A logo with columns and stars

Description automatically generatedA black and grey logo with a book and a black hat

Description automatically generated with medium confidence**A logo of a building with columns

Description automatically generated**

**A REDUX – INSPIRED PODCAST APP WITH DYNAMIC THEMS FOR ANDROID**

**A PROJECT REPORT**

***Submitted by***

**ABINAYA. R (812022104003)**

**KAVIYA. K (812022104036)**

**KAVIYA. S (812022104038)**

**BACHELOR OF COMPUTER SCIENCE ENGINEERING**

**IN**

**FIFTH SEMESTER**

**COMPUTER SCIENCE ENGINEERING**

**M.A.M. COLLEGE OF ENGINEERING AND TECHNOLOGY, TRICHY**

**ANNA UNIVERSITY : CHENNAI 600 025**

**NOVEMBER 2024**

**TABLE OF CONTENTS**

**Chapter 1-Abstract**

**Chapter 2-Introduction**

**Chapter 3-Objevtives**

**Chapter 4-Description**

**Chapter 5-Features and Functionality**

**Chapter 6-Projectoverview**

**Chapter 7-Scope and key features**

**Chapter 8-source code and output**

**Chapter 9-Conclusion**

# **CHAPTER 1 – ABSTRACT**

Podcast Plus is an innovative Android app that redefines the podcast listening experience. Inspired by Redux, this app integrates dynamic themes, intuitive navigation, and personalized features. Built using Java/Kotlin, Podcast Plus leverages the Android Architecture Components and Redux principles to ensure scalability, maintainability, and seamless performance.

**Key Features:**

1. Dynamic Themes: Choose from various themes or create custom ones.

2. Podcast Discovery: Browse top charts, categories, and recommendations.

3. Personalized Feed: Subscribe, download, and manage episodes.

4. Offline Playback: Listen anywhere, anytime.

5. Multi-Tab Support: Organize podcasts into custom tabs.

6. Episode Notifications: Stay updated on new releases.

7. Search and Filtering: Quick access to specific podcasts or episodes.

# **CHAPTER 2 - INTRODUCTION**

Redux architecture has gained significant popularity in web development for managing state in applications efficiently. However, its principles can be applied to mobile app development as well, including Android development using Kotlin. In this blog post, we will explore Redux architecture and how it can be implemented in Android Kotlin applications to manage complex state effectively.

What is Redux Architecture? Redux is a predictable state container for JavaScript apps, but its core principles can be generalised to any programming language or platform. At its core, Redux follows three fundamental principles:

Single Source of Truth: The state of your whole application is stored in an object tree within a single store. This makes it easier to manage and debug the state of the application.

State is Read-Only: The only way to change the state is to emit an action, an object describing what happened. This ensures that the state transitions in your app are predictable and transparent.

Changes are made with Pure Functions: To specify how the state tree is transformed by actions, you write pure reducers. A reducer is a pure function that takes the previous state and an action, and returns the next state.

# **CHAPTER 3 - OBJECTIVES**

objective (Redux Toolkit is our official recommended approach for writing Redux logic. It wraps around the Redux core, and contains packages and functions that we think are essential for building a Redux app. Redux Toolkit builds in our suggested best practices, simplifies most Redux tasks, prevents common mistakes, and makes it easier to write Redux applications.

RTK includes utilities that help simplify many common use cases, including store setup, creating reducers and writing immutable update logic, and even creating entire "slices" of state at once

.

Whether you're a brand new Redux user setting up your first project, or an experienced user who wants to simplify an existing application, Redux Toolkit can help you make your Redux code better.)

**Technical Details:**

1. Programming Languages: Java, Kotlin

2. Architecture: Model-View-ViewModel (MVVM) with Redux-inspired state management

3. Android Components: LiveData, Room Persistence Library, Navigation Component

4. Third-Party Libraries: Retrofit, OkHttp, Glide

5. Database: SQLite with Room Persistence Library

**Code Structure:**

1. com.podcastplus (main package)

- data (models, database, API)

- domain (business logic, use cases)

- presentation (UI, views, adapters)

- redux (state management, actions, reducers)

**Example Code Snippet (Kotlin):**

// PodcastRepository.kt (data layer)

suspend fun getPodcasts(): List<Podcast> {

val response = apiService.getPodcasts()

return response.podcasts.map { it.toPodcast() }

}

// PodcastViewModel.kt (presentation layer)

fun loadPodcasts() {

viewModelScope.launch {

val podcasts = podcastRepository.getPodcasts()

\_podcasts.value = podcasts

}

}

API Documentation:

Available upon request.

License:

Open-source (MIT License).

Future Development:

1. Integrate machine learning-based recommendations.

2. Implement podcast analytics.

3. Enhance social sharing features.

Contact:

[Your Name]

[Your Email]

[Your GitHub Repository] )

# **CHAPTER 4 – DESCRIPTION**

**Podcast Plus: Enhance Your Listening Experience**

Discover, download, and enjoy your favorite podcasts with Podcast Plus, a feature-rich Android app inspired by Redux principles. Personalize your listening experience with dynamic themes, intuitive discovery, and seamless playback.

**Key Features:**

- Dynamic Themes: Choose from various themes or create custom ones

- Podcast Discovery: Browse top charts, categories, and recommendations

- Offline Playback: Listen anywhere, anytime

- Multi-Tab Support: Organize podcasts into custom tabs

- Episode Notifications: Stay updated on new releases

- Search and Filtering: Quick access to specific podcasts or episodes

**Designed for You:**

- User-friendly interface

- Customizable font sizes and styles

- Secure data storage and encryption

- Battery-efficient playback

**Get Started:**

1. Search for podcasts or browse top charts

2. Subscribe and download episodes

3. Customize your theme and settings

4. Enjoy seamless playback and offline listening

**Download Now:**

[Google Play Store Link]

**System Requirements:**

- Android 10+

- 4.4+ rating

- 50MB storage space

**Support:**

Email: mailto:support@podcastplus.app

Twitter: @PodcastPlusApp

Website: podcastplus.app

**Join the Community:**

Share your favorite podcasts and themes with fellow listeners on social media using #PodcastPlus.

# **CHAPTER 5 -FEATURES AND FUNCTIONALITIES**

**Project Overview: Podcast Plus**

**Project Type: Android App**

Category: Podcast Player

Tagline: "Enhance your podcast experience with dynamic themes and intuitive discovery"

Project Goal:

Create a feature-rich, user-friendly podcast app for Android, leveraging Redux-inspired state management and dynamic themes to provide a personalized listening experience.

Key Objectives:

1. Develop a scalable and maintainable architecture using Android Architecture Components and Redux principles.

2. Design an intuitive user interface with dynamic themes and customizable options.

3. Implement robust podcast discovery features, including top charts, categories, and recommendations.

4. Ensure seamless playback, offline support, and automatic episode management.

5. Provide secure data storage and encryption for offline episodes.

# 

# **CHAPTER 6 - PROJECT OVERVIEW**

The Adaptive Mail project is aimed at developing a cutting-edge, user-friendly email client that addresses common challenges faced by email users today. As email remains one of the most important forms of communication, both for personal and business purposes, there is a growing need for innovative tools that can simplify email management, improve efficiency, and enhance security. While traditional email clients offer basic functions like sending and receiving emails, there is a significant gap in the market for an email client that is flexible, intelligent, and capable of personalizing the user experience to meet the demands of modern users.

## **Purpose and Motivation**

The core purpose of Adaptive Mail is to bridge this gap by providing an email client that adapts to the needs of its users. The main goal is to offer a flexible, intelligent email management system that can be personalized and optimized for every individual, whether they are a business professional, a student, or a freelancer. Traditional email clients can be cumbersome to manage, especially for users dealing with large volumes of emails. Emails often get lost in cluttered inboxes, making it time-consuming and difficult to prioritize and respond to important messages. Adaptive Mail solves this problem by leveraging artificial intelligence (AI) and machine learning to automatically categorize and prioritize emails, filter spam, and provide intelligent suggestions for responses. Additionally, the app is designed with a focus on security, privacy, and customization, ensuring that users have full control over how they interact with their emails.

## **Target Audience**

Adaptive Mail caters to a wide range of users, including:

* **Business Professionals**: Those who deal with large volumes of emails daily, requiring a more organized system to prioritize work emails, automate responses, and maintain an efficient workflow.
* **Freelancers**: Individuals who need a flexible email management system that integrates with productivity tools like calendars, task managers, and cloud storage services.
* **Students**: Those looking for an easy-to-use and secure email client that can help them manage educational communication, assignments, and personal emails without the complexity of traditional systems.
* **General Users**: Anyone who values simplicity, security, and customization when managing their personal or professional emails.

The app is designed to help users maintain an organized and clutter-free inbox, improve email response times, and ultimately increase productivity.

# **CHAPTER 7 – SCOPE AND KEY FEATURES**

**Podcast Plus Features and Functionality**

**Core Features**

1. Podcast Discovery: Browse top charts, categories, and recommendations.

2. Personalized Feed: Subscribe, download, and manage episodes.

3. Offline Playback: Listen anywhere, anytime.

4. Dynamic Themes: Choose from various themes or create custom ones.

5. Multi-Tab Support: Organize podcasts into custom tabs.

6. Episode Notifications: Stay updated on new releases.

7. Search and Filtering: Quick access to specific podcasts or episodes.

**Playback Features**

1. Play, Pause, and Resume episodes.

2. Skip forward and backward (30-second increments).

3. Speed adjustment (0.5x, 1x, 1.5x, 2x).

4. Volume control and mute.

5. Automatic playback of next episode.

**Library and Organization**

1. Podcast library with customizable sorting.

2. Episode archive with filtering options.

3. Favorites and downloads management.

4. Automatic episode deletion (optional).

**Discovery and Recommendations**

1. Top Charts: Browse popular podcasts.

2. Categories: Explore podcasts by genre.

3. Recommendations: Based on listening history.

4. Podcast suggestions: Related to subscribed podcasts.

**Settings and Customization**

1. Dynamic Themes: Choose from pre-made or create custom.

2. Font size and style adjustment.

3. Notification preferences.

4. Offline storage location selection.

5. Automatic updates and downloads.

**Social Sharing and Integration**

1. Share podcast episodes on social media.

2. Copy episode links.

3. Email podcast recommendations.

**Notifications and Alerts**

1. New episode notifications.

2. Download completion notifications.

3. Error notifications (e.g., network issues).

**Analytics and Feedback**

1. Podcast engagement tracking.

2. User feedback and ratings.

3. Crash reporting.

**Technical Features**

1. Redux-inspired state management.

2. Android Architecture Components.

3. Room Persistence Library for database management.

4. Retrofit and OkHttp for API calls.

5. Glide for image loading.

**Security Features**

1. Secure podcast data storage.

2. Encryption for offline episodes.

3. Secure API requests.

**Accessibility Features**

1. Screen reader support.

2. High contrast theme.

3. Font size adjustment.

**Future Development**

1. Machine learning-based recommendations.

2. Podcast analytics.

3. Enhanced social sharing features.

4. Integration with popular music streaming services.)

# **CHAPTER 8 – SOURCE CODE AND OUTPUT**

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 = ""

)

}

}

}

}

}

}

}

# **Output**



**Figure 1. Project Output**

# **CHAPTER 9– CONCLUSION**

**A sample podcast app that features a full-featured, Redux-style architecture and showcases dynamic themes.**

**Project Description**

**Project Description:**

A project that demonstrates the use of Android Jetpack Compose to build a UI for a podcast player app. The app allows users to choose , play and pause podcasts.

**Architecture**

**Learning Outcomes :**

**By end of this project:**

You’ll be able to work on Android studio and build an app.

You’ll be able to integrate the database accordingly.

**Project Workflow:**

Users register into the application.

After registration , user logins into the application.

User enters into the main page

The app allows users to choose , play and pause podcasts.

**Note:**

To complete the project you need to finish up the tasks listed below:

**Tasks:**

1.Required initial steps

2.Creating a new project.

3.Adding required dependencies.

4.Creating the database classes.

5.Building application UI and connecting to database.

6.Modifying AndroidManifest.xml

7.Running the application.)

# **REFERENCES**

This project is overviewed by Github Google