)

val gap = View(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_32)
    )
}

cardContentLayout.addView(gap)

val incomeExpensesLayout = LinearLayout(context).apply {
    orientation = LinearLayout.HORIZONTAL
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
    weightSum = 2f
}
```

```
val incomeSection = LinearLayout(context).apply {
    orientation = LinearLayout.HORIZONTAL
    layoutParams = LinearLayout.LayoutParams(
        0, ViewGroup.LayoutParams.WRAP_CONTENT, 1f
    )
    gravity = Gravity.START or Gravity.CENTER_VERTICAL
}

val incomeArrow = ImageView(context).apply {
    setImageResource(R.drawable.income_arrow)
    layoutParams = LinearLayout.LayoutParams(
        resources.getDimensionPixelSize(R.dimen.dp_24),
        resources.getDimensionPixelSize(R.dimen.dp_24)
    ).apply {
        setMargins(0, 0, resources.getDimensionPixelSize(R.dimen.dp_8), 0)
    }
}

val combinedText1 = SpannableString("Income\n\$ ${"%.2f".format(totalIncome)}").apply {
    setSpan(AbsoluteSizeSpan(18, true), 0, 6, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
    setSpan(AbsoluteSizeSpan(24, true), 7, length, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
}
```

```
val incomeTextView = TextView(context).apply {
    text = combinedText1
    setTextColor(ContextCompat.getColor(context, android.R.color.white))
    setTypeface(typeface, android.graphics.Typeface.BOLD)
}

incomeSection.addView(incomeArrow)
incomeSection.addView(incomeTextView)

val expensesSection = LinearLayout(context).apply {
    orientation = LinearLayout.HORIZONTAL
    layoutParams = LinearLayout.LayoutParams(0, ViewGroup.LayoutParams.WRAP_CONTENT, 1f)
    gravity = Gravity.END or Gravity.CENTER_VERTICAL
}

val expenseArrow = ImageView(context).apply {
    setImageResource(R.drawable.expense_arrow)
    layoutParams = LinearLayout.LayoutParams(
        resources.getDimensionPixelSize(R.dimen.dp_24),
        resources.getDimensionPixelSize(R.dimen.dp_24)
    ).apply {
        setMargins(0, 0, resources.getDimensionPixelSize(R.dimen.dp_8), 0)
    }
}
```

```
        val combinedText2 = SpannableString("Expenses\n\$ ${"%.2f".format(totalExpense)}").apply {
            setSpan(AbsoluteSizeSpan(18, true), 0, 8, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
            setSpan(AbsoluteSizeSpan(24, true), 9, length, Spanned.SPAN_EXCLUSIVE_EXCLUSIVE)
        }

        val expensesTextView = TextView(context).apply {
            text = combinedText2
            setTextColor(ContextCompat.getColor(context, android.R.color.white))
            setTypeface(typeface, android.graphics.Typeface.BOLD)
        }

        expensesSection.addView(expenseArrow)
        expensesSection.addView(expensesTextView)

        incomeExpensesLayout.addView(incomeSection)
        incomeExpensesLayout.addView(expensesSection)

        cardContentLayout.addView(incomeExpensesLayout)

        addView(cardContentLayout)
    }
    linearLayout.addView(greenCard, 1)
}
```

f) Linear Layout 2

Layout ini digunakan untuk menampilkan judul untuk transaksi terbaru dan tombol untuk berpindah ke halaman ringkasan secara horizontal dengan baik

```
val linearLayout2 = LinearLayout(this).apply {
    orientation = LinearLayout.HORIZONTAL
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
    weightSum = 3f
}

val recentText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()
    text = "Recent Transactions"
    textSize = 24f
    setTextColor(ContextCompat.getColor(context, R.color.black))
    layoutParams = LinearLayout.LayoutParams(
        0, ViewGroup.LayoutParams.WRAP_CONTENT, 2f
    ).apply {
        setMargins(
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            0
        )
    }
    gravity = Gravity.START or Gravity.CENTER_VERTICAL
}
```

```
val summaryText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()

    val content = "Get Summary"
    val underlinedText = SpannableString(content).apply {
        setSpan(UnderlineSpan(), 0, content.length, 0)
    }
    text = underlinedText
    textSize = 16f
    setTextColor(ContextCompat.getColor(context, R.color.blue))
    layoutParams = LinearLayout.LayoutParams(
        0, ViewGroup.LayoutParams.WRAP_CONTENT, 1f
    ).apply {
        setMargins(
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            0
        )
    }
    gravity = Gravity.END or Gravity.CENTER_VERTICAL
    setTypeface(typeface, android.graphics.Typeface.BOLD)

    setOnClickListener {
        startActivity(Intent(this@Home, Summary::class.java))
    }
}
```

g) Item transaksi

Bagian ini digunakan untuk menampilkan daftar transaksi terbaru. Tampilan akan dibedakan berdasarkan jenis transaksi, sebagai contoh icon berbeda dan nominal akan berwarna hijau jika transaksi merupakan pemasukan namun akan berwarna merah jika merupakan pengeluaran

```
fun addTransactionItem(parent: LinearLayout, type: String, category: String, date: String, amount: String) {
    val transactionItemLayout = LinearLayout(this).apply {
        orientation = LinearLayout.HORIZONTAL
        layoutParams = LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
        )
        setPadding(
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_8),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            resources.getDimensionPixelSize(R.dimen.dp_8)
        )
    }
```

```
val icon = ImageView(this).apply {
    if (type == "Income")
    {
        setImageResource(R.drawable.btn_1)
    } else {
        setImageResource(R.drawable.img1)
    }
    layoutParams = LinearLayout.LayoutParams(
        resources.getDimensionPixelSize(R.dimen.dp_50),
        resources.getDimensionPixelSize(R.dimen.dp_50)
    ).apply {
        setMargins(
            0,
            0,
            resources.getDimensionPixelSize(R.dimen.dp_16),
            0
        )
    }
}

val textLayout = LinearLayout(this).apply {
    orientation = LinearLayout.VERTICAL
    layoutParams = LinearLayout.LayoutParams(
        0, ViewGroup.LayoutParams.WRAP_CONTENT, 1f
    )
}
```

```
val transactionCategory = TextView(this).apply {
    text = category
    textSize = 16f
    setTextColor(ContextCompat.getColor(context, R.color.black))
}

val transactionDate = TextView(this).apply {
    text = date
    textSize = 12f
    setTextColor(ContextCompat.getColor(context, R.color.darkblue))
}
```

```
    val transactionAmount = TextView(this).apply {
        text = amount
        textSize = 24f
        if (type == "Income")
        {
            setTextColor(ContextCompat.getColor(context, R.color.green))
        } else {
            setTextColor(ContextCompat.getColor(context, R.color.red))
        }
        layoutParams = LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
        ).apply {
            marginStart = resources.getDimensionPixelSize(R.dimen.dp_16)
            gravity = Gravity.END
        }
        setTypeface(typeface, android.graphics.Typeface.BOLD)
    }

    textLayout.addView(transactionCategory)
    textLayout.addView(transactionDate)

    transactionItemLayout.addView(icon)
    transactionItemLayout.addView(textLayout)
    transactionItemLayout.addView(transactionAmount)

    parent.addView(transactionItemLayout)
}
```

```
val transactionListLayout = LinearLayout(this).apply {
    orientation = LinearLayout.VERTICAL
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_8),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_8)
    )
}
```

h) Tombol tambah transaksi

```
val fab = FloatingActionButton(this).apply {
    id = ViewGroup.generateViewId()
    setImageResource(R.drawable.btn_3)
    contentDescription = "Add Transaction"
    backgroundTintList = ContextCompat.getColorStateList(context, R.color.white)
    setOnClickListener {
        startActivity(Intent(this@Home, AddTransaction::class.java))
    }
}
```

i) Penambahan jeda

```
val gap = View(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_80)
    )
}
```

j) Pengaturan layout & margin

```
headerLayout.addView(headerBackground)
headerLayout.addView(greetingText)
headerLayout.addView(usernameText)

linearLayout.addView(headerLayout)
linearLayout2.addView(recentText)
linearLayout2.addView(summaryText)
linearLayout.addView(linearLayout2)
lifecycleScope.launch {
    val db = TransactionDatabase.getDatabase(this@Home)
    val transactionDao = db.transactionDao()
    val transactionList = transactionDao.getAllTransactions()

    transactionList.forEach { transaction ->
        addTransactionItem(
            transactionListLayout,
            transaction.transactionType,
            transaction.transactionCategory,
            transaction.date,
            "$${transaction.amount}"
        )
    }
}
linearLayout.addView(transactionListLayout)
linearLayout.addView(gap)
```

```
scrollView.addView(linearLayout)
rootConstraint.addView(scrollView)
rootConstraint.addView(fab)

val headerSet = ConstraintSet()
headerSet.clone(headerLayout)

headerSet.connect(headerBackground.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
headerSet.connect(headerBackground.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
headerSet.connect(headerBackground.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
headerSet.connect(headerBackground.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM)

headerSet.connect(greetingText.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_32))
headerSet.connect(greetingText.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_16))

headerSet.connect(usernameText.id, ConstraintSet.TOP, greetingText.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_8))
headerSet.connect(usernameText.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_16))

headerSet.applyTo(headerLayout)

val rootSet = ConstraintSet()
rootSet.clone(rootConstraint)

rootSet.connect(scrollView.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
rootSet.connect(scrollView.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(scrollView.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
rootSet.connect(scrollView.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM)
```

```
rootSet.connect(scrollView.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
rootSet.connect(scrollView.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(scrollView.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
rootSet.connect(scrollView.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM)

rootSet.connect(fab.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(fab.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.constrainWidth(fab.id, ConstraintSet.WRAP_CONTENT)
rootSet.constrainHeight(fab.id, ConstraintSet.WRAP_CONTENT)
rootSet.connect(gap.id, ConstraintSet.TOP, fab.id, ConstraintSet.BOTTOM)

rootSet.applyTo(rootConstraint)

setContentView(rootConstraint)
```

5) Halaman Penambahan Transaksi
   a) Layout root

```
val rootConstraint = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
}
```

b) Header

```
val headerLayout = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.top_gradient_bg_height)
    )
}

val headerBackground = ImageView(this).apply {
    id = ConstraintLayout.generateViewId()
    setImageResource(R.drawable.top_gradient_display_bg)
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
    scaleType = ImageView.ScaleType.CENTER_CROP
}
```

```
val icon = ImageView(this).apply {
    id = ConstraintLayout.generateViewId()
    setImageResource(R.drawable.arrow)
    layoutParams = ConstraintLayout.LayoutParams(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
    setOnClickListener {
        startActivity(Intent(this@AddTransaction, Home::class.java))
    }
}

val titleText = TextView(this).apply {
    id = ConstraintLayout.generateViewId()
    text = "Add Transaction"
    textSize = 24f
    setTextColor(ContextCompat.getColor(context, R.color.white))
    setTypeface(typeface, android.graphics.Typeface.BOLD)
}
```

c) Linear Layout 1

Layout digunakan untuk mengatur konten secara vertikal dengan rapi

```
val linearLayout1 = LinearLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    orientation = LinearLayout.VERTICAL
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
}
```

d) Form

```
val form = CardView(this).apply {
    id = ConstraintLayout.generateViewId()
    radius = resources.getDimension(R.dimen.dp_16)
    cardElevation = resources.getDimension(R.dimen.dp_8)
    setCardBackgroundColor(ContextCompat.getColor(context, R.color.white))
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_400)
    ).apply {
        setMargins(
            resources.getDimensionPixelSize(R.dimen.dp_16),
            -resources.getDimensionPixelSize(R.dimen.dp_100),
            resources.getDimensionPixelSize(R.dimen.dp_16),
            0
        )
    }
```

Buat layout untuk penataan isi form

```
val cardContentLayout = LinearLayout(context).apply {
    orientation = LinearLayout.VERTICAL
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
}
```

Dropdown untuk pemilihan tipe transaksi (pemasukan / pengeluaran)

```
val transactionTypeSpinner = Spinner(context).apply {
    val adapter = ArrayAdapter(
        context,
        android.R.layout.simple_spinner_item,
        listOf("Income", "Expenses")
    )
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    this.adapter = adapter
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
}
```

Textarea untuk nominal transaksi

```
val amountEditText = EditText(context).apply {
    hint = "Enter Amount"
    inputType = android.text.InputType.TYPE_CLASS_NUMBER or android.text.InputType.TYPE_NUMBER_FLAG_DECIMAL
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
    setTextSize(TypedValue.COMPLEX_UNIT_SP, 16f)
}
```

Pemilihan tanggal transaksi dengan default tanggal saat ini. Tanggal disimpan sebagai string dengan format dd-mm-yyyy

```
val dateButton = Button(context).apply {
    text = "Select Date"
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
    setOnClickListener {
        val calendar = Calendar.getInstance()
        val datePicker = DatePickerDialog(
            context,
            { _, year, month, dayOfMonth ->
                text = String.format("%02d-%02d-%d", dayOfMonth, month + 1, year)
            },
            calendar.get(Calendar.YEAR),
            calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH)
        )
        datePicker.show()
    }
}

val calendar = Calendar.getInstance()
val defaultDate = String.format("%02d-%02d-%d", calendar.get(Calendar.DAY_OF_MONTH), calendar.get(Calendar.MONTH) + 1, calendar.get(Calendar.YEAR))
dateButton.text = defaultDate
```

Dropdown untuk pemilihan kategori transaksi

```
val expenseArray = listOf("Grocery", "Entertainment", "Netflix")
val incomeArray = listOf("Upwork", "Gig work", "Bank Interest")

val typeSelectionSpinner = Spinner(context).apply {
    val adapter = ArrayAdapter(
        context,
        android.R.layout.simple_spinner_item,
        incomeArray
    )
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
    this.adapter = adapter
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
}
```

```
transactionTypeSpinner.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {
    override fun onItemSelected(parentView: AdapterView<*>, view: View?, position: Int, id: Long) {
        val updatedAdapter: ArrayAdapter<String> = when (position) {
            0 -> ArrayAdapter(context, android.R.layout.simple_spinner_item, incomeArray)
            1 -> ArrayAdapter(context, android.R.layout.simple_spinner_item, expenseArray)
            else -> ArrayAdapter(context, android.R.layout.simple_spinner_item, incomeArray)
        }

        updatedAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
        typeSelectionSpinner.adapter = updatedAdapter
    }

    override fun onNothingSelected(parentView: AdapterView<*>) { }
}
```

Button Submit. Pertama akan dipastikan bahwa nominal valid sebelum tombol bisa diklik

```
val submitButton = Button(context).apply {
    text = "Submit"
    setBackgroundColor(ContextCompat.getColor(context,R.color.purple_500))
    setTextColor(ContextCompat.getColor(context,R.color.white))
    setPadding(
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16),
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )

    amountEditText.setOnFocusChangeListener { _, hasFocus ->
        if (!hasFocus) {
            val amountText = amountEditText.text.toString()
            isEnabled = !amountText.isNullOrEmpty() && amountText.toDoubleOrNull() != null && amountText.toDouble() > 0
        }
    }
```

```
    setOnClickListener {
        val db = TransactionDatabase.getDatabase(context)
        val transactionDao = db.transactionDao()

        val transaction = TransactionEntity(
            transactionType = transactionTypeSpinner.selectedItem.toString(),
            amount = amountEditText.text.toString().toDouble(),
            date = dateButton.text.toString(),
            transactionCategory = typeSelectionSpinner.selectedItem.toString()
        )

        lifecycleScope.launch {
            transactionDao.createTransaction(transaction)
        }

        startActivity(Intent(this@AddTransaction, Home::class.java))
    }
}
```

e) Pembuatan jeda 16/32 dp

```
val gap1 = View(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
}

val gap2 = View(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_16)
    )
}

val gap3 = View(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_32)
    )
}

val gap4 = View(context).apply {
    layoutParams = LinearLayout.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_32)
    )
}
```

f) Pengaturan layout & margin

```
        cardContentLayout.apply {
            addView(transactionTypeSpinner)
            addView(gap1)
            addView(amountEditText)
            addView(gap2)
            addView(dateButton)
            addView(gap3)
            addView(typeSelectionSpinner)
            addView(gap4)
            addView(submitButton)
        }

        addView(cardContentLayout)
    }


    headerLayout.addView(headerBackground)
    headerLayout.addView(icon)
    headerLayout.addView(titleText)

    linearLayout1.addView(headerLayout)
    linearLayout1.addView(form)

    rootConstraint.addView(linearLayout1)

    val headerSet = ConstraintSet()
    headerSet.clone(headerLayout)
```

```
headerSet.connect(headerBackground.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
headerSet.connect(headerBackground.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
headerSet.connect(headerBackground.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
headerSet.connect(headerBackground.id, ConstraintSet.BOTTOM, ConstraintSet.PARENT_ID, ConstraintSet.BOTTOM)

headerSet.connect(icon.id, ConstraintSet.START, headerLayout.id, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_32))
headerSet.connect(icon.id, ConstraintSet.TOP, headerLayout.id, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_64))

headerSet.connect(titleText.id, ConstraintSet.START, icon.id, ConstraintSet.END, resources.getDimensionPixelSize(R.dimen.dp_32))
headerSet.connect(titleText.id, ConstraintSet.TOP, headerLayout.id, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_64))

headerSet.applyTo(headerLayout)

setContentView(rootConstraint)
```

6) Halaman Rangkuman Transaksi

a) Layout root

```
val rootConstraint = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    )
}
```

b) Header

```
val topBar = ConstraintLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_80)
    )
    setPadding(16, 16, 16, 16)
}

val backArrow = ImageView(this).apply {
    id = View.generateViewId()
    setImageResource(R.drawable.black_back_arrow)
    layoutParams = ConstraintLayout.LayoutParams(
        resources.getDimensionPixelSize(R.dimen.dp_32),
        resources.getDimensionPixelSize(R.dimen.dp_32)
    )
    setOnClickListener {
        startActivity(Intent(this@Summary, Home::class.java))
    }
}
```

```
val titleText = TextView(this).apply {
    id = View.generateViewId()
    text = "Statistics"
    textSize = 24f
    setTextColor(ContextCompat.getColor(context, R.color.black))
    layoutParams = ConstraintLayout.LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.MATCH_PARENT
    ).apply {
        leftToLeft = backArrow.id
        rightToRight = ConstraintLayout.LayoutParams.PARENT_ID
        topToTop = backArrow.id
        bottomToBottom = backArrow.id
    }
}
```

c) Scroll Layout

Layout digunakan sehingga tampilan bisa discroll

```
val scrollView = NestedScrollView(this).apply {
    id = ConstraintLayout.generateViewId()
    layoutParams = ConstraintLayout.LayoutParams(
        ConstraintLayout.LayoutParams.MATCH_PARENT,
        0
    ).apply {
        topToBottom = topBar.id
        startToStart = ConstraintSet.PARENT_ID
        endToEnd = ConstraintSet.PARENT_ID
        bottomToBottom = ConstraintSet.PARENT_ID
    }
}
```

d) Linear Layout

Layout ini digunakan untuk mengatur isi konten secara vertikal dan karena scroll layout hanya dapat memiliki 1 child layout

```
val linearLayout = LinearLayout(this).apply {
    id = ConstraintLayout.generateViewId()
    orientation = LinearLayout.VERTICAL
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        ViewGroup.LayoutParams.WRAP_CONTENT
    )
}
```

e) Pie Chart untuk pemasukan & pengeluaran

Pertama harus diambil dahulu seluruh data kemudian jumlahkan nominal berdasarkan kategori.

```
var incomePieChart = PieChart(this).apply {
    id = View.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_300)
    )
}

lifecycleScope.launch {
    val transactions = transactionDao.getAllTransactions()

    val incomeCategories = mutableMapOf<String, Double>()
    transactions.forEach {
        if (it.transactionType == "Income") {
            incomeCategories[it.transactionCategory] = incomeCategories.getOrDefault(it.transactionCategory, 0.0) + it.amount
        }
    }

    val pieEntries = incomeCategories.map { PieEntry(it.value.toFloat(), it.key) }

    val dataSet = PieDataSet(pieEntries, "").apply {
        colors = ColorTemplate.MATERIAL_COLORS.toList()
        sliceSpace = 4f
        valueTextSize = 16f
        valueTextColor = ContextCompat.getColor(this@Summary, R.color.black)
    }
```

```
    val pieData = PieData(dataSet)
    incomePieChart.apply {
        this.data = pieData
        invalidate()
        description.isEnabled = false
        setCenterText("Incomes")
        setDrawEntryLabels(true)
        setEntryLabelColor(ContextCompat.getColor(this@Summary, R.color.black))
        setCenterTextSize(20f)
        legend.textSize = 14f
        legend.orientation = LegendOrientation.VERTICAL
        setExtraBottomOffset(50f)
    }
}
```

f) Grafik garis untuk pemasukan & pengeluaran

Untuk mempermudah operasi, agregasi akan dilakukan melalui SQL sehingga hasil SQL dapat langsung digunakan sebagai data untuk grafik garis

```
val incomeLineChart = LineChart(this).apply {
    id = View.generateViewId()
    layoutParams = ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.MATCH_PARENT,
        resources.getDimensionPixelSize(R.dimen.dp_400)
    )
}

lifecycleScope.launch {
    val dailyIncome = transactionDao.getDailyIncome()

    val dateFormat = SimpleDateFormat("dd-MM-yyyy", Locale.getDefault())
    val labels = dailyIncome.map {
        try {
            val parsedDate = dateFormat.parse(it.date)
            parsedDate?.let { dateFormat.format(it) } ?: "Invalid"
        } catch (e: Exception) {
            "Invalid"
        }
    }

    val incomeEntries = dailyIncome.mapIndexed { index, dailyTransaction ->
        Entry(index.toFloat(), dailyTransaction.total.toFloat())
    }
```

```
val incomeDataSet = LineDataSet(incomeEntries, "Income").apply {
    color = ColorTemplate.COLORFUL_COLORS[0]
    valueTextColor = resources.getColor(android.R.color.black, null)
    valueTextSize = 12f
    lineWidth = 2f
    setCircleColor(ColorTemplate.COLORFUL_COLORS[0])
    circleRadius = 4f
}

val formatter = object : ValueFormatter() {
    override fun getFormattedValue(value: Float): String {
        val index = value.toInt()
        return if (index in labels.indices) labels[index] else ""
    }
}

incomeLineChart.data = LineData(incomeDataSet)
```

```
    incomeLineChart.apply {
        description.isEnabled = false

        axisLeft.enableGridDashedLine(0f, 0f, 0f)
        axisRight.isEnabled = false

        xAxis.apply {
            setDrawGridLines(false)
            setDrawAxisLine(false)
            position = XAxis.XAxisPosition.BOTTOM
            labelRotationAngle = -45f
            valueFormatter = formatter
            granularity = 1f
            isGranularityEnabled = true
            textColor = resources.getColor(android.R.color.black, null)
            textSize = 12f
        }

        legend.isEnabled = false

        setExtraBottomOffset(30f)

        invalidate()
    }
}
```

g) Pengaturan layout & margin

```
topBar.addView(backArrow)
topBar.addView(titleText)

linearLayout.addView(incomePieChart)
linearLayout.addView(incomeLineText)
linearLayout.addView(incomeLineChart)
linearLayout.addView(expensePieChart)
linearLayout.addView(expenseLineText)
linearLayout.addView(expenseLineChart)

scrollView.addView(linearLayout)

rootConstraint.addView(topBar)
rootConstraint.addView(scrollView)

val topBarSet = ConstraintSet()
topBarSet.clone(topBar)

topBarSet.connect(backArrow.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_32))
topBarSet.connect(backArrow.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_32))
topBarSet.connect(titleText.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP, resources.getDimensionPixelSize(R.dimen.dp_32))
topBarSet.connect(titleText.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
topBarSet.connect(titleText.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
```

```
topBarSet.applyTo(topBar)

val rootSet = ConstraintSet()
rootSet.clone(rootConstraint)

rootSet.connect(topBar.id, ConstraintSet.TOP, ConstraintSet.PARENT_ID, ConstraintSet.TOP)
rootSet.connect(topBar.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(topBar.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)

rootSet.connect(incomePieChart.id, ConstraintSet.TOP, scrollView.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(incomePieChart.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(incomePieChart.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)

rootSet.connect(incomeLineText.id, ConstraintSet.TOP, incomePieChart.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(incomeLineText.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(incomeLineText.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)

rootSet.connect(incomeLineChart.id, ConstraintSet.TOP, incomeLineText.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(incomeLineChart.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_16))
rootSet.connect(incomeLineChart.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END, resources.getDimensionPixelSize(R.dimen.dp_16))

rootSet.connect(expensePieChart.id, ConstraintSet.TOP, incomeLineChart.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(expensePieChart.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(expensePieChart.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)
```

```
rootSet.connect(expenseLineText.id, ConstraintSet.TOP, expensePieChart.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(expenseLineText.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START)
rootSet.connect(expenseLineText.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END)

rootSet.connect(expenseLineChart.id, ConstraintSet.TOP, expensePieChart.id, ConstraintSet.BOTTOM, resources.getDimensionPixelSize(R.dimen.dp_32))
rootSet.connect(expenseLineChart.id, ConstraintSet.START, ConstraintSet.PARENT_ID, ConstraintSet.START, resources.getDimensionPixelSize(R.dimen.dp_16))
rootSet.connect(expenseLineChart.id, ConstraintSet.END, ConstraintSet.PARENT_ID, ConstraintSet.END, resources.getDimensionPixelSize(R.dimen.dp_16))


rootSet.applyTo(rootConstraint)

setContentView(rootConstraint)
```