PROJECT REPORT

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

1. INTRODUCTION

1.1 Overview

A project that demonstratesthe use of Android Jetpack Compose to build a UI for a podcast player app. The app allows users

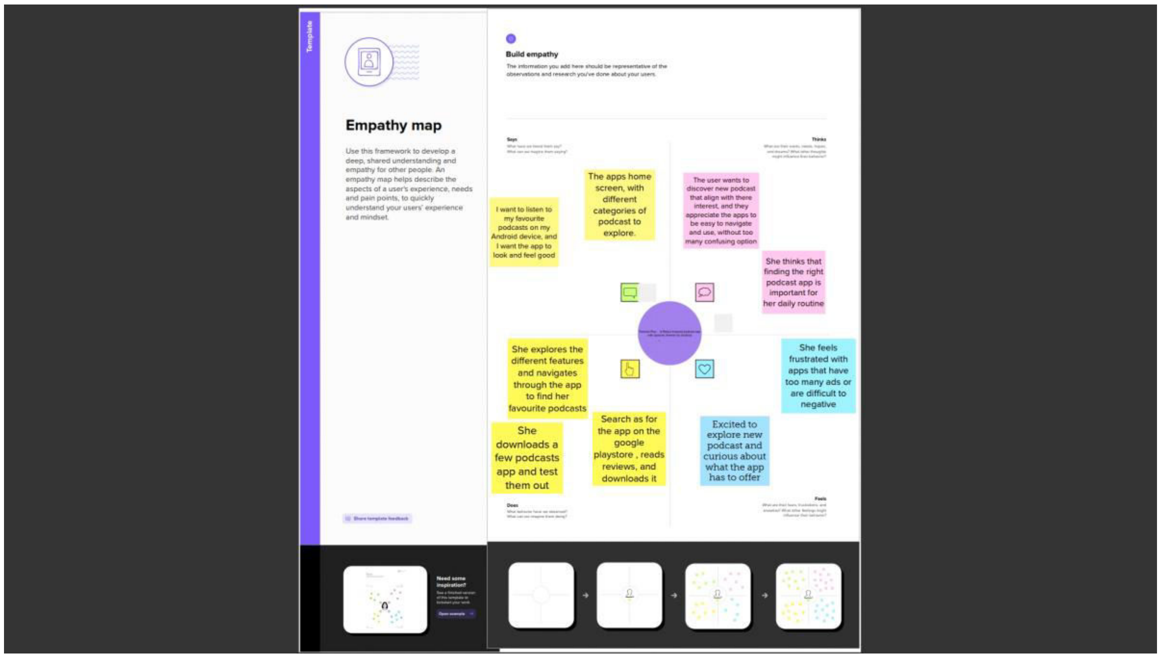
to choose , play and pause podcasts.

* 1.2 Purpose

Podcasts , Which can include audio,video,PDF,and ePub files, are subscribed to and downloaded through web syndication or streamed online to a computer or mobile device. Subscribers are then able to view, listen to, and transfer the episodes to a variety off media players,or podcatchers.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map



Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

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

I want to listen to my favourite podcasts on my Android device, and I want the app to look and feel good

The apps home screen, with different categories of podcast to explore.

The user wants to discover new podcast that align with there interest, and they appreciate the apps to be easy to navigate and use, without too many confusing option

She thinks that finding the right podcast app is important for her daily routine

She explores the different features and navigates through the app to find her favourite podcasts

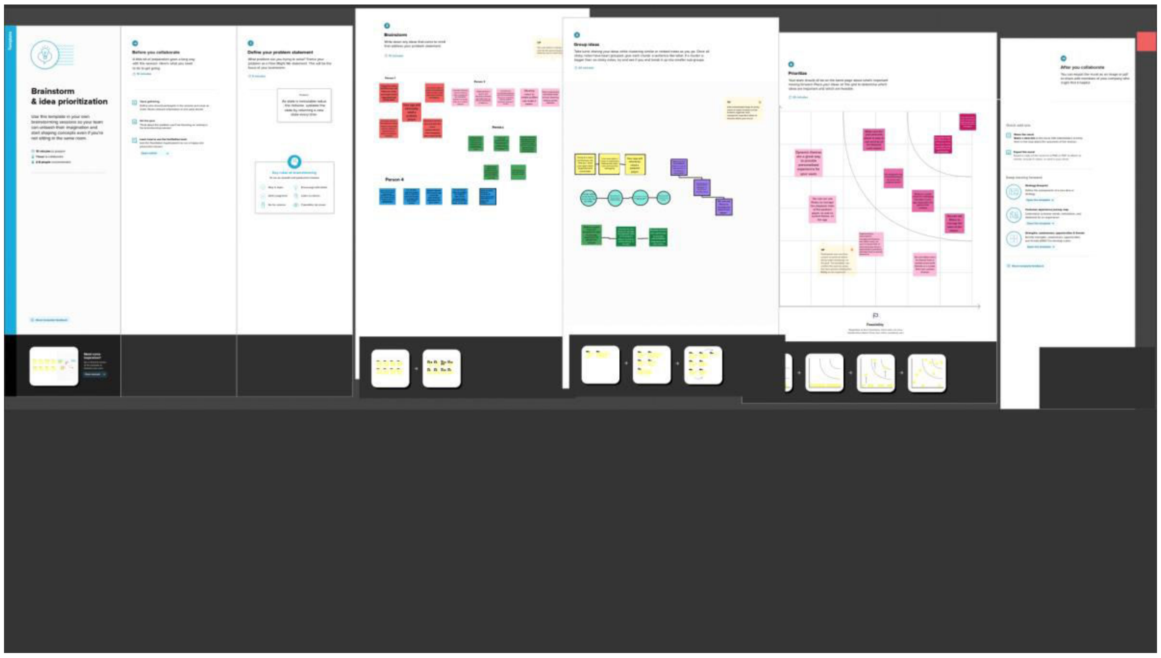
She downloads a few podcasts app and test them out

Search as for the app on the google playstore , reads reviews, and downloads it

Excited to explore new podcast and curious about what the app has to offer

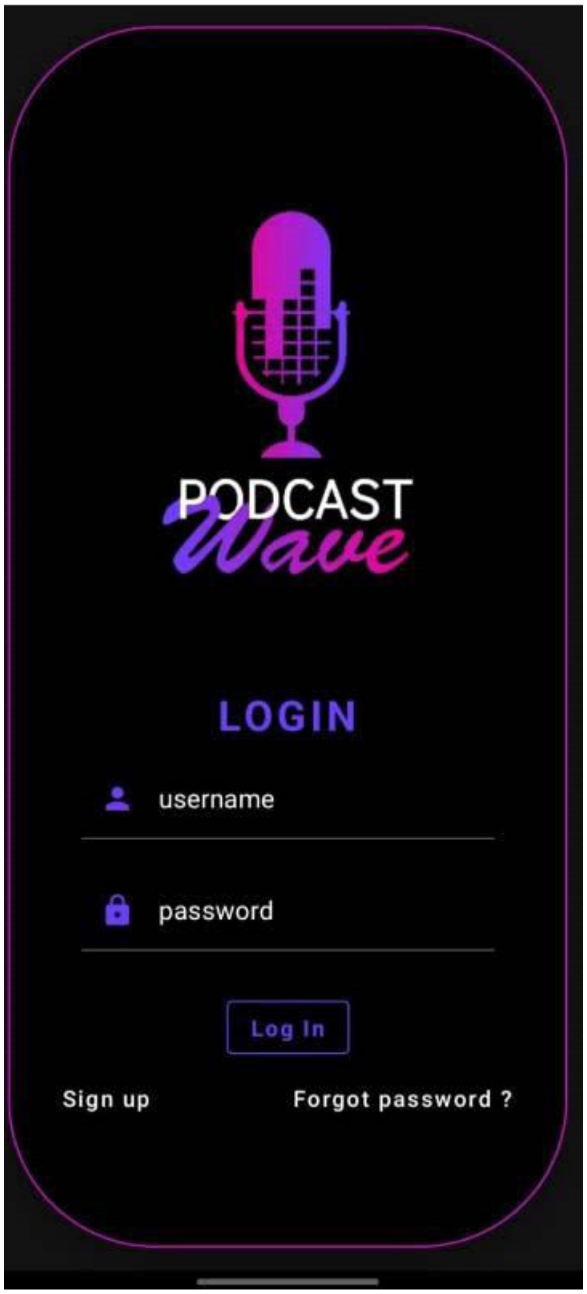
She feels frustrated with apps that have too many ads or are difficult to negative

2.2 Ideation & Brainstorming Map

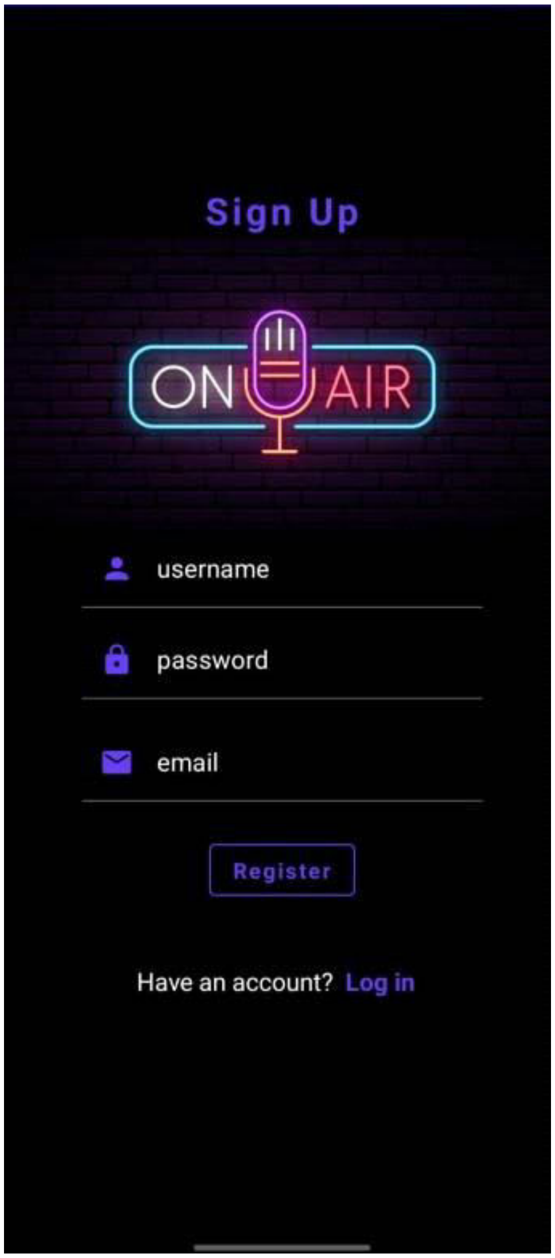


3. RESULT

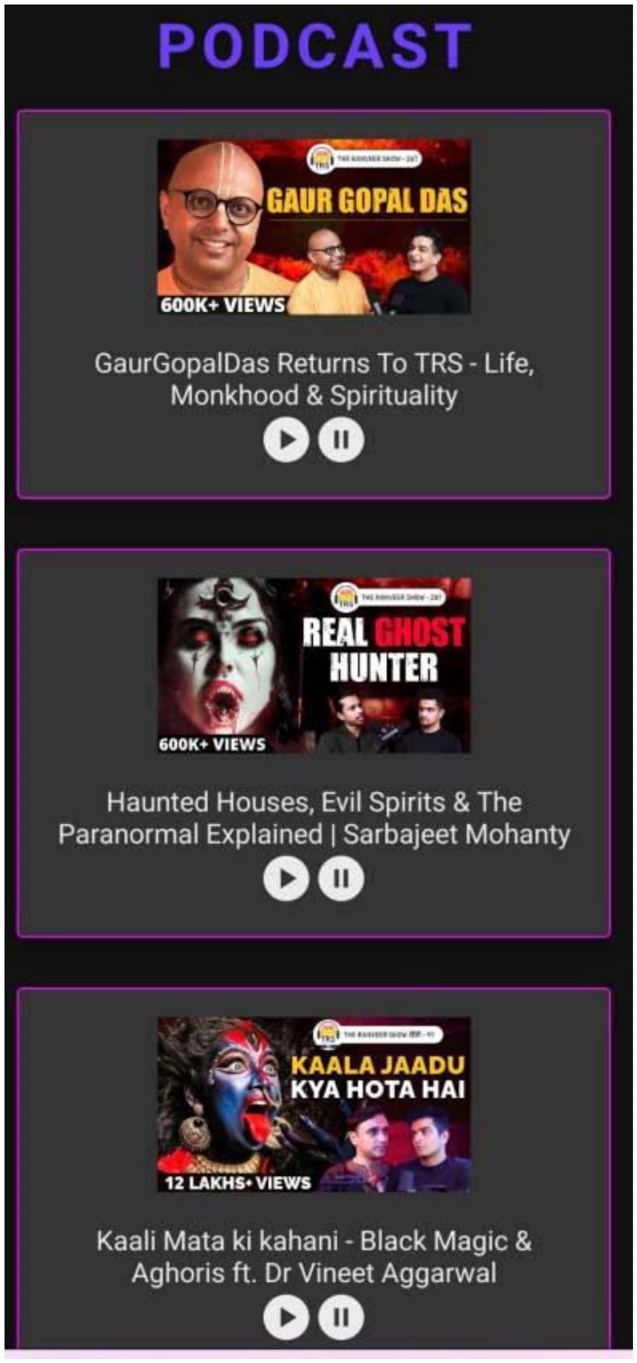
Login Page



Register Page


Sign Up
ON AIR
username
password
email
Register
Have an account? Log in

Main News Headlines Page



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

4. ADVANTAGES & DISADVANTAGES

- **Advantages**
  - Convenience
  - Easy
  - **Finding and audience**
- **Disadvantages**
  - Accessbility
  - Loose of control

5. APPLICATIONS

The Project provides only the latest news headlines. The App can be used as personal app for individuals. The App will increase the General knowledge of the users and keep them updated at the time. The project will be useful to all the sections of the society.

6. CONCLUSION

Podcasts have become increasingly popular in recent years, with millions of people tuning in to their favorite shows each week. With this surge in popularity, there are now many different podcast apps available to choose from. Some of the most popular options include Apple Podcasts, Spotify, Google Podcasts, Overcast, Pocket Casts, and Stitcher.

7. FUTURE SCOPE

The present App can be enhanced by adding more features. User may be allowed to customize the app based on their requirements. More categories can be included the future edition of the App.

8. APPENDIX

```
// 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

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)

}ao.kt



// 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

        }



    }

    Footer

    © 2023 GitHub, Inc.

    Footer navigation

    Terms

    Privacy

    Security

    Status

    Docs


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)
    }}
```

// RegisterActivity.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)

}
```

// 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 = ""
                            )
                        }


                    }
                }


            }
        }
    }
}
```

// 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>
```