INTRODUCTION

Overview

A podcast is a program made available in digital format for download over the Internet. Podcasting is distinguished from other forms of digital audio content because it can be subscribed and easily accessed from a personal computer. Further, it can be downloaded to a mobile device, such as an Android and iPod where the user is able to listen to or watch it anywhere, anytime. Podcasting is used in new and evolving ways to publish information for healthcare education, patient care, professional continuing education, and in support of healthcare research.

Podcast Plus is not a term or concept that I am familiar with. It is possible that you may be referring to a specific podcast network or platform, but without further context, it's difficult for me to provide a detailed overview.

Generally speaking, podcasts are digital audio or video files that are made available online and can be downloaded or streamed. They can cover a wide range of topics, including news, entertainment, education, and more. Podcasts have become increasingly popular in recent years, with many people turning to them as a convenient way to stay informed and entertained.

If you have any more specific information about what you are referring to when you say "Podcast Plus", I would be happy to provide you with more detailed information.

Purpose

As I mentioned earlier, "Podcast Plus" is not a specific term or concept that I am familiar with. Therefore, I cannot provide a definitive answer to what its purpose might be.

However, in general, the purpose of podcasts is to provide listeners with informative or entertaining content on a particular subject or topic. Podcasts can be a way for individuals or organizations to share their knowledge, insights, and experiences with a wider audience. They can also be used as a marketing tool, as they can help to build brand awareness and establish thought leadership in a particular industry or niche.

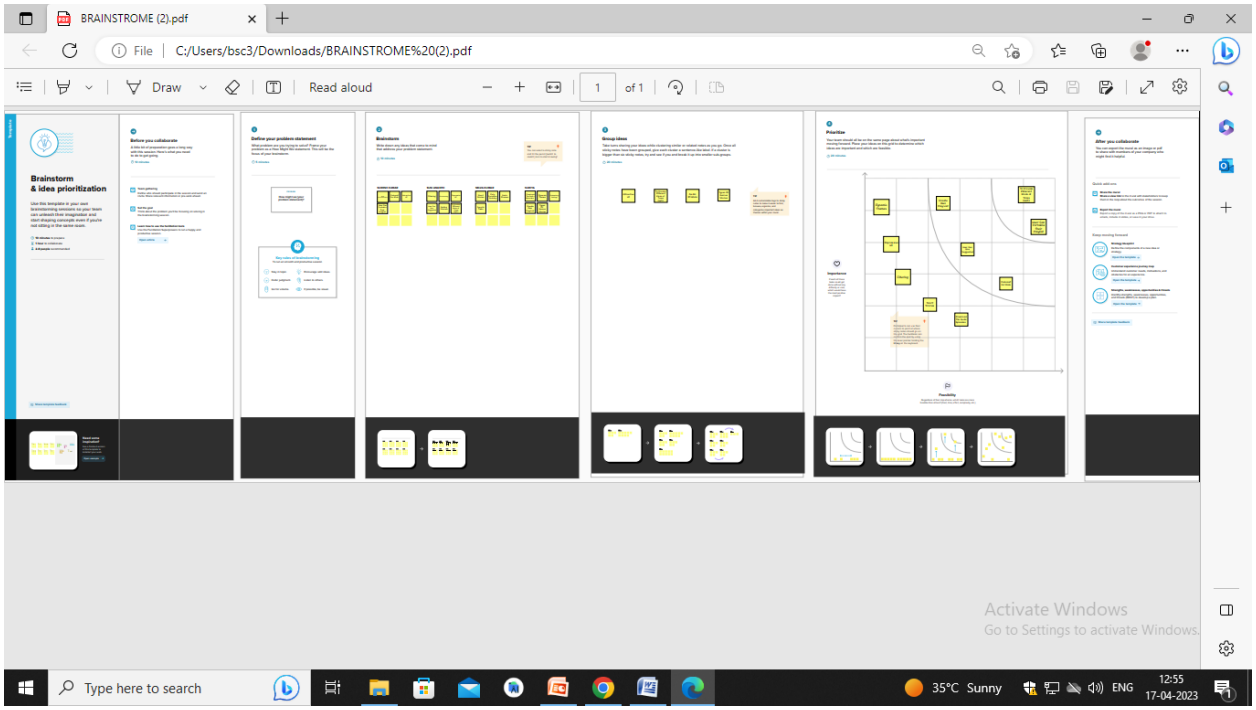
Podcasts can also serve as a way to connect with like-minded individuals or communities. Many listeners enjoy engaging with podcast hosts and other listeners through social media or other online platforms. This can create a sense of community and foster discussions around important issues or topics.

Again, without more specific information about what you mean by "Podcast Plus", it is difficult for me to provide a more detailed answer.

EMPATHY MAP

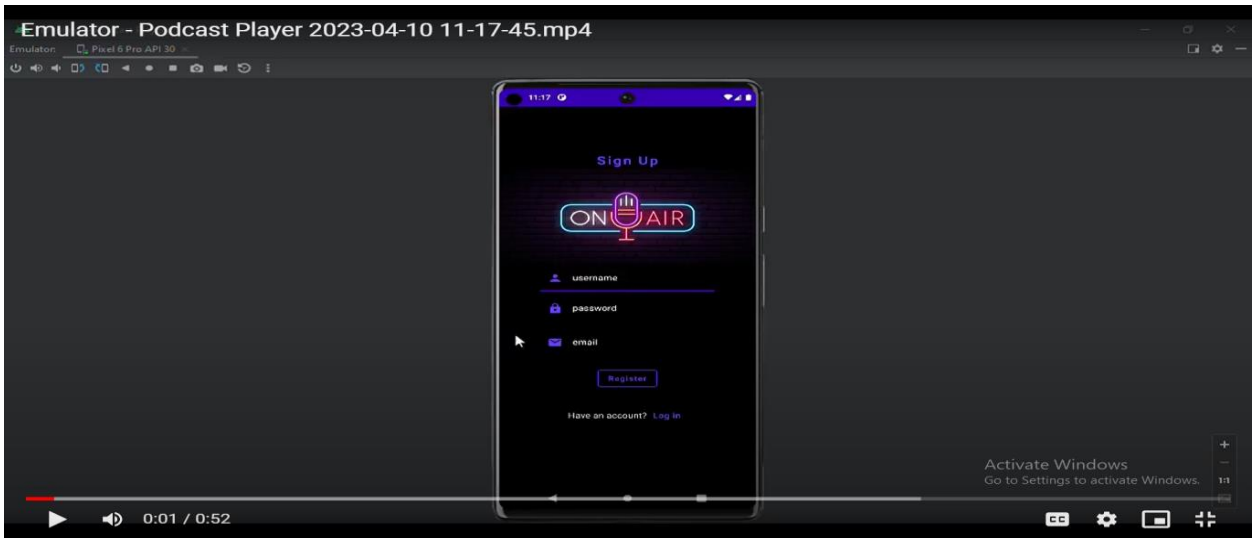

Template
Empathy map
Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.
Share template feedback
Need some inspiration?
See a finished version of this template to kickstart your work.
Open example
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 magine them saying?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
Download the audios
Feel Dynamic Themes
User Attractive
Attract a Large Audience
Build a Community Effortlessly
Podcast Can Get Personal
Podcast Plus: A Redux-Inspired Podcast App with Dynamic Themes For Android
Episode Title Optimization
podcast publishing
podcast recording
No Lagging Occure
User Don't Fear the Copiright
A Wish Man's Fear on Audible
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

BRAINSTORME

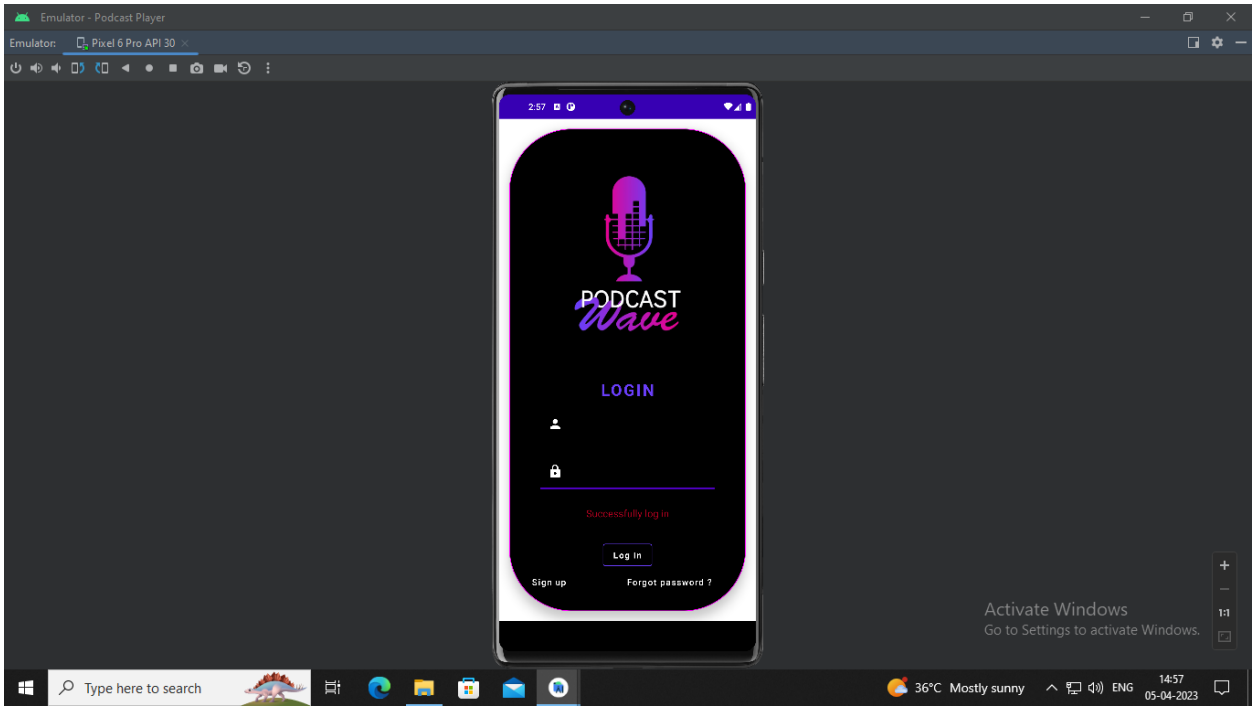


RESULT

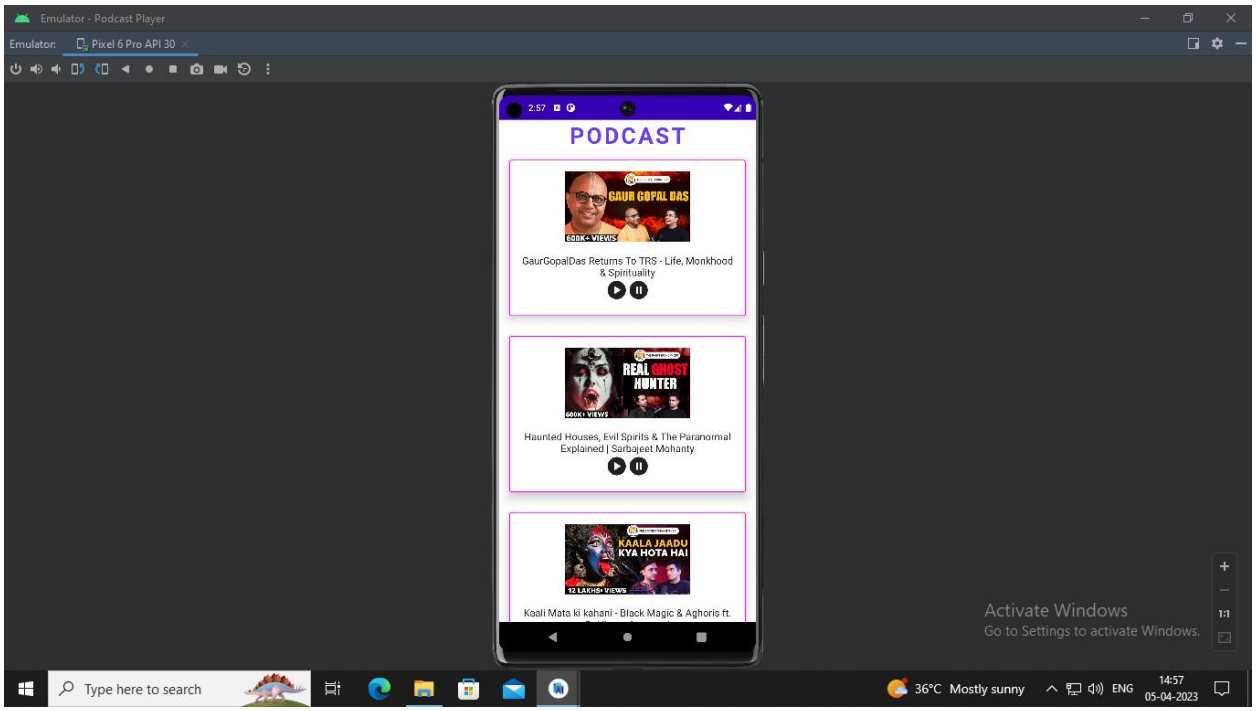
REGISTER



LOGIN



HOME PAGE


Emulator - Podcast Player
Emulator: Pixel 6 Pro API 30
PODCAST
GAUR GOPAL DAS
GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality
REAL GHOST HUNTER
Haunted Houses, Evil Spirits & The Paranormal Explained | Sarbajeet Mohanty
KAALA JAADU KYA HOTA HAI
Kaali Mata ki kahani - Black Magic & Aghoris ft.
Activate Windows
Go to Settings to activate Windows.
Type here to search
36°C Mostly sunny
ENG
14:57
05-04-2023

ADVANTAGE

Podcasts offer several advantages, including:

- Convenience: Podcasts are easy to listen to and can be accessed anytime, anywhere, using a smartphone, tablet, or computer.
- Variety: There are podcasts available on a wide range of topics, from news and current events to entertainment and self-improvement.
- Learning: Podcasts are an excellent way to learn new things and stay up-to-date on the latest trends and developments in your field of interest.
- Entertainment: Podcasts can be a fun and engaging way to pass the time and provide a source of entertainment.
- Flexibility: Unlike live radio shows, podcasts can be paused, rewound, and fast-forwarded, allowing listeners to control the pace and timing of their listening experience.
- Cost-effective: Most podcasts are free to listen to, making them an affordable alternative to other forms of entertainment or learning.
- Community: Podcasts can create a sense of community among listeners who share similar interests, providing an opportunity to connect with like-minded individuals from around the world.

DISADVANTAGE

While podcasts have numerous advantages, there are also some disadvantages, including:

- Lack of interactivity: Unlike live events or interactive discussions, podcasts are a one-way medium, which means that listeners cannot ask questions or engage in discussions with the host or guests.
- Requires internet access: Listening to podcasts requires a stable internet connection, which may not always be available or reliable in certain areas.
- Limited accessibility: Some people with hearing impairments may not be able to fully enjoy podcasts without subtitles or transcripts.

- Information overload: With the abundance of podcasts available, it can be overwhelming to choose which ones to listen to, leading to a sense of information overload.
- Inconsistent quality: Since anyone can create a podcast, the quality of content can vary widely, making it difficult for listeners to find high-quality, reliable sources of information.
- Time-consuming: Podcasts can range from a few minutes to several hours in length, and finding the time to listen to them all can be challenging, especially for people with busy schedules.
- Addiction: Listening to podcasts can be addictive, leading to binge-listening and potentially interfering with other activities and responsibilities.

APPLICATION

A podcast application is a software program that allows users to discover, download, and listen to podcasts on their smartphones, tablets, or computers. Some of the most popular podcast applications include:

- Apple Podcasts: This is the default podcast app on iPhones and iPads, and it comes pre-installed on all iOS devices.
- Spotify: This popular music streaming service also offers a vast selection of podcasts, making it an all-in-one platform for music and audio content.
- Google Podcasts: This free app for Android devices is designed to help users discover and listen to podcasts easily.
- Stitcher: This podcast app offers a wide range of podcasts on various topics, including news, politics, sports, and entertainment.
- Pocket Casts: This popular app is available on both iOS and Android and offers a sleek and user-friendly interface that makes it easy to discover and listen to podcasts.
- Overcast: This app is available only on iOS devices and is known for its advanced features, including voice boost, smart speed, and playlists.
- Podbean: This app is available on both iOS and Android and offers a vast selection of podcasts, including both popular and indie shows.

These are just a few examples of the many podcast applications available, each with its own unique features and benefits. Users can choose the one that best fits their needs and preferences.

CONCLUSION

In conclusion, podcasts have become an increasingly popular form of media that offers numerous advantages, such as convenience, variety, learning, entertainment, flexibility, cost-effectiveness, and community. However, there are also some disadvantages to consider, such as the lack of interactivity, the need for internet access, limited accessibility, inconsistent quality, information overload, time consumption, and addiction.Despite these drawbacks, the popularity of podcasts continues to grow, and there are now numerous podcast applications available that make it easy to discover, download, and listen to podcasts. Whether you are interested in news and current events, entertainment, self-improvement, or any other topic, there is likely a podcast out there that will pique your interest.

FUTURE SCOPE

The future scope of podcasts looks promising, as they continue to gain popularity and evolve with new technologies and trends. Here are some potential future developments in the world of podcasts:

- Increased Diversity: As the popularity of podcasts continues to grow, we can expect to see more diverse content creators and topics, catering to a broader range of interests and audiences.
- Interactive Features: New technologies like voice assistants, artificial intelligence, and augmented reality could allow for more interactive podcast experiences, such as real-time Q&A sessions or interactive storytelling.
- Monetization: With the growth of podcast listenership, we can expect to see more monetization opportunities for podcast creators, such as sponsorships, advertising, or subscription-based models.
- Enhanced Production Quality: As podcasting becomes more mainstream, we can expect to see a higher standard of production quality, with better audio equipment, editing, and storytelling techniques.
- Integration with Other Media: Podcasts may increasingly integrate with other media forms, such as video or written content, to offer a more comprehensive and engaging experience for listeners.

Overall, the future of podcasts looks bright, with plenty of opportunities for growth and innovation. As technology and user preferences evolve, we can expect to see new and exciting developments in the world of podcasting.

SOURCE CODE

```
 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.White
        )
    },
    placeholder = {
        Text(
            text = "username",
            color = Color.White
        )
    },
    colors = TextFieldDefaults.textFieldColors(
        backgroundColor = Color.White
    )

)

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

TextField(
    value = password,
    onValueChange = { password = it },
```

```
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Lock,
            contentDescription = "lockIcon",
            tint = Color.White
        )
    },
    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.White)
        }

        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)
  }}
```

MAIN

```
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 = ""
                )
            }

        }
    }

}
```

REGISTRATION

```
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(0xFFEFEDF7),
            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(0xFFF9F9FA)
            )
        },
        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(0xFFFEFEFF)
            )
        },
        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(0xFFF8F7FC)
            )
        },
        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)
}
```

USER

```
package com.example.podcastplayer

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?,

    )
```

USER DAO

```
package com.example.podcastplayer

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)
}
```

USER 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
    }
}
    }
}
```

USER DATABASE 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
    }

}
```