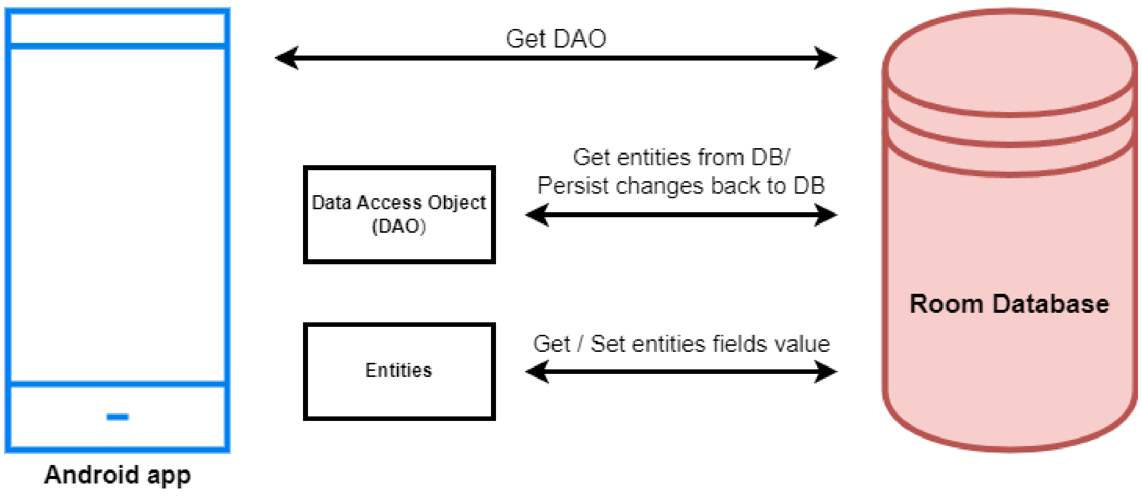
INTRODUCTION

1.1 OVERVIEW

A **podcast** is an audio programme, just like talk radio. But you subscribe to it on your smartphone and listen to it whenever you like. A podcast is a series of spoken word, audio episodes all focused on a particular topic or theme, like cycling or start ups.you can subscribe to the show with an app on your phone and listen to episodes whenever you like on your headphones in the car or through speakers. You can use it tell users anything to want, which makes it a powerful tool to convince people to listen.

ARCHITECTURE

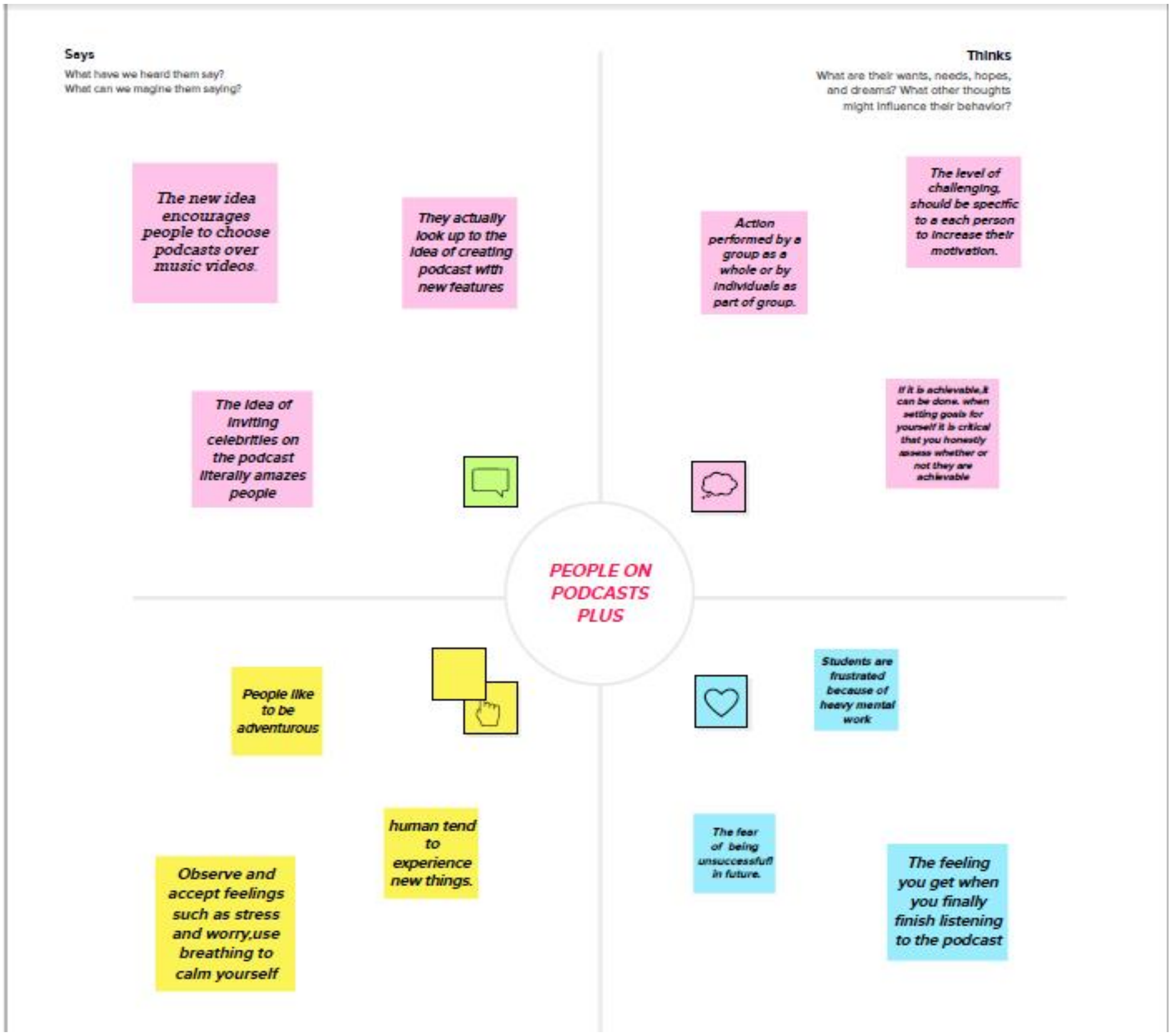

Get DAO
Data Access Object (DAO)
Get entities from DB/ Persist changes back to DB
Entities
Get / Set entities fields value
Room Database
Android app

PURPOSE

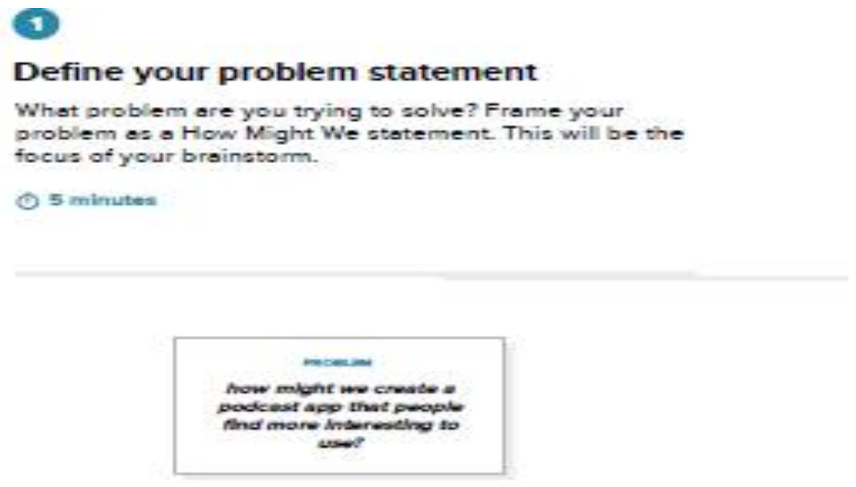
Podcasts, which can include audio, video,PDF and epub files are subscribed to and download through web syndication or streamed online to a computer or mobile device. Its primarily about the ease of consuming information. users particularly appreciate the wide variety of shows and their accessibility. Subscribers are then able to view, listen to, and transfer the episodes to a variety of media players, or podcatches.

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP


Says
What have we heard them say?
What can we magine them saying?
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
The new idea encourages people to choose podcasts over music videos.
They actually look up to the idea of creating podcast with new features
The idea of inviting celebrities on the podcast literally amazes people
Action performed by a group as a whole or by individuals as part of group.
The level of challenging, should be specific to a each person to increase their motivation.
If it is achievable,it can be done. when setting goals for yourself it is critical that you honestly assess whether or not they are achievable
PEOPLE ON PODCASTS PLUS
People like to be adventurous
human tend to experience new things.
Observe and accept feelings such as stress and worry,use breathing to calm yourself
Students are frustrated because of heavy mental work
The fear of being unsuccessfull in future.
The feeling you get when you finally finish listening to the podcast

2.2 IDEATION AND BRAINSTORMING MAP



1

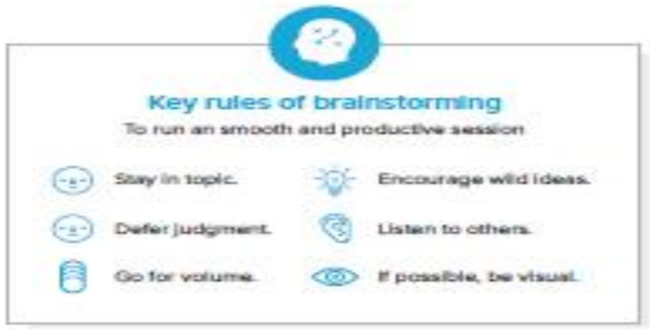
Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

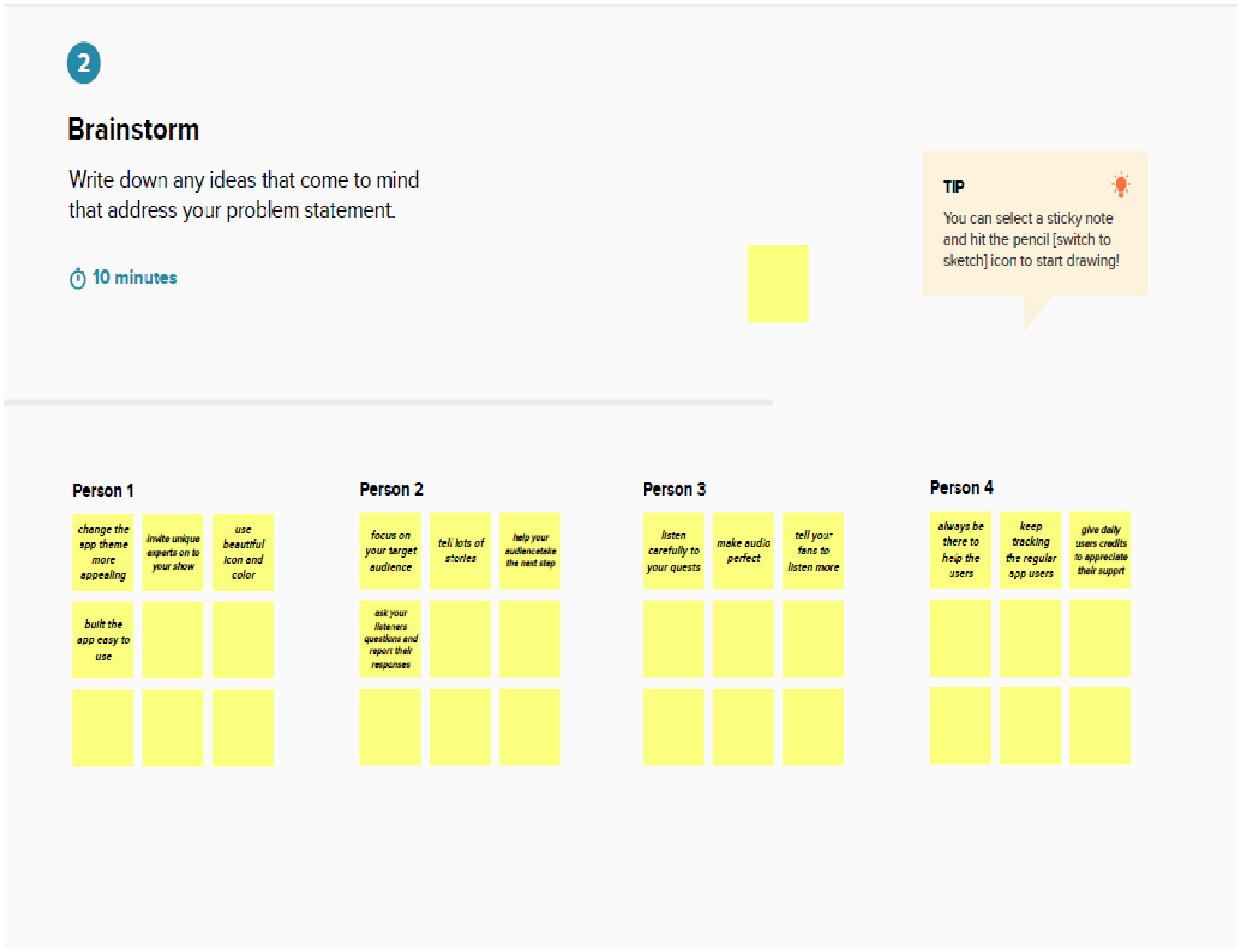
how might we create a podcast app that people find more interesting to use?



Key rules of brainstorming

To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1

- change the app theme more appealing
- invite unique experts on to your show
- use beautiful icon and color
- built the app easy to use

Person 2

- focus on your target audience
- tell lots of stories
- help your audiencetake the next step
- ask your listeners questions and report their responses

Person 3

- listen carefully to your quests
- make audio perfect
- tell your fans to listen more

Person 4

- always be there to help the users
- keep tracking the regular app users
- give daily users credits to appreciate their supprt

3

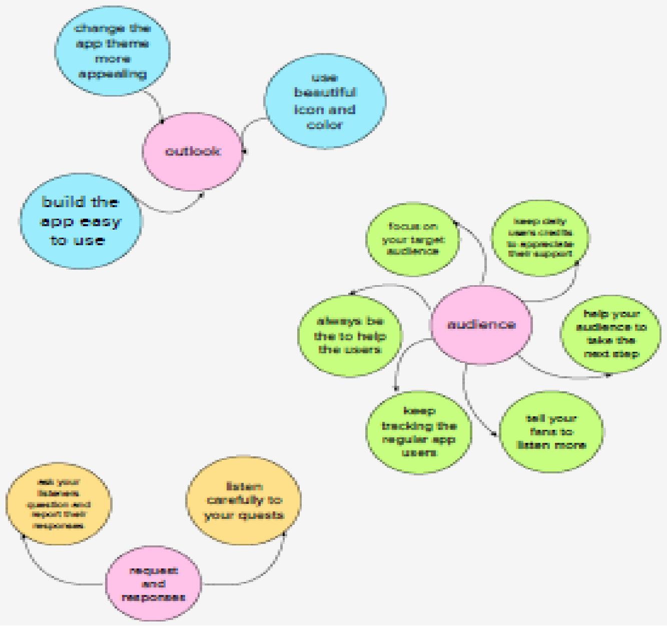
Group Ideas

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

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

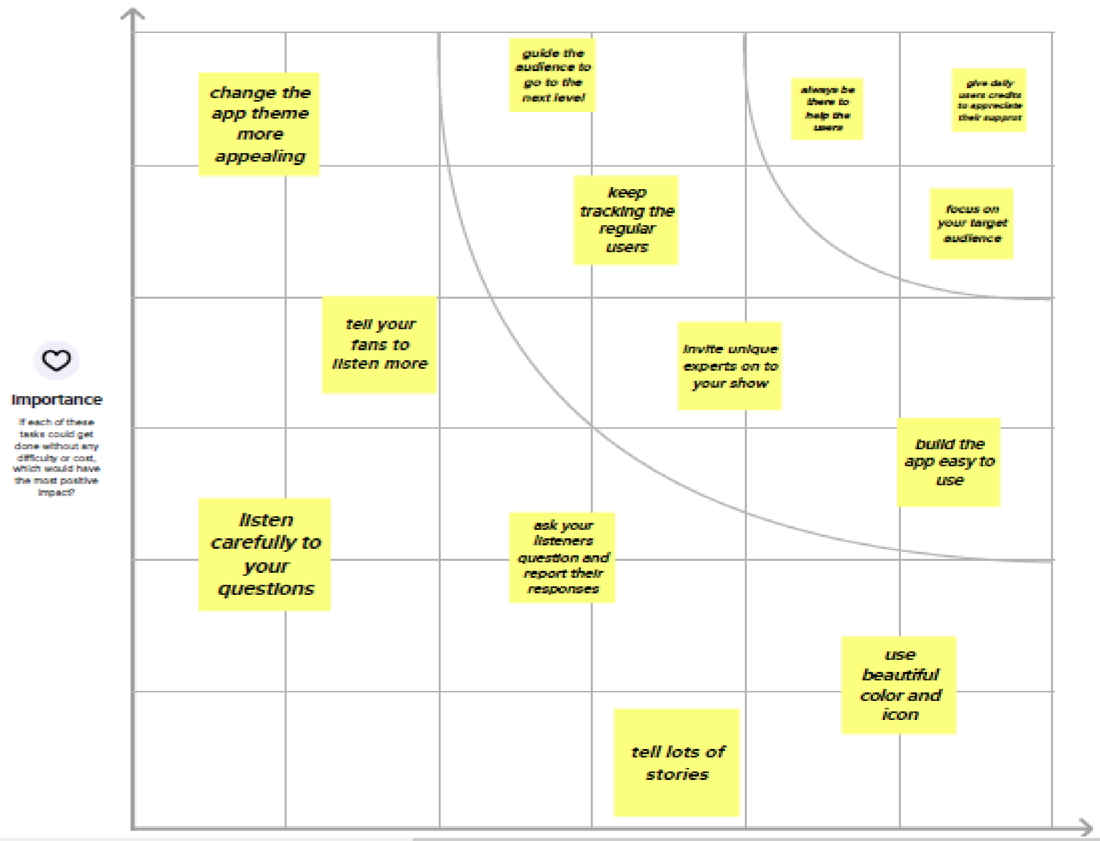

change the app theme more appealing
use beautiful icon and color
outlook
build the app easy to use
focus on your target audience
keep daily users credits to appreciate their support
always be the to help the users
audience
help your audience to take the next step
keep tracking the regular app users
tell your fans to listen more
ask your listeners question and report their responses
listen carefully to your guests
request and responses

4

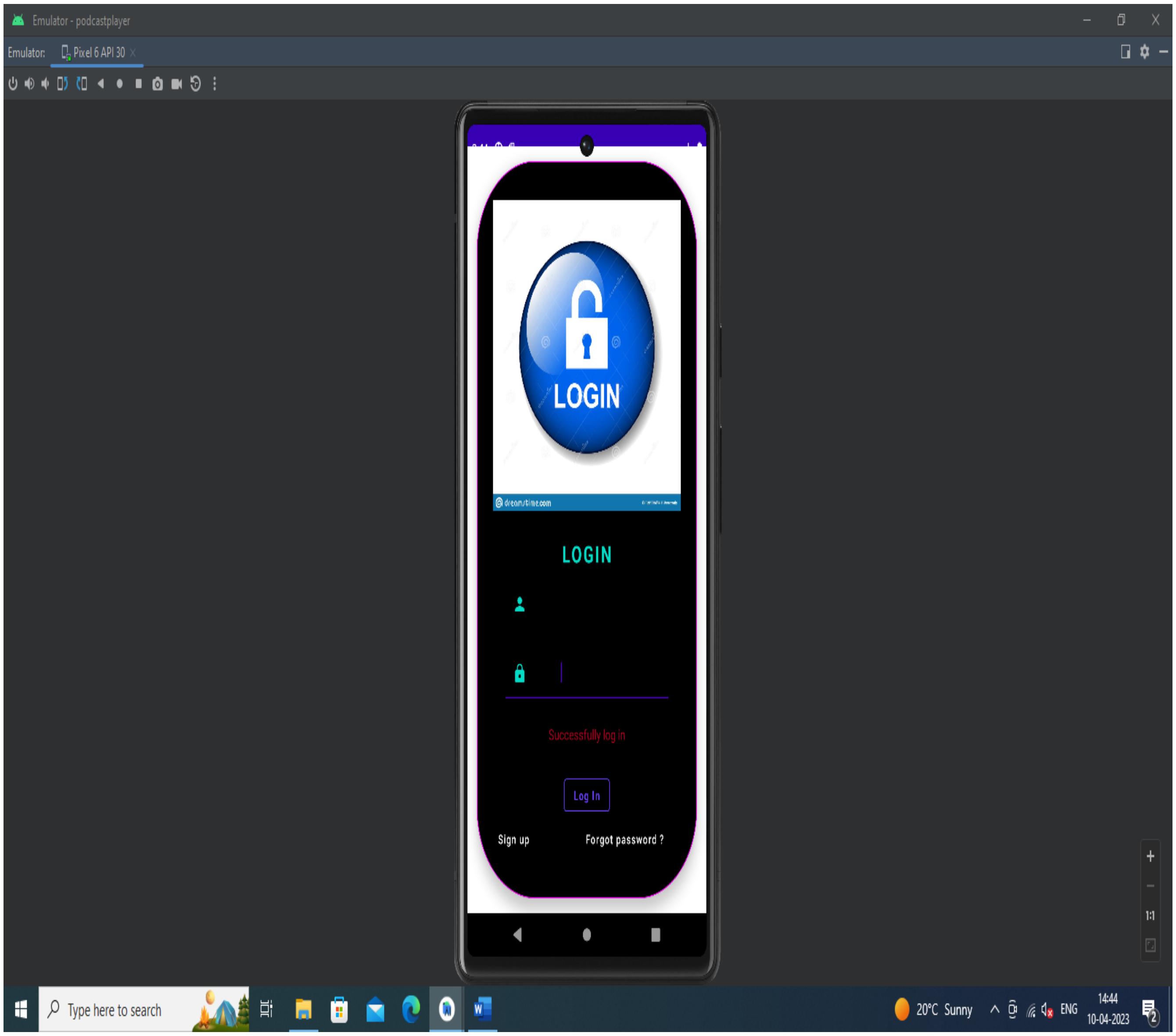
Prioritize

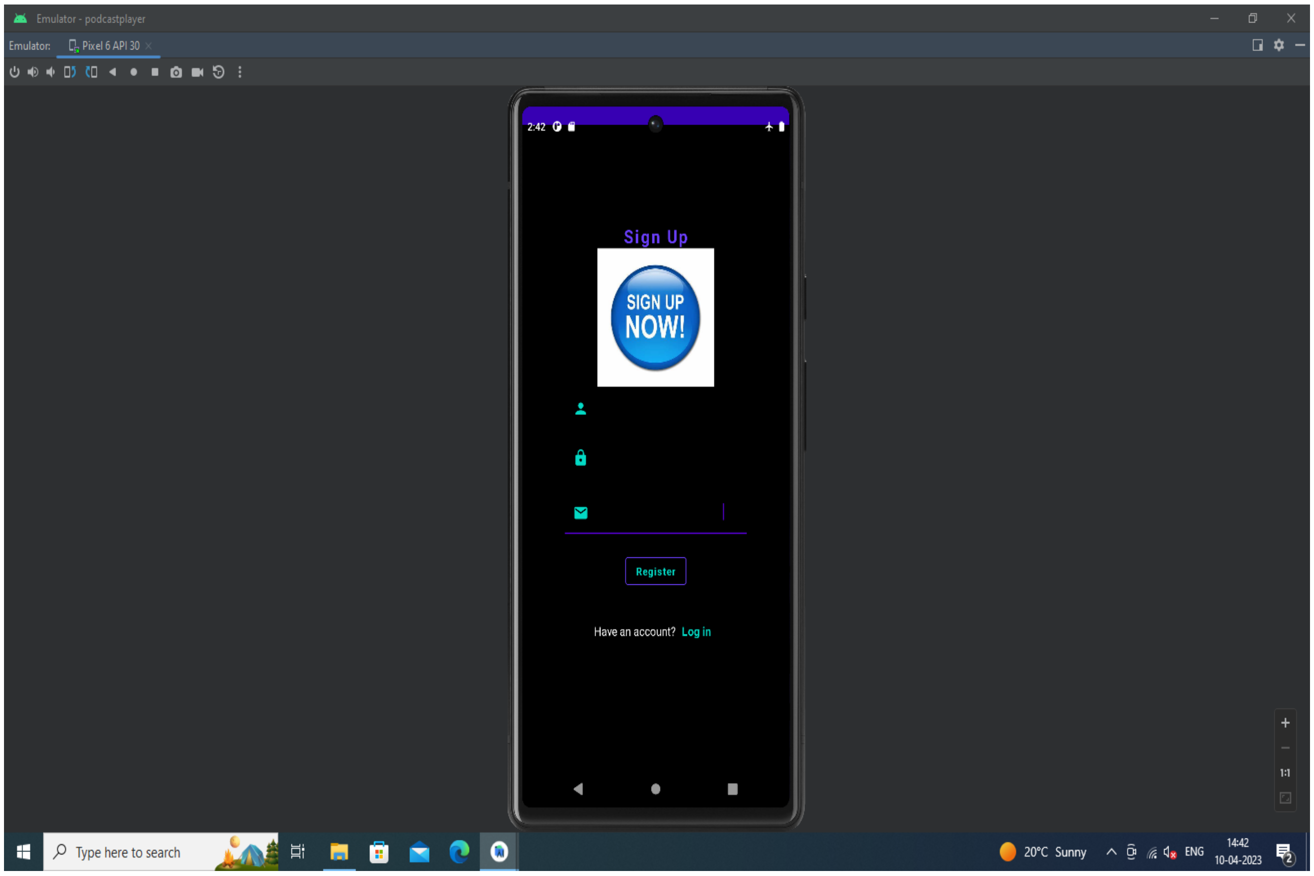
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

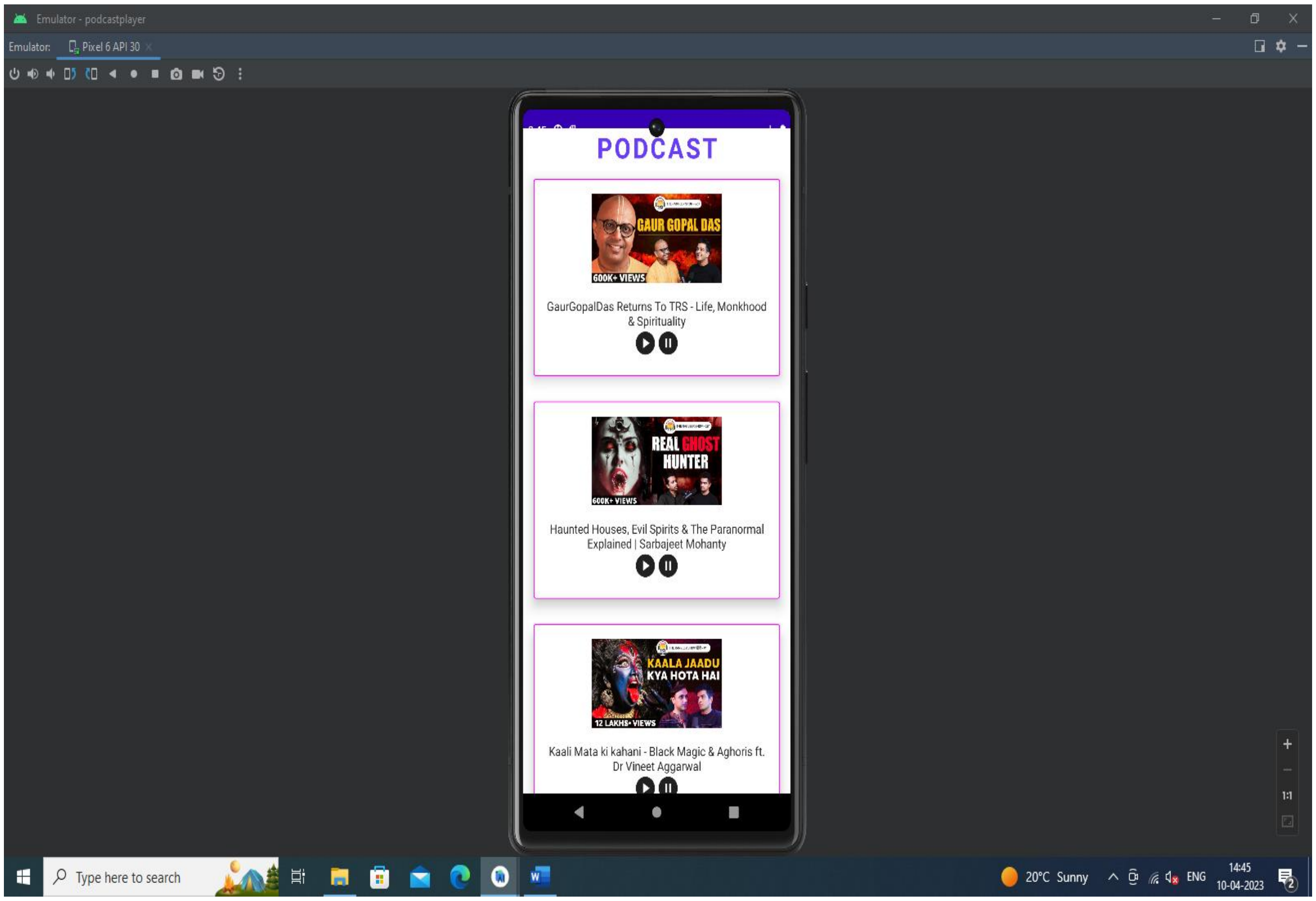
20 minutes


Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
change the app theme more appealing
guide the audience to go to the next level
always be there to help the users
give daily users credits to appreciate their support
keep tracking the regular users
focus on your target audience
tell your fans to listen more
invite unique experts on to your show
build the app easy to use
listen carefully to your questions
ask your listeners question and report their responses
use beautiful color and icon
tell lots of stories

RESULT


Emulator - podcastplayer
Emulator: Pixel 6 API 30
LOGIN
dreamstime.com
LOGIN
Successfully log in
Log In
Sign up
Forgot password ?
Type here to search
20°C Sunny
ENG
14:44
10-04-2023


Emulator - podcastplayer
Emulator: Pixel 6 API 30
Sign Up
SIGN UP NOW!
Register
Have an account? Log in
Type here to search
20°C Sunny
ENG
14:42
10-04-2023


Emulator - podcastplayer

Emulator: Pixel 6 API 30

PODCAST

GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality

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

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

ADVANTAGES AND DISADVANTAGES:

ADVANTAGES

Podcasting is a very convenient medium of communication, especially for the audience. You can listen to what you want and whenever you want. All you need is a smartphone and a podcast listening app. Podcast is an on-demand technology Podcast listeners do not require any special medium to listen to their favorite podcasts. Almost everyone has a smartphone these days and that's all it is required to listen to a podcast. Since the smartphone is mobile in nature people can listen to podcasts practically anywhere. Podcasting is not a very expensive medium for advertisers or podcasters. The setup and making costs are fairly less. The ability of podcasts to reach a large audience at low costs makes it and effective medium of communication

- Convenience
- No restriction on time
- Personalized content
- Podcasts are portable
- Podcasts have Direct connected with audience
- Podcasts cut cost

DISADVANTAGES

Internet is required for people to access the podcasts and it becomes difficult to reach to a wider audience if internet is not available. Larger podcasts with high file sizes and video podcasts become a major issue. There is still a large population in developing and underdeveloped countries which d not have access to the internet. This can become a barrier for reaching your desired audience. With millions of podcasts on thousands of topics out there, it is very difficult to rank on Google podcasts or iTunes for a particular topic.

- ➢ IP and content protection is difficult
- ➢ It requires preparation
- ➢ Your guest's equipment must be good for remote podcasting
- ➢ The guest can make or break your podcast.

APPLICATION

They can be used to convey instructional information from the teacher or trainer, motivational stories, and auditory case studies. Podcasts can also be used by the learners as artifacts and evidence of learning; for example, a student might prepare a brief podcast as a summary of a concept in lieu of writing an essay.

- Media
- personal
- schools

CONCLUSION

Podcasting app**s** make it easy to navigate your podcast library, download new episodes, bookmark shows, and browse your listening history**.** Never miss a beat using a podcast app, subscribe, and get notified when a new episode is published. Podcasts have a much better method of reaching your audience because they are more versatile than trying to listen to a video on a smartphone while driving**.** Podcast Users are More Engaged: Podcasts are better for engagement because people can listen at their pace**.** the podcast app project is a mobile application that provides users with a convenient and easy way to browse, search, and listen to podcasts. This project utilizes various components and functionalities to create an intuitive user interface that enhances the user experience. One of the key features of the podcast app project is its ability to fetch podcast data from RSS feeds. This allows users to discover new podcasts and stay up-to-date with their favorite shows. Additionally, the app's audio player provides a seamless listening experience, making it easy for users to enjoy their favorite podcasts on the go.

FUTURE SCOPE

- Raised content standards
- Podcasts as marketing tools
- The land grab of audio content industry
- Influencer host & voice search

APPENDIX

A.SOURCE CODE

GRADLE SCRIPTS

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.podcastplayer'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.podcastplayer"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
}
```

CREATING DATABASE CLASSES

USER DATA CLASS

```
plugins {

    id 'com.android.application'

    id 'org.jetbrains.kotlin.android'

}



android {

    namespace 'com.example.travelapp'

    compileSdk 33



    defaultConfig {

        applicationId "com.example.travelapp"
        minSdk 21

        targetSdk 33

        versionCode 1

        versionName "1.0"


        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

        vectorDrawables {

            useSupportLibrary true

        }

    }


    buildTypes {

        release {

            minifyEnabled false

            proguardFiles getDefaultProguardFile('proguard-android-
optimize.txt'), 'proguard-rules.pro'

        }

    }

    compileOptions {

        sourceCompatibility JavaVersion.VERSION_1_8

                targetCompatibility JavaVersion.VERSION_1_8

    }

    kotlinOptions {

        jvmTarget = '1.8'

    }

    buildFeatures {

        compose true

    }

    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'

    }

    packagingOptions {

        resources {

            excludes += '/META-INF/{AL2.0,LGPL2.1}'

        }

    }

}


dependencies {


    implementation 'androidx.core:core-ktx:1.7.0'

    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'

    implementation 'androidx.activity:activity-compose:1.3.1'

    implementation "androidx.compose.ui:ui:$compose_ui_version"

    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"

    implementation 'androidx.compose.material:material:1.2.0'

    testImplementation 'junit:junit:4.13.2'

    androidTestImplementation 'androidx.test.ext:junit:1.1.5'

    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

    androidTestImplementation "androidx.compose.ui:ui-test-
junit4:$compose_ui_version"

    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"

    debugImplementation "androidx.compose.ui:ui-test-
manifest:$compose_ui_version"

    // Adding Room dependencies

    implementation 'androidx.room:room-common:2.5.0'

    implementation 'androidx.room:room-ktx:2.5.0'
```

```
}
```

USER DAO INTERFACE

```
package com.example.podcastplayer

import androidx.room.*

@Dao
interface UserDao {

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

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

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

USER DATABASE CLASS

```
package com.example.podcastplayer

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

import androidx.room.Room

import androidx.room.RoomDatabase


@Database(entities = [User::class], version = 1)

abstract class UserDatabase : RoomDatabase() {


    abstract fun userDao(): UserDao


    companion object {


        @Volatile

        private var instance: UserDatabase? = null


        fun getDatabase(context: Context): UserDatabase {

            return instance ?: synchronized(this) {

                val newInstance = Room.databaseBuilder(

                    context.applicationContext,

                    UserDatabase::class.java,

                    "user_database"

                ).build()

                instance = newInstance

                newInstance

            }

        }

    }

}
```

USER DATABASE HELPER CLASS

```
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

    }


}
```

LOGIN ACTIVITY

```
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(0xFF03DAC5),

                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(0xFF03DAC5)

                    )

                },
                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(0xFF03DAC5)
                    )
                },
                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)
    }}
```

REGISTRATION ACTIVITY

```
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(0xFF03DAC5)
                )
            },
            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(0xFF03DAC5)
                )
            },
            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(0xFF03DAC5)
                )
            },
            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(0xFF03DAC5)
    )
}



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(0xFF03DAC5)
        )
```

```
            }


        }

    }

}

private fun startLoginActivity(context: Context) {

    val intent = Intent(context, LoginActivity::class.java)

    ContextCompat.startActivity(context, intent, null)

}
```

MAIN ACTIVITY

```
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.app_src_main_res_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.app_src_main_res_raw_audio_2)


                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 = ""

                            )

                        }


                    }

                }


            }
            }

        }

}
```

ANDROID MANIFEST

```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>
```