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

Abstract:

The Podcast Project App is a mobile application designed to make it easier for podcasters to create, and distribute their podcast. The app allows users to easily upload their podcast episodes and set up a podcast website. It also provides a suite of powerful analytics tools to help users better understand their audience and optimize their content. Additionally, the app includes access to a global community of podcasters, giving users the support and resources, they need to succeed. With its intuitive design and comprehensive features, the Podcast Project App is the perfect platform for aspiring podcasters to start and grow their podcast. The Podcast Project app would enable users to create their own podcast series. They could record episodes, edit audio, and upload their podcast to popular streaming services. The app would also provide users with tools to analyze the success of their podcast, including metrics such as reviews.

Introduction :

This document serves as the project documentation for the podcast application project. This project aims to create a mobile app that allows users to access and listening audio podcasts. The application will provide users with a wide selection of audio podcasts, including both popular and niche titles, and will allow them to create personal playlists of their favorite podcasts. Additionally, the application will support social features, such as user profile pages, podcast sharing, and podcast recommendations. The concept of this project is to create an interactive podcast platform that allows users to easily find podcasts. The platform will be tailored to each user's interests and preferences, allowing them to browse through recommended podcasts and easily find content that they like. The platform will also feature an interactive social aspect, where users can connect and share their thoughts and opinions on different podcast episodes. Additionally, users will be able to create playlists of their favorite episodes and easily search for new ones. The platform will also provide access to other related content such as news and blog posts about the topics discussed in the podcast episodes. The goal of this project is to create an easy-to-use and engaging platform that helps users find and share the best podcast content.

Objectives

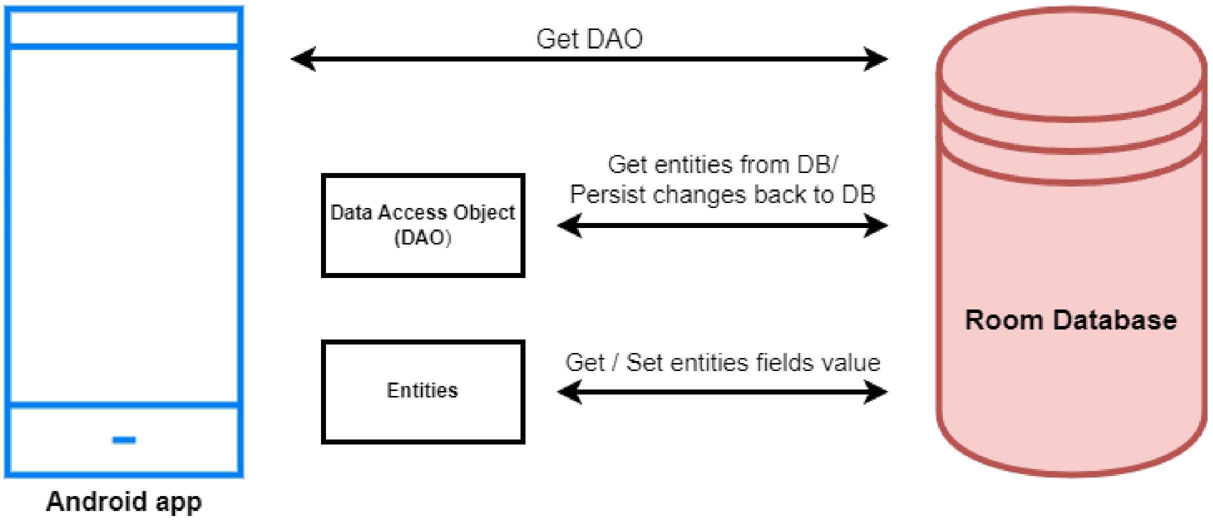
- To implement an efficient and maintainable state management solution using Redux principles.

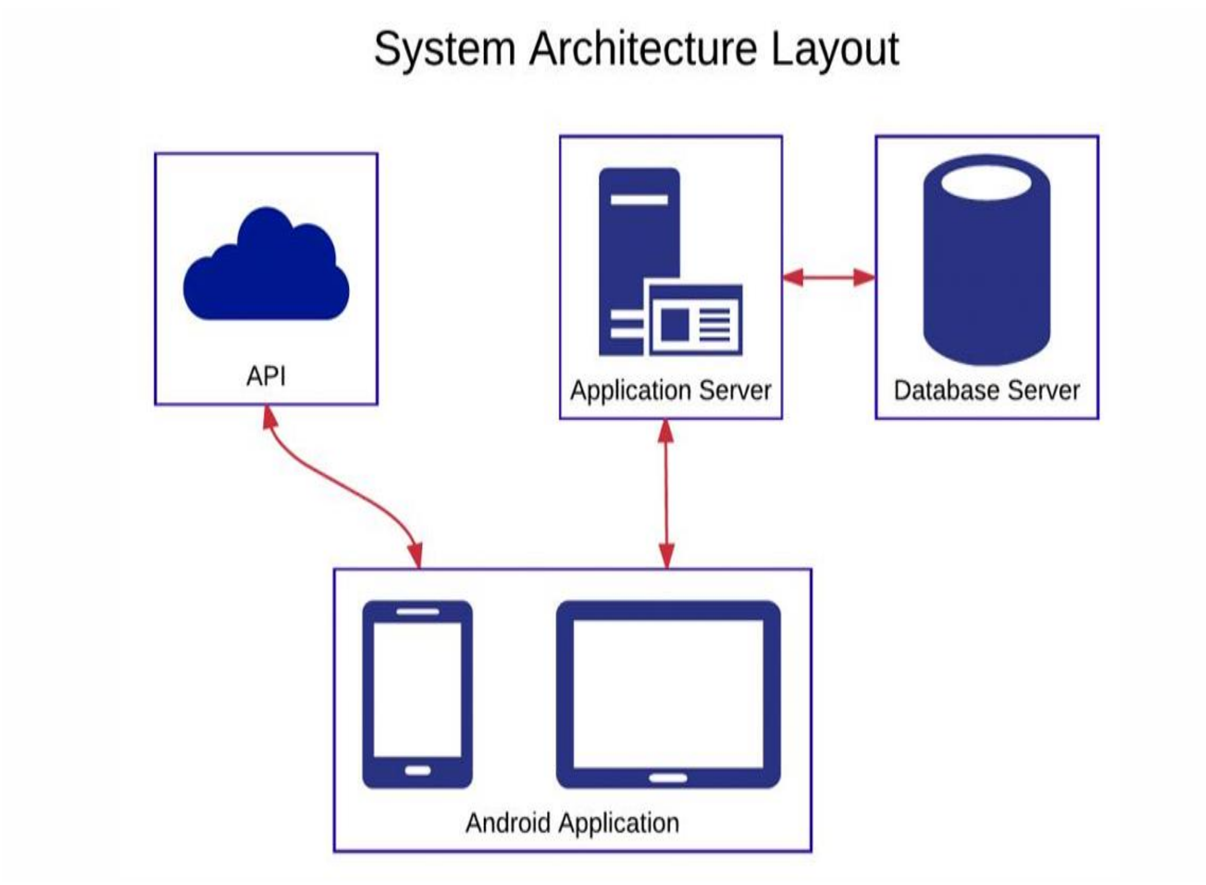
- To provide dynamic themes that adapt to user preferences and environments.
- To ensure a smooth and intuitive user experience for podcast discovery and playback.
- To integrate essential features for podcast management and interaction.

Features

- **Advanced State Management**: Utilizing Redux principles to manage app state efficiently.
- **Dynamic Themes**: Offering a variety of themes that can change based on user preference, time of day, or ambient light.
- **Podcast Discovery**: Curated recommendations, trending podcasts, and search functionality.
- **Custom Playlists**: Users can create and manage custom playlists.
- **Offline Listening**: Download episodes for offline playback.
- **User Interactions**: Allowing users to comment, rate, and share episodes.

Architecture :


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



System Requirements :

Operating System: Android-7 or later

Ram: 1Gb (Minimum)

Rom:20Mb (Minimum)

Internet connection

Tools Used:

Android Studio

Firebase

Kotlin

Proposed System :

The proposed system for the podcast app project is to create a user-friendly app that allows users to access a wide variety of podcasts. The app will be able to search through a database of podcasts, allowing users to easily find and access the content they are looking for. Additionally, the app will include features such as the ability to share podcasts with others, and bookmark favorite podcasts. Finally, the app will include a recommendation system, so users can discover new content they may be interested in.The proposed podcast app system aims to provide users with an intuitive and user-friendly platform for discovering, and listening to their favorite podcasts. The app will have a sleek and modern design that allows users to easily browse through a wide variety of podcasts based on genre, popularity, or personal preferences. The system will also offer personalized recommendations and preferences. Additionally, the system will provide social sharing features, allowing users to share their favorite episodes with friends and on social media. The podcast app system will be available on multiple platforms Android and enhancements to ensure a seamless listening experience for all users.

Working :

The podcast app project is an Android application that allows users to browse, search, and listen to podcasts. Here are some of the key components and functionalities of the podcast app project that work with Android Studio:

[1] User Interface: The podcast app project uses Android Studio to create the user interface. The user interface includes various components such as buttons, text views, and list views that allow users to interact with the app.

[2] Podcast Feed: The podcast app project uses RSS feeds to retrieve podcast data. Android Studio is used to create the code that fetches the podcast data from the RSS feeds.

[3] Audio Player: The podcast app project includes an audio player that allows users to listen to podcast episodes. Android Studio is used to create the audio player user interface and handle audio playback.

[4] Search Functionality: The podcast app project includes a search feature allowing users to search for podcasts by keyword or category. Android Studio is used to create the search user interface and to handle the search functionality.

[5] Notifications: The podcast app project includes a notification system that alerts users when new episodes of their favorite podcasts are available. Android Studio is used to create the notification system.

[6] Overall, Android Studio is a key component of the podcast app project, providing the tools needed to create the user interface, handle audio playback, retrieve podcast data, and manage user authentication.

Program :

Log in activity code

```
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)
```

Main activity code

```
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),
                                contentDescribe
```

Registration activity code

```
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",
```

User code

```
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",
```

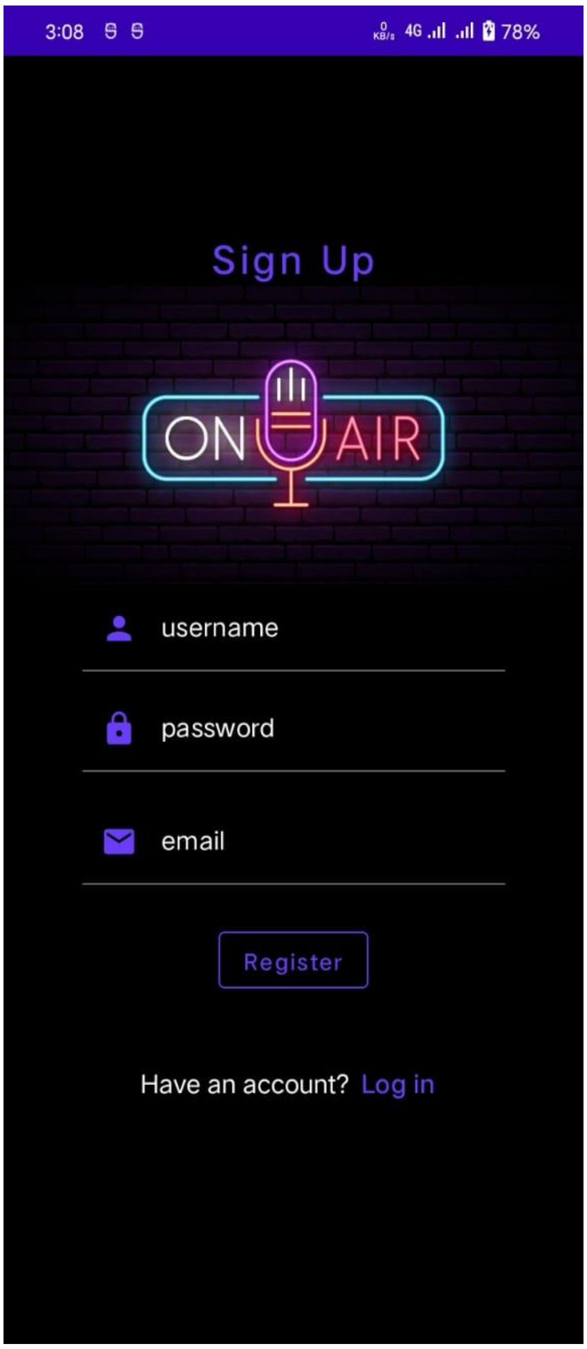
User database code

```
package com.example.podcastplayer
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
    companion object {
        @Volatile
        private var instance: UserDatabase? = null
        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

User database helper code

```
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)),
```

output :


3:08 78%

Sign Up

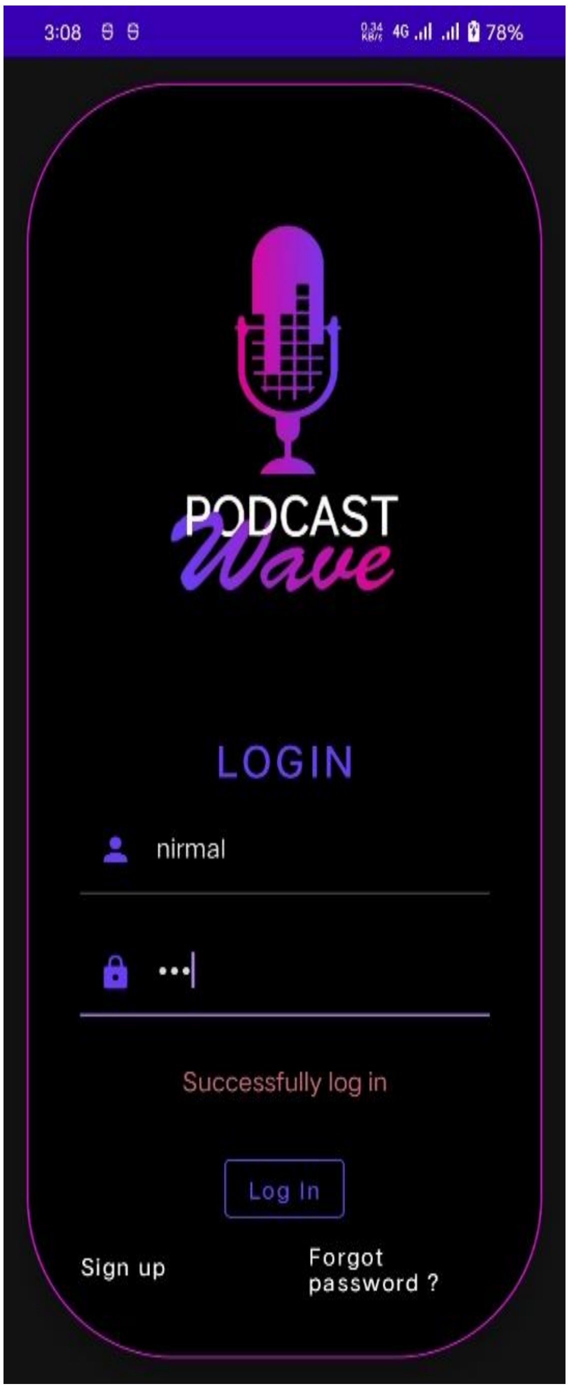
username

password

email

Register

Have an account? Log in


3:08 78%

LOGIN

nirmal

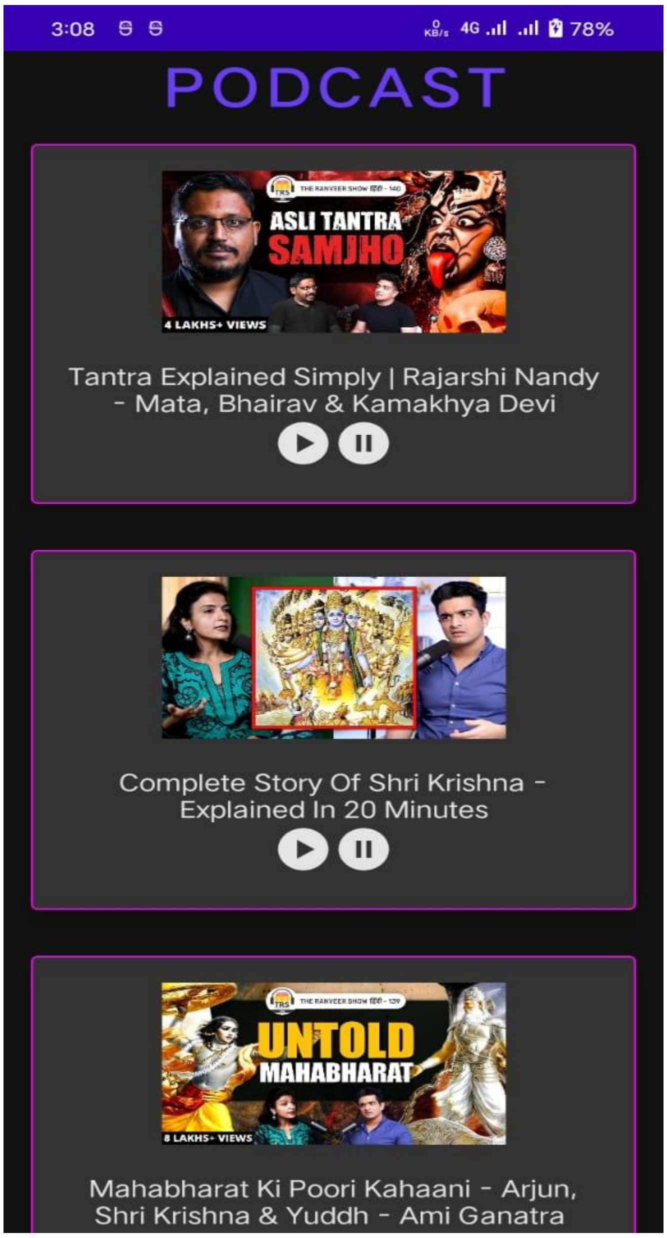
•••

Successfully log in

Log In

Sign up

Forgot password ?



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

Challenges and Solutions

- **State Management Complexity**: Managing app state efficiently as the app scales. Solution: Using Redux to ensure predictable state changes and maintainability.

- **Dynamic Theming**: Adapting themes to various user preferences and conditions. Solution: Implementing a flexible theme engine with real-time adjustments.
- **Performance Optimization**: Ensuring smooth performance with increasing features. Solution: Code optimization, lazy loading, and effective memory management.

Future Enhancements

- **AI Recommendations**: Utilizing AI to provide personalized podcast recommendations.
- **Cross-Platform Support**: Expanding the app to iOS and web platforms.
- **Enhanced User Engagement**: Introducing social features like following, messaging, and live streaming.

Conclusion

The Redux-Inspired Podcast App with Dynamic Themes is set to redefine the podcast listening experience by combining efficient state management with visually engaging and adaptive themes. It aims to be an essential tool for podcast enthusiasts, providing a seamless and personalized experience.