**AndroidManifest.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
 <uses-permission android:name="android.permission.INTERNET"/>  
  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.TransactionApp"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.TransactionApp">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Transaction.kt:**

package com.example.transactionapp.data  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "transactions")  
data class Transaction(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val amount: Double,  
 val status: String,  
 val timestamp: Long  
)

**TransactionDao.kt:**

package com.example.transactionapp.data  
  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import kotlinx.coroutines.flow.Flow  
  
@Dao  
interface TransactionDao {  
 @Insert  
 suspend fun insertTransaction(transaction: Transaction)  
  
 @Query("SELECT \* FROM transactions ORDER BY timestamp DESC")  
 fun getAllTransactions(): Flow<List<Transaction>>  
}

**TransactionDatabase.kt:**

package com.example.transactionapp.data  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
  
@Database(entities = [Transaction::class], version = 1, exportSchema = false)  
abstract class TransactionDatabase : RoomDatabase() {  
 abstract fun transactionDao(): TransactionDao  
  
 companion object {  
 @Volatile  
 private var INSTANCE: TransactionDatabase? = null  
  
 fun getDatabase(context: Context): TransactionDatabase {  
 return INSTANCE ?: synchronized(this) **{** val instance = Room.databaseBuilder(  
 context.applicationContext,  
 TransactionDatabase::class.java,  
 "transaction\_database"  
 ).build()  
 INSTANCE = instance  
 instance  
 **}** }  
 }  
}

**TransactionRepository.kt:**

package com.example.transactionapp.repository  
  
import com.example.transactionapp.data.Transaction  
import com.example.transactionapp.data.TransactionDao  
import kotlinx.coroutines.flow.Flow  
  
class TransactionRepository(private val dao: TransactionDao) {  
 suspend fun insertTransaction(transaction: Transaction) = dao.insertTransaction(transaction)  
 fun getAllTransactions(): Flow<List<Transaction>> = dao.getAllTransactions()  
}

**TransactionScreen.kt:**

package com.example.transactionapp.ui  
  
import android.app.Activity  
import android.content.Context  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material3.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.platform.LocalContext  
import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import com.razorpay.Checkout  
import org.json.JSONObject  
  
@Preview(showSystemUi = true)  
@Composable  
fun TransactionScreen() {  
 val context = LocalContext.current  
 val activity = context as? Activity ?: return // Ensure context is an Activity  
 var amount by remember **{** mutableStateOf("") **}** Column(  
 modifier = Modifier  
 .fillMaxSize()  
 .padding(16.dp)  
 ) **{** OutlinedTextField(  
 value = amount,  
 onValueChange = **{** amount = it **}**,  
 label = **{** Text("Enter Amount") **}**,  
 modifier = Modifier.fillMaxWidth()  
 )  
  
 Button(  
 onClick = **{** val amountValue = amount.toDoubleOrNull()  
 if (amountValue != null && amountValue > 0) {  
 startPayment(activity, amountValue)  
 } else {  
 // Handle invalid input (optional: show a Toast message)  
 }  
 **}**,  
 modifier = Modifier.padding(top = 16.dp)  
 ) **{** Text("Pay Now")  
 **}  
 }**}  
  
fun startPayment(activity: Activity, amount: Double) {  
 try {  
 val checkout = Checkout()  
 checkout.setKeyID("rzp\_test\_zR8qUls1pk3om4") // Replace with your Razorpay API Key  
  
 val paymentDetails = JSONObject().apply **{** put("name", "Your Business Name")  
 put("description", "Payment for XYZ")  
 put("currency", "INR")  
 put("amount", (amount \* 100).toInt()) // Convert to paisa  
 **}** checkout.open(activity, paymentDetails)  
 } catch (e: Exception) {  
 e.printStackTrace()  
 }  
}  
  
@Preview(showSystemUi = true)  
@Composable  
fun TransactionScreenPreview() {  
 TransactionScreen()  
}

**TransactionViewModel.kt:**

package com.example.transactionapp.ui  
  
import android.app.Application  
import androidx.lifecycle.AndroidViewModel  
import androidx.lifecycle.*viewModelScope*import com.example.transactionapp.data.Transaction  
import com.example.transactionapp.data.TransactionDatabase  
import com.example.transactionapp.repository.TransactionRepository  
import kotlinx.coroutines.flow.Flow  
import kotlinx.coroutines.launch  
  
class TransactionViewModel(application: Application) : AndroidViewModel(application) {  
 private val repository: TransactionRepository  
  
 init {  
 val dao = TransactionDatabase.getDatabase(application).transactionDao()  
 repository = TransactionRepository(dao)  
 }  
  
 val transactions: Flow<List<Transaction>> = repository.getAllTransactions()  
  
 fun addTransaction(amount: Double, status: String) {  
 *viewModelScope*.*launch* **{** val transaction = Transaction(amount = amount, status = status, timestamp = System.currentTimeMillis())  
 repository.insertTransaction(transaction)  
 **}** }  
}

**MainActivity.kt:**

package com.example.transactionapp  
  
import android.os.Bundle  
import android.widget.Toast  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import com.example.transactionapp.ui.TransactionScreen  
import com.example.transactionapp.ui.TransactionViewModel  
import com.razorpay.PaymentResultListener  
  
class MainActivity : ComponentActivity(), PaymentResultListener {  
 private lateinit var viewModel: TransactionViewModel  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 viewModel = TransactionViewModel(application)  
  
 setContent **{** TransactionScreen()  
 **}** }  
  
 override fun onPaymentSuccess(razorpayPaymentId: String?) {  
 viewModel.addTransaction(amount = 100.0, status = "Success") // Dummy amount, update accordingly  
 Toast.makeText(this, "Payment Successful!", Toast.LENGTH\_SHORT).show()  
 }  
  
 override fun onPaymentError(code: Int, response: String?) {  
 viewModel.addTransaction(amount = 100.0, status = "Failed") // Dummy amount, update accordingly  
 Toast.makeText(this, "Payment Failed!", Toast.LENGTH\_SHORT).show()  
 }  
}

AndroidManifest.xml :

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools">  
 <uses-permission android:name="android.permission.INTERNET"/>  
  
  
 <application  
 android:allowBackup="true"  
 android:dataExtractionRules="@xml/data\_extraction\_rules"  
 android:fullBackupContent="@xml/backup\_rules"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.TransactionApp"  
 tools:targetApi="31">  
 <activity  
 android:name=".MainActivity"  
 android:exported="true"  
 android:label="@string/app\_name"  
 android:theme="@style/Theme.TransactionApp">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

Doctor.kt:

package com.example.transactionapp.data  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "doctors")  
data class Doctor(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val name: String,  
 val upiId: String  
)

DoctorDao.kt:

package com.example.transactionapp.data  
  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import kotlinx.coroutines.flow.Flow  
  
@Dao  
interface DoctorDao {  
 @Insert  
 suspend fun insertDoctor(doctor: Doctor)  
  
 @Query("SELECT \* FROM doctors")  
 fun getAllDoctors(): Flow<List<Doctor>>  
  
 @Query("SELECT \* FROM doctors WHERE id = :doctorId")  
 fun getDoctorById(doctorId: Int): Flow<Doctor>  
}

Transaction.kt:

package com.example.transactionapp.data  
  
import androidx.room.Entity  
import androidx.room.PrimaryKey  
  
@Entity(tableName = "transactions")  
data class UserTransaction(  
 @PrimaryKey(autoGenerate = true) val id: Int = 0,  
 val amount: Double,  
 val status: String,  
 val timestamp: Long  
)

TransactionDao.kt:

package com.example.transactionapp.data  
  
import androidx.room.Dao  
import androidx.room.Insert  
import androidx.room.Query  
import androidx.room.Transaction  
import kotlinx.coroutines.flow.Flow  
  
@Dao  
interface TransactionDao {  
 @Insert  
 suspend fun insertTransaction(transaction: UserTransaction)  
  
 @Query("SELECT \* FROM transactions ORDER BY timestamp DESC")  
 fun getAllTransactions(): Flow<List<UserTransaction>>  
}

TransactionDatabase.kt:

package com.example.transactionapp.data  
  
import android.content.Context  
import androidx.room.Database  
import androidx.room.Room  
import androidx.room.RoomDatabase  
//import androidx.room.Transaction  
  
@Database(entities = [UserTransaction::class,Doctor::class], version = 1, exportSchema = false)  
abstract class TransactionDatabase : RoomDatabase() {  
 abstract fun transactionDao(): TransactionDao  
 abstract fun doctorDao():DoctorDao  
  
 companion object {  
 @Volatile  
 private var INSTANCE: TransactionDatabase? = null  
  
 fun getDatabase(context: Context): TransactionDatabase {  
 return INSTANCE ?: *synchronized*(this) **{** val instance = Room.databaseBuilder(  
 context.*applicationContext*,  
 TransactionDatabase::class.*java*,  
 "transaction\_database"  
 ).build()  
 INSTANCE = instance  
 instance  
 **}** }  
 }  
}

TransactionRepository.kt:

package com.example.transactionapp.repository  
  
import com.example.transactionapp.data.Doctor  
import com.example.transactionapp.data.UserTransaction  
import com.example.transactionapp.data.TransactionDao  
import com.example.transactionapp.data.DoctorDao  
import kotlinx.coroutines.flow.Flow  
  
class TransactionRepository(private val transactionDao: TransactionDao, private val doctorDao: DoctorDao) {  
 suspend fun insertTransaction(transaction: UserTransaction) = transactionDao.insertTransaction(transaction)  
 fun getAllTransactions(): Flow<List<UserTransaction>> = transactionDao.getAllTransactions()  
  
 suspend fun insertDoctor(doctor: Doctor) = doctorDao.insertDoctor(doctor)  
 fun getAllDoctors(): Flow<List<Doctor>> = doctorDao.getAllDoctors()  
}

TransactionScreen.kt:

package com.example.transactionapp.ui  
  
import android.app.Activity  
import android.app.Application  
import android.content.Intent  
import android.net.Uri  
import android.util.Log  
import androidx.compose.foundation.layout.\*  
import androidx.compose.material3.\*  
import androidx.compose.runtime.\*  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.platform.*LocalContext*import androidx.compose.ui.tooling.preview.Preview  
import androidx.compose.ui.unit.dp  
import com.example.transactionapp.data.Doctor  
import kotlinx.coroutines.flow.Flow  
  
@Composable  
fun TransactionScreen(viewModel: TransactionViewModel) {  
 val context = *LocalContext*.current  
 val activity = context as? Activity ?: return  
 var amount by remember **{** *mutableStateOf*("") **}** var selectedDoctor by remember **{** *mutableStateOf*<Doctor?>(null) **}** val doctorList by viewModel.doctors.collectAsState(initial = *emptyList*())  
  
 var expanded by remember **{** *mutableStateOf*(false) **}** Column(modifier = Modifier.*fillMaxSize*().*padding*(16.*dp*)) **{** OutlinedTextField(  
 value = amount,  
 onValueChange = **{** amount = **it }**,  
 label = **{** Text("Enter Amount") **}**,  
 modifier = Modifier.*fillMaxWidth*()  
 )  
  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 // Dropdown to Select Doctor  
 Button(onClick = **{** expanded = true **}**) **{** Text(selectedDoctor?.name ?: "Select Doctor")  
 **}** DropdownMenu(expanded = expanded, onDismissRequest = **{** expanded = false **}**) **{** doctorList.*forEach* **{** doctor **->** DropdownMenuItem(  
 text = **{** Text(doctor.name) **}**,  
 onClick = **{** selectedDoctor = doctor  
 expanded = false  
 **}** )  
 **}  
 }** Spacer(modifier = Modifier.*height*(16.*dp*))  
  
 Button(  
 onClick = **{** val amountValue = amount.*toDoubleOrNull*()  
 if (amountValue != null && amountValue > 0 && selectedDoctor != null) {  
 *startUPIPayment*(activity, amountValue, selectedDoctor!!.upiId)  
 }  
 **}**,  
 modifier = Modifier.*padding*(top = 16.*dp*),  
 enabled = amount.*isNotBlank*() && selectedDoctor != null  
 ) **{** Text("Pay Now")  
 **}  
 }**}  
  
fun startUPIPayment(activity: Activity, amount: Double, upiId: String) {  
 val formattedAmount = String.*format*("%.2f", amount)  
  
 val uri = Uri.Builder()  
 .scheme("upi")  
 .authority("pay")  
 .appendQueryParameter("pa", upiId) // Payee UPI ID  
 .appendQueryParameter("pn", "Doctor") // Payee Name  
 .appendQueryParameter("tn", "Doctor Payment") // Transaction Note  
 .appendQueryParameter("am", formattedAmount) // Amount  
 .appendQueryParameter("cu", "INR") // Currency  
 .build()  
  
 Log.d("UPI\_PAYMENT", "Generated URI: $uri")  
  
 val intent = Intent(Intent.*ACTION\_VIEW*, uri)  
 val chooser = Intent.createChooser(intent, "Pay with UPI")  
 activity.startActivity(chooser)  
}  
  
@Preview(showSystemUi = true)  
@Composable  
fun TransactionScreenPreview() {  
 TransactionScreen(viewModel = TransactionViewModel(Application()))  
}

TransactionViewModel:

package com.example.transactionapp.ui  
  
import android.app.Application  
import androidx.lifecycle.AndroidViewModel  
import androidx.lifecycle.*viewModelScope*import androidx.room.Transaction  
import com.example.transactionapp.data.Doctor  
import com.example.transactionapp.data.UserTransaction  
import com.example.transactionapp.data.TransactionDatabase  
import com.example.transactionapp.repository.TransactionRepository  
import kotlinx.coroutines.flow.Flow  
import kotlinx.coroutines.launch  
  
class TransactionViewModel(application: Application) : AndroidViewModel(application) {  
 private val repository: TransactionRepository  
 val transactions: Flow<List<UserTransaction>>  
 val doctors: Flow<List<Doctor>>  
  
 init {  
 val db = TransactionDatabase.getDatabase(application)  
 repository = TransactionRepository(db.transactionDao(), db.doctorDao())  
 transactions = repository.getAllTransactions()  
 doctors = repository.getAllDoctors()  
 }  
  
 fun addTransaction(amount: Double, status: String) {  
 *viewModelScope*.*launch* **{** val transaction = UserTransaction(amount = amount, status = status, timestamp = System.currentTimeMillis())  
 repository.insertTransaction(transaction)  
 **}** }  
  
 fun addDoctor(name: String, upiId: String) {  
 *viewModelScope*.*launch* **{** val doctor = Doctor(name = name, upiId = upiId)  
 repository.insertDoctor(doctor)  
 **}** }  
  
 fun processTransaction(amountInPaisa: Int) {  
 val amountInRupees = amountInPaisa / 100.0 // Convert back to rupees for database storage  
  
 *viewModelScope*.*launch* **{** val transaction = UserTransaction(  
 amount = amountInRupees,  
 status = "Pending",  
 timestamp = System.currentTimeMillis()  
 )  
 repository.insertTransaction(transaction)  
 **}** }  
}

MainActivity:

package com.example.transactionapp  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import com.example.transactionapp.ui.TransactionScreen  
import com.example.transactionapp.ui.TransactionViewModel  
  
class MainActivity : ComponentActivity() {  
 private lateinit var viewModel: TransactionViewModel  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 viewModel = TransactionViewModel(*application*)  
  
 *setContent* **{** TransactionScreen(viewModel = viewModel)  
 **}** }  
}