Podcast Plus: A Redux-Inspired Podcast App With Dynamic Themes For Android

1. Introduction

1.1Overview:

Podcasts are audio files or series of episodes that are available on the internet for downloading or streaming. They can cover various topics, such as politics, culture, celebrities, drama, and comedy. Podcasts can be listened to on different devices and at different times, unlike radio or TV. To listen to podcasts, listeners need to install an app and subscribe to the shows they want to hear.

To get yourself noticed, you need to create a podcast website. This way, you can gain new listeners, promote your site effectively, and make a single point of access for all your audio files. In addition, you can connect with other podcasters who share expertise in your niche. This will create an opportunity for you to collaborate with others. Hence, you can invite them for a talk on a specific show and maintain your consistency concerning the release schedule.

The best podcasts in your niche help you discover the wants of your subscribers. You can analyze the other competitors to know the length of the episodes, their content delivery style, etc.

The name you decide for your podcast must entice your audience to click and listen. Hence, please keep it simple. Additionally, take advantage of search engine optimization and use relevant keywords in your content to help new listeners find you.

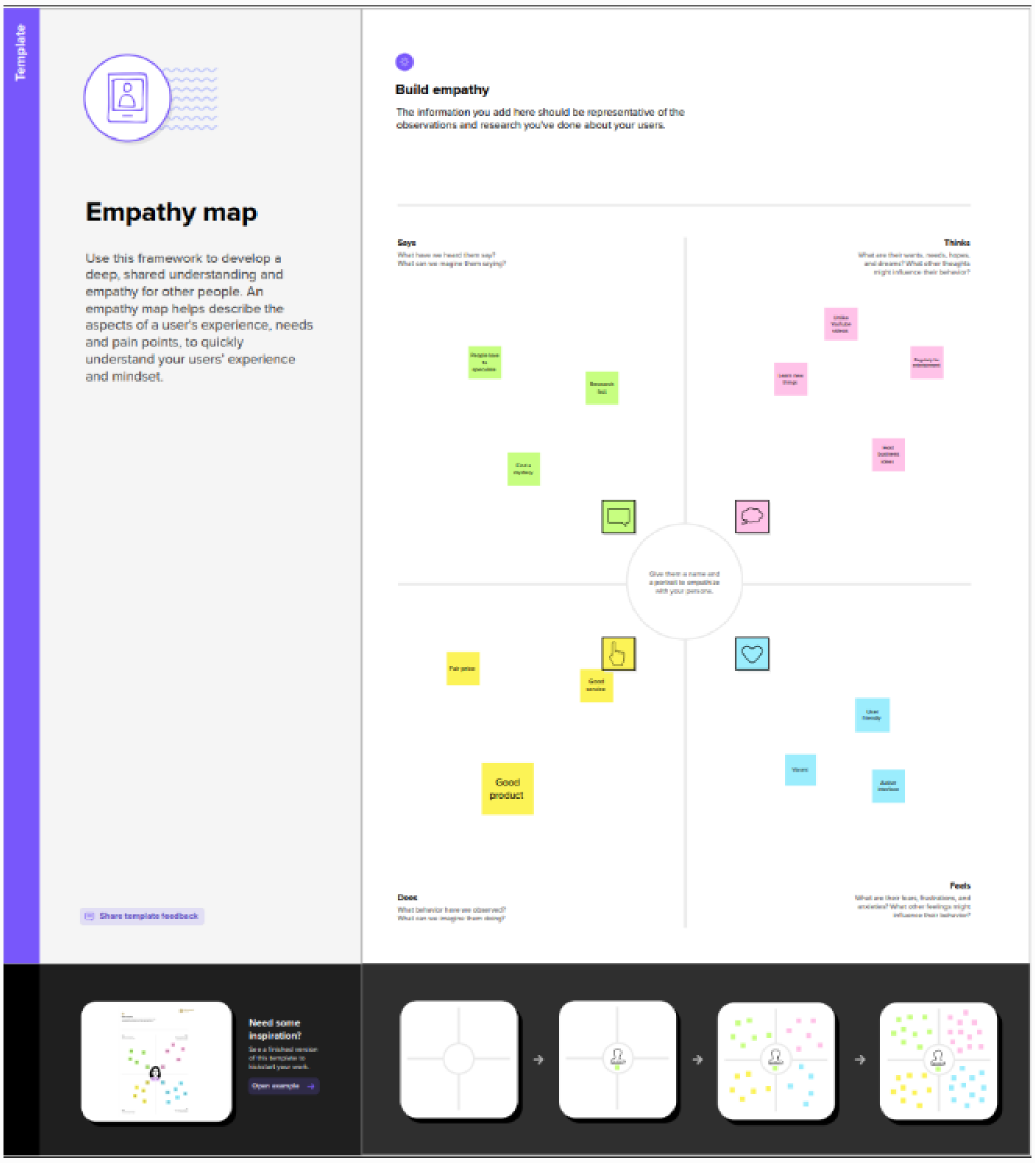
1.2Purpose:

A podcast is essentially an audio broadcast you can listen to on your computer or mobile device. Many people think of podcast as radio shows you can download and listen to at your convenience, but they can be so much more! podcast can be used for entertainment, education, or even marketing purpose.

A podcast generator maintains a central list of the files on a server as a web feed that one can access through the Internet. The listener or viewer uses special client application software on a computer or media player, known as a podcast client, which accesses this web feed, checks it for updates, and downloads any new files in the series. This process can be automated to download new files automatically, so it may seem to listeners as though podcasters broadcast or "push" new episodes to them. Podcast files can be stored locally on the user's device, or streamed directly. There are several different mobile applications that allow people to follow and listen to podcasts. Many of these applications allow users to download podcasts or stream them on demand. Most podcast players or applications allow listeners to skip around the podcast and to control the playback speed.

2. Problem Definition & Design Thinking:

2.1Empathy 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 imagine them saying?

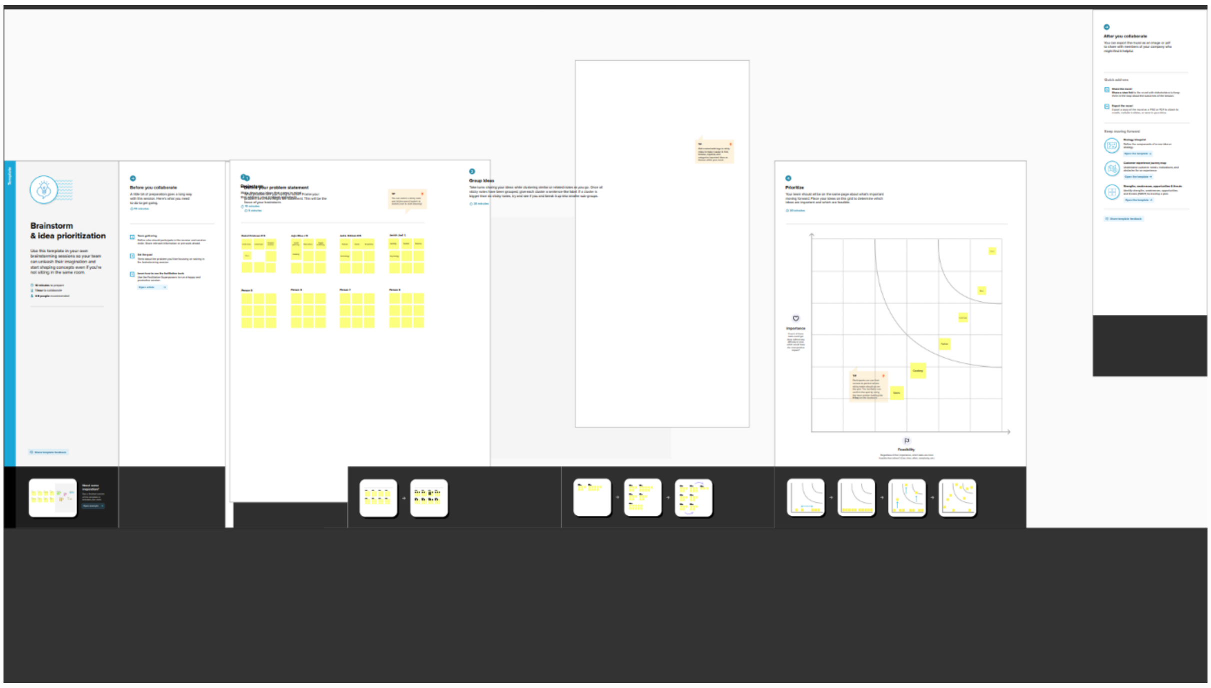
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

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

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?

2.2Ideation & Brainstroming



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

Define your problem statement

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

Importance

Feasibility

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

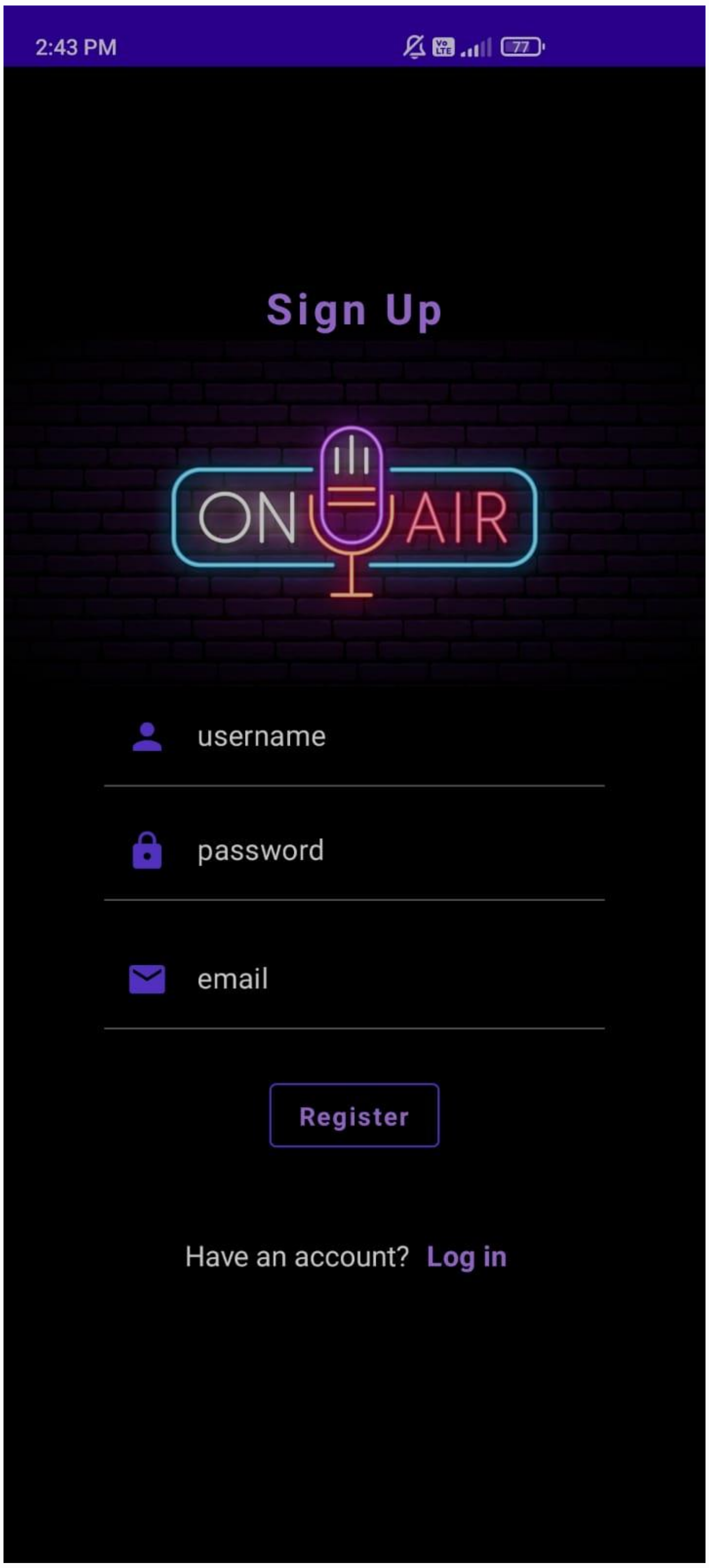
Quick add-ons

Keep moving forward

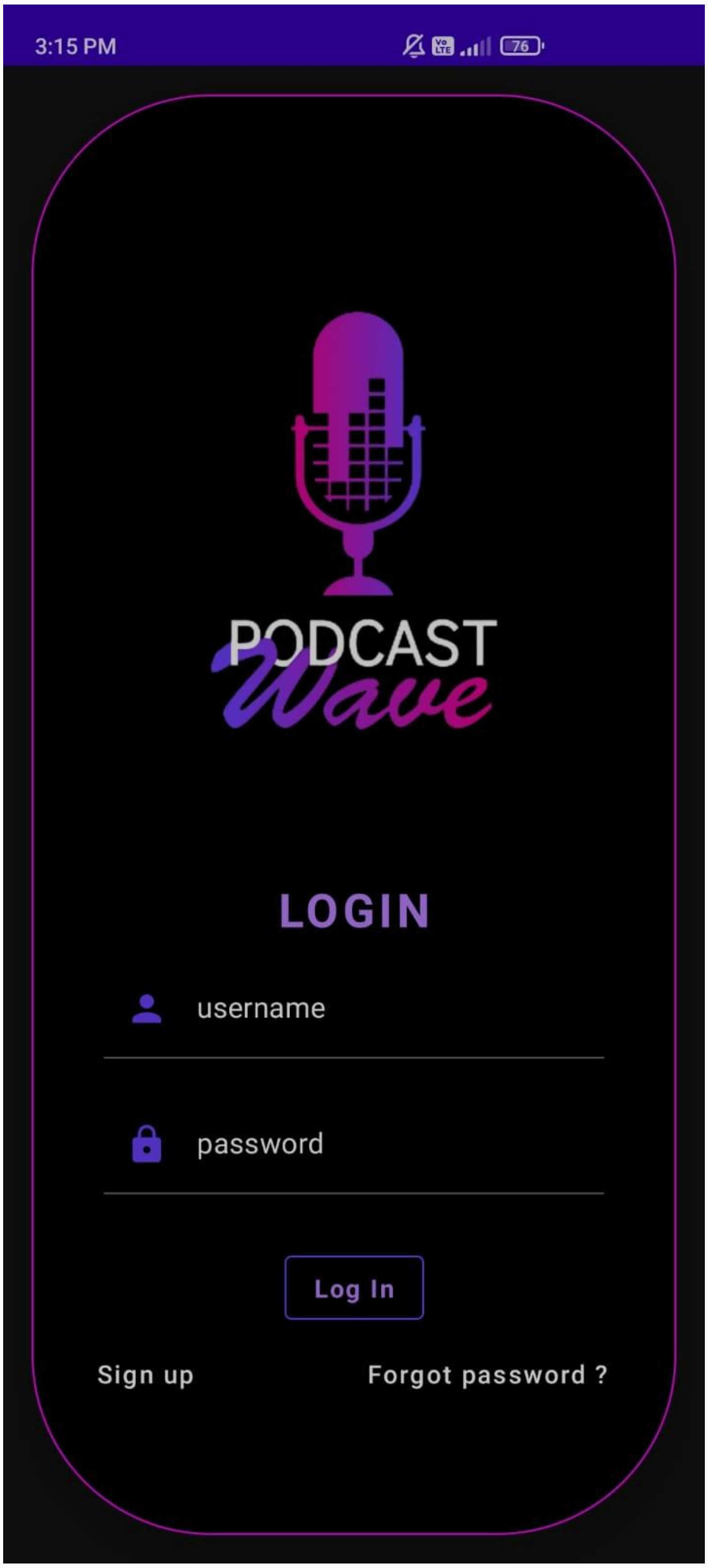
3.Result:

Final Output of the Application :

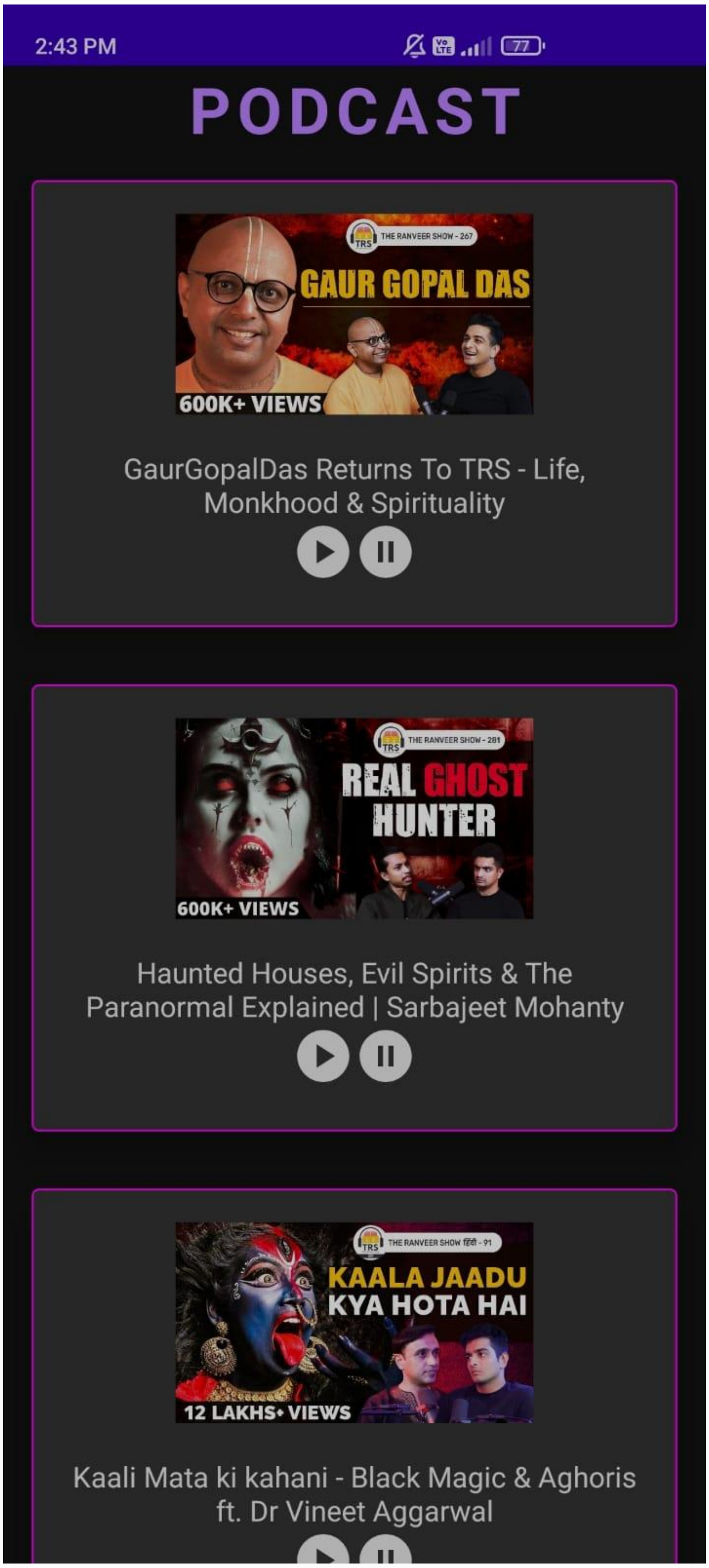
Signup page:



Login page:


3:15 PM
76
PODCAST
Wave
LOGIN
username
password
Log In
Sign up
Forgot password ?

Main page:

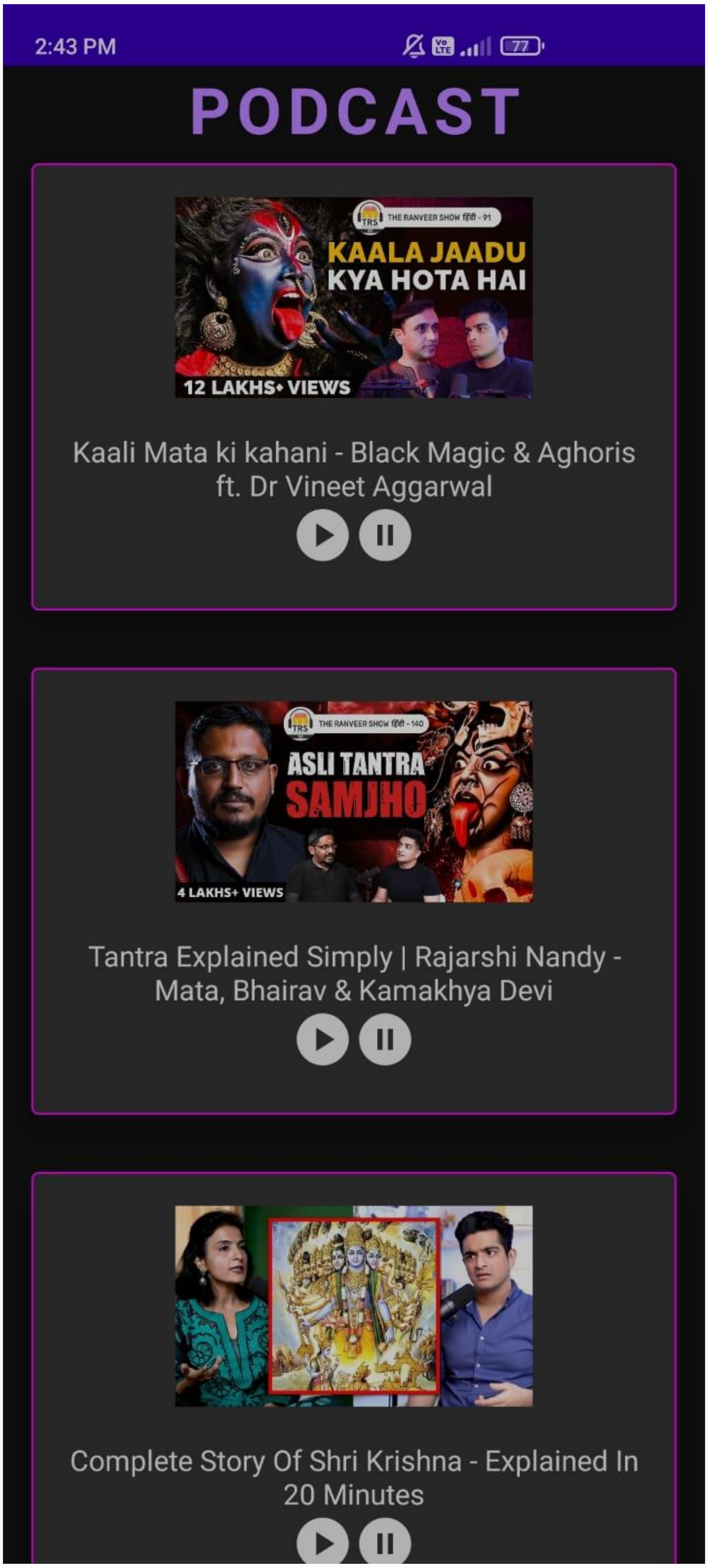

2:43 PM

PODCAST

GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality

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

Kaali Mata ki kahani - Black Magic & Aghoris ft. Dr Vineet Aggarwal



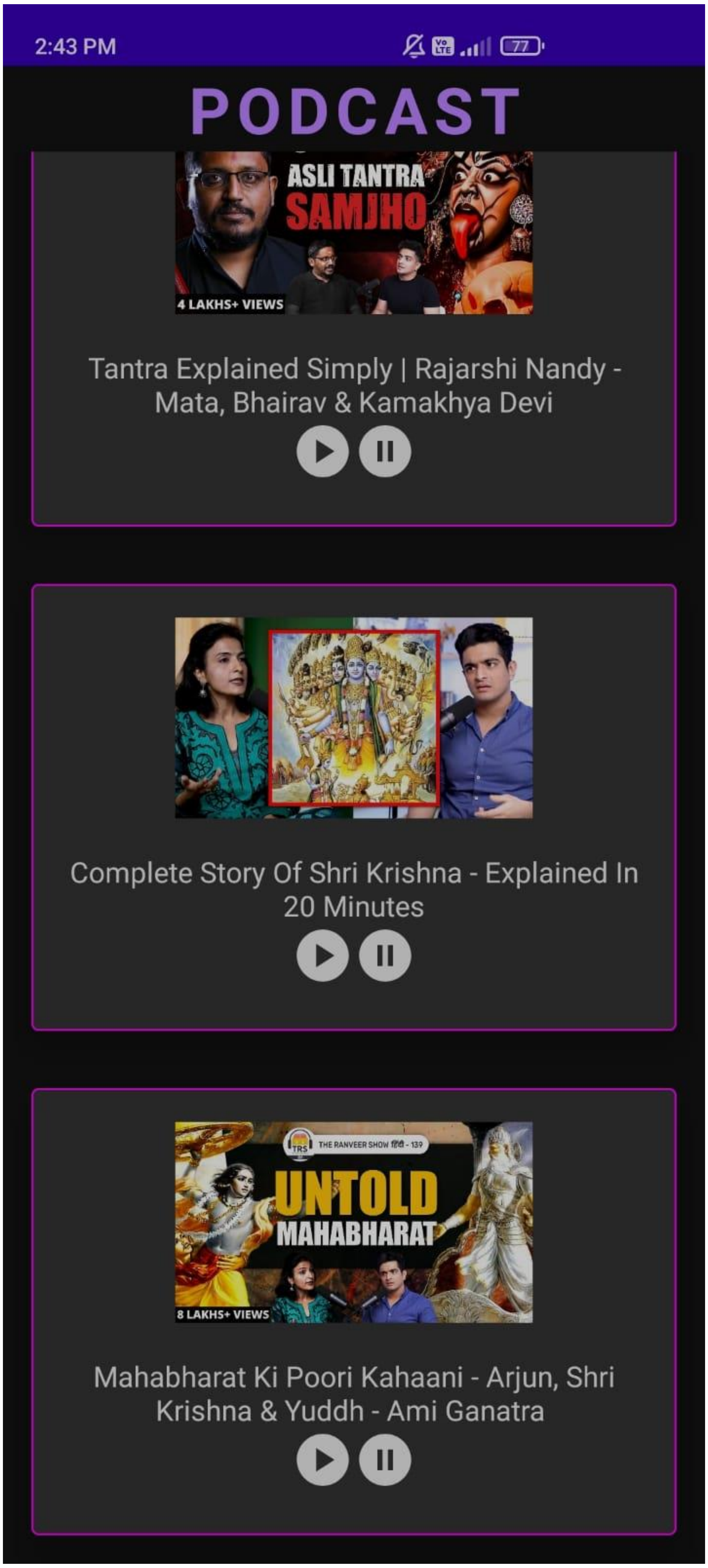
2:43 PM

PODCAST

Kaali Mata ki kahani - Black Magic & Aghoris ft. Dr Vineet Aggarwal

Tantra Explained Simply | Rajarshi Nandy - Mata, Bhairav & Kamakhya Devi

Complete Story Of Shri Krishna - Explained In 20 Minutes



2:43 PM

PODCAST

Tantra Explained Simply | Rajarshi Nandy - Mata, Bhairav & Kamakhya Devi

Complete Story Of Shri Krishna - Explained In 20 Minutes

Mahabharat Ki Poori Kahaani - Arjun, Shri Krishna & Yuddh - Ami Ganatra

4. Advantage & Disadvantage

*Advantage of podcast:

One of the benefits of podcasts is that you don't have to worry that your audience is busy offline when you make your sales pitch.

A listener can download or stream the file and replay it at whatever time is convenient for him, even while jogging or driving to work.

If someone chooses to subscribe to your podcast feed, he can get any podcasts of interest downloaded automatically.

If you want all your employees to hear what you have to say, podcasting is often easier than trying to get everyone together for one meeting.

*Disadvantage of podcast:

If you want to reach an audience have access to the Internet, then podcasting isn't the way to do it.

Podcasts – especially video podcasts, according to the University of Minnesota Duluth – are often large files, which also makes them hard to access for anyone with a slow Internet connection.

If you want to make an audio podcast available to a customer who's hearing-impaired or anyone who doesn't have a compatible player, you need to turn the material into a written transcript as well.

As the importance of podcasts grows though, accessibility issues are mitigated.

5. Application:

- Podcasts offer a convenient way to catch up on news, sports, and special interests at home or on the go.
- There are many apps with tons of podcasts you can listen to for free or with a paid subscription.
- We recommend services like Audible, Apple Podcasts, Spotify, Google Podcasts, and more.

6. Conclusion:

With the increase of easy-to-use programs, and the increase of mobile technology, we are able to create, access, and listen to podcasts wherever we go. There is a huge selection of podcasts being offered to consumers, with new ones being offered everyday. No matter what the subject you are interested in, chances are there is a podcast for it.

Podcasts offer educators and students new ways to share knowledge and communicate about understandings. I hope that this learning object has increased your understanding of podcasting and has allowed you to consider using them in your life.

As a concluding activity please complete the poll below. Also, if you are interested you can deepen your understanding of podcasting using the resources collected

7. Future scope:

The future of podcasting is filled with so much excitement and innovation, it's almost hard to keep up with what's next. Podcasting has grown so much that about 197 million Americans know about them. That's about 70% of US citizens that are aware of podcasts.

To further understand how much podcasting continues to grow, there are two million podcasts registered on Google. That's a whole lot of content out there! And the industry is only going up from here.

The proof of this predicted growth in podcasting is apparent in the trends we've picked up and listed below. From these trends, doubt the future of podcasting is filled with dynamic ideas and innovation

8. Appendix:

A.Source 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)

}
```

User.kt

```
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?,


    )
```

UserDao.kt

```
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)
}
```

UserDatabase.kt

```
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
            }
        }
    }
}
```

UserDatabaseHelper.kt

```
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

    }


}
```