CONTENTS

- **INTRODUCTION**

The main objective of the podcast app project is to create a user-friendly and comprehensive platform that allows users to discover, stream, and manage their favorite podcasts, while also providing features such as personalized recommendations, easy-to-use search functionality, and a seamless listening experience across multiple devices.

1.1 Overview :

The podcast app project aims to create a platform for users to discover, subscribe, and listen to podcasts from various sources. The app will provide users with a personalized experience by recommending podcasts based on their listening history, interests, and preferences.

The app will have a user-friendly interface, allowing users to search for podcasts by name, topic, or keyword. It will also provide a feature for users to create and curate their playlists, enabling them to save their favorite episodes and share them with others.

The app will be integrated with various podcast hosting platforms, allowing users to access a wide range of podcasts from around the world. It will also provide a feature for users to rate and review podcasts, helping others to discover new and exciting content.

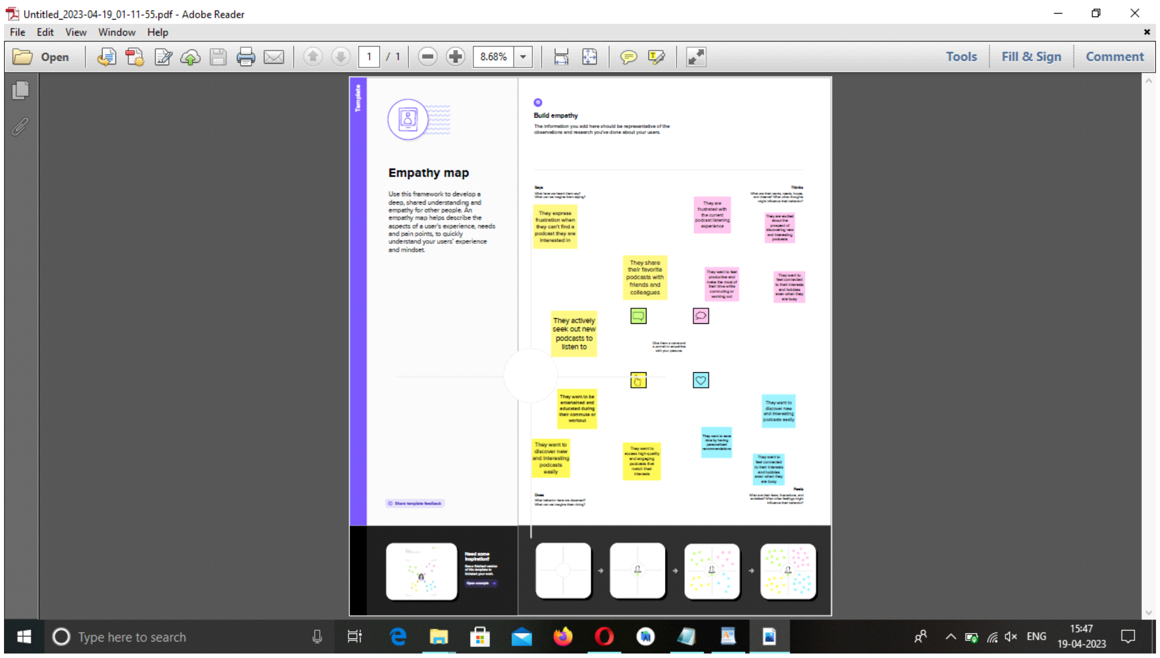
The app will have a social element, allowing users to connect with other podcast enthusiasts, share recommendations, and engage in discussions about their favorite podcasts. It will also provide a feature for users to follow their favorite podcasters and receive notifications about new episodes and updates.

1.2 Purpose :

purpose of the podcast app project is to create a user-friendly platform that empowers podcast enthusiasts to discover, subscribe, and listen to their favorite podcasts, while also fostering a vibrant community of podcasters and listeners. By providing personalized recommendations, social features, and seamless integration with various podcast hosting platforms, our app aims to enhance the listening experience and promote the growth of the podcasting industry**.**

- **PROBLEM DEFINITION & THINKING**

2.1 Empathy Map :


Template

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.

Build empathy

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

Says
What have we heard them say? What can we imagine them saying?

They express frustration when they can't find a podcast they are interested in

They share their favorite podcasts with friends and colleagues

They actively seek out new podcasts to listen to

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

They are frustrated with the current podcast listening experience

They are excited about the prospect of discovering new and interesting podcasts

They want to feel productive and make the most of their time while commuting or working out

They want to feel connected to their interests and hobbies even when they are busy

Give them a name and a portrait to empathize with your persona.

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

They want to be entertained and educated during their commute or workout

They want to discover new and interesting podcasts easily

They want to access high-quality and engaging podcasts that match their interests

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

They want to discover new and interesting podcasts easily

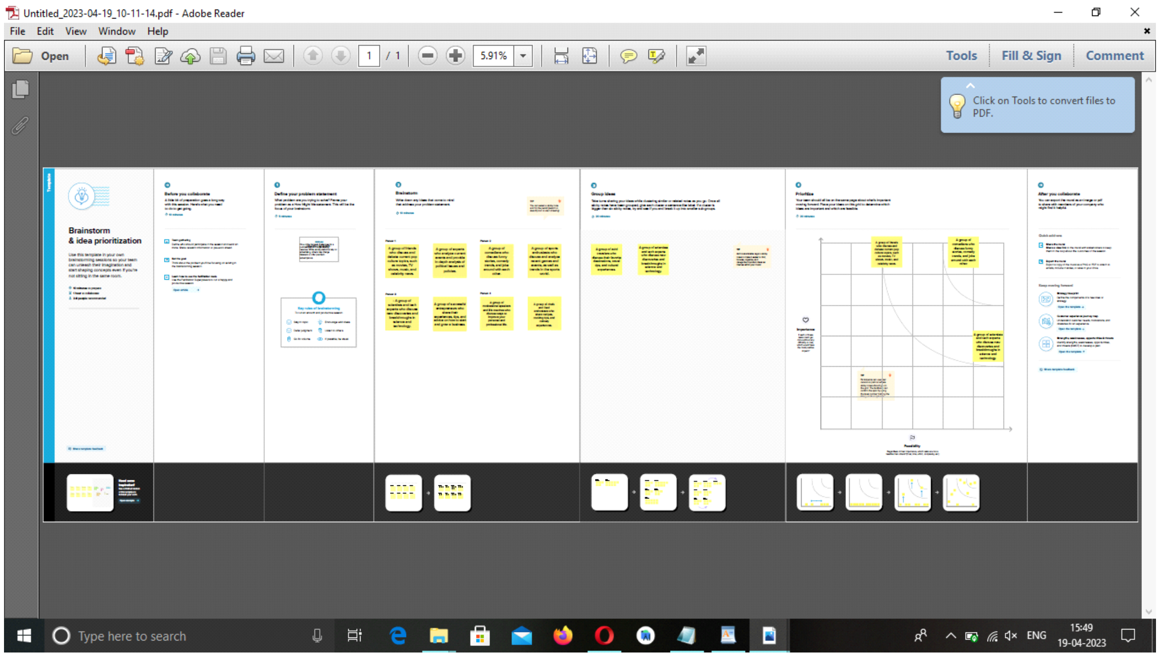
They want to save time by having personalized recommendations

They want to feel connected to their interests and hobbies even when they are busy

Share template feedback

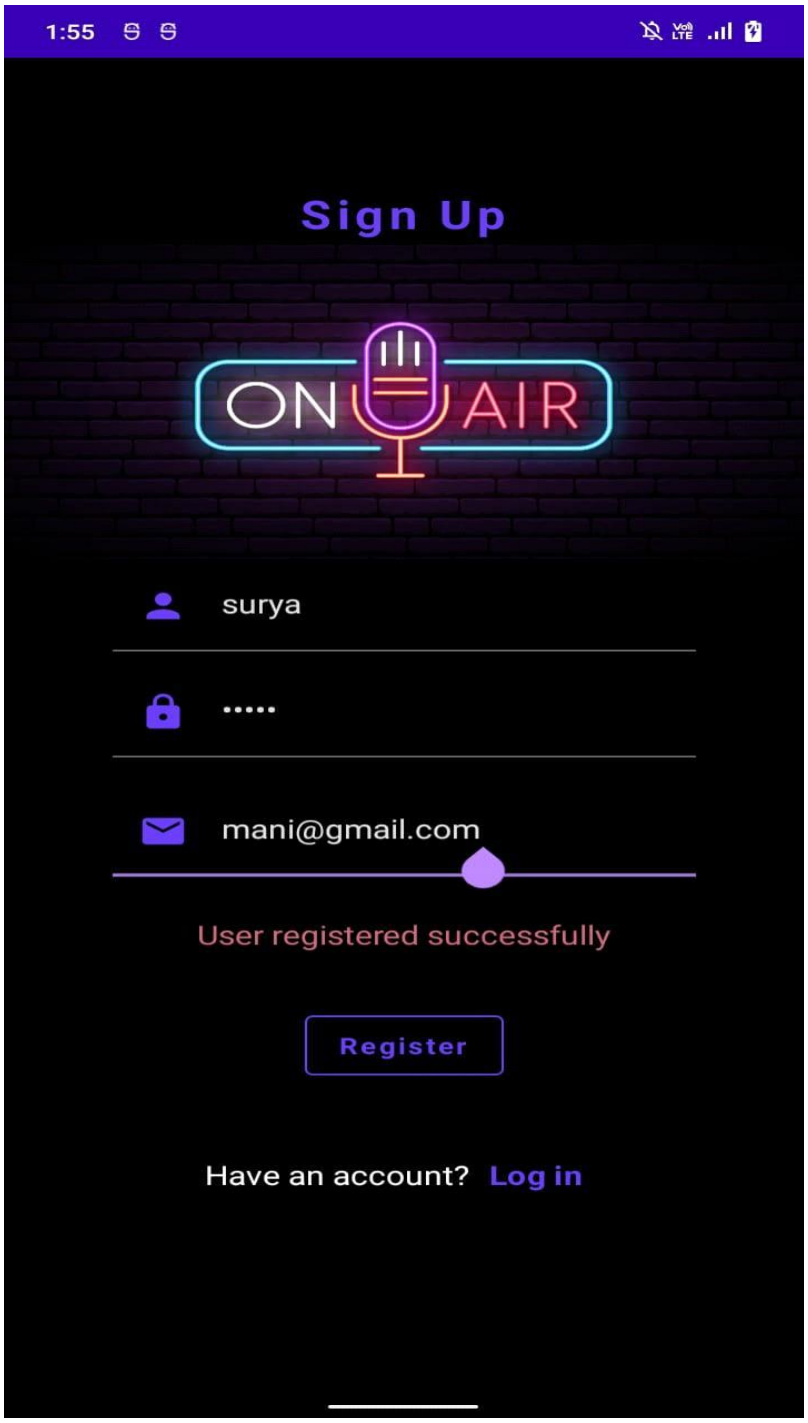
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example

2.2 Ideation & Brainstorming Map :


Untitled_2023-04-19_10-11-14.pdf - Adobe Reader
File Edit View Window Help
Open
1 / 1
5.91%
Tools
Fill & Sign
Comment
Click on Tools to convert files to PDF.
Brainstorm & idea prioritization
Type here to search
ENG
15:49
19-04-2023

- **RESULT**

Register Page :


1:55

Sign Up

surya

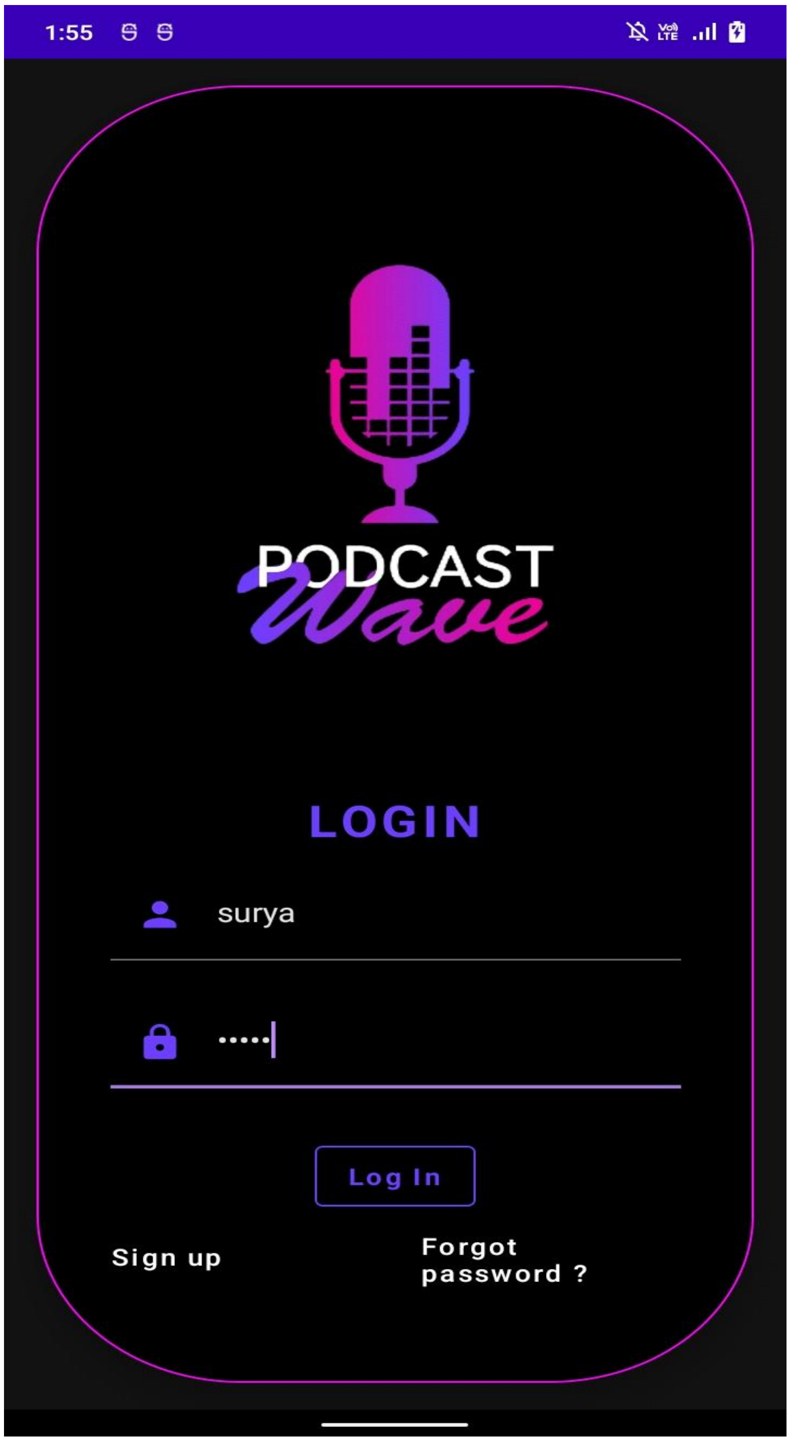
•••••

mani@gmail.com

User registered successfully

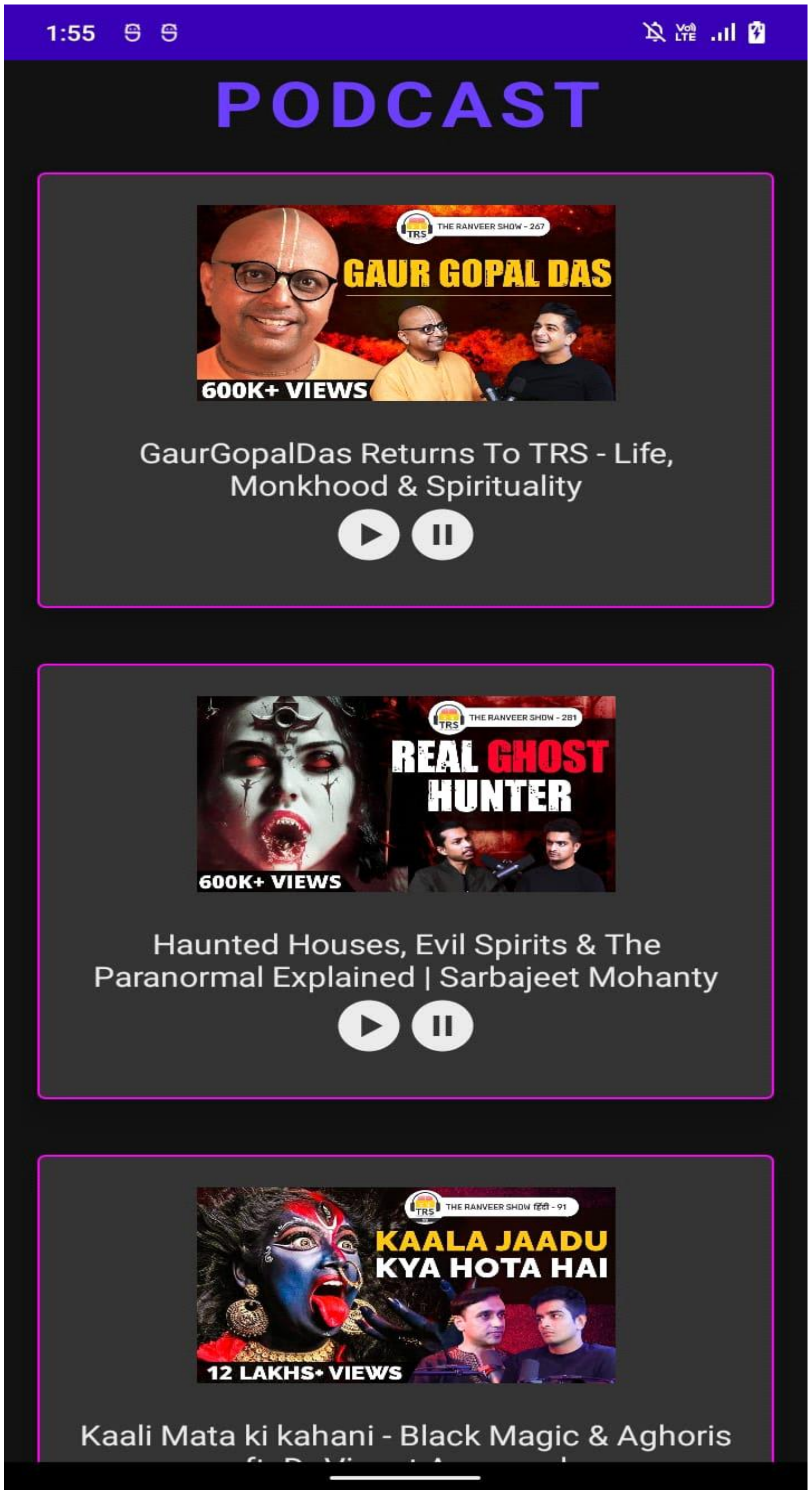
Have an account? Log in

Login Page :



Home Page :

Advantages :



1:55

PODCAST

GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality

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

Kaali Mata ki kahani - Black Magic & Aghoris

- **ADVANTAGES & DISADVANTAGES**

Advantages :

Large Audience: Podcasts are becoming increasingly popular, and a podcast app project can attract a large audience. This can help to build a user base and increase brand recognition.

Convenient and Accessible: A podcast app project allows users to listen to their favorite podcasts on-the-go, whether they are commuting, exercising, or doing other activities. This convenience and accessibility make it easier for users to engage with the content.

Personalization: A podcast app project can provide personalized recommendations based on a user's listening history and preferences. This can help to increase user engagement and satisfaction.

Monetization: A podcast app project can generate revenue through advertising, subscription models, or sponsorships. This can provide a sustainable revenue stream for the project.

Community Building: A podcast app project can help to build a community of podcast listeners and creators. This can foster a sense of belonging and engagement among users, which can lead to increased retention and growth.

Analytics: A podcast app project can provide valuable analytics about user behavior and engagement, which can help to optimize the app and improve the user experience.

Disadvantages :

Technical Challenges: Developing a podcast app can be technically challenging, especially when it comes to integrating with various podcast hosting platforms and managing large volumes of audio data.

Monetization: Monetizing a podcast app can be difficult, especially if the app is free to use. Ads and sponsorships are the most common ways to generate revenue from podcasts, but it can be challenging to attract advertisers without a significant audience.

- **APPLICATIONS**

- Listening to podcasts: The primary application of a podcast app is to allow users to easily discover, subscribe to, and listen to their favorite podcasts.
- Discovering new content: A podcast app can provide users with personalized recommendations based on their listening history, allowing them to discover new podcasts they may enjoy.
- Education and learning: Podcasts can be a valuable source of educational content, and a podcast app can provide users with access to a wide range of educational podcasts on topics such as history, science, and technology.
- Entertainment: Podcasts cover a wide range of genres, including comedy, true crime, and sports, providing users with a source of entertainment that can be enjoyed while commuting, working out, or doing other activities.
- Brand building and marketing: Companies and individuals can use a podcast app to create and distribute their own podcasts, building their brand and reaching new audiences.
- Community building: Podcasts can create a sense of community among listeners, and a podcast app can facilitate discussions and interactions among listeners through features such as comments and forums.

• CONCLUSION

In conclusion, a podcast app project can be a challenging but rewarding undertaking. Developing a podcast app requires careful planning and execution, as well as technical expertise to integrate with various hosting platforms and manage large volumes of audio data. Despite these challenges, a podcast app can have a variety of applications, including allowing users to listen to podcasts, discover new content, learn and educate themselves, entertain themselves, build their brand and market their products, and create a sense of community among listeners. By providing users with a convenient and accessible way to access a wide range of audio content, a podcast app can become a valuable tool for both individuals and businesses.

• FUTURE SCOPES

Personalization: One of the most significant trends in the podcast industry is personalization. As the amount of content available continues to grow, podcast apps will need to provide more personalized recommendations and features to help users discover the content they want.

AI and Voice Technology: The integration of artificial intelligence (AI) and voice technology into podcast apps can provide users with a more natural and intuitive way to interact with the app, as well as more personalized recommendations and content.

Monetization: As the podcast industry continues to grow, podcast apps will need to find new ways to monetize their content beyond advertising and sponsorships. Subscription models and paid content could become more prevalent.

Podcast Creation: Podcast apps could offer more tools for users to create their own podcasts, such as editing tools, sound effects, and music libraries.

APPENDIX

Source Code:

LoginActivity:

```
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:

```
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:

```
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?) {
```

```kotlin
        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)
        }
```