PODCAST PLUS APP

R.Rajeshwari
H.Thasneem Fathima
P.Sivaranjani
M.Lavanya

1.Introduction

1.1 overview

A project that demonstrates the use of Android Jetpack Compose to build a UI for a podcast player app. The app allows users to choose , play and pause podcasts.

Project Workflow:

- ***Users register into the application.***

- ***After registration , user logins into the application.***

- ***User enters into the main page***

- ***The app allows users to choose , play and pause podcasts.***

1.2 purpose

A podcasts app is designed to provide users with a convenient way to discover, download, and

listen to audio content that is typically in the form of episodic series. Podcasts can cover a wide range of topics, such as news, entertainment, education, sports, politics, technology, and more.

The primary purpose of a podcasts app is to make it easy for users to find and subscribe to podcasts that interest them. The app can provide curated lists of popular podcasts or suggest new ones based on a user's

listening history. Once subscribed, the app can automatically download new episodes as they become available, allowing users to listen offline at their convenience.

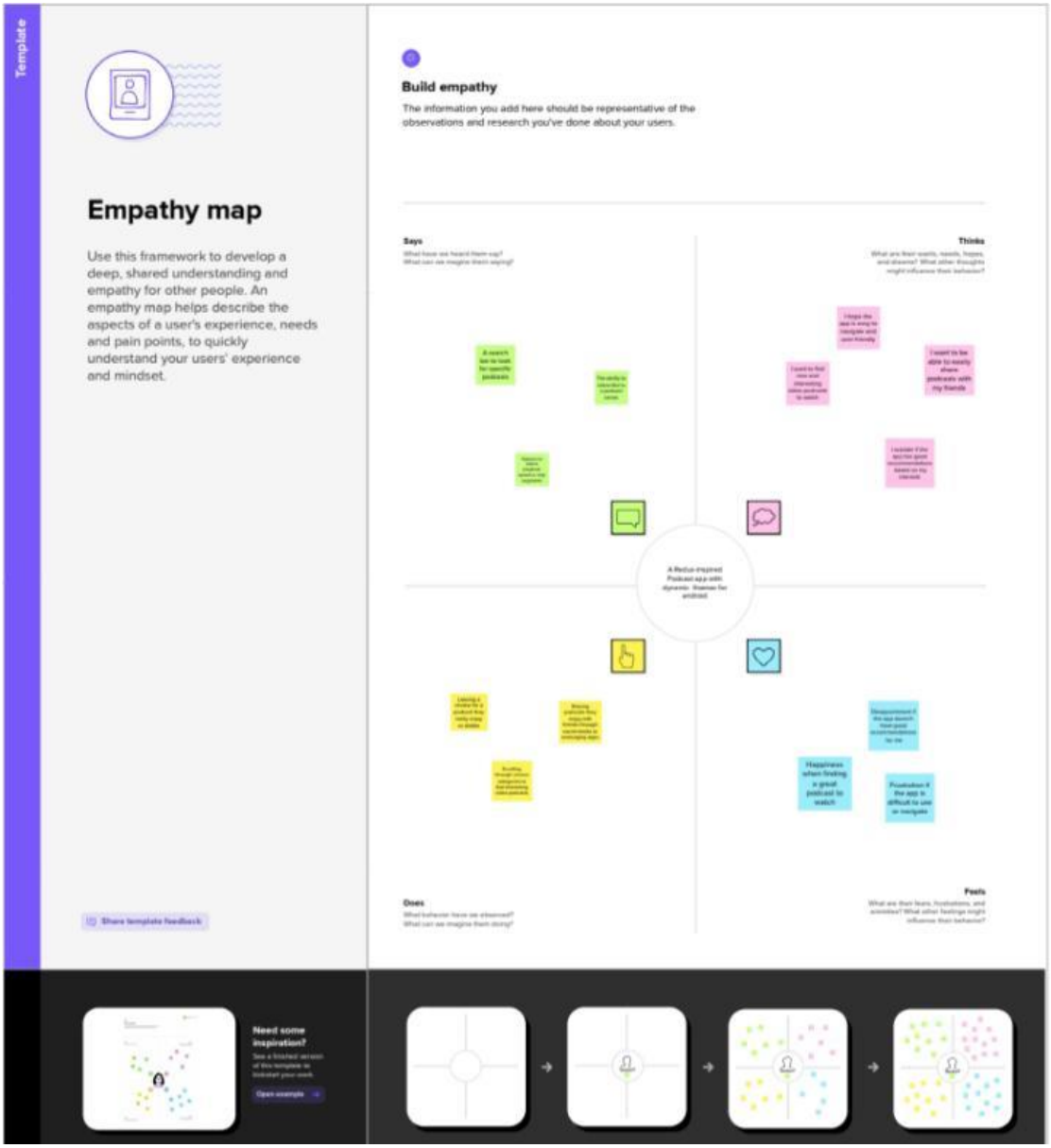
Podcasts apps can also offer features such as playback speed control, sleep timers, bookmarks, and sharing options. Additionally, some apps may provide a social aspect, allowing users to follow and connect with other listeners

or hosts, leave reviews and ratings, and participate in discussions.

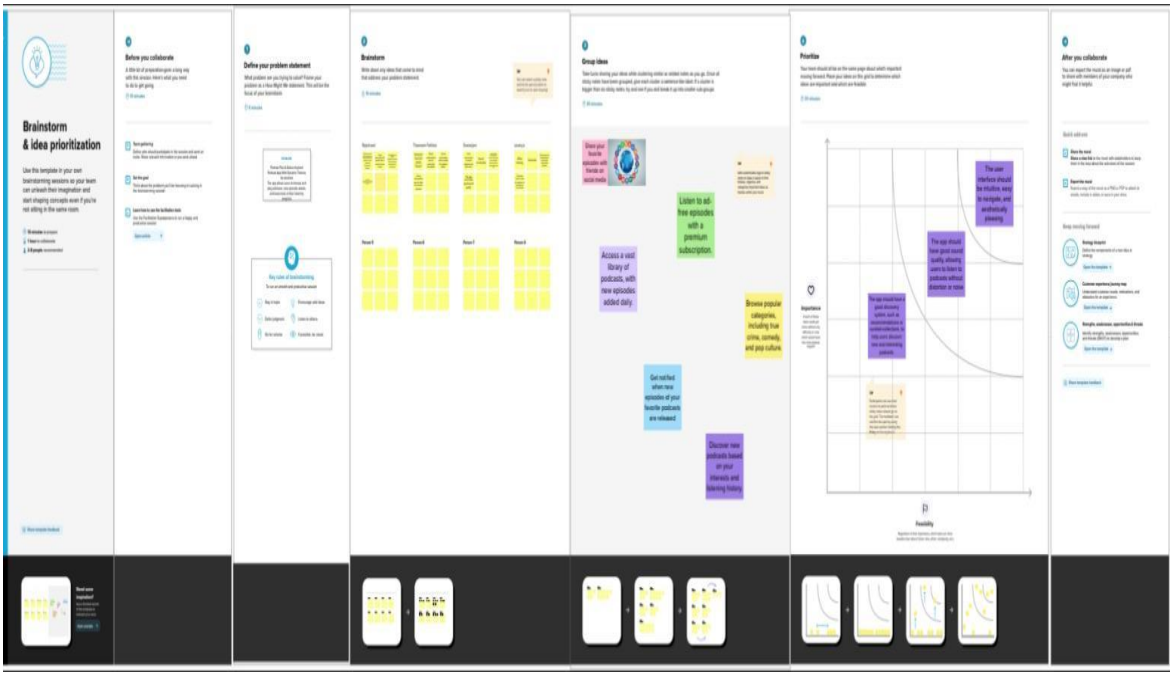
Overall, the purpose of a podcasts app is to provide a streamlined and personalized listening experience for users, helping them discover and consume audio content on their own terms.

2.problem definition and design thinking

2.1 Empathy map



2.2 ideation &Brainstorming

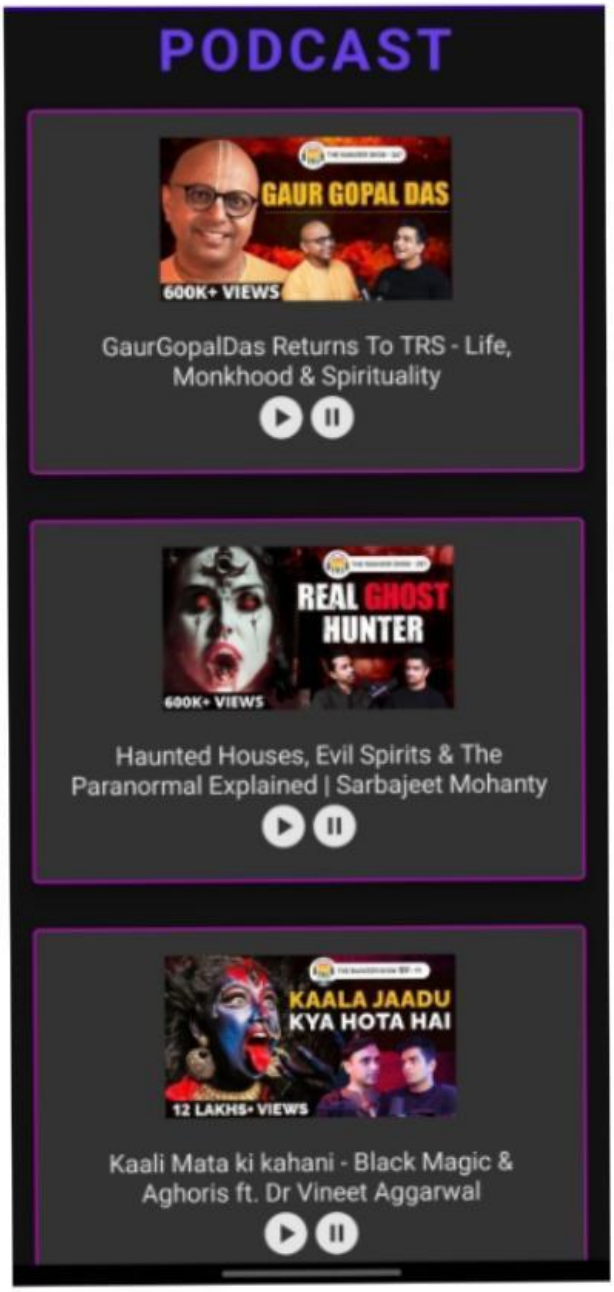


3.Result

Login Page

Register Page :

Main Podcast Page :



4. Advantages and Disadvantages

Advantages:

- ***Easy to access and use:*** ***Podcast apps are easy to download and use. Users can browse and listen to their favorite podcasts with just a few clicks.***

- ***Wide selection of content:*** ***There are thousands of podcasts***

available on different topics, providing users with a vast range of choices to choose from.

- ***Can listen anytime:*** ***Podcasts can be downloaded and listened to at any time, making it convenient for users to consume content while commuting, exercising, or doing other activities.***

- ***Free or affordable:*** ***Many podcast apps are free or***

have a minimal subscription fee, making it a cost-effective way to access high-quality content.

- ***Great for multitasking:*** ***Unlike reading or watching videos, podcasts are great for multitasking as they can be listened to while doing other activities like cooking, cleaning, or working out.***

Disadvantages:

- ***Internet connectivity required:*** ***Podcasts need an internet connection to be streamed or downloaded, which may be a limitation in areas with poor network coverage.***

- ***Limited visual content*****:** ***Since podcasts are primarily audio-based, they may not provide enough visual content to***

enhance the listening experience.

- ***Audio quality*: The audio quality of a podcast may vary depending on the quality of the recording equipment and the skills of the host or guest. Poor audio quality can make it hard to listen to or understand the content.**

- ***Overwhelming choice:* With so many podcasts available, it can be**

overwhelming to find the right one. It can be time-consuming to browse through the content and find the podcast that suits your interests.

- ***Hard to keep up:*** ***There are so many podcasts being released every day, making it difficult to keep up with all the new content. It can be a challenge to stay up to date with your favorite***

shows and discover new ones.

5.Application

Podcast app solutions can be applied to a variety of applications, including but not limited to:

Personal Development:

Podcast app solutions can be developed for personal development, including self-help, motivational, and inspirational podcasts.

News and Information:

Podcast app solutions can be developed for news and information, including daily news briefings, current events, and investigative journalism.

Entertainment:

Podcast app solutions can be developed for entertainment, including comedy, storytelling, and pop culture podcasts.

Education:

Podcast app solutions can be developed for educational purposes, including lectures, discussions, and interviews with experts in various fields.

Business:

Podcast app solutions can be developed for business purposes, including interviews with entrepreneurs, advice for

startups, and insights into industry trends.

Sports:

Podcast app solutions can be developed for sports fans, including analysis, commentary, and interviews with athletes and coaches.

Health and Fitness:

Podcast app solutions can be developed for health and fitness

enthusiasts, including nutrition advice, workout routines, and discussions about mental health.

Science and Technology:

Podcast app solutions can be developed for science and technology enthusiasts, including discussions about the latest breakthroughs and interviews with experts.

About Android Studio application:

Android Studio is the official integrated development environment (IDE) for building Android apps. It was first released by Google in 2013 and has since become the most popular development environment for Android app developers.

Android Studio is based on the IntelliJ IDEA community

edition, and it includes many tools and features designed specifically for developing Android apps. Some of the key features of Android Studio include:

User Interface (UI) Designer:

Android Studio includes a powerful UI designer that allows developers to easily create and modify app layouts using drag-and-drop tools. The UI designer supports a

variety of layouts, including linear, relative, and constraint layouts.

Code Editor:

Android Studio includes a powerful code editor that provides syntax highlighting, code completion, and other features to help developers write clean, efficient code. The code editor also supports debugging and refactoring tools.

Emulator:

Android Studio includes a built-in emulator that allows developers to test their apps on different Android devices without needing to own the actual devices. The emulator supports a wide range of Android versions and device configurations.

Gradle Build System:

Android Studio uses the Gradle build system, which makes it easy to

manage dependencies and build complex apps with multiple modules.

Version Control:

Android Studio supports version control systems like Git, allowing developers to easily manage their code changes and collaborate with other team members.

Performance Profiling:

Android Studio includes performance profiling tools that allow developers to identify performance bottlenecks in their apps and optimize their code for better performance.

Overall, Android Studio is a powerful tool for developing high-quality Android apps. It provides a range of features and tools that make it easy for developers

to build, test, and deploy their apps.

6. Conclusion

In conclusion, the podcast player app developed using Android Jetpack Compose provides a simple and user-friendly way for users to access and control their favorite podcasts. The app's intuitive interface allows users to easily browse through available podcasts,

select the episode they want to listen to, and control playback with the play and pause buttons. By leveraging the capabilities of Jetpack Compose, the project demonstrates how developers can create a modern and engaging user interface that is easy to use and navigate. Additionally, the app's functionality can be easily extended to include additional features, such as the ability to save favorite episodes or search for new podcasts. Overall,

the podcast player app is an excellent example of how Jetpack Compose can be used to create high-quality, user-friendly apps that provide a seamless experience for users

7. Future scope

The podcast industry is rapidly growing, and there are several potential areas for a podcast app to expand its scope in the future. Here are some possibilities:

Personalization:

Personalization is key for many apps, and podcast apps are no exception. In the future, a podcast app could use artificial intelligence (AI) algorithms to recommend new podcasts to users based on their listening history, preferences, and behavior.

Original Content:

While most podcast apps currently host existing

podcasts, there is an opportunity to create original content specifically for the app. This could include producing and publishing new podcasts, as well as creating new types of content, such as short-form audio clips or behind-the-scenes interviews with podcast hosts.

Social Integration:

As podcasts become more mainstream, there is an opportunity for

podcast apps to integrate social features. Users could follow and connect with their friends, share their favorite podcasts on social media, and discover new podcasts through their social networks.

Ad Innovation:

Podcasts are a popular medium for advertising, but current ad formats can be disruptive to the listening experience. In the future, a podcast app

could experiment with new ad formats that are less intrusive and more engaging, such as product placements or sponsored content.

Monetization:

As the podcast industry continues to grow, there are new opportunities for monetization. A podcast app could explore different revenue models, such as subscription-based access

to exclusive content or sponsorships from brands.

Localization:

While podcasts are popular globally, there is an opportunity for a podcast app to focus on local or regional content. This could include providing podcasts in different languages, as well as featuring podcasts that cover local news and events.

Education:

Podcasts are an excellent medium for educational content, and a podcast app could cater to this niche. In the future, a podcast app could provide curated collections of podcasts on different topics, as well as interactive features such as quizzes and challenges

8.Appendix

A. Source code

Creating the database classes

Step 1 : Create User data class

```
package com.example.podcastplayer

import androidx.room.ColumnInfo
import androidx.room.Entity
```

```
import
androidx.room.Primar
yKey


@Entity(tableName =
"user_table")
data class User(

@PrimaryKey(autoGen
erate = 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?,

    )
```

Step 2 : Create an UserDao interface

```
package
com.example.podcast
player

import
androidx.room.*

@Dao
interface UserDao {

    @Query("SELECT *
FROM user_table
WHERE email
= :email")
    suspend fun
getUserByEmail(email
: String): User?
```

```
    @Insert(onConflict
=
OnConflictStrategy.RE
PLACE)
    suspend fun
insertUser(user: User)

    @Update
    suspend fun
updateUser(user:
User)

    @Delete
    suspend fun
deleteUser(user:
User)
```

```
}
```

Step 3 : Create an UserDatabase class

```
package com.example.podcastplayer

import android.content.Context
import androidx.room.Database
import androidx.room.Room
```

```
import
androidx.room.RoomD
atabase

@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.applicationCo
ntext,

UserDatabase::class.ja
va,

"user_database"
                ).build()
                instance =
newInstance
                newInstance
            }
        }
    }
}
```

Step 4 : Create an UserDatabaseHelper class

```
package com.example.podcastplayer

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
```

```
import
android.database.Curs
or
import
android.database.sqlit
e.SQLiteDatabase
import
android.database.sqlit
e.SQLiteOpenHelper


class
UserDatabaseHelper(c
ontext: Context) :

SQLiteOpenHelper(con
text,
```

```
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_NA
ME TEXT, " +

"$COLUMN_LAST_NA
ME TEXT, " +
```

```
"$COLUMN_EMAIL
TEXT, " +

"$COLUMN_PASSWOR
D TEXT" +
                ")"


db?.execSQL(createTa
ble)
    }

    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_F
```

```
IRST_NAME,
user.firstName)

values.put(COLUMN_L
AST_NAME,
user.lastName)

values.put(COLUMN_E
MAIL, user.email)

values.put(COLUMN_P
ASSWORD,
user.password)

db.insert(TABLE_NAM
E, null, values)
        db.close()
```

```
    }


@SuppressLint("Rang
e")
    fun
getUserByUsername(u
sername: String):
User? {
        val db =
readableDatabase
        val cursor: Cursor
=
db.rawQuery("SELECT
* FROM $TABLE_NAME
WHERE
$COLUMN_FIRST_NA
```

```
ME = ?",
arrayOf(username))
        var user: User? =
null
        if
(cursor.moveToFirst()
) {
            user = User(
                id =
cursor.getInt(cursor.g
etColumnIndex(COLU
MN_ID)),
                firstName =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_FIRST_NAME)),
```

```
                lastName =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_LAST_NAME)),
                email =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_EMAIL)),
                password =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
```

```
    }

@SuppressLint("Rang
e")
    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.g
etColumnIndex(COLU
MN_ID)),
                firstName =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_FIRST_NAME)),
                lastName =
cursor.getString(curso
```

```
r.getColumnIndex(CO
LUMN_LAST_NAME)),
                email =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_EMAIL)),
                password =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }
```

```
@SuppressLint("Rang
e")
    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.g
etColumnIndex(COLU
MN_ID)),
                    firstName
=
cursor.getString(curso
r.getColumnIndex(CO
LUMN_FIRST_NAME)),
                    lastName
=
```

```
cursor.getString(curso
r.getColumnIndex(CO
LUMN_LAST_NAME)),
                email =
cursor.getString(curso
r.getColumnIndex(CO
LUMN_EMAIL)),
                password
=
cursor.getString(curso
r.getColumnIndex(CO
LUMN_PASSWORD)),
                )

users.add(user)
```

```
            } while
(cursor.moveToNext()
)
        }
        cursor.close()
        db.close()
        return users
    }

}
```

*Building application UI and connecting to database.

Step 1: Creating LoginActivity.kt with database

```
package com.example.podcastplayer

import android.content.Context
import android.content.Intent
import android.os.Bundle
```

```
import
androidx.activity.Com
ponentActivity
import
androidx.activity.comp
ose.setContent
import
androidx.compose.fou
ndation.BorderStroke
import
androidx.compose.fou
ndation.Image
import
androidx.compose.fou
ndation.background
```

```
import
androidx.compose.fou
ndation.layout.*
import
androidx.compose.fou
ndation.shape.Rounde
dCornerShape
import
androidx.compose.mat
erial.*
import
androidx.compose.mat
erial.icons.Icons
import
androidx.compose.mat
erial.icons.filled.Lock
```

```
import
androidx.compose.mat
erial.icons.filled.Perso
n
import
androidx.compose.run
time.*
import
androidx.compose.ui.A
lignment
import
androidx.compose.ui.
Modifier
import
androidx.compose.ui.g
raphics.Color
```

```
import
androidx.compose.ui.r
es.painterResource
import
androidx.compose.ui.t
ext.font.FontWeight
import
androidx.compose.ui.t
ext.input.PasswordVis
ualTransformation
import
androidx.compose.ui.t
ooling.preview.Previe
w
import
androidx.compose.ui.u
nit.dp
```

```
import
androidx.compose.ui.u
nit.em
import
androidx.compose.ui.u
nit.sp
import
androidx.core.content.
ContextCompat
import
com.example.podcast
player.ui.theme.Podca
stPlayerTheme

class LoginActivity :
ComponentActivity() {
```

```
    private lateinit var
databaseHelper:
UserDatabaseHelper
    override fun
onCreate(savedInstan
ceState: Bundle?) {

super.onCreate(saved
InstanceState)
        databaseHelper =
UserDatabaseHelper(t
his)
        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.d
p).fillMaxWidth()
```

```
    ) {

        Column(
            Modifier
                .background
(Color.Black)
                .fillMaxHeigh
t()
                .fillMaxWidt
h()
                .padding(bot
tom = 28.dp, start =
28.dp, end = 28.dp),

horizontalAlignment =
```

```
Alignment.CenterHoriz
ontally,

verticalArrangement =
Arrangement.Center
        )

        {

            Image(
                painter =
painterResource(R.dra
wable.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.text
FieldColors(

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
=
PasswordVisualTransf
ormation(),
```

```
                colors =
TextFieldDefaults.text
FieldColors(backgroun
dColor =
Color.Transparent)
            )

Spacer(modifier =
Modifier.height(12.dp)
)

            if
(error.isNotEmpty())
{
                Text(
                    text =
error,
```

```
                    color =
MaterialTheme.colors.
error,
                    modifier =
Modifier.padding(verti
cal = 16.dp)
                )
            }

            Button(
                onClick = {
                    if
(username.isNotEmpt
y() &&
password.isNotEmpty(
)) {
```

```
                    val user
=
databaseHelper.getUs
erByUsername(userna
me)
                    if
(user != null &&
user.password ==
password) {
                        error
= "Successfully log in"

context.startActivity(

Intent(

context,
```

```
MainActivity::class.jav
a
                        )
                    )

//onLoginSuccess()
                } else {
                    error
= "Invalid username
or password"
                }
            } else {
                error =
"Please fill all fields"
            }
        },
```

```
                border =
BorderStroke(1.dp,
Color(0xFF6a3ef9)),
                colors =
ButtonDefaults.button
Colors(backgroundCol
or = 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::c
lass.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)
}}

Step 2 : Creating RegistrationActivity.kt with database

package com.example.podcastplayer

```
import
android.content.Conte
xt
import
android.content.Intent
import
android.os.Bundle
import
androidx.activity.Com
ponentActivity
import
androidx.activity.comp
ose.setContent
import
androidx.compose.fou
ndation.BorderStroke
```

```
import
androidx.compose.fou
ndation.Image
import
androidx.compose.fou
ndation.background
import
androidx.compose.fou
ndation.layout.*
import
androidx.compose.mat
erial.*
import
androidx.compose.mat
erial.icons.Icons
```

```
import
androidx.compose.mat
erial.icons.filled.Email
import
androidx.compose.mat
erial.icons.filled.Lock
import
androidx.compose.mat
erial.icons.filled.Perso
n
import
androidx.compose.run
time.*
import
androidx.compose.ui.A
lignment
```

```
import
androidx.compose.ui.
Modifier
import
androidx.compose.ui.d
raw.alpha
import
androidx.compose.ui.g
raphics.Color
import
androidx.compose.ui.l
ayout.ContentScale
import
androidx.compose.ui.r
es.painterResource
```

```
import
androidx.compose.ui.t
ext.font.FontWeight
import
androidx.compose.ui.t
ext.input.PasswordVis
ualTransformation
import
androidx.compose.ui.t
ooling.preview.Previe
w
import
androidx.compose.ui.u
nit.dp
import
androidx.compose.ui.u
nit.em
```

```
import
androidx.compose.ui.u
nit.sp
import
androidx.core.content.
ContextCompat
import
com.example.podcast
player.ui.theme.Podca
stPlayerTheme

class
RegistrationActivity :
ComponentActivity()
{ private lateinit var
databaseHelper:
UserDatabaseHelper
```

```
    override fun
onCreate(savedInstan
ceState: Bundle?) {

super.onCreate(saved
InstanceState)
        databaseHelper =
UserDatabaseHelper(t
his)
        setContent {

PodcastPlayerTheme {
                // A surface
container using the
'background' color
from the theme
                Surface(
```

```
                    modifier =
Modifier.fillMaxSize(),
                    color =
MaterialTheme.colors.
background
                ) {

RegistrationScreen(thi
s,databaseHelper)
                }
            }
        }
    }
}

@Composable
```

```
fun
RegistrationScreen(co
ntext: 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(C
olor.Black)
            .fillMaxHeight(
)
            .fillMaxWidth()
,
```

```
horizontalAlignment =
Alignment.CenterHoriz
ontally,

verticalArrangement =
Arrangement.Center
    )

    {
        Row {
            Text(
                text = "Sign
Up",
                color =
Color(0xFF6a3ef9),
```

```
                fontWeight
= FontWeight.Bold,
                fontSize =
24.sp, style =
MaterialTheme.typogr
aphy.h1,

letterSpacing =
0.1.em
            )
        }

        Image(
            painter =
painterResource(id =
R.drawable.podcast_si
gnup),
```

```
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.text
FieldColors(

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
=
PasswordVisualTransf
ormation(),
            colors =
TextFieldDefaults.text
FieldColors(backgroun
dColor =
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.text
FieldColors(backgroun
dColor =
Color.Transparent)
        )
```

```
        Spacer(modifier
=
Modifier.height(8.dp))

        if
(error.isNotEmpty())
{
            Text(
                text = error,
                color =
MaterialTheme.colors.
error,
                modifier =
Modifier.padding(verti
cal = 16.dp)
            )
        }
```

```
Button(
    onClick = {
        if
(username.isNotEmpt
y() &&
password.isNotEmpty(
) &&
email.isNotEmpty()) {
            val user =
User(
                id =
null,

firstName =
username,
```

```
lastName = null,
                    email =
email,

password = password
                )

databaseHelper.insert
User(user)
                error =
"User registered
successfully"
                // Start
LoginActivity using
the current context
```

```
context.startActivity(
                    Intent(

context,

LoginActivity::class.ja
va
                        )
                    )

                } else {
                    error =
"Please fill all fields"
                }
            },
```

```
            border =
BorderStroke(1.dp,
Color(0xFF6a3ef9)),
            colors =
ButtonDefaults.button
Colors(backgroundCol
or = Color.Black),
            modifier =
Modifier.padding(top
= 16.dp)
        ) {
            Text(text =
"Register",
                fontWeight
= FontWeight.Bold,
                color =
Color(0xFF6a3ef9)
```

```
        )
    }


    Row(
        modifier =
Modifier.padding(30.d
p),

verticalAlignment =
Alignment.CenterVerti
cally,

horizontalArrangemen
t =
Arrangement.Center
    ) {
```

```
            Text(text =
"Have an account?",
color = Color.White)


TextButton(onClick =
{

context.startActivity(
                Intent(
                    context,

LoginActivity::class.ja
va
                )
                )
        })
```

```
            {
            Text(text =
"Log in",
                fontWeight
= FontWeight.Bold,
                style =
MaterialTheme.typogr
aphy.subtitle1,
                color =
Color(0xFF6a3ef9)
            )
        }

    }
    }
}
```

```
private fun
startLoginActivity(con
text: Context) {
    val intent =
Intent(context,
LoginActivity::class.ja
va)

ContextCompat.startA
ctivity(context, intent,
null)
}
```

Step 3 : Creating MainActivity.kt file

```
package
com.example.podcast
player

import
android.content.Conte
xt
import
android.media.MediaPl
ayer
import
android.os.Bundle
import
androidx.activity.Com
ponentActivity
```

```
import
androidx.activity.comp
ose.setContent
import
androidx.compose.fou
ndation.BorderStroke
import
androidx.compose.fou
ndation.Image
import
androidx.compose.fou
ndation.layout.*
import
androidx.compose.fou
ndation.rememberScr
ollState
```

```
import
androidx.compose.fou
ndation.verticalScroll
import
androidx.compose.mat
erial.*
import
androidx.compose.run
time.*
import
androidx.compose.ui.A
lignment
import
androidx.compose.ui.
Modifier
```

```
import
androidx.compose.ui.g
raphics.Color
import
androidx.compose.ui.r
es.painterResource
import
androidx.compose.ui.t
ext.font.FontWeight
import
androidx.compose.ui.t
ext.style.TextAlign
import
androidx.compose.ui.u
nit.dp
```

```
import
androidx.compose.ui.u
nit.em
import
androidx.compose.ui.u
nit.sp
import
com.example.podcast
player.ui.theme.Podca
stPlayerTheme

class MainActivity :
ComponentActivity() {
    override fun
onCreate(savedInstan
ceState: Bundle?) {
```

```
super.onCreate(saved
InstanceState)
            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(horizontalAlig
nment =
Alignment.CenterHoriz
ontally,
verticalArrangement =
Arrangement.Center)
{
```

```
            Text(text =
"PODCAST",
                modifier =
Modifier.fillMaxWidth(
),
                textAlign =
TextAlign.Center,
                color =
Color(0xFF6a3ef9),
                fontWeight
= FontWeight.Bold,
                fontSize =
36.sp,
                style =
MaterialTheme.typogr
aphy.h1,
```

```
letterSpacing =
0.1.em

            )
        }

        Column(modifier
= Modifier
            .fillMaxSize()
            .verticalScroll(r
ememberScrollState()
)) {


            Card(
```

```
                elevation =
12.dp,
                border =
BorderStroke(1.dp,
Color.Magenta),
                modifier =
Modifier
                    .padding(
16.dp)
                    .fillMaxWi
dth()
                    .height(25
0.dp)
            )
            {
                val mp:
MediaPlayer =
```

```
MediaPlayer.create(co
ntext, R.raw.audio)

            Column(
                modifier =
Modifier.fillMaxSize(),

horizontalAlignment =
Alignment.CenterHoriz
ontally
                ) {

                Image(
                    painter
= painterResource(id
= R.drawable.img),
```

```
contentDescription =
null,

modifier = Modifier
                    .heig
ht(150.dp)
                    .widt
h(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)
                    .fillMaxWi
dth()
                    .height(25
0.dp)
            )
```

```
                {
                    val mp:
MediaPlayer =
MediaPlayer.create(co
ntext, R.raw.audio_1)

                    Column(
                        modifier =
Modifier.fillMaxSize(),

horizontalAlignment =
Alignment.CenterHoriz
ontally

                    ) {

                        Image(
```

```
                painter
= painterResource(id
= R.drawable.img_1),

contentDescription =
null,

modifier = Modifier
                .heig
ht(150.dp)
                .widt
h(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)
                    .fillMaxWi
dth()
```

```
                    .height(25
0.dp)
                )
                {
                    val mp:
MediaPlayer =
MediaPlayer.create(co
ntext, R.raw.audio_2)

                    Column(
                        modifier =
Modifier.fillMaxSize(),

horizontalAlignment =
Alignment.CenterHoriz
ontally
```

```
) {

                Image(
                    painter
= painterResource(id
= R.drawable.img_2),

contentDescription =
null,

modifier = Modifier
                        .heig
ht(150.dp)
                        .widt
h(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)
```

```
                .fillMaxWi
dth()
                .height(25
0.dp)
            )
            {
                val mp:
MediaPlayer =
MediaPlayer.create(co
ntext, 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)
                .fillMaxWi
dth()
                .height(25
0.dp)
            )
            {
                val mp:
MediaPlayer =
MediaPlayer.create(co
ntext, R.raw.audio_4)

                Column(
                    modifier =
Modifier.fillMaxSize(),
```

```
horizontalAlignment =
Alignment.CenterHoriz
ontally
                ) {

                    Image(
                        painter
= painterResource(id
= R.drawable.img_4),

contentDescription =
null,

modifier = Modifier
                            .heig
ht(150.dp)
```

```
                .widt
h(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)
                    .fillMaxWi
dth()
                    .height(25
0.dp)
            )
            {
                val mp:
MediaPlayer =
MediaPlayer.create(co
ntext, R.raw.audio_5)
```

```
                Column(
                    modifier =
Modifier.fillMaxSize(),

horizontalAlignment =
Alignment.CenterHoriz
ontally
                ) {

                    Image(
                        painter
= painterResource(id
= R.drawable.img_5),

contentDescription =
null,
```

```
modifier = Modifier
                    .heig
ht(150.dp)
                    .widt
h(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 =
""
                    )
                }

            }
        }

    }

  }
}
```

```
}
```