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

Team ID :NM2023TMID12296

Team Leader : ARCHANA R

Team Members :

ASHIKA ASFANA A

DEEPA S

HEMALATHA R

VEMBU A

1.INTRODUCTION:

1.1 Overview:

Podcast Plus is an online platform that offers a curated selection of high-quality podcasts across various categories such as news, entertainment, sports, technology, and more. It is a subscription-based service that allows users to access exclusive content, ad-free listening, early access to new episodes, and personalized recommendations based on their listening history.

1.2 Purpose:

Improved user experience: By using the Redux architecture and dynamic themes, the app could offer a more user-friendly and personalized experience for podcast listeners.

Increased engagement and retention: A well-designed podcast app with advanced features and customization options could help to attract and retain users.

Potential revenue generation: Depending on the app's business model, a Redux-inspired podcast app could generate revenue through advertising, subscriptions, or in-app purchases.

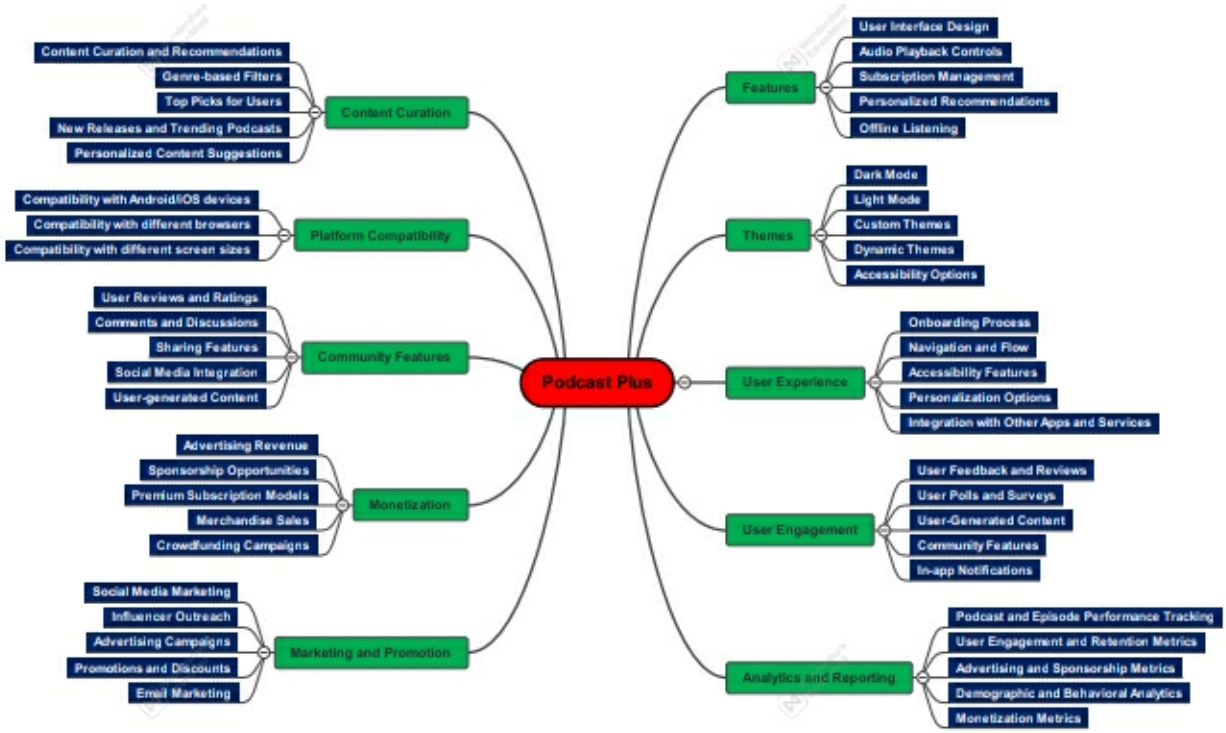
Learning and skill development: Developing a project like this could provide an opportunity for developers to learn new skills and techniques, such as using Redux and implementing dynamic themes.

2.Problem Definition & Design Thinking:

2.1 Empathy Map:

Source: XPLANE.com

2.2 Ideation & Brainstorming Map:



3. RESULT:

4. ADVANTAGES AND DISADVANTAGES:

4.1 ADVANTAGES:

1) Personalized recommendations based on user preferences and listening history.
2) Dynamic themes that adapt to user preferences and context for a more engaging experience.
3) Efficient state management using the Redux architecture for a smoother and more responsive user experience.

4.2 DISADVANTAGES:

1) Accessibility for some audience can be an issue. Internet is required for people to access the podcasts and it becomes difficult to reach to a wider audience if internet is not available.
2) Finding and reaching to your audience.
3) IP and content protection is difficult.

5. APPLICATIONS:

- Efficient playback of podcast episodes: The app's state management system inspired by Redux ensures that podcasts are played smoothly and without interruptions.
- Customizable visual interface: The dynamic themes feature allows users to customize the app's visual interface according to their preferences.
- Enhanced user experience: The combination of advanced features and a user-friendly interface makes Podcast Plus an excellent app for those who want an exceptional podcast listening experience.
- Time-saving features: Podcast Plus allows users to download episodes for offline listening, which can save time and data usage.

6. CONCLUSION:

Podcast Plus is an Android app designed to provide an exceptional podcast listening experience. It boasts advanced features such as a state management system inspired by Redux and dynamic themes that make it highly customizable and visually appealing. With smooth playback of podcast episodes and a user-friendly interface, Podcast Plus is an excellent choice for Android users who are looking for a high-quality podcast app.

7. FUTURE SCOPE:

Recommendation system: Implement a recommendation system that suggests new podcasts to users based on their listening history, search queries, and interests. This could be powered by machine learning algorithms that analyze user data to generate personalized recommendations.

Social features: Add social features to the app, such as the ability to follow other users, share podcasts with friends, and see what others are listening to. This could increase user engagement and create a sense of community within the app.

Advanced playback controls: Add advanced playback controls, such as variable speed playback, customizable skip intervals, and automatic silence trimming. This could provide users with greater control over their listening experience and make it easier to consume more content in less time.

8. APPENDIX:

Source Code:

Podcast Plus: A Redux-Inspired Podcast App with Dynamic Themes for Android

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/podcast_icon"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.PodcastPlayer"
        tools:targetApi="31">
```

```
        <activity
            android:name=".RegistrationActivity"
            android:exported="false"
            android:label="@string/title_activity_registration"
            android:theme="@style/Theme.PodcastPlayer" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="@string/title_activity_login"
            android:theme="@style/Theme.PodcastPlayer" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.PodcastPlayer">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Color.kt

```
package com.example.podcastplayer.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)
```

Shape.kt

```
package com.example.podcastplayer.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(4.dp),
    large = RoundedCornerShape(0.dp)
)
```

Theme.kt

```
package com.example.podcastplayer.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200

    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
)

@Composable
fun PodcastPlayerTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

Type.kt

```
package com.example.podcastplayer.ui.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)
```

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 = ""
                        )
                    }

                }
            }

        }
```

```
        }
    }
}
```

RegistrationAcitivty.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
```

```kotlin
@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

```kotlin
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
    }

}
```