# INTRODUCTION

# Overview

# Podcast Plus is an Android application that allows users to listen to their favorite podcasts while also providing them with a personalized and seamless experience. The app is built using the Redux architecture pattern, which provides a simple and efficient way to manage the app's state and ensure consistency across different components.

# One of the key features of Podcast Plus is its dynamic theming options, which enable users to customize the app's appearance to their liking. Users can choose from a range of pre-defined themes or create their own custom themes using the app's intuitive interface.

# The app's user interface is designed to be intuitive and user-friendly, making it easy for users to find and listen to their favorite podcasts. The app also includes features such as bookmarks, downloads, and subscriptions, which allow users to keep track of their favorite shows and episodes.

# Podcast Plus leverages the power of modern Android development tools and libraries, ensuring that the app is fast, efficient, and easy to maintain. The app is built using Kotlin, the latest and most popular programming language for Android development. It also uses popular libraries such as Dagger, Retrofit, and Room, which provide robust support for dependency injection, network requests, and local data storage.

# Overall, Podcast Plus is a modern and innovative podcast app that offers a unique combination of features and functionality. Its use of the Redux architecture pattern and dynamic theming options make it stand out from other podcast apps, while its intuitive user interface and powerful development tools make it a pleasure to use and maintain.

# 1.2 Overview

# The purpose of Podcast Plus is to provide Android users with a modern and innovative podcast app that offers a personalized and seamless listening experience. By using the Redux architecture pattern and dynamic theming options, Podcast Plus aims to simplify the app's logic, reduce code complexity, and provide users with greater control over the app's appearance.

# Podcast Plus is designed to be user-friendly and intuitive, making it easy for users to find and listen to their favorite podcasts. The app also includes features such as bookmarks, downloads, and subscriptions, which allow users to keep track of their favorite shows and episodes.

The project aims to leverage the power of modern Android development tools and libraries to ensure that the app is fast, efficient, and easy to maintain. By using Kotlin, Dagger, Retrofit, and Room, Podcast Plus provides robust support for dependency injection, network requests, and local data storage.

Overall, the purpose of Podcast Plus is to provide Android users with a podcast app that combines the latest in Android development tools and techniques with a personalized and seamless listening experience. By providing a unique combination of features and functionality, Podcast Plus aims to become the go-to podcast app for Android users.

# 2.PROBLEM DEFINITION& DESIGN THINKING

# 2.1 EMPATHY MAP

# 

# 2.2 IDEATION& BRAINSTORMING MAP

# Features:

# Customizable themes for individual podcasts.

# Integration with social media platforms to share favorite episodes and podcasts.

# Smart recommendations based on listening history and preferences.

# User-created playlists for episodes or podcasts.

# Option to download or stream episodes.

# Option to save episodes for offline listening.

# Option to change playback speed.

# Sleep timer function.

# Bookmarking feature to save progress and easily pick up where you left off.

# In-app messaging system for communicating with other users and sharing podcast recommendations.

# Integration with voice assistants like Google Assistant and Amazon Alexa.

# Marketing Ideas:

# Targeting podcast enthusiasts on social media platforms.

# Partnering with popular podcasters to promote the app.

# Offering exclusive content or early access to new episodes for users of the app.

# Running targeted ads on podcast-related websites and forums.

# Offering a referral program for users who recommend the app to others.

# Creating a user-generated content campaign where users share their favorite personalized podcast themes on social media.

# Monetization Strategies:

# Offering a free version with ads and a paid version without ads.

# Offering a premium subscription with access to exclusive content or features.

# Partnering with podcast networks for revenue sharing on promoted episodes or shows.

# Offering sponsorship opportunities for brands to promote their products or services through targeted podcast advertisements.

# Potential Challenges:

# Competing with established podcast apps like Spotify and Apple Podcasts.

# Ensuring compatibility with a wide range of Android devices and operating systems.

# Securing partnerships with popular podcasters and networks.

# Managing and analyzing user data to provide accurate recommendations and personalized experiences.

# Balancing user privacy concerns with targeted advertising opportunities.

# 3.RESULTS

# 3.1 Activity & Screenshot

# LOGIN IMAGE

# 

# Trailhead Profile Public URL

# Team Lead - https://trailblazer.me/id/logeshwar

# Team Member 1 -https://trailblazer.me/id/sathish2873

# Team Member 2 -https://trailblazer.me/id/karsk13

# Team Member 3 -https://trailblazer.me/id/tamils12

# ADVANTAGES & DISADVANTAGE

# Advantages:

# 1.Customizable themes: The ability to create custom themes for individual podcasts can enhance the user experience and make it easier for listeners to differentiate between different shows.

# 2.Smart recommendations: Smart recommendations based on listening history and preferences can help users discover new podcasts that they may not have found otherwise.

# 3.User-created playlists: User-created playlists can make it easier for users to organize and manage their favorite episodes or podcasts.

# 4.Option to download or stream: The option to download or stream episodes gives users the flexibility to listen to podcasts wherever and whenever they want, even without an internet connection.

# 5.Bookmarking and sleep timer features: Bookmarking and sleep timer features can make it easier for users to keep track of their progress and fall asleep listening to their favorite shows without having to worry about battery drain.

# 6.Integration with social media and voice assistants: Integration with social media platforms and voice assistants can make it easier for users to share their favorite podcasts and control playback without having to open the app.

# Disadvantages:

# 1.Competition with established apps: Podcast Plus will have to compete with established podcast apps like Spotify and Apple Podcasts, which may make it difficult to gain traction in a crowded market.

# 2.Compatibility issues: Ensuring compatibility with a wide range of Android devices and operating systems can be challenging and may require ongoing updates and maintenance.

# 3.Partnership limitations: Securing partnerships with popular podcasters and networks can be difficult, which may limit the app's ability to offer exclusive content or early access to new episodes.

# 4.User privacy concerns: Collecting and analyzing user data to provide personalized experiences and targeted advertising opportunities may raise privacy concerns among users.

# 5.Monetization challenges: Finding the right monetization strategy that balances user needs with revenue generation can be a challenge, particularly in a market where many podcast apps offer free versions with ads.

# APPLICATIONS

# 1.Personal entertainment: The app can be used by individuals who enjoy listening to podcasts for entertainment purposes. The ability to customize themes for individual podcasts and smart recommendations can enhance the listening experience and make it easier for users to discover new shows.

# 2.Commuting: Podcast Plus can be used by commuters who want to listen to their favorite podcasts while on the go. The app's option to download episodes for offline listening and change playback speed can make it easier for users to customize their listening experience during their daily commute.

# 3.Fitness: Podcast Plus can be used by fitness enthusiasts who want to listen to motivational or educational podcasts while working out. The app's sleep timer and playback speed options can also make it easier for users to customize their listening experience during workouts.

# 4.Education: Podcast Plus can be used by students or professionals who want to listen to educational podcasts to enhance their learning or professional development. The app's smart recommendations can help users discover new shows related to their interests or field of study.

# 5.Social networking: Podcast Plus's in-app messaging system can be used by users to communicate with other podcast enthusiasts, share recommendations, and discuss their favorite shows.

# 6.Marketing: Podcast Plus can be used by podcasters or podcast networks to promote their shows to a wider audience. The app's integration with social media platforms can also make it easier for podcasters to share their shows with their followers and promote their brand.

# 7.Business and productivity: Podcast Plus can be used by professionals who want to listen to business-related or productivity-focused podcasts while on the go. The app's smart recommendations can help users discover new shows related to their interests or industry, and the ability to save episodes for offline listening can make it easier for users to stay up-to-date on the latest industry news and trends.

# CONCLUSION

# Podcast Plus is a feature-rich podcast app that offers a personalized and customizable listening experience for Android users. With its dynamic themes, smart recommendations, and in-app messaging system, Podcast Plus sets itself apart from other podcast apps on the market. The app can be used for personal entertainment, commuting, fitness, education, social networking, marketing, and business and productivity purposes. While there are some potential disadvantages to using the app, such as limited compatibility with some podcast hosting platforms and the need for a paid subscription to access some premium features, the advantages of Podcast Plus make it a strong choice for podcast enthusiasts looking for a unique and customized listening experience.

# FUTURE SCOPE

# Expansion to iOS: Currently, Podcast Plus is only available for Android devices. Expanding the app to iOS would allow it to reach a wider audience and increase its user base.

# Integration with more podcast hosting platforms: While Podcast Plus currently integrates with popular platforms like Apple Podcasts and Spotify, expanding its compatibility with other hosting platforms could attract new users who use less well-known podcast services.

# Addition of new features: Podcast Plus could continue to add new features to enhance the listening experience for users, such as voice-activated controls, AI-generated episode summaries, or integration with virtual assistants like Amazon's Alexa or Google Assistant.

# Collaboration with podcast networks: Partnering with podcast networks or individual podcasters could provide opportunities for exclusive content, collaborations, and promotional opportunities.

# Expansion into international markets: Podcasts are growing in popularity worldwide, and expanding the app's reach into international markets could increase its user base and potential revenue streams.

# Integration with other forms of media: Integrating Podcast Plus with other forms of media, such as books, music, or video, could create a more immersive and personalized listening experience for users.

# APPENDIX

# A.SOURCE CODE

# 9.1 LOGIN ACTIVITY

# 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.BorderStroke

# 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.material.icons.Icons

# import androidx.compose.material.icons.filled.Lock

# import androidx.compose.material.icons.filled.Person

# import androidx.compose.runtime.\*

# import androidx.compose.ui.Alignment

# import androidx.compose.ui.Modifier

# import androidx.compose.ui.graphics.Color

# import androidx.compose.ui.res.painterResource

# 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.em

# import androidx.compose.ui.unit.sp

# import androidx.core.content.ContextCompat

# import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

# class LoginActivity : ComponentActivity() {

# private lateinit var databaseHelper: UserDatabaseHelper

# override fun onCreate(savedInstanceState: Bundle?) {

# super.onCreate(savedInstanceState)

# databaseHelper = UserDatabaseHelper(this)

# setContent {

# PodcastPlayerTheme {

# // A surface container using the 'background' color from the theme

# Surface(

# modifier = Modifier.fillMaxSize(),

# color = MaterialTheme.colors.background

# ) {

# 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("") }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# shape = RoundedCornerShape(100.dp),

# modifier = Modifier.padding(16.dp).fillMaxWidth()

# ) {

# Column(

# Modifier

# .background(Color.Black)

# .fillMaxHeight()

# .fillMaxWidth()

# .padding(bottom = 28.dp, start = 28.dp, end = 28.dp),

# horizontalAlignment = Alignment.CenterHorizontally,

# verticalArrangement = Arrangement.Center

# )

# {

# Image(

# painter = painterResource(R.drawable.podcast\_login),

# contentDescription = "", Modifier.height(400.dp).fillMaxWidth()

# )

# Text(

# text = "LOGIN",

# color = Color(0xFF6a3ef9),

# fontWeight = FontWeight.Bold,

# fontSize = 26.sp,

# style = MaterialTheme.typography.h1,

# letterSpacing = 0.1.em

# )

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

# TextField(

# value = username,

# onValueChange = { username = it },

# leadingIcon = {

# Icon(

# imageVector = Icons.Default.Person,

# contentDescription = "personIcon",

# tint = Color(0xFF6a3ef9)

# )

# },

# placeholder = {

# Text(

# text = "username",

# color = Color.White

# )

# },

# colors = TextFieldDefaults.textFieldColors(

# backgroundColor = Color.Transparent

# )

# )

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

# TextField(

# value = password,

# onValueChange = { password = it },

# leadingIcon = {

# Icon(

# imageVector = Icons.Default.Lock,

# contentDescription = "lockIcon",

# tint = Color(0xFF6a3ef9)

# )

# },

# placeholder = { Text(text = "password", color = Color.White) },

# visualTransformation = PasswordVisualTransformation(),

# colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

# )

# Spacer(modifier = Modifier.height(12.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"

# }

# },

# border = BorderStroke(1.dp, Color(0xFF6a3ef9)),

# colors = ButtonDefaults.buttonColors(backgroundColor = Color.Black),

# modifier = Modifier.padding(top = 16.dp)

# ) {

# Text(text = "Log In", fontWeight = FontWeight.Bold, color = Color(0xFF6a3ef9))

# }

# Row(modifier = Modifier.fillMaxWidth()) {

# TextButton(onClick = {

# context.startActivity(

# Intent(

# context,

# RegistrationActivity::class.java

# ))})

# {

# Text(

# text = "Sign up",

# color = Color.White

# )

# }

# Spacer(modifier = Modifier.width(80.dp))

# TextButton(onClick = { /\* Do something! \*/ })

# {

# Text(

# text = "Forgot password ?",

# color = Color.White

# )

# }

# }

# }

# }

# fun startMainPage(context: Context) {

# val intent = Intent(context, MainActivity::class.java)

# ContextCompat.startActivity(context, intent, null)

# }}

# 9.2 MAIN ACTIVITY

# import android.content.Context

# import android.media.MediaPlayer

# import android.os.Bundle

# import androidx.activity.ComponentActivity

# import androidx.activity.compose.setContent

# import androidx.compose.foundation.BorderStroke

# import androidx.compose.foundation.Image

# 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.graphics.Color

# 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.em

# import androidx.compose.ui.unit.sp

# import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

# class MainActivity : ComponentActivity() {

# override fun onCreate(savedInstanceState: Bundle?) {

# super.onCreate(savedInstanceState)

# setContent {

# PodcastPlayerTheme {

# // A surface container using the 'background' color from the theme

# Surface(

# modifier = Modifier.fillMaxSize(),

# color = MaterialTheme.colors.background

# ) {

# playAudio(this)

# }

# }

# }

# }

# }

# @Composable

# fun playAudio(context: Context) {

# Column(modifier = Modifier.fillMaxSize()) {

# Column(horizontalAlignment = Alignment.CenterHorizontally, verticalArrangement = Arrangement.Center) {

# Text(text = "PODCAST",

# modifier = Modifier.fillMaxWidth(),

# textAlign = TextAlign.Center,

# color = Color(0xFF6a3ef9),

# fontWeight = FontWeight.Bold,

# fontSize = 36.sp,

# style = MaterialTheme.typography.h1,

# letterSpacing = 0.1.em

# )

# }

# Column(modifier = Modifier

# .fillMaxSize()

# .verticalScroll(rememberScrollState())) {

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp),

# )

# Text(

# text = "GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_1)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img\_1),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp)

# )

# Text(

# text = "Haunted Houses, Evil Spirits & The Paranormal Explained | Sarbajeet Mohanty",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_2)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img\_2),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp)

# )

# Text(

# text = "Kaali Mata ki kahani - Black Magic & Aghoris ft. Dr Vineet Aggarwal",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_3)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img\_3),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp),

# )

# Text(

# text = "Tantra Explained Simply | Rajarshi Nandy - Mata, Bhairav & Kamakhya Devi",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_4)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img\_4),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp),

# )

# Text(

# text = "Complete Story Of Shri Krishna - Explained In 20 Minutes",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# Card(

# elevation = 12.dp,

# border = BorderStroke(1.dp, Color.Magenta),

# modifier = Modifier

# .padding(16.dp)

# .fillMaxWidth()

# .height(250.dp)

# )

# {

# val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_5)

# Column(

# modifier = Modifier.fillMaxSize(),

# horizontalAlignment = Alignment.CenterHorizontally

# ) {

# Image(

# painter = painterResource(id = R.drawable.img\_5),

# contentDescription = null,

# modifier = Modifier

# .height(150.dp)

# .width(200.dp),

# )

# Text(

# text = "Mahabharat Ki Poori Kahaani - Arjun, Shri Krishna & Yuddh - Ami Ganatra ",

# textAlign = TextAlign.Center,

# modifier = Modifier.padding(start = 20.dp, end = 20.dp)

# )

# Row() {

# IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.play),

# contentDescription = ""

# )

# }

# IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

# Icon(

# painter = painterResource(id = R.drawable.pause),

# contentDescription = ""

# )

# }

# }

# }

# }

# }

# }

# 9.3 REG ACTIVITY

# 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.BorderStroke

# import androidx.compose.foundation.Image

# import androidx.compose.foundation.background

# import androidx.compose.foundation.layout.\*

# import androidx.compose.material.\*

# import androidx.compose.material.icons.Icons

# import androidx.compose.material.icons.filled.Email

# import androidx.compose.material.icons.filled.Lock

# import androidx.compose.material.icons.filled.Person

# import androidx.compose.runtime.\*

# import androidx.compose.ui.Alignment

# import androidx.compose.ui.Modifier

# import androidx.compose.ui.draw.alpha

# import androidx.compose.ui.graphics.Color

# import androidx.compose.ui.layout.ContentScale

# import androidx.compose.ui.res.painterResource

# 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.em

# import androidx.compose.ui.unit.sp

# import androidx.core.content.ContextCompat

# import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

# class RegistrationActivity : ComponentActivity() { private lateinit var databaseHelper: UserDatabaseHelper

# override fun onCreate(savedInstanceState: Bundle?) {

# super.onCreate(savedInstanceState)

# databaseHelper = UserDatabaseHelper(this)

# setContent {

# PodcastPlayerTheme {

# // A surface container using the 'background' color from the theme

# Surface(

# modifier = Modifier.fillMaxSize(),

# color = MaterialTheme.colors.background

# ) {

# 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

# .background(Color.Black)

# .fillMaxHeight()

# .fillMaxWidth(),

# horizontalAlignment = Alignment.CenterHorizontally,

# verticalArrangement = Arrangement.Center

# )

# {

# Row {

# Text(

# text = "Sign Up",

# color = Color(0xFF6a3ef9),

# fontWeight = FontWeight.Bold,

# fontSize = 24.sp, style = MaterialTheme.typography.h1,

# letterSpacing = 0.1.em

# )

# }

# Image(

# painter = painterResource(id = R.drawable.podcast\_signup),

# contentDescription = ""

# )

# TextField(

# value = username,

# onValueChange = { username = it },

# leadingIcon = {

# Icon(

# imageVector = Icons.Default.Person,

# contentDescription = "personIcon",

# tint = Color(0xFF6a3ef9)

# )

# },

# placeholder = {

# Text(

# text = "username",

# color = Color.White

# )

# },

# colors = TextFieldDefaults.textFieldColors(

# backgroundColor = Color.Transparent

# )

# )

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

# TextField(

# value = password,

# onValueChange = { password = it },

# leadingIcon = {

# Icon(

# imageVector = Icons.Default.Lock,

# contentDescription = "lockIcon",

# tint = Color(0xFF6a3ef9)

# )

# },

# placeholder = { Text(text = "password", color = Color.White) },

# visualTransformation = PasswordVisualTransformation(),

# colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

# )

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

# TextField(

# value = email,

# onValueChange = { email = it },

# leadingIcon = {

# Icon(

# imageVector = Icons.Default.Email,

# contentDescription = "emailIcon",

# tint = Color(0xFF6a3ef9)

# )

# },

# placeholder = { Text(text = "email", color = Color.White) },

# colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

# )

# Spacer(modifier = Modifier.height(8.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"

# }

# },

# border = BorderStroke(1.dp, Color(0xFF6a3ef9)),

# colors = ButtonDefaults.buttonColors(backgroundColor = Color.Black),

# modifier = Modifier.padding(top = 16.dp)

# ) {

# Text(text = "Register",

# fontWeight = FontWeight.Bold,

# color = Color(0xFF6a3ef9)

# )

# }

# Row(

# modifier = Modifier.padding(30.dp),

# verticalAlignment = Alignment.CenterVertically,

# horizontalArrangement = Arrangement.Center

# ) {

# Text(text = "Have an account?", color = Color.White)

# TextButton(onClick = {

# context.startActivity(

# Intent(

# context,

# LoginActivity::class.java

# )

# )

# })

# {

# Text(text = "Log in",

# fontWeight = FontWeight.Bold,

# style = MaterialTheme.typography.subtitle1,

# color = Color(0xFF6a3ef9)

# )

# }

# }

# }

# }

# private fun startLoginActivity(context: Context) {

# val intent = Intent(context, LoginActivity::class.java)

# ContextCompat.startActivity(context, intent, null)

# }

# 9.4 USER

# 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?,

# )

# 9.5 UserDao

# 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)

# }

# 9.6User Database

# package com.example.podcastplayer

# 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

# }

# }

# }

# }

# 9.7 User Data base Helper

# package com.example.podcastplayer

# import android.annotation.SuppressLint

# import android.content.ContentValues

# import android.content.Context

# import android.database.Cursor

# import android.database.sqlite.SQLiteDatabase

# import android.database.sqlite.SQLiteOpenHelper

# class UserDatabaseHelper(context: Context) :

# SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

# companion object {

# private const val DATABASE\_VERSION = 1

# private const val DATABASE\_NAME = "UserDatabase.db"

# private const val TABLE\_NAME = "user\_table"

# private const val COLUMN\_ID = "id"

# private const val COLUMN\_FIRST\_NAME = "first\_name"

# private const val COLUMN\_LAST\_NAME = "last\_name"

# private const val COLUMN\_EMAIL = "email"

# private const val COLUMN\_PASSWORD = "password"

# }

# override fun onCreate(db: SQLiteDatabase?) {

# val createTable = "CREATE TABLE $TABLE\_NAME (" +

# "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +

# "$COLUMN\_FIRST\_NAME TEXT, " +

# "$COLUMN\_LAST\_NAME TEXT, " +

# "$COLUMN\_EMAIL TEXT, " +

# "$COLUMN\_PASSWORD TEXT" +

# ")"

# db?.execSQL(createTable)

# }

# override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

# db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

# onCreate(db)

# }

# fun insertUser(user: User) {

# val db = writableDatabase

# val values = ContentValues()

# values.put(COLUMN\_FIRST\_NAME, user.firstName)

# values.put(COLUMN\_LAST\_NAME, user.lastName)

# values.put(COLUMN\_EMAIL, user.email)

# values.put(COLUMN\_PASSWORD, user.password)

# db.insert(TABLE\_NAME, null, values)

# db.close()

# }

# @SuppressLint("Range")

# fun getUserByUsername(username: String): User? {

# val db = readableDatabase

# val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username))

# var user: User? = null

# if (cursor.moveToFirst()) {

# user = User(

# id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

# firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

# lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

# email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

# password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

# )

# }

# cursor.close()

# db.close()

# return user

# }

# @SuppressLint("Range")

# fun getUserById(id: Int): User? {

# val db = readableDatabase

# val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

# var user: User? = null

# if (cursor.moveToFirst()) {

# user = User(

# id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

# firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

# lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

# email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

# password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

# )

# }

# cursor.close()

# db.close()

# return user

# }

# @SuppressLint("Range")

# fun getAllUsers(): List<User> {

# val users = mutableListOf<User>()

# val db = readableDatabase

# val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

# if (cursor.moveToFirst()) {

# do {

# val user = User(

# id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

# firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

# lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

# email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

# password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

# )

# users.add(user)

# } while (cursor.moveToNext())

# }

# cursor.close()

# db.close()

# return users

# }

# }