Owl-M: A Material Design Study App

Abstract:

Materio is an innovative study app designed to help designers,developers and enthusiasts master google's material design principles.this interactive platform provides an in-depth exploration of material design concepts,components,and best practices.

Key features:

- Interactive lessons:engaging tutorials and exercises covering material design fundamental, layout,typography,color,and more.
- Offline access:learn anywhere,anytime,without internet connectivity.

Introduction:

Material is a design system created by google to help teams build high-quality digital experiences for android,ios,flutter,and the web.Master the art of google's material design principle and create stunning user –friendly interfaces.Materio is your comprehensive guide to learning and implementing material design,featuring interactive lessons,real-world examples,and a community forum.

Project Description:

A project that demonstrates the use of Android Jetpack Compose to build a UI for a Owl-M: a material design study app. Owl-M app is a sample project built using the Android Compose UI toolkit. A Compose implementation of the Owl Material study.

System Requirements:

Operating System: Android-7 or later

Ram:1Gb (Minimum)

Rom:20Mb (Minimum)

Internet connection

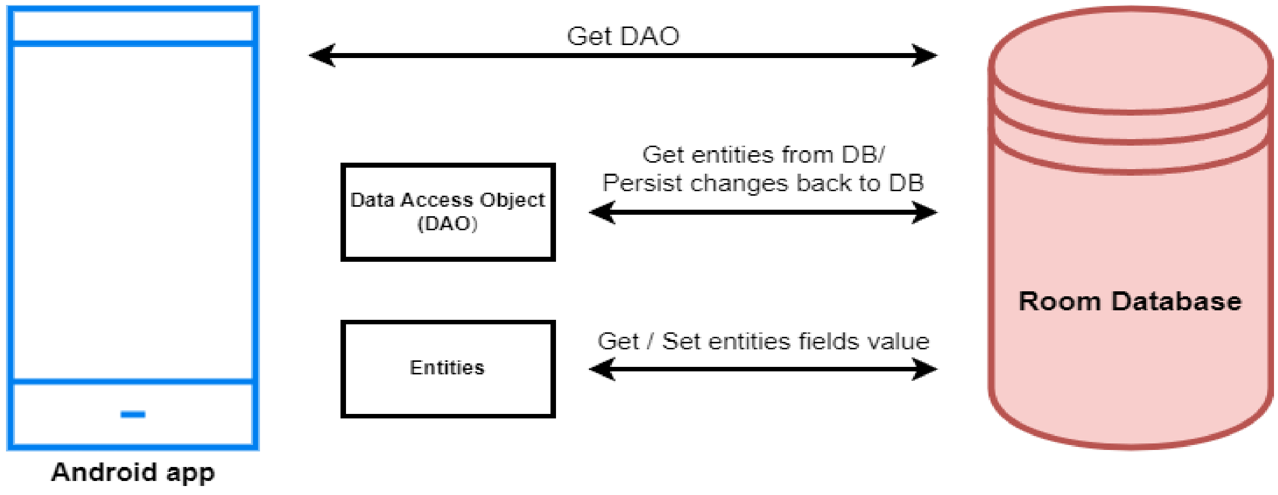
Tools Used:

Android Studio

Firebase

Kotlin

Architecture


Android app
Get DAO
Data Access Object (DAO)
Get entities from DB/ Persist changes back to DB
Entities
Get / Set entities fields value
Room Database

Project Workflow:

- Users register into the application.
- After registration , user logins into the application.
- User enters into the main page
- User can view the subject themes on selecting theme he can read about it.

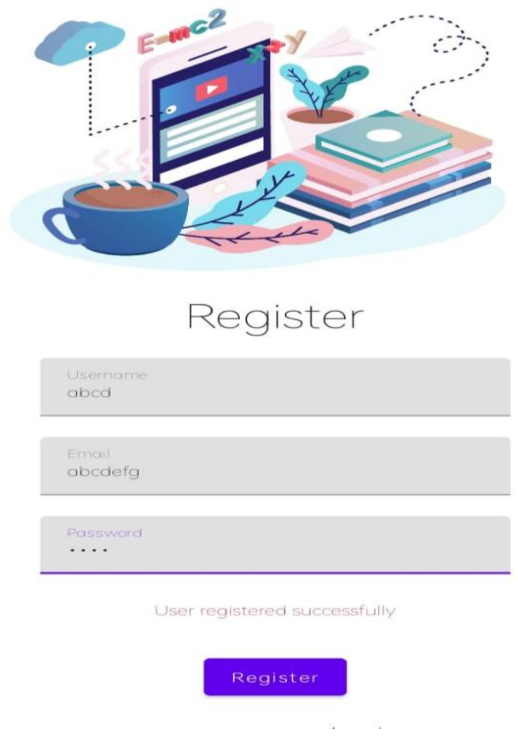
Tasks:

1.Required initial steps
2.Creating a new project.
3.Adding required dependencies.
4.Creating the database classes.
5.Building application UI and connecting to database.
6.Using AndroidManifest.xml
7.Running the application.

Program:

```
package com.example.owlapplication

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

@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,

    )
package com.example.owlapplication

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
package com.example.owlapplication

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

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

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "$COLUMN_FIRST_NAME TEXT, " +
            "$COLUMN_LAST_NAME TEXT, " +
            "$COLUMN_EMAIL TEXT, " +
            "$COLUMN_PASSWORD TEXT" +
            ")"

        db?.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }

    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }
}
```

Output



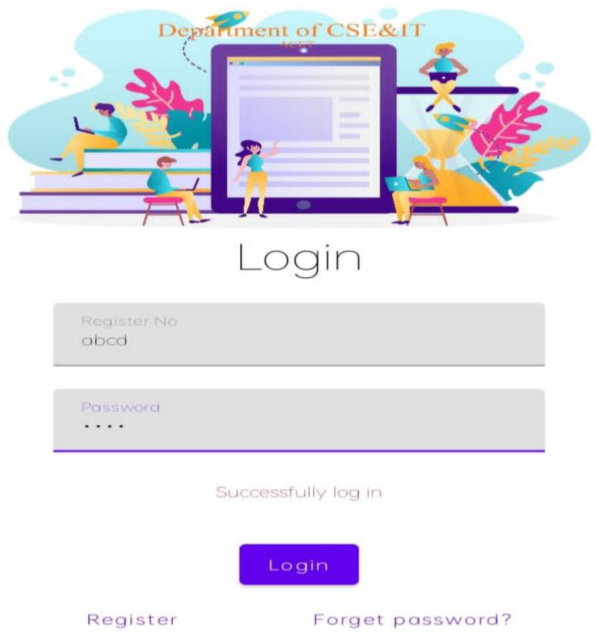
Register

Username
abcd

Email
abcdefg

Password
····

User registered successfully

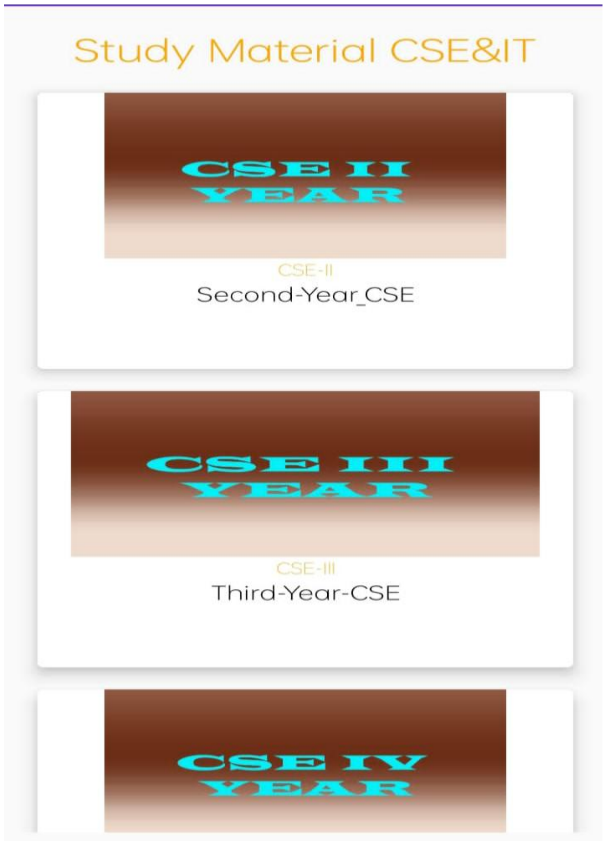


Login

Register No
abcd

Password
····

Successfully log in



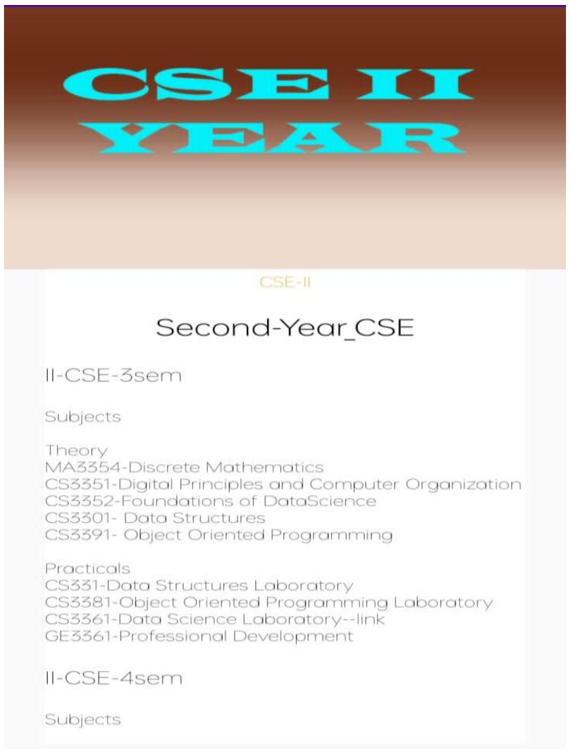
Study Material CSE&IT

CSE-II

Second-Year_CSE

CSE-III

Third-Year-CSE



CSE-II

Second-Year_CSE

II-CSE-3sem

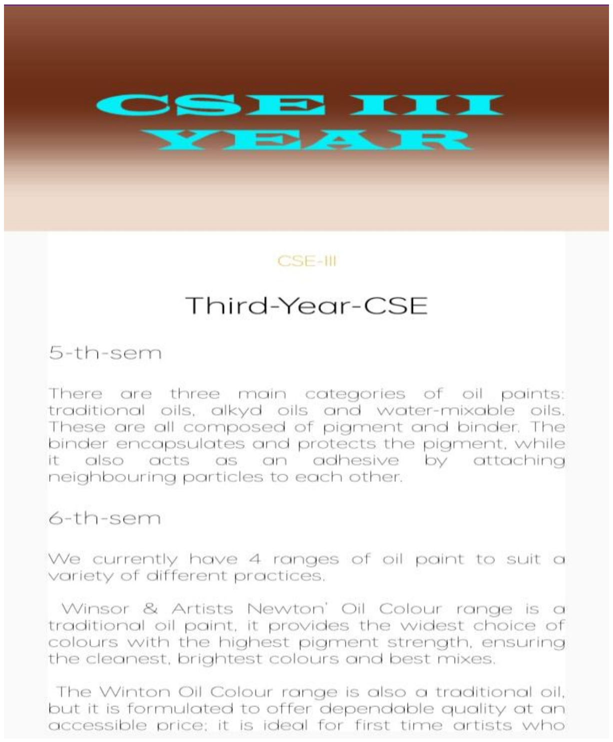
Subjects

Theory
MA3354-Discrete Mathematics
CS3351-Digital Principles and Computer Organization
CS3352-Foundations of DataScience
CS3301- Data Structures
CS3391- Object Oriented Programming

Practicals
CS331-Data Structures Laboratory
CS3381-Object Oriented Programming Laboratory
CS3361-Data Science Laboratory--link
GE3361-Professional Development

II-CSE-4sem

Subjects


CSE III YEAR

CSE-III

Third-Year-CSE

5-th-sem

There are three main categories of oil paints: traditional oils, alkyd oils and water-mixable oils. These are all composed of pigment and binder. The binder encapsulates and protects the pigment, while it also acts as an adhesive by attaching neighbouring particles to each other.

6-th-sem

We currently have 4 ranges of oil paint to suit a variety of different practices.

Winsor & Artists Newton' Oil Colour range is a traditional oil paint, it provides the widest choice of colours with the highest pigment strength, ensuring the cleanest, brightest colours and best mixes.

The Winton Oil Colour range is also a traditional oil, but it is formulated to offer dependable quality at an accessible price; it is ideal for first time artists who

Conclusion:

The Material Design Study App will serve as a valuable tool for developers and designers looking to master Material Design principles and best practices for Android development. By offering interactive tutorials, demos, and code examples, the app will provide hands-on learning opportunities that will help users implement beautiful and functional UIs in their Android apps. With a focus on user experience, smooth animations, and responsive design, the app will also serve as a showcase of Material Design's capabilities, encouraging users to embrace this design language in their own projects.

Future Scope for Material Design Study App:

The **Material Design Study App** has the potential to evolve and expand, incorporating new features, technologies, and educational content to stay current with the latest developments in Android UI/UX design. Below are some possible future directions for the app, focusing on evolving trends in mobile development, , user interaction, and learning experiences.