



L OVELY
P ROFESSIONAL
U NIVERSITY

Transforming Education Transforming India

CA PROJECT REPORT

On

LOOPER MAN

Submitted By

Lakshya Choudhary

Registration No: 12002824

Section: KO202

Course Code: CSE227

Roll No: 12

Programme Name: B.Tech (CSE)

Under The Guidance Of

Shubhita Ma'am

School Of Computer Science and Engineering

Lovely Professional University

DESCRIPTION

"Looper Man" is your ultimate companion in the quest for finding talented music producers across India. This innovative Android app offers a seamless platform connecting music enthusiasts with skilled producers, fostering creativity and collaboration in the vibrant Indian music scene. "Looper Man" is not just a directory; it is a dynamic community hub pulsating with energy and passion for music.

FEATURES COVERED

1. **Firestore Authentication:** "Looper Man" prioritizes the security and privacy of its users with Firestore Authentication. Safeguard your account with robust authentication methods, including email/password login, phone number verification, and social media sign-in options. Enjoy peace of mind knowing that your personal information and interactions within the app are protected by industry-leading security measures.
2. **Producer Location:** Explore the geographical landscape of music production in India with the "Looper Man" app's integrated Maps feature. Discover producers near you or in your desired location, visually represented on an interactive map interface. View detailed profiles, studio addresses, and contact information directly within the map view, making it easy to connect with producers in your area and beyond.
3. **Producers List in Shared Preferences:** Seamlessly access your favourite producers and preferred collaborators with the convenience of Shared Preferences in "Looper Man." Save your selected producers to a personalized list stored locally on your device, ensuring quick and easy access to their profiles and contact details whenever inspiration strikes. Organize your list based on genres, specialties, or project requirements, streamlining your creative workflow and fostering productive collaborations.
4. **Book Appointments:** Streamline the scheduling process and book recording sessions, meetings, and consultations with producers directly through the "Looper Man" app. Browse producers' availability calendars, select desired time slots, and confirm appointments with just a few taps. Receive instant notifications and reminders to stay on track with your scheduled appointments, ensuring efficient coordination and effective time management for your music projects.

ABSTRACTION

In the heart of India's bustling music scene lies a transformative force, poised to redefine the very essence of music production. "Looper Man" emerges as a beacon of innovation, a groundbreaking Android application meticulously designed to democratize and revolutionize the landscape of music creation across the nation. At its core, "Looper Man" embodies a visionary abstraction, a testament to the convergence of technology, community, and creativity in shaping the future of Indian music.

With an unwavering commitment to accessibility and inclusivity, "Looper Man" transcends barriers, welcoming musicians of all backgrounds and skill levels into its vibrant ecosystem. It serves as a conduit, seamlessly connecting aspiring artists with a diverse array of skilled producers, forging bonds that transcend geographical boundaries and cultural divides. Through its intuitive interface and comprehensive features, "Looper Man" empowers users to navigate the intricate tapestry of music production with confidence and ease.

Central to the ethos of "Looper Man" is the celebration of diversity and the amplification of voices that have long been marginalized within the music industry. By providing a platform where artists from every corner of India can share their unique perspectives and cultural heritage, "Looper Man" becomes a catalyst for social change, fostering a more inclusive and representative music landscape.

Moreover, "Looper Man" stands as a testament to the transformative power of collaboration and community. Through its dynamic forums, collaborative projects, and educational resources, "Looper Man" cultivates a thriving community of artists, producers, and enthusiasts united by their passion for music. Here, knowledge is freely exchanged, mentorship flourishes, and creativity knows no bounds, ushering in a new era of artistic expression and innovation.

Fuelling the engine of "Looper Man" is its unwavering dedication to technological advancement and user experience excellence. Leveraging cutting-edge features such as Firebase Authentication, real-time mapping of producer locations, and seamless appointment booking functionalities, "Looper Man" sets a new standard for user-centric design and functionality in music production applications.

Topics Covered

- **Splash Screen**
- **Custom Animations**
- **Wi-Fi Adapter**
- **Explicit Intent**
- **Geo Location**
- **Fire Base Authentication**
- **Shared Preferences**
- **Scroll View**
- **List Views**
- **Toasts**
- **Card View**
- **Drawable**
- **Scroll View**
- **Date Picker**

Screen Shots of Application


Splash Screen Activity



Login In Without Wi-Fi Connection Message

7:25 Tue, 23 Apr 37%

Looper Man




Looper Man

Enter Email

Enter Password

Login

[New Here? Register](#)


 Wi-Fi is off. Please turn on Wi-Fi to continue.

New User Register Activity

7:26 Tue, 23 Apr 37%

Looper Man

Find Your Magic Producer



Full Name

Email

Phone

Password

Confirm Password

Register As An Producer ☐


Register

Already Registered? Login

Existing User Login Page

7:26 Tue, 23 Apr 7:26 Tue, 23 Apr 37%

Looper Man



Looper Man

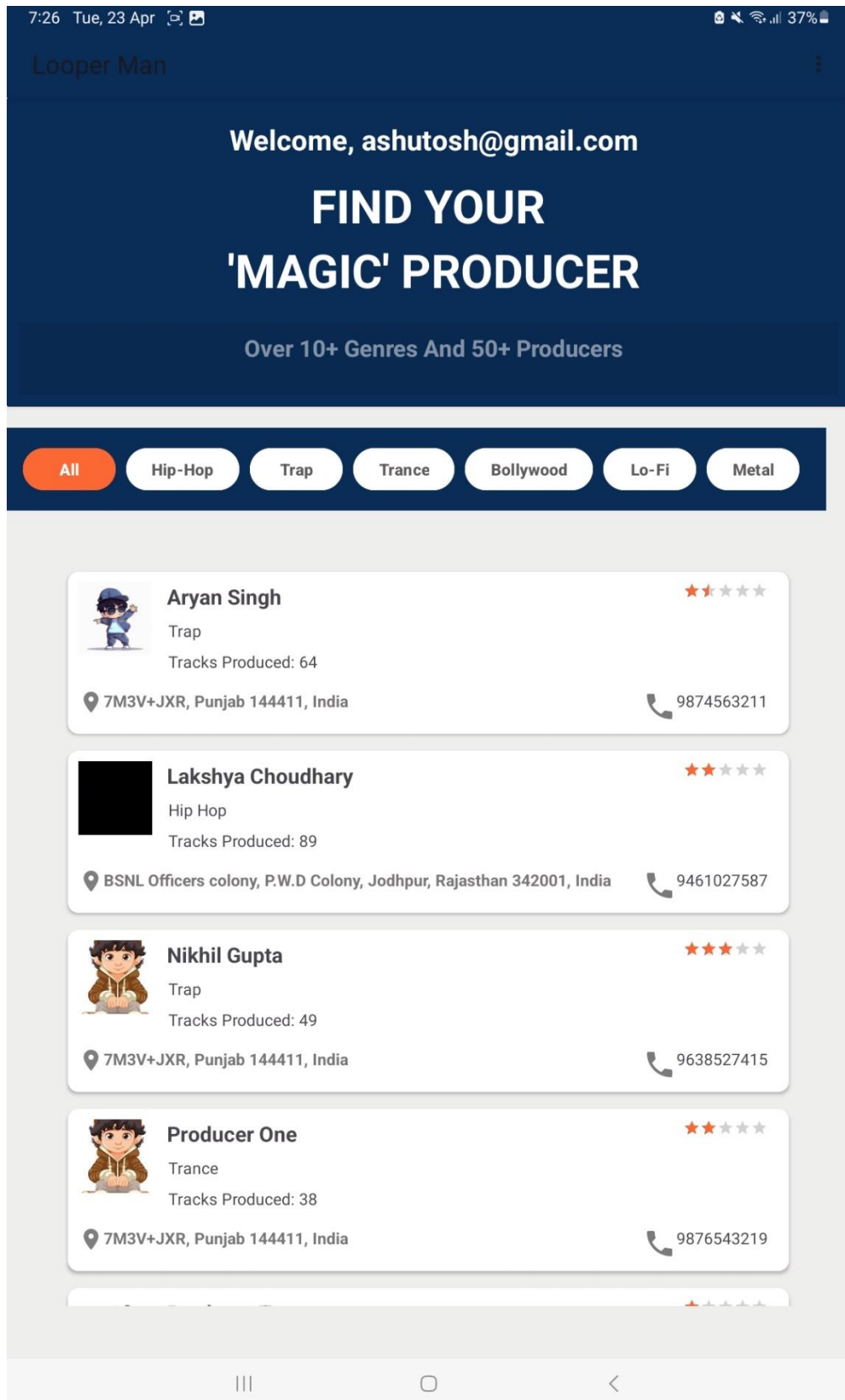
Enter Email

Enter Password

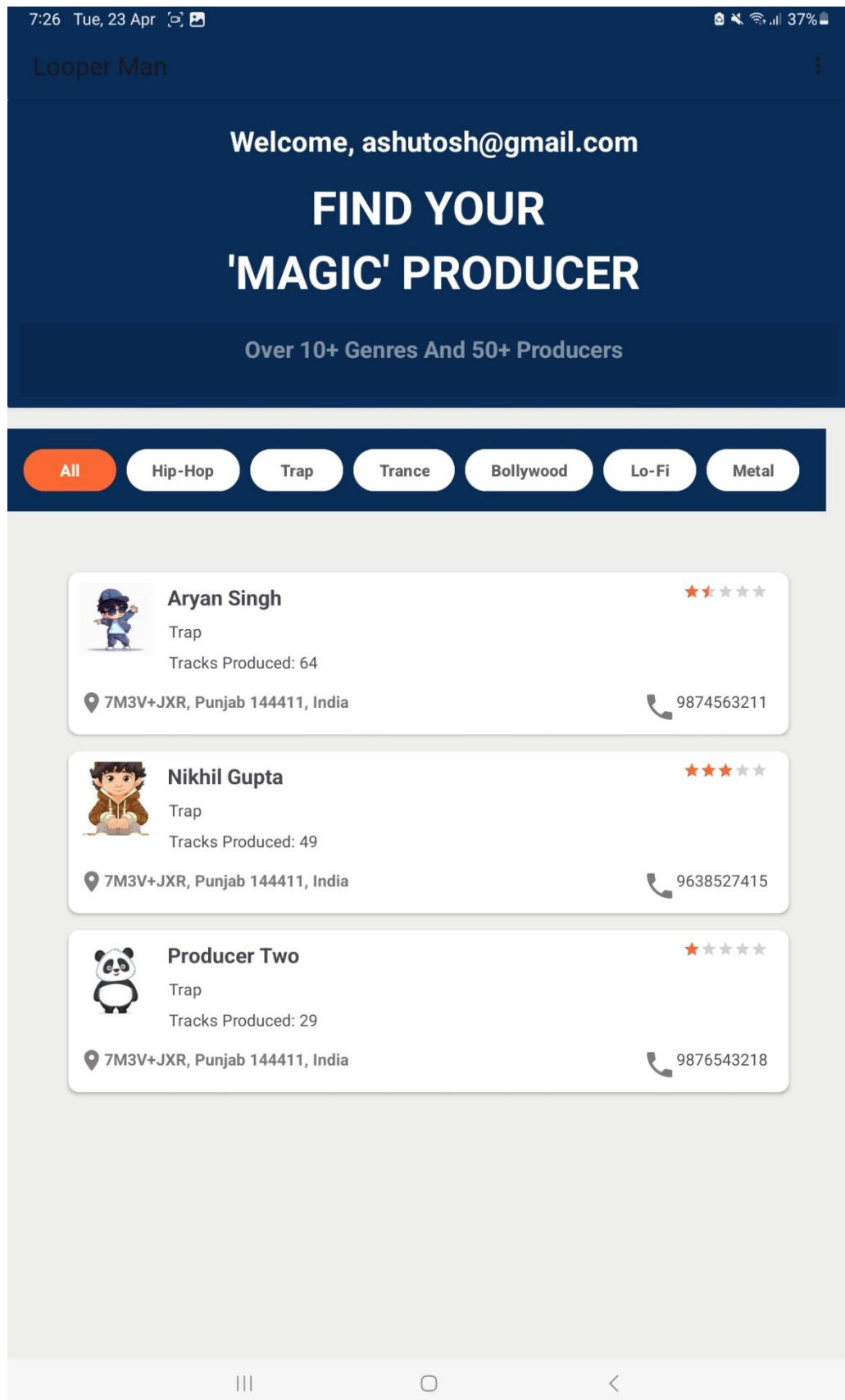
Login

New Here? Register

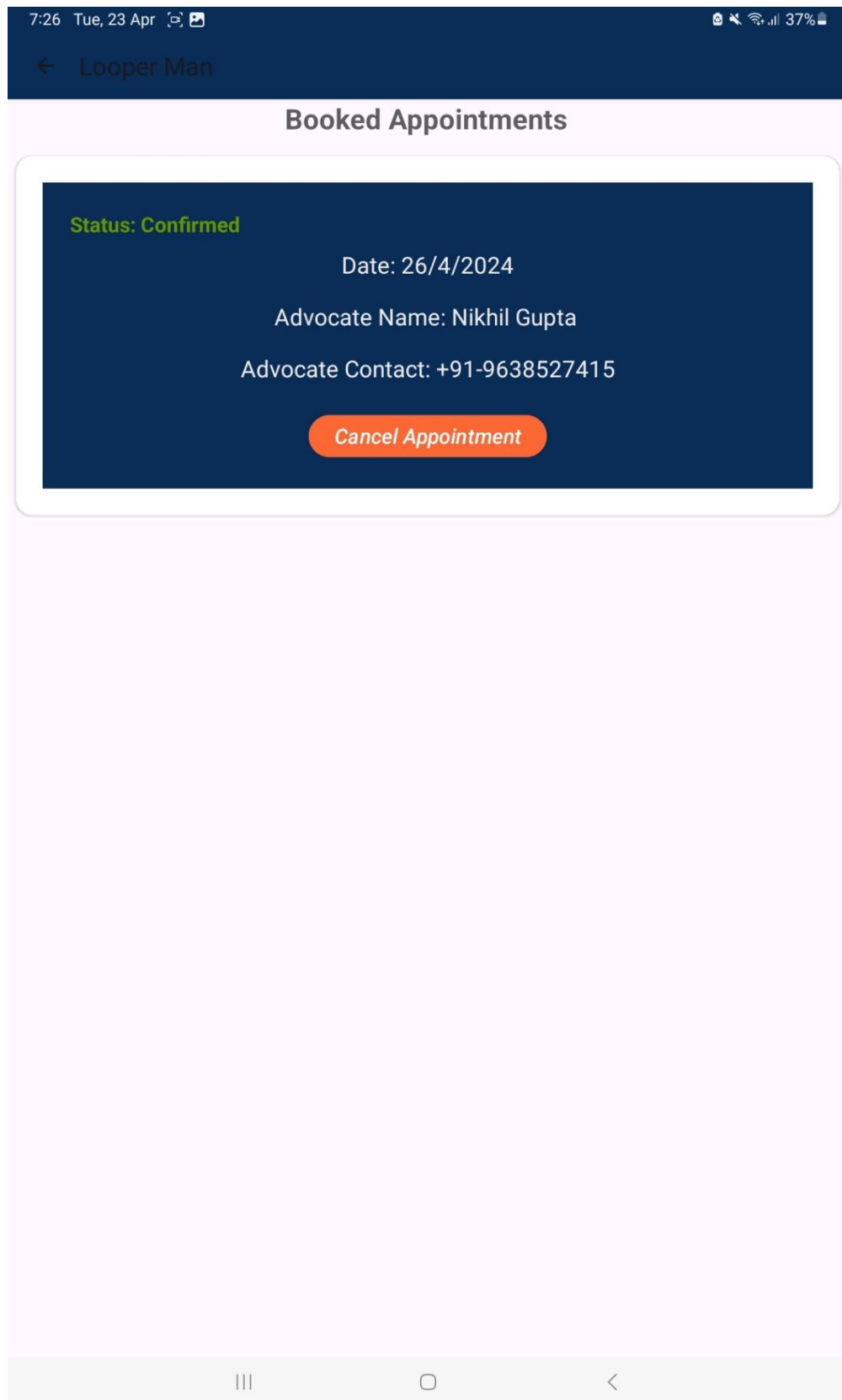
Main “Looper Man” Page



Filter Using Genre Page




Booking An Appointment



Producer Profile Activity

7:27 Tue, 23 Apr

Looper Man



Nikhil Gupta

Specialization: Trap

License Number: 345676543

Experience: 5

About:
Trap Based Music Producer From Mumbai

Tracks Produced: 49

Rating: 3.1079664

Location: 7M3V+JXR, Punjab 144411, India

Edit Profile


Producer Profile Edit Activity

7:27 Tue, 23 Apr

37%

←

Looper Man



Nikhil Gupta

Trap

345676543

Trap Based Music Producer From Mumbai

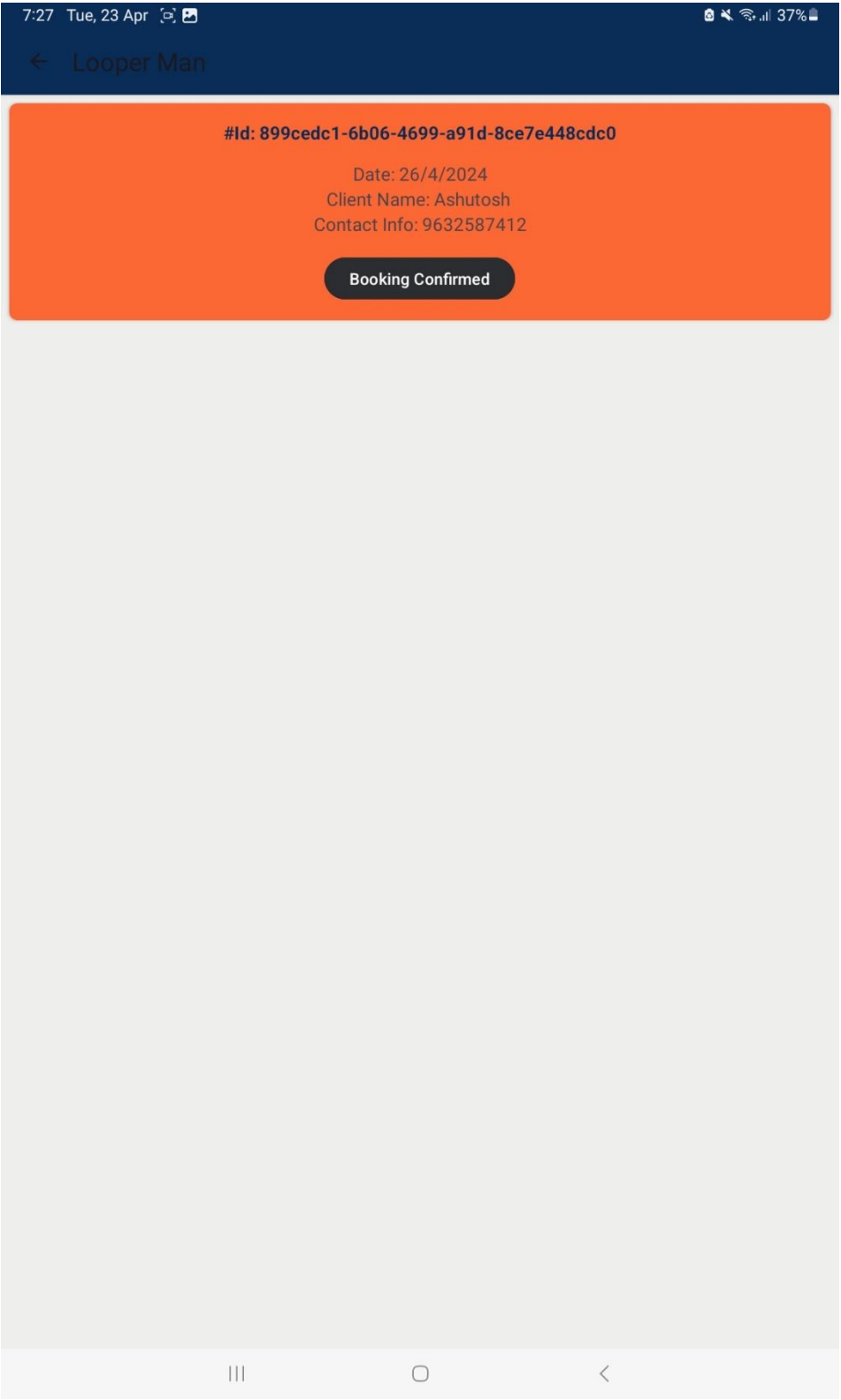
49

5

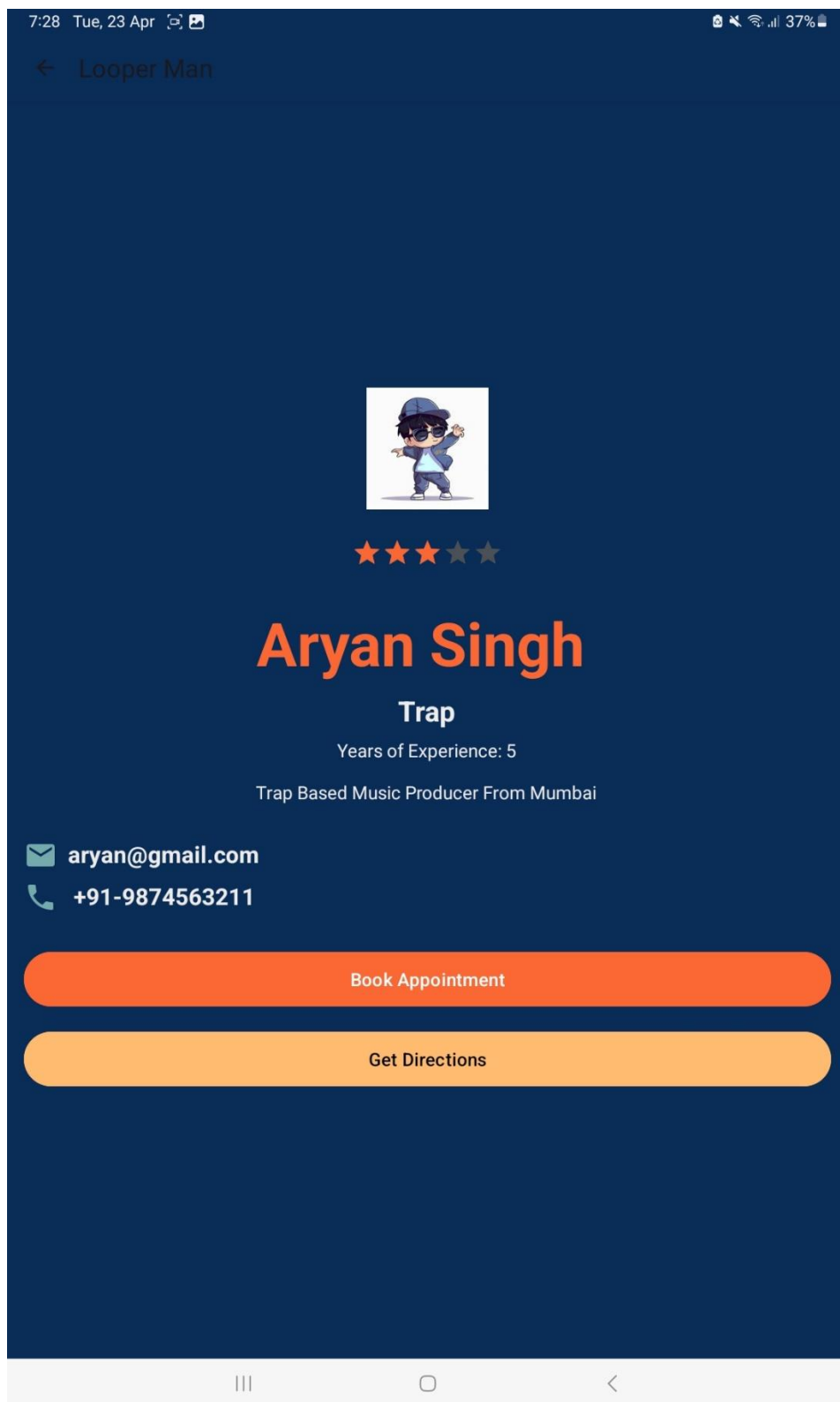
7M3V+JXR, Punjab 144411, India

Update

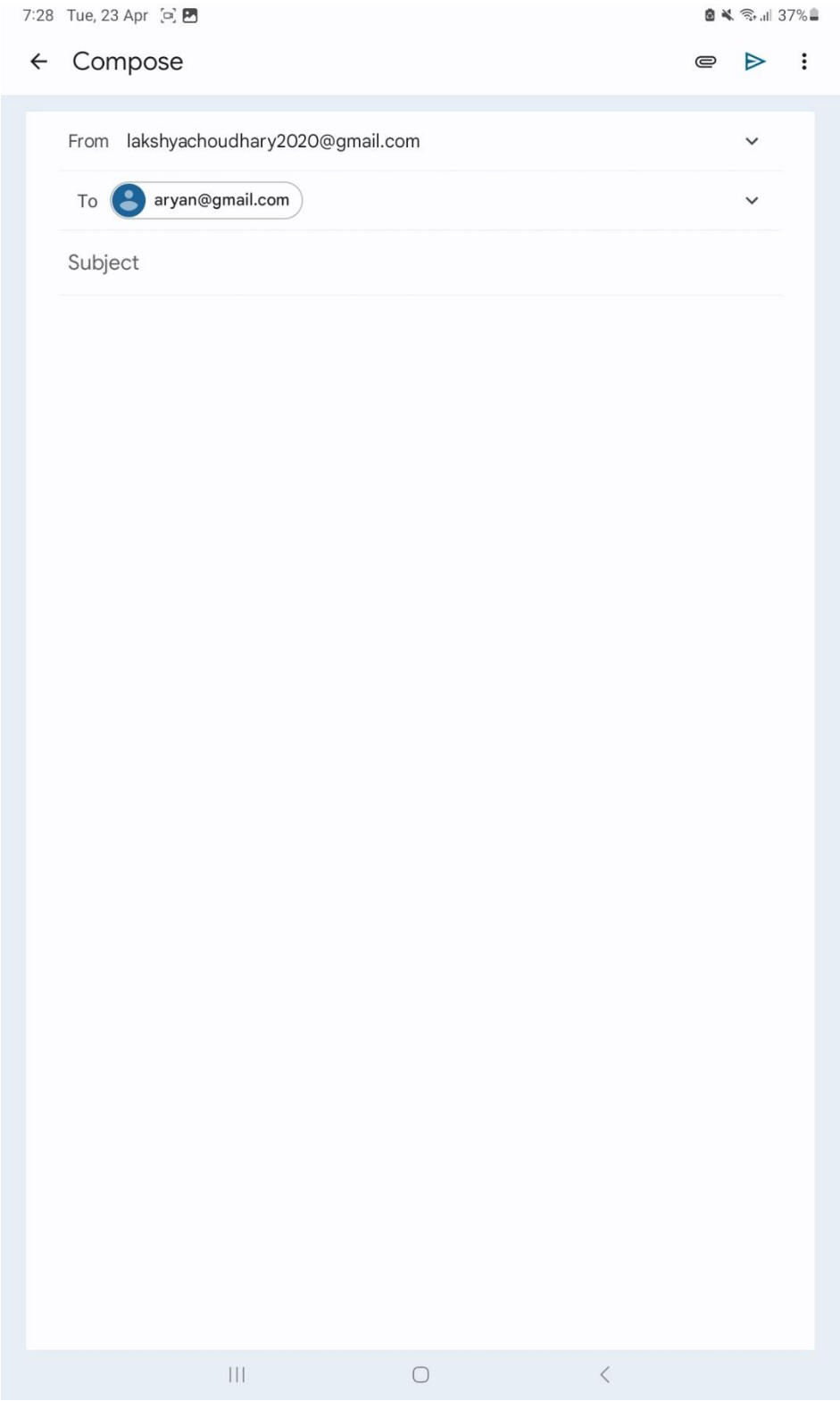
Producer Confirm Appointment Activity



Booking Appointment with Producer Activity



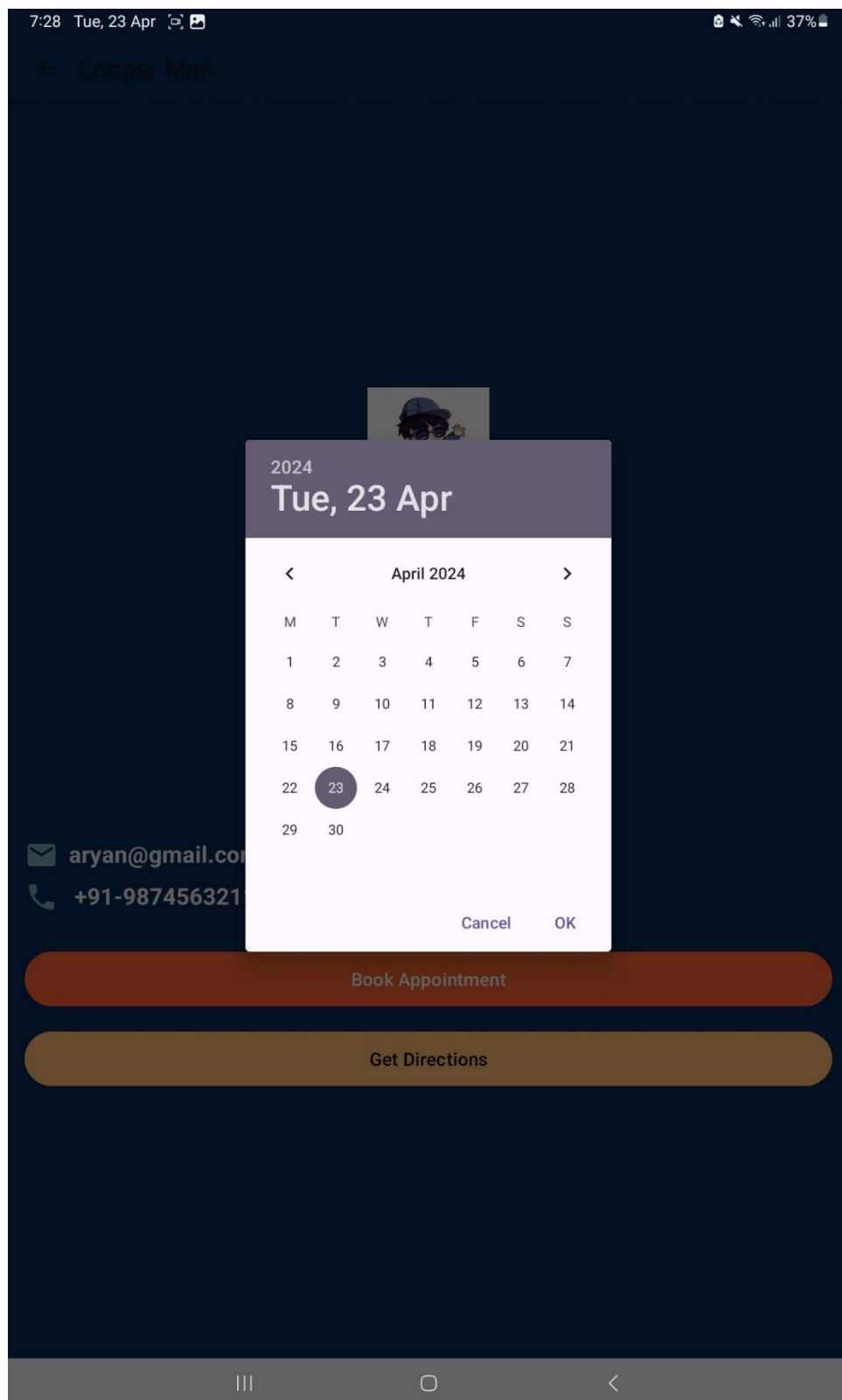
Sending Mail to Producer



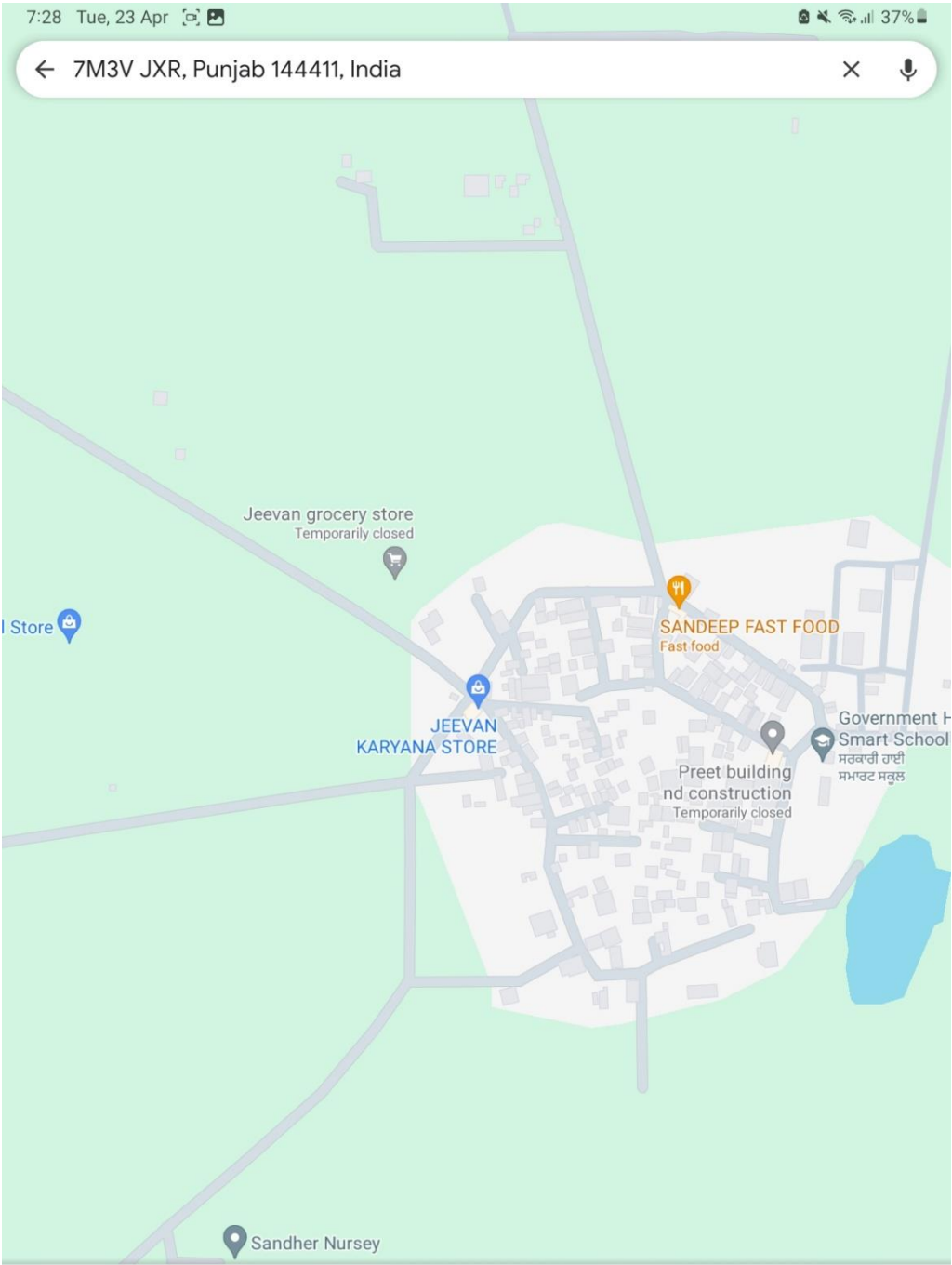
Calling Producer



Choosing Appointment Date with Producer



Opening Producer's Location Using Maps



Partial matches

7M3V-JXR, Punjab 144411, India

"Punjab 144411, India"

Khajurla, Punjab 144411

Postal code



Don't see what you're looking for?

[ADD A PLACE](#)



Motive Behind “Looper Man”

"Looper Man" is driven by a singular motive: to democratize the music production landscape in India. We believe that every aspiring musician, regardless of background or resources, deserves access to a platform where talent can thrive and creativity knows no bounds. By seamlessly connecting music enthusiasts with skilled producers from diverse backgrounds and genres, we aim to foster collaboration, innovation, and artistic expression within the vibrant Indian music scene. Our mission is to empower users to discover, connect, and create without limitations, providing a supportive community, intuitive tools, and valuable resources every step of the way. Whether you're a budding artist looking to collaborate or an established producer seeking new opportunities, "Looper Man" is your ultimate ally in realizing your musical aspirations and sharing your passion with the world. Join us in shaping the future of music production in India, one beat, melody, and collaboration at a time. "Looper Man" strives to bridge the gap between aspiring musicians and seasoned producers, fostering mentorship and knowledge exchange to nurture talent and elevate the music industry. With a commitment to inclusivity and diversity, we celebrate the rich tapestry of Indian music, amplifying voices and stories that reflect the cultural mosaic of our nation. By harnessing the power of technology and community, "Looper Man" empowers individuals to break barriers, defy conventions, and make their mark on the global stage of music production. Join our movement to unleash creativity, ignite passion, and shape the future of music in India and beyond. Together, let's make every beat count, as we embark on an exhilarating journey of sonic exploration and artistic discovery.

Producer.kt

```
package com.example.looperman

class Producer : User {
    var specialization: String = ""
    var licenseNumber: String = ""
    var about: String = ""
    var tracksProduced: Int = 0
    var rating: Float = 0.0f
    var location: String = ""
    var experience : String = ""

    constructor(
        specialization: String,
        licenseNumber: String,
        about: String,
        tracksProduced: Int,
        rating: Float,
        location: String,
        experience :String,
        name: String,
        email: String,
        pass: String,
        image: String,
        phone: String,
        role: String
    ) : super(name, email, pass, image, phone, role)
{
    this.specialization = specialization
    this.licenseNumber = licenseNumber
    this.about = about
    this.tracksProduced = tracksProduced
    this.rating = rating
    this.location = location
    this.experience = experience
}

    constructor() : super() {
        // No-argument constructor
    }
}
```

Producer Activity.kt

```
package com.example.looperman

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.Menu
import android.view.MenuItem
import android.widget.Button
import android.widget.TextView
import androidx.core.content.ContextCompat
import com.bumptech.glide.Glide
import
com.bumptech.glide.load.engine.DiskCacheStrategy
import com.bumptech.glide.request.RequestOptions
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import
com.google.firebase.database.ValueEventListener
import de.hdodenhof.circleimageview.CircleImageView
import java.text.DecimalFormat

class ProducerActivity : AppCompatActivity() {
    private lateinit var userId: String
    private lateinit var nameTextView: TextView
    private lateinit var specializationValueTextView:
TextView
    private lateinit var licenseNumberValueTextView:
TextView
    private lateinit var aboutValueTextView: TextView
    private lateinit var tracksProducedValueTextView:
TextView
    private lateinit var ratingValueTextView:
TextView
    private lateinit var locationValueTextView:
```

```

TextView
    private lateinit var databaseReference:
DatabaseReference
    private lateinit var profile : CircleImageView
    private lateinit var experienceValueTextView
:TextView

    private lateinit var update :Button
    private var imgLink :String = ""
    private lateinit var sharedPreferences:
SharedPreferences
    val decimalFormat = DecimalFormat("#.##")

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_producer)

        supportActionBar?.setBackgroundDrawable(ColorDrawable
(ContextCompat.getColor(this, R.color.primary)))
        window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

        // Initialize TextViews
        nameTextView =
findViewById(R.id.nameValueTextView)
        specializationValueTextView =
findViewById(R.id.specializationValueTextView)
        licenseNumberValueTextView =
findViewById(R.id.licenseNumberValueTextView)
        aboutValueTextView =
findViewById(R.id.aboutValueTextView)
        tracksProducedValueTextView =
findViewById(R.id.tracksProducedValueTextView)
        ratingValueTextView =
findViewById(R.id.ratingValueTextView)
        locationValueTextView =
findViewById(R.id.locationValueTextView)
        profile = findViewById(R.id.profile)
        update = findViewById(R.id.updateButton)
        experienceValueTextView =
findViewById(R.id.experienceValueTextView)

```

```

        sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)

        userId =
sharedPreferences.getString("username",
 "").toString()

        val sanitizedEm = sanitizeUserId(userId)
        Log.e("Sanitised Mail --->",
"${sanitizedEm}")

        databaseReference =
FirebaseDatabase.getInstance().getReference("Users").
child(sanitizedEm)
        fetchDataFromFirebase()

        update.setOnClickListener{
            var intent =
Intent(this,updateDetails::class.java)

intent.putExtra("specialization",specializationValueT
extView.text.toString())

intent.putExtra("name",nameTextView.text.toString())

intent.putExtra("licenseNumber",licenseNumberValueTex
tView.text.toString())

intent.putExtra("about",aboutValueTextView.text.toStr
ing())

intent.putExtra("tracksProduced",tracksProducedValueT
extView.text.toString())

intent.putExtra("location",locationValueTextView.text
.toString())

intent.putExtra("sanitizedEm",sanitizedEm.toString())
            intent.putExtra("image",imgLink)

intent.putExtra("experience",experienceValueTextView.
text.toString())
            startActivity(intent)

```



```

    }
}

private fun fetchDataFromFirebase() {

databaseReference.addListenerForSingleValueEvent(object : ValueEventListener {
    override fun onDataChange(dataSnapshot:
DataSnapshot) {
        val user =
dataSnapshot.getValue(Producer::class.java)
        user?.let {
            // Display user details in the
welcome activity

            nameTextView.text = it.name
            specializationValueTextView.text
= it.specialization
            licenseNumberValueTextView.text =
it.licenseNumber
            aboutValueTextView.text =
it.about
            tracksProducedValueTextView.text
= it.tracksProduced.toString()
            ratingValueTextView.text =
decimalFormat.format(it.rating)
            experienceValueTextView.text =
it.experience

            val parts =
it.location.split(",")

            // Find locality and admin
information

            var locality = ""
            var admin = ""
            for (part in parts) {
                if
(part.trim().startsWith("locality")) {
                    locality =
part.split("=")[1]
                } else if
(part.trim().startsWith("admin")) {
                    admin =

```

```

part.split("=") [1]
        }
    }

    locationValueTextView.text =
"$locality, $admin"

    if(it.image != null){
        imgLink = it.image.toString()

Glide.with(applicationContext).load(it.image)
        .apply(
            RequestOptions()

.error(R.drawable.user)

.placeholder(R.drawable.user)

.diskCacheStrategy(DiskCacheStrategy.ALL)
            ).into(profile)
        }
        updateUI(user)
    }
}

    override fun onCancelled(databaseError: DatabaseError) {
        Log.e("Firebase", "Error fetching data from Firebase: ${databaseError.message}")
        // Handle error (e.g., display a message to the user)
    }
})
}

    private fun sanitizeUserId(userId: String): String {
        return userId.replace(".", "_")
            .replace("#", "_")
            .replace("$", "_")
            .replace("[", "_")
            .replace("]", "_")
            .replace(" ", "_")
            .toLowerCase()
    }
}

```

```

        override fun onCreateOptionsMenu(menu: Menu?):
Boolean {

    menuInflater.inflate(R.menu.producer_menu_list, menu)
        return true
    }

    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        when (item.itemId) {
            R.id.action_profile -> {

                return true
            }
            R.id.action_appointment -> {
                val intent =
Intent(this, ViewAppointments::class.java)
                startActivity(intent)
                return true
            }
            R.id.action_logout -> {
                val intent =
Intent(this, Login::class.java)
                startActivity(intent)
                return true
            }
            else -> return
super.onOptionsItemSelected(item)
        }
    }

    override fun onResume() {
        Log.d("On Resume -----> ", "onResume: ")
        fetchDataFromFirebase()
        super.onResume()
    }

    private fun updateUI(user: Producer) {
        // Update UI elements with data from Firebase
        nameTextView.text = user.name
        specializationValueTextView.text =
user.specialization
        licenseNumberValueTextView.text =
user.licenseNumber
    }

```

```
        aboutValueTextView.text = user.about
        tracksProducedValueTextView.text =
user.tracksProduced.toString()
        ratingValueTextView.text =
user.rating.toString()
        locationValueTextView.text = user.location

        if (user.image != null) {
            // Load image using Glide or any other
image loading library

Glide.with(applicationContext).load(user.image)
            .apply(
                RequestOptions()
                    .error(R.drawable.user)
                    .placeholder(R.drawable.user)

            .diskCacheStrategy(DiskCacheStrategy.ALL)
                ).into(profile)
        }
    }
}
```

Producer Adapter.kt

```
package com.example.looperman

import android.content.Intent
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.RatingBar
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide

class ProducerAdapter(private var producerList:
List<Producer>) :
RecyclerView.Adapter<ProducerAdapter.Producer
ViewHolder>() {

    override fun onCreateViewHolder(parent:
ViewGroup, viewType: Int): ProducerViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.
producer_items, parent, false)
        return ProducerViewHolder(view)
    }

    fun updateList(newList: List<Producer >) {
        producerList = newList
        notifyDataSetChanged()
    }

    override fun onBindViewHolder(holder:
ProducerViewHolder, position: Int) {
        val currentProducer = producerList[position]

        holder.nameTextView.text =
currentProducer.name
        holder.specializationTextView.text =
currentProducer.specialization
        holder.tracksProducedTextView.text = "Tracks
Produced: ${currentProducer.tracksProduced}"
        holder.locationTextView.text =
```

```

currentProducer.location
        holder.ratingsRatingBar.rating =
currentProducer.rating
        holder.editTextPhone.text =
currentProducer.phone

        // Load profile picture using Glide
        Glide.with(holder.itemView.context)
            .load(currentProducer.image)
            .placeholder(R.drawable.user) //
Placeholder image
            .error(R.drawable.user) // Error image
            .into(holder.profilePicImageView)
        // Set click listener for the card
        holder.itemView.setOnClickListener {

            val context = holder.itemView.context
            val intent = Intent(context,
ProducerBook::class.java)

            intent.putExtra("specialization",currentProducer.spec
ialization)

            intent.putExtra("name",currentProducer.name)

            intent.putExtra("licenseNumber",currentProducer.licen
seNumber)

            intent.putExtra("about",currentProducer.about)

            intent.putExtra("tracksProduced",currentProducer.trac
ksProduced)

            intent.putExtra("location",currentProducer.location)

            intent.putExtra("email",currentProducer.email)

            intent.putExtra("phone",currentProducer.phone)

            intent.putExtra("image",currentProducer.image)

            intent.putExtra("experience",currentProducer.experien
ce)

```

```

intent.putExtra("rating",currentProducer.rating)
                context.startActivity(intent)
            }
        }

        override fun getItemCount() = ProducerList.size

        inner class ProducerViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
            val profilePicImageView: ImageView =
itemView.findViewById(R.id.profilePicImageView)
            val nameTextView: TextView =
itemView.findViewById(R.id.nameTextView)
            val ratingsRatingBar: RatingBar =
itemView.findViewById(R.id.ratingsRatingBar)
            val specializationTextView: TextView =
itemView.findViewById(R.id.specializationTextView)
            val tracksProducedTextView: TextView =
itemView.findViewById(R.id.tracksProducedsWonTextView
)
            val locationTextView: TextView =
itemView.findViewById(R.id.locationTextView)
            val editTextPhone:TextView =
itemView.findViewById(R.id.editTextPhone)
        }
    }
}

```

ProducerBook.kt

```
package com.example.looperman

import android.app.DatePickerDialog
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.core.content.ContextCompat
import com.bumptech.glide.Glide
import com.google.firebase.database.FirebaseDatabase
import java.util.Calendar
import java.util.UUID

class ProducerBook : AppCompatActivity() {
    private lateinit var profilePicImageView:
    ImageView
    private lateinit var fullNameTextView: TextView
    private lateinit var specializationTextView:
    TextView
    private lateinit var yearsOfExperienceTextView:
    TextView
    private lateinit var emailTextView: TextView
    private lateinit var phoneTextView: TextView
    private lateinit var bioTextView: TextView
    private lateinit var bookAppointmentButton:
    Button
    private lateinit var getDirectionsButton: Button
    private lateinit var sharedPreferences:
    SharedPreferences
    private val database =
    FirebaseDatabase.getInstance().reference
```



```

        private var selectedDate: String = ""

        private var userId: String = ""
        private var ProducerId: String = ""

        override fun onCreate(savedInstanceState:
Bundle?) {
            super.onCreate(savedInstanceState)

setContentViews(R.layout.activity_Producer_book)

supportActionBar?.setBackgroundDrawable(ColorDrawable
(ContextCompat.getColor(this, R.color.primary)))
            window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

supportActionBar?.setDisplayHomeAsUpEnabled(true)

            // Initialize views
            profilePicImageView =
findViewById(R.id.profilePicImageView)
            fullNameTextView =
findViewById(R.id.fullNameTextView)
            specializationTextView =
findViewById(R.id.specializationTextView)
            yearsOfExperienceTextView =
findViewById(R.id.yearsOfExperienceTextView)
            emailTextView =
findViewById(R.id.emailTextView)
            phoneTextView =
findViewById(R.id.phoneTextView)
            bioTextView = findViewById(R.id.bioTextView)
            bookAppointmentButton =
findViewById(R.id.bookAppointmentButton)
            getDirectionsButton =
findViewById(R.id.getDirectionsButton)

            Glide.with(this)
                .load(intent.getStringExtra("image"))
                .placeholder(R.drawable.user) //
Placeholder image
                .error(R.drawable.user) // Error image
                .into(profilePicImageView)

```

```

        fullNameTextView.text =
intent.getStringExtra("name")
        specializationTextView.text =
intent.getStringExtra("specialization")
        yearsOfExperienceTextView.text = "Years of
Experience: ${intent.getStringExtra("experience")}"
        emailTextView.text =
intent.getStringExtra("email")
        phoneTextView.text = "+91-
${intent.getStringExtra("phone")}"
        bioTextView.text =
intent.getStringExtra("about")

        bookAppointmentButton.setOnClickListener {
            // Implement booking appointment logic
            sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)
            userId =
sharedPreferences.getString("username",
"").toString()
            ProducerId =
intent.getStringExtra("email").toString()
            showDatePicker()
        }

        getDirectionsButton.setOnClickListener {
            // Implement getting directions logic

openGoogleMaps(intent.getStringExtra("location").toString())
        }

        // Open default call app when phone number is
clicked
        phoneTextView.setOnClickListener {
            val phoneUri =
Uri.parse("tel:${intent.getStringExtra("phone")}")
            val callIntent =
Intent(Intent.ACTION_DIAL, phoneUri)
            startActivity(callIntent)
        }

        // Open default email app when email is

```

```

clicked
        emailTextView.setOnClickListener {
            val emailUri =
Uri.parse("mailto:${intent.getStringExtra("email")}")
            val emailIntent =
Intent(Intent.ACTION_SENDTO, emailUri)
            startActivity(emailIntent)
        }
    }

    private fun openGoogleMaps(location: String) {
        val gmmIntentUri =
Uri.parse("geo:0,0?q=$location")
        val mapIntent = Intent(Intent.ACTION_VIEW,
gmmIntentUri)

        mapIntent.setPackage("com.google.android.apps.maps")
        startActivity(mapIntent)
    }

    private fun bookAppointment(userId: String,
ProducerId: String, dateTime: String) {
        // Generate a unique appointment ID
        val appointmentId =
UUID.randomUUID().toString()

        // Create an instance of AppointmentModel
with the appointment details
        val appointment = AppointmentModel(
            appointmentId,
            ProducerId,
            userId,
            dateTime,
            false // Set isConfirmed to false
initially
        )

        // Push the appointment data to the database

        database.child("appointments").child(appointmentId).s
etValue(appointment)
            .addOnSuccessListener {
                Toast.makeText(this, "Appointment
booked for: $dateTime", Toast.LENGTH_SHORT).show()

```

```

        }
        .addOnFailureListener {
            Toast.makeText(this, "Failed to book
appointment. Please try again.",
Toast.LENGTH_SHORT).show()
        }
    }

    private fun showDatePicker() {
        // Get current date
        val calendar = Calendar.getInstance()
        val year = calendar.get(Calendar.YEAR)
        val month = calendar.get(Calendar.MONTH)
        val day = calendar.get(Calendar.DAY_OF_MONTH)

        val datePickerDialog = DatePickerDialog(
            this,
            { _, selectedYear, selectedMonth,
selectedDay ->
                selectedDate =
"$selectedDay/${selectedMonth + 1}/${selectedYear}"
                bookAppointment(userId, ProducerId,
selectedDate)
            },
            year,
            month,
            day
        )
        datePickerDialog.show()
    }
}

```

AppointmentAdapter.kt

```
package com.example.locatemyProducer

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.database.DatabaseReference

class AppointmentAdapter(
    var appointments: List<AppointmentModel>,
    private val database: DatabaseReference
) :
    RecyclerView.Adapter<AppointmentAdapter.ViewHolder>()
{

    inner class ViewHolder(itemView: View) :
        RecyclerView.ViewHolder(itemView) {
        val appointmentIdTextView: TextView =
            itemView.findViewById(R.id.appointmentIdTextView)
        val dateTextView: TextView =
            itemView.findViewById(R.id.dateTextView)
        val clientNameTextView: TextView =
            itemView.findViewById(R.id.clientNameTextView)
        val clientContactTextView: TextView =
            itemView.findViewById(R.id.clientContactTextView)
        val confirmButton: Button =
            itemView.findViewById(R.id.confirmButton)
    }

    override fun onCreateViewHolder(parent:
        ViewGroup, viewType: Int): ViewHolder {
        val view =
```

```

LayoutInflater.from(parent.context).inflate(R.layout.appointment_item, parent, false)
    return ViewHolder(view)
}

    override fun onBindViewHolder(holder: ViewHolder,
position: Int) {
        val appointment = appointments[position]
        holder.appointmentIdTextView.text = "#Id:
${appointment.appointmentId}"
        holder.dateTextView.text = "Date:
${appointment.date}"
        val clientId: String =
appointment.clientId.toString()
        val sanitizedEm = clientId.replace(".",
"_").replace("#", "_").replace("$", "_").replace("[" ,
"_").replace("]", "_")

        // Fetch client details from the database
        database.child("Users").child(sanitizedEm ?:
"" ).get().addOnSuccessListener { snapshot ->
            if (snapshot.exists()) {
                val clientName =
snapshot.child("name").value.toString()
                val clientContact =
snapshot.child("phone").value.toString()
                holder.clientNameTextView.text =
"Client Name: $clientName"
                holder.clientContactTextView.text =
"Contact Info: $clientContact"
            }
        }.addOnFailureListener {
            // Handle database fetch failure
        }

        if (appointment.isConfirmed == true) {
            holder.confirmButton.text = "Booking
Confirmed"

            holder.confirmButton.setBackgroundColor(ContextCompat
.getColor(holder.itemView.context, R.color.gray))

            holder.confirmButton.setTextColor(ContextCompat.getCo

```

```

lor(holder.itemView.context, android.R.color.white))
// Set text color to white
        holder.confirmButton.isEnabled = false //
Disable the button
    } else {
        holder.confirmButton.text = "Confirm
Appointment"

holder.confirmButton.setBackgroundColor(ContextCompat
.getColor(holder.itemView.context, R.color.primary))
        holder.confirmButton.isEnabled = true //
Enable the button
    }
    // Handle confirm button click
    holder.confirmButton.setOnClickListener {
        val appointmentId =
appointment.appointmentId
        if (appointmentId != null) {

confirmAppointment(holder.itemView.context,holder, app
ointmentId)
        }
    }
}

    private fun confirmAppointment(context: Context,
holder: ViewHolder, appointmentId: String) {
        // Update appointment status in the database
to confirmed

database.child("appointments").child(appointmentId).c
hild("confirmed").setValue(true)
        .addOnSuccessListener {
            // Handle success
            holder.confirmButton.text = "Booking
Confirmed"

holder.confirmButton.setBackgroundColor(ContextCompat
.getColor(holder.itemView.context, R.color.gray))

holder.confirmButton.setTextColor(ContextCompat.getCo
lor(holder.itemView.context, android.R.color.white))
// Set text color to white
            holder.confirmButton.isEnabled =

```

```
false // Disable the button
        Toast.makeText(context, "Appointment
confirmed", Toast.LENGTH_SHORT).show()
    }
    .addOnFailureListener {
        // Handle failure
        Toast.makeText(context, "Appointment
confirmation failed", Toast.LENGTH_SHORT).show()
    }
}

override fun getItemCount(): Int {
    return appointments.size
}
}
```


ClientAppointmentAdapter.kt

```
package com.example.locatemyProducer

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.core.content.ContextCompat
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.database.*

class AppointmentClientAdapter(
    var appointments: List<AppointmentModel>,
    private val database: DatabaseReference
) :
    RecyclerView.Adapter<AppointmentClientAdapter.ViewHolder>() {

    inner class ViewHolder(itemView: View) :
        RecyclerView.ViewHolder(itemView) {
        val statusTextView: TextView =
            itemView.findViewById(R.id.statusTextView)
        val dateTextView: TextView =
            itemView.findViewById(R.id.dateTextView)
        val ProducerNameTextView: TextView =
            itemView.findViewById(R.id.ProducerNameTextView)
        val ProducerContactTextView: TextView =
            itemView.findViewById(R.id.ProducerContactTextView)
        val cancelAppointmentButton: Button =
            itemView.findViewById(R.id.cancelAppointmentButton)
    }

    override fun onCreateViewHolder(parent:
        ViewGroup, viewType: Int): ViewHolder {
        val view =
            LayoutInflater.from(parent.context)
                .inflate(R.layout.client_booking, parent,
                    false)
```

```

        return ViewHolder(view)
    }

    override fun onBindViewHolder(holder: ViewHolder,
position: Int) {
        val appointment = appointments[position]
        // Set appointment details
        if (appointment.isConfirmed == true) {
            holder.statusTextView.text = "Status:
Confirmed"

holder.statusTextView.setTextColor(ContextCompat.getCo
lor(holder.itemView.context,
android.R.color.holo_green_dark))
        } else {
            holder.statusTextView.text = "Status:
Pending"

holder.statusTextView.setTextColor(ContextCompat.getCo
lor(holder.itemView.context,
android.R.color.holo_blue_light))
        }
        holder.dateTextView.text = "Date:
${appointment.date}"

        fetchProducerDetails(holder,
appointment.ProducerId ?: "")

        // Handle cancel appointment button click

holder.cancelAppointmentButton.setOnClickListener {
        // Implement cancel appointment logic
        here
    }
}

// Fetch Producer details from the database and
set them to TextViews
    private fun fetchProducerDetails(holder:
ViewHolder, ProducerId: String) {

        val sanitizedEm = ProducerId.replace(".",
"_").replace("#", "_").replace("$", "_").replace("[",
"_").replace("]", "_")

```

```

        val ProducerRef =
FirebaseDatabase.getInstance().reference.child("Users
").child(sanitizedEm)

ProducerRef.addListenerForSingleValueEvent(object :
ValueEventListener {
    override fun onDataChange(dataSnapshot:
DataSnapshot) {
        if (dataSnapshot.exists()) {
            val ProducerName =
dataSnapshot.child("name").value.toString()
            val ProducerContact =
dataSnapshot.child("phone").value.toString()
            holder.ProducerNameTextView.text
= "Producer Name: $ProducerName"

holder.ProducerContactTextView.text = "Producer
Contact: +91-$ProducerContact"

        } else {
            // Handle tracksProduced where
Producer data does not exist
            holder.ProducerNameTextView.text
= "Producer Name: Not found"

holder.ProducerContactTextView.text = "Producer
Contact: Not found"

        }
    }

    override fun onCancelled(databaseError:
DatabaseError) {
        // Handle database error
        // For simplicity, showing a Toast
message here. You can replace it with a more robust
error handling mechanism.
        holder.ProducerNameTextView.text =
"Producer Name: Error"
        holder.ProducerContactTextView.text =
"Producer Contact: Error"
    }
})

```

```
}
```

AppointmentModel.kt

```
package com.example.looperman

import com.google.firebase.database.Exclude
import com.google.firebase.database.IgnoreExtraProperties

@IgnoreExtraProperties
data class AppointmentModel(
    var appointmentId: String? = null,
    var ProducerId: String? = null,
    var clientId: String? = null,
    var date: String? = null,
    var isConfirmed: Boolean? = null
) {
    @Exclude
    fun toMap(): Map<String, Any?> {
        return mapOf(
            "appointmentId" to appointmentId,
            "ProducerId" to ProducerId,
            "clientId" to clientId,
            "date" to date,
            "isConfirmed" to isConfirmed
        )
    }
}
```

ClientBookedActivity.kt

```
package com.example.looperman

import android.content.Context
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.core.content.ContextCompat
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.Query
import
com.google.firebase.database.ValueEventListener

class Client_booked_appointment : AppCompatActivity()
{
    private lateinit var bookingRecyclerView:
RecyclerView
    private lateinit var database: DatabaseReference
    private lateinit var adapter:
AppointmentClientAdapter
    private lateinit var sharedPreferences:
SharedPreferences
    private var clientId : String = ""

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_client_booked_appoin
tment)

        supportActionBar?.setBackgroundDrawable(ColorDrawable
```

```

(ContextCompat.getColor(this, R.color.primary)))
        window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

supportActionBar?.setDisplayHomeAsUpEnabled(true)

        bookingRecyclerView =
findViewById(R.id.bookingRv)

        database =
FirebaseDatabase.getInstance().reference.child("appointments")

        sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)
        clientId =
sharedPreferences.getString("username",
"").toString()

        adapter =
AppointmentClientAdapter(emptyList(), database)
        bookingRecyclerView.adapter = adapter

        val layoutManager = LinearLayoutManager(this)
        bookingRecyclerView.layoutManager =
layoutManager

        fetchAppointmentsForProducer(clientId ?: "")
    }

    private fun
fetchAppointmentsForProducer(clientId: String) {
        val query: Query =
database.orderByChild("clientId").equalTo(clientId)
        query.addListenerForSingleValueEvent(object :
ValueEventListener {
            override fun onDataChange(dataSnapshot:
DataSnapshot) {
                val appointmentsList =
mutableListOf<AppointmentModel>()

                if (dataSnapshot.exists()) {

```

```

                                for (appointmentSnapshot in
dataSnapshot.children) {
                                    val appointment =
appointmentSnapshot.getValue(AppointmentModel::class.java)
                                        appointment?.let {
                                            appointmentsList.add(it)
                                        }
                                    }
                                adapter.appointments =
appointmentsList
                                adapter.notifyDataSetChanged()
                            }
                        }

                        override fun onCancelled(databaseError:
DatabaseError) {
                            // Handle database error
                        }
                    })
                }
            }
        }
    }
}

```

LoginActivity.kt

```
package com.example.looperman

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import android.net.ConnectivityManager
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Login : AppCompatActivity() {

    private lateinit var firebaseAuth: FirebaseAuth
    private lateinit var password: EditText
    private lateinit var register: TextView
    private lateinit var loginButton: Button
    private lateinit var email: EditText
    private lateinit var sharedPreferences:
SharedPreferences

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        supportActionBar?.setBackgroundDrawable(ColorDrawable(
ContextCompat.getColor(this, R.color.primary)))
        window?.statusBarColor =
```



```

ContextCompat.getColor(this, R.color.primary)

        email = findViewById(R.id.emailId)
        password = findViewById(R.id.password)
        loginButton = findViewById(R.id.loginButton)
        register = findViewById(R.id.signupText)
        firebaseAuth = FirebaseAuth.getInstance()
        loginButton.setOnClickListener {
            val userEmail = email.text.toString()
            val pass = password.text.toString()

            if (userEmail.isNotEmpty() &&
pass.isNotEmpty()) {
                if (isWifiEnabled()) {

firebaseAuth.signInWithEmailAndPassword(userEmail,
pass)

                                .addOnCompleteListener {
loginTask ->

                                    try {
                                        if

(loginTask.isSuccessful) {
                                                    // Successfully
logged in, fetch user data from the database

fetchUserDataFromDatabase(userEmail, pass)
                                                } else {
                                                    throw

loginTask.exception ?: Exception("Unknown error")
                                                }
                                            } catch (e: Exception) {
                                                handleLoginError(e)
                                            }
                                        }
                                    } else {
                                        Toast.makeText(this, "Wi-Fi is
off. Please turn on Wi-Fi to continue.",
Toast.LENGTH_SHORT).show()
                                    }
                                    } else {
                                        Toast.makeText(this, "Please enter
email and password", Toast.LENGTH_SHORT).show()
                                    }
                                }
}

```

```

        register.setOnClickListener {
            val intent: Intent = Intent(this,
SignUp::class.java)
            startActivity(intent)
        }
        sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)

email.setText(sharedPreferences.getString("username",
""))

password.setText(sharedPreferences.getString("password", ""))
    }

    private fun fetchUserDataFromDatabase(userEmail:
String, pass: String) {
        val sanitizedEm = userEmail.replace(".",
"_").replace("#", "_").replace("$", "_").replace("[",
"_").replace("]", "_")
        val reference =
FirebaseDatabase.getInstance().getReference("Users").
child(sanitizedEm)

reference.addListenerForSingleValueEvent(object :
ValueEventListener {
    override fun onDataChange(dataSnapshot:
DataSnapshot) {
        if (dataSnapshot.exists()) {
            val user =
dataSnapshot.getValue(User::class.java)
            if (user != null) {
                val userName = user.name
                val userImage = user.image
                val userRole = user.role
                val userEmail = user.email
                Log.d("User Data", "User
Name: $userName, User Image: $userImage")

                val editor =
sharedPreferences.edit()

```

```

        userEmail)
        pass)
        userName)
        userImage)
        editor.putString("username",
        editor.putString("password",
        editor.putString("userName",
        editor.putString("userImage",
        editor.apply()

        val targetActivity = when
(userRole) {
            "producer" ->
ProducerActivity::class.java
            else ->
MainActivity::class.java // Default interface for
unknown roles
        }

        // Start the MainActivity and
pass the user data
        val intent =
Intent(applicationContext , targetActivity)
        intent.putExtra("USER_ID",
userEmail)
        intent.putExtra("userName",
userName)
        intent.putExtra("userImage",
userImage)
        startActivity(intent)
    } else {
        Log.e("User Data", "Failed to
deserialize user data")
    }
    } else {
        Log.e("User Data", "DataSnapshot
does not exist")
    }
}

    override fun onCancelled(databaseError:
DatabaseError) {
        Log.e("Database Error", "Error:
${databaseError.message}")
    }
}

```

```

        handleLoginError(Exception("Database
Error: ${databaseError.message}"))
    }
}

}

private fun handleLoginError(exception:
Exception) {
    runOnUiThread {
        Toast.makeText(this, "Login Error:
${exception.message}", Toast.LENGTH_SHORT).show()
    }
}

private fun isWifiEnabled(): Boolean {
    val connectivityManager =
getSystemService(Context.CONNECTIVITY_SERVICE) as
ConnectivityManager
    val networkInfo =
connectivityManager.activeNetworkInfo
    return networkInfo?.type ==
ConnectivityManager.TYPE_WIFI
}
}

```

MainActivity.kt

```
package com.example.looperman

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.Menu
import android.view.MenuItem
import android.widget.Button
import android.widget.TextView
import androidx.core.content.ContextCompat
import
androidx.recyclerview.widget.DefaultItemAnimator
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import
com.google.android.material.bottomnavigation.BottomNa
vigationView
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import
com.google.firebase.database.ValueEventListener

class MainActivity : AppCompatActivity() {
    lateinit var userName: TextView
    private lateinit var rv: RecyclerView
    private lateinit var databaseReference:
DatabaseReference
    private lateinit var adapter: ProducerAdapter
    private val mainList = ArrayList<Producer>()
    private lateinit var sharedPreferences:
SharedPreferences
    private lateinit var allButton: Button
    private lateinit var realStateButton: Button
```

```

private lateinit var immigrationButton: Button
private lateinit var corporateButton: Button
private lateinit var familyButton: Button
private lateinit var financialButton: Button
private lateinit var criminalButton: Button

override fun onCreate(savedInstanceState:
Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    supportActionBar?.setBackgroundDrawable(
        ColorDrawable(
            ContextCompat.getColor(
                this,
                R.color.primary
            )
        )
    )
    window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

    allButton = findViewById(R.id.all)
    realStateButton =
findViewById(R.id.realState)
    immigrationButton =
findViewById(R.id.Immigration)
    corporateButton =
findViewById(R.id.Corporate)
    familyButton = findViewById(R.id.family)
    financialButton =
findViewById(R.id.financial)
    criminalButton = findViewById(R.id.criminal)

    allButton.setOnClickListener {
filterBySpecialization("") } // Empty string for all
    realStateButton.setOnClickListener {
filterBySpecialization("Hip Hop") }
    immigrationButton.setOnClickListener {
filterBySpecialization("Trap") }
    corporateButton.setOnClickListener {
filterBySpecialization("Trance") }

```

```

        familyButton.setOnClickListener {
filterBySpecialization("Bollywood") }
        financialButton.setOnClickListener {
filterBySpecialization("Lo-Fi") }
        criminalButton.setOnClickListener {
filterBySpecialization("Metal") }

        sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)
        userName = findViewById(R.id.userName)
        userName.setText("Welcome, " +
sharedPreferences.getString("username", ""))

        rv = findViewById(R.id.ProducerRecyclerView)
        adapter = ProducerAdapter(mainList)
        val layoutManager = LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false)

        rv.layoutManager = layoutManager
        rv.itemAnimator = DefaultItemAnimator()
        rv.adapter = adapter // Set the previously
initialized adapter here

        databaseReference =
FirebaseDatabase.getInstance().getReference("Users")
        loadData()
    }

    override fun onCreateOptionsMenu(menu: Menu?):
Boolean {
        menuInflater.inflate(R.menu.user_menu, menu)
        return true
    }

    override fun onOptionsItemSelected(item:
MenuItem): Boolean {
        when (item.itemId) {
            R.id.action_profile -> {
                return true
            }
            R.id.action_appointment -> {
                val intent =
Intent(this, Client_booked_appointment::class.java)

```

```

        startActivity(intent)
        return true
    }
    R.id.action_logout -> {
        val intent =
Intent(this, Login::class.java)
        startActivity(intent)
        return true
    }
    else -> return
super.onOptionsItemSelected(item)
    }
    }
    private fun
filterBySpecialization(specialization: String) {
        if (specialization.isEmpty()) {
            // If the filter is empty, load all data
            loadData()
        } else {
            // Filter the mainList based on the
specialization
            val filteredList = mainList.filter {
it.specialization == specialization }
            adapter.updateList(filteredList) //
Update RecyclerView with filtered list
        }
    }

    private fun loadData() {

databaseReference.addValueEventListener(object :
ValueEventListener {
        override fun onDataChange(snapshot:
DataSnapshot) {
            mainList.clear() // Clear the list to
avoid duplicates

            for (userSnapshot in
snapshot.children) {
                val user =
userSnapshot.getValue(Producer::class.java)
                if (user?.role == "producer") {
                    mainList.add(user)
                }
            }
        }
    })
    }

```



```
        }
        adapter.notifyDataSetChanged() //
Notify the adapter of the data change
    }

    override fun onCancelled(error:
DatabaseError) {
        // Handle any errors here
        Log.e("Firebase", "Data retrieval
failed: $error")
    }
})
}
}
```

SignUp.kt

```
package com.example.locatemyProducer

import android.content.Intent
import android.graphics.drawable.ColorDrawable
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.Switch
import android.widget.TextView
import android.widget.Toast
import androidx.cardview.widget.CardView
import androidx.core.content.ContextCompat
import com.bumptech.glide.Glide
import com.google.firebase.FirebaseException
import com.google.firebase.auth.FirebaseAuth
import
com.google.firebase.auth.FirebaseAuthInvalidCredentialException
import
com.google.firebase.auth.FirebaseAuthUserCollisionException
import
com.google.firebase.auth.FirebaseAuthWeakPasswordException
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.StorageReference
import com.google.firebase.storage.UploadTask
import de.hdodenhof.circleimageview.CircleImageView
import kotlin.random.Random

class SignUp : AppCompatActivity() {
    private lateinit var firebaseAuth: FirebaseAuth
    private lateinit var nameEditText: EditText
    private lateinit var loginText: TextView
    private lateinit var roleSwitch :Switch
```

```

        private lateinit var emailEditText: EditText
        private lateinit var phoneEditText: EditText
        private lateinit var passwordEditText: EditText
        private lateinit var confirmPasswordEditText:
EditText
        private lateinit var registerButton: Button
        private lateinit var db: FirebaseDatabase
        lateinit var databaseReference: DatabaseReference
        private lateinit var storageReference:
StorageReference

        lateinit var circleImage: CircleImageView
        lateinit var linkImg: String
        lateinit var role :String
        override fun onCreate(savedInstanceState:
Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_sign_up)

            supportActionBar?.setBackgroundDrawable(ColorDrawable
(ContextCompat.getColor(this, R.color.primary)))
            window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

            nameEditText =
findViewById(R.id.editTextName)
            emailEditText =
findViewById(R.id.editTextEmail)
            phoneEditText =
findViewById(R.id.editTextPhone)
            passwordEditText =
findViewById(R.id.editTextPassword)
            confirmPasswordEditText =
findViewById(R.id.editTextConfirmPassword)
            registerButton =
findViewById(R.id.buttonRegister)
            firebaseAuth = FirebaseAuth.getInstance()
            db = FirebaseDatabase.getInstance()
            circleImage = findViewById(R.id.profile)
            linkImg=""
            role="client"
            loginText = findViewById(R.id.loginText)
            roleSwitch = findViewById(R.id.roleSwitch)

```

```

        circleImage.setOnClickListener{
            intent = Intent()
            intent.setType("image/*")

            intent.setAction(Intent.ACTION_GET_CONTENT)

            startActivityResult(Intent.createChooser(intent, "Choose the image"), 1)

        }

        roleSwitch.setOnCheckedChangeListener { _,
isChecked ->
            role = if (isChecked) "producer" else
"client"
        }

        registerButton.setOnClickListener {
            val name = nameEditText.text.toString()
            val email = emailEditText.text.toString()
            val phone = phoneEditText.text.toString()
            val password =
passwordEditText.text.toString()
            val confirmPassword =
confirmPasswordEditText.text.toString()

            if(email.isNotEmpty() &&
password.isNotEmpty() && name.isNotEmpty() &&
confirmPassword.isNotEmpty() && linkImg.isNotEmpty()
&& phone.isNotEmpty()) {

                firebaseAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener {

                    if (it.isSuccessful) {
                        Toast.makeText(this, "User
created", Toast.LENGTH_SHORT).show()

                    }

                }

                realtimeDatabase(name, email, password, phone, role)
            }
        }
    }
}

```

```

        var i: Intent = Intent(this,
Login::class.java)
        startActivity(i)
    } else {
        try {
            throw it.exception()!!
        }
        catch( e:
FirebaseAuthInvalidCredentialsException) {

Toast.makeText(this,"Invalid Credentials Check your
email", Toast.LENGTH_SHORT).show()
        }catch (e:
FirebaseAuthWeakPasswordException) {
            Toast.makeText(this,
"Weak Paasword", Toast.LENGTH_SHORT).show()
        } catch (e:
FirebaseAuthUserCollisionException) {
            Toast.makeText(this,
"Some error! Please try again"+e.toString(),
Toast.LENGTH_SHORT).show()
        } catch (e: Exception) {

            Toast.makeText(
                this,

it.exception.toString(),

                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
else {
    Toast.makeText(this,"Empty Fields are
not allowed", Toast.LENGTH_SHORT).show()
}
}

loginText.setOnClickListener{
    val intent :Intent =
Intent(this,Login::class.java)
    startActivity(intent)
}

```

```

    }
    fun realtimeDatabase(nam: String, em: String,
pass: String,phone:String,role:String) {
        Toast.makeText(this, linkImg,
Toast.LENGTH_LONG).show()
        val sanitizedEm = em.replace(".",
" ").replace("#", "_").replace("$", "_").replace("[",
"_").replace("]", "_")
        val reference =
FirebaseDatabase.getInstance().getReference("Users").
child(sanitizedEm)
        // Check the role and create an instance of
the corresponding model
        val users = if (role == "producer") {
            val specialization = "" // Add
specialization as needed
            val licenseNumber = "" // Add
licenseNumber as needed
            val about = ""
            val tracksProduced = Random.nextInt(1,
101)
            val rating = Random.nextDouble(0.1, 5.0)
            val location = ""
            val experience = ""
            Producer(specialization, licenseNumber,
about ,tracksProducedWon.toInt() ,rating.toFloat()
,location , experience ,nam, em, pass, linkImg,
phone, role )
        } else {
            User(nam, em, pass, linkImg, phone ,role)
        }

reference.setValue(users).addOnCompleteListener {
task ->
    try {
        if (task.isSuccessful) {
            Toast.makeText(this, "Your data
saved", Toast.LENGTH_SHORT).show()
        } else {
            throw task.exception ?:
Exception("Unknown error")
        }
    } catch (e: Exception) {
        if (e is FirebaseException) {

```

```

        // Handle Firebase-related
        exceptions
        Log.e("Firebase Error -----
>",e.message.toString())

        Toast.makeText(this, "Firebase
Error: ${e.message}", Toast.LENGTH_SHORT).show()
    } else {
        // Handle other exceptions
        Log.e("Error -----
>",e.message.toString())
        Toast.makeText(this, "Error:
${e.message}", Toast.LENGTH_SHORT).show()
    }
}

}

override fun onActivityResult(requestCode: Int,
resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode,
resultCode, data)
    if(requestCode==1 && data!=null &&
resultCode== RESULT_OK && data.getData()!=null){

Glide.with(this).load(data.data).into(circleImage)
        onImageAdd(data.data)
    }
}

fun onImageAdd(imageUri: Uri?) {
    if (imageUri != null) {
        // Set the desired filename for the
        uploaded image
        storageReference =

FirebaseStorage.getInstance().reference.child("Images
/"+System.currentTimeMillis()+".jpg")

        storageReference.putFile(imageUri)
            .addOnSuccessListener { taskSnapshot:
UploadTask.TaskSnapshot? ->
                storageReference.downloadUrl
                    .addOnSuccessListener {

```

```

downloadUrl: Uri ->
    linkImg =
downloadUrl.toString()
    }
    .addOnFailureListener { e:
Exception? ->
    }
    }
    .addOnFailureListener { e: Exception?
->
    // Handle any errors if the image
upload fails
    }
    }
}

```


SplashActivity.kt

```
package com.example.looperman
import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.ImageView
import android.widget.TextView

class SplashActivity : Activity() {
    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        val imageView =
findViewById<ImageView>(R.id.imageView)
        val textView =
findViewById<TextView>(R.id.textView)

        // Apply fade-in animation to the ImageView
        val fadeIn =
AnimationUtils.loadAnimation(this, R.anim.fade_in)
        imageView.startAnimation(fadeIn)

        // Apply fade-out animation to the ImageView
        val fadeOut =
AnimationUtils.loadAnimation(this, R.anim.fade_out)
        fadeIn.setAnimationListener(object :
Animation.AnimationListener {
            override fun onAnimationStart(animation:
Animation) {
                // Animation started
            }

            override fun onAnimationEnd(animation:
Animation) {
                // Animation ended
                imageView.startAnimation(fadeOut)
                // After fade-out animation ends,
```

```

start the LoginActivity
        fadeOut.setAnimationListener(object :
Animation.AnimationListener {
            override fun
onAnimationStart(animation: Animation) {
                // Animation started
            }

            override fun
onAnimationEnd(animation: Animation) {
                // Animation ended
            }

startActivity(Intent(this@SplashActivity,
Login::class.java))
                finish() // Finish current
activity
            }

            override fun
onAnimationRepeat(animation: Animation) {
                // Animation repeated
            }
        })
    }

    override fun onAnimationRepeat(animation:
Animation) {
        // Animation repeated
    }
})
}
}

```

ProducerDetailsUpdate.kt

```
package com.example.looperman

import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.drawable.ColorDrawable
import android.location.Address
import android.location.Geocoder
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.bumptech.glide.Glide
import com.bumptech.glide.load.engine.DiskCacheStrategy
import com.bumptech.glide.request.RequestOptions
import com.google.android.gms.location.FusedLocationProvider
Client
import com.google.android.gms.location.LocationServices
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import de.hdodenhof.circleimageview.CircleImageView
import java.util.Locale
import kotlin.reflect.TypeOf

class updateDetails : AppCompatActivity() {
    private lateinit var nameEditText: TextView
    private lateinit var specializationEditText:
EditText
    private lateinit var licenseNumberEditText:
EditText
    private lateinit var aboutEditText: EditText
    private lateinit var tracksProducedEditText:
EditText
```

```

        private lateinit var locationEditText: EditText
        private lateinit var updateButton: Button
        private lateinit var profile : CircleImageView
        private lateinit var experienceEditText:EditText

        private lateinit var fusedLocationClient:
FusedLocationProviderClient

        private var name:String=""
        private var specialization:String=""
        private var licenseNumber:String=""
        private var about:String=""
        private var tracksProduced:String=""
        private var location:String=""
        private var experience:String=""

        private var email:String=""

        private lateinit var dbRef: DatabaseReference
        lateinit var producer: Producer

        companion object {
            const val
PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION = 100
        }

        override fun onCreate(savedInstanceState:
Bundle?) {
            super.onCreate(savedInstanceState)

            setContentView(R.layout.activity_update_details)

            supportActionBar?.setBackgroundDrawable(ColorDrawable
(ContextCompat.getColor(this, R.color.primary)))
            window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

            supportActionBar?.setDisplayHomeAsUpEnabled(true)

            fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)

```

```

        // Initializing views
        nameEditText =
findViewById(R.id.nameEditText)
        specializationEditText =
findViewById(R.id.specializationEditText)
        licenseNumberEditText =
findViewById(R.id.licenseNumberEditText)
        aboutEditText =
findViewById(R.id.aboutEditText)
        tracksProducedEditText =
findViewById(R.id.casesWonEditText)
        locationEditText =
findViewById(R.id.locationEditText)
        updateButton =
findViewById(R.id.updateButton)
        profile = findViewById(R.id.profile)
        experienceEditText =
findViewById(R.id.experienceEditText)

        // Assigning retrieved values to TextViews

specializationEditText.setText(intent.getStringExtra(
"specialization"))

nameEditText.setText(intent.getStringExtra("name"))

licenseNumberEditText.setText(intent.getStringExtra("
licenseNumber"))

aboutEditText.setText(intent.getStringExtra("about"))

tracksProducedEditText.setText(intent.getStringExtra(
"tracksProduced"))

locationEditText.setText(intent.getStringExtra("locat
ion"))
        email =
intent.getStringExtra("sanitizedEm").toString()

experienceEditText.setText(intent.getStringExtra("exp
erience"))

        producer = Producer()

```

```

        // Fetch current location after views are
        initialized
        fetchCurrentLocation()

Glide.with(applicationContext).load(intent.getStringExtra("image"))
        .apply(
            RequestOptions()
                .error(R.drawable.user)
                .placeholder(R.drawable.user)

        .diskCacheStrategy(DiskCacheStrategy.ALL)
            ).into(profile)

        // Initialize click listener for update
        button
            updateButton.setOnClickListener {
                specialization =
specializationEditText.text.toString()
                licenseNumber =
licenseNumberEditText.text.toString()
                about = aboutEditText.text.toString()
                tracksProduced =
tracksProducedEditText.text.toString()
                location =
locationEditText.text.toString()
                experience =
experienceEditText.text.toString()
                updateUserDetails(name, specialization,
licenseNumber, about, tracksProduced, location ,
experience)
            }
        }

        private fun updateUserDetails(name:String,
specialization:String
,licenseNumber:String,about:String,caseWon:String ,
location: String ,experience:String){
            dbRef =
FirebaseDatabase.getInstance().getReference("Users")

            val updates = HashMap<String, Any>()
            updates["specialization"] = specialization

```

```

        updates["licenseNumber"] = licenseNumber
        updates["about"] = about
        updates["tracksProduced"] = caseWon.toInt()
        updates["location"] = location
        updates["experience"] = experience

    dbRef.child(email).updateChildren(updates).addOnCompleteListener{
        Toast.makeText(this, "Updated Successfully", Toast.LENGTH_SHORT).show()
        val intent =
            Intent(this, ProducerActivity::class.java)
            startActivity(intent)
        }.addOnFailureListener{ error ->
            Toast.makeText(this, "Deleting Error ${error.message}", Toast.LENGTH_SHORT).show()
        }
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

    private fun fetchCurrentLocation() {
        // Check location permissions
        if (ContextCompat.checkSelfPermission(
            this,
            android.Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED
        ) {
            // Permission is not granted
            ActivityCompat.requestPermissions(
                this,
                arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),
                PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION
            )
        } else {
            // Permission has already been granted

```

```

        fusedLocationClient.lastLocation
            .addOnSuccessListener { location ->
                // Got last known location. In
                some rare situations this can be null.
                if (location != null) {
                    val geocoder = Geocoder(this,
Locale.getDefault())
                    val list: List<Address> =

geocoder.getFromLocation(location.latitude,
location.longitude, 1)!!
                    // Assign location
                    coordinates to the locationEditText field

locationEditText.setText(list[0].getAddressLine(0).to
String())

                    } else {
                        // Location is null
                        Toast.makeText(
                            this,
                            "Unable to retrieve
current location",
                                Toast.LENGTH_SHORT
                            ).show()
                    }
                }
            }.addOnFailureListener { e ->
                // Handle failure
                Toast.makeText(
                    this,
                    "Failed to retrieve current
location: ${e.message}",
                        Toast.LENGTH_SHORT
                    ).show()
            }
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode,

```



```

permissions, grantResults)
    when (requestCode) {
        PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION
-> {
            // If request is cancelled, the
            result arrays are empty.
            if ((grantResults.isNotEmpty() &&
grantResults[0] ==
PackageManager.PERMISSION_GRANTED)) {
                // Permission granted, fetch the
                current location
                fetchCurrentLocation()
            } else {
                // Permission denied
                Toast.makeText(
                    this,
                    "Permission denied. Unable to
retrieve current location",
                    Toast.LENGTH_SHORT
                ).show()
            }
            return
        }
    }
}
}
}

```

Client.kt

```
package com.example.looperman

open class User {
    var name: String? = null
    var email: String? = null
    var pass: String? = null
    var image: String? = null
    var phone:String?=null
    var role: String?=null

    // Add a no-argument constructor
    constructor()

    constructor(name: String, email: String, pass:
String, image: String , phone:String ,role:String) {
        this.name = name
        this.email = email
        this.pass = pass
        this.image = image
        this.phone = phone
        this.role = role
    }
}
```

ViewAppointments.kt

```
package com.example.looperman

import android.content.Context
import android.content.SharedPreferences
import android.graphics.drawable.ColorDrawable
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.content.ContextCompat
import
androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.*

class ViewAppointments : AppCompatActivity() {

    private lateinit var appointmentsRecyclerView:
RecyclerView
    private lateinit var appointmentAdapter:
AppointmentAdapter
    private lateinit var auth: FirebaseAuth
    private lateinit var database: DatabaseReference
    private lateinit var sharedPreferences:
SharedPreferences

    private var producerId : String = ""

    override fun onCreate(savedInstanceState:
Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_view_appointments)

        supportActionBar?.setBackgroundDrawable(ColorDrawable
(ContextCompat.getColor(this, R.color.primary)))
    }
}
```

```

        window?.statusBarColor =
ContextCompat.getColor(this, R.color.primary)

supportActionBar?.setDisplayHomeAsUpEnabled(true)

        auth = FirebaseAuth.getInstance()
        database =
FirebaseDatabase.getInstance().reference

        appointmentsRecyclerView =
findViewById(R.id.appointmentsRecyclerView)
        appointmentsRecyclerView.layoutManager =
LinearLayoutManager(this)
        appointmentAdapter =
AppointmentAdapter(emptyList(), database)
        appointmentsRecyclerView.adapter =
appointmentAdapter

        sharedPreferences =
getSharedPreferences("userData",
Context.MODE_PRIVATE)
        producerId =
sharedPreferences.getString("username",
"").toString()

        fetchAppointmentsForProducer(producerId ?:
"")
    }

    private fun
fetchAppointmentsForProducer(producerId: String) {

        val query: Query =
database.child("appointments").orderByChild("producer
Id").equalTo(producerId)
        query.addListenerForSingleValueEvent(object :
ValueEventListener {
            override fun onDataChange(dataSnapshot:
DataSnapshot) {
                val appointmentsList =
mutableListOf<AppointmentModel>()

                if (dataSnapshot.exists()) {
                    for (appointmentSnapshot in

```

```

dataSnapshot.children) {
    val appointment =
appointmentSnapshot.getValue(AppointmentModel::class.
java)
        appointment?.let {
            appointmentsList.add(it)
        }
    }
    appointmentAdapter.appointments =
appointmentsList

appointmentAdapter.notifyDataSetChanged()
    }
}

    override fun onCancelled(databaseError:
DatabaseError) {
        // Handle database error
    }
}
}
}
}

```

Conclusion

In conclusion, "Looper Man" emerges not just as a mere application, but as a dynamic force driving transformation within the Indian music industry. By seamlessly connecting aspiring musicians with seasoned producers, fostering collaboration, and amplifying creativity, "Looper Man" transcends the boundaries of traditional music production platforms.

Through its intuitive interface, robust feature set, and commitment to inclusivity, "Looper Man" empowers individuals from all walks of life to realize their musical aspirations and forge meaningful connections within a vibrant community of artists and enthusiasts. As users embark on their musical journeys within the "Looper Man" ecosystem, they become architects of a new paradigm, where barriers dissolve, collaboration flourishes, and the transformative power of music knows no limits.

In essence, "Looper Man" is more than just an app; it's a catalyst for change, a platform for innovation, and a celebration of the rich tapestry of Indian music. As we look to the future, let us embrace the spirit of "Looper Man" and embark on a collective journey of sonic exploration, artistic discovery, and collaborative creation. Together, let us shape the melody of tomorrow and leave an indelible mark on the ever-evolving landscape of music production in India and beyond.