Owl-M: A Material Design Study App

S. Mahadevi

K.N. Poojasri

N. Karmenika

S. Subitha

Introduction

1.1 overview

A project that demonstrates the use of Android Jetpack Compose to build a UI for an 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.

Project Workflow:

● Users register into the application.

● After registration, user logins into the application.

● User enters the main page

● User can view the subject themes on selecting theme he can read about it.

1.2 purpose

Owl-M is a material design study app designed to help users enhance their learning experience. The primary purpose of Owl-M is to provide users with a convenient and accessible way to create, organize, and access study materials.

With Owl-M, users can create digital flashcards, organize study notes, and set reminders to stay on track with their

learning goals. The app also provides a range of customizable study modes, such as quiz mode, where users can test their knowledge and track their progress.

One of the key features of Owl-M is its sleek and intuitive material design interface. The app provides users with a clean and minimalist design, making it easy to navigate and use. Additionally, Owl-M offers a range of customization options, allowing users to personalize the app to their preferred style.
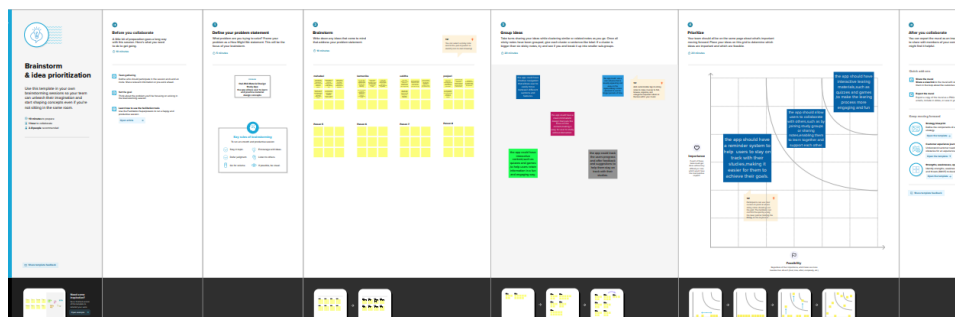
Overall, Owl-M is a powerful tool for students and anyone looking to enhance their learning experience. Its user-friendly interface and range of features make it a valuable addition to any study routine.

2.problem definition and design thinking

2.1 Empathy map
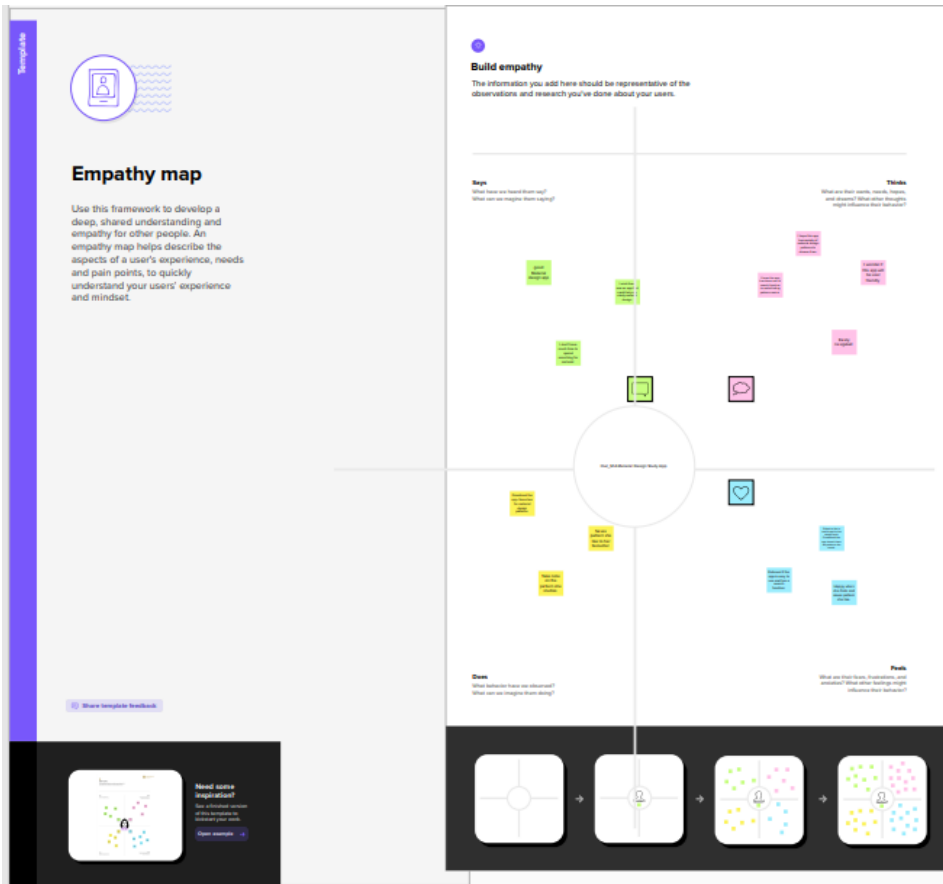
2.2 ideation & Brainstorming
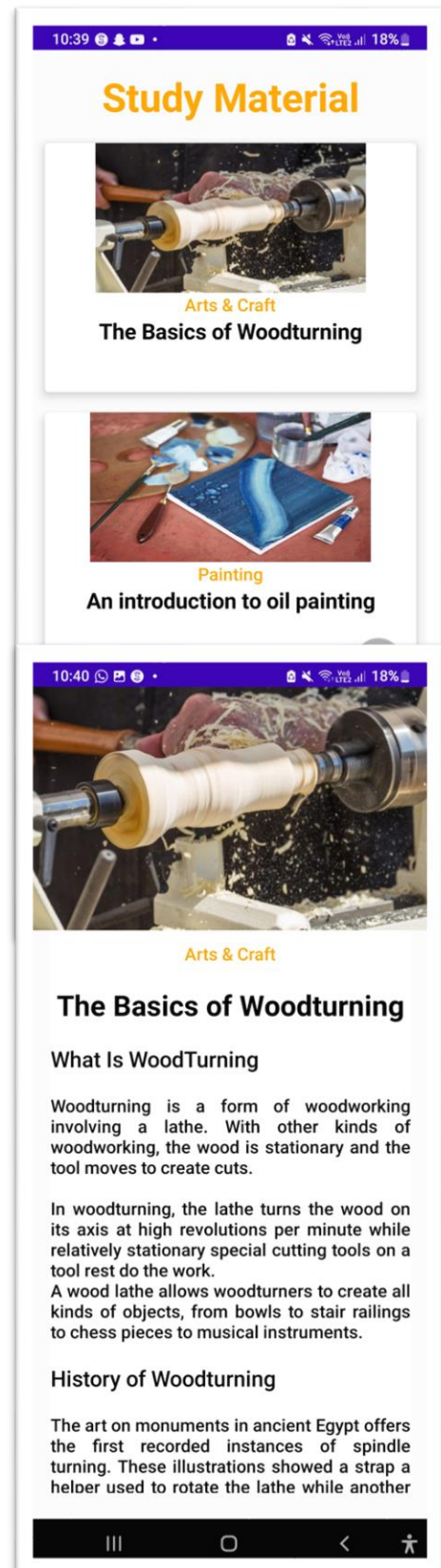

Login Page:

Register Page:

Main page:

Page no:

4. Advantages and Disadvantages

Advantages:

- Material Design: Owl-M is
  designed using Google's

- Material Design language, which makes it visually appealing and easy to use.
- Interactive learning: Owl-M offers interactive features such as flashcards, quizzes, and games to help users retain information.
- Personalized learning: Owl-M allows users to customize their study plan and set reminders to study at specific times, making it a great tool for self-paced learning.
- Progress tracking: Owl-M tracks users' progress and provides feedback on areas that need improvement, which helps users stay motivated and focused on their learning goals.
- Multi-device support: Owl-M is available on multiple devices, including mobile phones, tablets, and desktops, which makes it convenient for users to access their study material anytime and anywhere.

Disadvantages:

- Limited content: Owl-M offers a limited selection of study material, which may not be sufficient for some users' needs.

- No social features: Owl-M does not offer any social features, such as discussion forums or collaboration tools, which may limit users' ability to connect with other learners.
- Premium features: Some of Owl-M's more advanced features, such as personalized study plans and progress tracking, are only available to premium subscribers.
- No offline mode: Owl-M requires an internet connection to access study material, which may be inconvenient for users in areas with limited internet access.
- Limited customization: Although Owl-M offers some customization options, such as the ability to set reminders and choose study themes, there are limits to how much users can personalize their study experience.

5.Application

Learning:

❖ A study app can help students learn new concepts and information by providing interactive lessons, quizzes, and flashcards.

Revision:

❖ Students can use study apps to revise and reinforce what they have learned in class or through self-study.

Time management:

❖ A study app can help students manage their time by creating schedules, setting reminders, and tracking progress.

Collaboration:

❖ Study apps can facilitate collaboration between students by allowing them to share notes, assignments, and study materials.

Test preparation:

❖ A study app can help students prepare for exams by providing practice tests, mock exams, and feedback on their performance.

Language learning:

- ❖ Study apps can help users learn new languages through interactive lessons, vocabulary lists, and audio recordings.

Skill development:

- ❖ Study apps can also help users develop new skills by providing access to courses, tutorials, and instructional videos.

Professional development:

- ❖ Professionals can use study apps to keep up to date with new developments in their field, develop new skills, and earn certifications.

About Android Studio Application:

- Code editing: Android Studio provides a code editor with features like code completion, syntax highlighting, and refactoring. It supports multiple languages, including Java, Kotlin, and C++.
- Visual layout editor: Android Studio has a visual layout editor that allows developers to design the user interface (UI) of their app using a drag-and-drop interface. The layout editor provides tools to create

complex layouts and preview how the UI will look on different devices.

- Build system: Android Studio includes Gradle, a build system used to compile and package Android apps. Gradle is highly customizable and allows developers to automate many aspects of the build process.
- Debugging and testing: Android Studio has tools for debugging and testing Android apps, including a debugger, emulator, and integration with various testing frameworks.
- Integration with Google services: Android Studio has built-in support for integrating Google services like Google Maps, Google Cloud Platform, and Firebase.

About Android Studio

application:

- ♦ Studio is the official integrate development environment (IDE) for building Android Apps. It was first released by Google in 2013 and has since become the most popular development environment for Android app developers.

♦   Android Studio is based on the IntelliJ IDEA community edition, and it includes many tools and features designed specifically for developing Android apps Some of the key features of Android Studio include:

User Interface (UI) Designer:

Android Studio includes a powerful Designer that allows developers to easily create and modify app layouts using drag-and-drop tools. The UI designer supports a variety of layouts, including linear, relative, and constraint layouts.

Code Editor:

Android Studi includes a powerful code editor that provides syntax highlighting, code completion, and other features to help developers write clean, efficient code. The code editor also supports debugging and refactoring tools.

Emulator:

Android Studio includes a built-in emulator that allows developers to test their apps on different Android devices without needing to own the actual devices. The emulator supports a wide range of Android versions and device configurations.

Gradle Build System:

Android Studio uses the Gradle build system, which makes it easy to manage dependencies and build complex apps with multiple modules.

Version Control:

Android Studio supports version control systems like Git, allowing developers to easily manage their code changes and collaborate with other team members.

Performance Profiling:

Android Studio includes performance profiling tools that allow developers to identify performance bottlenecks in their apps and optimize their code for better performance. Overall, Android Studio is a powerful tool for developing high-quality Android apps. It provides a range of features and tools that make it easy for developers to build, test, and deploy their apps.

6. Conclusion

In conclusion, an owl material design study app could be a useful educational tool for learning about Google's material design principles, which is a design language used in the development of Android apps and other digital products. Such an app could include interactive lessons, tutorials, and examples to help users understand the core principles of material design, as well as quizzes and assessments to test their knowledge.

Overall, a material design study app could be beneficial for anyone interested in design, from beginners to professionals, who want to develop their skills and stay up-to-date with the latest design trends. By providing users with the tools and resources they need to learn about material design, an owl material design study app could help them create more engaging and effective designs for their Android apps and other digital products.

7. Future scope

The future scope for an owl material design study app is promising as there is an increasing demand for mobile app development and design. Here are some potential future developments for the app:

Integration with augmented reality (AR) and virtual

Reality (VR) technology:

With the increasing use of AR and VR in mobile app development, the owl material design study app could integrate these technologies to provide users with a more immersive learning experience.

Expansion to other platforms:

While material design is primarily used for Android apps, it could be applied to other platforms such as iOS, web, and desktop applications. The app could expand its scope to cover these platforms and provide design resources and tutorials for these areas.

Collaboration and community features:

The app could introduce features that enable users to collaborate and share their design work with other users. This could include a community forum, design challenges, and opportunities to showcase their work.

learning experiences:

The app could use artificial intelligence and machine learning algorithms to provide personalized learning experiences for users based on their interests, skill level, and learning pace.

8. Appendix

A. Source code

Creating user database classes:

Step 1: Create User data class

```
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?,
```

```kotlin
    @ColumnInfo(name = "password") val password:
String?,

    )
```

Step 2:

Create an User Dao interface

```kotlin
package com.example.owlapplication

import androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT * FROM user_table WHERE email =
:email")
    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)
```

```kotlin
    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

Step 3 : Create an UserDatabase class

```kotlin
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
```

```kotlin
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
        }
    }
}
```

Step 4 : Create an UserDatabaseHelper class

```kotlin
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"
```

```kotlin
        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)
```

```kotlin
    }

    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()
    }
```

```kotlin
@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)),
```

```kotlin
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASS
WORD)),
        )
    }
    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)),
```

```kotlin
            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> {
```

```kotlin
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)),
```

```kotlin
            )
            users.add(user)
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return users
  }

}
```

Building application UI and connecting to database.

Step 1: Creating LoginActivity.kt with database

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
```

```kotlin
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) {
```

```kotlin
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"
```

```kotlin
        )
        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)
        )
```

```kotlin
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 {
```

```kotlin
            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)

}
```

Step 2 : Creating RegisterActivity.kt with database

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

```kotlin
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?) {
```

```kotlin
        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),
```

```kotlin
    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") },
```

```kotlin
        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") },
```

```kotlin
        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 {
```

```kotlin
                    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,
```

```kotlin
                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)

}
```

Step 3: Creating MainActivity.kt file

In MainActivity.kt file the main application is developed

● **Before creating UI, we need to add some images in drawable which are in res**

```xml
<?xml version="1.0" encoding="utf-8"?>

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path
        android:fillColor="#3DDC84"
        android:pathData="M0,0h108v108h-108z" />
    <path
        android:fillColor="#00000000"
        android:pathData="M9,0L9,108"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
```

```
    android:fillColor="#00000000"

    android:pathData="M19,0L19,108"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M29,0L29,108"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M39,0L39,108"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M49,0L49,108"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
```

```xml
<path
    android:fillColor="#00000000"
    android:pathData="M59,0L59,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M69,0L69,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M79,0L79,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M89,0L89,108"
    android:strokeWidth="0.8"
```

```xml
        android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M99,0L99,108"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,9L108,9"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,19L108,19"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M0,29L108,29"
```

```
    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,39L108,39"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,49L108,49"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,59L108,59"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"
```

```
    android:pathData="M0,69L108,69"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,79L108,79"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,89L108,89"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M0,99L108,99"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path
```

```
    android:fillColor="#00000000"

    android:pathData="M19,29L89,29"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M19,39L89,39"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M19,49L89,49"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
<path

    android:fillColor="#00000000"

    android:pathData="M19,59L89,59"

    android:strokeWidth="0.8"

    android:strokeColor="#33FFFFFF" />
```

```xml
<path
    android:fillColor="#00000000"
    android:pathData="M19,69L89,69"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M19,79L89,79"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M29,19L29,89"
    android:strokeWidth="0.8"
    android:strokeColor="#33FFFFFF" />
<path
    android:fillColor="#00000000"
    android:pathData="M39,19L39,89"
    android:strokeWidth="0.8"
```

```xml
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M49,19L49,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M59,19L59,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M69,19L69,89"
        android:strokeWidth="0.8"
        android:strokeColor="#33FFFFFF" />
    <path
        android:fillColor="#00000000"
        android:pathData="M79,19L79,89"
```

```
        android:strokeWidth="0.8"

        android:strokeColor="#33FFFFFF" />
</vector>
```
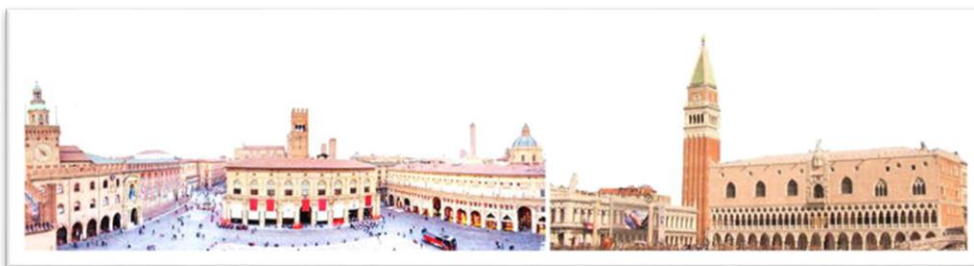
Add some image: add first image
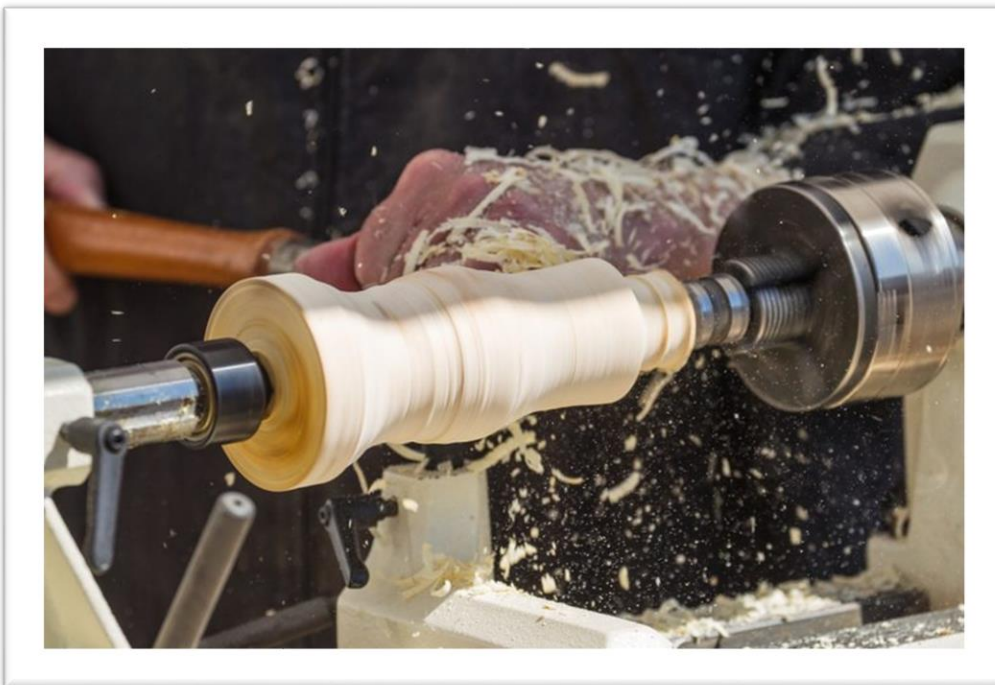
Img_2.png
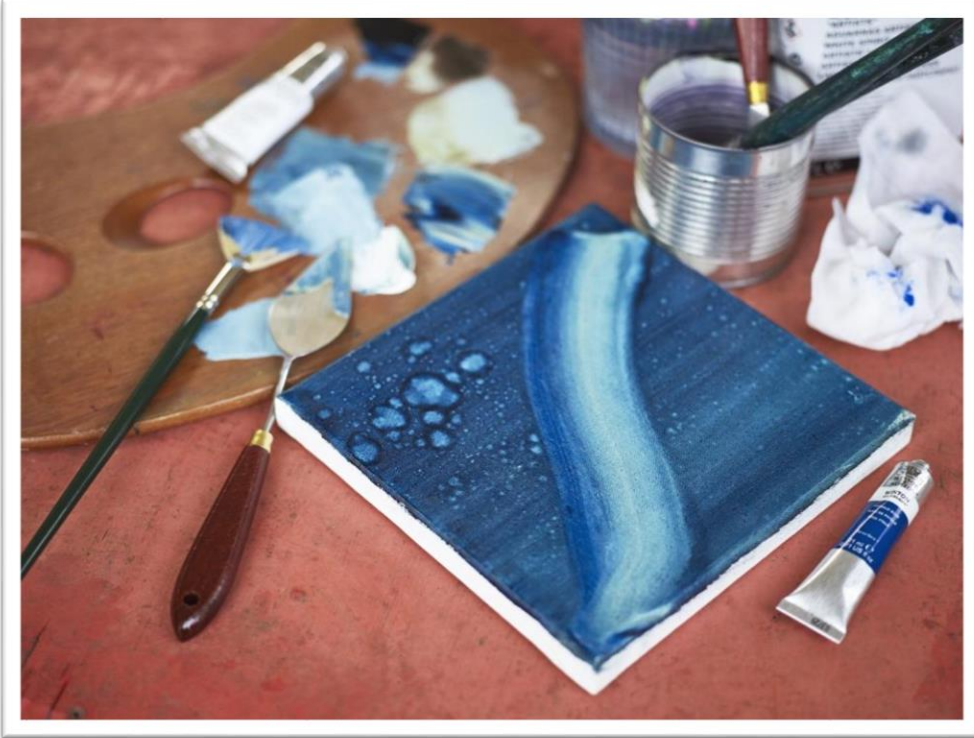
Img_3.png

Img_4.png

study_login.jpg

study_signup.jpg

MainActivity.kt

package com.example.owlapplication

import android.content.Context

import android.content.Intent

import android.os.Bundle

```kotlin
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.Card
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
```

```kotlin
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            StudyApp(this)
        }
    }
}

@Composable
fun StudyApp(context: Context) {

    Column(
        modifier = Modifier
            .padding(20.dp)
            .verticalScroll(rememberScrollState())
```

```kotlin
    ) {

        Text(text = "Study Material",

            fontSize = 36.sp,

            fontWeight = FontWeight.Bold,

            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)
```

```
            )
        },
        elevation = 8.dp
    )
    {
        Column(
            horizontalAlignment =
Alignment.CenterHorizontally
        ) {
            Image(
                painterResource(id = R.drawable.img_1),
contentDescription = "",
                modifier = Modifier
                    .height(150.dp)
                    .scale(scaleX = 1.2F, scaleY = 1F)
            )
            Text(text = stringResource(id =
R.string.course1),color = Color(0xFFFFA500),
                fontSize = 16.sp)
```

```kotlin
        Text(
            text = stringResource(id = R.string.topic1),
            fontWeight = FontWeight.Bold,
            fontSize = 20.sp,
            textAlign = TextAlign.Center,
        )
    }
}

Spacer(modifier = Modifier.height(20.dp))

//    02
Card(
    modifier = Modifier
        .fillMaxWidth()
        .height(250.dp)
        .clickable {
            context.startActivity(
                Intent(context, MainActivity3::class.java)
```

```kotlin
        )
    },
    elevation = 8.dp
)
{
    Column(
    horizontalAlignment =
Alignment.CenterHorizontally
    ) {
        Image(
            painterResource(id = R.drawable.img_2),
contentDescription = "",
            modifier = Modifier
                .height(150.dp)
                .scale(scaleX = 1.4F, scaleY = 1F)
        )
        Text(text = stringResource(id =
R.string.course2),color = Color(0xFFFFA500),
            fontSize = 16.sp)
```

```kotlin
            Text(
                text = stringResource(id = R.string.topic2),
                fontWeight = FontWeight.Bold,
                fontSize = 20.sp,
                textAlign = TextAlign.Center,
            )
        }
    }

    Spacer(modifier = Modifier.height(20.dp))

    //      03
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(250.dp)
            .clickable {
                context.startActivity(
                    Intent(context, MainActivity4::class.java)
```

```kotlin
            )
        },
        elevation = 8.dp
    )
    {
        Column(
            horizontalAlignment =
Alignment.CenterHorizontally
        ) {
            Image(
                painterResource(id = R.drawable.img_3),
contentDescription = "",
                modifier = Modifier
                    .height(150.dp)
                    .scale(scaleX = 1.2F, scaleY = 1F)
            )
            Text(text = stringResource(id =
R.string.course3),color = Color(0xFFFFA500),
                fontSize = 16.sp)
```

```kotlin
        Text(
            text = stringResource(id = R.string.topic3),
            fontWeight = FontWeight.Bold,
            fontSize = 20.sp,
            textAlign = TextAlign.Center,
        )
    }
}


Spacer(modifier = Modifier.height(20.dp))

//      04
Card(
    modifier = Modifier
        .fillMaxWidth()
        .height(250.dp)
        .clickable {
            context.startActivity(
```

```kotlin
                    Intent(context, MainActivity5::class.java)

                )
            },
        elevation = 8.dp
    )
    {

    Column(
        horizontalAlignment =
Alignment.CenterHorizontally
    ) {
        Image(
            painterResource(id = R.drawable.img_4),
contentDescription = "",
            modifier = Modifier
                .height(150.dp)
                .scale(scaleX = 1.2F, scaleY = 1F)
        )
        Text(text = stringResource(id =
R.string.course4),color = Color(0xFFFFA500),
```

```kotlin
                    fontSize = 16.sp)


            Text(

                text = stringResource(id = R.string.topic4),

                fontWeight = FontWeight.Bold,

                fontSize = 20.sp,

                textAlign = TextAlign.Center,

            )

        }

      }



  }

}
```

```kotlin
package com.example.owlapplication


import android.os.Bundle

import androidx.activity.ComponentActivity
```

```kotlin
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
```

```kotlin
import
com.example.owlapplication.ui.theme.OwlApplicationT
heme

class MainActivity2 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting()
        }
    }
}
@Composable
fun Greeting() {
    Column(
        modifier = Modifier.padding(start = 26.dp, end =
26.dp, bottom = 26.dp)
            .verticalScroll(rememberScrollState())
            .background(Color.White),
        verticalArrangement = Arrangement.Top
```

```kotlin
) {

    Image(
        painterResource(id = R.drawable.img_1),
        contentDescription = "",
        modifier =
Modifier.align(Alignment.CenterHorizontally)
            .scale(scaleX = 1.5F, scaleY = 1.5F)
    )

    Spacer(modifier = Modifier.height(60.dp))

    Text(
        text = stringResource(id = R.string.course1),
        color = Color(0xFFFFA500),
        fontSize = 16.sp,
        modifier =
Modifier.align(Alignment.CenterHorizontally)
    )

    Spacer(modifier = Modifier.height(20.dp))
```

```kotlin
Text(
    text = stringResource(id = R.string.topic1),
    fontWeight = FontWeight.Bold,
    fontSize = 26.sp,
    modifier =
Modifier.align(Alignment.CenterHorizontally)

)
Spacer(modifier = Modifier.height(20.dp))
Text(
    text = stringResource(id =
R.string.subheading1_1),
    modifier = Modifier.align(Alignment.Start),
    fontSize = 20.sp
)

Spacer(modifier = Modifier.height(20.dp))

Text(
    text = stringResource(id = R.string.text1_1),
```

```kotlin
            modifier = Modifier.align(Alignment.Start),

            textAlign = TextAlign.Justify,

            fontSize = 16.sp

        )

        Spacer(modifier = Modifier.height(20.dp))

        Text(

            text = stringResource(id =
R.string.subheading1_2),

            modifier = Modifier.align(Alignment.Start),

            fontSize = 20.sp

        )

        Spacer(modifier = Modifier.height(20.dp))

        Text(

            text = stringResource(id = R.string.text1_2),

            modifier = Modifier.align(Alignment.Start),

            textAlign = TextAlign.Justify,

            fontSize = 16.sp
```

```
        )



    }
}
```

Step 5: Creating MainActivity3.kt file

```
package com.example.owlapplication

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.foundation.rememberScrollState

import androidx.compose.foundation.verticalScroll

import androidx.compose.material.Text
```

```kotlin
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity3 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting1()
        }
    }
```

```kotlin
}
@Composable
fun Greeting1() {
    Column(
        modifier = Modifier.padding(start = 26.dp, end =
26.dp, bottom = 26.dp)
            .verticalScroll(rememberScrollState())
            .background(Color.White),
        verticalArrangement = Arrangement.Top
    ) {

        Image(
            painterResource(id = R.drawable.img_2),
            contentDescription = "",
            modifier =
Modifier.align(Alignment.CenterHorizontally)
                .scale(scaleX = 1.2F, scaleY = 1F)
        )

        Spacer(modifier = Modifier.height(20.dp))
```

```kotlin
Text(
    text = stringResource(id = R.string.course2),
    color = Color(0xFFFFA500),
    fontSize = 16.sp,
    modifier =
Modifier.align(Alignment.CenterHorizontally)
    )

    Spacer(modifier = Modifier.height(20.dp))

    Text(
        text = stringResource(id = R.string.topic2),
        fontWeight = FontWeight.Bold,
        fontSize = 26.sp,
        modifier =
Modifier.align(Alignment.CenterHorizontally)

    )
    Spacer(modifier = Modifier.height(20.dp))
    Text(
```

```kotlin
        text = stringResource(id =
R.string.subheading2_1),

        modifier = Modifier.align(Alignment.Start),

        fontSize = 20.sp

    )

    Spacer(modifier = Modifier.height(20.dp))

    Text(

        text = stringResource(id = R.string.text2_1),

        modifier = Modifier.align(Alignment.Start),

        textAlign = TextAlign.Justify,

        fontSize = 16.sp

    )

    Spacer(modifier = Modifier.height(20.dp))

    Text(

        text = stringResource(id =
R.string.subheading2_2),

        modifier = Modifier.align(Alignment.Start),
```

```kotlin
            fontSize = 20.sp
        )

        Spacer(modifier = Modifier.height(20.dp))

        Text(
            text = stringResource(id = R.string.text2_2),
            modifier = Modifier.align(Alignment.Start),
            textAlign = TextAlign.Justify,
            fontSize = 16.sp
        )

    }
}
```

Step 6: Creating MainActivity4.kt file

```kotlin
package com.example.owlapplication
```

```
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.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
```

```kotlin
import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.owlapplication.ui.theme.OwlApplicationTheme


class MainActivity4 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting2()
        }
    }
}
@Composable
fun Greeting2() {
```

```kotlin
Column(
    modifier = Modifier.padding(start = 26.dp, end =
26.dp, bottom = 26.dp)
        .verticalScroll(rememberScrollState())
        .background(Color.White),
    verticalArrangement = Arrangement.Top
) {

    Image(
        painterResource(id = R.drawable.img_3),
        contentDescription = "",
        modifier =
Modifier.align(Alignment.CenterHorizontally)
            .scale(scaleX = 1.5F, scaleY = 2F)
    )

    Spacer(modifier = Modifier.height(60.dp))

    Text(
        text = stringResource(id = R.string.course3),
```

```kotlin
            color = Color(0xFFFFA500),

            fontSize = 16.sp,

            modifier =
Modifier.align(Alignment.CenterHorizontally)

        )


        Spacer(modifier = Modifier.height(20.dp))


        Text(

            text = stringResource(id = R.string.topic3),

            fontWeight = FontWeight.Bold,

            fontSize = 26.sp,

            modifier =
Modifier.align(Alignment.CenterHorizontally)


        )

        Spacer(modifier = Modifier.height(20.dp))

        Text(

            text = stringResource(id =
R.string.subheading3_1),
```

```kotlin
        modifier = Modifier.align(Alignment.Start),

        fontSize = 20.sp

    )

    Spacer(modifier = Modifier.height(20.dp))

    Text(

        text = stringResource(id = R.string.text3_1),

        modifier = Modifier.align(Alignment.Start),

        textAlign = TextAlign.Justify,

        fontSize = 16.sp

    )

    Spacer(modifier = Modifier.height(20.dp))
    Text(

        text = stringResource(id =
R.string.subheading3_2),

        modifier = Modifier.align(Alignment.Start),

        fontSize = 20.sp

    )
```

```
    Spacer(modifier = Modifier.height(20.dp))

    Text(
        text = stringResource(id = R.string.text3_2),
        modifier = Modifier.align(Alignment.Start),
        textAlign = TextAlign.Justify,
        fontSize = 16.sp
    )




    }
}
```

Step 7: Creating MainActivity5.kt file

package com.example.owlapplication

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

```kotlin
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
```

```kotlin
import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.sp

import com.example.owlapplication.ui.theme.OwlApplicationTheme

class MainActivity5 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting3()
        }
    }
}
@Composable
fun Greeting3() {
    Column(
        modifier = Modifier.padding(start = 26.dp, end = 26.dp, bottom = 26.dp)
            .verticalScroll(rememberScrollState())
```

```kotlin
        .background(Color.White),
    verticalArrangement = Arrangement.Top
) {

    Image(
        painterResource(id = R.drawable.img_4),
        contentDescription = "",
        modifier =
Modifier.align(Alignment.CenterHorizontally)
            .scale(scaleX = 1.5F, scaleY = 1.5F)
    )

    Spacer(modifier = Modifier.height(60.dp))

    Text(
        text = stringResource(id = R.string.course4),
        color = Color(0xFFFFA500),
        fontSize = 16.sp,
        modifier =
Modifier.align(Alignment.CenterHorizontally)
```

```kotlin
        )

        Spacer(modifier = Modifier.height(20.dp))

        Text(
            text = stringResource(id = R.string.topic4),
            fontWeight = FontWeight.Bold,
            fontSize = 26.sp,
            modifier =
Modifier.align(Alignment.CenterHorizontally)

        )
        Spacer(modifier = Modifier.height(20.dp))
        Text(
            text = stringResource(id =
R.string.subheading4_1),
            modifier = Modifier.align(Alignment.Start),
            fontSize = 20.sp
        )

        Spacer(modifier = Modifier.height(20.dp))
```

```kotlin
Text(
    text = stringResource(id = R.string.text4_1),
    modifier = Modifier.align(Alignment.Start),
    textAlign = TextAlign.Justify,
    fontSize = 16.sp
)

Spacer(modifier = Modifier.height(20.dp))
Text(
    text = stringResource(id =
R.string.subheading4_2),
    modifier = Modifier.align(Alignment.Start),
    fontSize = 20.sp
)

Spacer(modifier = Modifier.height(20.dp))

Text(
    text = stringResource(id = R.string.text4_2),
```

```
            modifier = Modifier.align(Alignment.Start),

            textAlign = TextAlign.Justify,

            fontSize = 16.sp

        )




    }

}
```

Task 6:

Modifying AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>

<manifest
xmlns:android="http://schemas.android.com/apk/res/
android"

    xmlns:tools="http://schemas.android.com/tools">

    <application

        android:allowBackup="true"
```

```xml
android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.OwlApplication"
        tools:targetApi="31">
        <activity
            android:name=".RegisterActivity"
            android:exported="false"
            android:label="@string/title_activity_register"
            android:theme="@style/Theme.OwlApplication"
/>
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="MainActivity"
```

```xml
        android:theme="@style/Theme.OwlApplication"
/>
    <activity
        android:name=".MainActivity5"
        android:exported="false"
        android:label="@string/title_activity_main5"
        android:theme="@style/Theme.OwlApplication"
/>
    <activity
        android:name=".MainActivity4"
        android:exported="false"
        android:label="@string/title_activity_main4"
        android:theme="@style/Theme.OwlApplication"
/>
    <activity
        android:name=".MainActivity3"
        android:exported="false"
        android:label="@string/title_activity_main3"
        android:theme="@style/Theme.OwlApplication"
/>
```

```xml
<activity
    android:name=".MainActivity2"
    android:exported="false"
    android:label="@string/title_activity_main2"
    android:theme="@style/Theme.OwlApplication"
/>
<activity
    android:name=".LoginActivity"
    android:exported="true"
    android:label="@string/app_name"

android:theme="@style/Theme.OwlApplication">
    <intent-filter>
        <action
android:name="android.intent.action.MAIN" />

        <category
android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```
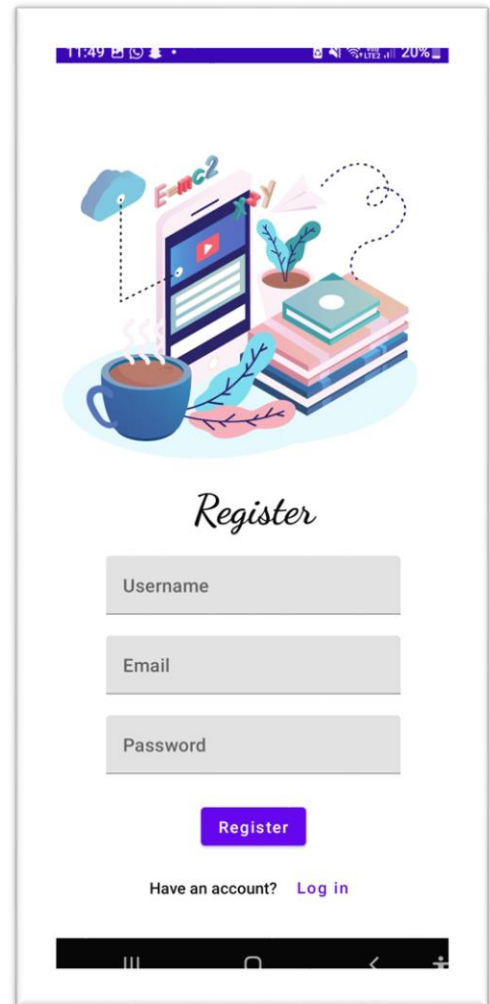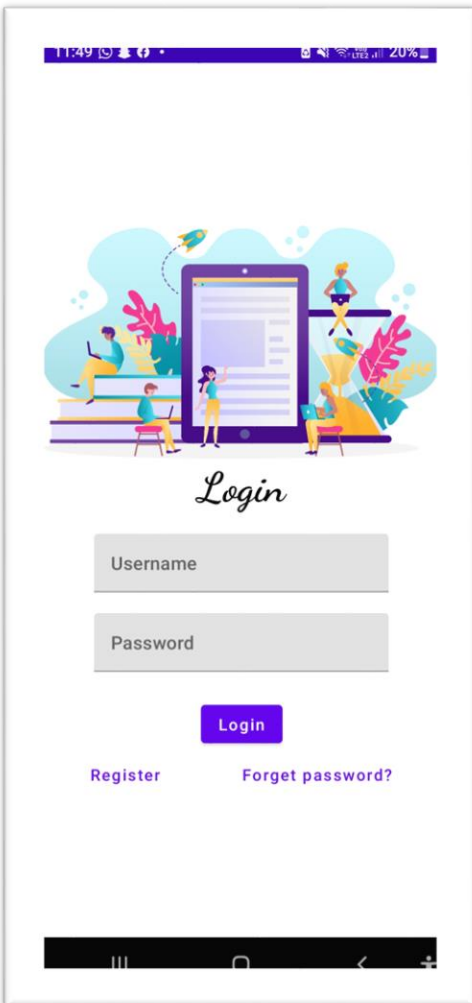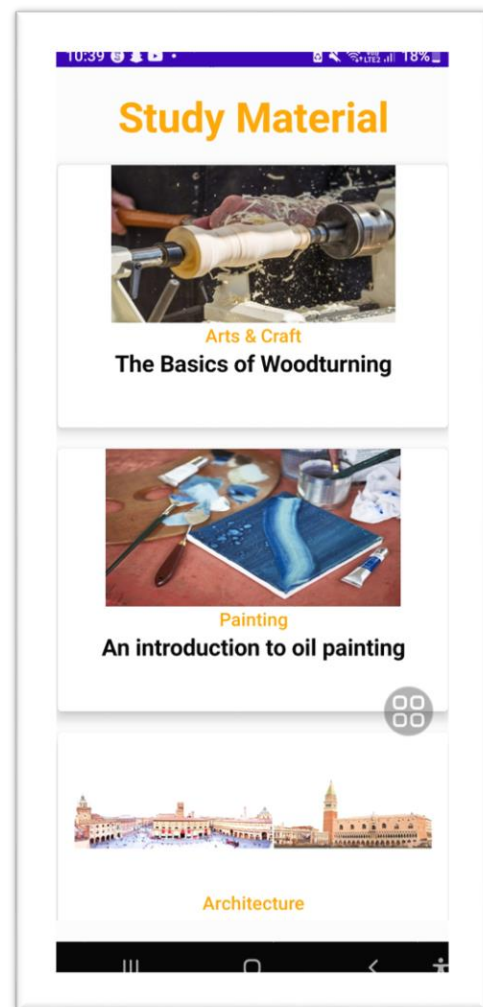
```
  </application>

</manifest>
```

Task 7:

Running the application:

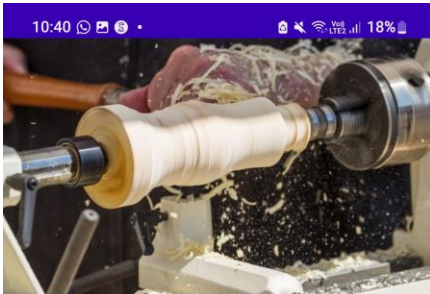**Final Output of the Application:**

Register Page :

Login Page :

MainPage :

Book page:

The Basics of Woodturning

What Is WoodTurning

Woodturning is a form of woodworking involving a lathe. With other kinds of woodworking, the wood is stationary and the tool moves to create cuts.

In woodturning, the lathe turns the wood on its axis at high revolutions per minute while relatively stationary special cutting tools on a tool rest do the work.
A wood lathe allows woodturners to create all kinds of objects, from bowls to stair railings to chess pieces to musical instruments.

History of Woodturning

The art on monuments in ancient Egypt offers the first recorded instances of spindle turning. These illustrations showed a strap a helper used to rotate the lathe while another