Arulmighu Palaniandavar Collage of Arts and Culture

Palani – 624 601

A

Project Report

On

“Material Design Study App “

For the course

NAAN MUDHALVAN

SUBMITTED BY

VISWANATHAN M (C0S11617)

|  |  |  |
| --- | --- | --- |
| SR NO. | CHAPTER NAME | PAGE NO. |
| 01. | Information | 4 |
| 02. | Project Definition & Design Thinking | 5 |
| 03. | Result | 7 |
| 04. | Advantages & Disadvantages | 12 |
| 05. | Applications | 17 |
| 06. | Conclusion | 19 |
| 07. | Future scope | 20 |
| 08. | Appendix | 23 |

01. INTRODUCTION

1.1 Overview

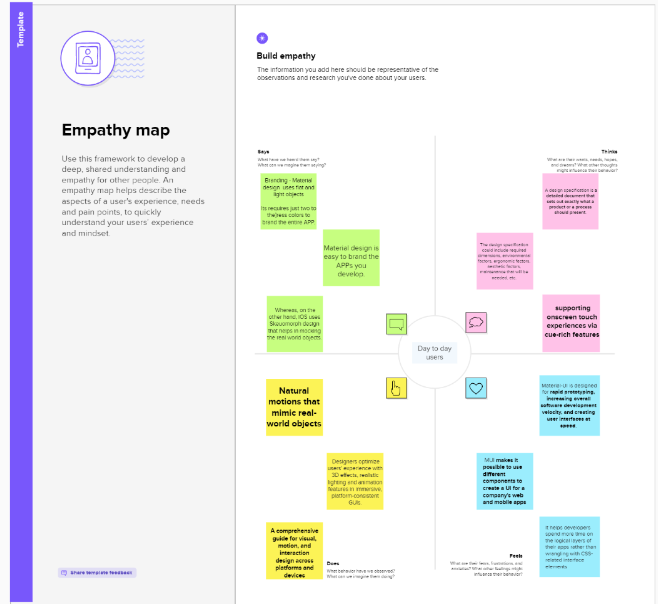
Material Design is an Android-oriented design language created by Google, supporting onscreen touch experiences via cue-rich features and natural motions that mimic real-world objects. Designers optimize users' experience with 3D effects, realistic lighting and animation features in immersive, platform-consistent GUIs.

1.2 Purpose

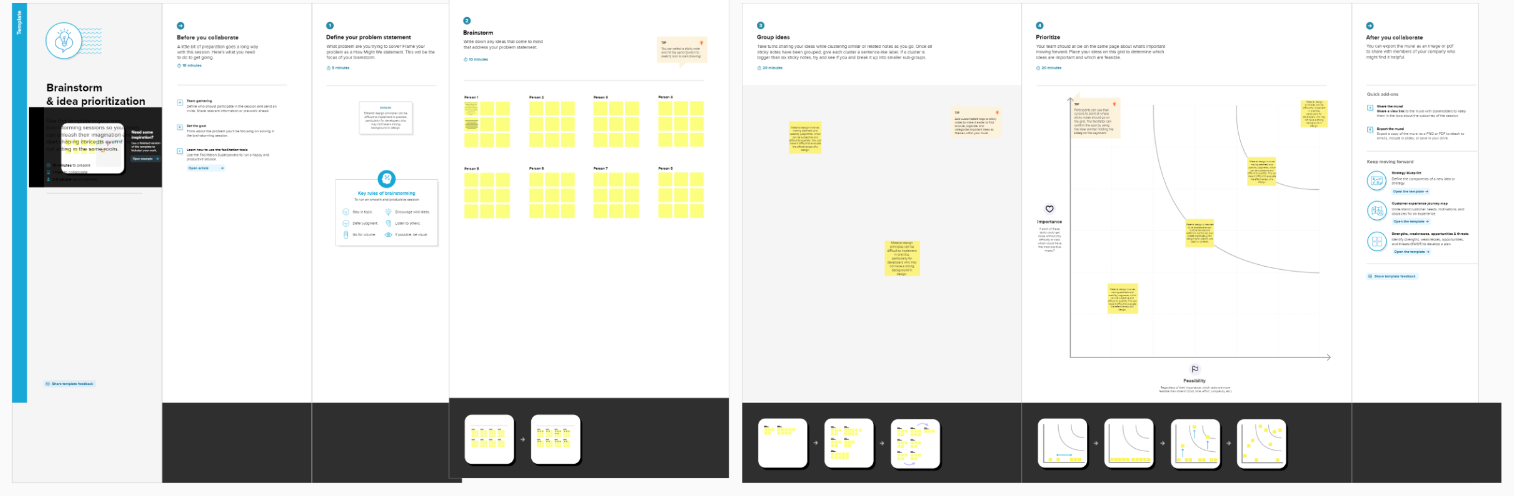
With Material Design, the goal is to deliver high-quality output consistently across platforms, giving users control over clearly indicated, pleasant-looking components that behave like real-world objects.

02. Problem Definition & Design Thinking

2.1 Empathy Map

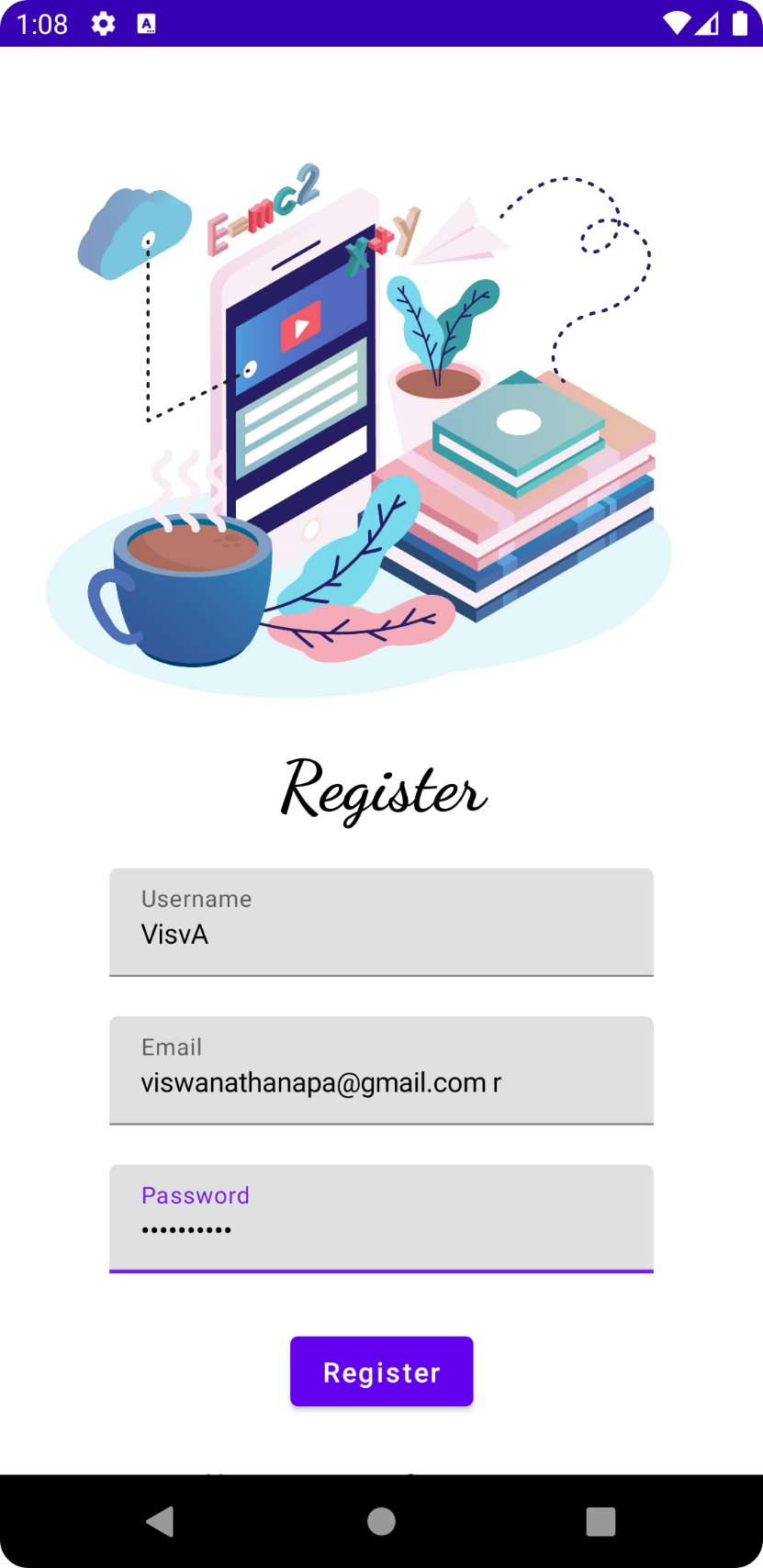


2.2 Ideation & Brainstorming Map

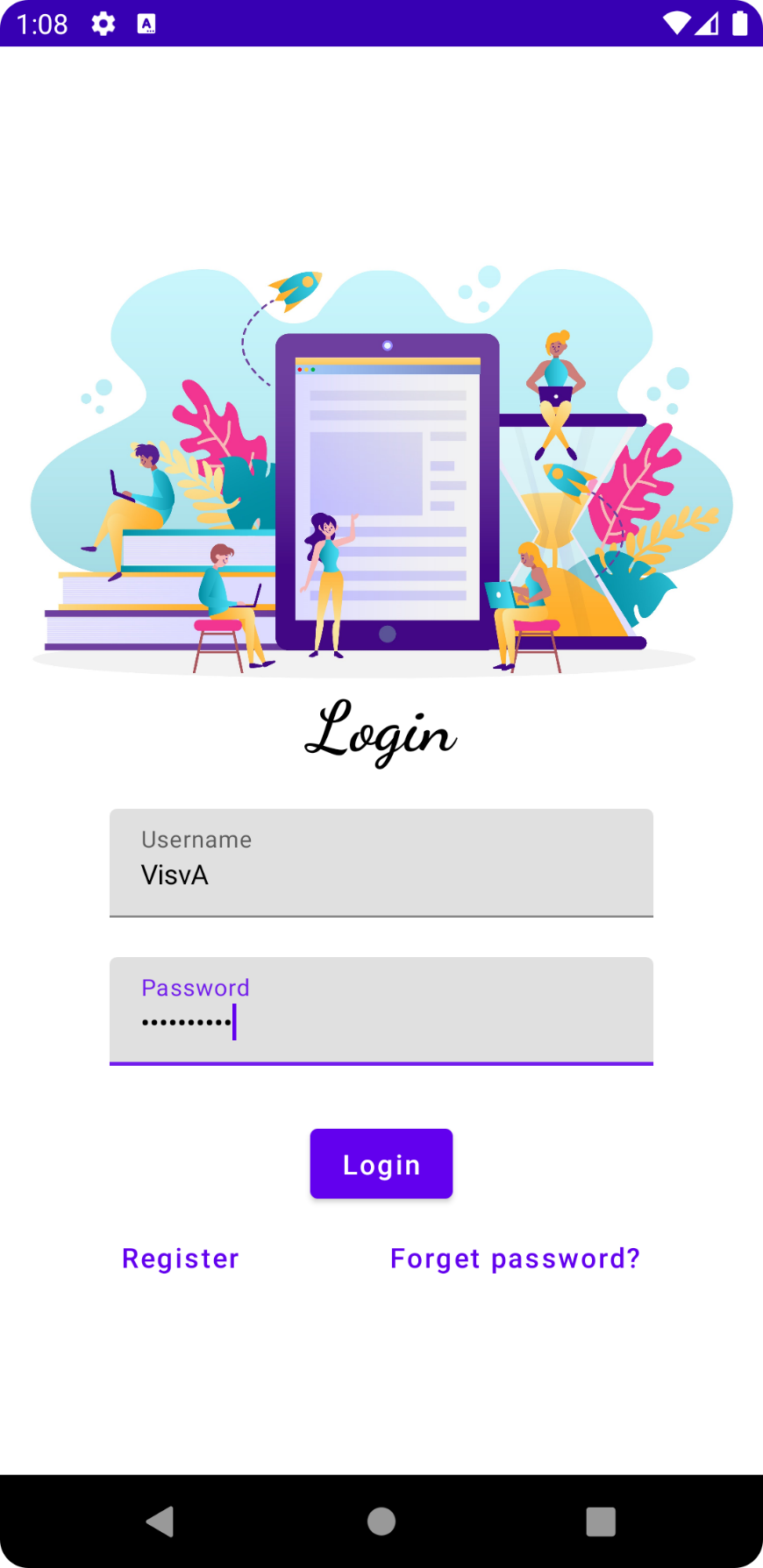


03. RESULT

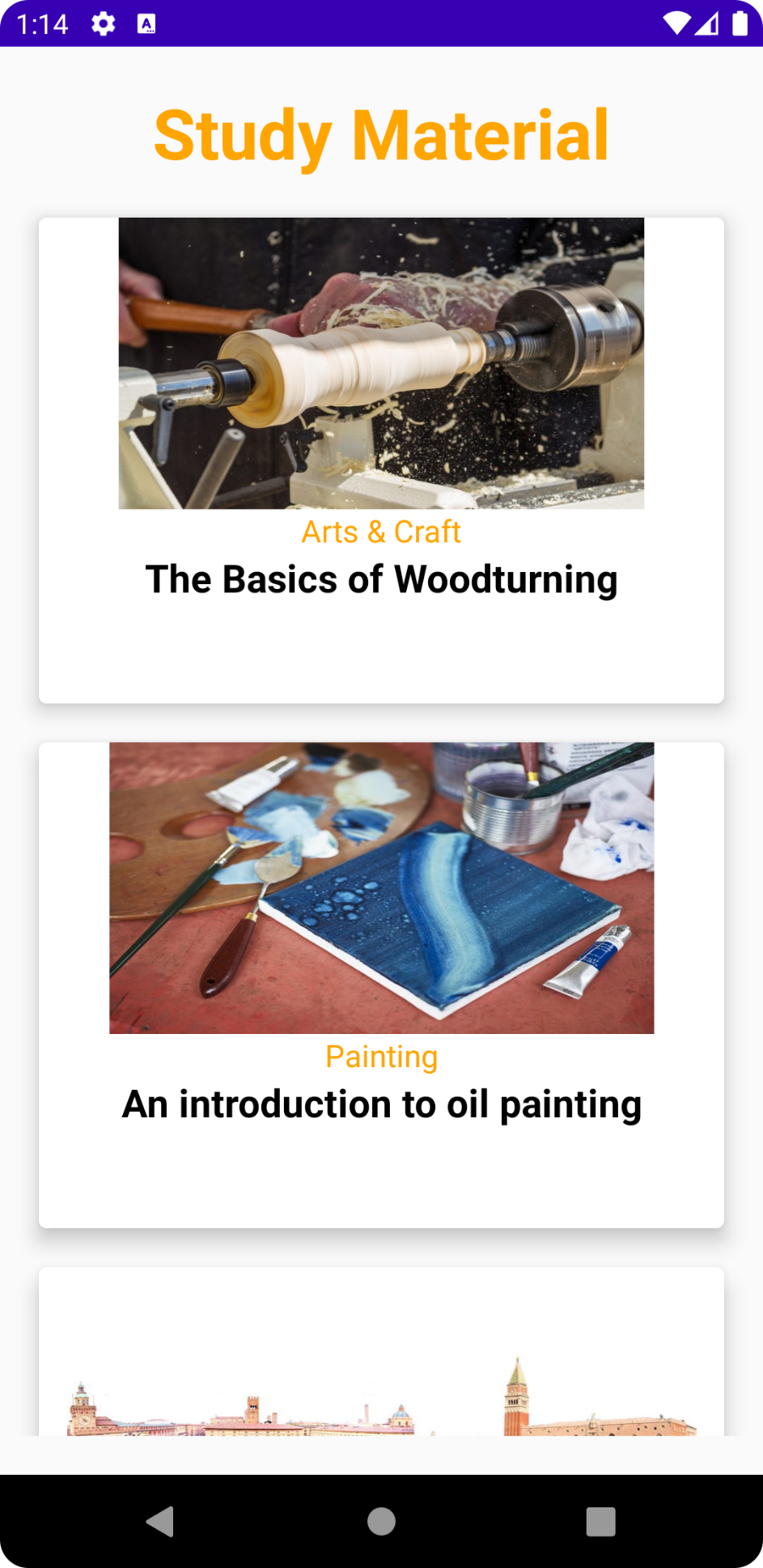
Register page :



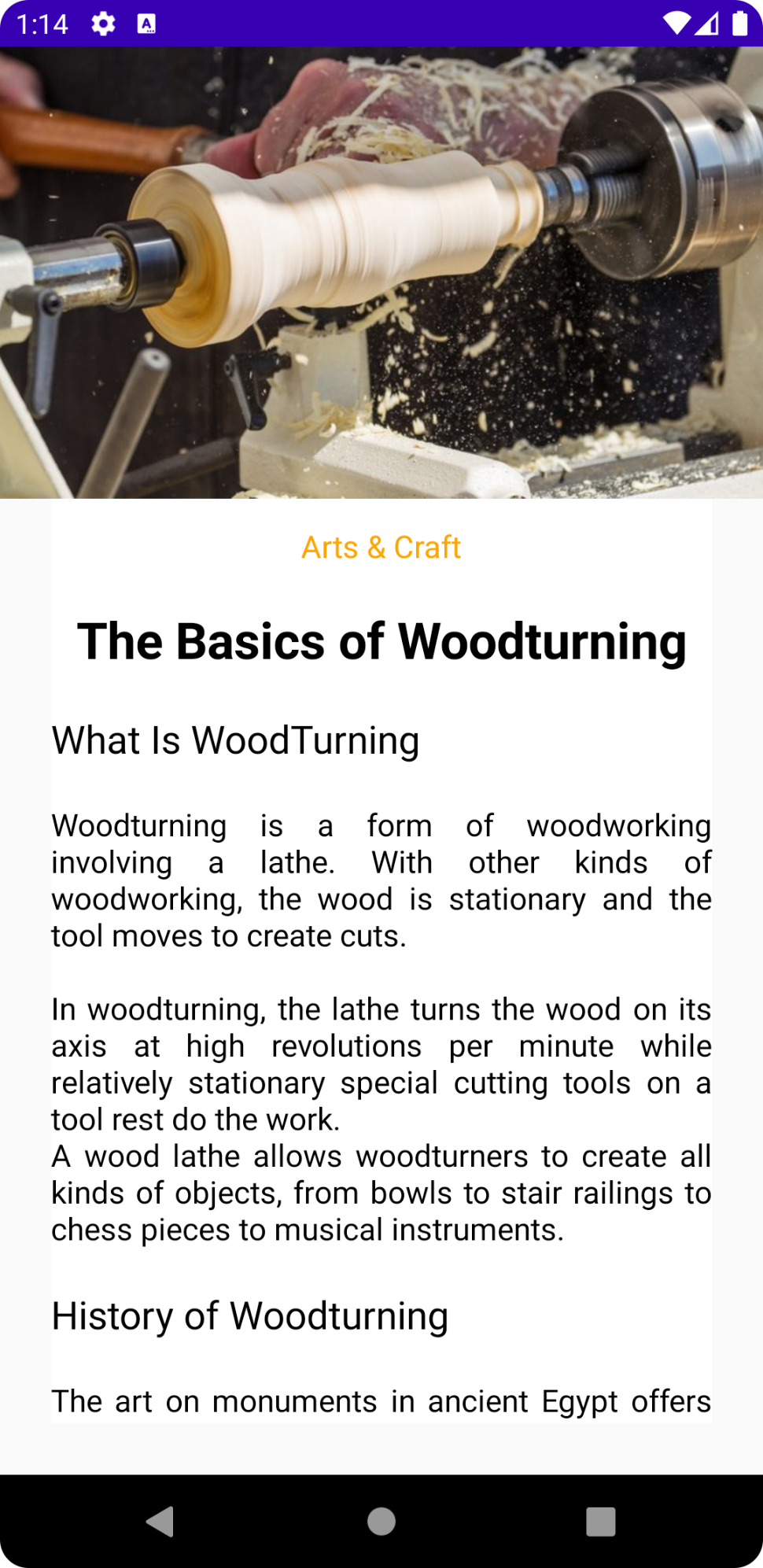
Login page :



Main page :



Book page :



04. ADVANTAGES & DISADVANTAGES

Advantages of Material Design Study

As with any well-established design system, there are some major pros to using a Material Design system that designers should consider.

Google’s Material Design is effectively an entire design ecosystem, rather than just a set of style guidelines. If there’s a potential design situation that exists, Material Design likely has a comprehensive set of rules for how to handle it. That includes complex use cases that are often overlooked by less comprehensive design systems. This can be very comforting for designers who want that kind of structure.

Google maintains Material Design and keeps extensive documentation for how to use and implement it. This kind of support and documentation can be lacking for many modern design systems. Despite all this comprehensiveness and documentation, Material Design remains a fairly flexible design library. Within the guidelines, much of the specifics of how to implement the design are left entirely up to the designer.

More granular advantages for Material Design include things like subtle skeuomorphism, which sets it apart from flat design and makes it more intuitive for many users. Another user-friendly feature is that user feedback in the form of haptic feedback, subtle animations, and similar things are built into the guidelines. It has a very simplified sense of physics, too, which makes interactions more intuitive. Material Design was built on a mobile-first sensibility, which makes sense considering its original purpose was for designing Android apps. It also promotes animation in designs, both for user feedback and to hint at how different features function.

Disadvantages of Material Design Study

While Material Design has very obvious pros, that doesn’t mean there aren’t cons that go along with using it.

First up, Material Design is immediately identifiable and is strongly associated with Google and, specifically, Android. While this isn’t necessarily a bad thing for everyone, it’s potentially a negative for some.

One big reason that it might be a negative is that it limits the effectiveness of other branding while using the Google design system. Yes, designers can incorporate logos, colour palettes (within the Material Design guidelines), and other differentiating factors to support the brand identity, but a product following the Material Design specifications will almost always also be associated with Google.

Since motion and animation are promoted within the Material Design guidelines, sites or apps that don’t incorporate it can seem to users as if they’re missing something. People associate the motion characteristics of Material Design with the visual characteristics, which can leave designs without motion lacking.

Sure, one solution is to always incorporate motion in designs that follow the Material Design specs. But extensive animations can be very resource-heavy on mobile devices, resulting in higher data usage and faster battery depletion. It’s a balancing act designer have to consider when working within the Material Design guidelines.

Beginners may find that the Material Design specification is more complicated and harder to implement than other styles like flat design. Because the Material Design system is so comprehensive, there are a lot more things to consider and adhere to than many new designers may be comfortable with.

Its comprehensiveness can also lead to some designers feeling constrained and unable to fully realize their own creativity. It can also stifle innovation, since virtually any design challenge has been planned for and solutions offered. While being helpful in many cases, this can prevent designers from taking new approaches to problems, while also limiting the number of new ideas that may occur.

There are also some usability issues in Material Design for websites and apps that can make them very user-unfriendly. One of the biggest issues is with the so-called “mystery meat” navigation encountered on many mobile design apps. Icons are often used in place of text, and while sometimes the icons are immediately recognizable and fairly usable, at other times they’re not.

A circle to indicate “Home” is significantly harder to identify than the house icon that was previously used in most Android interfaces. This is a prime example of placing form over function, which is a holdover from Material Design’s flat design roots.

And it’s not just in the lower navigation bar. Material Design’s preference for including circular floating action buttons is also a usability issue. These circular buttons only include space for an icon, with no assistive text included. And because icons can be so open to interpretation, in many cases, users are left questioning what these buttons actually do.

05. APPLICATIONS

The applications of material design study are numerous, and they can be found in many different industries and fields. Here are a few examples:

1.User interface design:

Material design is primarily used in user interface design for mobile and web applications. It provides a consistent visual language for buttons, menus, icons, and other interface elements.

2.Product design:

Material design principles can be applied to the design of physical products as well. For example, a product could be designed with a textured surface that creates a sense of depth and tactility, or it could use color and light to convey information.

3.Branding:

Material design can also be used as a branding tool. By using consistent visual elements across different platforms and materials, a company can create a recognizable brand identity.

4.Architecture:

Material design can be applied to architecture as well. For example, a building could be designed with a textured facade that creates interesting shadows and highlights throughout the day.

5.Fashion:

Material design principles can also be applied to the design of clothing and accessories. For example, a designer could use a textured fabric or a specific colour palette to create a cohesive collection.

Overall, material design study has a wide range of applications, and its principles can be applied to many different fields and industries to create a consistent and visually appealing user experience

06. CONCLUSION

If an app is being built primarily for the Android platform, then using Material Design is an easy choice. Because of Google’s widespread adoption, any app based on Material Design principles is going to feel like a native app.

That said, there are plenty of other use cases outside of the Android platform where Material Design is a solid choice. As the design system matures even further, those situations are bound to increase. Designers should, at the very least, familiarize themselves with the guidelines so that they can determine for themselves when it’s appropriate to use Material Design, and when other systems might be better suited.

07.FUTURE SCOPE

The future scope of material design study is very promising. As technology continues to evolve, there will be an increasing demand for designers who can create intuitive, visually appealing, and user-friendly interfaces across different devices and platforms.

Here are some potential areas of growth for material design study in the future:

1.Virtual and augmented reality:

As virtual and augmented reality become more popular, there will be a need for designers who can create immersive and intuitive interfaces that work seamlessly with these new technologies.

2.Internet of Things:

The Internet of Things (IoT) is a rapidly growing field that involves connecting everyday objects to the internet. Material design principles can be applied to create intuitive interfaces for these devices that are easy for users to navigate.

3. Wearable technology:

Wearable technology is another area that is expected to grow in the coming years. Material design principles can be used to create interfaces for smartwatches, fitness trackers, and other wearable devices that are both visually appealing and functional.

4.Artificial intelligence:

As artificial intelligence (AI) becomes more prevalent, there will be a need for designers who can create interfaces that can effectively communicate with and respond to users.

5.Sustainability:

Material design principles can also be applied to create sustainable products and interfaces that are environmentally friendly and reduce waste.

Overall, the future scope of material design study is broad and varied, and there are many exciting opportunities for designers who are interested in this field. As technology continues to evolve, material design will continue to be an important aspect of creating intuitive and user-friendly interfaces for a wide range of devices and platforms.

08. APPENDIX

Source code:

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

Create an UserDao interface

|  |
| --- |
| 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) |
|  |  |
|  | @Update |
|  | suspend fun updateUser(user: User) |
|  |  |
|  | @Delete |
|  | suspend fun deleteUser(user: User) |
| +`k65rz2 | } |

Create an User Database class

|  |
| --- |
| package com.example.owlapplication |
|  |  |
|  | import android.content.Context |
|  | import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [User::class], version = 1) |
|  | abstract class UserDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun userDao(): UserDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: UserDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): UserDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | UserDatabase::class.java, |
|  | "user\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

Create an UserDatabaseHelper class

|  |
| --- |
| package com.example.owlapplication |
|  |  |
|  | import android.annotation.SuppressLint |
|  | import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class UserDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) { |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "UserDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "user\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_FIRST\_NAME = "first\_name" |
|  | private const val COLUMN\_LAST\_NAME = "last\_name" |
|  | private const val COLUMN\_EMAIL = "email" |
|  | private const val COLUMN\_PASSWORD = "password" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "$COLUMN\_FIRST\_NAME TEXT, " + |
|  | "$COLUMN\_LAST\_NAME TEXT, " + |
|  | "$COLUMN\_EMAIL TEXT, " + |
|  | "$COLUMN\_PASSWORD TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertUser(user: User) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_FIRST\_NAME, user.firstName) |
|  | values.put(COLUMN\_LAST\_NAME, user.lastName) |
|  | values.put(COLUMN\_EMAIL, user.email) |
|  | values.put(COLUMN\_PASSWORD, user.password) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getUserByUsername(username: String): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username)) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getUserById(id: Int): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllUsers(): List<User> { |
|  | val users = mutableListOf<User>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | users.add(user) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return users |
|  | } |
|  |  |
|  | } |

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 |
|  | 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) |
|  | } |

Creating RegisterActivity.kt with database

Database connection in RegisterActivity.kt

|  |
| --- |
| 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) |
|  | } |

Creating MainActivity.kt file

|  |
| --- |
| 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.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 |
|  | 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()) |
|  |  |
|  | ) { |
|  |  |
|  | 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) |
|  |  |
|  | 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) |
|  |  |
|  | ) |
|  | }, |
|  | 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) |
|  |  |
|  | 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) |
|  |  |
|  | ) |
|  | }, |
|  | 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) |
|  |  |
|  | 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( |
|  | 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), |
|  | fontSize = 16.sp) |
|  |  |
|  | Text( |
|  | text = stringResource(id = R.string.topic4), |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 20.sp, |
|  | textAlign = TextAlign.Center, |
|  | ) |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |
|  | } |

Creating MainActivity2.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 |
|  | 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 |
|  | import com.example.owlapplication.ui.theme.OwlApplicationTheme |
|  |  |
|  | 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 |
|  | ) { |
|  |  |
|  | 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)) |
|  |  |
|  | 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), |
|  | 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 |
|  | ) |
|  |  |
|  |  |
|  |  |
|  |  |
|  | } |
|  | } |
|  |  |