OWL-M: A MATERIAL DESIGN-STUDY APPLICATION BUILD WITH THE USE OF ANDROID JETPACK COMPOSE USER INTERFACE

lV B. TECH (Computer Science & Engineering)

Submitted by:

20AP1A0507
21AP5A0502
20AP1A0549
20AP1A0551

Under the supervision of
G.S.V.R. ABHISHEK

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BHIMAVARAM INSTITUTE OF ENGINEERING & TECHNOLOGY

PENNADA, BHIMAVARAM-534243, ANDHRA PREDESH.

BHIMAVARAM INSTITUTE OF

ENGINEERING & TECHNOLOGY

PENNADA, BHIMAVARAM-534243, ANDHRA PREDESH.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled **"OWL-M: A MATERIAL DESIGN-STUDY APPLICATION BUILD WITH THE USE OF ANDROID JETPACK COMPOSE USER INTERFACE"** that is being submitted by BHIMIREDDY CHAITANYA KUMAR REDDY (20AP1A0507) in partial fulfilment for the award of **Bachelor of Technology in Computer Science and Engineering to the Bhimavaram Institute of Engineering and Technology, Jawaharlal Nehru Technological Kakinada** is a record of bonafide work carried out by them under my guidance and supervision.

The results embedded in this dissertation have not been submitted to any other university/ institute for the award of any degree/diploma.

PROJECT SUPERVISOR	HEAD OF DEPARTMENT
G.S.V.R. ABHISHEK	U.V.S.VINOD
Assistant Professor	Professor & HOD
Department of CSE	Department of CSE

Acknowledgement

I would like to deliver my special gratitude to my class teacher and our project Co- Ordinator **G.S.V.R. ABHISHEK sir** Assistant Professor of CSE Department for their valuable suggestions and constant motivation that greatly helped us in successful completion of the project and our principal **DR.K.SURESH sir** Principal ,Bhimavaram Institute of Engineering & Technology-Pennada for his support and constant motivation in successful completion of the project and our HOD **sir U.V.S.VINOD, A**ssistant Professor & HOD, Department of CSE for showing their interest in my topic of research and granting mean incredible chance to do this project on the topic of Building a blog App .

Finally, I would thank my classmates who have helped me a lot.

By:

1.B CHAITANYA KUMAR REDDY (20AP1A0507)

2.M KOWSALYA LAKSHMI (21AP5A0502)

3.V SAIRAMI REDDY (20AP1A0549)

4.Y JAYADEEP (20AP1A0551)

DECLARATION

We have by declare that project report entitled "**OWL-M: A MATERIAL DESIGN-STUDY APPLICATION BUILD WITH THE USE OF ANDROID JETPACK COMPOSE USER INTERFACE** "is genuine project work carried out by us in the Bachelor of Degree COMPUTER SCIENCE AND ENGINEERING Degree course of JAWAHARLALA NEHRU TECHNOLOGY UNIVERSITY, KAKINADA and has not been submitted to any other courses (or) university for award of degree of us.

By:

1.B CHAITANYA KUMAR REDDY (20AP1A0507)

2.M KOWSALYA LAKSHMI (21AP5A0502)

3.V SAIRAMI REDDY (20AP1A0549)

4.Y JAYADEEP (20AP1A0551)

Date:

Place:

PROJECT REPORT ON:

OWL-M: A MATERIAL DESIGN-STUDY APPLICATION BUILD WITH THE USE OF ANDROID JETPACK COMPOSE USERINTERFACE

TEAM ID

LTVIP2023TMID04886

TEAM LEADER

BHIMIREDDY CHAITANYA KUMAR REDDY

TEAM MEMBERS

MANCHIGANTI KOWSALYA LAKSHMI

VATRAPU SAIRAMI REDDY

YERRAM JAYADEEP

TABLE OF CONTENTS

ABSTRACT

KOTLIN. Here I will explain about the app has been designed with simplicity and user-friendliness This PROJECT is all about MATERIAL DESIGN STUDY ANDROID APPLICATION using in mind, ensuring that users can easily read the study materials. If you are looking to read various topics in a single app, Owl-M: A Material Design Study App is the perfect application to do it.

1. INTRODUCTION

Introducing a prototype project that demonstrates the use of Android Jetpack Compose to build a UI (User Interface) for Owl-M: a Material design study application. OwlM app is a sample project built using the Android Compose UI toolkit with the help of KOTLIN language.

With the help of this Application, we can read any material for study purpose. It will be easy to read the information provided in this application, because it gives a clear and cut out study material with perfect Alignment, Headings, Subheadings, and finally the descriptive information.

1.1 Overview

The Owl-M: A Material Design Study App is a real-time reading application developed using Kotlin programming language and Jetpack Compose UI toolkit. The app allows users to sign up(register) and log in using their Username and password, making it quick and easy to use. Once logged in, users can start reading the information that is arranged in there as a separate read file.

The app has been designed with simplicity and user-friendliness in mind, ensuring that users can easily read the study materials. If you are looking to read various topics in a single app, Owl-M: A Material Design Study App is the perfect application to do it.

Overall, the Owl-M: A Material Design Study App is a reliable and easy-to-use studying application that allows users to read various topics at any time whenever and wherever you want. With its user-friendly interface and robust features, this app is an excellent choice for anyone who is looking for a studying app.

1.2 Purpose

The main purpose of this Owl-M: A Material Design Study App is to provide users with a reliable and easy-to use reading platform for study purpose. In today's fast- paced world, people are often busy with their phones and find it challenging to concentrate on studying because of its distraction. The Owl-M: A Material Design Study App aims to build a bridge to this gap by providing a platform where users can easily study with their Android mobile phones from anywhere and whenever they have the time.

1.3 Introduction to Kotlin

Kotlin took inspiration from many programming languages, including (but not limited to) Java, Scala, C# and Groovy. One of the main ideas behind Kotlin is being *pragmatic*, i.e., being a programming language useful for day-to-day development, which helps the users get the job done via its features and its tools. Thus, a lot of design decisions were and still are influenced by how beneficial these decisions are for Kotlin users.

Kotlin is a multiplatform, statically typed, general-purpose programming language. Currently, as of version 1.91.9, it supports compilation to the following platforms.

- JVM (Java Virtual Machine)
- JS (JavaScript)
- Native (native binaries for various architectures)

Furthermore, it supports transparent interoperability between different platforms via its Kotlin Multiplatform Project (Kotlin MPP) feature.

The type system of Kotlin distinguishes at compile time between nullable and non-nullable types, achieving null-safety, i.e., guaranteeing the absence of runtime errors caused by the absence of value (i.e., null value). Kotlin also extends its static type system with elements of gradual and flow typing, for better interoperability with other languages and ease of development.

Kotlin is an object-oriented language which also has a lot of functional programming elements. From the object-oriented side, it supports nominal subtyping with bounded parametric polymorphism (akin to generics) and mixed-site variance. From the functional programming side, it has first-class support for higher-order functions and lambda literals.

This specification covers Kotlin/Core, i.e., fundamental parts of Kotlin which should function *mostly* the same way regardless of the underlying platform. These parts include such important things as language expressions, declarations, type system and overload resolution.

Important: due to the complexities of platform-specific implementations, platforms may extend, reduce or change the way some aspects of Kotlin/Core function. We mark these platform-dependent Kotlin/Core fragments in the specification to the best of our abilities.

Platform-specific parts of Kotlin and its multiplatform capabilities will be covered in their respective sub-specifications, i.e., Kotlin/JVM, Kotlin/JS and Kotlin/Native.

Compatibility

Kotlin Language Specification is still in progress and has **experimental** stability level, meaning no compatibility should be expected between even incremental releases, any functionality can be added, removed, or changed without warning.

Experimental features

In several cases this specification discusses *experimental* Kotlin features, i.e., features which are still in active development and which may be changed in the future. When so, the specification talks about the *current* state of said features, with no guarantees of their future stability (or even existence in the language).

The experimental features are marked as such in the specification to the best of our abilities.

Acknowledgments

We would like to thank the following people for their invaluable help and feedback during the writing of this specification.

Note: the format is “First name Last name”, ordered by last name

4 Zalim Basharov
5 Andrey Breslav
6 Roman Elizarov
7 Stanislav Erokhin
8 Neal Gafter
9 Dmitrii Petrov
10 Victor Petukhov
11 Dmitry Savvinov
12 Anastasiia Spaseeva
13 Mikhail Zarechenskii
14 Denis Zharkov

Why Use Kotlin?

- Kotlin is fully compatible with Java
- Kotlin works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- Kotlin is concise and safe
- Kotlin is easy to learn, especially if you already know Java
- Kotlin is free to use
- Big community/support

Benefits of Kotlin

Some of the advantages of Kotlin are the following:

- **Interoperability.** Kotlin interoperates with Java because they compile to the same byte code. Kotlin can be compiled into JavaScript or an LLVM encoder, which enables programmers to perform just-in-time compiling to ensure that code embedded in another program runs smoothly. It also shares tooling with Java. These features make it easy to migrate Java applications to Kotlin.
- **Safety.** Kotlin is designed to help avoid common coding errors that can break code or leave vulnerabilities in it. The language features null safety and eliminating null pointer exception errors.
- **Clarity.** Kotlin eliminates some of the redundancy in the basic syntax of popular languages like Java. Kotlin is a timesaver for developers because it provides more

concise code. Developers can write programs with less boilerplate code, increasing their productivity.

- **Tooling support.** Kotlin has tooling support from Android with tools optimized for Android development, including Android Studio, Android KTX and Android SDK.
- **Community support.** Although Kotlin is a relatively new language compared to Java, it has a community of developers who work to improve the language and provide documentation

2.REQUIREMENTS

2.1 Software

We use a software called Android Studio for this project with the version Hedgehog 8.2.

Android Studio is the official Integrated Development Environment (IDE) for Android app development. Based on the powerful code editor and developer tools from IntelliJ IDEA , Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Live Edit to update composables in emulators and physical devices in real time
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Each project in Android Studio contains one or more modules with source code files and resource files. The types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level, under **Gradle Scripts**.

Each app module contains the following folders:

- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Kotlin and Java source code files, including JUnit test code.
- **res**: Contains all non-code resources such as UI strings and bitmap images.

Android Studio helps you debug and improve the performance of your code, including inline debugging and performance analysis tools.

Inline debugging

Use inline debugging to enhance your code walkthroughs in the debugger view with inline verification of references, expressions, and variable values.

Inline debug information includes:

- Inline variable values
- Objects that reference a selected object
- Method return values
- Lambda and operator expressions
- Tooltip values

To enable inline debugging, in the **Debug** window, click **Settings** and select **Show Variable Values in Editor**.

Performance profilers

Android Studio provides performance profilers so you can easily track your app's memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyze network requests.

To use performance profilers, with your app running on a device or emulator, open the Android Profiler by selecting **View > Tool Windows > Profiler**.

For more information about performance profilers, see Profile your app performance.

Heap dump

When profiling memory usage in Android Studio, you can simultaneously initiate garbage collection and dump the Java heap to a heap snapshot in an Android-specific HPROF binary format file. The HPROF viewer displays classes, instances of each class, and a reference tree to help you track memory usage and find memory leaks.

For more information about working with heap dumps, see Capture a heap dump.

Memory Profiler

Use Memory Profiler to track memory allocation and watch where objects are being allocated when you perform certain actions. These allocations help you optimize your app's performance and memory use by adjusting the method calls related to those actions.

For information about tracking and analyzing allocations, see View memory allocations.

Data file access

The Android SDK tools, such as Systrace and Logcat, generate performance and debugging data for detailed app analysis.

To view the available generated data files:

1. Open the Captures tool window.
2. In the list of the generated files, double-click a file to view the data.
3. Right-click any HPROF files to convert them to the standard.
4. Investigate your RAM usage file format.

2.2 Hardware

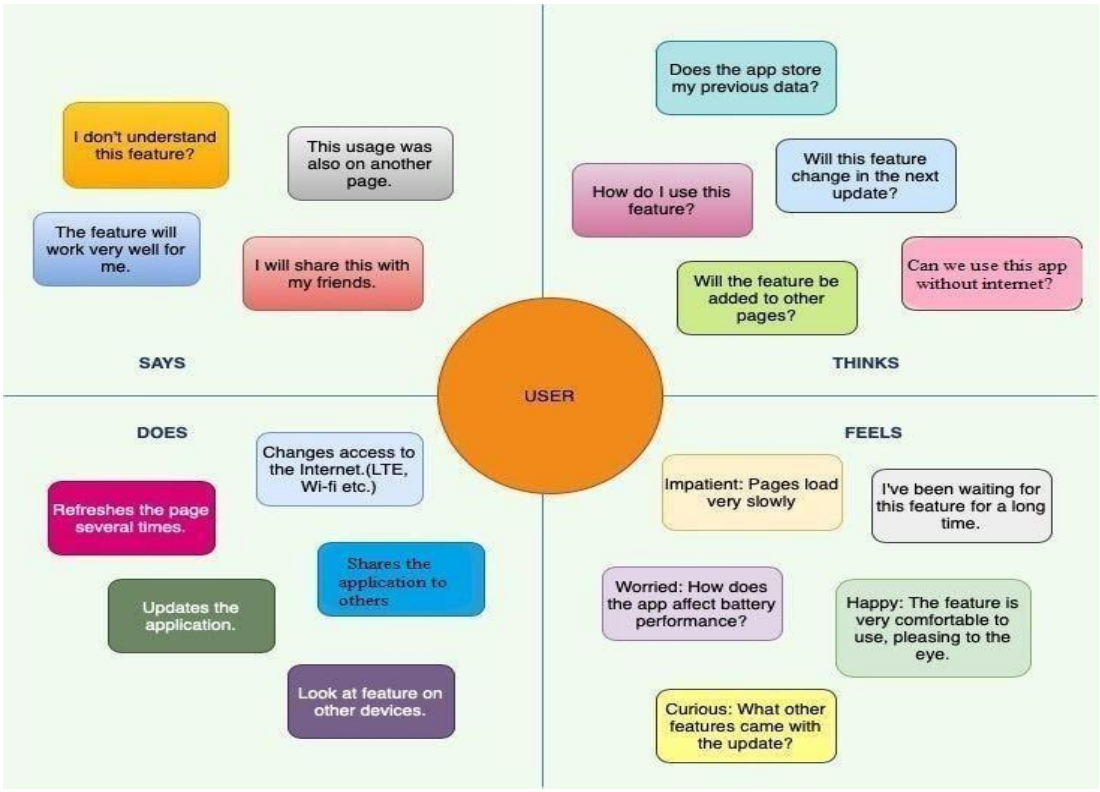
The following are the Hardware requirements for Android Studio on Windows.

- 64-bit Microsoft® Windows® 8/10/11
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator) •

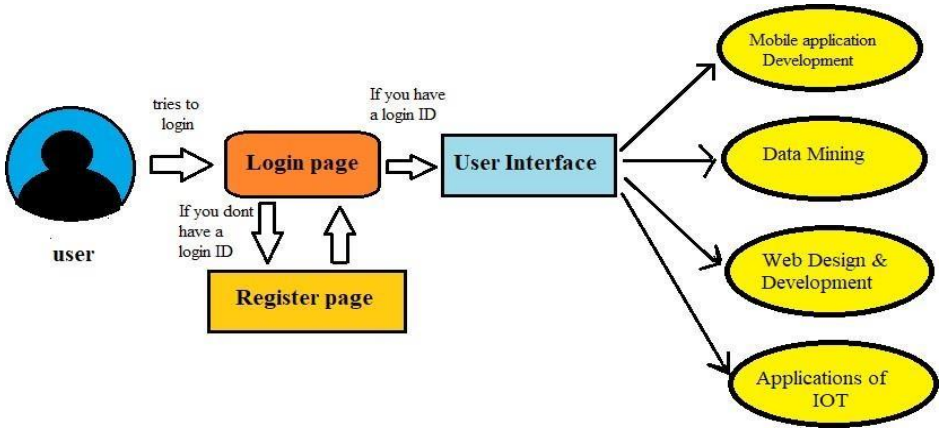
  1280 x 800 minimum screen resolution

3. PROBLEM DEFINITION & DESIGN THINKING

3.1 Empathy map



3.2 Ideation & Brainstorming Map



3.3 Login and Register page Use Case

4 ADVANTAGES & DISADVANTAGES AND APPLICATIONS

4.1 Advantages:

We can use the application to learn easily without anyone's help. We can study the content of this app via an android mobile phone from anywhere and at any time.

Mobile devices are frequently situated and owned by the same person; they make the educational process continuous. Unlike traditional teaching methods, the students may complete work at any time that is convenient for them, and teachers can shift the passive share of instruction beyond the classroom.

It is quite easy for individuals to gain access to whatever they want. The content will be arranged in a well-defined manner to make the user to study easily.

This app is a cost-effective application because in the previous decade, many of us were using Books for referring and reading. The Books we have used had cost us in their demand full way. But by using this app we can refer or read the content without any cost.

4.2 Disadvantages:

we can get the content at free of cost but we should have a android mobile phone with internet connection which will be an expense to us.

Unexpected software and Hardware issues can lead to a destruction of the content present inside the application which will lead us to a major problem.

There would not be a physical interaction between two or more people because it is a study app. If you are someone who believes in personal interaction then such apps are not for You

4.3 APPLICATIONS

Identify the target audience: Determine who the app is for, whether it's for students, professionals, or a particular age group.

Define the purpose: Determine the specific purpose of the app, such as helping users to learn a particular subject or improve their study habits.

Develop the app features: Based on the identified target audience and purpose, develop features that will help users achieve their goals. These could include features such as flashcards, quizzes, progress tracking, study reminders, and more.

Design the app interface: Develop a user-friendly and visually appealing interface that will make it easy for users to navigate and engage with the app.

Test and refine: Test the app with a small group of users and gather feedback to refine the app and improve its functionality.

4.4 FUTURE SCOPE

The **Owl-M: a Material design study application** can have a wide range of future applications, some of which are:

Personalized Learning: The application can be used to provide personalized learning experiences to students. The software can analyze the student's learning style and provide customized learning materials, activities, and assessments.

Gamification: The application can be used to gamify learning and make it more engaging for students. The software can incorporate game elements such as points, badges, and leaderboards to motivate students to learn.

Artificial Intelligence: The application can incorporate artificial intelligence (AI) to provide intelligent tutoring systems. The software can use machine learning algorithms to analyze student data and provide personalized recommendations for learning.

Collaboration: The application can enable collaboration between students and teachers. The software can provide tools for online discussions, group projects, and peer-to-peer feedback.

Adaptive Assessments: The application can provide adaptive assessments that adjust to the student's level of understanding. The software can analyze the student's responses and provide questions of appropriate difficulty level.

Virtual Reality: The application can use virtual reality (VR) to provide immersive learning experiences. The software can create virtual environments that simulate real-world scenarios and enable students to learn by doing.

Overall, a study application has the potential to revolutionize education by providing personalized, engaging, and adaptive learning experiences to students.

5. IMPLEMENTATION

5.1 CODE LOGIN PAGE CODE

```
package com.example.owlapplication

import android.content.Context import
android.content.Intent import
android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.material.* import
androidx.compose.runtime.* import
androidx.compose.ui.Alignment import
androidx.compose.ui.Modifier import
androidx.compose.ui.graphics.Color import
androidx.compose.ui.layout.ContentScale import
androidx.compose.ui.res.painterResource import
androidx.compose.ui.text.font.FontFamily import
androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview import
androidx.compose.ui.unit.dp import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.owlapplication.ui.theme.OwlApplicationTheme

class LoginActivity : ComponentActivity() {     private
lateinit var databaseHelper: UserDatabaseHelper     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)        databaseHelper =
UserDatabaseHelper(this)        setContent {
          LoginScreen(this, databaseHelper)
      }
   }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

   var username by remember { mutableStateOf("") }
var password by remember { mutableStateOf("") }     var error
by remember { mutableStateOf("") }

   Column(        modifier =
Modifier.fillMaxSize().background(Color.White),
```

```
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.study_login), contentDescription = "")

        Text(             fontSize = 36.sp,
fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
            text = "Login"
        )
        Spacer(modifier = Modifier.height(10.dp))

        TextField(             value = username,
onValueChange = { username = it },
label = { Text("Username") },             modifier
= Modifier.padding(10.dp)
                .width(280.dp)
        )
        TextField(             value = password,
onValueChange = { password = it },
label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
modifier = Modifier.padding(10.dp)
                .width(280.dp)
        )

        if (error.isNotEmpty()) {             Text(
text = error,                 color =
MaterialTheme.colors.error,
modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
val user = databaseHelper.getUserByUsername(username)
if (user != null && user.password == password) {
                    error = "Successfully log in"
context.startActivity(                          Intent(
context,
                            MainActivity::class.java
                        )
                    )
                    //onLoginSuccess()
                }                   else {                          error =
"Invalid username or password"
```

```
                }

            } else {                    error =
"Please fill all fields"
            }
        },
        modifier = Modifier.padding(top = 16.dp)
    ) {
        Text(text = "Login")
    }
    Row {
        TextButton(onClick = {context.startActivity(
            Intent(
context,
                RegisterActivity::class.java
            )
        )}
        )
        { Text(text = "Register") }
        TextButton(onClick = {
        })

        {
            Spacer(modifier = Modifier.width(60.dp))
            Text(text = "Forget password?")
        }
    }
  }
}
private fun startMainPage(context: Context) {     val
intent = Intent(context, MainActivity::class.java)
   ContextCompat.startActivity(context, intent, null)
}
```

REGISTRATION PAGE CODE

```
package com.example.owlapplication

import android.content.Context import
android.content.Intent import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image import
androidx.compose.foundation.background import
androidx.compose.foundation.layout.* import
androidx.compose.material.* import
androidx.compose.runtime.* import
```

```
androidx.compose.ui.Alignment import
androidx.compose.ui.Modifier import
androidx.compose.ui.graphics.Color import
androidx.compose.ui.layout.ContentScale import
androidx.compose.ui.res.painterResource import
androidx.compose.ui.text.font.FontFamily import
androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview import
androidx.compose.ui.unit.dp import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.owlapplication.ui.theme.OwlApplicationTheme

class RegisterActivity : ComponentActivity() {     private
lateinit var databaseHelper: UserDatabaseHelper     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)         databaseHelper =
UserDatabaseHelper(this)         setContent {
            RegistrationScreen(this, databaseHelper)
        }
    }
}

@Composable  fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {

    var username by remember { mutableStateOf("") }
var password by remember { mutableStateOf("") }     var
email by remember { mutableStateOf("") }     var error
by remember { mutableStateOf("") }

    Column(         modifier =
Modifier.fillMaxSize().background(Color.White),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center
    ) {

        Image(painterResource(id = R.drawable.study_signup), contentDescription = "")

        Text(
            fontSize = 36.sp,
            fontWeight = FontWeight.ExtraBold,
fontFamily = FontFamily.Cursive,
            text = "Register"
        )
        Spacer(modifier = Modifier.height(10.dp))
TextField(             value = username,
onValueChange = { username = it },
```

```
label = { Text("Username") },            modifier
= Modifier
.padding(10.dp)
            .width(280.dp)

        )
        TextField(            value = email,
onValueChange = { email = it },
label = { Text("Email") },            modifier
= Modifier
.padding(10.dp)
            .width(280.dp)
        )
        TextField(            value = password,
onValueChange = { password = it },
label = { Text("Password") },
          visualTransformation = PasswordVisualTransformation(),
modifier = Modifier                .padding(10.dp)
            .width(280.dp)
        )


        if (error.isNotEmpty()) {            Text(
text = error,                color =
MaterialTheme.colors.error,
modifier = Modifier.padding(vertical = 16.dp)
          )
        }

        Button(
onClick = {
            if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty())
{                  val user = User(
id = null,                    firstName =
username,
lastName = null,                    email
= email,                    password = password
                )
                databaseHelper.insertUser(user)
error = "User registered successfully"                  // Start LoginActivity
using the current context
                context.startActivity(
Intent(                    context,
LoginActivity::class.java
                  )
                )
```

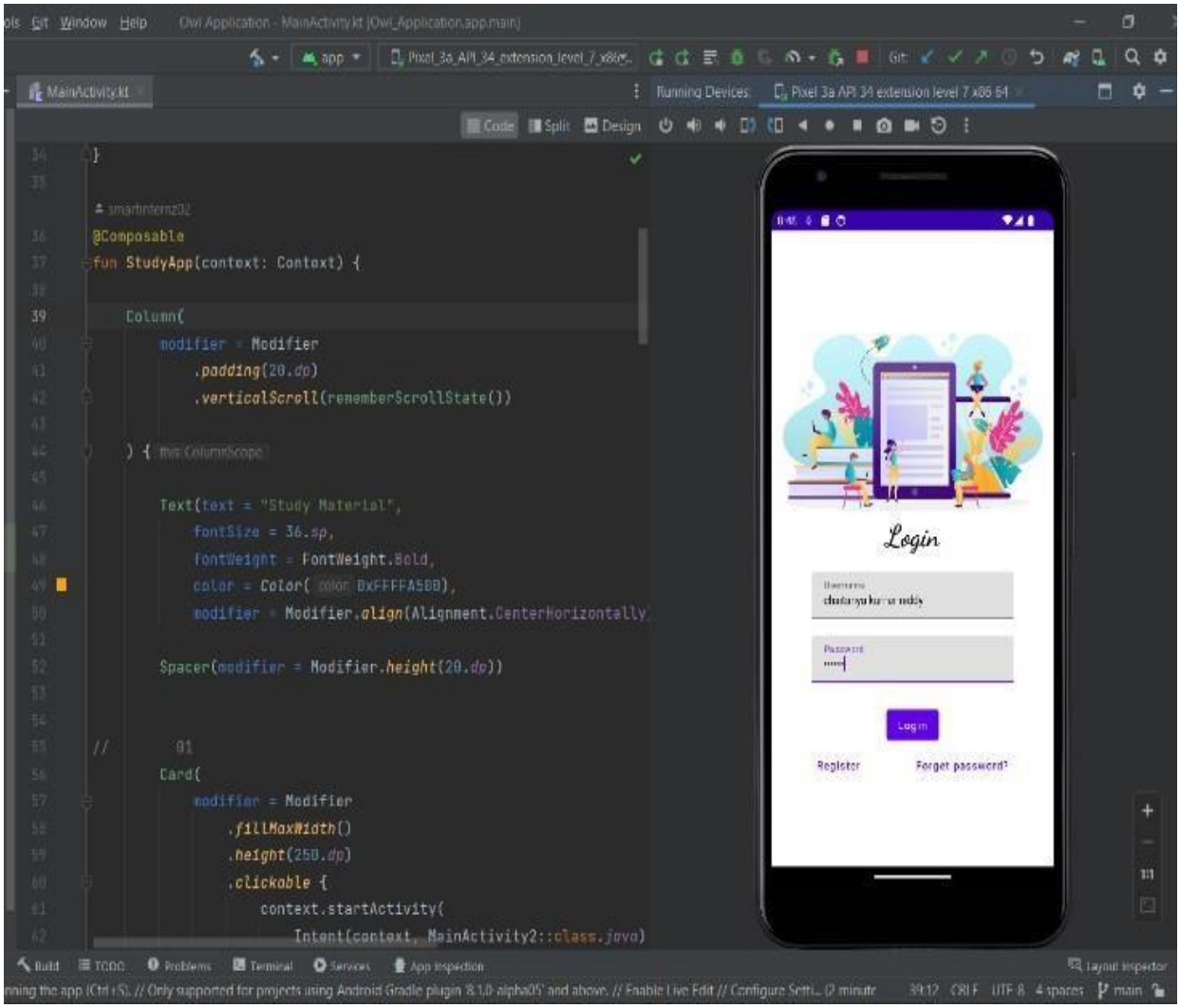
```
            } else {                    error =
"Please fill all fields"
            }
        },
        modifier = Modifier.padding(top = 16.dp)
    ) {
        Text(text = "Register")
    }
    Spacer(modifier = Modifier.width(10.dp))
Spacer(modifier = Modifier.height(10.dp))

    Row() {          Text(                modifier = Modifier.padding(top = 14.dp),
text = "Have an account?"
        )
        TextButton(onClick = {
context.startActivity(
Intent(                    context,
                LoginActivity::class.java
            )
        )
        })

        {
            Spacer(modifier = Modifier.width(10.dp))
            Text(text = "Log in")
        }
    }
  }
}
private fun startLoginActivity(context: Context) {    val
intent = Intent(context, LoginActivity::class.java)
  ContextCompat.startActivity(context, intent, null)
}
```

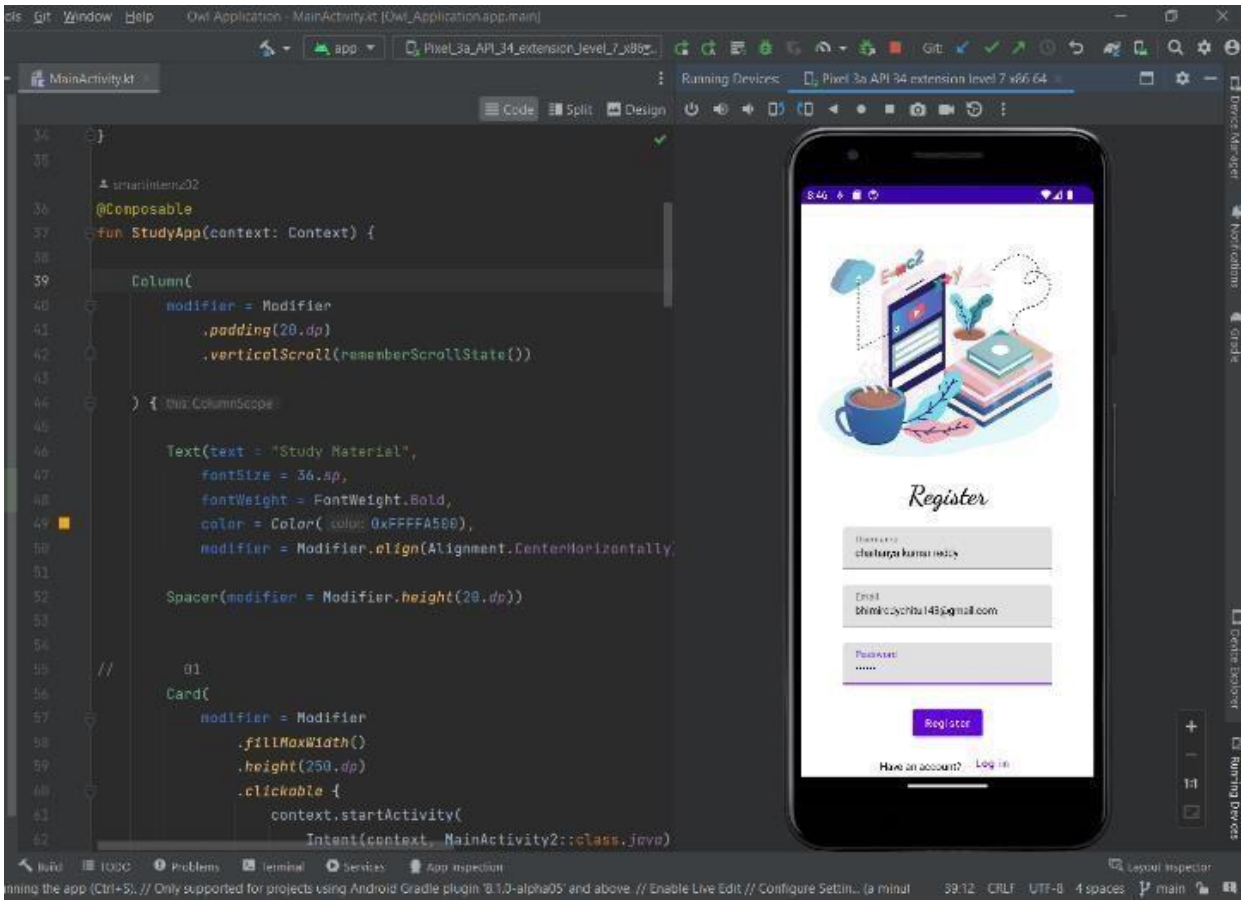
5.2 OUTCOME

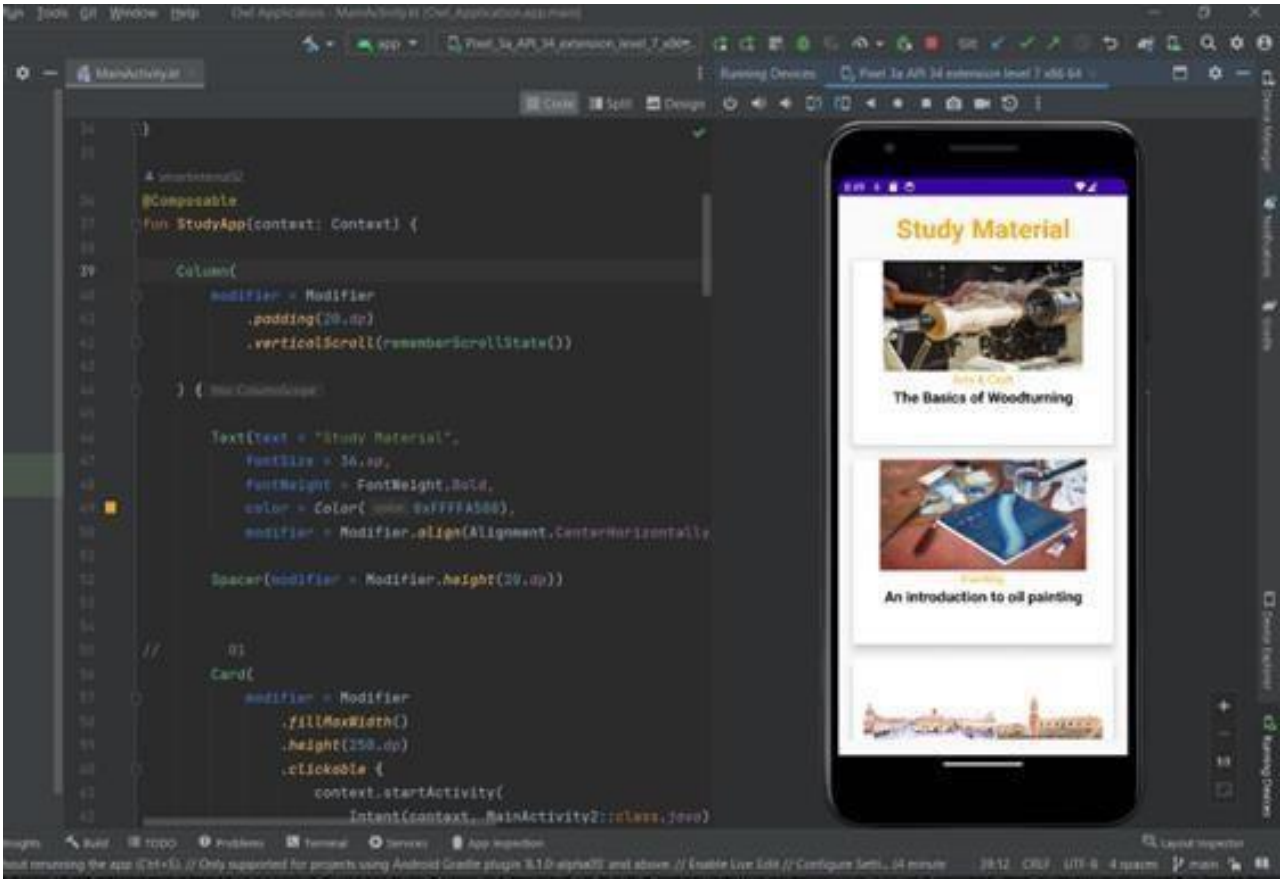
1) Here, the Login Screen appears on the window. If we have a login ID, we can directly login to our account. If not, we must Register as a new user and sign up to get a Login ID.

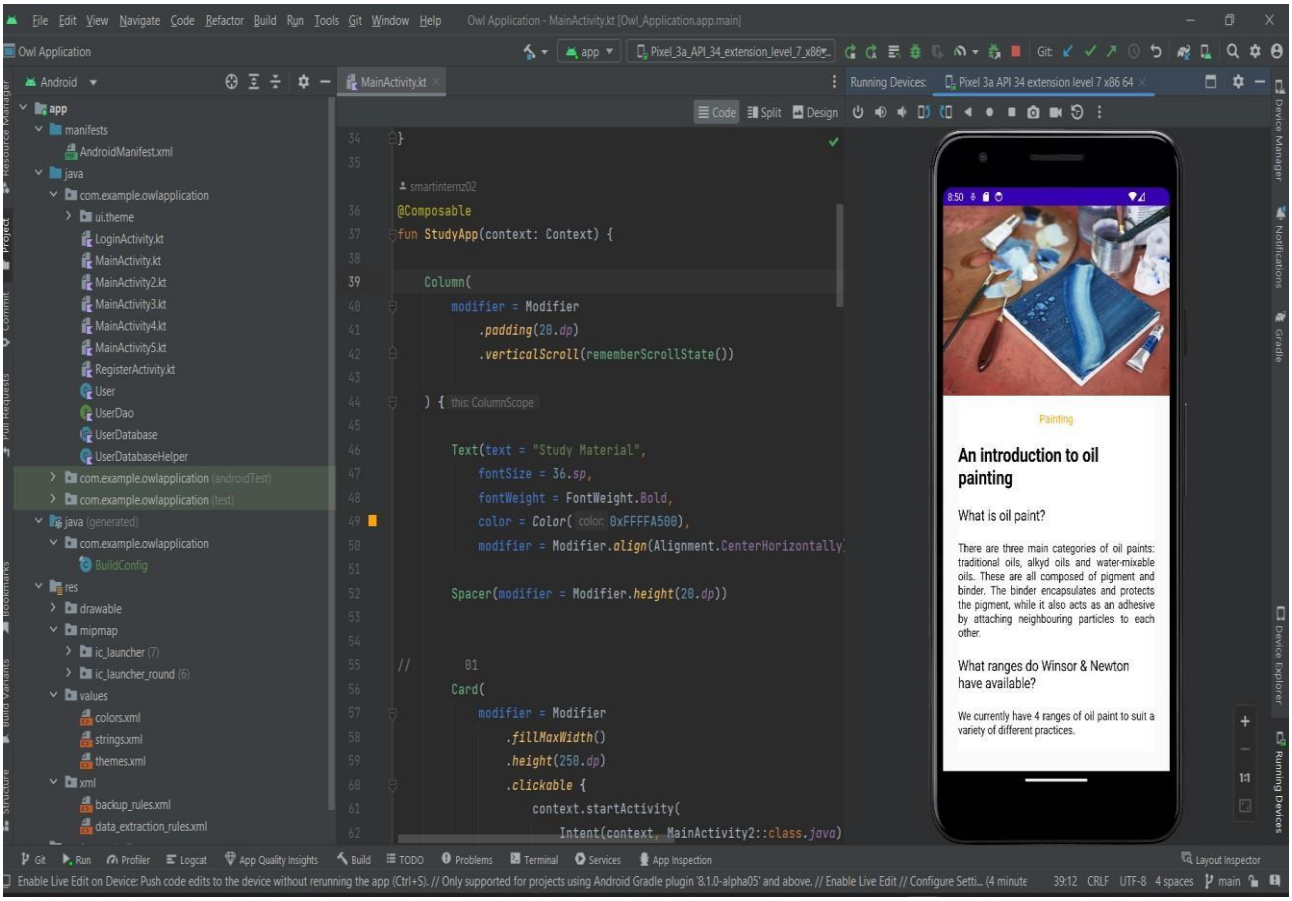
2) So, this is the Register window where we have given the certain details: Username, Email ID and Password. By doing this activity we can get a new Login ID.

3) After the Register, we can Login into the study app via the Login page by typing the Username and Password. Then, we can access the content according to our requirement.

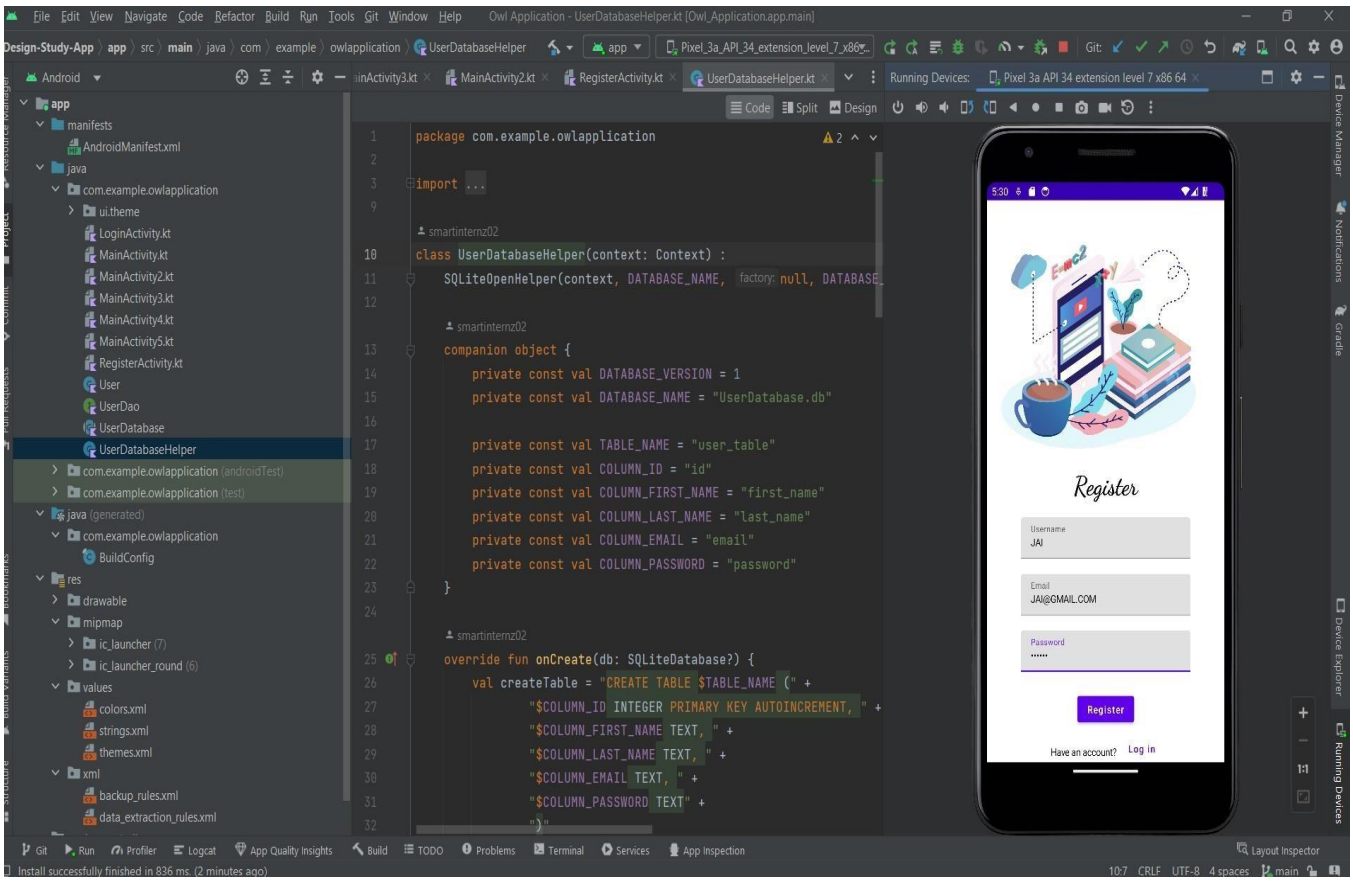

}
@Composable
fun StudyApp(context: Context) {
Column(
modifier = Modifier
.padding(20.dp)
.verticalScroll(rememberScrollState())
) {
Text(text = "Study Material",
fontSize = 36.sp,
fontWeight = FontWeight.Bold,
color = Color(color: 0xFFFFA500),
modifier = Modifier.align(Alignment.CenterHorizontally
Spacer(modifier = Modifier.height(20.dp))
// 01
Card(
modifier = Modifier
.fillMaxWidth()
.height(250.dp)
.clickable {
context.startActivity(
Intent(context, MainActivity2::class.java)
Login
Login
Register
Forget password?

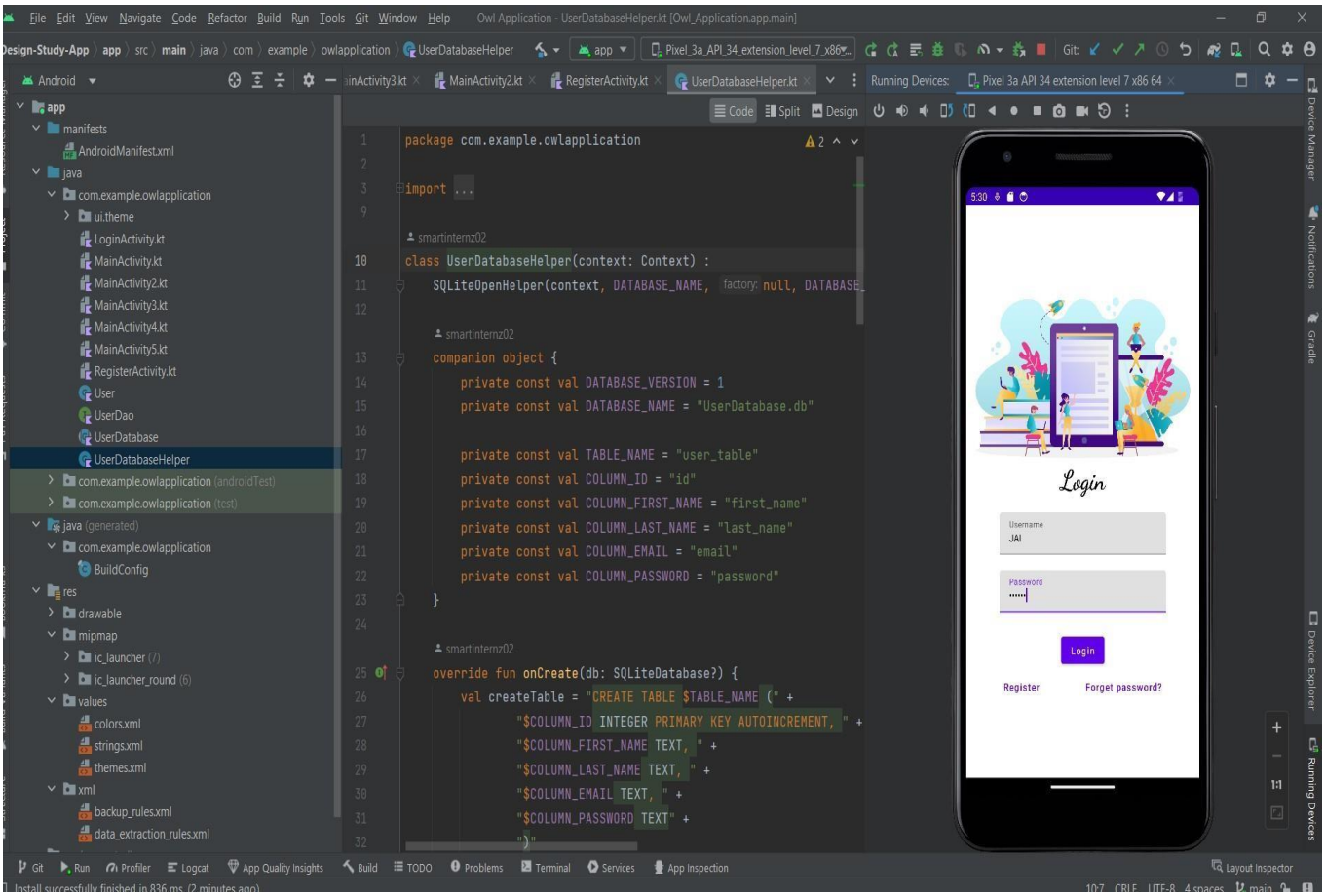

MainActivity.kt
Code
Split
Design
Running Devices
Pixel 3a API 34 extension level 7 x86 64
@Composable
fun StudyApp(context: Context) {
Column(
modifier = Modifier
.padding(20.dp)
.verticalScroll(rememberScrollState())
) {
Text(text = "Study Material",
fontSize = 36.sp,
fontWeight = FontWeight.Bold,
color = Color(
modifier = Modifier.align(Alignment.CenterHorizontally
Spacer(modifier = Modifier.height(20.dp))
// 01
Card(
modifier = Modifier
.fillMaxWidth()
.height(250.dp)
.clickable {
context.startActivity(
Intent(context, MainActivity2::class.java)
Register
Register
Have an account? Log in

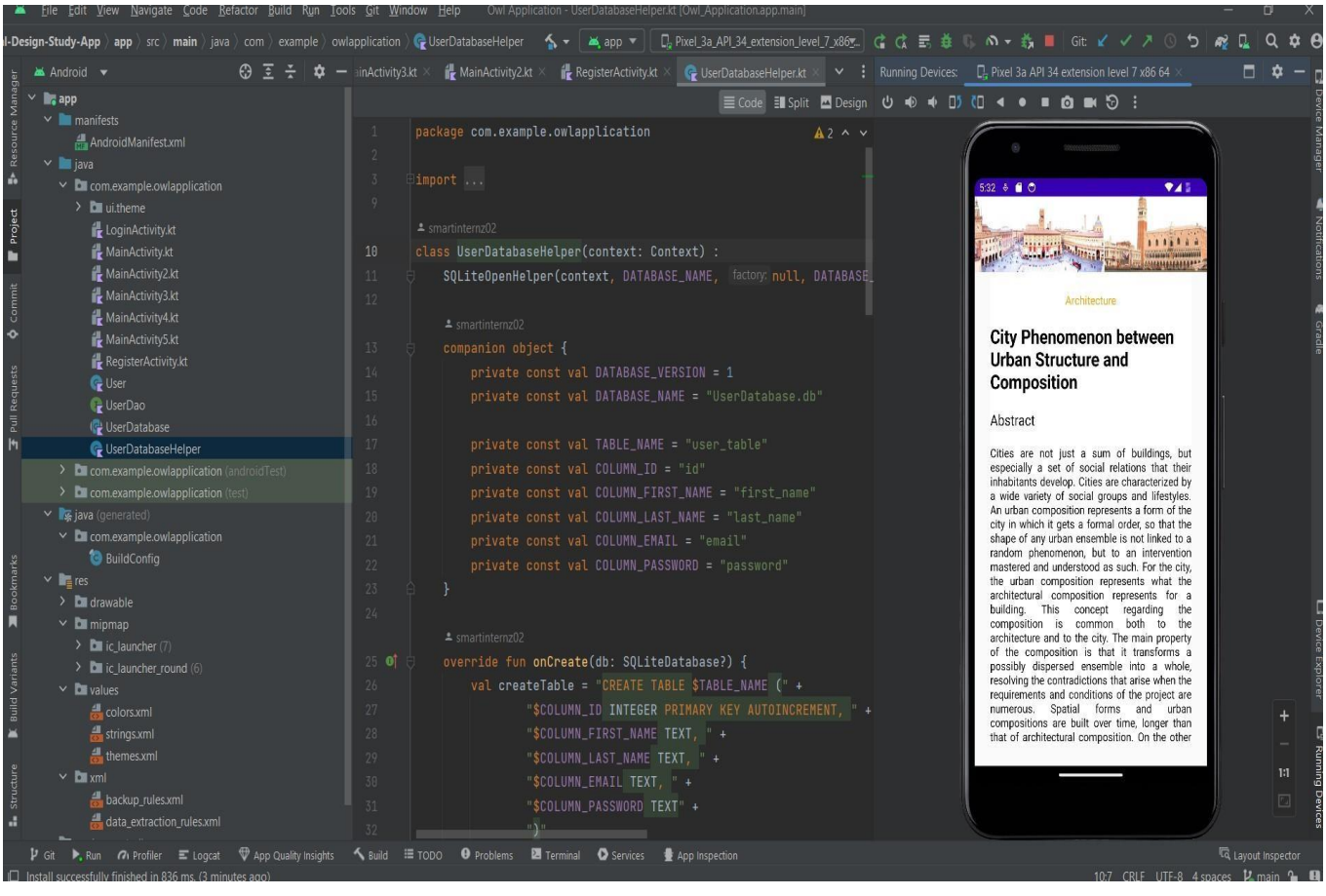

@Composable
fun StudyApp(context: Context) {
Column(
modifier = Modifier
.padding(20.dp)
.verticalScroll(rememberScrollState())
Text(text = "Study Material",
fontSize = 36.sp,
fontWeight = FontWeight.Bold,
color = Color(
modifier = Modifier.align(Alignment.CenterHorizontally
Spacer(modifier = Modifier.height(20.dp))
// 01
Card(
modifier = Modifier
.fillMaxWidth()
.height(250.dp)
.clickable {
context.startActivity(
Intent(context, MainActivity2::class.java)
Study Material
The Basics of Woodturning
An introduction to oil painting

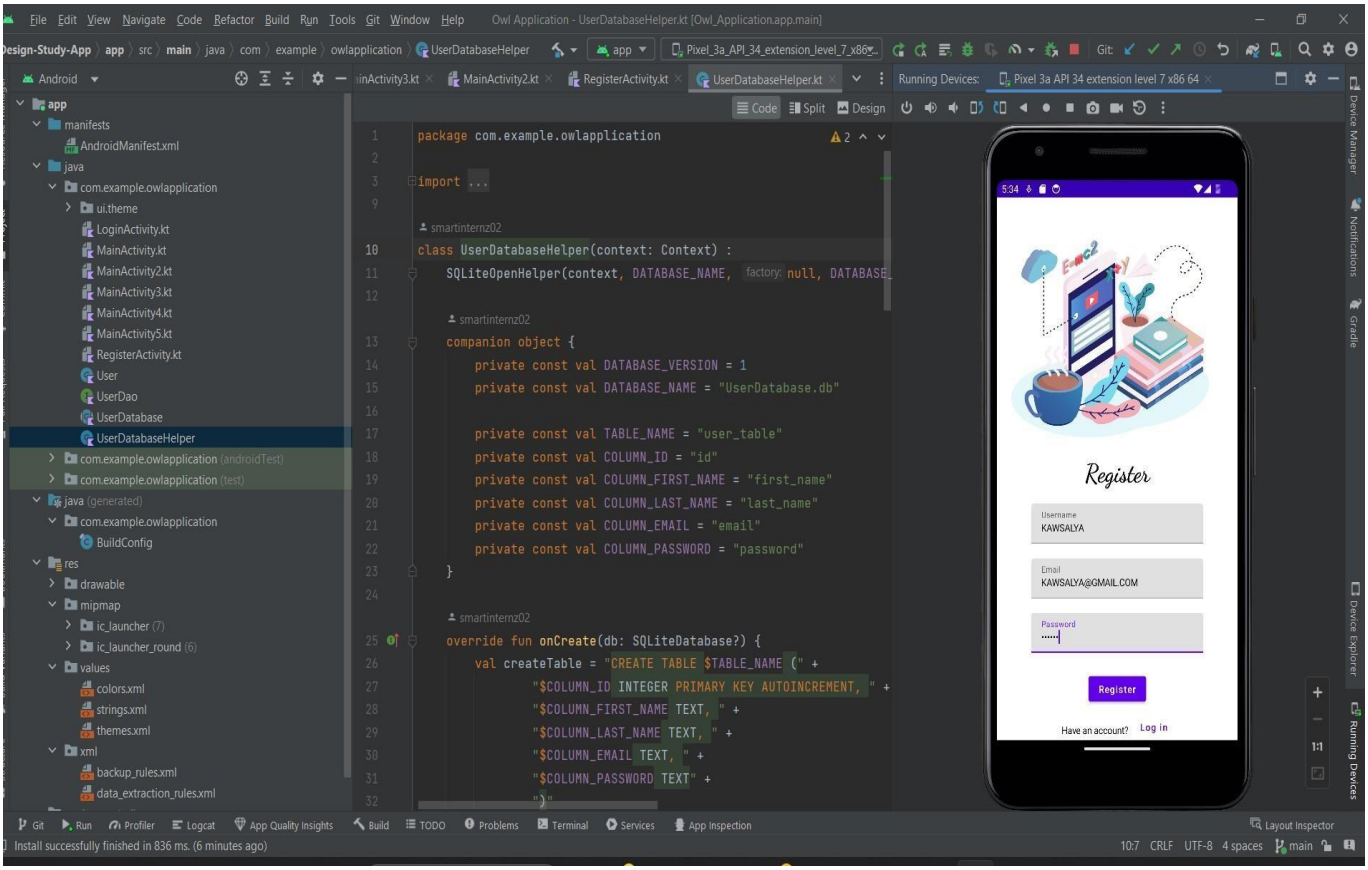

Owl Application - MainActivity.kt [Owl_Application.app.main]
Owl Application
MainActivity.kt
Pixel 3a API 34 extension level 7 x86 64
Running Devices:
Code
Split
Design
@Composable
fun StudyApp(context: Context) {
Column(
modifier = Modifier
.padding(20.dp)
.verticalScroll(rememberScrollState())
) { this: ColumnScope
Text(text = "Study Material",
fontSize = 36.sp,
fontWeight = FontWeight.Bold,
color = Color(color: 0xFFFFA500),
modifier = Modifier.align(Alignment.CenterHorizontally)
Spacer(modifier = Modifier.height(20.dp))
// 01
Card(
modifier = Modifier
.fillMaxWidth()
.height(250.dp)
.clickable {
context.startActivity(
Intent(context, MainActivity2::class.java)
Painting
An introduction to oil painting
What is oil paint?
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.
What ranges do Winsor & Newton have available?
We currently have 4 ranges of oil paint to suit a variety of different practices.

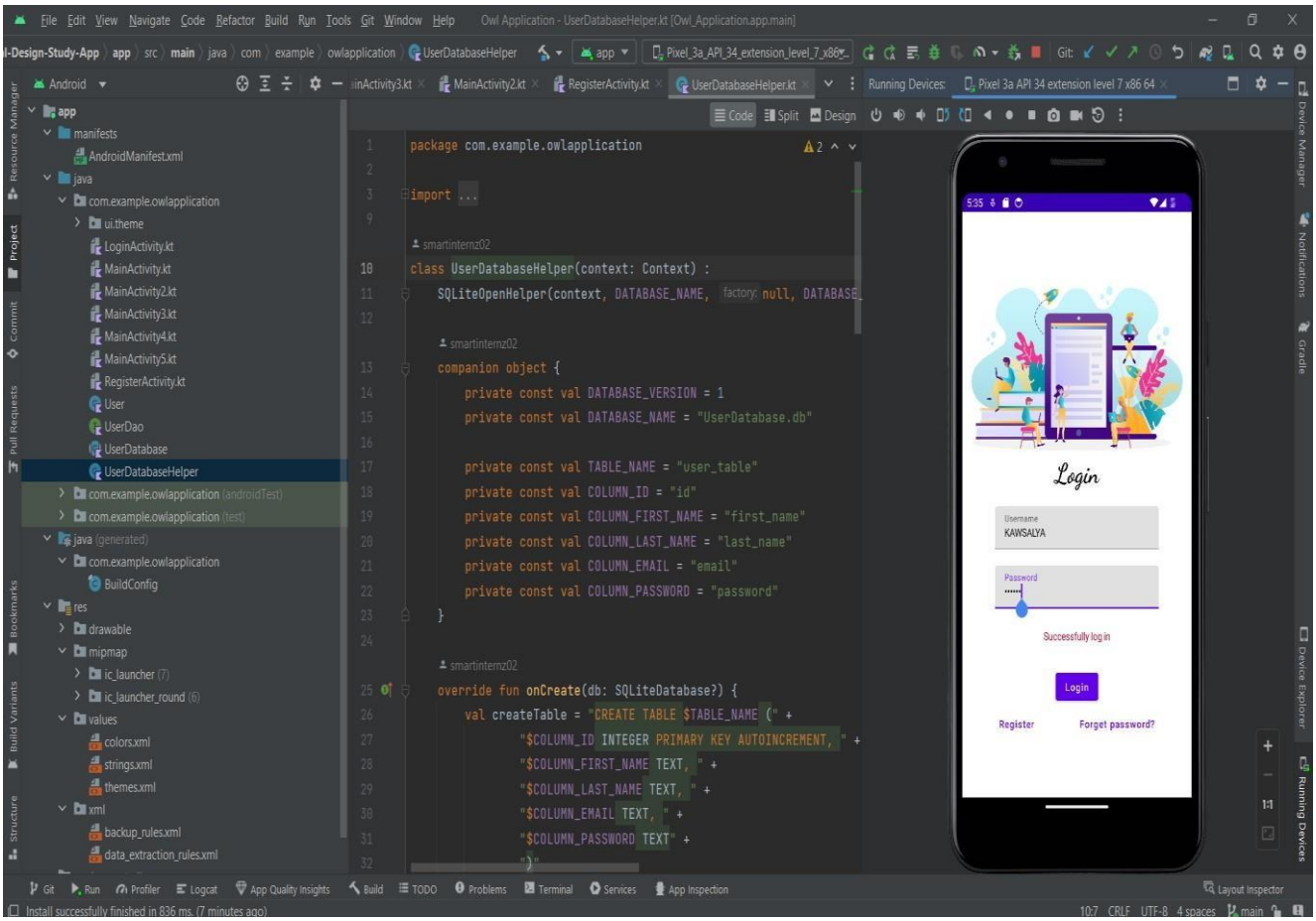
6. TESTING

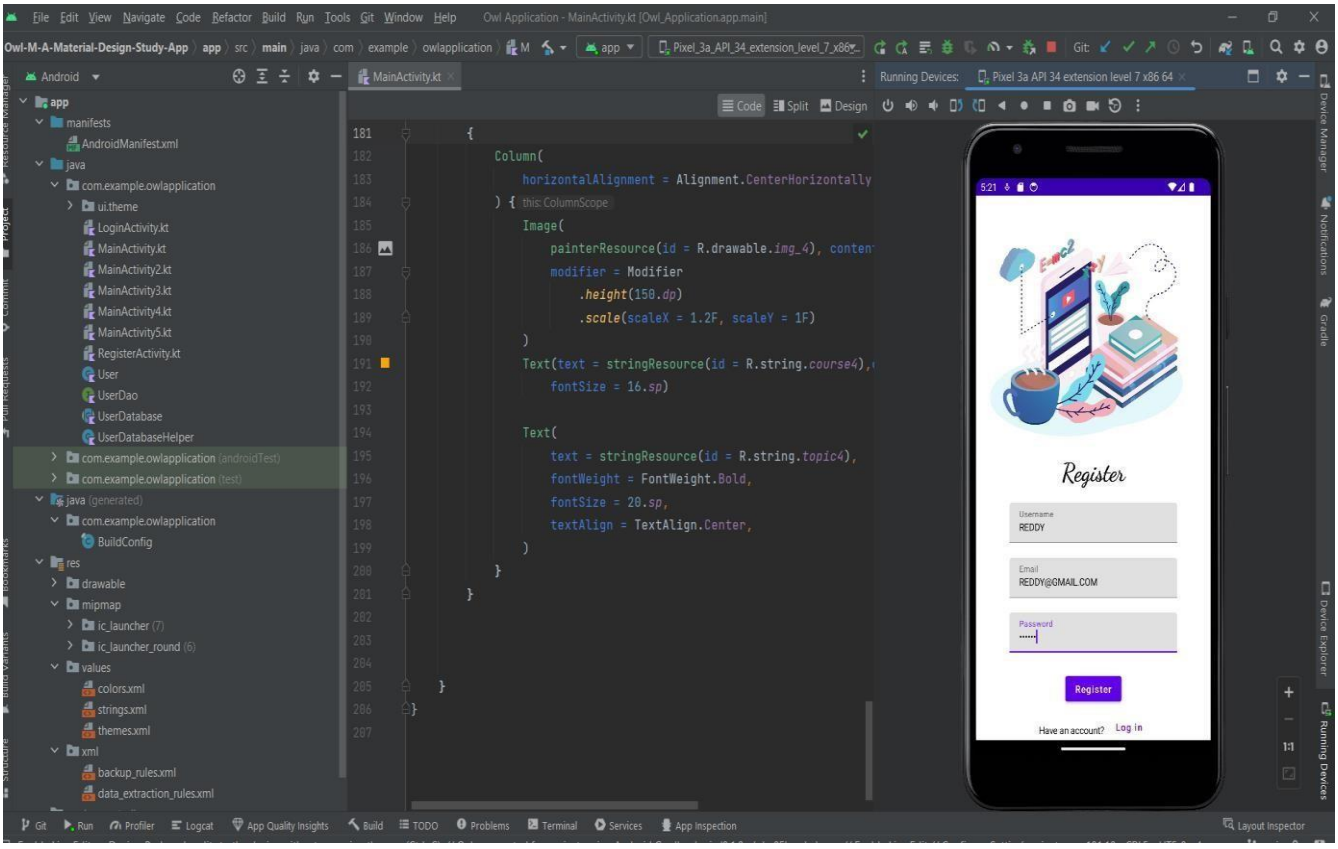

Owl Application - UserDatabaseHelper.kt [Owl_Application.app.main]
UserDatabaseHelper.kt
Pixel 3a API 34 extension level 7 x86 64
Running Devices:
package com.example.owlapplication
import ...
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_
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" +
Register
Username
JAI
Email
JAI@GMAIL.COM
Password
Register
Have an account? Log in

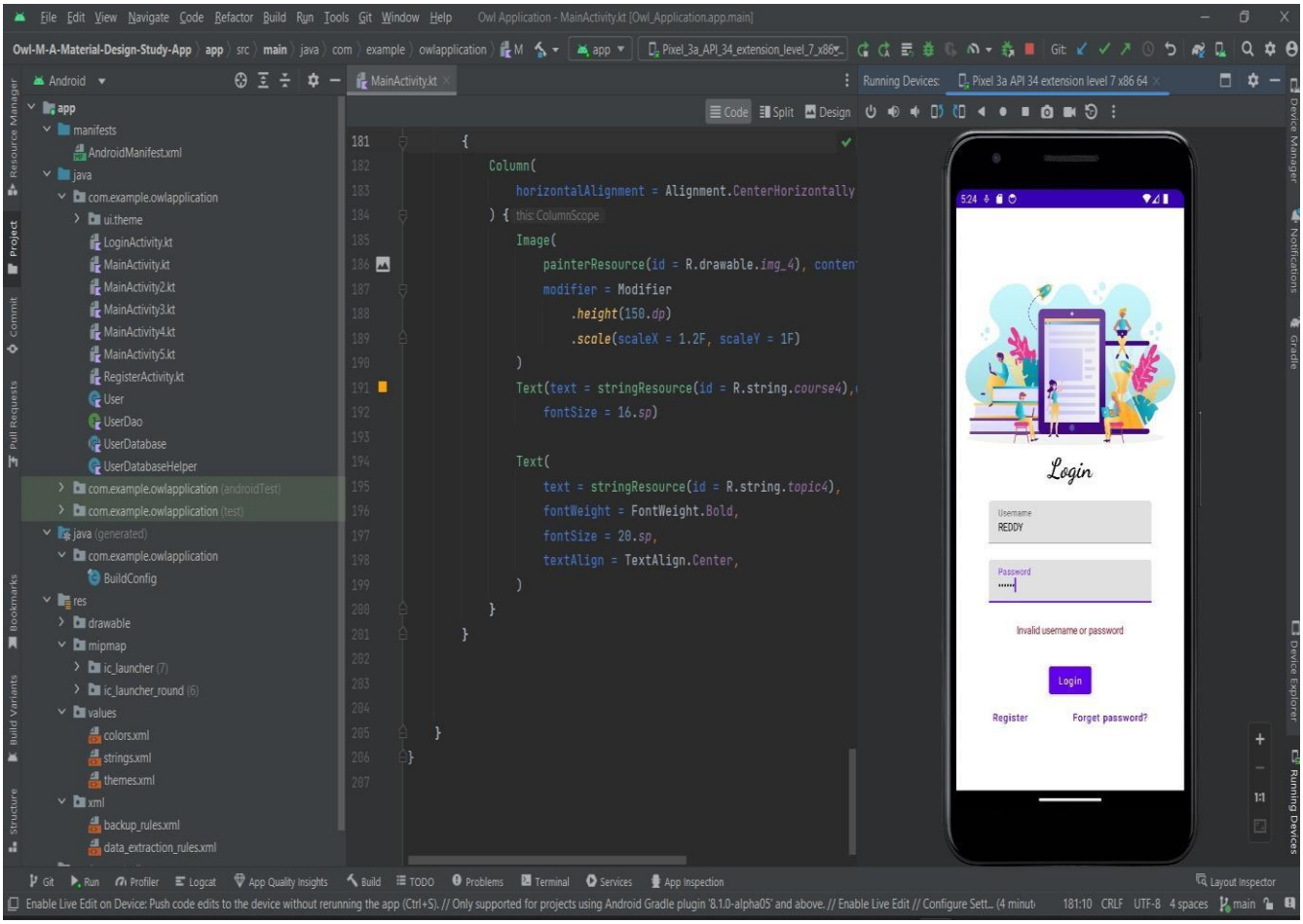

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Owl Application - UserDatabaseHelper.kt [Owl_Application.app.main]
Design-Study-App app src main java com example owlapplication UserDatabaseHelper
Android
app
manifests
AndroidManifest.xml
java
com.example.owlapplication
ui.theme
LoginActivity.kt
MainActivity.kt
MainActivity2.kt
MainActivity3.kt
MainActivity4.kt
MainActivity5.kt
RegisterActivity.kt
User
UserDao
UserDatabase
UserDatabaseHelper
com.example.owlapplication (androidTest)
com.example.owlapplication (test)
java (generated)
com.example.owlapplication
BuildConfig
res
drawable
mipmap
ic_launcher (7)
ic_launcher_round (6)
values
colors.xml
strings.xml
themes.xml
xml
backup_rules.xml
data_extraction_rules.xml
MainActivity3.kt
MainActivity2.kt
RegisterActivity.kt
UserDatabaseHelper.kt
Code
Split
Design
package com.example.owlapplication
import ...
smartinternz02
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_
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" +
")"
Running Devices:
Pixel 3a API 34 extension level 7 x86 64
5:30
Login
Username
JAI
Password
Login
Register
Forget password?
Git
Run
Profiler
Logcat
App Quality Insights
Build
TODO
Problems
Terminal
Services
App Inspection
Layout Inspector
Install successfully finished in 836 ms. (2 minutes ago)

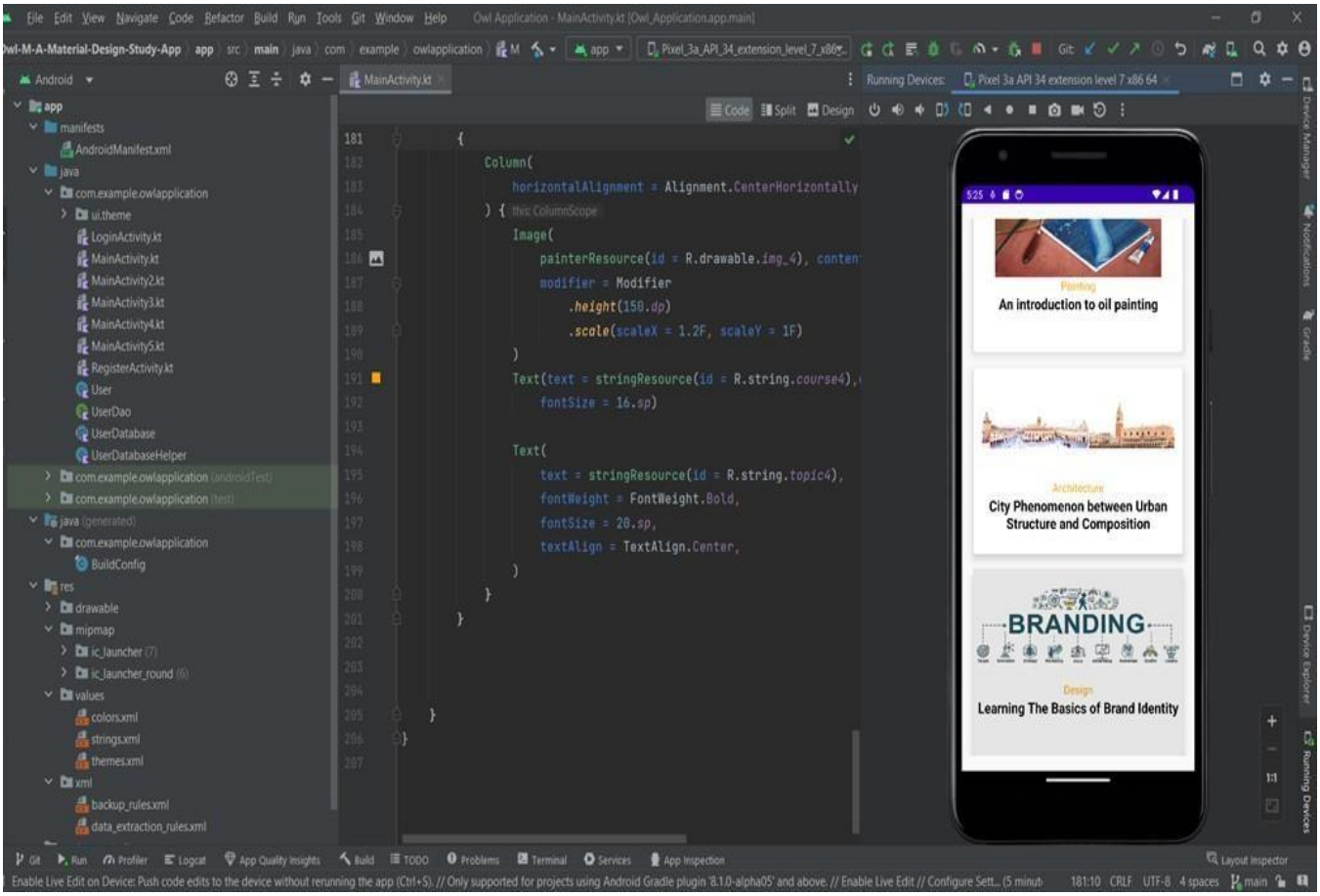

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Owl Application - UserDatabaseHelper.kt [Owl_Application.app.main]
Design-Study-App app src main java com example owlapplication UserDatabaseHelper
Android
app
manifests
AndroidManifest.xml
java
com.example.owlapplication
ui.theme
LoginActivity.kt
MainActivity.kt
MainActivity2.kt
MainActivity3.kt
MainActivity4.kt
MainActivity5.kt
RegisterActivity.kt
User
UserDao
UserDatabase
UserDatabaseHelper
com.example.owlapplication (androidTest)
com.example.owlapplication (test)
java (generated)
com.example.owlapplication
BuildConfig
res
drawable
mipmap
ic_launcher (7)
ic_launcher_round (6)
values
colors.xml
strings.xml
themes.xml
xml
backup_rules.xml
data_extraction_rules.xml
MainActivity3.kt
MainActivity2.kt
RegisterActivity.kt
UserDatabaseHelper.kt
Code
Split
Design
package com.example.owlapplication
import ...
smartinternz02
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_
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" +
")"
Running Devices:
Pixel 3a API 34 extension level 7 x86 64
5:32
Architecture
City Phenomenon between Urban Structure and Composition
Abstract
Cities are not just a sum of buildings, but especially a set of social relations that their inhabitants develop. Cities are characterized by a wide variety of social groups and lifestyles. An urban composition represents a form of the city in which it gets a formal order, so that the shape of any urban ensemble is not linked to a random phenomenon, but to an intervention mastered and understood as such. For the city, the urban composition represents what the architectural composition represents for a building. This concept regarding the composition is common both to the architecture and to the city. The main property of the composition is that it transforms a possibly dispersed ensemble into a whole, resolving the contradictions that arise when the requirements and conditions of the project are numerous. Spatial forms and urban compositions are built over time, longer than that of architectural composition. On the other
Git
Run
Profiler
Logcat
App Quality Insights
Build
TODO
Problems
Terminal
Services
App Inspection
Layout Inspector
Install successfully finished in 836 ms. (3 minutes ago)


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Owl Application - UserDatabaseHelper.kt [Owl_Application.app.main]
Design-Study-App app src main java com example owlapplication UserDatabaseHelper
Pixel_3a_API_34_extension_level_7_x86
Android
app
manifests
AndroidManifest.xml
java
com.example.owlapplication
ui.theme
LoginActivity.kt
MainActivity.kt
MainActivity2.kt
MainActivity3.kt
MainActivity4.kt
MainActivity5.kt
RegisterActivity.kt
User
UserDao
UserDatabase
UserDatabaseHelper
com.example.owlapplication (androidTest)
com.example.owlapplication (test)
java (generated)
com.example.owlapplication
BuildConfig
res
drawable
mipmap
ic_launcher (7)
ic_launcher_round (6)
values
colors.xml
strings.xml
themes.xml
xml
backup_rules.xml
data_extraction_rules.xml
inActivity3.kt
MainActivity2.kt
RegisterActivity.kt
UserDatabaseHelper.kt
Code
Split
Design
Running Devices:
Pixel 3a API 34 extension level 7 x86 64
package com.example.owlapplication
import ...
smartinternz02
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_
smartinternz02
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"
}
smartinternz02
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" +
")"
Register
Username
KAWSALYA
Email
KAWSALYA@GMAIL.COM
Password
Register
Have an account? Log in
Git Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection
Layout Inspector
Install successfully finished in 836 ms. (6 minutes ago)
10:7 CRLF UTF-8 4 spaces main


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Owl Application - UserDatabaseHelper.kt [Owl_Application.app.main]
Design-Study-App app src main java com example owlapplication UserDatabaseHelper
Pixel_3a_API_34_extension_level_7_x86
Android
app
manifests
AndroidManifest.xml
java
com.example.owlapplication
ui.theme
LoginActivity.kt
MainActivity.kt
MainActivity2.kt
MainActivity3.kt
MainActivity4.kt
MainActivity5.kt
RegisterActivity.kt
User
UserDao
UserDatabase
UserDatabaseHelper
com.example.owlapplication (androidTest)
com.example.owlapplication (test)
java (generated)
com.example.owlapplication
BuildConfig
res
drawable
mipmap
ic_launcher (7)
ic_launcher_round (6)
values
colors.xml
strings.xml
themes.xml
xml
backup_rules.xml
data_extraction_rules.xml
inActivity3.kt
MainActivity2.kt
RegisterActivity.kt
UserDatabaseHelper.kt
Code
Split
Design
Running Devices:
Pixel 3a API 34 extension level 7 x86 64
package com.example.owlapplication
import ...
smartinternz02
class UserDatabaseHelper(context: Context) :
SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_
smartinternz02
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"
}
smartinternz02
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" +
")"
Login
Username
KAWSALYA
Password
Successfully log in
Login
Register
Forget password?
Git Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection
Layout Inspector
Install successfully finished in 836 ms. (7 minutes ago)
10:7 CRLF UTF-8 4 spaces main


Owl Application - MainActivity.kt [Owl_Application.app.main]
Owl-M-A-Material-Design-Study-App
MainActivity.kt
Pixel 3a API 34 extension level 7 x86 64
Register
Username
REDDY
Email
REDDY@GMAIL.COM
Password
Register
Have an account? Log in


Owl Application - MainActivity.kt [Owl_Application.app.main]
Owl-M-A-Material-Design-Study-App
MainActivity.kt
Pixel 3a API 34 extension level 7 x86 64
Login
Username
REDDY
Password
Invalid username or password
Login
Register
Forget password?



APPENDIX

my source code :
https://github.com/chaitanyakumarredd/OWL-M-A-MATERIAL-DESIGN-STUDY-APP.gitl-m-a-material-design-studt-app

GOOGLE DEVELOPER PROFILE PUBLIC URL

Team Lead - https://g.dev/chaitanyakumarreddy

Team Member 1 - https://g.dev/KowsalyaLakshmi

Team Member 2 - https://g.dev/sairamireddy

Team Member 3 - https://g.dev/JayadeepYerram

7. CONCLUSION

In conclusion, the Owl-M: a Material design study application is a reliable and easytouse reading platform that allows users to read the content that is used for refering and reading via our Android mobile phones. The app's integration with Android Jetpack Compose and use of Kotlin programming language ensure that the app is both robust and userfriendly, providing a seamless experience for users. Whether you're looking to read in your convienient time, you can use this application via your mobile phones that should have been connected to internet. You can use this application only with the help of internet.