Android Application Development

“Mind Vault Study App: Your Essential Companion for Smarter, More Effective Study Habits"

Team ID: NM2024TMID05648

Submitted By

Madhumitha D(Team Leader) F96080CD1FBD8CC9B3D48F30210C7292

Archana M(Team Member) 8AAA163B70660F6BC3342472107CD6E6

Aravinth N(Team Member) ED3BB668FA3B5D5B44F47CC735BE80C5

Koshiya P(Team Member) E0D21BDC52CE3BC9153406B7088E9E95

Thirumalini T(Team Member) 03AABE6A24707D0922643021D380F0F0

Under the Guidance of mentor

DR. M.Fareena

ASSISTANT PROFESSOR

Department of Computer Science

Anna University Regional Campus Coimbatore(Code 7100)

Coimbatore 641046

DEPARTMENT OF COMPUTER SCIENCE

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE

COIMBATORE 641046

TABLE OF CONTENTS

1) Overview
2) Architecture
3) Project Workflow
4) Problem Statement
5) Solution
6) System Requirements
7) Testing Requirements
8) Network Requirements
9) Application Screenshots
10) Applications
11) Project Hurdles
12) Conclusion
13) Future Scope
14) Source Code

1)OVERVIEW

The goal of the productivity tool Mind Vault is to assist users stay mentally sharp, increase cognitive function, and improve focus. Mind Vault serves as a complete mental wellness companion by providing a range of effective tools and features that are intended to improve focus and mental clarity. The main purpose of the program is to assist users in developing a focused, disciplined attitude to work so they remain attentive and productive all day long.

The main feature of Mind Vault is its Focus Mode, which helps users get rid of distractions by muting pointless apps, disabling alerts, and avoiding interruptions. For those who must fully focus on job, education, or artistic endeavors, this function is especially helpful. Mind Vault facilitates sustained focus and allows people to work at their peak for extended periods of time by establishing a distraction free environment.

Mind Vault provides comprehensive progress tracking and productivity insights to go along with Focus Mode. It automatically records session lengths and activity levels, giving users information on how well they've focused and handled their time. Users may monitor trends, spot gaps in productivity, and make data driven decisions on how to enhance their mental performance with the app's performance reports. In addition to increasing awareness, this degree of surveillance enables users to improve their work habits for better outcomes.

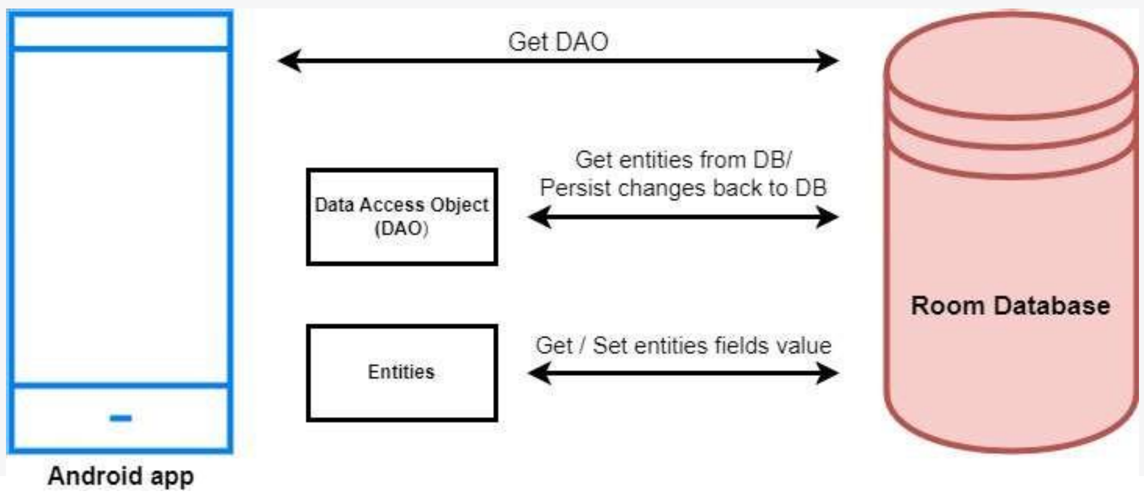
Mind Vault takes a personalized approach to user experience, offering tailored recommendations based on individual patterns and usage data. As the app learns from your focus habits, it suggests tips and strategies to improve mental clarity, sustain attention for longer periods, and eliminate distractions. These personalized insights ensure that each user gets the most out of their time, helping them refine their workflow to suit their cognitive strengths.

Jetpack Compose was used to create Mind Vault's clean, contemporary interface, which was designed with simplicity and use in mind. Even the most tech averse users can simply explore the app's features and settings because to its user friendly design. Whether you're monitoring progress, triggering Focus Mode, or tracking productivity, Mind Vault offers a smooth and entertaining experience.

Mind Vault also has gamification components for people who wish to maintain motivation and engagement. In addition to reinforcing productive habits, achievements, badges, and progress milestones provide a sense of success. These components make productivity more enjoyable and fulfilling, which inspires individuals to maintain enhancing their mental health and concentration.

In the end, Mind Vault is a mental performance booster that helps users track their cognitive output, maintain attention, and improve their work habits. It is more than just a productivity tool. Mind Vault is the ideal tool to help you sustain high levels of focus, get better outcomes, and realize your full cognitive potential—whether you're working, studying, or just trying to stay mentally sharp.

2)ARCHITECTURE


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

3)PROJECT WORKFLOW

- **User registration:** Users provide their personal information and create an account to access the study materials.
- **User login:** Once registered, users can log in to the application using their username, email and password to access the study materials.
- **Main page:** After logging in, the user is directed to the main page of the application, where they can access the courses.
- **Study Material:** The user can access the course and they can complete the course based on their interest.

4)PROBLEM STATEMENT

Despite widespread recognition of the benefits of consistent study habits and effective learning strategies, many students and lifelong learners struggle to find accessible, user friendly tools to support their educational goals. Current solutions often lack personalization, are overwhelming in complexity, or fail to provide actionable insights that help users track and improve their study habits over time. This results in ineffective study routines, reduced motivation, and diminished learning outcomes. The Mind Vault Study App aims to bridge this gap by offering a simple, engaging platform that helps users monitor their study patterns, set achievable goals, and receive personalized insights to enhance their learning experience.

5)SOLUTION

The Mind Vault Study App was created as a straightforward yet effective tool to assist users in developing better study habits and achieving their academic objectives. Users can create personal goals, record study sessions, classify them by subject, and track their progress over time with the app. Users can simply monitor how much time they spend studying, assess their level of focus, and examine trends in their study habits thanks to an intuitive, entertaining interface.

In order to assist users understand their productivity and make wise adjustments, the software also offers clear data visualizations that highlight patterns. The app's contemporary, responsive design, which was created with Jetpack Compose, guarantees seamless navigation and user friendliness, making it perfect for everyday usage without being overbearing. The Mind Vault Study encourages regular study practices with practical insights.

6)SYSTEM REQUIREMENTS

1.HARDWARE REQUIREMENTS:

1. **Development Device (Computer/Laptop):**
   - **Processor**: Intel i5 (8th generation or higher) or AMD Ryzen 5 equivalent
   - **RAM**: 8 GB minimum (16 GB recommended for smoother performance)
   - **Storage**: 100 GB of available storage space
   - **Graphics**: Integrated graphics (discrete GPU optional for emulator acceleration)

2. **Mobile Device for Testing (Optional)**:
   - **Operating System**: Android 8.0 (Oreo) or higher
   - **RAM**: 2 GB minimum
   - **Storage**: At least 50 MB of free storage space for the app

Software Requirements:

1. **Operating System** (for development):
    - Windows 10 or 11 (64 bit), macOS (Big Sur or later), or Linux (Ubuntu 20.04 or later)
2. **Development Tools**:
    - **Android Studio**: Version 4.2 or higher
    - **JDK**: Java Development Kit 11 or higher
    - **Android SDK**: Android SDK API Level 26 or higher
3. **Libraries and Frameworks**:
    - **Android Jetpack Compose**: For building UI components
    - **Room Database**: For local data storage (optional but recommended for sleep data persistence)
    - **Kotlin Coroutines**: For managing asynchronous tasks smoothly
    - **Dagger Hilt**: For dependency injection (optional, for larger app projects)
4. **Database**:
    - **SQLite or Room Database**: For offline data storage and sleep record persistence
    - **SharedPreferences**: For lightweight data (e.g., user settings, last session data)
5. **Additional Tools (Optional)**:
    - **Firebase**: For remote data storage, analytics, and crash reporting (optional)
    - **Git**: For version control and collaboration, with a GitHub or GitLab repository for project storage
    - **Postman**: For testing any APIs (if integrated with other health services)

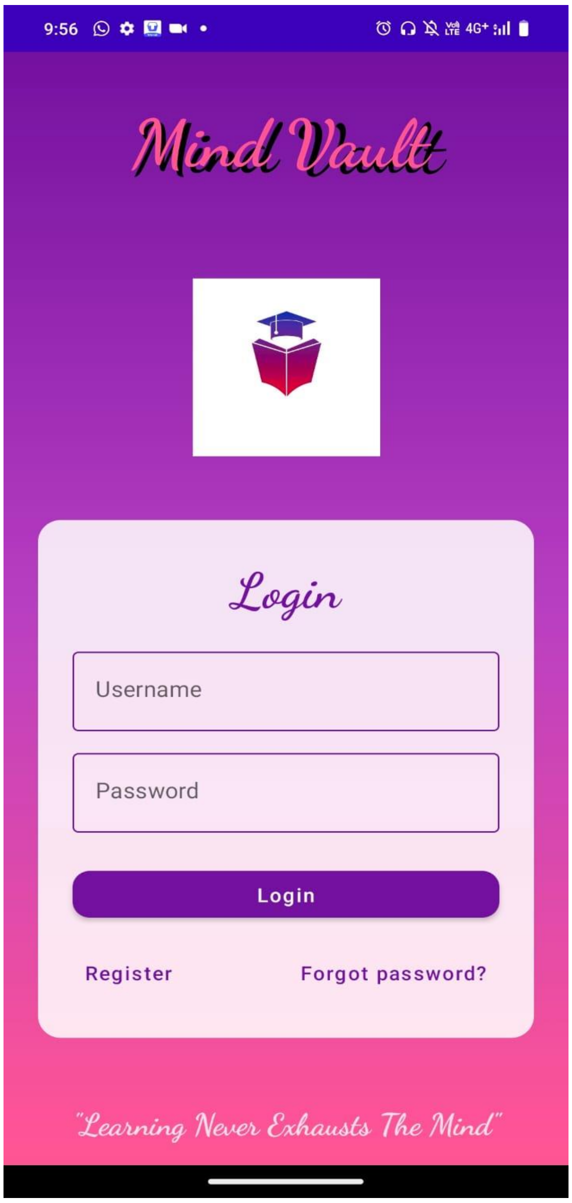
7)Testing Requirements

1. **Android Emulator**: Configured in Android Studio with API Level 26 or higher
2. **Physical Android Device** (optional): For real world testing, ideally with Android 8.0 (Oreo) or higher

8)Network Requirements (Optional)

- **Internet Connection**: Required only for features such as updates, analytics, or remote data storage if using Firebase or other cloud services.

<u>9)APPLICATION SCREENSHOTS</u>



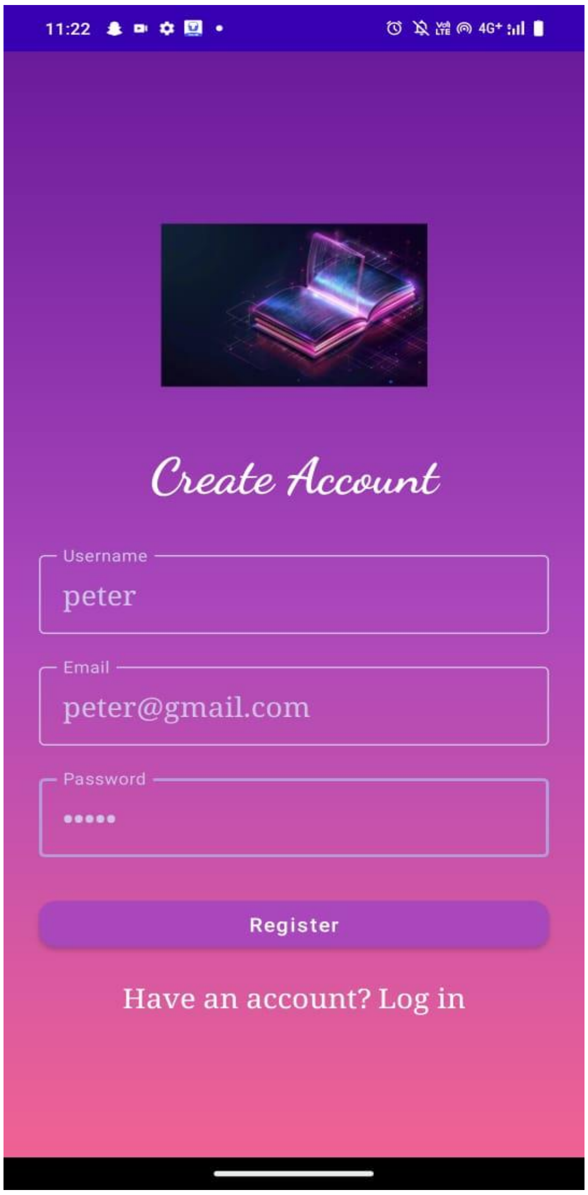
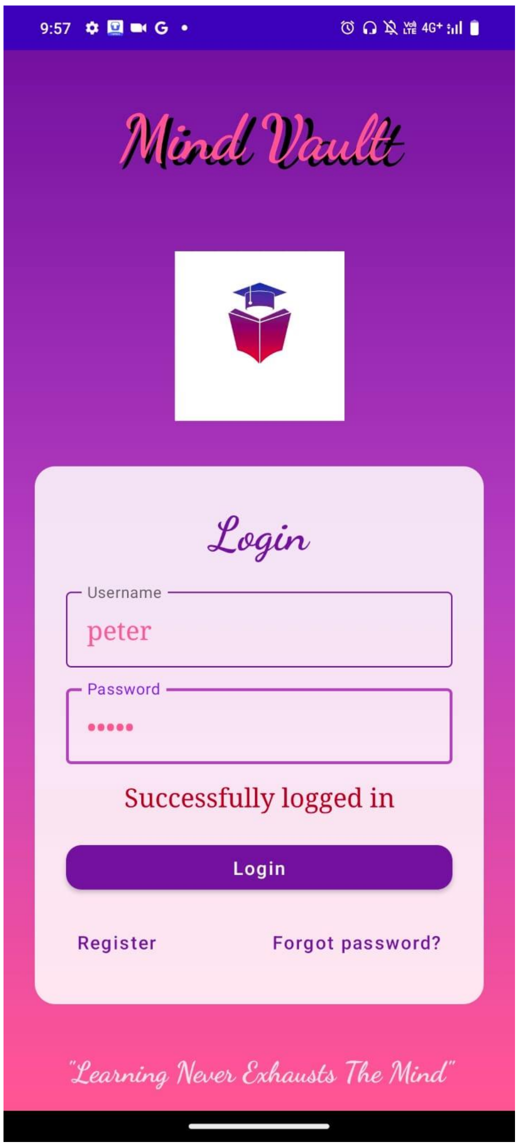


Figure 1: Login page **Figure 2: Register page**



Successfully logged in

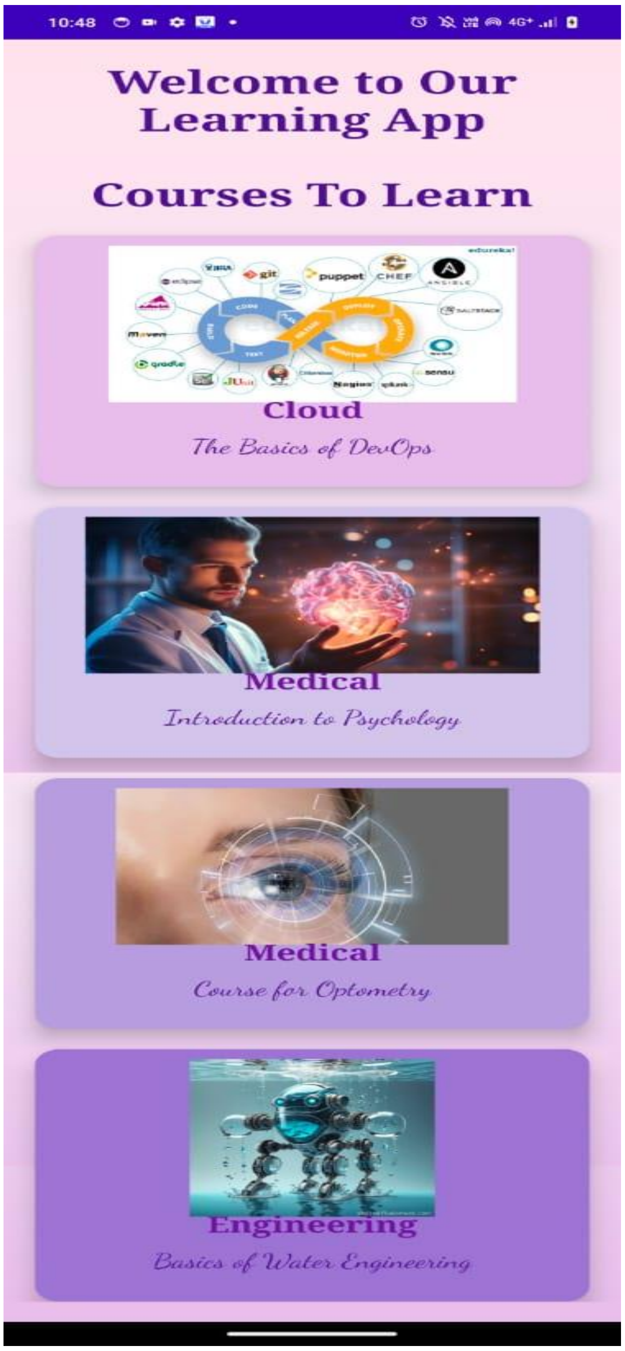
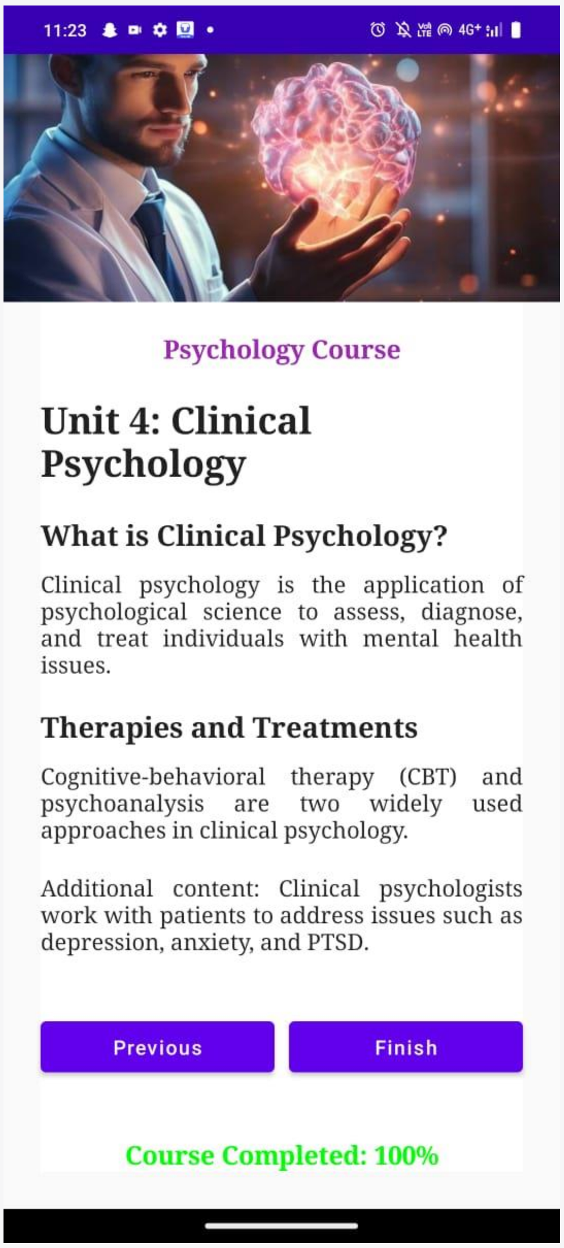


Figure 3: Login page after Entering details

Figure 4: Course page (Home page)



Psychology Course

Unit 4: Clinical Psychology

What is Clinical Psychology?

Clinical psychology is the application of psychological science to assess, diagnose, and treat individuals with mental health issues.

Therapies and Treatments

Cognitive-behavioral therapy (CBT) and psychoanalysis are two widely used approaches in clinical psychology.

Additional content: Clinical psychologists work with patients to address issues such as depression, anxiety, and PTSD.

Course Completed: 100%

Figure 5: Course Content Page

10)Applications

- **Developing Regular Study Routines:** By monitoring study trends and recognizing behaviors that affect productivity, such as preferred subjects or ideal study hours, Mind Vault assists users in creating and maintaining regular study routines. Users can modify their habits as a result of this information, which eventually improves concentration and learning effectiveness.

- **Handling Academic Difficulties**: Mind Vault can identify trends and possible distractions that might be affecting a student's performance if they are having trouble focusing or with particular topics. In order to develop a customized learning plan that takes into account their particular difficulties, users can share these findings with teachers or tutors.
- **Assisting with Exam Preparation:** During crucial times, as while preparing for an exam, Mind Vault helps students organize and monitor their study sessions. It assists by giving a concise summary of the amount of time spent on each topic.
- **Research and Education Insights:** Teachers and researchers examining study habits, patterns, and trends across various groups can benefit greatly from the app's aggregated, anonymized data. These observations can be used to discover elements that support or undermine student achievement and to guide instructional strategies.

11)PROJECT HURDLES

1. User Engagement and Retention

Challenge: Keeping users consistently engaged with the app can be difficult, especially in educational apps where users may lose motivation over time.

Solution: To address this, we focused on gamifying the learning experience, adding features like streaks, points, and achievement badges to encourage regular app use.

2. Content Quality and Relevance

Challenge: Ensuring high quality, accurate, and relevant content for various subjects and education levels required extensive research and expert consultation.

Solution: We collaborated with educators and subject matter experts to develop reliable and effective study materials tailored to user needs.

3. Adaptability to Different Learning Styles

Challenge: Users have diverse learning preferences (visual, auditory, kinesthetic, etc.), and catering to all styles within one app can be complex.

Solution: To accommodate different styles, Mind Vault offers multiple learning modes, such as flashcards, quizzes, and summaries, giving users the flexibility to choose what works best for them.

4. Data Privacy and Security

Challenge: As with any educational app, protecting user data—particularly if users are minors—is crucial to comply with privacy regulations and build trust.

Solution: We implemented robust data encryption, and followed best practices in data security and compliance with relevant privacy laws (e.g., COPPA, GDPR).

5. Performance and Technical Scalability

Challenge: Ensuring smooth performance as the app grows in features and user base was an ongoing challenge.

Solution: Our team optimized app code for speed, reduced data usage, and implemented scalable server infrastructure to handle large numbers of users without slowdowns.

6. User Feedback and Iterative Improvements

Challenge: Gathering meaningful feedback from users and quickly adapting to meet their needs required a dedicated feedback loop and agile development process.

Solution: We integrated in app feedback options and conducted user testing sessions to collect insights, prioritizing changes based on user feedback to ensure the app’s relevance and usability.

7. Balancing Features with Usability

Challenge: Adding too many features risks making the app feel cluttered, while limiting features can make it feel incomplete.

Solution: We focused on a core set of features that add significant value to the user experience while maintaining a clean, intuitive interface. Features were rolled out gradually to ensure seamless integration.

8. Funding and Resource Allocation

Challenge: Developing a comprehensive educational tool with limited funding required careful budget management and resource allocation.

Solution: By prioritizing features based on user needs and potential impact, we ensured that each development phase was as cost effective as possible.

These points can help convey the challenges faced and the thoughtful strategies used to overcome them, highlighting the app's development depth and user centered approach. Let me know if you need any additional points or detail on specific hurdles!

12)CONCLUSION :

Mind Vault could be a useful tool for managing academic difficulties, creating productive study habits, and maximizing learning outcomes in a variety of educational contexts. It helps users to keep an eye on their productivity, pinpoint areas for growth, and establish long lasting study habits by monitoring study patterns and offering tailored insights. In an era where self directed study and remote learning are becoming more popular, Mind Vault gives users the ability to take charge of their education and make wise decisions. The software helps students and lifetime learners accomplish their academic goals more successfully and independently by encouraging consistency and attention.

13)FUTURE SCOPE

The Mind Vault Study App has a bright future ahead of it thanks to technological developments that will enable even more customized and interesting features. Better algorithms might make it possible to analyze study patterns more intelligently and provide tailored advice on study methods, the best times to learn, and how to organize study sessions. The user experience might be further improved by integration with productivity tools and AI driven insights, which would allow for customized recommendations to increase focus and memory. The protection of user data will be guaranteed by improved privacy and security features. Achievement badges and streaks are examples of gamification features that could motivate users to meet their study objectives and make learning more pleasurable. With continued improvements, Mind Vault could develop into a vital tool for both students and lifelong learners, encouraging productive study habits.

14)SOURCE CODE

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.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Brush
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.unit.dp
import androidx.compose.ui.unit.sp
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 {
            OwlApplicationTheme {
                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("") }

    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(
                brush = Brush.verticalGradient(
                    colors = listOf(
                        Color(0xFF6A1B9A), // Deep Purple
                        Color(0xFFAB47BC), // Lighter Purple
                        Color(0xFFF06292)  // Vibrant Pink
                    )
                )
            )
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(horizontal = 24.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.SpaceBetween //
Aligns items from top to bottom with spacing
        ) {
            // App Name with Stroke Effect
            Box(modifier = Modifier.padding(top = 30.dp)) {
                Text(
                    text = "Mind Vault",
                    fontSize = 50.sp,
                    fontWeight = FontWeight.Bold,
                    fontFamily = FontFamily.Cursive,
                    color = Color.Black, // Stroke color
                    modifier = Modifier.offset(2.dp, 2.dp) // Offset to
create stroke effect
                )
                Text(
                    text = "Mind Vault",
                    fontSize = 48.sp,
                    fontWeight = FontWeight.Bold,
                    fontFamily = FontFamily.Cursive,
                    color = Color(0xFFF06292) // Inner text color
                )
            }

            // Image below the app name
            Image(
                painter = painterResource(id = R.drawable.loginowl),
                contentDescription = null,
                contentScale = ContentScale.Fit,
                modifier = Modifier
                    .size(150.dp)
                    .padding(top = 20.dp)
            )

            // Login box
            Box(
                modifier = Modifier
                    .fillMaxWidth()
                    .align(Alignment.CenterHorizontally)
                    .background(Color.White.copy(alpha = 0.85f),
RoundedCornerShape(16.dp))
                    .padding(24.dp)
            ) {
                Column(horizontalAlignment =
Alignment.CenterHorizontally) {
                    Text(
                        text = "Login",
                        fontSize = 36.sp,
                        fontWeight = FontWeight.ExtraBold,
                        fontFamily = FontFamily.Cursive,
                        color = Color(0xFF6A1B9A)
                    )
                    Spacer(modifier = Modifier.height(10.dp))

                    OutlinedTextField(
                        value = username,
                        onValueChange = { username = it },
                        label = { Text("Username") },
                        modifier = Modifier
                            .fillMaxWidth()
                            .padding(vertical = 4.dp),
                        textStyle =
MaterialTheme.typography.body1.copy(
                            color = Color(0xFFF06292)
                        ),
                        colors =
TextFieldDefaults.outlinedTextFieldColors(
                            focusedBorderColor = Color(0xFFAB47BC),
                            cursorColor = Color(0xFF6A1B9A),
                            unfocusedBorderColor = Color(0xFF6A1B9A)
                        )
                    )

                    OutlinedTextField(
                        value = password,
                        onValueChange = { password = it },
                        label = { Text("Password") },
                        visualTransformation =
PasswordVisualTransformation(),
                        modifier = Modifier
                            .fillMaxWidth()
                            .padding(vertical = 4.dp),
                        textStyle =
MaterialTheme.typography.body1.copy(
                            color = Color(0xFFF06292)
                        ),
                        colors =
TextFieldDefaults.outlinedTextFieldColors(
                            focusedBorderColor = Color(0xFFAB47BC),
                            cursorColor = Color(0xFF6A1B9A),
                            unfocusedBorderColor = Color(0xFF6A1B9A)
                        )
                    )

                    if (error.isNotEmpty()) {
                        Text(
                            text = error,
                            color = MaterialTheme.colors.error,
                            modifier = Modifier.padding(top = 8.dp)
                        )
                    }

                    Spacer(modifier = Modifier.height(16.dp))

                    Button(
                        onClick = {
                            if (username.isNotEmpty() &&
password.isNotEmpty()) {
                    val user =
databaseHelper.getUserByUsername(username)
                    if (user != null && user.password ==
password) {
                        error = "Successfully logged in"
                        context.startActivity(Intent(context,
MainActivity::class.java))
                    } else {
                        error = "Invalid username or password"
                    }
                            } else {
                    error = "Please fill all fields"
                            }
                        },
                        colors =
ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF6A1B9A)),
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(50.dp)
                            .padding(vertical = 8.dp),
                        shape = RoundedCornerShape(12.dp)
                    ) {
                        Text(text = "Login", color = Color.White)
                    }

                    Spacer(modifier = Modifier.height(8.dp))

                    Row(
                        modifier = Modifier.fillMaxWidth(),
                        horizontalArrangement =
Arrangement.SpaceBetween
                    ) {
                        TextButton(
                            onClick = {
                                context.startActivity(Intent(context,
RegisterActivity::class.java))
                            },
                            colors =
ButtonDefaults.textButtonColors(contentColor =
Color(0xFF6A1B9A))
                        ) {
                            Text(text = "Register")
                        }
                        TextButton(
                            onClick = { /* Add forgot password
functionality */ },
                            colors =
ButtonDefaults.textButtonColors(contentColor =
Color(0xFF6A1B9A))
                        ) {
                            Text(text = "Forgot password?")
                        }
                    }
                }
            }
```

```
            Text(
                text = "\"Learning Never Exhausts The Mind\"",
                fontSize = 22.sp,
                fontWeight = FontWeight.Bold,
                fontFamily = FontFamily.Cursive,
                color = Color.White.copy(alpha = 0.8f),
                modifier = Modifier.padding(bottom = 16.dp)
            )
        }
    }
}
```

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.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Brush
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.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 {
            OwlApplicationTheme {
                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("") }

    Box(
        modifier = Modifier
            .fillMaxSize()
            .background(
                brush = Brush.verticalGradient(
                    colors = listOf(
                        Color(0xFF6A1B9A), // Deep Purple
                        Color(0xFFAB47BC), // Lighter Purple
                        Color(0xFFF06292)  // Vibrant Pink
                    )
                )
            )
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(horizontal = 24.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {

            Image(
                painter = painterResource(id = R.drawable.resource),
                contentDescription = "Register Icon",
                contentScale = ContentScale.Fit,
                modifier = Modifier
                    .size(180.dp)
                    .padding(bottom = 16.dp)
            )
            // Header Text
            Text(
                text = "Create Account",
                fontSize = 36.sp,
                fontWeight = FontWeight.ExtraBold,
                fontFamily = FontFamily.Cursive,
                color = Color.White
            )

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

            // Updated Input Fields with light color style for contrast
            OutlinedTextField(
                value = username,
                onValueChange = { username = it },
                label = { Text("Username", color =
    Color(0xFFD1C4E9)) },
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(vertical = 8.dp),
                textStyle =
    MaterialTheme.typography.body1.copy(color =
    Color(0xFFD1C4E9)),
                colors = TextFieldDefaults.outlinedTextFieldColors(
                    focusedBorderColor = Color(0xFFB39DDB),
                    cursorColor = Color(0xFFCE93D8),
                    unfocusedBorderColor = Color(0xFFD1C4E9),
                    textColor = Color.White
                )
            )

            OutlinedTextField(
                value = email,
                onValueChange = { email = it },
                label = { Text("Email", color = Color(0xFFD1C4E9)) },
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(vertical = 8.dp),
                textStyle =
    MaterialTheme.typography.body1.copy(color =
    Color(0xFFD1C4E9)),
                colors = TextFieldDefaults.outlinedTextFieldColors(
                    focusedBorderColor = Color(0xFFB39DDB),
                    cursorColor = Color(0xFFCE93D8),
                    unfocusedBorderColor = Color(0xFFD1C4E9),
                    textColor = Color.White
                )
            )

            OutlinedTextField(
                value = password,
                onValueChange = { password = it },
                label = { Text("Password", color =
    Color(0xFFD1C4E9)) },
                visualTransformation =
    PasswordVisualTransformation(),
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(vertical = 8.dp),
                textStyle =
    MaterialTheme.typography.body1.copy(color =
    Color(0xFFD1C4E9)),
                colors = TextFieldDefaults.outlinedTextFieldColors(
                    focusedBorderColor = Color(0xFFB39DDB),
                    cursorColor = Color(0xFFCE93D8),
                    unfocusedBorderColor = Color(0xFFD1C4E9),
                    textColor = Color.White
                )
            )

            if (error.isNotEmpty()) {
                Text(
                    text = error,
                    color = MaterialTheme.colors.error,
                    modifier = Modifier.padding(top = 8.dp)
                )
            }

            Spacer(modifier = Modifier.height(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"
                    context.startActivity(
                        Intent(context, LoginActivity::class.java)
                    )
                    } else {
                    error = "Please fill all fields"
                    }
                },
                colors = ButtonDefaults.buttonColors(backgroundColor
    = Color(0xFFAB47BC)),
                modifier = Modifier
                    .fillMaxWidth()
                    .height(50.dp)
                    .padding(vertical = 8.dp),
                shape = RoundedCornerShape(12.dp)
            ) {
                Text(text = "Register", color = Color.White)
            }

            Spacer(modifier = Modifier.height(16.dp))

            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.Center
            ) {
                Text(
                    text = "Have an account?",
                    color = Color.White,
                    fontFamily = FontFamily.Serif,
                    modifier = Modifier.padding(end = 4.dp)
                )
                Text(
                    text = "Log in",
                    fontFamily = FontFamily.Serif,
                    color = Color.White,
                    modifier = Modifier
                    .clickable {
                        context.startActivity(Intent(context,
    LoginActivity::class.java))
```

```
                    }
                )
            }
        }
    }
}

private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

<u>MainActivity.kt</u>

```
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.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.Card
import androidx.compose.material.MaterialTheme
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.Brush
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.font.FontFamily
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 MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            OwlApplicationTheme {
                StudyApp(this)
            }
        }
    }
}

@Composable
fun StudyApp(context: Context) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(
            brush = Brush.verticalGradient(
                colors = listOf(Color(0xFFFCE4EC),
Color(0xFFF3E5F5), Color(0xFFE1BEE7))
            )
        )
        .padding(20.dp)
        .verticalScroll(rememberScrollState())
    ) {
        Text(
            text = "Welcome to Our Learning App!\n\nCourses To
Learn",
            fontFamily = FontFamily.Serif,
            fontSize = 32.sp,
            fontWeight = FontWeight.ExtraBold,
            color = Color(0xFF4A148C),
            modifier = Modifier.align(Alignment.CenterHorizontally),
            textAlign = TextAlign.Center
        )

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

        // First Course Card
        CourseCard(
            context = context,
            imageRes = R.drawable.devops,
            courseTitle = stringResource(id = R.string.course1),
            topic = stringResource(id = R.string.topic1),
            cardColor = Color(0xFFE1BEE7),
            destination = MainActivity2::class.java
        )

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

```
// Second Course Card
CourseCard(
    context = context,
    imageRes = R.drawable.psycology,
    courseTitle = stringResource(id = R.string.course2),
    topic = stringResource(id = R.string.topic2),
    cardColor = Color(0xFFD1C4E9),
    destination = MainActivity3::class.java
)

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

// Third Course Card
CourseCard(
    context = context,
    imageRes = R.drawable.optometry,
    courseTitle = stringResource(id = R.string.course3),
    topic = stringResource(id = R.string.topic3),
    cardColor = Color(0xFFB39DDB),
    destination = MainActivity4::class.java
)

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

// Fourth Course Card
CourseCard(
    context = context,
    imageRes = R.drawable.water,
    courseTitle = stringResource(id = R.string.course4),
    topic = stringResource(id = R.string.topic4),
    cardColor = Color(0xFFC7B2EB),
    destination = MainActivity5::class.java
)
```

```
    }
}

@Composable
fun CourseCard(
    context: Context,
    imageRes: Int,
    courseTitle: String,
    topic: String,
    cardColor: Color,
    destination: Class<*>
) {
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .height(270.dp)
            .clickable {
                context.startActivity(Intent(context, destination))
            },
        backgroundColor = cardColor,
        shape = RoundedCornerShape(16.dp),
        elevation = 10.dp
    ) {
        Column(
            modifier = Modifier.padding(16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Center
        ) {
            Image(
                painter = painterResource(id = imageRes),
                contentDescription = courseTitle,
                modifier = Modifier
                    .height(120.dp)
                .scale(1.3f)
            )

            Spacer(modifier = Modifier.height(16.dp)) // Added more
space between image and title

            // Course Title (Below the image)
            Text(
                text = courseTitle,
                color = Color(0xFFB546FF),
                fontSize = 21.sp,
                fontWeight = FontWeight.Bold,
                fontFamily = FontFamily.Serif, // Serif font family for
the title
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(6.dp)) // Spacing
between title and topic

            // Topic
            Text(
                text = topic,
                fontFamily = FontFamily.Serif,
                fontWeight = FontWeight.Bold,
                fontSize = 23.sp,
                color = Color(0xFF28262A),
                textAlign = TextAlign.Center
            )
        }
}
}
```

MainActivity2.kt

```
package com.example.owlapplication

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.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import androidx.compose.runtime.*
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.platform.LocalContext
import androidx.compose.ui.res.painterResource
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 {
            OwlApplicationTheme {
                Greeting()
            }
        }
    }
}

data class Unit(
    val title: String,
    val content: String,
    val imageRes: Int,
    val subheading1: String,
    val text1: String,
    val subheading2: String,
    val text2: String,
    val extraContent: String // Added extra content for more details
)

@Composable
fun Greeting() {
    val units = listOf(
        Unit(
            "Unit 1: Introduction to DevOps",
            "DevOps is a set of practices that combines software
development (Dev) and IT operations (Ops).",
            R.drawable.devops1,  // Replace with an actual image
resource
            "What is DevOps?",
            "DevOps is the cultural shift and integration of
development and operations teams to improve productivity and
efficiency.",
            "DevOps Practices",
            "Some common DevOps practices include Continuous
Integration, Continuous Delivery, and Infrastructure as Code.",
            "Additional content: DevOps is not just about tools but
also about people, culture, and processes that work together to
deliver faster and more reliably."
        ),
        Unit(
            "Unit 2: Version Control with Git",
            "Git is a distributed version control system that allows
multiple developers to work on a project simultaneously.",
            R.drawable.devops1,  // Replace with an actual image
resource
            "What is Git?",
            "Git is an open-source version control system that tracks
changes in your codebase.",
            "Git Commands",
            "Some common Git commands include git clone, git
commit, git push, and git pull.",
            "Additional content: Git allows branching and merging,
which helps developers to work on different parts of a project
independently and then combine their work."
        ),
        Unit(
            "Unit 3: Continuous Integration and Continuous
Deployment",
            "CI/CD is the practice of automating the stages of app
development and deployment.",
            R.drawable.devops1,  // Replace with an actual image
resource
            "What is CI/CD?",
            "Continuous Integration and Continuous Deployment are
practices that aim to increase the speed and quality of software
releases.",
            "CI/CD Tools",
            "Popular CI/CD tools include Jenkins, GitHub Actions,
and GitLab CI.",
            "Additional content: CI/CD involves automating the
testing and deployment processes to ensure that the software is
always in a deployable state."
        )
    )

    var currentUnitIndex by remember { mutableStateOf(0) }
    val currentUnit = units[currentUnitIndex]
    val context = LocalContext.current
    // We track if the user has completed the course
    var completedUnits by remember { mutableStateOf(0) }
    val totalUnits = units.size
    var progressPercentage by remember { mutableStateOf(0) }

    // Calculate the progress percentage only after the last unit is
completed
    LaunchedEffect(currentUnitIndex) {
        // Set progress percentage only when the user reaches Unit 3
(i.e., the last unit)
        progressPercentage = if (currentUnitIndex == totalUnits - 1)
{
            100 // Set to 100% once the user finishes Unit 3
        } else {
            (completedUnits * 100) / totalUnits
        }
    }

    Column(
        modifier = Modifier
            .padding(start = 26.dp, end = 26.dp, bottom = 26.dp)
            .verticalScroll(rememberScrollState())
            .background(Color.White),
        verticalArrangement = Arrangement.Top
    ) {
        // Display the image of the current unit
        Image(
            painterResource(id = currentUnit.imageRes),
            contentDescription = currentUnit.title,
            modifier = Modifier
                .align(Alignment.CenterHorizontally)
                .scale(scaleX = 1.5F, scaleY = 1.5F)
        )

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

        // Course title
        Text(
            text = "DevOps Course",
            color = Color(0xFFFFA500),
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.align(Alignment.CenterHorizontally)
        )

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

        // Unit title
        Text(
            text = currentUnit.title,
            fontWeight = FontWeight.Bold,
            fontSize = 26.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally)
        )
```

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

// Subheading and content for the current unit
Text(
    text = currentUnit.subheading1,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text1,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

Text(
    text = currentUnit.subheading2,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text2,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

// Additional content
Text(
    text = currentUnit.extraContent,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(40.dp))

// Navigation buttons
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Previous Button (Only show for Unit 2 and Unit 3)
    if (currentUnitIndex > 0) {
        Button(
            onClick = { currentUnitIndex-- },
            modifier = Modifier.weight(1f)
        ) {
            Text(text = "Previous")
        }
    }

    Spacer(modifier = Modifier.width(10.dp))
```

```
            // Next Button (Show for all units except Unit 3)
            if (currentUnitIndex < totalUnits - 1) {
                Button(
                    onClick = {
                        currentUnitIndex++
                        completedUnits++
                    },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Next")
                }
            } else {
                Button(
                    onClick = {
                        completedUnits = totalUnits
                        val intent = Intent(context,
MainActivity::class.java)
                        context.startActivity(intent)
                    },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Finish")
                }
            }
        }

        Spacer(modifier = Modifier.height(40.dp))

        if (currentUnitIndex == totalUnits - 1) {
            Text(
                text = "Course Completed: $progressPercentage%",
                fontWeight = FontWeight.Bold,
            fontSize = 18.sp,
            modifier =
Modifier.align(Alignment.CenterHorizontally),
            color = Color.Green
          )
        }
    }
}
```

<u>MainActivity3.kt</u>

```
package com.example.owlapplication

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.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import androidx.compose.runtime.*
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.platform.LocalContext
import androidx.compose.ui.res.painterResource
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 MainActivity3 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            OwlApplicationTheme {
                PsychologyCourse()
            }
        }
    }
}

data class Unit1(
    val title: String,
    val content: String,
    val imageRes: Int,
    val subheading1: String,
    val text1: String,
    val subheading2: String,
    val text2: String,
    val extraContent: String // Added extra content for more details
)

@Composable
fun PsychologyCourse() {
    val units = listOf(
        Unit1(
            "Unit 1: Introduction to Psychology",
            "Psychology is the scientific study of behavior and mental
processes.",
            R.drawable.psy,  // Replace with an actual image resource
            "What is Psychology?",
            "Psychology is the study of the mind and behavior,
involving various branches such as clinical, cognitive, and
behavioral psychology.",
            "History of Psychology",
            "Psychology has evolved over time from a philosophical
field into a rigorous scientific discipline.",
            "Additional content: Psychology seeks to understand
mental processes and behaviors through observation, research,
and experimentation."
        ),
        Unit1(
            "Unit 2: Cognitive Psychology",
            "Cognitive psychology focuses on mental processes such
as perception, memory, and problem-solving.",
            R.drawable.psycology,  // Replace with an actual image
resource
            "What is Cognitive Psychology?",
            "Cognitive psychology studies internal mental processes,
such as how people perceive, think, remember, and learn.",
            "Key Concepts",
            "Memory, attention, problem-solving, and decision-
making are some of the key areas studied in cognitive
psychology.",
            "Additional content: Cognitive psychology investigates
how information is processed and how individuals interpret and
respond to their environment."
        ),
        Unit1(
            "Unit 3: Social Psychology",
            "Social psychology studies how people influence and are
influenced by others in social contexts.",
            R.drawable.psy,  // Replace with an actual image resource
            "What is Social Psychology?",
            "Social psychology explores how people's thoughts,
feelings, and behaviors are shaped by their interactions with
others.",
            "Group Dynamics",
            "Group behavior, social influence, and leadership are key
areas in social psychology.",
            "Additional content: Social psychology also examines
phenomena like conformity, prejudice, and obedience in group
settings."
        ),
        Unit1(
            "Unit 4: Clinical Psychology",
            "Clinical psychology focuses on diagnosing and treating
mental health disorders.",
            R.drawable.psycology,  // Replace with an actual image
resource
            "What is Clinical Psychology?",
            "Clinical psychology is the application of psychological
science to assess, diagnose, and treat individuals with mental
health issues.",
            "Therapies and Treatments",
            "Cognitive-behavioral therapy (CBT) and psychoanalysis
are two widely used approaches in clinical psychology.",
            "Additional content: Clinical psychologists work with
patients to address issues such as depression, anxiety, and PTSD."
        )
    )

    var currentUnitIndex by remember { mutableStateOf(0) }
```

```
val currentUnit = units[currentUnitIndex]
val context = LocalContext.current

// Track if the user has completed the course
var completedUnits by remember { mutableStateOf(0) }
val totalUnits = units.size
var progressPercentage by remember { mutableStateOf(0) }

// Calculate progress percentage
LaunchedEffect(currentUnitIndex) {
   progressPercentage = if (currentUnitIndex == totalUnits - 1)
{
      100 // Set to 100% once the user finishes Unit 4
   } else {
      (completedUnits * 100) / totalUnits
   }
}

Column(
   modifier = Modifier
      .padding(start = 26.dp, end = 26.dp, bottom = 26.dp)
      .verticalScroll(rememberScrollState())
      .background(Color.White),
   verticalArrangement = Arrangement.Top
) {
   // Display the image of the current unit
   Image(
      painterResource(id = currentUnit.imageRes),
      contentDescription = currentUnit.title,
      modifier = Modifier
         .align(Alignment.CenterHorizontally)
         .scale(scaleX = 1.5F, scaleY = 1.5F)
   )
```

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

// Course title
Text(
    text = "Psychology Course",
    color = Color(0xFF9C27B0),
    fontSize = 18.sp,
    fontWeight = FontWeight.Bold,
    modifier = Modifier.align(Alignment.CenterHorizontally)
)

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

// Unit title
Text(
    text = currentUnit.title,
    fontWeight = FontWeight.Bold,
    fontSize = 26.sp,
    modifier = Modifier.align(Alignment.CenterHorizontally)
)

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

// Subheading and content for the current unit
Text(
    text = currentUnit.subheading1,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))
```

```
Text(
    text = currentUnit.text1,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

Text(
    text = currentUnit.subheading2,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text2,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

// Additional content
Text(
    text = currentUnit.extraContent,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(40.dp))

// Navigation buttons
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Previous Button (Only show for Unit 2 and Unit 3)
    if (currentUnitIndex > 0) {
        Button(
            onClick = { currentUnitIndex-- },
            modifier = Modifier.weight(1f)
        ) {
            Text(text = "Previous")
        }
    }

    Spacer(modifier = Modifier.width(10.dp))

    // Next Button (Show for all units except Unit 4)
    if (currentUnitIndex < totalUnits - 1) {
        Button(
            onClick = {
                currentUnitIndex++
                completedUnits++
            },
            modifier = Modifier.weight(1f)
        ) {
            Text(text = "Next")
        }
```

```
            } else {
                Button(
                    onClick = {
                        completedUnits = totalUnits
                        val intent = Intent(context,
MainActivity::class.java)
                        context.startActivity(intent)
                    },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Finish")
                }
            }
        }

        Spacer(modifier = Modifier.height(40.dp))

        if (currentUnitIndex == totalUnits - 1) {
            Text(
                text = "Course Completed: $progressPercentage%",
                fontWeight = FontWeight.Bold,
                fontSize = 18.sp,
                modifier =
Modifier.align(Alignment.CenterHorizontally),
                color = Color.Green
            )
        }
    }
}
```

<u>MainActivity4.kt</u>

```
package com.example.owlapplication

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.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import androidx.compose.runtime.*
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.platform.LocalContext
import androidx.compose.ui.res.painterResource
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 MainActivity4 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            OwlApplicationTheme {
                Greeting4()
            }
        }
    }
}

data class Unit4(
    val title: String,
    val content: String,
    val imageRes: Int,
    val subheading1: String,
    val text1: String,
    val subheading2: String,
    val text2: String,
    val extraContent: String
)

@Composable
fun Greeting4() {
    val units = listOf(
        Unit4(
            "Unit 1: Introduction to Optometry",
            "Optometry is the practice of examining the eyes and
visual systems for defects, visual abnormalities, and prescribing
corrective lenses.",
            R.drawable.optometry1,  // Replace with an actual image
resource
            "What is Optometry?",
            "Optometry involves diagnosing and treating vision
problems and eye diseases.",
            "Key Components",
            "Optometrists are trained to prescribe glasses, contact
lenses, and other treatments to improve vision.",
            "Additional content: Regular eye exams are essential for
maintaining good vision and preventing future complications."
```

```
        ),
        Unit4(
            "Unit 2: The Anatomy of the Eye",
            "Understanding the anatomy of the eye is crucial for
diagnosing and treating vision-related problems.",
            R.drawable.optometry,  // Replace with an actual image
resource
            "The Eye Structure",
            "The human eye consists of various parts, such as the
cornea, retina, and optic nerve, each playing a crucial role in
vision.",
            "How Light Enters the Eye",
            "Light enters the eye through the cornea and is focused by
the lens onto the retina.",
            "Additional content: The retina contains photoreceptor
cells that convert light into signals that are sent to the brain."
        ),
        Unit4(
            "Unit 3: Common Eye Conditions",
            "Various eye conditions affect people's vision and can
range from mild to severe.",
            R.drawable.optometry1,  // Replace with an actual image
resource
            "Common Eye Diseases",
            "Some common conditions include nearsightedness,
farsightedness, glaucoma, cataracts, and macular degeneration.",
            "Diagnosis and Treatment",
            "Optometrists use various tools to diagnose eye diseases
and prescribe appropriate treatments or refer patients to
specialists.",
            "Additional content: Early detection of eye diseases can
prevent severe vision loss and improve treatment outcomes."
        ),
        Unit4(
            "Unit 4: Corrective Lenses and Contacts",
            "Corrective lenses and contact lenses are common
solutions to refractive errors in vision.",
            R.drawable.optometry,  // Replace with an actual image
resource
            "Glasses vs Contacts",
            "Both glasses and contacts help correct refractive errors
like nearsightedness, farsightedness, and astigmatism.",
            "Lens Types",
            "There are different types of lenses for specific vision
needs, including single vision, bifocal, and progressive lenses.",
            "Additional content: Choosing between glasses and
contacts depends on lifestyle preferences and the severity of
vision problems."
        )
    )

    var currentUnitIndex by remember { mutableStateOf(0) }
    val currentUnit = units[currentUnitIndex]
    val context = LocalContext.current

    // We track if the user has completed the course
    var completedUnits by remember { mutableStateOf(0) }
    val totalUnits = units.size
    var progressPercentage by remember { mutableStateOf(0) }

    // Calculate the progress percentage only after the last unit is
completed
    LaunchedEffect(currentUnitIndex) {
        // Set progress percentage only when the user reaches Unit 4
(i.e., the last unit)
        progressPercentage = if (currentUnitIndex == totalUnits - 1)
```

```
{
        100 // Set to 100% once the user finishes Unit 4
    } else {
        (completedUnits * 100) / totalUnits
    }
}

Column(
    modifier = Modifier
        .padding(start = 26.dp, end = 26.dp, bottom = 26.dp)
        .verticalScroll(rememberScrollState())
        .background(Color.White),
    verticalArrangement = Arrangement.Top
) {
    // Display the image of the current unit
    Image(
        painterResource(id = currentUnit.imageRes),
        contentDescription = currentUnit.title,
        modifier = Modifier
            .align(Alignment.CenterHorizontally)
            .scale(scaleX = 1.5F, scaleY = 1.5F)
    )

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

    // Course title
    Text(
        text = "Optometry Course",
        color = Color(0xFF3F51B5),
        fontSize = 18.sp,
        fontWeight = FontWeight.Bold,
        modifier = Modifier.align(Alignment.CenterHorizontally)
    )
```

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

// Unit title
Text(
    text = currentUnit.title,
    fontWeight = FontWeight.Bold,
    fontSize = 26.sp,
    modifier = Modifier.align(Alignment.CenterHorizontally)
)

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

// Subheading and content for the current unit
Text(
    text = currentUnit.subheading1,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text1,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

Text(
    text = currentUnit.subheading2,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text2,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

// Additional content
Text(
    text = currentUnit.extraContent,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(40.dp))

// Navigation buttons
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Previous Button (Only show for Unit 2, 3, and 4)
```

```
            if (currentUnitIndex > 0) {
                Button(
                    onClick = { currentUnitIndex-- },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Previous")
                }
            }

            Spacer(modifier = Modifier.width(10.dp))

            // Next Button (Show for all units except Unit 4)
            if (currentUnitIndex < totalUnits - 1) {
                Button(
                    onClick = {
                        currentUnitIndex++
                        completedUnits++
                    },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Next")
                }
            }  else {
                Button(
                    onClick = {
                        completedUnits = totalUnits
                        val intent = Intent(context,
MainActivity::class.java)
                        context.startActivity(intent)
                    },
                    modifier = Modifier.weight(1f)
                ) {
                    Text(text = "Finish")
                }
            }
        }

        Spacer(modifier = Modifier.height(40.dp))

        if (currentUnitIndex == totalUnits - 1) {
            Text(
                text = "Course Completed: $progressPercentage%",
                fontWeight = FontWeight.Bold,
                fontSize = 18.sp,
                modifier =
Modifier.align(Alignment.CenterHorizontally),
                color = Color.Green
            )
        }
      }
}
```

MainActivity5.kt

```
package com.example.owlapplication

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.foundation.rememberScrollState
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.*
import androidx.compose.runtime.*
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.platform.LocalContext
import androidx.compose.ui.res.painterResource
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 MainActivity4 : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            OwlApplicationTheme {
                Greeting4()
            }
        }
    }
}

data class Unit4(
    val title: String,
    val content: String,
    val imageRes: Int,
    val subheading1: String,
    val text1: String,
    val subheading2: String,
    val text2: String,
    val extraContent: String
)

@Composable
fun Greeting4() {
    val units = listOf(
        Unit4(
            "Unit 1: Introduction to Optometry",
            "Optometry is the practice of examining the eyes and
visual systems for defects, visual abnormalities, and prescribing
corrective lenses.",
            R.drawable.optometry1,  // Replace with an actual image
resource
            "What is Optometry?",
            "Optometry involves diagnosing and treating vision
problems and eye diseases.",
            "Key Components",
            "Optometrists are trained to prescribe glasses, contact
lenses, and other treatments to improve vision.",
            "Additional content: Regular eye exams are essential for
maintaining good vision and preventing future complications."
        ),
        Unit4(
            "Unit 2: The Anatomy of the Eye",
            "Understanding the anatomy of the eye is crucial for
diagnosing and treating vision-related problems.",
            R.drawable.optometry,  // Replace with an actual image
resource
            "The Eye Structure",
            "The human eye consists of various parts, such as the
```

```
cornea, retina, and optic nerve, each playing a crucial role in
vision.",
            "How Light Enters the Eye",
            "Light enters the eye through the cornea and is focused by
the lens onto the retina.",
            "Additional content: The retina contains photoreceptor
cells that convert light into signals that are sent to the brain."
        ),
        Unit4(
            "Unit 3: Common Eye Conditions",
            "Various eye conditions affect people's vision and can
range from mild to severe.",
            R.drawable.optometry1,  // Replace with an actual image
resource
            "Common Eye Diseases",
            "Some common conditions include nearsightedness,
farsightedness, glaucoma, cataracts, and macular degeneration.",
            "Diagnosis and Treatment",
            "Optometrists use various tools to diagnose eye diseases
and prescribe appropriate treatments or refer patients to
specialists.",
            "Additional content: Early detection of eye diseases can
prevent severe vision loss and improve treatment outcomes."
        ),
        Unit4(
            "Unit 4: Corrective Lenses and Contacts",
            "Corrective lenses and contact lenses are common
solutions to refractive errors in vision.",
            R.drawable.optometry,  // Replace with an actual image
resource
            "Glasses vs Contacts",
            "Both glasses and contacts help correct refractive errors
like nearsightedness, farsightedness, and astigmatism.",
            "Lens Types",
            "There are different types of lenses for specific vision
needs, including single vision, bifocal, and progressive lenses.",
            "Additional content: Choosing between glasses and
contacts depends on lifestyle preferences and the severity of
vision problems."
        )
    )

    var currentUnitIndex by remember { mutableStateOf(0) }
    val currentUnit = units[currentUnitIndex]
    val context = LocalContext.current

    // We track if the user has completed the course
    var completedUnits by remember { mutableStateOf(0) }
    val totalUnits = units.size
    var progressPercentage by remember { mutableStateOf(0) }

    // Calculate the progress percentage only after the last unit is
completed
    LaunchedEffect(currentUnitIndex) {
        // Set progress percentage only when the user reaches Unit 4
(i.e., the last unit)
        progressPercentage = if (currentUnitIndex == totalUnits - 1)
{
            100 // Set to 100% once the user finishes Unit 4
        } else {
            (completedUnits * 100) / totalUnits
        }
    }

    Column(
        modifier = Modifier
```

```
            .padding(start = 26.dp, end = 26.dp, bottom = 26.dp)
            .verticalScroll(rememberScrollState())
            .background(Color.White),
        verticalArrangement = Arrangement.Top
    ) {
        // Display the image of the current unit
        Image(
            painterResource(id = currentUnit.imageRes),
            contentDescription = currentUnit.title,
            modifier = Modifier
                .align(Alignment.CenterHorizontally)
                .scale(scaleX = 1.5F, scaleY = 1.5F)
        )

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

        // Course title
        Text(
            text = "Optometry Course",
            color = Color(0xFF3F51B5),
            fontSize = 18.sp,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.align(Alignment.CenterHorizontally)
        )

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

        // Unit title
        Text(
            text = currentUnit.title,
            fontWeight = FontWeight.Bold,
            fontSize = 26.sp,
            modifier = Modifier.align(Alignment.CenterHorizontally)
```

```
)

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

// Subheading and content for the current unit
Text(
    text = currentUnit.subheading1,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text1,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

Text(
    text = currentUnit.subheading2,
    fontWeight = FontWeight.Bold,
    fontSize = 20.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(10.dp))

Text(
    text = currentUnit.text2,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

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

// Additional content
Text(
    text = currentUnit.extraContent,
    textAlign = TextAlign.Justify,
    fontSize = 16.sp,
    modifier = Modifier.align(Alignment.Start)
)

Spacer(modifier = Modifier.height(40.dp))

// Navigation buttons
Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceBetween
) {
    // Previous Button (Only show for Unit 2, 3, and 4)
    if (currentUnitIndex > 0) {
        Button(
            onClick = { currentUnitIndex-- },
            modifier = Modifier.weight(1f)
        ) {
            Text(text = "Previous")
        }
    }
```

```
        Spacer(modifier = Modifier.width(10.dp))

        // Next Button (Show for all units except Unit 4)
        if (currentUnitIndex < totalUnits - 1) {
            Button(
                onClick = {
                    currentUnitIndex++
                    completedUnits++
                },
                modifier = Modifier.weight(1f)
            ) {
                Text(text = "Next")
            }
        }  else {
            Button(
                onClick = {
                    completedUnits = totalUnits
                    val intent = Intent(context,
MainActivity::class.java)
                    context.startActivity(intent)
                },
                modifier = Modifier.weight(1f)
            ) {
                Text(text = "Finish")
            }
        }
    }

    Spacer(modifier = Modifier.height(40.dp))

    if (currentUnitIndex == totalUnits - 1) {
        Text(
            text = "Course Completed: $progressPercentage%",
                fontWeight = FontWeight.Bold,
                fontSize = 18.sp,
                modifier =
Modifier.align(Alignment.CenterHorizontally),
                color = Color.Green
            )
        }
    }
}
```

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_NA
ME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
         )
      }
      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_NA
ME)),
            lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
          )
      }
      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_NA
ME)),
               lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
               email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
               password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
            )
            users.add(user)
          } while (cursor.moveToNext())
      }
      cursor.close()
      db.close()
      return users
   }

}
```