

**OWL-M: A MATERIAL DESIGN STUDY APP**  
**A PROJECT**

**Submitted by**

**B.VENKATESH: 20201231506246**

**N.ULAGANATHAN: 20201231506244**

**M.SUBRAMANI: 20201231506236**

**M.MATHAVAN: 20201231506224**

**MENTOR**

**Dr.T.ARUL RAJ M.Sc.,M.Phil.,Ph.D**

*in partial fulfillment of the requirements for the award of degree of*

**BACHELOR OF SCIENCE**



**SRI PARAMAKALYANI COLLEGE**  
**ALWARKURICHI – 627 412**  
**APRIL - 2023**

## TABLE OF CONTENTS

Chapter No.	Title	Page No.
<b>I</b>	<b>INTRODUCTION</b>	
	1.1 Overview	
	1.2 Purpose	
<b>II</b>	<b>PROBLEM DEFINITION &amp; DESIGN THINKING</b>	
	2.1 Empathy Map	
	2.2 Ideation & Brainstorming Map	
<b>III</b>	<b>RESULT</b>	<b>13</b>
<b>IV</b>	<b>ADANTAGES &amp; DISADANTAGES</b>	<b>20</b>
<b>V</b>	<b>APPLICATIONS</b>	<b>23</b>
<b>VI</b>	<b>CONCLUSION</b>	<b>25</b>
<b>VII</b>	<b>FUTURE SCOPE</b>	<b>26</b>
	<b>APPENDIX</b>	<b>26</b>
	(i) <b>Source Code</b>	<b>26</b>

# I.INTRODUCTION

The Study Material Android Application is specifically designed for SCSVMV university students. This android application aims to help the students. In their academics by providing them their required study materials which are uploaded by the staffs itself so that no need for students to spend their valuable time for searching the study materials online

## 1.1 OVERVIEW

Material Designs is a design system that was created by Google in 2014. It is a set of design guidelines that was inspired by the card-based layout used in Google Now. What set Material Designs apart from the conventional Design Systems used back then was the fact that it was a paper-based design style that was distinctly different from the flat design style popularized by Apple.

As Material Designs gained in popularity, it began to serve as an element of branding for Google as it soon came to be identified as the signature UI of Google websites and app-based services. Today, Material Designs is more than just a Google UI, as it has been adopted by the wider design community across the globe. It has been established as an excellent choice for various types of design implementations, and defines guidelines for grids, typography, colour, space, scale and more.

Material Design Components (MDC Android) offers designers and developers a way to implement Material Design in their Android application. Material design in Android is one of the key features that attracts and engages the customer towards the application. This is a special type of design, which is guided by A Study Material. So in this article, it has been introduced to the basic things that need to be considered before designing or developing any Materialistic Android Application.

Study Material Design guidelines have become the signature look of their websites and apps. Still, there are plenty of use cases outside of A Study Material platforms where Material Design is also a solid choice.

Materials design for functional applications carried out using conventional trial and error methodologies is expensive and time consuming. Machine learning techniques can be used to design materials suitable for functional applications and discover materials for specific applications from existing literature.

This study demonstrates a machine learning approach where a combinatorial analysis was used to design perovskite structures and a novel methodology was

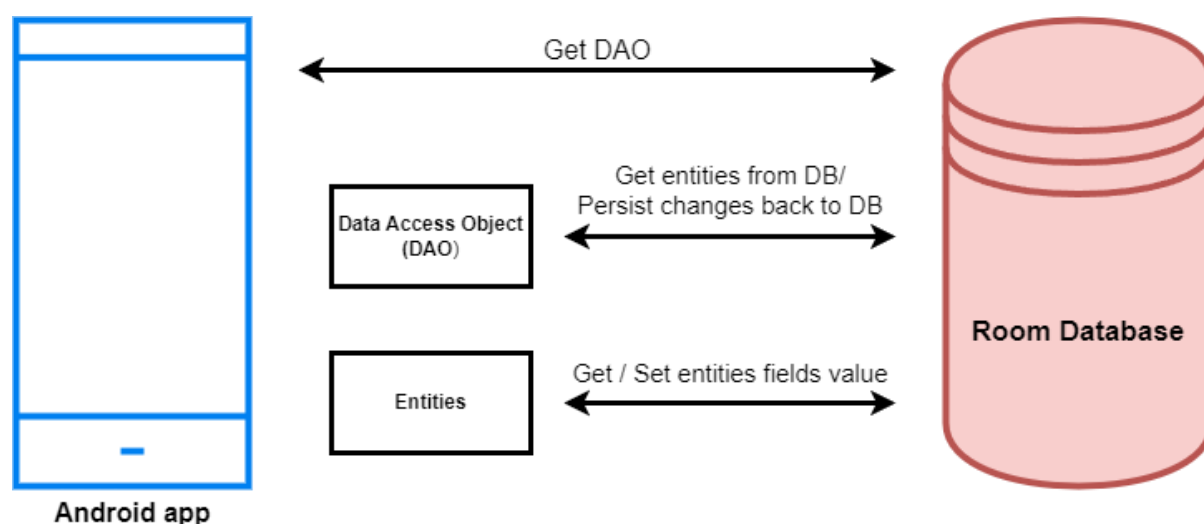
devised to calculate descriptor values which were used for property prediction of the designed perovskites.

New and improved materials have long fostered innovation. The discovery of new materials leads to new product concepts and manufacturing techniques.

## PROJECT DESCRIPTION:

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.

## ARCHITECTURE :



## 1.2 PURPOSE :

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. To use material design in your Android apps, follow the guidelines defined in the material design specification and use the new components and styles available in the material design support library.

The online applications for academic purposes are called online learning app. Such applications make use of an internet connection. An Online learning app can be accessed from a smartphone. It is a technology-based study tool that enables information sharing. It is commonly known as mobile apps for learning.

Online learning app is a great learning tool for students who do not have access to a physical classroom. It is useful for students who are working professionals and wish to improve. It allows students living in remote areas to attend classes.

This method of learning has many benefits like ease of education. An Online learning app is installed on the smartphone and can be used anytime. It gives access to both live sessions and pre-recorded classes to students. Trainers use audio-visual mode for teaching. Students engage in the online classroom and interact with other students very easily.

There are many online learning apps offering millions of courses. Many schools and colleges also it gives students the benefit of attending classes online in admissions. These allow students to choose subjects of their interest. There are both paid and free courses available to choose from. Students even get a certificate of completion when they finish a course.

The method of online learning apps is user-friendly. Students who attend the session have to mark their attendance. Students get assignments to complete within deadlines. Students can track their performances. They can even give feedback for the improvement of learning and teaching methods.

Now all schools, colleges, and tuition classes are adopting online systems, because it is safe for the health of students. E.g. Zoom and Google meet are being used in many places. A simpler way than that is to develop mobile apps. Now everyone has Smartphones and the proper internet connection, so having an educational app for online learning for schools, colleges and tuition classes is very important for both teachers and students these days.

The main purpose of Material Design is the creation of a new visual language that combines principles of good design with technical and scientific innovation. Designer Matías Duarte explained that, "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch." Google states that their new design language is based on paper and ink but implementation takes place in an advanced manner

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

When you begin changing aspects of your UI, such as color and typography, Material Theming tools apply your design vision throughout your user experience. These tools allow easy switching between design and code workflows by providing specific values for all customizable attributes. Customizing these values creates a Material Theme for your product.

Material Theming consists of three main actions: customizing your theme, applying it across your design mocks, and using it in code.

Material Design comes designed with a built-in, baseline theme that can be used as-is, straight out of the proverbial box. You can then customize it as little, or as much, as you like to make Material work for your product.

We all remember the good old websites. They were extremely light, loaded fairly quickly (especially given technology back in time), and delivered amazing experiences according to the standards in the 90s and in the 00s.

But Internet has emerged and what it looked like space to gather information has rapidly monetized. Once the money started to play a huge role online, websites started to compete against each other by adding lots of images, subsections, long descriptions, and loads and loads of stuff, making sites load slower and creating an inferior user experience.

The rise of smartphones created another problem – how website content will appear on a smaller screen? As a result, Google introduced Material Design – an answer to the ever-increasing user demands.

In this article, we'll provide a material design definition, go over the main principles, and see how icons and colors are used in material design. We will end the article with some useful resources you can apply to your projects.

For a design system – and any pattern, in general, is crucial to have a specific checklist of requirements, in order to be qualified as such. Google Material Design guidelines are there to help you. Below, we will show you some principles you need to apply in your design, so it can be regarded as “Material”.

Material Design has its name for a reason. Unlike Flat Design, here we see a completely different concept. A concept that uses visual hierarchy to create a real-world feel to designs. Human beings can make associations. Material design heavily relies on 3D forms, to add more depth and realistic user interfaces.

Shadows are a crucial element across all Material Design components. They should be carefully chosen, as they aim to improve user experience. Using shadows will better orientate users about their current location on your app or website. Bear in mind all shadows should be subtle and non-intrusive.

The Material Design color palette is specific. First and foremost, you should use a maximum of two main Material Design colors – one primary color and one accent color. You're also limited to 3 hues per each. Since you're not left with much “room” for maneuvers, it is considered pivotal to use bold colors to capture attention.

Material Design is all about creating a consistent user experience. This means you should follow the same hierarchy across all pages – same hover button effects, same on-click colors, same site structure, and page colors. Your users should intuitively

select components from the website/app and easily know where they are at any moment.

Images also play an important role in the whole “picture” (pun intended). They should be aligned perfectly, appear in full HD quality at least, and be edge-to-edge. The latest is key to the success of the whole concept. If images are a part of your design, they should also define the primary and accent colors, so all hues fit naturally in the design.

In Material Design, whitespace plays just as important a role as components, colors, and shadows. Why? Because utilizing it could significantly improve readability, thus creating better UX. Negative spaces also serve another purpose – they can put the focus on the desired element, for example, the call-to-action button.

Motion design leverages user experience by creating interactions between the content producer and content consumer. By providing visual feedback upon each action taken by your users, you drive more engagement. Each Material Design animation should be intuitive, smooth, and authentic. For example, if users click backward, they should see their screen going from left to right, as this is their natural comprehension of turning back pages.

Each component you insert into your user interface serves a specific purpose. One of the main requirements is to use components in their respective field – if you have a roadmap, having a progress map might make great sense. But will it make sense on your Pricing page? A rule of thumb is – if you’re not sure you need it, avoid it.

Material Design lays the foundation of the new wave in UX/UI design – minimalism, and it follows Google’s concept – “less is more”. If your goal is to utilize Material UI Design, then you should get rid of unnecessary elements and components, and leave room for what truly matters.

Google developers are aware not everyone is born a graphic designer, that’s why they have developed their own **color matching tool** which allows you to select your primary color, and then automatically generates the additional hues which match it. The same logic is applied to the secondary color.

## II.PROBLEM DEFINITION AND DESIGN THINKING

### PROBLEM DEFINITION:

The group or individual understand and can prioritize the current challenges that they require to improve

### DESIGN THINKING:

Design thinking is a structured method of problem solving that follows three stages: collaboration, innovation, and acceleration.

## 2.1 EMPATHY MAP

An empathy map is a visualization that captures our understanding of a set of users. It is a simple yet powerful tool for making sense of user research and communicating what we know about our users to design-team partners and stakeholders so that we can discuss and prioritize user needs collaboratively.

Traditional empathy maps have a 4-quadrant structure, with equal space devoted to capturing what the user:

**Says:** Direct or paraphrased representative quotes from users

**Thinks:** Users' thoughts, motivations, and needs framed in first-person statements

**Does:** Typical or observed user behaviours and actions

**Feels:** Users' emotional states during the process

While it's fairly easy to grasp the simple format of an empathy map, some teams struggle to figure out how the tool fits into the overall design process. When is the best time to create an empathy map?

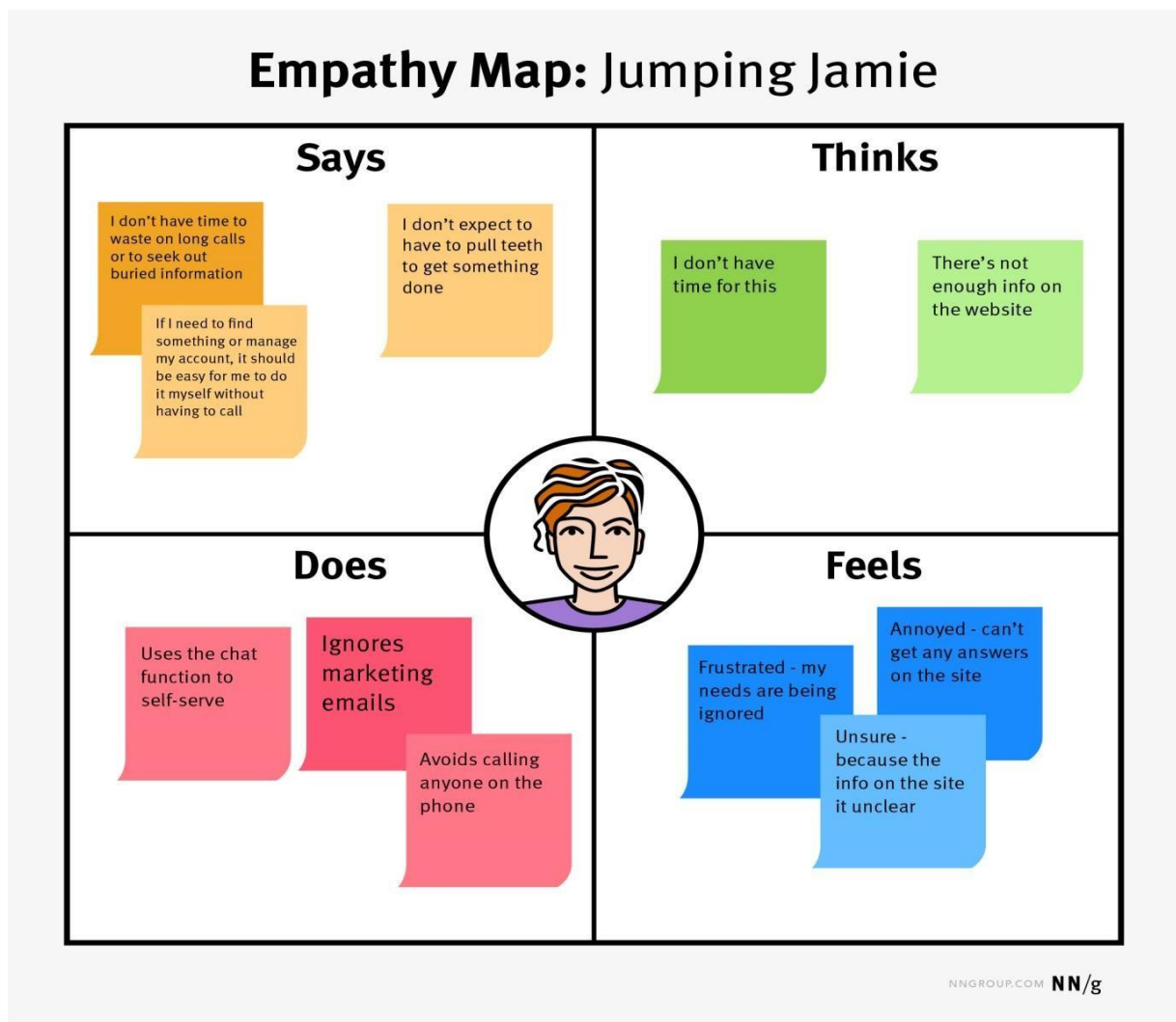
Empathy maps are flexible. They can be useful at various times throughout the design process, including:



**Before research** has happened, to plan and shape future studies

**During research**, to capture users' needs, attitudes, and experiences

**After research** has been conducted, to communicate research findings and build understanding of our users



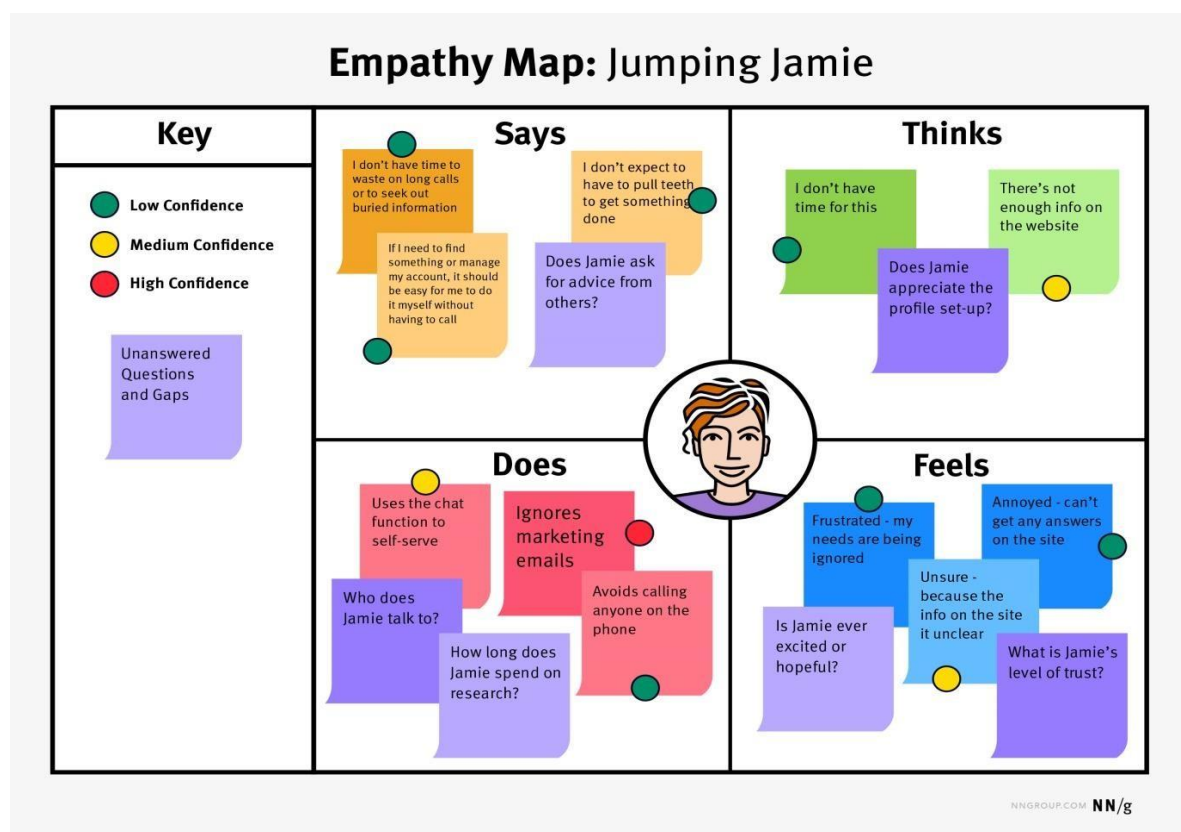
## Empathy Maps to Plan Future Research

Many people primarily think of empathy maps as a method for documenting insights from research that has already happened. While that application is useful, empathy maps can also be a powerful tool for planning and shaping *future* research endeavors.

When planning future research, teams can use the same 4-quadrant structure to assess gaps and unknowns about a particular user group. During a collaborative activity, direct the team to document what it currently knows about the user group in focus. The team should devote time to each quadrant, documenting things that team members believe users say, think, do, and feel. Place each known item on its own sticky note. Those sticky notes are then assigned to the relevant quadrant on the map.

If no new research has been conducted in quite some time (or ever), consider assigning a confidence rating to each item — with 1 = low confidence, 2 = medium confidence, and 3 = high confidence.

After capturing current knowledge, the team can step back and reflect on what's missing, asking: Where are there gaps? What don't we know? What are we most unsure about? The team could use the map to document these questions, which should inform future research studies.



## 2.2 IDEATION & BRAINSTROMING MAP

### IDEATION:

The Ideation First Burst Map is a visual map of the initial ideas and solutions that are identified following research and empathy building during the Inspiration phase of design thinking.

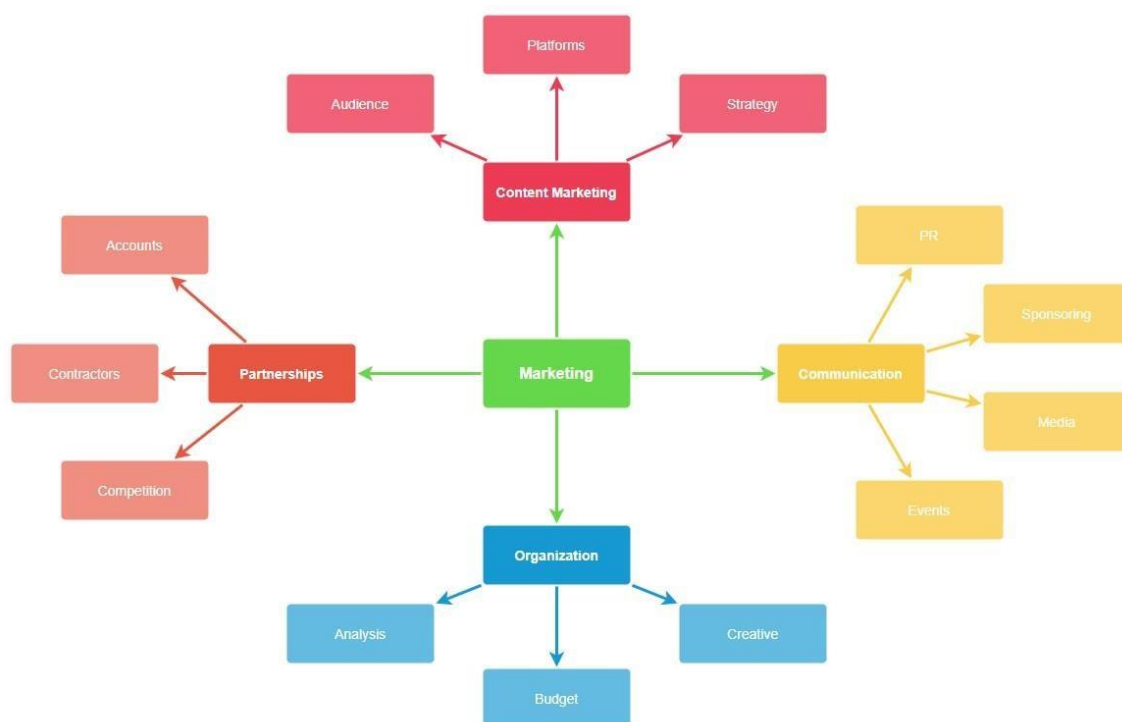
### BRAINSTROMING MAP:

A mind map is a visualization technique and brainstorming tool which allows you to explore central idea a, and all of its related topics, in a non-linear way

A mind map is a visual diagram that helps you organize or explore related ideas, categories, and concepts.

Many of you might remember it from school, breaking down our ecosystem and highlighting the relationships between different animals.

But don't worry. That's not the only use case. For example, you can use it to break down all the different potential factors that impact marketing in a company.



It takes the main topic, marketing, and breaks it down into some of the main focus areas of that business.

A mind map helps you grasp the bigger picture with a single glance. For example, imagine if I tried to list out all the items using an indented bullet list.

With a single category and only a few sub-items, it works.

But imagine scaling this up to 10 categories, many with more subcategories than 3. Not only will it be a useless visual reference, but you can't show the interrelationships between sections.

There's a reason mind maps are the go-to for visualizing objects with tons of moving parts.

You can add pictures to highlight the different concepts and make the mind map even more effective. That helps your brain internalize all the moving parts much more efficiently.

Having all the moving parts on paper isn't just useful for categorization. It can help you focus your thinking, make new connections, and come up with high-impact ideas.

Even though mind maps started as a memory or communication tool, they didn't stop there.

Nowadays, many professionals rely on mind maps for ideation and brainstorming. They are the perfect vehicle for finding ideas when there are a lot of factors involved.

Flashback to the statistic that we mentioned earlier — more than 8 in 10 people find that mind maps help them solve complex tasks.

Mind mapping is a powerful instrument for finding ideas. It's because of the same reasons that make it such a useful tool for learning.

You can clearly list, review, and internalize multiple factors and the relationship between them.

That makes mind mapping the perfect option when you need to make complex decisions.

That's why the career centre at Duke University recommends using mind maps. Deciding on a career path is a big decision with a lot of factors to consider.


Enter the mind map. It's the best way to create a high-level overview of your life. If you just tried to write everything down on paper, you'd naturally create something similar.

## III.RESULT

### REGISTER PAGE

User register into the application

12:36



## Register

Username

Email

Password

Register


Have an account?

Log in

# LOGIN PAGE


After registration, user login into the application

12:36

A colorful illustration featuring a large tablet displaying a login form. Several stylized figures are interacting with the tablet: one is sitting on a stack of books to the left, another is standing and pointing at the screen, and a third is sitting on a chair to the right. The background includes abstract shapes, a rocket, and various plants.

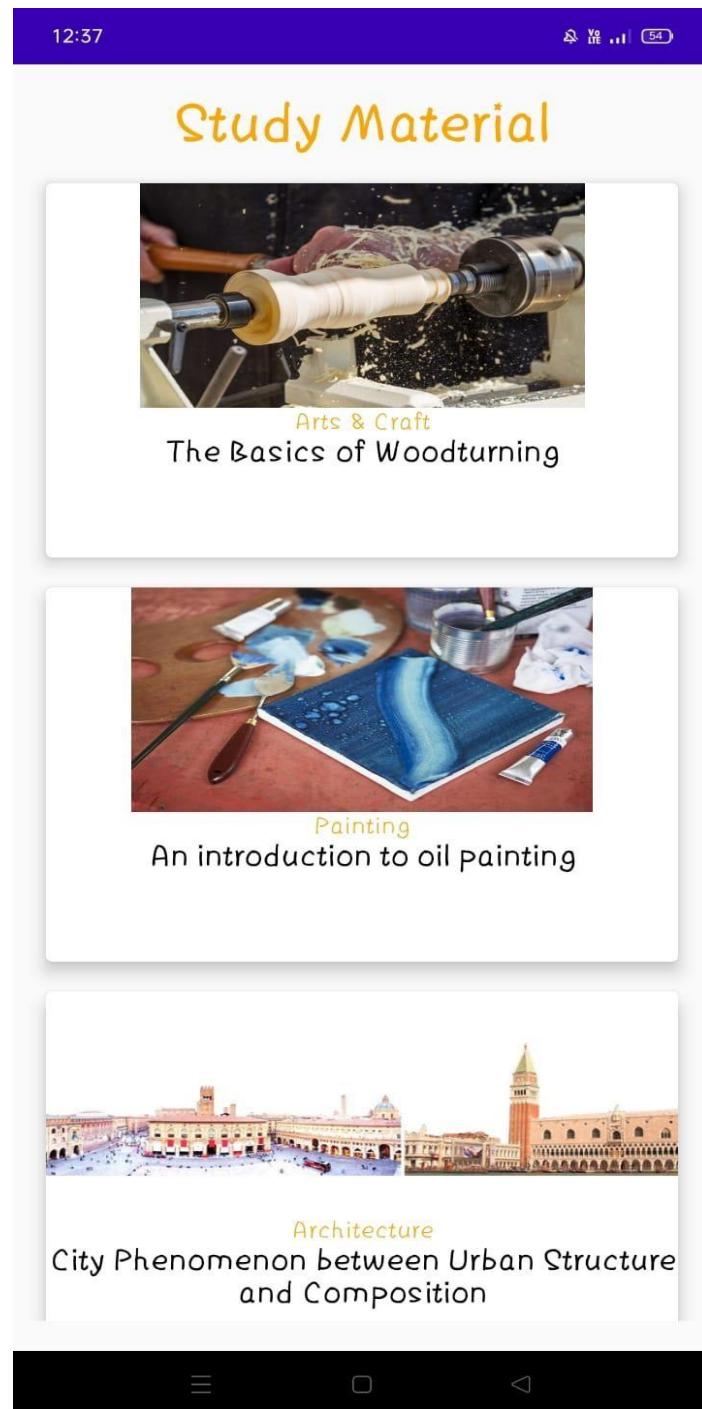
## Login

[Register](#) [Forget password?](#)

A black mobile navigation bar with three white icons: a hamburger menu, a square, and a back arrow.

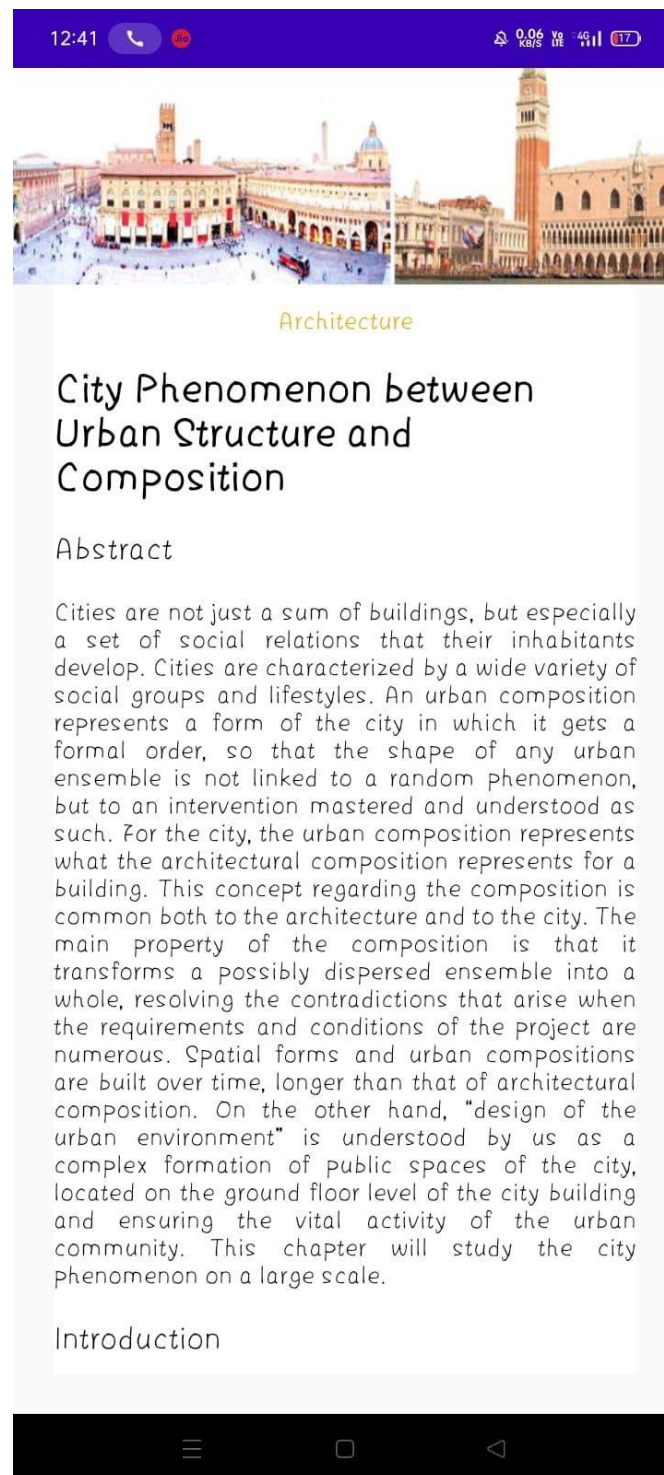
## MAIN PAGE

User enters into the main page



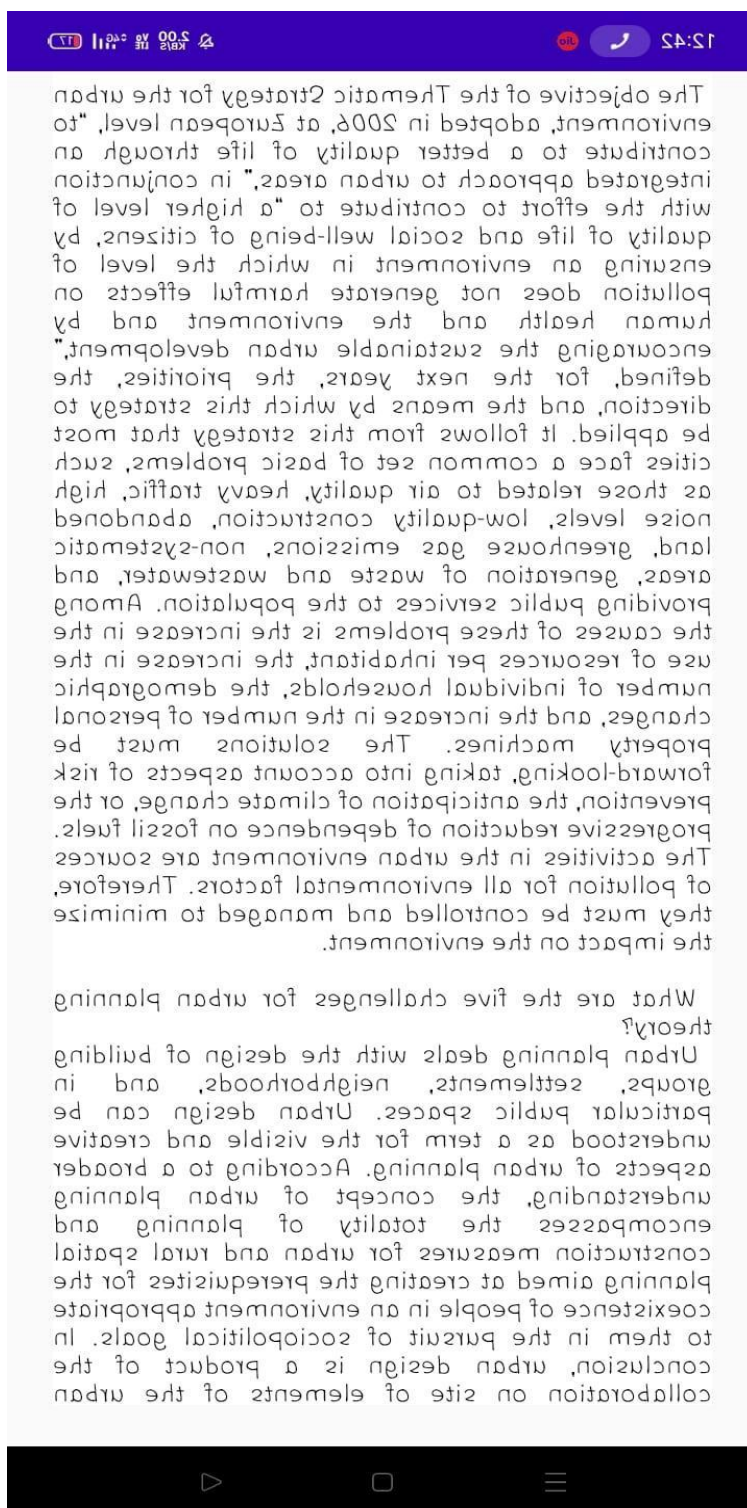
# CONTENT PAGE

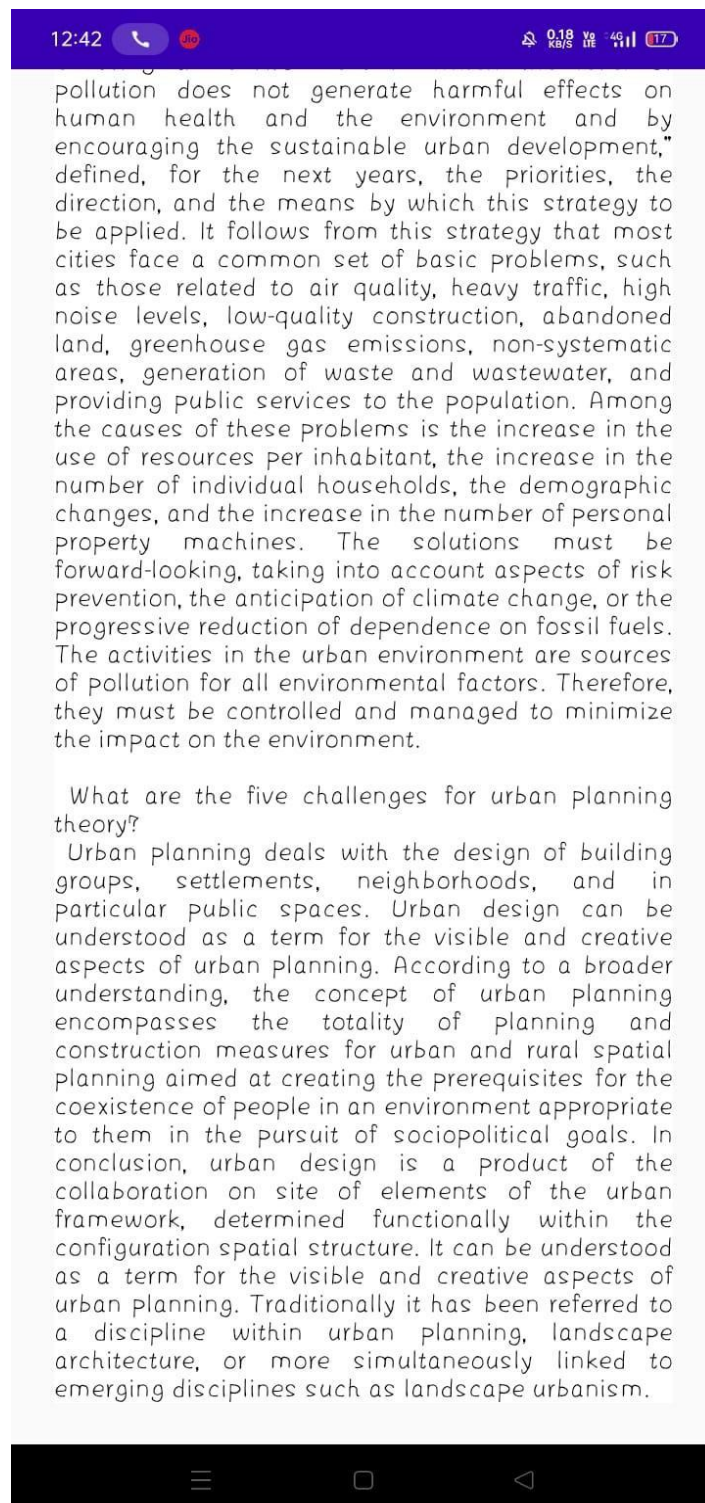
User can see the content page











## IV.ADVANTAGES & DISADVANTAGES

### 4.1ADVANTAGES

In the wake of the global pandemic, the education sector has seen a massive expansion of technology. With technology on its side, the entire industry is discovering new ways of doing things. It's not that technology has not been used in education before, but the use of educational applications has been limited.

Using technology used to be a choice, but now it's a requirement. This has led to the adoption of educational software development via mobile applications, allowing companies, particularly in the education sector, to reach new heights. During compulsory distance learning, it was evident that there was a huge demand for technical tools and systems that allowed professors to communicate with their students, track their learning progress, and distribute their courses.

#### *1. Graceful animations*

Material designs come with smooth, engaging, and flexible UI transitions and animations, making it easier for users to understand the relationship between different elements on their devices. Animations not only clarify the relationship of UI components but also help craft user's attention at specific points. For example, movement from one screen to another or hovering over/clicking any element on your web page can be used to highlight certain elements such as images or buttons, etc.

#### *2. Responsive layout*

It offers fast-scrolling screens and quick content updates, which are important for the mobile age. The use of fast page-switching animations helps users navigate through different screens without having to wait for the screen to load. The 'card' layout design makes it easier to scroll vertically or horizontally, whereas elements inside each card can be pushed across the boundaries of their parent cards with ease.

#### *3. Self-contained UI components*

Material design introduces self-contained UI components that are much easier to maintain and recognize. This not only speeds up coding but also creates consistency in the user experience. These reusable UI components come loaded with preset API styles that are ready to be plugged into your website or app's development framework. Therefore, all you need is just a little knowledge about how these API codes work, and you are good to go!

#### 4. Adapts well to all screen sizes

You do not need to worry about how your website or app will look on any device. Material Design takes care of the responsiveness part, providing your app with a uniform look, so users can navigate through their screens easily no matter what device they are using. In addition to this, Google has recently introduced material template theming which helps developers get started by choosing from a set of pre-designed UI components and customizing them according to their needs. This method has been designed specifically for better responsive designs and faster coding.

#### 5. Less time required to create animations

By standardizing the way different elements should be animated inside an application or webpage, material design speeds up the process of making animations. The use of common animation time scales for all elements inside the app or webpage makes it easier to create equilibrium between different components, saving a lot of development time in the process.

#### 6. Material design is flexible

Developers can easily customize this new design language by using color, typography, and graphic elements that best suit their needs. This allows developers to choose the right material design that fits perfectly into their existing UI or UX patterns. Material Design can also be integrated with any CSS framework like Bootstrap, Foundation, etc., which means you can use Google's official material theme along with third-party plugins like jQuery, JavaScript, etc., resulting in better-animated interfaces for your mobile apps or websites.

## 4.2 DISADVANTAGES

While Material Design has very obvious pros, that don't mean there aren't cons that go along with using it. First up, Material Design is immediately identifiable and is strongly associated with A Study Material 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 A Study Material design system. Yes, designers can incorporate logos, color 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 A Study Material. 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 designers 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.

### *1. Too many colors*

Because the interface consists of many objects and different components, it becomes difficult for users to look at screens that contain too much color; especially if they're older or color blind.

People might think that this isn't a problem since there is an option for users to change their theme to suit what they prefer (i.e., light, dark, etc.), but sometimes people don't like having lots of options or choices; they would want the site/app to remain simple and minimalistic, without too much hassle.

### *2. The animations*

The objects which the material design uses move to different positions on the screen when a user performs an action such as scrolling down or clicking on an object; this can be distracting for users who aren't interested in what's moving around them. Some people might find that it makes their eyes hurt after a while. It may also not be suitable for people with epilepsy or other conditions that could affect how they perceive movement.

### *3. Too many distractions*

Material design is good in some ways because it allows users to interact with content easily by performing certain actions (such as swiping left or right), but there are times when features like these become too distracting and make things difficult for users to focus on what they're actually trying to do.

## **V.APPLICATIONS**

### **1. Enhanced Interaction**

Experts say that apps in education can make children more interactive and activate better engagement between parents and children. The most effective way is to engage with the children while they are using applications. Interaction tendency in children is enhanced by mobile applications.

### **2. Novel learning techniques**

Thoughts of traditional methods of learning accompany a generic feeling of boredom. They do not favor drifting from the monotonous learning patterns of restricted and upright book learning, thus dissipating the engagement factor.

Technology in the guise of apps is helping those looking for some newness in the universe of learning. In addition to the feel of novelty, apps add an element of fun and involvement to the learning process. Through games, puzzles or other challenging tasks, app learning stimulates the brain cells to actively metabolize the input unleashing a new perspective.

### **3. Parent-teacher communication**

The ideal concept of frequent parent teacher interactions finds its space in the articles and books regarding performance enhancement but not in reality. Owing to the tight schedule of both the parties, it is just not possible to maintain the rapport through physical interactions. But now, we have apps. Teachers can attend to the queries of the parents anytime and anywhere through an ominous device called the phone. This fosters transparency regarding the child's growth at school.

### **4. Online resources**

The power of digital world lies in the ginormous amount of resources that fill its nooks and corners. The wealth of this platform implicates its popularity among knowledge seekers. The reach of this platform makes it a favourite to people who cannot afford the luxury of full

time courses in schools or colleges. Mobile applications help them access a compendium of eBooks and pdfs and other online materials and the freedom to access it beyond the boundaries of time and space.

## **5. Entertainment**

According to studies, mobile apps promote entertainment. Learning is no more a passive activity, it's active with applications. Lessons transforming to games can change the face of education. Children will enable a kind of interest in learning. Level based apps instil determination to pass each level. Apps without doubt enhance education. *No more boring home works and tough class lectures.*

## **6. Availability 24/7**

Unlike school, mobile applications are available round the clock. No need to be worried about schedules. Anywhere can be a classroom. App learning is not time-bound learning, its relaxed learning.

## **7. Leisure Hours Utilization**

No responsible parents want their kids to get addicted to the "idiot box". Too much internet usage and talking over the phone for hours are not wise options for killing time. This is where mobile apps prove their worth. Mobile app learning is one among the wisest choices of utilizing your free time actively.

## **8. Routine tasks**

It's a relief to get all the mundane tasks done with a few taps. Be it tasks like fee payments, other transactions which require us to stand in a queue for hours or the laborious job of marking attendance that drives teachers crazy with the amount of paperwork smiling back at them each day. All this drudgery has been put to an end simply by having apps in place. The life of each individual associated with the ecosystem is now simple and functioning, more efficient.



### **9.Filling in the gaps**

The wheel of time has spun to drive the progress to land us into the world we live in today.

The advancement that schools have seen eliminated a lot many glitches that prevailed in the education system. A major one being the lack of interaction between the teachers and the students. Apps and websites have been created to help reduce the gap not just between the students and the educators but also among parents and the teachers. Students and parents can be kept in the loop of every event, schedule change or announcement.

### **9. Better Earth**

While millions of trees are cut down for making papers for the traditional method of learning, mobile apps in education requires just a download. It means a greener earth for future generations.

Mobile learning process has sustainability. Completing a lesson with an app is much more effective as it is learning from experience rather than from compulsion.

### **10. Systematic Learning Activated**

Smart learning is one thing and systematic learning is next. App based learning enables both. Mobile apps help in systematic learning. Apps are arranged in such a way that, it promotes not only a craving for learning but systematic learning.

## **VI.CONCLUSION**

Material Design is a great concept. It creates consistent user experiences and simplifies website and app design, ensuring all users will easily access buttons and layouts. While no design system is perfect, having common standards that will reduce bad experiences is always a good idea!

## **VII.FUTURE SCOPE**

It is very essential to keep the material organized, ordered, and in a systematic manner to avoid any trouble at the workplace. Therefore, managing material in an organization requires keeping a record of all the raw material, storage, control, distribution, and supervising all the processes.

Companies are always in need of skilled professionals to perform these tasks and therefore hire professionals, skilled material managers, to oversee all these operations for smooth handling and functioning of the organization.

As a Material Management Professional has the capability to plan and buy goods and regulate inventory in order to satisfy the company's goals. These are professionals also have supply chain and inventory control experience.

They collaborate with other managers to determine supply requirements and handle purchase supplies and materials in accordance with standards.

Moreover, they also maintain favourable relationships with suppliers and keep accurate records of procurement activities, material quantities, and requirements.

## **VIII.APPENDIX**

### **8.1 SOURCE CODE**

The source code for the above project was given in the Github link

<https://github.com/Venkat310502/androidstudyapp>

## 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,
        onChange = { username = it },
        label = { Text("Username") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )
    TextField(
        value = email,
        onChange = { email = it },
        label = { Text("Email") },
        modifier = Modifier
            .padding(10.dp)
            .width(280.dp)
    )
    TextField(
        value = password,
        onChange = { 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)
}

```

## LOGINACTIVITY.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 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)  
}
```

## MAINACTIVITY.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.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(0xFFFFFA500),
            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,
        )
    }
}
}
}

```

## MAINACTIVITY2.KT

```

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(0xFFFFFA500),  
    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

    )

}

}

```

### MAINACTIVITY3.KT

```
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

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



```

    }
}

@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))
        Text(
            text = stringResource(id = R.string.course2),
            color = Color(0xFFFFFA500),
            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(
        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),
        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
    )
}
}

```

## MAINACTIVITY4.KT

```

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
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() {

    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),
        color = Color(0xFFFFFA500),
        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),
        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
}
}
}

```

## MAINACTIVITY5.KT

```
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
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 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())

        .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(0xFFFFFA500),

            fontSize = 16.sp,

            modifier = Modifier.align(Alignment.CenterHorizontally)

```



```
)  
  
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))  
  
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
    )
}
}

```

## USER.KT

```

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?,

)

```

## USERDAO.KT

```
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)
}
```

## USERDATABASE.KT

```
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

        }

    }

}

```

## USERDATABASEHELPER.KT

```

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
    }

    @SuppressWarnings("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
    }

    @SuppressWarnings("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

}

}
```