TABLE OF CONTENT

TABLE OF CONTENT

1.1 Overview

Determine the scope of your podcast: Before you start creating your podcast, you need to determine what topics you will cover and who your target audience will be. Will your podcast bea general overview of a particular subject, or will it be focused on a specific niche? This will help you determine the length of your episodes, the tone of your podcast, and the type of guests or experts you may want to invite on as guests.

Choose your podcast format: There are different podcast formats to choose from, such as interviews, solo shows, co-hosted shows. panel discussions, and more. You need to determine which format will work best for your topic and target audience.

Develop your podcast content: Once you have determined your topic and format, you need to develop your podcast content. This includes writing scripts for your episodes, outlining key points to cover, and identifying any guests or experts you may want to interview.

Record and edit your podcast: After you have developed your content, it's time to record and edit your podcast. You will need to invest in quality recording equipment and software to ensure that your audio quality is top-notch. Editing your podcast involves cutting out any unnecessary segments, adding in music or sound effects, and enhancing the audio quality.

Publish and promote your podcast: Once your podcast is ready to go, it's time to publish and promote it. You can use podcast hosting platforms like Podbean, Buzzsprout, or Anchor to distribute your podcast to popular platforms like Apple Podcasts, Spotify, and Google Podcasts. You can also promote your podcast on social media, through email newsletters, and by reaching out to other podcasts or influencers in your industry.

1.2 purpose

A podcast project can be a great way to document the purpose and use of a project. Podcasts are an increasingly popular medium for communicating information, and can be easily accessed and shared by anyone with an internet connection. Here are some of the benefits of using a podcast to

document your project:

Reach a wider audience: Podcasts can be accessed from anywhere in the world, making them a great way to reach a global audience. By creating a podcast about your project, you can share information with people who may not have access to other forms of documentation.

Build engagement: Podcasts can be a great way to build engagement with your audience. By creating a regular podcast about your project, you can keep people informed about the latest developments and encourage them to get involved.

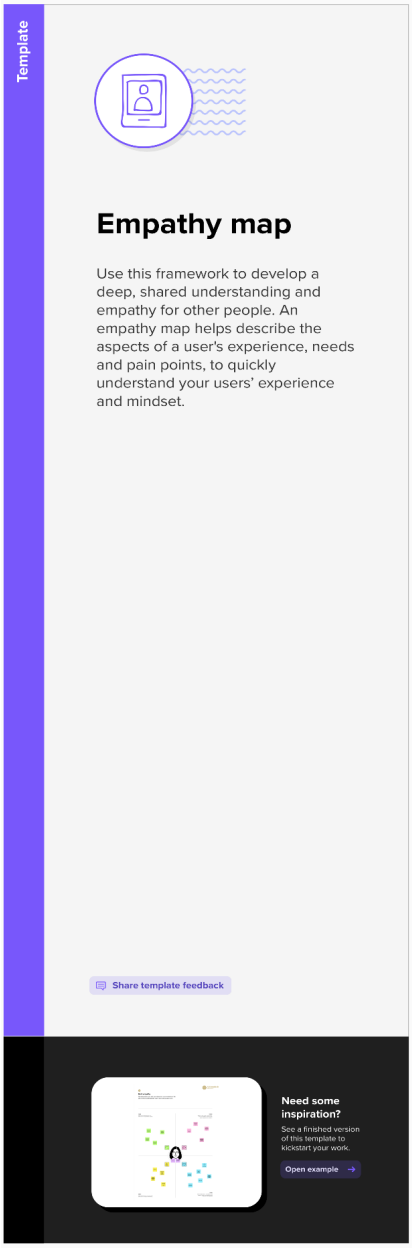
Create a personal connection: Podcasts can create a more personal connection with your audience. By hearing your voice and getting to know you through your podcast, listeners may feel more connected to your project and be more likely to support it.

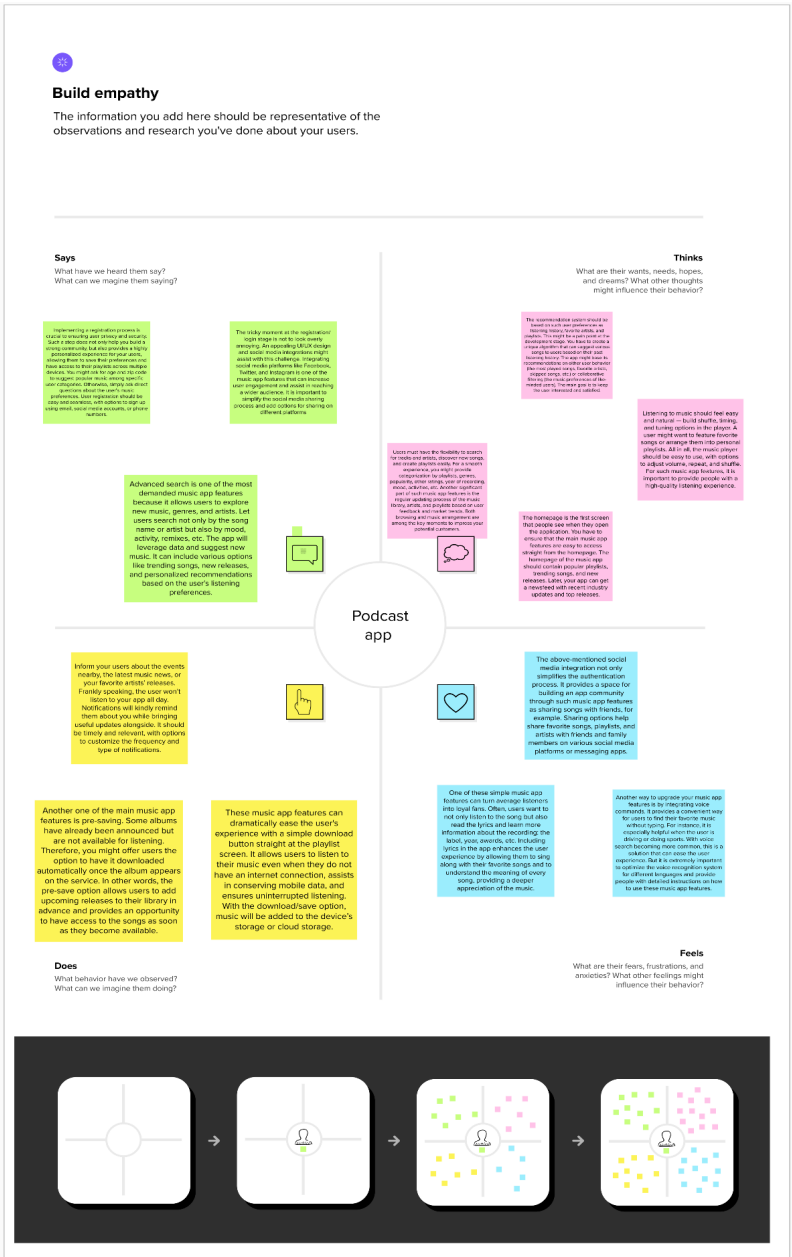
Provide in-depth information: Podcasts allow you to provide in-depth information about your project in a way that may not be possible through other forms of documentation. You can use interviews, discussions, and other formats to provide a more detailed understanding of your project.

Foster community: Podcasts can help foster a sense of community around your project. By creating a regular podcast, you can bring together people who are interested in your project and create a space for them to share ideas and collaborate.

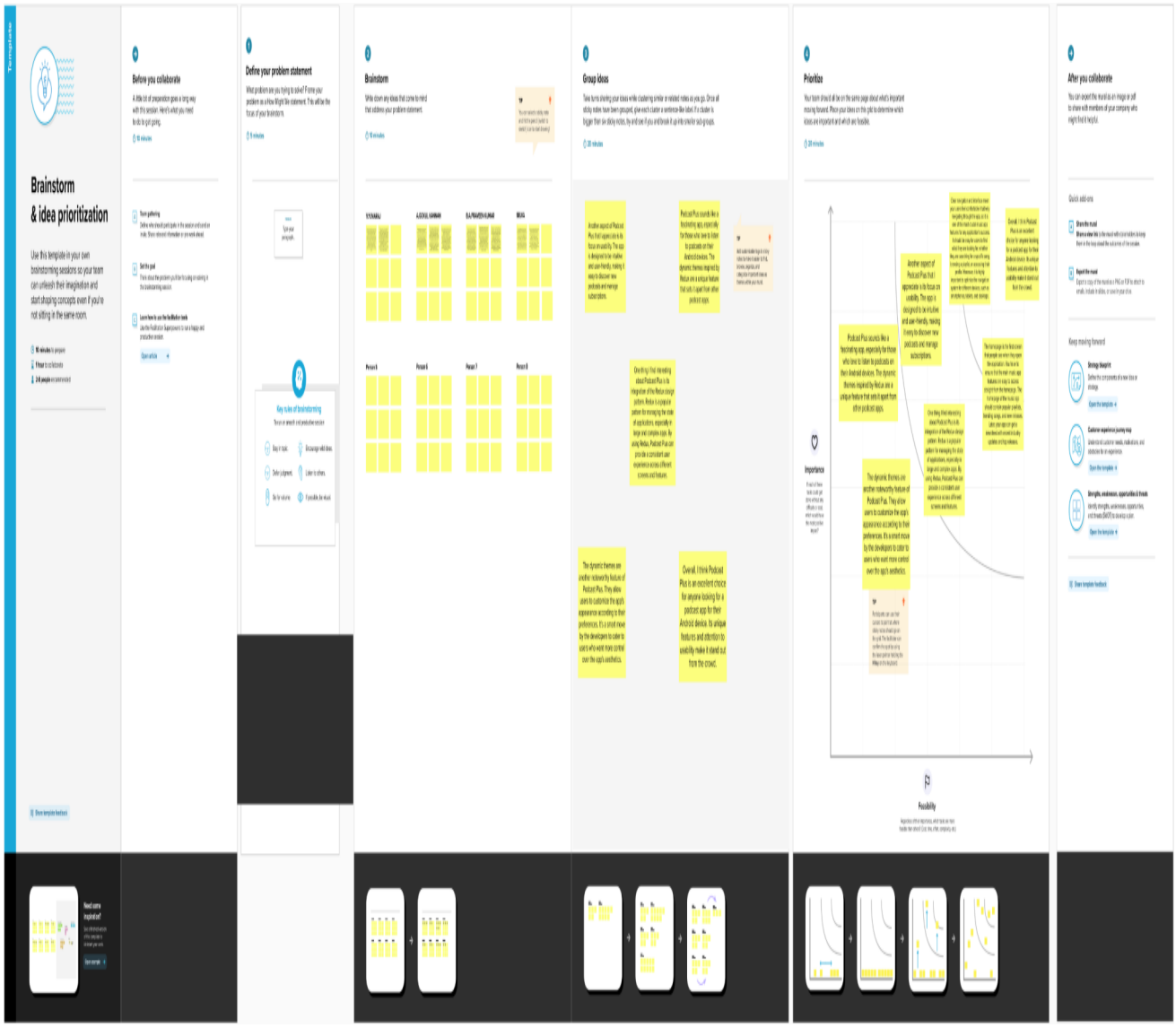
2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy map





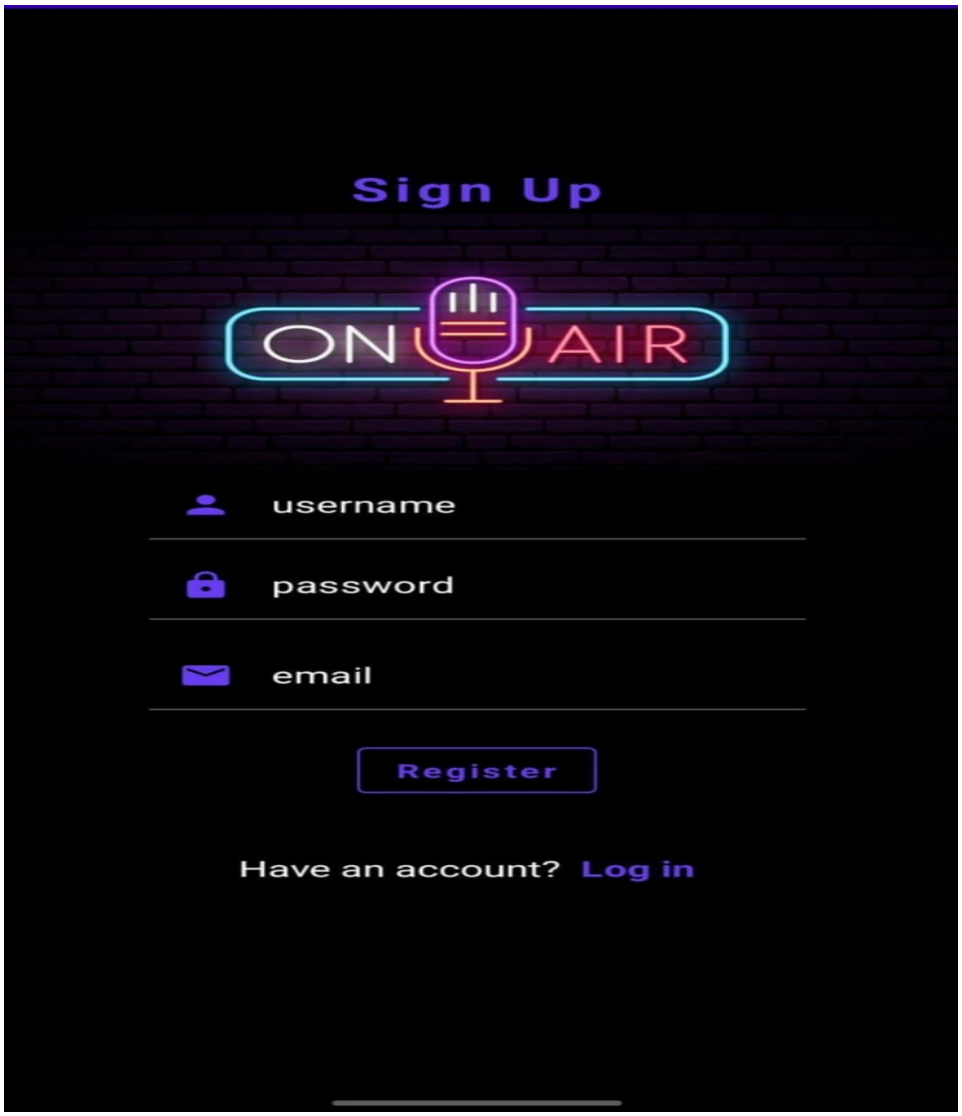
2.2 Ideation & Brainstorming map



3. RESULT

3.1 Final output of the project

screenshot 1



Screenshot 2

Screenshot 3

4. ADVANTAGES AND DISADVANTAGES

4.1 Advantages

Accessibility:

Podcasts can be accessed anytime, anywhere, as long as the listener has an internet connection and a compatible device. This makes it easier for people to consume content on their own time and at their own pace.

Convenience:

Unlike other forms of content, such as videos or blog posts, podcasts can be listened to while doing other activities, such as driving, exercising, or cooking.

Building rapport:

Podcasts can help build rapport between the host and listeners. The intimate nature of the medium allows hosts to share their personalities, opinions, and stories with their audience, which can help establish trust and connection.

Niche content:

Podcasts can cater to specific niches and interests that may not be covered by traditional media outlets. This allows for a diverse range of voices and perspectives to be heard.

Longevity:

Podcast episodes remain available online for as long as the host chooses to keep them up. This means that new listeners can discover and consume older content, which can help increase the lifespan and reach of the podcast.

4.2 Disadvantages

Simple User Interface:

The app should have a simple and user-friendly interface to make it easy for the users to navigate through the app and use its features.

Voice Recording Feature:

The app should have a voice recording feature that will allow users to easily record and save their podcast episodes. This feature should be simple to use and should not require any technical skills.

Cloud Storage:

The app should have cloud storage capability so that users can save their recordings in the cloud, which will ensure that they don't lose their work if their device is lost or damaged.

Transcription Feature:

The app should have a transcription feature that will allow users to transcribe recordings into text. This feature will be particularly useful for users who have difficulties with reading or writing.

5. APPLICATIONS

5.1 Entertainment:

Podcasts can be used as a form of entertainment, covering topics such as comedy, music, movies, and TV shows.

5.2 News and Current Events:

Podcasts can be used to provide in-depth analysis of news and current events, often featuring interviews with experts in various fields.

5.3 Education:

Podcasts can be used to provide educational content on a wide range of topics, including science, history, literature, and language learning.

5.4 Business:

Podcasts can be used to provide insights on business and entrepreneurship, featuring interviews with successful business leaders and experts.

5.5 Health and Wellness:

Podcasts can be used to provide information and insights on health and wellness topics, including fitness, nutrition, and mental health.

5.6 Sports:

Podcasts can be used to provide analysis and commentary on sports, including interviews with athletes, coaches, and other experts in the field.

5.7 Personal Development:

Podcasts can be used to provide guidance and inspiration for personal growth and development, covering topics such as mindfulness, productivity, and motivation.

6. CONCLUSION

Documentation is critical:

Developing a podcast app is a complex task, and documenting the various aspects of the project is essential for future reference. This documentation can help new team members understand the project, and it can also serve as a reference for maintenance and troubleshooting.

User experience is crucial:

A podcast app's success depends largely on its user experience. It should be easy to navigate, search for podcasts, and play episodes. The app should also have features such as playlists and recommendations to enhance the user experience.

Technical considerations are important:

Developing a podcast app requires knowledge of various technologies, such as API integration, database management, and media streaming. Careful consideration must be given to these technical aspects to ensure the app's stability and performance.

Collaboration is key:

Developing a podcast app requires a team effort. Collaboration between team members is essential to ensure that everyone is on the same page, and progress is made efficiently. Tools such as project management software and version control systems can aid in collaboration.

Overall, the podcast project app for documentation was a valuable experience. It allowed me to gain insight into the various aspects of developing a podcast app and reinforced the importance of documentation, user experience, technical considerations, and collaboration.

7. FUTURE SCOPE

Define the purpose and scope of the app: Before you start designing and building the app, you need to have a clear understanding of what you want to accomplish with it. Determine what the app will be used for, who the target audience is, and what features it will include.

Research existing apps: Look at other podcast project apps that are available in the market. Analyze their features, functionalities, user interface, and user experience. This will help you identify what works well and what can be improved upon.

Create a wireframe and prototype: A wireframe is a visual representation of the app's layout and functionality. Use wireframing tools like Figma or Sketch to create a rough sketch of the app's layout. Then, create a prototype to test the app's user experience.

Define the tech stack: Based on the scope of the project, determine what technology stack will be used to build the app. Will it be a native app or a hybrid app? What programming language will be used? Which database will store the data?

Develop the app: Once you have the wireframe and prototype, start building the app. Divide the app into modules, and start building each module one by one. Test each module as you go along to ensure that everything is working correctly.

Test and debug: Once the app is built, test it rigorously to ensure that it works as intended. Identify any bugs or issues that need to be fixed.

Launch the app: Once you're confident that the app is ready, launch it on the App Store or Google Play Store. Promote the app on social media and other channels to get the word out.

Maintain and update: After launching the app, continue to maintain and update it regularly. Listen to feedback from users and implement improvements to keep the app relevant and useful.

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)
}
```

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
```

```
    }
}
```

LOGINACTIVITY.kt

Search or jump to…

Pulls

Issues

Codespaces

Marketplace

Explore

@yuvaraja001

smartinternz02

/

PodcastPlayer

Public

Fork your own copy of smartinternz02/PodcastPlayer

Code

Issues

Pull requests

Actions

Projects

Security

Insights

PodcastPlayer/app/src/main/java/com/example/podcastplayer/LoginActivity.kt

@keerthibacha

keerthibacha Podcast project

Latest commit baae6ba on Mar 10

History

1 contributor

197 lines (177 sloc) 6.86 KB

```
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)
    }}
```

REGISTRTIONACTIVITY.kt

[3:58 am, 11/04/2023] Yuvaraj Bca 2: 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))
```
[3:59 am, 11/04/2023] Yuvaraj Bca 2:
```
    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::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 = ""
                    )
                }

            }
        }

    }
}
}
}
```

Android manifest.kt

```
<?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>
```
complete program coding and run the app hardware device