CONTENTS

INTRODUCTION

The main objective of the podcast app for Android for all things audio entertainments. This app can access a vast library of podcasts, audio books, and music, all in one convenient location.

1.1 **Overview:**

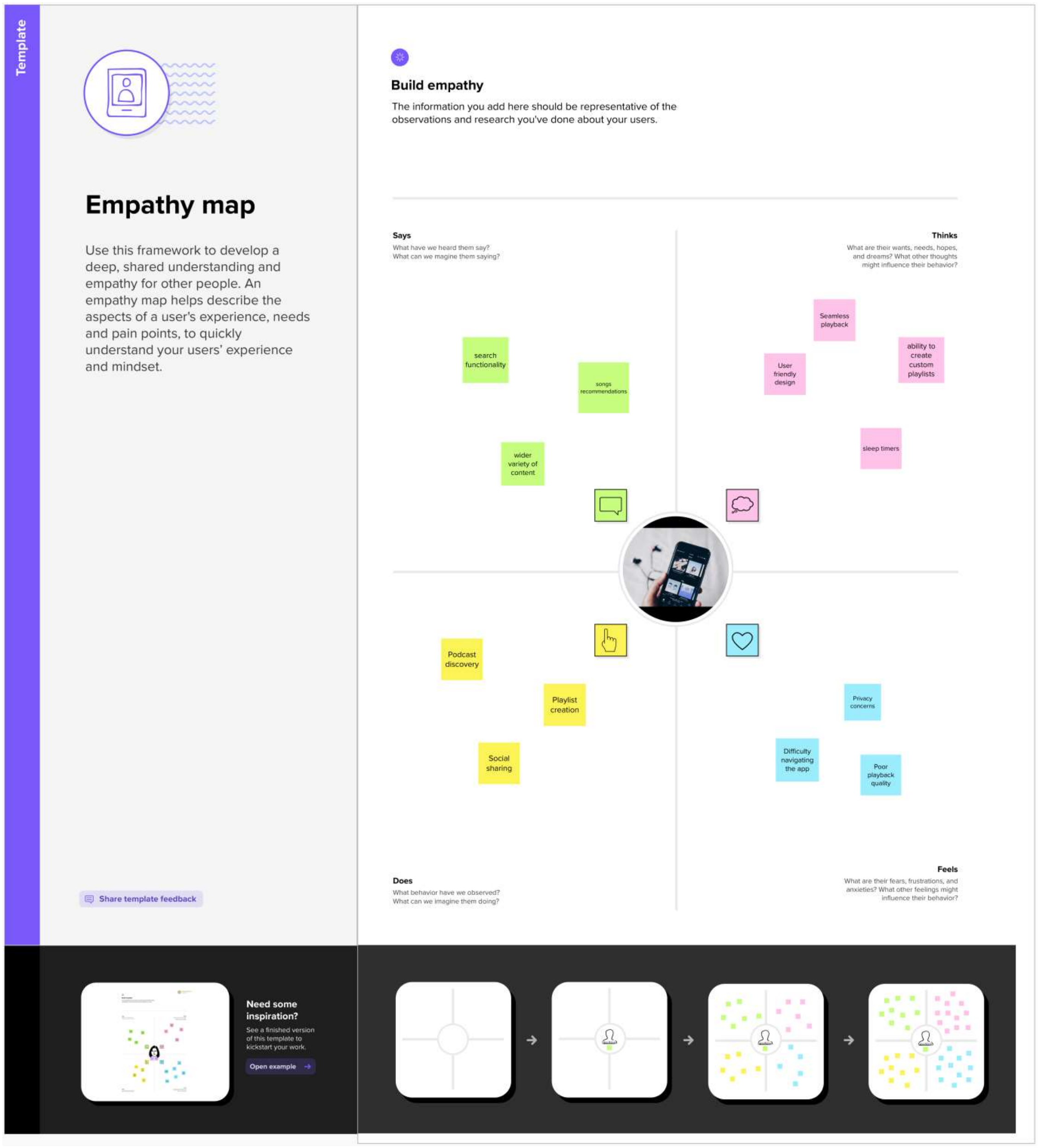
- To ensure your privacy and security, we have included a user-friendly sign-in and sign-up page.
- This app also features online music streaming, allowing you to listen to your favorite songs on the go.
- In addition to music, This app provides a vast array of podcasts covering a wide range of topics.

1.2 Purpose:

- Purpose Statement: This app aims to provide an immersive and convenient listening experience for users.
- With a simple and intuitive interface, users can easily discover, subscribe, and listen to their favourite podcasts.
- This app prioritizes user privacy by offering a secure sign-in and sign-up process.

PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map:



Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

The information you add here should be representative of the observations and research you've done about your users.

What have we heard them say? What can we magine them saying?

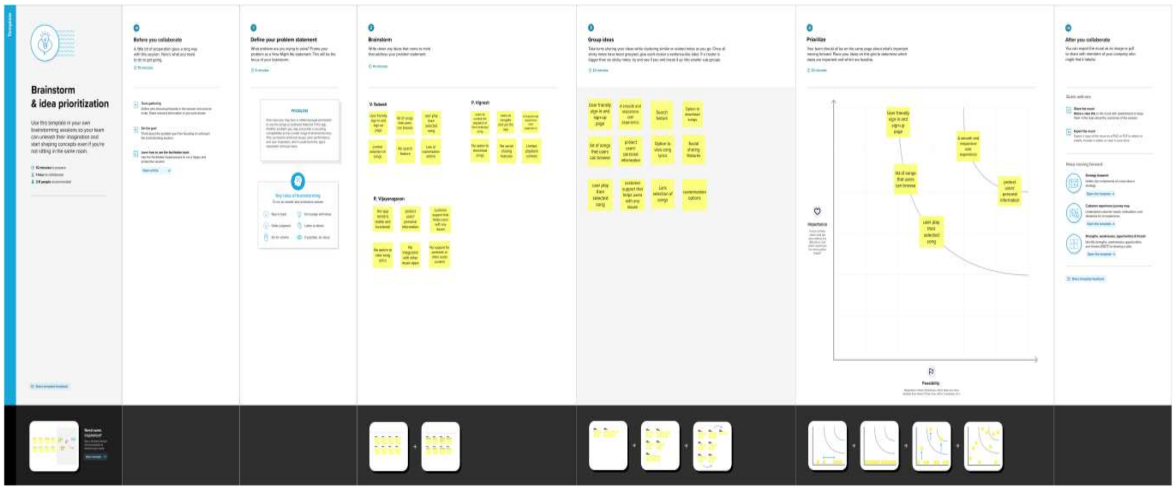
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

What behavior have we observed? What can we imagine them doing?

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

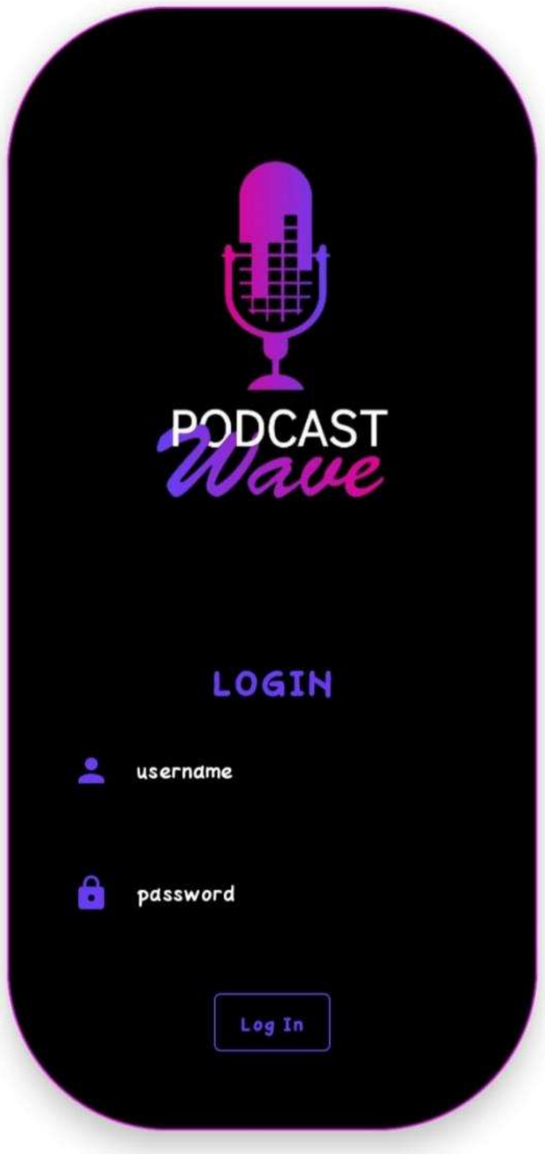
See a finished version of this template to kickstart your work.

2.2 Ideation & Brainstorming Map:

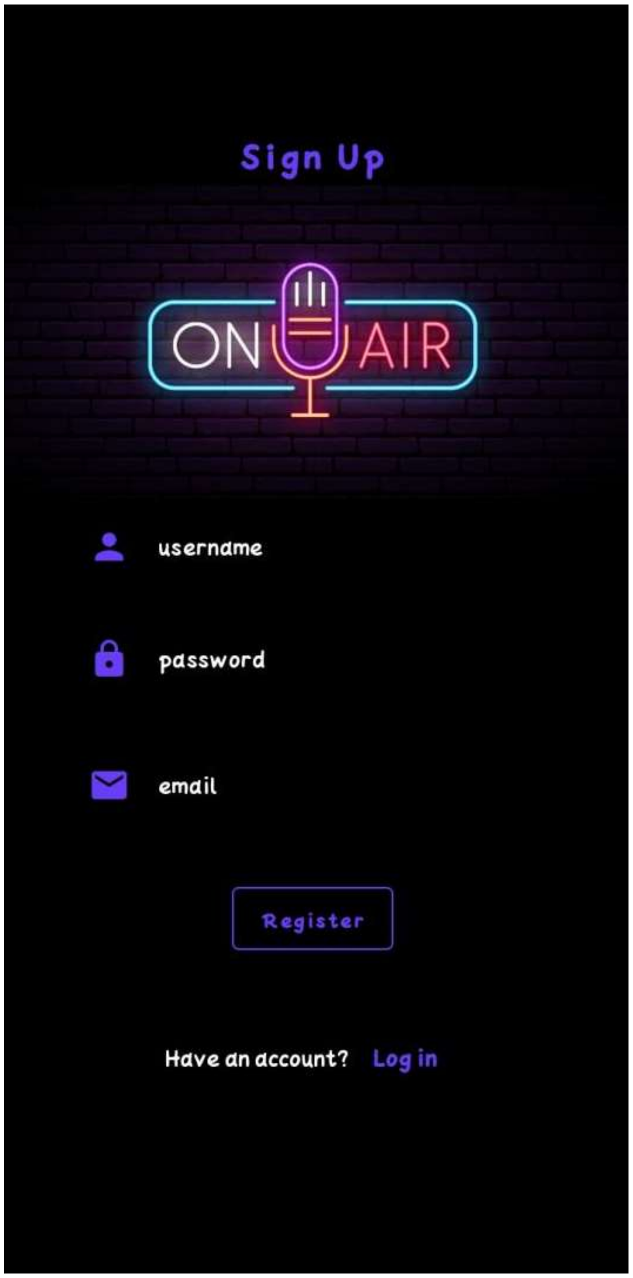

Brainstorm
& idea prioritization
Before you collaborate
Define your problem statement
PROBLEM
Brainstorm
V. Subash
P. Vignesh
R. Vijayaragavan
Group ideas
User friendly sign in and sign up page
A smooth and responsive user experience
Search feature
Option to download songs
list of songs that users can browse
protect users' personal information
Option to view song lyrics
Social sharing features
user play their selected song
customer support that helps users with any issues
Lack selection of songs
customization options
Prioritize
Importance
Feasibility
After you collaborate

RESULT

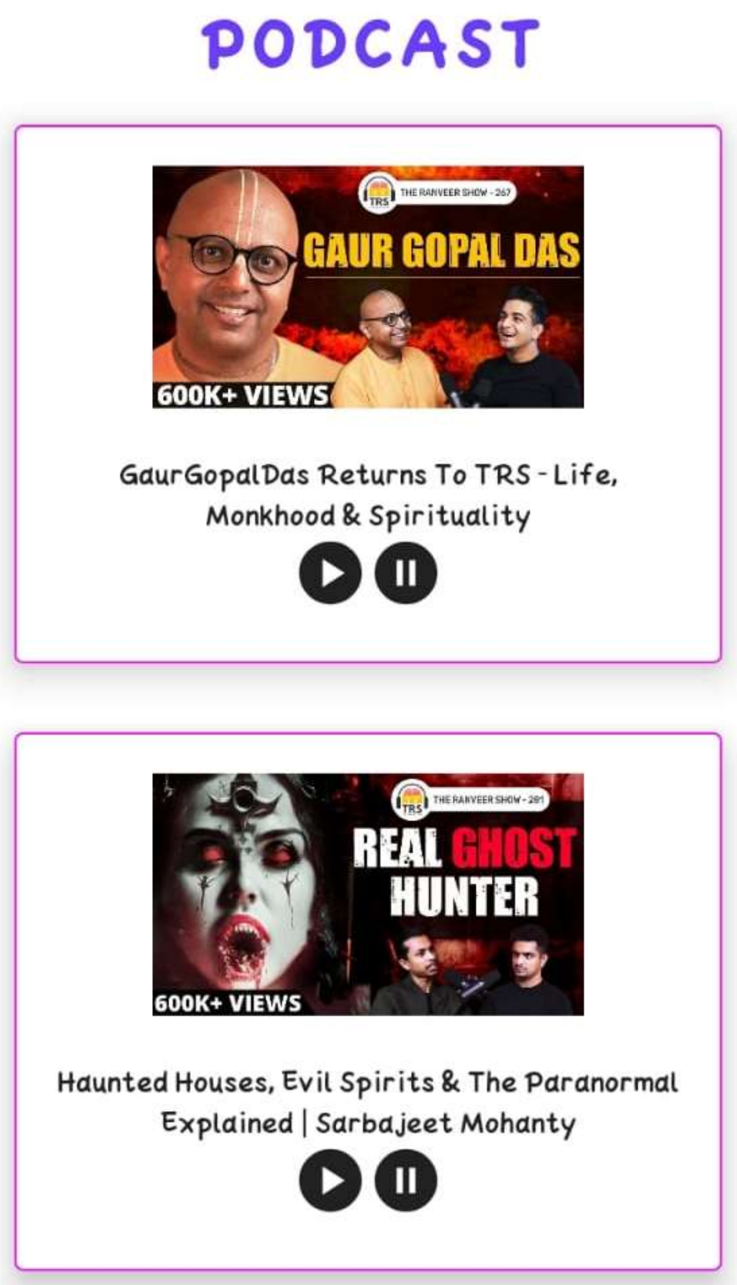
Sign in Page:



Sign up page:


Sign Up
ON AIR
username
password
email
Register
Have an account? Log in

Home Page:



PODCAST

GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality

Haunted Houses, Evil Spirits & The Paranormal Explained | Sarbajeet Mohanty

ADVANTAGES & DISADVANTAGES

1.1 Advantages:

- Access to a vast library of audio content: With this app, users can access a wide variety of audio content, including interviews, news, comedy, educational shows, and more.
- Personalized recommendations: A good podcast app will use advanced algorithms to analyze a user's listening history and preferences.
- Convenient listening experience: Unlike traditional radio, podcasts can be downloaded and played offline, meaning users can enjoy their favorite shows without an internet connection.
- Enhanced privacy and security: Just like your music app, a good podcast app should prioritize user privacy and security.

1.2 Disadvantages:

Limited access to certain shows: While many podcasts are available on multiple platforms, some shows may be exclusive to a particular app or network.

Quality and reliability of content: With so many podcasts available, it can be difficult for users to distinguish high-quality content from low-quality content.

Potential for security risks: While many podcast apps prioritize user privacy and security, there is always a risk of data breaches or other security issues.

APPLICATIONS

Entertainment: One of the main applications of this app is for entertainment purposes. Users can listen to their favorite shows and discover new content that aligns with their interests, providing a fun and engaging way to pass the time.

Education: Many podcasts are educational in nature, covering topics like history, science, and business. The podcast app can provide users with a convenient and accessible way to learn about new topics and expand their knowledge.

News and current events: The podcast app can also be a useful tool for staying up-to-date on news and current events. Many news outlets offer podcasts that cover the latest stories and provide in-depth analysis, making it easy for users to stay informed on the go.

Business and marketing: For business owners and marketers, the podcast app can be a valuable tool for reaching new audiences and building a following. By creating the podcast that provides value to listeners, businesses can establish themselves as thought leaders in their industry and attract new customers.

Health and wellness: Many podcasts focus on health and wellness topics, providing users with tips and advice for living a healthy lifestyle. The podcast app can be a useful tool for users who are looking to improve their physical and mental well-being.

CONCLUSION

We conclude that the podcast app on Android can provide users with a wealth of benefits, including access to a diverse range of audio content, personalized recommendations, and a convenient listening experience. With the ability to download episodes for offline playback and adjust playback speed and skip episodes, users can enjoy their favourite shows on their own terms. While there are potential disadvantages to using the podcast app, such as dependence on an internet connection and the quality and reliability of content, these can be mitigated by using trusted and reputable apps and being discerning when selecting shows to listen to.

FUTURE SCOPES

In future adding more sources to increasing use of AI and machine learning, podcast apps may become more advanced in their ability to personalize content recommendations and user experiences. By analyzing user behaviour and preferences, these apps can provide more accurate and targeted recommendations to users. As more smart devices become available, podcast apps may become integrated with other apps and devices, such as voice assistants, smart speakers, and car audio systems. This would make it easier for users to access and control their favourite podcasts across multiple devices. Improved content quality: With the growing popularity of podcasts, more content creators are entering the space, and the quality of content is likely to improve over time. This could lead to more high-quality, professional content that attracts new listeners and drives growth in the podcast industry. Some podcast apps may introduce interactive features that allow listeners to engage with the content in new ways, such as quizzes, polls, and interactive ads. This could make the listening experience more engaging and interactive for users.

APPENDIX

Source code:

https://github.com/vssubash500/Podcast_Plus

Code:

LoginActivity.kt

```
package com.example.podcastplayer
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)
   }}
```

MainActivity.kt:

```
package com.example.podcastplayer

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 = ""
)
}

                  }
                }

              }

            }
          }
}
```

RegistrationActivity.kt:

```
package com.example.podcastplayer

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)

          }
```