NAAN MUDHALVAN

PROJECT REPORT ON PODCAST PLUS:

A REDUX-INSPIRED APP WITH DYNAMIC THEMES FOR ANDROID

SUBMITTED BY

TEAM ID

NM2023TMID10614

TEAM LEADER

STEPHEN R YUPITHAR

TEAM MEMBERS

SAKTHI   SRI VIGNESH G.P   STEPHEN.R

Project Report

1. Introduction:
1.1 Overview

Podcast Plus: A Redux-Inspired Podcast App with Dynamic Themes for Android

Podcast Plus is a mobile app designed for listening to podcasts, inspired by the Redux framework used in web application development. The app allows users to search for and subscribe to their favorite podcasts, as well as discover new ones through recommendations and popular podcasts. The app also features a personalized homepage that displays the latest episodes from subscribed podcasts as well as curated recommendations based on listening history. Users can download episodes for offline listening, adjust playback speed, and create playlists. The Redux-inspired design of the app ensures a smooth and responsive user experience, with minimal lag and reliable performance even when dealing with large amounts of data. Overall, Podcast Plus offers a streamlined and intuitive way to discover and enjoy podcasts on the go.

1.2 Purpose

Podcast Plus is a mobile application designed for listening to podcasts. The app allows users to discover and subscribe to a wide range of podcasts, and offers a convenient platform for listening to them on-the-go.

Some of the key features of the app include:

1. Discovering new podcasts: Podcast Plus offers a curated selection of podcasts across various genres, making it easy for users to find new and interesting content.
2. Subscribing to podcasts: Once a user finds a podcast they enjoy, they can easily subscribe to be it within the app, which will ensure that new episodes are automatically downloaded and ready for listening.
3. Managing podcast subscriptions: The app allows users to manage their podcast subscriptions, including the ability to unsubscribe from podcasts that are no longer of interest.
4. Listening to podcasts: Podcast Plus provides a user-friendly interface for listening to podcasts, with features such as the ability to adjust playback speed, skip forward or back, and set sleep timers.

Overall, Podcast Plus is a useful tool for anyone who enjoys listening to podcasts and wants a convenient and easy-to-use platform for discovering and managing their favorite shows.

Podcast Plus, if built with a Redux-Inspired architecture, can be offer several benefits to both developers and users. Here are few things that can be achieved through a Redux-Inspired Podcast Plus app:
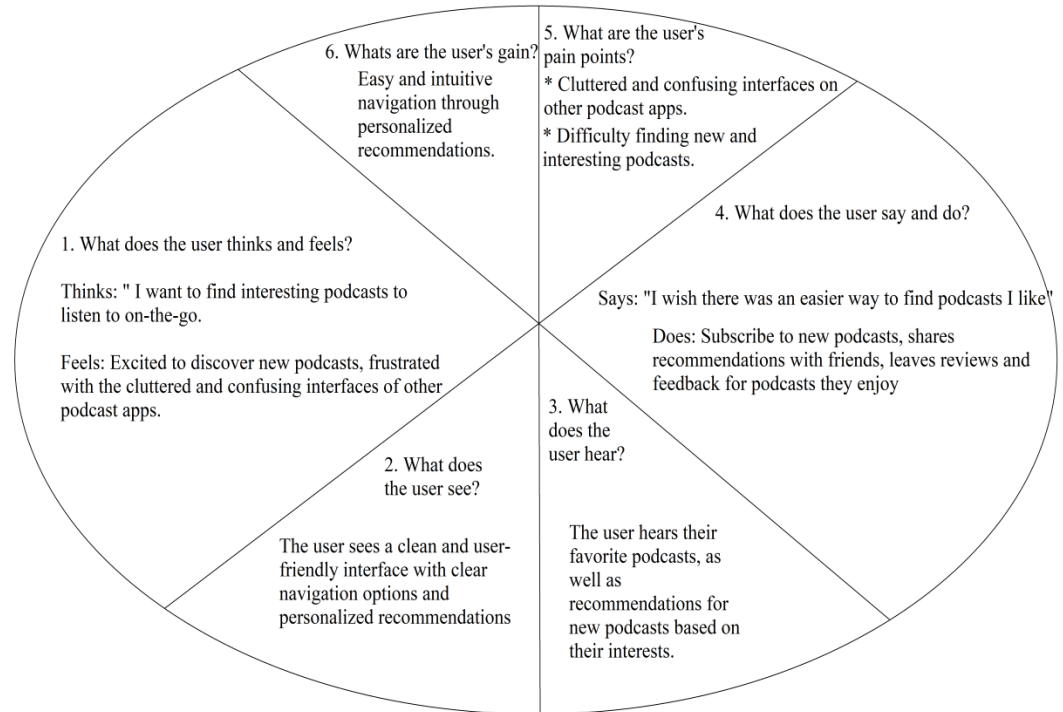
1. Predictable state management: Redux provides a predictable and centralized way to manage application state. This can be useful in Podcast Plus, where there may be several components that need to share the same data or update the same state. By using Redux, the app can ensure that the state is consistent and predictable across all components.
2. Time-travel debugging: One of the unique features of Redux is time-travel debugging, which allows developers to replay past actions and see how they affected the application state. This can be incredibly useful when debugging complex issues in Podcast Plus, as it allows developers to pinpoint the source of the problem and see how it propogated through the system.
3. Improved performance: By using Redux, the Podcast Plus app can minimize the number of unnecessary re-renders and improve performance. This is because Redux encourages developers to keep state changes to a minimum and only update components when necessary.
4. Module code: Another benefit of using Redux is that encourages developers to write modular and reusable code. This can make it easier to maintain and extend the app over time, as new features can be added without disrupting existing code.

Overall, a Redux-Inspired Podcast Plus app can offer improved state management, debugging capabilities, performance, and code modularity, which can benefit both developers and users.

2. Problem definition and Design Thinking:

2.1 Empathy Map:

By considering the user's thoughts, feelings, and pain points, as well as their gains, the Podcast Plus app can be designed to meet their needs and provide a better listening experience.

5. What are the user's pain points?
* Cluttered and confusing interfaces on other podcast apps.
* Difficulty finding new and interesting podcasts.

6. Whats are the user's gain?
Easy and intuitive navigation through personalized recommendations.

4. What does the user say and do?
Says: "I wish there was an easier way to find podcasts I like"
Does: Subscribe to new podcasts, shares recommendations with friends, leaves reviews and feedback for podcasts they enjoy

1. What does the user thinks and feels?
Thinks: " I want to find interesting podcasts to listen to on-the-go.
Feels: Excited to discover new podcasts, frustrated with the cluttered and confusing interfaces of other podcast apps.

3. What does the user hear?
The user hears their favorite podcasts, as well as recommendations for new podcasts based on their interests.

2. What does the user see?
The user sees a clean and user-friendly interface with clear navigation options and personalized recommendations

2.2 Ideation & Brainstorming map:

1. Introduction: What is Redux and How does it work?
2. Planning the app: What features will the app have and how will they be implemented using Redux?
3. Designing the user interface: How to create a user-friendly interface that integrates with redux?
4. Writing code: How to write code using the Redux architecture and what challenges may arise.
5. Testing the app: How to test the app and ensure it is bug-free?
6. Deploying the app: How to deploy the app to the app store and what steps are involved?
7. User feedback: How to gather feedback from users and use it to improve the app?

8. Scaling the app: How to scale the app as the number of users grows?
9. Building a community: How to build a community around the app and engage with users?
10. Conclusion: final thoughts in building an app with Redux and what's next for the podcast.

Potential Guests:

1. Dan Abramov – Creator of Redux
2. Mark Erikson – Redux maintainer and co-author of the Redux Toolkit.
3. Michael Jackson – Co-creator of react router and host of the react podcast.
4. Kent C. Dodds – Javascript expert and host of the chats with kent podcast.
5. Andrew Clark – Co-creator of react and author of the react-redux library.

Monetization Ideas:

1. Sponsored content from companies that provide tools and services for app development.
2. Selling merchandise related to the podcast.
3. Offering premium content for subscribers, such as exclusive episodes or additional resources and tutorials.
4. Accepting donations from listeners who want to support the podcast.

Marketing Ideas:

1. Partnering with app development conferences and events to promote the podcast.
2. Guest appearance on other technology-related podcasts to promote the show.
3. Creating social media accounts and posting regular updates and teasers for upcoming episodes.
4. Running a contest to encourage listeners to submit their own app ideas and featuring the best ones on the show.

Overall, this podcast could provide a valuable resource for developers looking to learn more about Redux and how to use in their own app development projects. With a strong lineup of guests and a focus on practical tips and advice, the show could attract a dedicated following of listeners interested in learning more about this popular architecture.

3. **Source Code**:

MainActivity.kt

```kotlin
package com.example.podcastapplication

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.podcastapplication.ui.theme.PodcastApplicationTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            PodcastApplicationTheme {
                // 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)
```

```kotlin
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,
```

```kotlin
        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 = ""
```

```kotlin
                )
            }

        }
    }

}


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() {
```

```kotlin
            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)
```

```kotlin
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,
```

```kotlin
        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.podcastapplication

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.podcastapplication.ui.theme.PodcastApplicationTheme

class RegistrationActivity : ComponentActivity() { private lateinit var databaseHelper:
UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            PodcastApplicationTheme {
                // A surface container using the 'background' color from the theme
                Surface(

```kotlin
                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(
```

```kotlin
            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(
```

```kotlin
            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
                        )
```

```kotlin
                )

        } 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)
        )
    }

    }
  }
}
```

```kotlin
private fun startLoginActivity(context: Context) {
    val intent = Intent(context, LoginActivity::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

LoginActivity.kt

package com.example.podcastapplication

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.podcastapplication.ui.theme.PodcastApplicationTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {
            PodcastApplicationTheme {
                // 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(),
```

```kotlin
                    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 = {
```

```kotlin
                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)
}}
```

AndroidManifest.xml

```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.PodcastApplication"
        tools:targetApi="31">
        <activity
            android:name=".RegistrationActivity"
            android:exported="false"
            android:label="@string/title_activity_registration"
            android:theme="@style/Theme.PodcastApplication" />
        <activity
            android:name=".MainActivity"
            android:exported="false"
            android:label="@string/title_activity_login"
            android:theme="@style/Theme.PodcastApplication" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.PodcastApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```
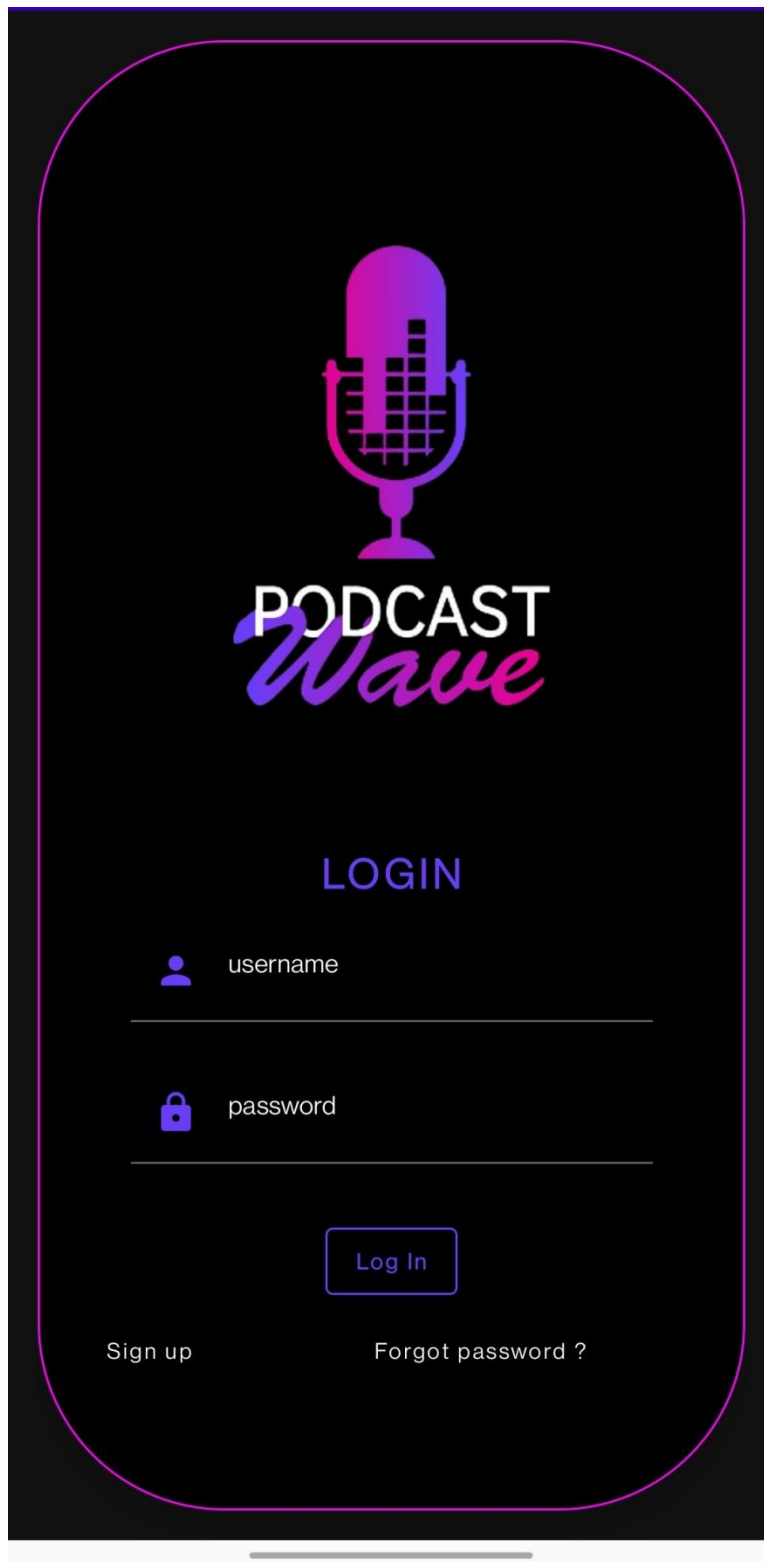
4. Screenshot:

HomePage -

SignUp –

Podcasts –

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

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

Complete Story Of Shri Krishna - Explained In 20 Minutes

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

5. Demonstration Video Link:

https://drive.google.com/file/d/1essLq0s4_NnaAvxmZMKB77zMCEVGOl_2/view?usp=share_link

6. Advantages and Disadvantages:

Advantages:

1. Dynamic Themes: Podcast Plus has dynamic themes which allow users to customize their podcast listening experience according to their preference.
2. User-friendly interface:
3. Redux-inspired architecture: The redux-inspired architecture provides a predictable state container for the app, making it more reliable and efficient.
4. Availability: Podcast plus is available on Android, making it accessible to a wide range of users.

Disadvantages:

1. Limited availability: As of now, Podcast Plus is only available on android, limiting its reach to iOS users.
2. Limited features: while Podcast Plus has dynamic themes, it may lack other features that are available in other podcast apps.
3. Performance issues: Depending on the user's device, Podcast Plus may have a performance issues, such as lag or crashes.
4. Cost: While Podcast Plus may be free, it may also have in-app purchases or require a subscription to access certain features, which can be a disadvantage for some users.

7. Applications:
1. Personalized podcast listening: The dynamic themes of Podcast plus allow users to customize their podcast listening experience, making it more personalized.
2. Education: Podcast Plus can be used as an educational tool, allowing users to listen to lectures, interviews, and other educational content on-the-go.
3. Entertainment: podcast plus can be used for entertainment purposes, as it allows users to access a variety of podcasts on different topics such as comedy, sports, music, religious belief and more.
4. News and current events: Podcast Plus can be used as a language learning tool by listening to podcasts in the target language.

5. Marketing and business: Podcast Plus can be used for marketing and business purposes, as it allows users to listen to podcasts on marketing strategies, business trends and more.

8. Conclusion:

In conclusion, Podcast Plus is a unique podcast app that offers dynamic themes and a redux-inspired architecture for a personalized and reliable listening experience on android devices. While the app has its advantages, suck as its user-friendly interface and availability, it may also have limitations, including limited availability for iOS users, potential performance issues, and potential cost. Nonetheless, the applications of Podcast Plus are diverse, ranging from entertainment to education, language learning, personal development, research, and more. Overall, podcast Plus offers a unique and customizable listening experience for android users