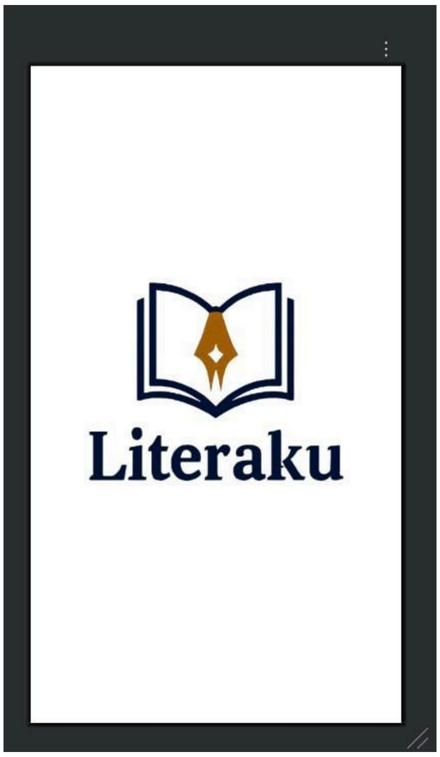
Nama: Muhammad Rizki Pratama

Npm  : 23312191

Kelas: IF 23 FX

**DOKUMENTASI**

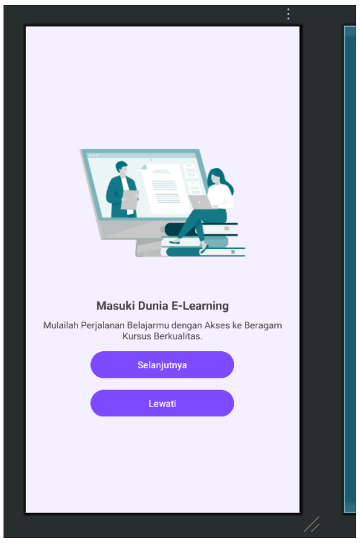
1. Tampilan awal ketika aplikasi pertama kali dijalankan. Lalu implementasi aktivitas splash screen yang akan tampil selama beberapa detik lalu berpindah ke halaman



berikutnya

2. **SplashActivity** tampil 3 detik.

1. Setelah itu menampilkan **FragmentIntro1**.
2. Di FragmentIntro1:
   Tombol "Selanjutnya" → ke `FragmentIntro2`.
   Tombol "Lewati" → langsung ke `FragmentUtama`.



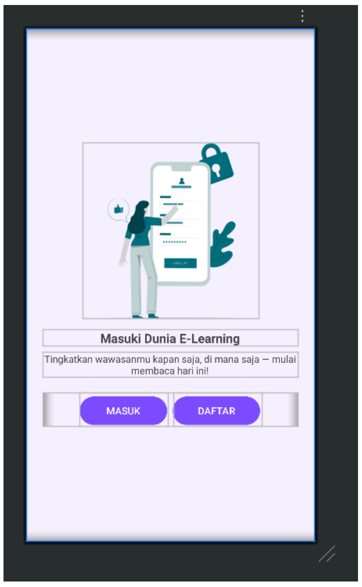
Masuki Dunia E-Learning

Mulailah Perjalanan Belajarmu dengan Akses ke Beragam Kursus Berkualitas.

3. Tampilan ini adalah **halaman ketiga** dari rangkaian onboarding atau pengenalan aplikasi.

**Dua Tombol Aksi**:

- **MASUK** → Arahkan ke halaman login.
- **DAFTAR** → Arahkan ke halaman pendaftaran.



Masuki Dunia E-Learning

Tingkatkan wawasanmu kapan saja, di mana saja – mulai membaca hari ini!

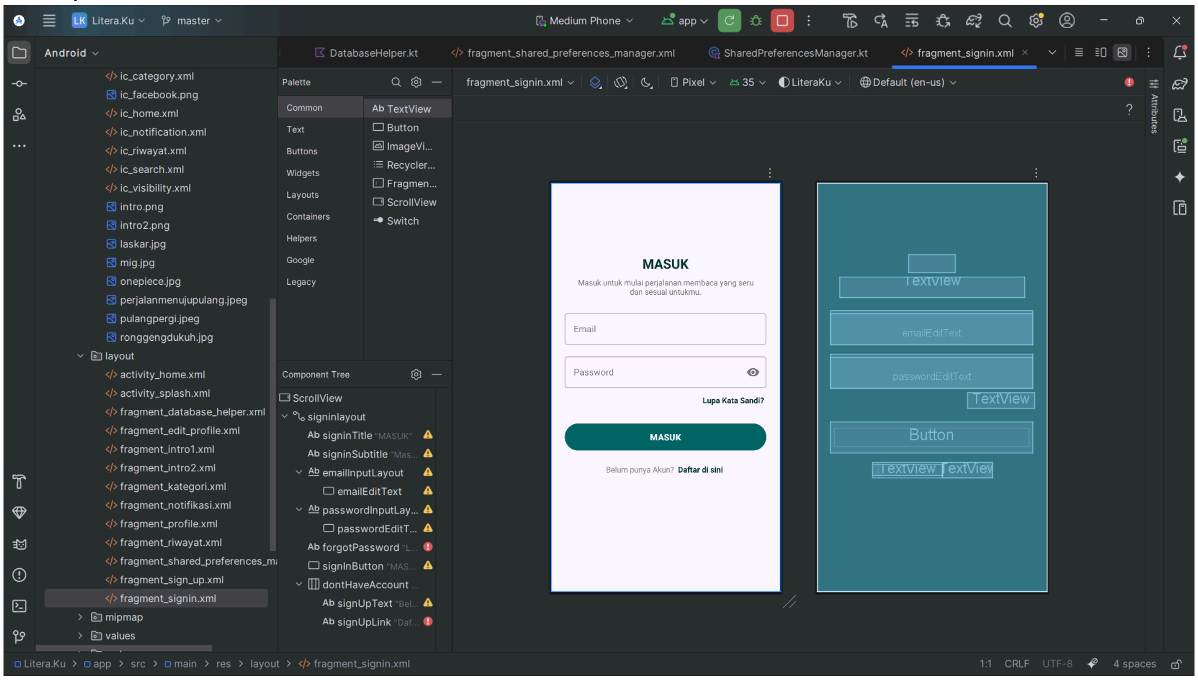
4. Saya membuat database untuk menampung id,name,email,password untuk bisa membuat fungsi pada tampilan signin dan signup.

```
package com.example.yourapp.database

import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

data class User(
   val id: Int,
   val fullName: String,
   val email: String,
   val password: String
)

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {

   companion object {
       private const val DATABASE_NAME = "userDB"
       private const val DATABASE_VERSION = 1
       private const val TABLE_USER = "user"
       private const val COLUMN_ID = "id"
       private const val COLUMN_NAME = "full_name"
       private const val COLUMN_EMAIL = "email"
       private const val COLUMN_PASSWORD = "password"
   }

   override fun onCreate(db: SQLiteDatabase?) {
       // Create the user table
       val CREATE_USER_TABLE = "CREATE TABLE $TABLE_USER (" +
               "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT," +
               "$COLUMN_NAME TEXT," +
               "$COLUMN_EMAIL TEXT UNIQUE," +
               "$COLUMN_PASSWORD TEXT)"
       db?.execSQL(CREATE_USER_TABLE)
   }

   override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
       // Drop older table if exists
       db?.execSQL("DROP TABLE IF EXISTS $TABLE_USER")
       // Create new table
       onCreate(db)
   }

   // Method to insert user
   fun insertUser(fullName: String, email: String, password: String): Long {
       val db = writableDatabase
       val values = ContentValues().apply {
           put(COLUMN_NAME, fullName)
           put(COLUMN_EMAIL, email)
           put(COLUMN_PASSWORD, password)
```

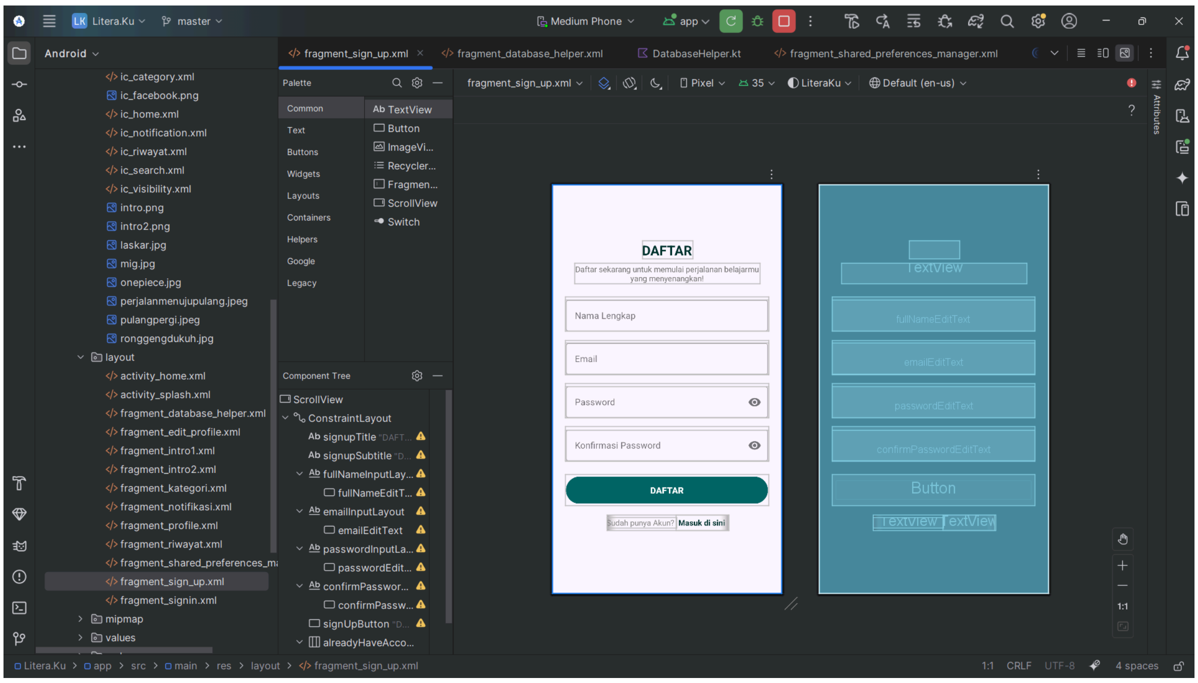
```kotlin
        }
        // Insert the data into the database and return the row ID
        return db.insert(TABLE_USER, null, values).also {
            db.close()  // Ensure the database is closed after the insert
        }
    }

    // Method to check if the user exists and validate credentials
    fun isValidUser(email: String, password: String): Boolean {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_EMAIL = ? AND
$COLUMN_PASSWORD = ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(email, password))

        val isValid = cursor.count > 0
        cursor.close()  // Always close the cursor
        db.close()  // Always close the database
        return isValid
    }

    // Method to get user data by email and password
    fun getUserData(email: String, password: String): User? {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_EMAIL = ? AND
$COLUMN_PASSWORD = ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(email, password))

        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_ID)),
                fullName =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME)),
                email =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_PASSWORD))
            )
        }
        cursor.close()
        db.close()
        return user
    }

    // Method to get user data by ID
    fun getUserById(userId: Int): User? {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_ID = ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(userId.toString()))

        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_ID)),
                fullName =
 cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME)),
                email =
 cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_EMAIL)),
                password =
 cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_PASSWORD))
            )
        }
        cursor.close()
        db.close()
        return user
    }

    // Method to update user data
    fun updateUser(userId: Int, fullName: String, email: String, password:
 String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, fullName)
            put(COLUMN_EMAIL, email)
            put(COLUMN_PASSWORD, password)
        }

        val result = db.update(TABLE_USER, values, "$COLUMN_ID = ?",
 arrayOf(userId.toString()))
        db.close()
        return result > 0
    }

    // Method to check if email already exists (excluding current user)
    fun isEmailExistsForOtherUser(email: String, currentUserId: Int): Boolean
 {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_EMAIL = ? AND
 $COLUMN_ID != ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(email,
 currentUserId.toString()))

        val exists = cursor.count > 0
        cursor.close()
        db.close()
        return exists
    }

    // Method to check if email already exists
    fun isEmailExists(email: String): Boolean {
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_EMAIL = ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(email))

        val exists = cursor.count > 0
        cursor.close()
        db.close()
        return exists
    }
```

```
    // Method to verify current password
    fun verifyCurrentPassword(userId: Int, currentPassword: String): Boolean
{
        val db = readableDatabase
        val query = "SELECT * FROM $TABLE_USER WHERE $COLUMN_ID = ? AND
$COLUMN_PASSWORD = ?"
        val cursor: Cursor = db.rawQuery(query, arrayOf(userId.toString(),
currentPassword))

        val isValid = cursor.count > 0
        cursor.close()
        db.close()
        return isValid
    }
}
```

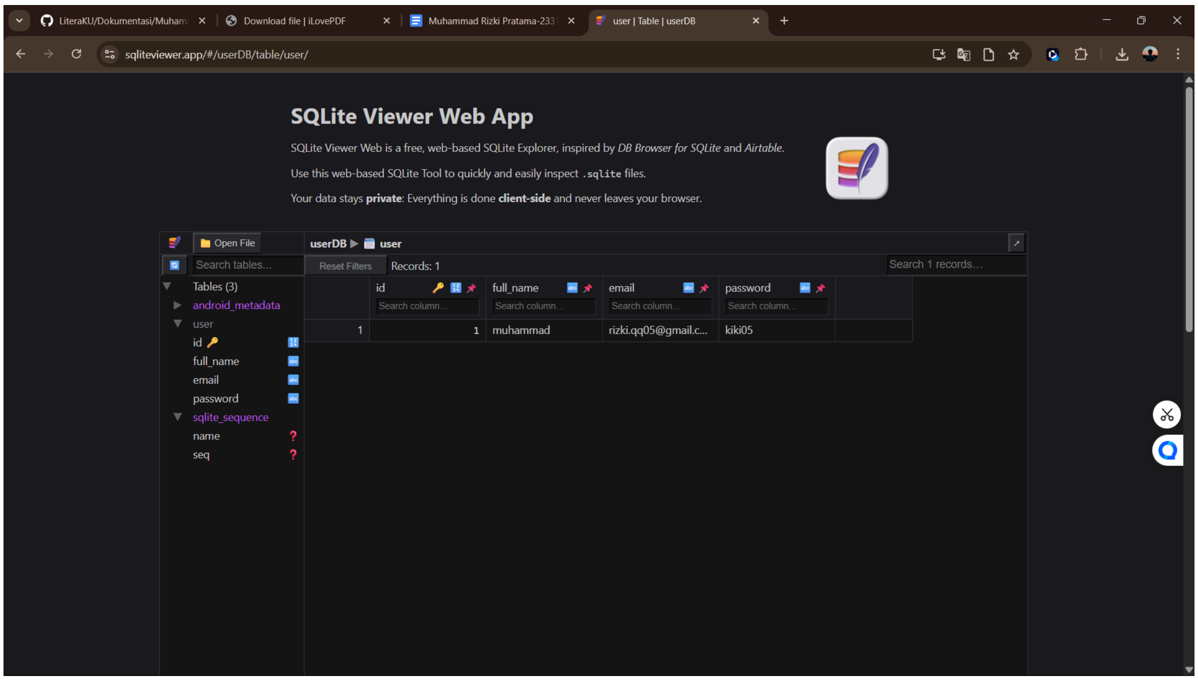
5. seperti ini tampilan halaman jika kita memasukan email,dan password maka akan masuk ketampilan utama



6.ini tampilan menu daftar/signup jika kita memasukan data nama,email,password maka akan data tersebut akan masuk ke data sqlite kita

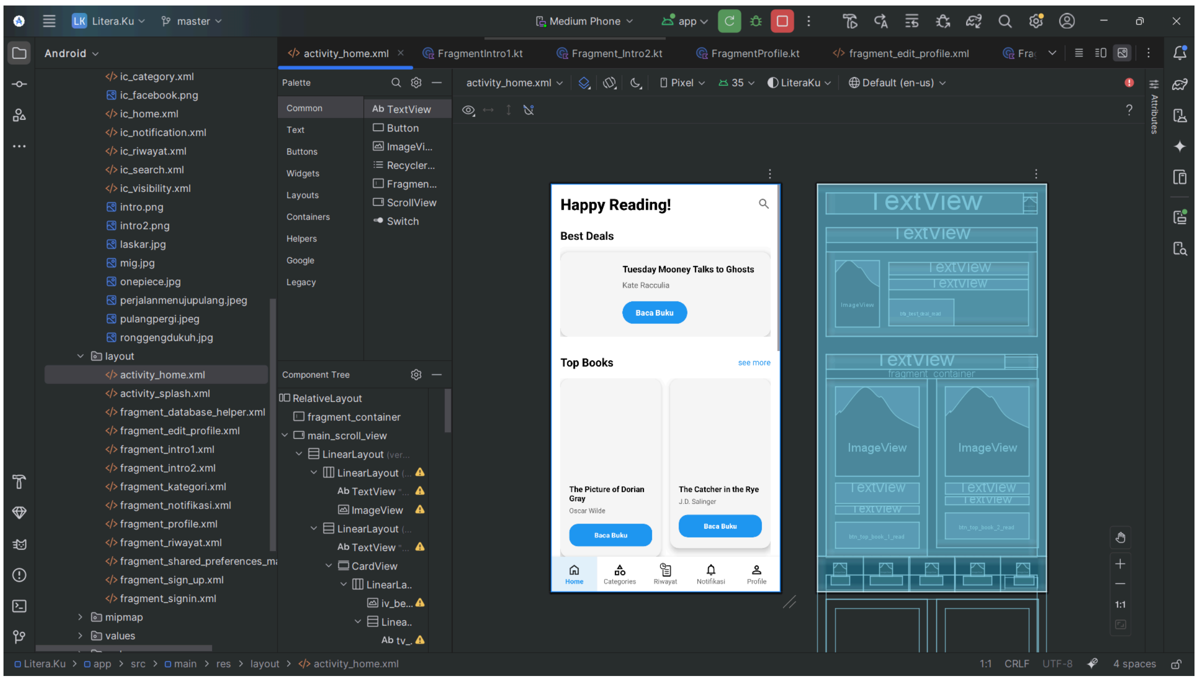


7.ini hasil tampilan isi database yang menampung data login akun kita

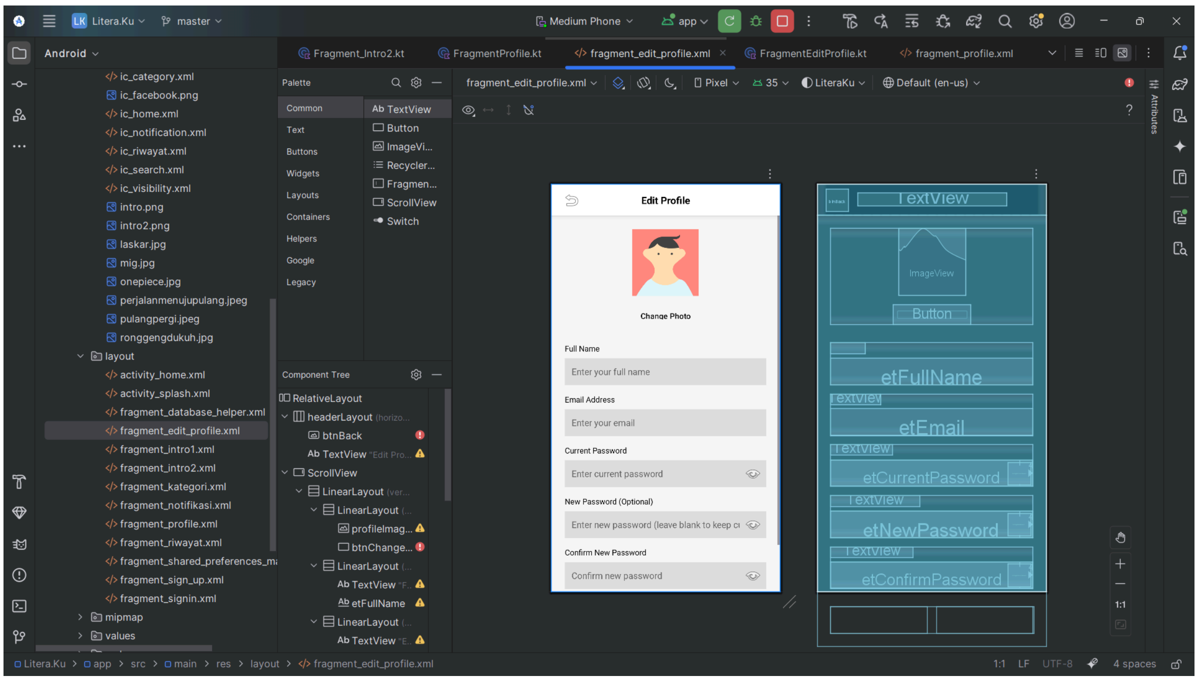

sqliteviewer.app/#/userDB/table/user/
SQLite Viewer Web App
SQLite Viewer Web is a free, web-based SQLite Explorer, inspired by DB Browser for SQLite and Airtable.
Use this web-based SQLite Tool to quickly and easily inspect .sqlite files.
Your data stays private: Everything is done client-side and never leaves your browser.
Open File
userDB ▶ user
Tables (3)
android_metadata
user
sqlite_sequence
Records: 1

| id | full_name | email | password |
|---|---|---|---|
| 1 | muhammad | rizki.qq05@gmail.c... | kiki05 |

8. ini tampilan awal saat kita sudah login masuk pertama kali dan ini masih polos dikarenakan blum dimasukan data buku agar kita bisa membacany dan ini kalo kita geser kebawah itu sudah bisa



9. saya membuat layout untuk bagian edit profile serta membuat fungsi agar kita bisa mengupdate data pribadi kita


Litera.Ku
master
Medium Phone
app
Android
Fragment_Intro2.kt
FragmentProfile.kt
fragment_edit_profile.xml
FragmentEditProfile.kt
fragment_profile.xml
Component Tree
RelativeLayout
headerLayout
btnBack
TextView
ScrollView
LinearLayout
profileImage
btnChangePhoto
etFullName
Edit Profile
Change Photo
Full Name
Enter your full name
Email Address
Enter your email
Current Password
Enter current password
New Password (Optional)
Enter new password (leave blank to keep current)
Confirm New Password
Confirm new password
etFullName
etEmail
etCurrentPassword
etNewPassword
etConfirmPassword
Litera.Ku > app > src > main > res > layout > fragment_edit_profile.xml