PROJECT TITLE

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

DESIGNED BY:

SRI PARAMAKALYANI COLLEGE (CODE-123), ALWARKURUCHI-627412

DEPARTMENT OF COMPUTER SCIENCE

MENTOR

Dr.T.Arul Raj M.SC., M.Phil., Ph.d

TEAM

TEAM LEADER

C.ROHITH RAGAVENDER (20201231506231)

TEAM MEMBERS

M.PARAMASIVAN (20201231506227)

M.SURIYA PRAKASH(20201231506242)

S.SUDHAN BOOPATHIRAJ(20201231506237)

PROJECT INDEX

CHAPTER-I INTRODUCTION:

1.1 Overview

A podcast is a program made available in digital format for download over the Internet. For example, an episodic series of digital audio files that a user can download to a personal device to listen to at a time of their choosing. Podcasts are primarily an audio medium, with some programs offering a supplemental video component. Streaming applications and podcasting services provide a convenient and integrated way to manage a personal consumption queue across many podcast sources and playback devices. There are also podcast search engines, which help users find and share podcast episodes.

A podcast series usually features one or more recurring hosts engaged in a discussion about a particular topic or current event. Discussion and content within a podcast can range from carefully scripted to completely improvised. Podcasts combine elaborate and artistic sound production with thematic concerns ranging from scientific research to slice-of-life journalism. Many podcast series provide an associated website with links and show notes, guest biographies, transcripts, additional resources, commentary, and occasionally a community forum dedicated to discussing the show's content.

The cost to the consumer is low, with many podcasts free to download. Some podcasts are underwritten by corporations or sponsored, with the inclusion of commercial advertisements. In other cases, a podcast could be a business venture supported by some combination of a paid subscription model, advertising or product delivered after sale. Because podcast content is often free, podcasting is often classified as a disruptive medium, adverse to the maintenance of traditional revenue models.

- **Home: With a hard-coded list of podcasts and an integrated search field.The screen adapts automatically depending on whether the TextField is empty or not.**
- **Favourite: Display liked podcasts and a button for each podcast to unlike it.**
- **Podcast: Displays information and the episodes list of a particular podcast.**
- **Episode: Display information about the episode, a dynamic slider that changes the duration synchronously, and other non-functional UI Buttons(Inspired from an existing UI)**

1.2 Purpose:

There are many uses for podcasting for the classroom. They can be used to convey instructional information from the teacher or trainer, motivational stories, and auditory case studies. Podcasts can also be used by the learners as artifacts and evidence of learning; for example, a student might prepare a brief podcast as a summary of a concept in lieu of writing an essay. Podcasts can also be used as a means of self-reflection on the learning processes or products. Podcasts can help keep students on the same page, including those that are absent. Absent students can use podcasts to see class lectures, daily activities, homework assignments, handouts, and more. A review of literature that reports the use of audio podcasts in K-12 and higher education found that individuals use existing podcasts or create their own podcasts.

According to Jonathan Copley, many students choose to use podcasts as a supplement to lecture materials. Before classes, students use podcasts to gain an overall understanding of the upcoming lecture, which makes them feel more confident and much more prepared for the class. The use of podcasts better prepares students for classes and promotes discussions. The download of podcasts peaks both immediately after a podcast has been uploaded and right before examinations or deadlines.Students use podcasts as part of their review for exams because it provides different methods of reinforcement of course material. This includes visual reinforcement of material, and testing of their knowledge base, and adding variety to the review experience. In addition, students who missed the lecture because of sickness or other reasons can use podcasts to catch up on their notes. Students learn better when they have a teacher present the materials, rather than going over other people's notes. Finally, students with disabilities and students who do not speak English as their first language use podcasts because they can listen to the material repeatedly.

According to Robin H. Kay, there are five key benefits regarding the use of video podcasts for students.

1. Students can control the pace of their own studies
2. Increase in motivation
3. Improvement in study habits
4. Positive impact on testing skills
5. Does not reduce class attendance

PODCASTS IN HIGHER EDUCATION

The use of podcasts for the purpose of education, whether by professional educators or amateurs, has grown to the point that an annual conference for educational podcasters called Sound

Education took place (although it was canceled during the COVID-19 pandemic).

Mobile Learning: Podcasting can be categorized as an m-learning strategy for teaching and learning. In 2004, Mussel burgh Grammar School pioneered podcast lessons with foreign language audio revision and homework. In the second half of 2005, a Communication Studies course at the University of Western Australia used *student-created podcasts* as the main assessment item. In 2005, "Students in the Write" was created for second-grade students at Morse Elementary School in Tarrytown, NY. On 21 February 2006, Lance Anderson, Dr. Chris Smith, Nigel Paice, and Debbie McGowan took part in the first podcast forum at Cambridge University. The event was hosted by the Centre for Applied Research in Educational Technologies.

Mobile Knowledge Transfer: Podcasting is also used by corporations to disseminate information faster and more easily. It can be seen as a further development of Rapid E-Learning as the content can be created fast and without much effort. Learners can learn in idle times which save time and money for them and the organizations. Audio podcasts can be used during other activities like driving a car, travelling by train, or riding a bus. A group often targeted is the sales-force, as they are highly mobile. There podcasting can be used for sales enablement.

Mathematical Learning: Audio-podcasts can also be used in mathematics education. With the recording of mathematical audio-podcasts, oral communication and representation are focused on. Audio-podcasts have been used in primary school as well as in teacher education. The process of producing the mentioned audio-podcasts in mathematics education facilitates reflection processes.

Scientific Learning: Podcasting is an emerging tool with a broad flexibility to deliver science-based information asynchronously in the online classroom or in online outreach programming.

Journalism Education: School podcasts can be created to expose students to journalism and new-media concepts. Regularly released "news" podcasts can be released by a school group.

Academic Journal Digests: The Society of Critical Care Medicine has a podcast used to update clinicians with summaries of important articles, as well as interviews.

Supply Chain Management Education: In October 2007, Stephan Brady presented his paper on "Podcasting in Supply Chain Education" at the CSCMP Educators Conference, which outlined how podcasting could be used in and outside of the classroom for

enhancing supply chain courses through blended, or hybrid learning.

Anxiety: Podcasts have been used to solve problems with college students' anxiety by allowing professors' lectures to be accessed after class so the students would not have to worry about missing any of the material if absent or tardy. According to Anthony Chan & Mark J.W. Lee, "The advent of consumer-level digital multimedia hardware and software have prompted the more techno-logically inclined instructors and educational designers to construct CD-ROM based re-sources to engage and excite students using the richness and flexibility of text, graphics, sound, video, animation and interactive content, as well as the combination of these elements".

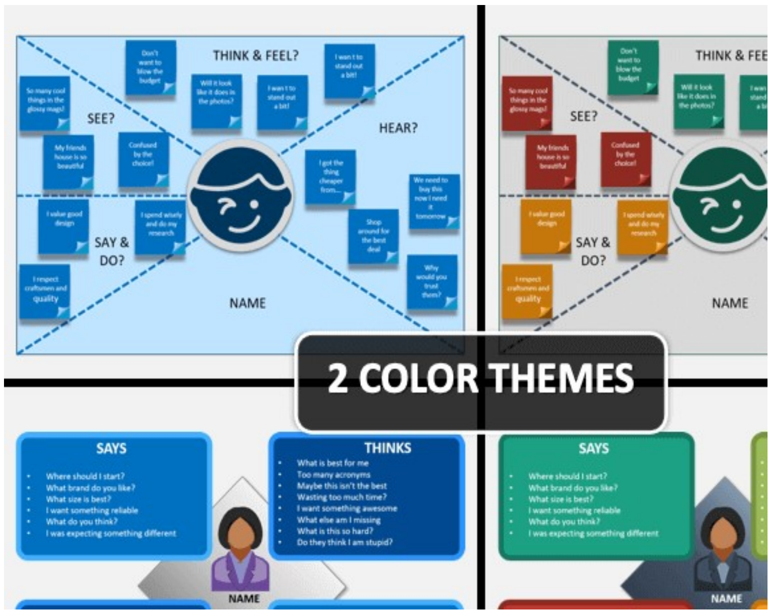
PUBLIC SERVICES

Podcasts have been used for advocacy. The 5,500 locked out staff of the Canadian Broadcasting Corporation were podcasting news and other programming during August and September 2005.

CHAPTER – II PROBLEM DEFINITIONS & DESIGN THINKING

2.1 Empathy Map

An empathy map is a widely-used visualization tool within the field of UX and HCI practice. In relation to empathetic design, the primary purpose of an empathy map is to bridge the understanding of the end user. Within context of its application, this tool is used to build a shared understanding of the user's needs and provide context to a user-centered solution.



STRUCTURE

The traditional empathy map begins with four categories: says thinks, does, and feels. At the center of the map, a user or persona is displayed to remind practitioners and stakeholders what type of individual this research is centered around. Each category of the empathy map represents a snapshot of the user's thoughts and feelings without any chronological order.

- Says category contains what the user says out loud during research or testing. Ideally, each point is written down as close to the users original words as possible.
- Thinks category contains what the user is thinking. While content may overlap with the *Says* category, thinks category exists to capture thoughts users may not want to share willing due to social factors, such as self-consciousness or politeness.
- Does category contains the user's action and behaviors. This contains what the user is physically doing and captures what actions users are taking.
- Feels category contains the user's emotional state in context with their experience. This typically contains information or phrases as to how they feel about the experience.

However, as time evolved, the empathy map has been updated to provide more context and information architecture within the industry.

Empathy maps could vary in forms, but they have common core elements. Other than the four traditional categories mentioned above, empathy map could also include other categories. Here are two other categories commonly used:

- See category contains information users observed through eyes. It could be what users see in the marketplace or in the immediate environment, other people's saying and doing, or the content they watch or read.
- Hear category is what user hears and how that impacts the user. It could be personal connections as well as other recourses such as media. Instead of documenting superficial information streams, team should focus on details that influence the user.

2.2 Ideation & Brainstorming Map

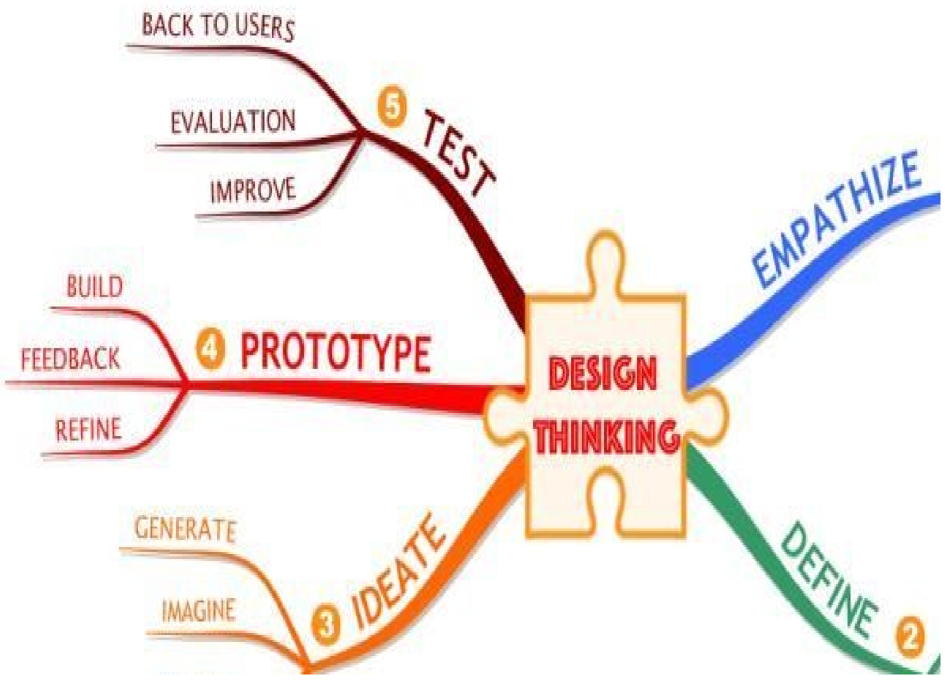
Ideation:

Ideation is the creative process of generating, developing, and communicating new ideas, where an idea is understood as a basic element of thought that can be either visual, concrete, or abstract. Ideation comprises all stages of a thought cycle, from innovation, to development, to actualization. Ideation can be conducted by individuals, organizations, or crowds. As such, it is an essential part of the design process, both in education and practice.

Brainstorming:

Brainstorming is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members.

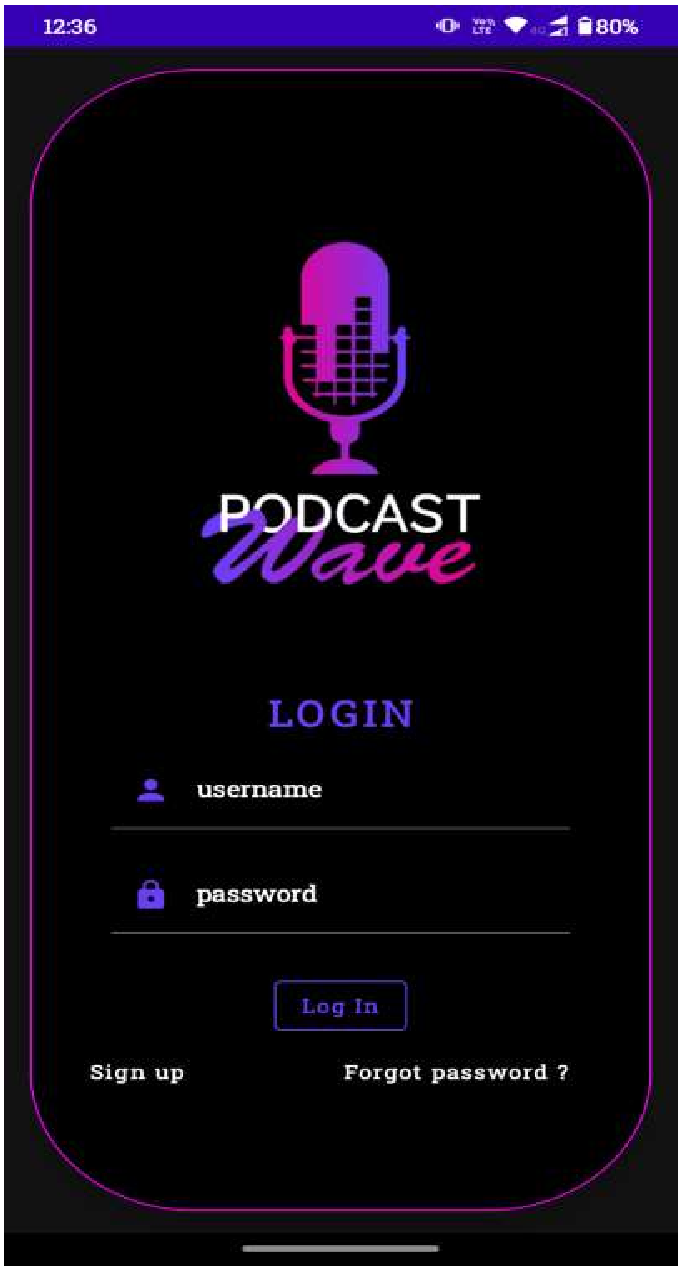
In other words, brainstorming is a situation where a group of people meet to generate new ideas and solutions around a specific domain of interest by removing inhibitions. People are able to think more freely and they suggest as many spontaneous new ideas as possible. All the ideas are noted down without criticism and after the brainstorming session the ideas are evaluated.



CHAPTER – III RESULT

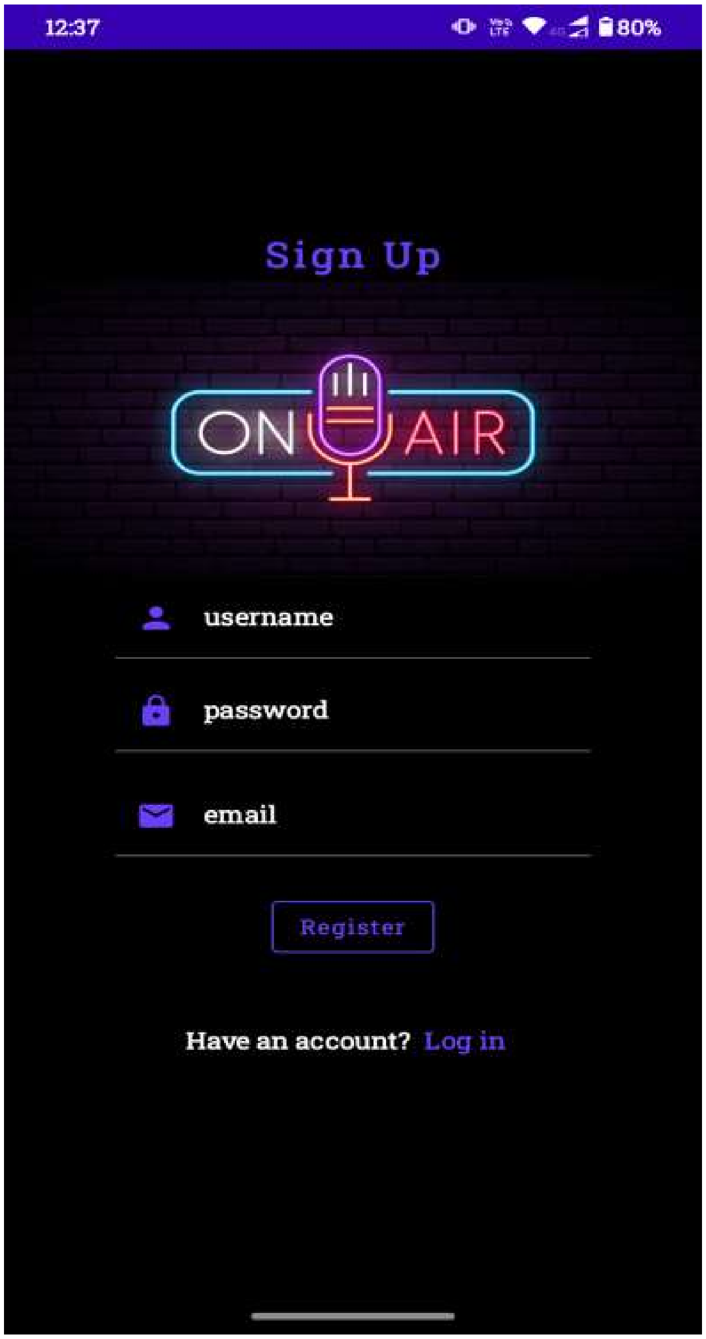
LOGIN PAGE

AFTER REGISTRATION, USER LOGIN INTO APPLICATION



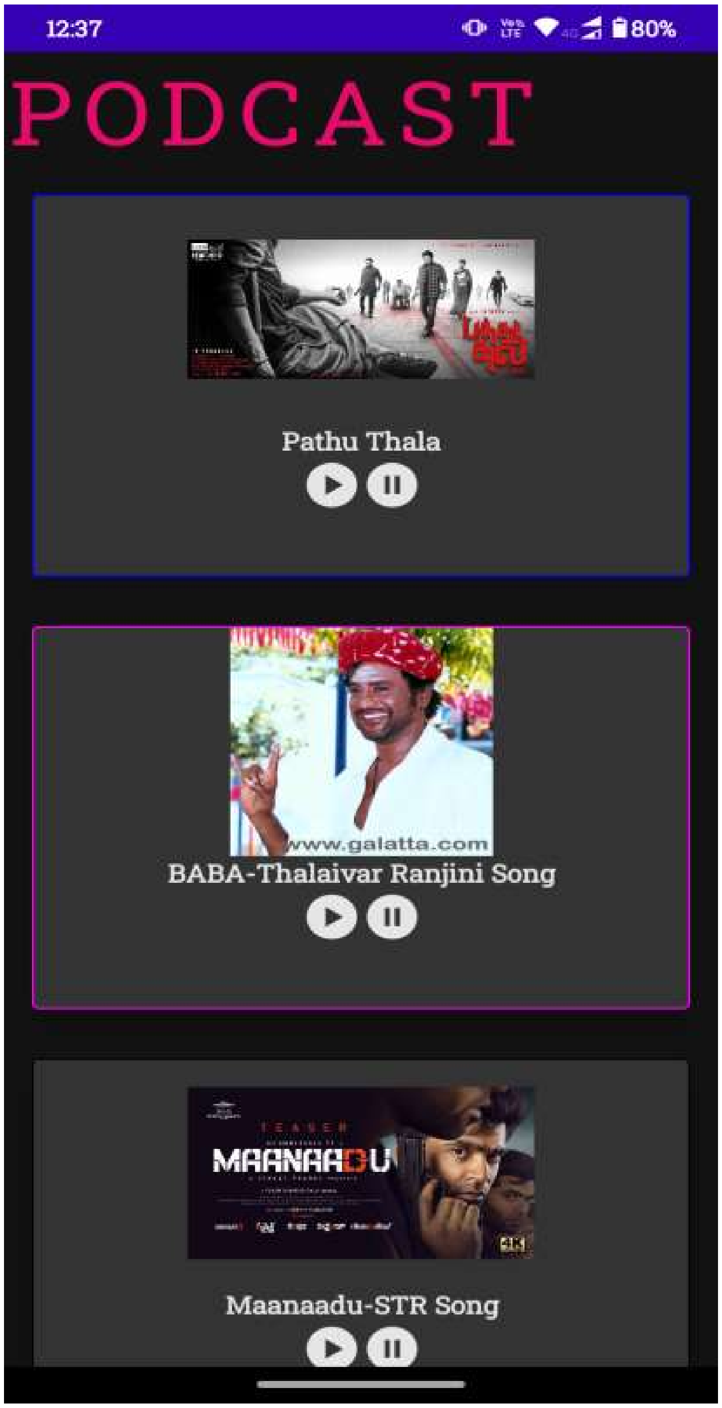
REGISTER PAGE

USER REGISTER INTO APPLICATION



MAIN PAGE

USER ENTERS INTO THE MAIN PAGE



CHAPTER - IV ADVANTAGES & DISADVANTAGES

4.1 Advantage

Dynamic Themes templates are abundant on the internet.You just need to be as descriptive as possible with what you need and what you are looking for.

After that you need to choose the template that you like the most and place the information .It is fast to get your Dynamic Themes UI and running as it may take even hours.

You just need to make some additions such as the image you want to use and insert everything else you want to publish

They are easy to use

A template can be used by any simple mortal that does not necessarily have Dynamic Themes design or programming knowledge

You just need to spend time adapting it for your needs if you want to add additional elements like for

examples,galleries,forms,social media buttons,action buttons among others.Usually templates are design friendly for you to drag and drop the elements where you need .

You have a variety of style to choose form

Now days, the templates available in the market comply with the industry standards, meaning that the design is attractive and made for specific business lines.

Remember that you will need a different template if you are building an

E-commerce or if you just need one page to display important information.

Some template providers even give you trail time for you to test if their template suits you according to your style and needs

Making a podcast is fairly simple. All you need is a computer or smartphone, software to record your presentation and a connection to upload it to the Internet. If you're looking to reach a wide audience, there are multiple sites that make podcasts available to listeners, including iTunes

4.2 Disadvantages of Dynamic Themes

They lack originality

Even when the information you provide may be original, the template will be not. Other companies and businesses may be using the same one and for this reason you will have to double your efforts to differentiate from the rest

They lack User experience

Templates are certainly beautiful, but the disadvantages is that they do not take into consideration the user experience and everything that this involves.

Your buyer person needs to feel that the Dynamic Themes was created for him or her and be reflected in it.

When you use a template, it is built in generic form and it lacks the pleasure, efficiency and fun that should be built based on the target users.This may be something negative as your buyer personas may not take the actions that you want them to take that may lead them to buying decision

You will have to make a direct change in the coding and in order to do this you need to have a basic knowledge in programming.

As you see, Dynamic Themes templates have their advantages and disadvantages, and you have to choose based on your requirements like time,budget and specific needs. The most important thing is to provide a user-friendly Dynamic Themes where your potential customers feel comfortable visiting.

CHAPTER - V APPLICATIONS

One of the benefits of podcasts is that you don't have to worry that your audience is busy offline when you make your sales pitch. A listener can download or stream the file and replay it at whatever time is convenient for him, even while jogging or driving to work. If someone chooses to subscribe to your podcast feed, he can get any podcasts of interest downloaded automatically. If you want all your employees to hear what you have to say, podcasting is often easier than trying to get everyone together for one meeting. Making a podcast is fairly simple. All you need is a computer or smartphone, software to record your presentation and a connection to upload it to the Internet. If you're looking to reach a wide audience, there are multiple sites that make podcasts available to listeners, including iTunes. You can buy software packages that also enable you to edit podcasts, add sound effects or

store past podcasts for you to use later. Editing improves the quality of your finished podcast, but it does take more work.

CHAPTER – VI CONCLUSION

This shouldn't seem like a lot of red tape, or, something that will restrict your creative license. On the contrary, the job of your podcast intros and outros is to support and enhance the actual content of your episodes.

You don't need to overthink them or spend much time on them. It's just about putting some good practices in place that make sure you're not needlessly losing listeners, and that you're always steadily growing your audience.

Over time, the things you want to include at the beginning and end of your episode will become second nature to you. Your main focus should always be on your topic.

Hopefully, that's given you a good idea of how you can make some improvements to your episodes though!

CHAPTER-VII FUTURE SCOPE

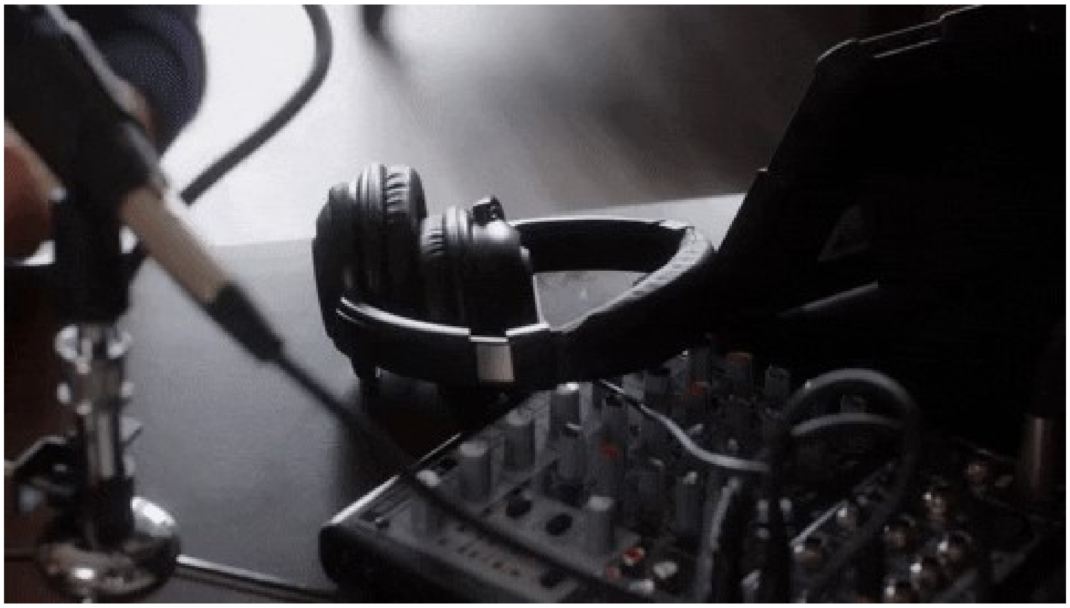
SOUND ENGINEERS

Guaranteeing the sound and right feeling is finished by sound specialists on the sound altering table. As podcasts are totally subject to the vehicle of sound, makers need to put a greater part of their emphasis on getting the sound right. Here, the sound specialists sparkle. They ensure the eventual outcome is at its closest to perfect for the listeners.

From eliminating background noise, and refining the podcaster's sound, to giving the perfect right musical cues at the right time, the sound engineers make the podcast what it is. Guaranteeing the sound and right emotion is finished by sound engineers on the sound altering table. They push their listeners to rejuvenate their accounts by the medium of sound.

PODCAST CONTENT CURATORS

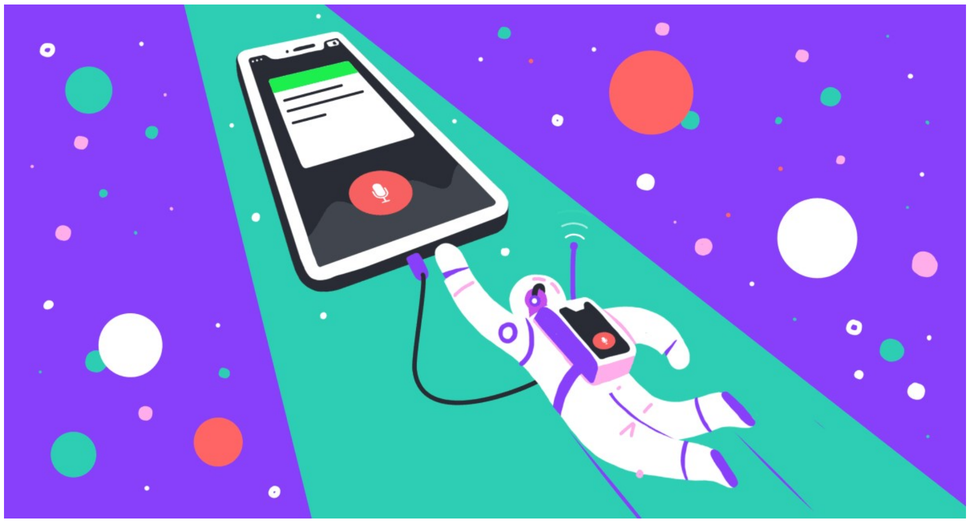
The Content Curators help the hosts and podcasts writers structure their accounts, and when vital assistance curates a specific section for the equivalent. Podcasts are on a par with the substance they present to their audience members. Subsequently, content curators assume a critical part as they help choose how the topic should be introduced. They measure the profundity of the story and comprehend the potential it is able to do.



The content curators help the hosts and the writers structure their accounts, and when important help curates a particular fragment for the same. They help in choosing whether the topic can be enveloped with one podcast or in the event that it ought to be ventured into an overarching series. It is an energizing position, as curators have an enormous stake in being the storytellers as well.

PODCAST PRODUCERS

Podcast Producers settle on an ultimate conclusion on the podcasts and help choose the correct target audience for every one of them. There are various genres of podcasts recordings and its maker is responsible for each part of it. They basically play out the job of a creative product manager, taking care of everything from the tone of the show, to how it ought to be advertised and to whom.



They settle on an official conclusion on the Podcast and help choose the correct objective crowds for every one of them. As podcasting is as yet in its early stages in the nation, the Podcast maker's job is critical as they are responsible for ensuring the podcast is totally prepared for listenership. It is additionally an exceptionally rewarding field right now as most makers earn more money than it has.These is common ideas I see across all conversations currently happening about the future (Currently being the last 60 years or so.)

1. Our guesses about the future will be wrong

2. Change will happen quicker than we think

3. The speed of changes is increasing (The next 150 years will change more than the last)

4. Our guess at the future is a reflection of what we find important now
5. We are in an important turning point is history , humans are globally connected and augmented by machines ,and we have access to more knowledge than ever

CHAPTER-VIII APPENDIX

SOURCECODE:

https://github.com/RohithRagavender/Podcast2.git

LOGIN ACTIVITY.KT

```
package com.example.podcasteapplication

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.podcasteapplication.ui.theme.PodcasteApplicationT
heme

import com.example.podcasteapplication.MainActivity

import com.example.podcasteapplication.RegistrationActivity
```

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

MAIN ACTIVITY.KT

```
package com.example.podcasteapplication
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.Composable

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.podcasteapplication.ui.theme.PodcasteApplicationT
heme
```

```
class MainActivity : ComponentActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContent {
      PodcasteApplicationTheme() {
        // 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(0xFFE60B6A),
            fontWeight = FontWeight.Black,
            fontSize = 67.sp,
            style = MaterialTheme.typography.h1,
            letterSpacing = 0.1.em
         )
      }
      Column(modifier = Modifier
         .fillMaxSize()
```

```
        .verticalScroll(rememberScrollState())) {
        Card(
            elevation = 12.dp,
            border = BorderStroke(1.dp, Color.Blue),
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth()
                .height(250.dp)
        )
        {
            val mp: MediaPlayer = MediaPlayer.create(context,
R.raw.audio_10)

            Column(
                modifier = Modifier.fillMaxSize(),
                horizontalAlignment = Alignment.CenterHorizontally
            ) {

                Image(
            painter = painterResource(id = R.drawable.img_7),
            contentDescription = null,
            modifier = Modifier
              .height(150.dp)
              .width(200.dp),
            )
          Text(
            text = "Pathu Thala-STR MOIVE",
            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_11)
      Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally
      ) {
        Image(
          painter = painterResource(id = R.drawable.img_6),
          contentDescription = null,
          modifier = Modifier
            .height(150.dp)
            .width(200.dp)
        )

        Text(
          text = "BABA-Thalaivar Ranjini Song",
          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.DarkGray),
        modifier = Modifier
            .padding(16.dp)
            .fillMaxWidth()
            .height(250.dp)
    )
    {
        val mp: MediaPlayer = MediaPlayer.create(context,
R.raw.audio_12)
```

```
Column(
   modifier = Modifier.fillMaxSize(),
   horizontalAlignment = Alignment.CenterHorizontally

){

   Image(
      painter = painterResource(id = R.drawable.img_12),
      contentDescription = null,
      modifier = Modifier
         .height(150.dp)
         .width(200.dp)
   )

   Text(
      text = "Maanaadu-STR Song",
      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.White),
        modifier = Modifier
            .padding(16.dp)
            .fillMaxWidth()
            .height(250.dp)
    )
    {
        val mp: MediaPlayer = MediaPlayer.create(context,
R.raw.audio_7)
```

```
Column(
  modifier = Modifier.fillMaxSize(),
  horizontalAlignment = Alignment.CenterHorizontally
) {

  Image(
    painter = painterResource(id = R.drawable.img_10),
    contentDescription = null,
    modifier = Modifier
      .height(150.dp)
      .width(200.dp),

  )

  Text(
    text = "LEO-THALAPATHY VIJAY",
    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.Red),
           modifier = Modifier
              .padding(16.dp)
              .fillMaxWidth()
              .height(250.dp)
        )
        {
           val mp: MediaPlayer = MediaPlayer.create(context,
R.raw.audio_8)

           Column(
              modifier = Modifier.fillMaxSize(),
              horizontalAlignment = Alignment.CenterHorizontally
           ) {
              Image(
                 painter = painterResource(id = R.drawable.img_9),
                 contentDescription = null,
                 modifier = Modifier
                     .height(150.dp)
                     .width(200.dp),
                 )
              Text(
                 text = "DADA-KAVIN KUMAR LOVE SONG",
                 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.Green),
    modifier = Modifier
        .padding(16.dp)
        .fillMaxWidth()
        .height(250.dp)
)
{
    val mp: MediaPlayer = MediaPlayer.create(context,
R.raw.audio_9)
```

```
        Column(
          modifier = Modifier.fillMaxSize(),
          horizontalAlignment = Alignment.CenterHorizontally
        ) {
          Image(
            painter = painterResource(id = R.drawable.img_8),
            contentDescription = null,
            modifier = Modifier
              .height(150.dp)
              .width(200.dp),
)
          Text(
            text = "DADA-KAVIN KUMAR LOVE BETWEEN
FATHER AND SON  ",
            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 = ""
              )
            }
          }
        }
      }
```

REGISTRATION ACTIVITY.KT

```
package com.example.podcasteapplication

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.podcasteapplication.LoginActivity

import com.example.podcasteapplication.UserDatabaseHelper

import
com.example.podcasteapplication.ui.theme.PodcasteApplicationT
heme

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

```
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    databaseHelper = UserDatabaseHelper(this)
    setContent {
      PodcasteApplicationTheme() {
        Surface(
          modifier = Modifier.fillMaxSize(),
          color = MaterialTheme.colors.background
        ) {
          RegistrationScreen(this,databaseHelper)
        }
      }
    }
  }
}
@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
  var username by remember { mutableStateOf("") }
```

```
 var password by remember { mutableStateOf("") }

 var email by remember { mutableStateOf("") }

 var error by remember { mutableStateOf("") }

Column(

    Modifier

        .background(Color.Black)

        .fillMaxHeight()

        .fillMaxWidth(),

    horizontalAlignment = Alignment.CenterHorizontally,

    verticalArrangement = Arrangement.Center

 )

 {

    Row {

        Text(

            text = "Sign Up",

            color = Color(0xFF6a3ef9),

            fontWeight = FontWeight.Bold,

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

            letterSpacing = 0.1.em
```

```
  )
}
Image(
  painter = painterResource(id = R.drawable.podcast_signup),
  contentDescription = ""
)
TextField(
  value = username,
  onValueChange = { username = it },
  leadingIcon = {
    Icon(
      imageVector = Icons.Default.Person,
      contentDescription = "personIcon",
      tint = Color(0xFF6a3ef9)
    )
  },
  placeholder = {
    Text(
      text = "username",
            color = Color.White
        )
    },
    colors = TextFieldDefaults.textFieldColors(
        backgroundColor = Color.Transparent
    )
)
Spacer(modifier = Modifier.height(8.dp))
TextField(
    value = password,
    onValueChange = { password = it },
    leadingIcon = {
        Icon(
            imageVector = Icons.Default.Lock,
            contentDescription = "lockIcon",
            tint = Color(0xFF6a3ef9)
        )
    },
            placeholder = { Text(text = "password", color = Color.White)
},
            visualTransformation = PasswordVisualTransformation(),
            colors =
TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
        )
        Spacer(modifier = Modifier.height(16.dp))
        TextField(
            value = email,
            onValueChange = { email = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Email,
                    contentDescription = "emailIcon",
                    tint = Color(0xFF6a3ef9)
                )
            },
            placeholder = { Text(text = "email", color = Color.White) },
            colors =
TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
        )
        Spacer(modifier = Modifier.height(8.dp))
        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }
        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()
&& email.isNotEmpty()) {
                    val user = user(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
                    databaseHelper.insertUser(user)
                    error = "User registered successfully"
                    // Start LoginActivity using the current context
                    context.startActivity(
                        Intent(
                            context,
                            LoginActivity::class.java
                        )
                    )
                } else {
                    error = "Please fill all fields"
                }
            },
            border = BorderStroke(1.dp, Color(0xFF6a3ef9)),
            colors = ButtonDefaults.buttonColors(backgroundColor =
    Color.Black),
            modifier = Modifier.padding(top = 16.dp)
        ) {
            Text(text = "Register",
                fontWeight = FontWeight.Bold,
                color = Color(0xFF6a3ef9)
            )
        }
        Row(
            modifier = Modifier.padding(30.dp),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center
        ) {
            Text(text = "Have an account?", color = Color.White)
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                LoginActivity::class.java
              )
            )
          })
          {
            Text(text = "Log in",
              fontWeight = FontWeight.Bold,
              style = MaterialTheme.typography.subtitle1,
              color = Color(0xFF6a3ef9)
            )
          }


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

USERDAO.KT

```
package com.example.podcasteapplication

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)

}
```

USER DATABASE.KT

```
package com.example.podcasteapplication

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 DATABASEHELPER.KT

```
package com.example.podcasteapplication
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_NA
ME)),

        lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),

        email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),

        password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),

      )

    }
```

```
        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_NA
ME)),
            lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
            email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
         )
      }
      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_NA
ME)),
                    lastName = 
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NA
ME)),
                    email = 
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                   password = 
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWOR
D)),
                )
               users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()

        return users

    }

}

}

}

}
```

USER.KT

```
package com.example.podcasteapplication

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey

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?,
)
```