**Project Report Template**

1. **INTRODUCTION**
2. **Overview :-**

A podcast is essentially an episodic series of digital audio files that can be downloaded or live streamed and listened to. Anyone and everyone can create a podcast these days. Several podcasts exist from individuals, radio networks, TV networks, companies, storytellers, new-podcast-only networks, etc.

There is no set or pre-determined length, style, format, or production style of podcasts. You can simply create a podcast by talking about things that interest you. What’s best about podcasts is that they are relatively cheap and easy to produce. And their popularity is growing by the day.

1. **Purpose :-**

A podcast can have many purposes, but the main one is to **entertain its audience**. Podcast listeners might have one of several reasons to subscribe to a podcast, such as to:

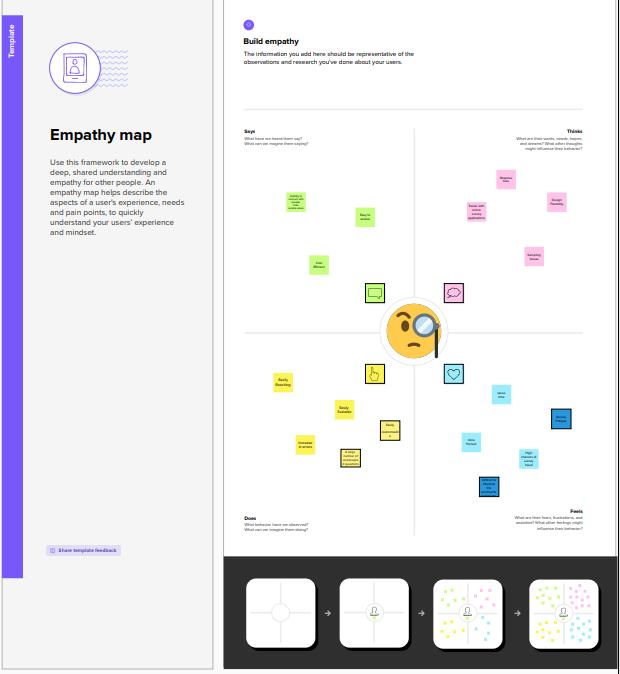
* Hear updates and breakdowns of current events.
* Learn about a new topic or industry.
* Laugh at cohosts’ riffing and jokes.
* Listen in on interviews with popular or famous guests.
* Experience an audio drama or narrative storytelling.

But behind each of these reasons is the desire to be entertained. Whether listeners want to learn something new or simply have something to take their minds off of a mundane task, they want to enjoy the experience of listening to a podcast.

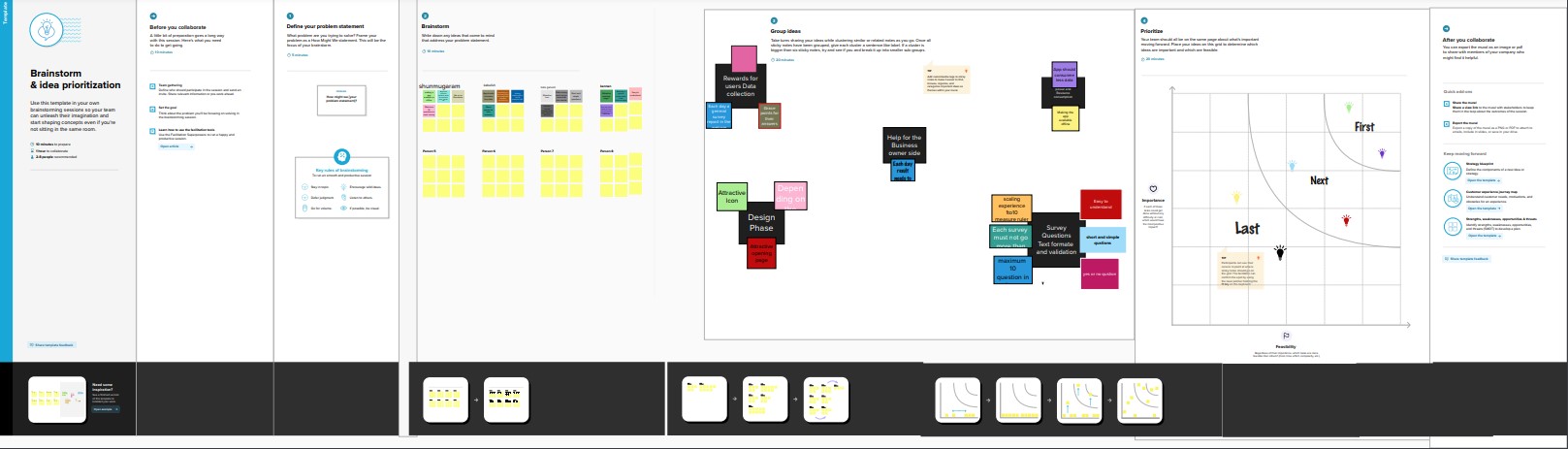
If they don’t enjoy it, they probably wouldn’t keep coming back.

**2.Problem Definition &DesignThinking:-**

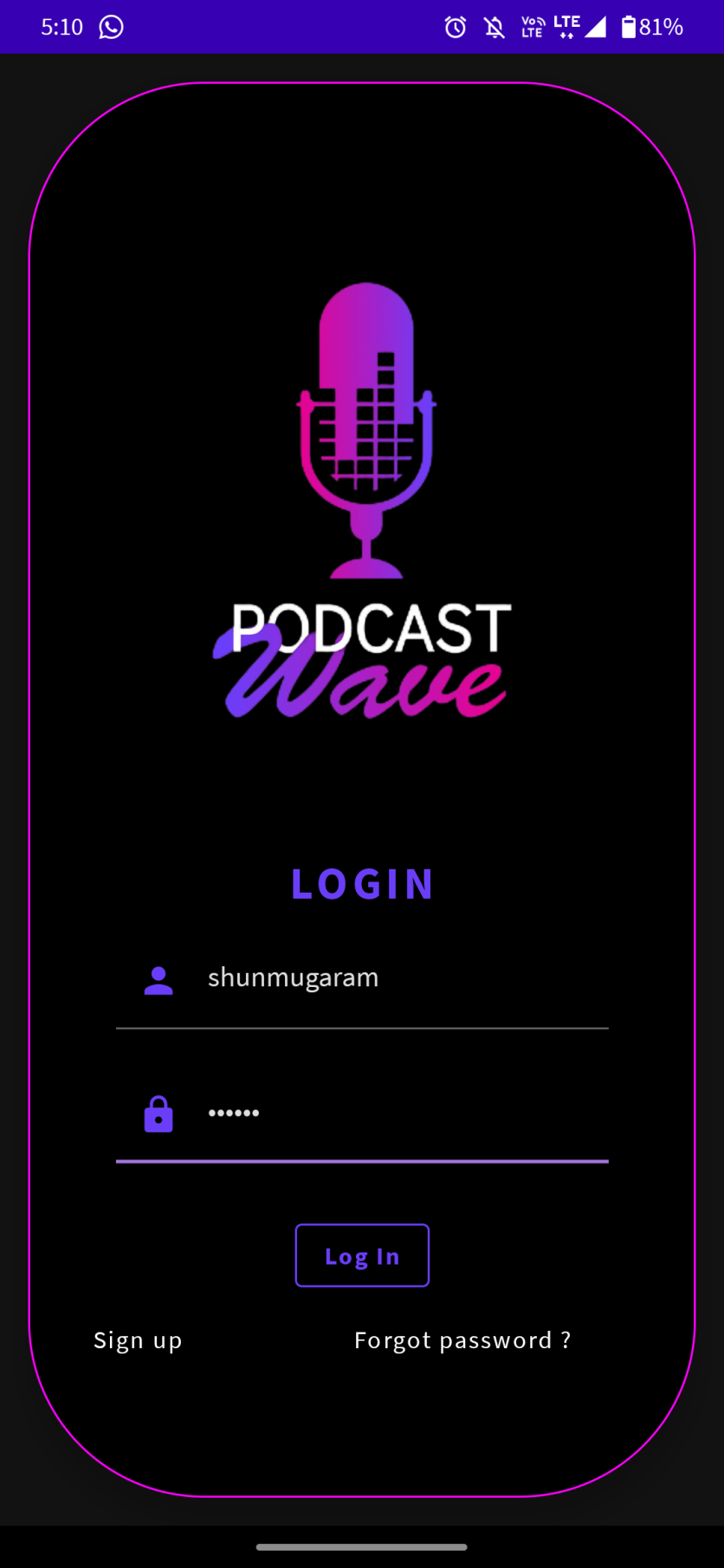
**2.1 Empathy Map:-**

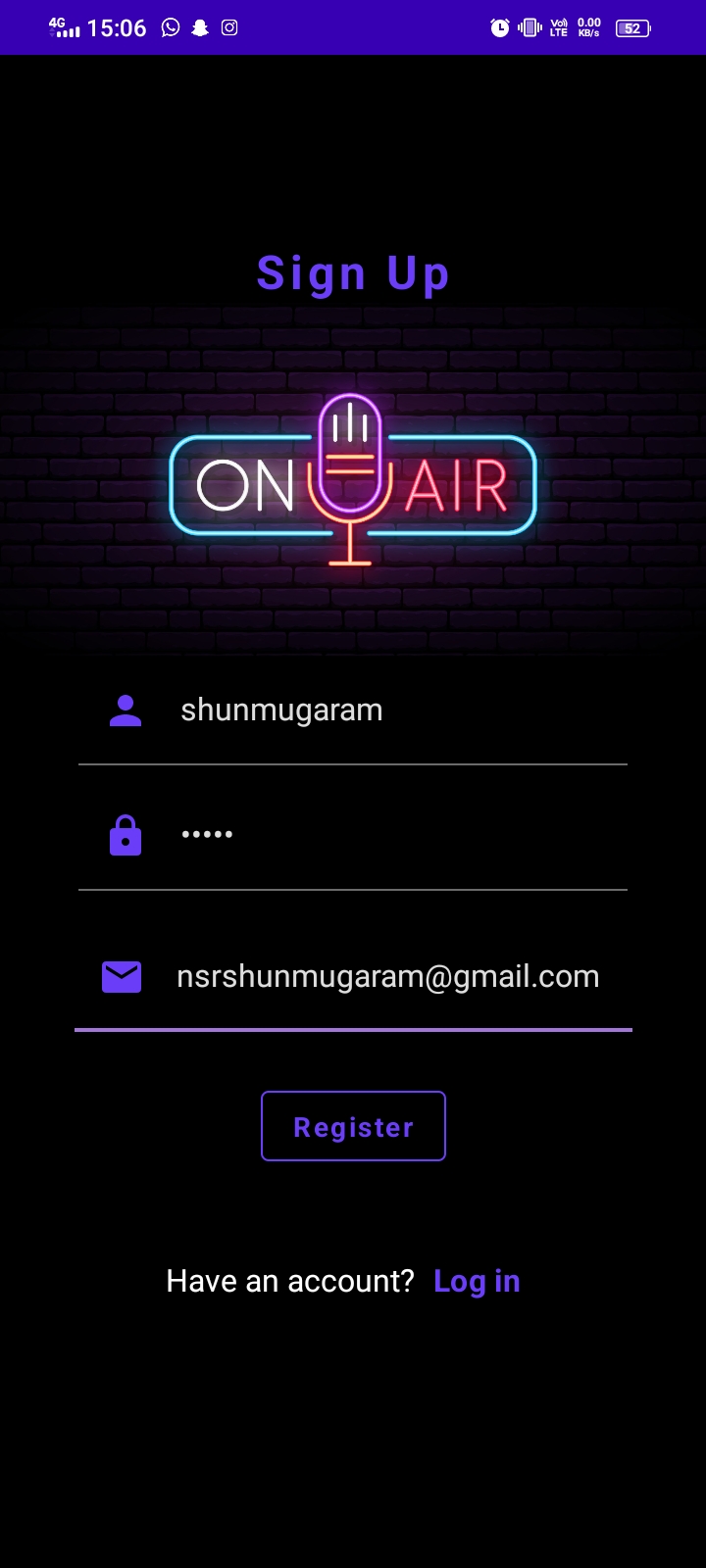
****

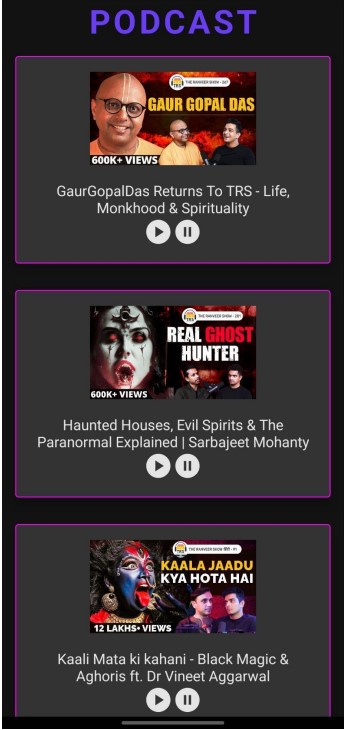
**2.2  Brainstorming Map:-**



**3.Result :-**







**Advantage :-**

**1. Convenience of Podcasting :**

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](https://quickstartpodcast.com/best-podcast-listening-apps-ios-and-android/). Podcast is an on demand technology

The most convenient part of podcasting experience is you can decide to the time and place of when you want to consume the content.

**2. No restriction on time for podcast listening :**

Podcasting is very good for listeners as they do not have to log in or tune in at a particular time similar to a radio or TV show.

You can chose the time when you want to listen to a podcast. You can listen to it whenever you are free ! . People listen to podcasts while jogging , walking as a pert of their exercise or they listen to it during their daily commute to office.

**Disdvantage :-**

### 1. Accessibility for some audience can be an issue

Internet is required for people to access the podcasts and it becomes difficult to reach to a wider audience if internet is not available.

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.

### 2. Finding and reaching to your 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.

It can be really disheartening and test of patience to produce episodes after episodes without any audience out there listening to you.

It takes a lot of time and efforts to find the right niche and audience for the content you are producing. With large scale production studios and movie / TV starts entering this space it has become really competative.

**5. Application :-**

Compatible with Android Auto and Apple CarPlay | Trim silence | Variable speed playback

Pocket Casts is a simple yet multi-faceted podcast app, making it our pick for the best podcast app. It offers hand-curated podcast suggestions to make it a simple matter to discover new podcasts and episodes. There is a seemingly endless database of podcast episodes, both new and old, with easy management of your favorite podcasts, so you can access them that much faster. Plus, there are extra tools like the ability to switch from dark to light background, plus several different modes, like Slide Over, Split View, and Picture in Picture to simplify viewing.

It is easy to build a playback queue to catch all your favorite moments with the ability to play between 0.5 and 3x. Even better, there is no need to subscribe, so you can enjoy your favorite shows in peace and without a financial obligation

**6. Conclusion :-**

## With the increase of easy-to-use programs, and the increase of mobile technology, we are able to create, access, and listen to podcasts wherever we go.  There is a huge selection of podcasts being offered to consumers, with new ones being offered everyday.  No matter what the subject you are interested in, chances are there is a podcast for it.  Still no? Why not create your own!   Podcasts offer educators and students new ways to share knowledge and communicate about understandings. I hope that this learning object has increased your understanding of podcasting and has allowed you to consider using them in your life.   As a concluding activity please complete the poll below.  Also, if you are interested you can deepen your understanding of podcasting using the resources collected below.

**7. Future Scope:-**

The future is bright, right now there is a bit of a gold rush. The genre is expanding in terms of audience. I can’t even guess how many of my friends and family have no idea about podcasts.Once this part of the market moves away from radio and starts on podcasts millions more listeners will be looking for something to listen to.I think old shows will find new audiences and revive. Companies who create content will invest more money into drams, mystery all sorts of other genres. High level performers will start to make revenue from appearing and playing roles on podcasts.The democratization of the genre will enable anyone to create an audience, but it will be as hard as ever to actually build one. Smaller shows will compete with larger networks for audience shares….shows that stand out will get deals…those shows will be exploited by the larger networks…and the cycle will continue.It’s going to be a lot of fun.

**8.Appendix**

**Source code :-**

Loginactivity.kt

package com.example.podcastplayer

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.BorderStroke

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.\*

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.\*

import androidx.compose.material.icons.Icons

import androidx.compose.material.icons.filled.Lock

import androidx.compose.material.icons.filled.Person

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.em

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

class LoginActivity : ComponentActivity() {

private lateinit var databaseHelper: UserDatabaseHelper

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

databaseHelper = UserDatabaseHelper(this)

setContent {

PodcastPlayerTheme {

// A surface container using the 'background' color from the theme

Surface(

modifier = Modifier.fillMaxSize(),

color = MaterialTheme.colors.background

) {

LoginScreen(this, databaseHelper)

}

}

}

}

}

@Composable

fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {

var username by remember { mutableStateOf("") }

var password by remember { mutableStateOf("") }

var error by remember { mutableStateOf("") }

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

shape = RoundedCornerShape(100.dp),

modifier = Modifier.padding(16.dp).fillMaxWidth()

) {

Column(

Modifier

.background(Color.Black)

.fillMaxHeight()

.fillMaxWidth()

.padding(bottom = 28.dp, start = 28.dp, end = 28.dp),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

)

{

Image(

painter = painterResource(R.drawable.podcast\_login),

contentDescription = "", Modifier.height(400.dp).fillMaxWidth()

)

Text(

text = "LOGIN",

color = Color(0xFF6a3ef9),

fontWeight = FontWeight.Bold,

fontSize = 26.sp,

style = MaterialTheme.typography.h1,

letterSpacing = 0.1.em

)

Spacer(modifier = Modifier.height(10.dp))

TextField(

value = username,

onValueChange = { username = it },

leadingIcon = {

Icon(

imageVector = Icons.Default.Person,

contentDescription = "personIcon",

tint = Color(0xFF6a3ef9)

)

},

placeholder = {

Text(

text = "username",

color = Color.White

)

},

colors = TextFieldDefaults.textFieldColors(

backgroundColor = Color.Transparent

)

)

Spacer(modifier = Modifier.height(20.dp))

TextField(

value = password,

onValueChange = { password = it },

leadingIcon = {

Icon(

imageVector = Icons.Default.Lock,

contentDescription = "lockIcon",

tint = Color(0xFF6a3ef9)

)

},

placeholder = { Text(text = "password", color = Color.White) },

visualTransformation = PasswordVisualTransformation(),

colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

)

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

if (error.isNotEmpty()) {

Text(

text = error,

color = MaterialTheme.colors.error,

modifier = Modifier.padding(vertical = 16.dp)

)

}

Button(

onClick = {

if (username.isNotEmpty() && password.isNotEmpty()) {

val user = databaseHelper.getUserByUsername(username)

if (user != null && user.password == password) {

error = "Successfully log in"

context.startActivity(

Intent(

context,

MainActivity::class.java

)

)

//onLoginSuccess()

} else {

error = "Invalid username or password"

}

} else {

error = "Please fill all fields"

}

},

border = BorderStroke(1.dp, Color(0xFF6a3ef9)),

colors = ButtonDefaults.buttonColors(backgroundColor = Color.Black),

modifier = Modifier.padding(top = 16.dp)

) {

Text(text = "Log In", fontWeight = FontWeight.Bold, color = Color(0xFF6a3ef9))

}

Row(modifier = Modifier.fillMaxWidth()) {

TextButton(onClick = {

context.startActivity(

Intent(

context,

RegistrationActivity::class.java

))})

{

Text(

text = "Sign up",

color = Color.White

)

}

Spacer(modifier = Modifier.width(80.dp))

TextButton(onClick = { /\* Do something! \*/ })

{

Text(

text = "Forgot password ?",

color = Color.White

)

}

}

}

}

fun startMainPage(context: Context) {

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

ContextCompat.startActivity(context, intent, null)

}}

Mainactivity.kt

package com.example.podcastplayer

import android.content.Context

import android.media.MediaPlayer

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.BorderStroke

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.\*

import androidx.compose.foundation.rememberScrollState

import androidx.compose.foundation.verticalScroll

import androidx.compose.material.\*

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.em

import androidx.compose.ui.unit.sp

import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

PodcastPlayerTheme {

// A surface container using the 'background' color from the theme

Surface(

modifier = Modifier.fillMaxSize(),

color = MaterialTheme.colors.background

) {

playAudio(this)

}

}

}

}

}

@Composable

fun playAudio(context: Context) {

Column(modifier = Modifier.fillMaxSize()) {

Column(horizontalAlignment = Alignment.CenterHorizontally, verticalArrangement = Arrangement.Center) {

Text(text = "PODCAST",

modifier = Modifier.fillMaxWidth(),

textAlign = TextAlign.Center,

color = Color(0xFF6a3ef9),

fontWeight = FontWeight.Bold,

fontSize = 36.sp,

style = MaterialTheme.typography.h1,

letterSpacing = 0.1.em

)

}

Column(modifier = Modifier

.fillMaxSize()

.verticalScroll(rememberScrollState())) {

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

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

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp),

)

Text(

text = "GaurGopalDas Returns To TRS - Life, Monkhood & Spirituality",

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_1)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

painter = painterResource(id = R.drawable.img\_1),

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp)

)

Text(

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

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_2)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

painter = painterResource(id = R.drawable.img\_2),

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp)

)

Text(

text = "Kaali Mata ki kahani - Black Magic & Aghoris ft. Dr Vineet Aggarwal",

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_3)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

painter = painterResource(id = R.drawable.img\_3),

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp),

)

Text(

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

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_4)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

painter = painterResource(id = R.drawable.img\_4),

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp),

)

Text(

text = "Complete Story Of Shri Krishna - Explained In 20 Minutes",

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

Card(

elevation = 12.dp,

border = BorderStroke(1.dp, Color.Magenta),

modifier = Modifier

.padding(16.dp)

.fillMaxWidth()

.height(250.dp)

)

{

val mp: MediaPlayer = MediaPlayer.create(context, R.raw.audio\_5)

Column(

modifier = Modifier.fillMaxSize(),

horizontalAlignment = Alignment.CenterHorizontally

) {

Image(

painter = painterResource(id = R.drawable.img\_5),

contentDescription = null,

modifier = Modifier

.height(150.dp)

.width(200.dp),

)

Text(

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

textAlign = TextAlign.Center,

modifier = Modifier.padding(start = 20.dp, end = 20.dp)

)

Row() {

IconButton(onClick = { mp.start() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.play),

contentDescription = ""

)

}

IconButton(onClick = { mp.pause() }, modifier = Modifier.size(35.dp)) {

Icon(

painter = painterResource(id = R.drawable.pause),

contentDescription = ""

)

}

}

}

}

}

}

}

Registrationactivity.kt

package com.example.podcastplayer

import android.content.Context

import android.content.Intent

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.BorderStroke

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.\*

import androidx.compose.material.\*

import androidx.compose.material.icons.Icons

import androidx.compose.material.icons.filled.Email

import androidx.compose.material.icons.filled.Lock

import androidx.compose.material.icons.filled.Person

import androidx.compose.runtime.\*

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.draw.alpha

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.layout.ContentScale

import androidx.compose.ui.res.painterResource

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.input.PasswordVisualTransformation

import androidx.compose.ui.tooling.preview.Preview

import androidx.compose.ui.unit.dp

import androidx.compose.ui.unit.em

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

import com.example.podcastplayer.ui.theme.PodcastPlayerTheme

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

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

databaseHelper = UserDatabaseHelper(this)

setContent {

PodcastPlayerTheme {

// A surface container using the 'background' color from the theme

Surface(

modifier = Modifier.fillMaxSize(),

color = MaterialTheme.colors.background

) {

RegistrationScreen(this,databaseHelper)

}

}

}

}

}

@Composable

fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {

var username by remember { mutableStateOf("") }

var password by remember { mutableStateOf("") }

var email by remember { mutableStateOf("") }

var error by remember { mutableStateOf("") }

Column(

Modifier

.background(Color.Black)

.fillMaxHeight()

.fillMaxWidth(),

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.Center

)

{

Row {

Text(

text = "Sign Up",

color = Color(0xFF6a3ef9),

fontWeight = FontWeight.Bold,

fontSize = 24.sp, style = MaterialTheme.typography.h1,

letterSpacing = 0.1.em

)

}

Image(

painter = painterResource(id = R.drawable.podcast\_signup),

contentDescription = ""

)

TextField(

value = username,

onValueChange = { username = it },

leadingIcon = {

Icon(

imageVector = Icons.Default.Person,

contentDescription = "personIcon",

tint = Color(0xFF6a3ef9)

)

},

placeholder = {

Text(

text = "username",

color = Color.White

)

},

colors = TextFieldDefaults.textFieldColors(

backgroundColor = Color.Transparent

)

)

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

TextField(

value = password,

onValueChange = { password = it },

leadingIcon = {

Icon(

imageVector = Icons.Default.Lock,

contentDescription = "lockIcon",

tint = Color(0xFF6a3ef9)

)

},

placeholder = { Text(text = "password", color = Color.White) },

visualTransformation = PasswordVisualTransformation(),

colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

)

Spacer(modifier = Modifier.height(16.dp))

TextField(

value = email,

onValueChange = { email = it },

leadingIcon = {

Icon(

imageVector = Icons.Default.Email,

contentDescription = "emailIcon",

tint = Color(0xFF6a3ef9)

)

},

placeholder = { Text(text = "email", color = Color.White) },

colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)

)

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

if (error.isNotEmpty()) {

Text(

text = error,

color = MaterialTheme.colors.error,

modifier = Modifier.padding(vertical = 16.dp)

)

}

Button(

onClick = {

if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) {

val user = User(

id = null,

firstName = username,

lastName = null,

email = email,

password = password

)

databaseHelper.insertUser(user)

error = "User registered successfully"

// Start LoginActivity using the current context

context.startActivity(

Intent(

context,

LoginActivity::class.java

)

)

} else {

error = "Please fill all fields"

}

},

border = BorderStroke(1.dp, Color(0xFF6a3ef9)),

colors = ButtonDefaults.buttonColors(backgroundColor = Color.Black),

modifier = Modifier.padding(top = 16.dp)

) {

Text(text = "Register",

fontWeight = FontWeight.Bold,

color = Color(0xFF6a3ef9)

)

}

Row(

modifier = Modifier.padding(30.dp),

verticalAlignment = Alignment.CenterVertically,

horizontalArrangement = Arrangement.Center

) {

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

TextButton(onClick = {

context.startActivity(

Intent(

context,

LoginActivity::class.java

)

)

})

{

Text(text = "Log in",

fontWeight = FontWeight.Bold,

style = MaterialTheme.typography.subtitle1,

color = Color(0xFF6a3ef9)

)

}

}

}

}

private fun startLoginActivity(context: Context) {

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

ContextCompat.startActivity(context, intent, null)

}

User.kt

package com.example.podcastplayer

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

@Entity(tableName = "user\_table")

data class User(

@PrimaryKey(autoGenerate = true) val id: Int?,

@ColumnInfo(name = "first\_name") val firstName: String?,

@ColumnInfo(name = "last\_name") val lastName: String?,

@ColumnInfo(name = "email") val email: String?,

@ColumnInfo(name = "password") val password: String?,

)

Userdeo.kt

package com.example.podcastplayer

import androidx.room.\*

@Dao

interface UserDao {

@Query("SELECT \* FROM user\_table WHERE email = :email")

suspend fun getUserByEmail(email: String): User?

@Insert(onConflict = OnConflictStrategy.REPLACE)

suspend fun insertUser(user: User)

@Update

suspend fun updateUser(user: User)

@Delete

suspend fun deleteUser(user: User)

}

Userdatabase.kt

package com.example.podcastplayer

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

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

abstract class UserDatabase : RoomDatabase() {

abstract fun userDao(): UserDao

companion object {

@Volatile

private var instance: UserDatabase? = null

fun getDatabase(context: Context): UserDatabase {

return instance ?: synchronized(this) {

val newInstance = Room.databaseBuilder(

context.applicationContext,

UserDatabase::class.java,

"user\_database"

).build()

instance = newInstance

newInstance

}

}

}

}

Userdatabasehelper.kt

package com.example.podcastplayer

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :

SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {

companion object {

private const val DATABASE\_VERSION = 1

private const val DATABASE\_NAME = "UserDatabase.db"

private const val TABLE\_NAME = "user\_table"

private const val COLUMN\_ID = "id"

private const val COLUMN\_FIRST\_NAME = "first\_name"

private const val COLUMN\_LAST\_NAME = "last\_name"

private const val COLUMN\_EMAIL = "email"

private const val COLUMN\_PASSWORD = "password"

}

override fun onCreate(db: SQLiteDatabase?) {

val createTable = "CREATE TABLE $TABLE\_NAME (" +

"$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +

"$COLUMN\_FIRST\_NAME TEXT, " +

"$COLUMN\_LAST\_NAME TEXT, " +

"$COLUMN\_EMAIL TEXT, " +

"$COLUMN\_PASSWORD TEXT" +

")"

db?.execSQL(createTable)

}

override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME")

onCreate(db)

}

fun insertUser(user: User) {

val db = writableDatabase

val values = ContentValues()

values.put(COLUMN\_FIRST\_NAME, user.firstName)

values.put(COLUMN\_LAST\_NAME, user.lastName)

values.put(COLUMN\_EMAIL, user.email)

values.put(COLUMN\_PASSWORD, user.password)

db.insert(TABLE\_NAME, null, values)

db.close()

}

@SuppressLint("Range")

fun getUserByUsername(username: String): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getUserById(id: Int): User? {

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString()))

var user: User? = null

if (cursor.moveToFirst()) {

user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

}

cursor.close()

db.close()

return user

}

@SuppressLint("Range")

fun getAllUsers(): List<User> {

val users = mutableListOf<User>()

val db = readableDatabase

val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null)

if (cursor.moveToFirst()) {

do {

val user = User(

id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)),

firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)),

lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)),

email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)),

password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)),

)

users.add(user)

} while (cursor.moveToNext())

}

cursor.close()

db.close()

return users

}

}