User.kt

|  |
| --- |
|  |
|  |
|  |  |
|  | import androidx.room.ColumnInfo |
|  | import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "user\_table") |
|  | data class User( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "first\_name") val firstName: String?, |
|  | @ColumnInfo(name = "last\_name") val lastName: String?, |
|  | @ColumnInfo(name = "email") val email: String?, |
|  | @ColumnInfo(name = "password") val password: String?, |
|  |  |
|  | ) |
|  |  |

UserDao.kt

|  |
| --- |
|  |
|  |  |
|  | import androidx.room.\* |
|  |  |
|  | @Dao |
|  | interface UserDao { |
|  |  |
|  | @Query("SELECT \* FROM user\_table WHERE email = :email") |
|  | suspend fun getUserByEmail(email: String): User? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertUser(user: User) |
|  |  |
|  | @Update |
|  | suspend fun updateUser(user: User) |
|  |  |
|  | @Delete |
|  | suspend fun deleteUser(user: User) |
|  | } |

UserDatabase.kt

|  |
| --- |
|  |
|  |
|  |
|  | import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [User::class], version = 1) |
|  | abstract class UserDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun userDao(): UserDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: UserDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): UserDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | UserDatabase::class.java, |
|  | "user\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

UserDatabaseHelpter.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class UserDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) { |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "UserDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "user\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_FIRST\_NAME = "first\_name" |
|  | private const val COLUMN\_LAST\_NAME = "last\_name" |
|  | private const val COLUMN\_EMAIL = "email" |
|  | private const val COLUMN\_PASSWORD = "password" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "$COLUMN\_ID INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "$COLUMN\_FIRST\_NAME TEXT, " + |
|  | "$COLUMN\_LAST\_NAME TEXT, " + |
|  | "$COLUMN\_EMAIL TEXT, " + |
|  | "$COLUMN\_PASSWORD TEXT" + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertUser(user: User) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_FIRST\_NAME, user.firstName) |
|  | values.put(COLUMN\_LAST\_NAME, user.lastName) |
|  | values.put(COLUMN\_EMAIL, user.email) |
|  | values.put(COLUMN\_PASSWORD, user.password) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getUserByUsername(username: String): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_FIRST\_NAME = ?", arrayOf(username)) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getUserById(id: Int): User? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var user: User? = null |
|  | if (cursor.moveToFirst()) { |
|  | user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return user |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllUsers(): List<User> { |
|  | val users = mutableListOf<User>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val user = User( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | firstName = cursor.getString(cursor.getColumnIndex(COLUMN\_FIRST\_NAME)), |
|  | lastName = cursor.getString(cursor.getColumnIndex(COLUMN\_LAST\_NAME)), |
|  | email = cursor.getString(cursor.getColumnIndex(COLUMN\_EMAIL)), |
|  | password = cursor.getString(cursor.getColumnIndex(COLUMN\_PASSWORD)), |
|  | ) |
|  | users.add(user) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return users |
|  | } |
|  |  |
|  | } |

Order.kt

|  |
| --- |
|  |
|  | Import android.room.ColumnInfor import androidx.room.Entity |
|  | import androidx.room.PrimaryKey |
|  |  |
|  | @Entity(tableName = "order\_table") |
|  | data class Order( |
|  | @PrimaryKey(autoGenerate = true) val id: Int?, |
|  | @ColumnInfo(name = "quantity") val quantity: String?, |
|  | @ColumnInfo(name = "address") val address: String?, |
|  | ) |

OrderDao.kt

|  |
| --- |
|  |
|  |  |
|  | Import android.room.\*  @Dao |
|  | interface OrderDao { |
|  |  |
|  | @Query("SELECT \* FROM order\_table WHERE address= :address") |
|  | suspend fun getOrderByAddress(address: String): Order? |
|  |  |
|  | @Insert(onConflict = OnConflictStrategy.REPLACE) |
|  | suspend fun insertOrder(order: Order) |
|  |  |
|  | @Update |
|  | suspend fun updateOrder(order: Order) |
|  |  |
|  | @Delete |
|  | suspend fun deleteOrder(order: Order) |
|  | } |

OrderDatabase.kt

|  |
| --- |
|  |
|  | Import android.content.Context  import androidx.room.Database |
|  | import androidx.room.Room |
|  | import androidx.room.RoomDatabase |
|  |  |
|  | @Database(entities = [Order::class], version = 1) |
|  | abstract class OrderDatabase : RoomDatabase() { |
|  |  |
|  | abstract fun orderDao(): OrderDao |
|  |  |
|  | companion object { |
|  |  |
|  | @Volatile |
|  | private var instance: OrderDatabase? = null |
|  |  |
|  | fun getDatabase(context: Context): OrderDatabase { |
|  | return instance ?: synchronized(this) { |
|  | val newInstance = Room.databaseBuilder( |
|  | context.applicationContext, |
|  | OrderDatabase::class.java, |
|  | "order\_database" |
|  | ).build() |
|  | instance = newInstance |
|  | newInstance |
|  | } |
|  | } |
|  | } |
|  | } |

OrderDatabaseHelpter.kt

|  |
| --- |
|  |
|  | Import android.annotation.SuppressLint  import android.content.ContentValues |
|  | import android.content.Context |
|  | import android.database.Cursor |
|  | import android.database.sqlite.SQLiteDatabase |
|  | import android.database.sqlite.SQLiteOpenHelper |
|  |  |
|  | class OrderDatabaseHelper(context: Context) : |
|  | SQLiteOpenHelper(context, DATABASE\_NAME, null,DATABASE\_VERSION){ |
|  |  |
|  | companion object { |
|  | private const val DATABASE\_VERSION = 1 |
|  | private const val DATABASE\_NAME = "OrderDatabase.db" |
|  |  |
|  | private const val TABLE\_NAME = "order\_table" |
|  | private const val COLUMN\_ID = "id" |
|  | private const val COLUMN\_QUANTITY = "quantity" |
|  | private const val COLUMN\_ADDRESS = "address" |
|  | } |
|  |  |
|  | override fun onCreate(db: SQLiteDatabase?) { |
|  | val createTable = "CREATE TABLE $TABLE\_NAME (" + |
|  | "${COLUMN\_ID} INTEGER PRIMARY KEY AUTOINCREMENT, " + |
|  | "${COLUMN\_QUANTITY} Text, " + |
|  | "${COLUMN\_ADDRESS} TEXT " + |
|  | ")" |
|  |  |
|  | db?.execSQL(createTable) |
|  | } |
|  |  |
|  | override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) { |
|  | db?.execSQL("DROP TABLE IF EXISTS $TABLE\_NAME") |
|  | onCreate(db) |
|  | } |
|  |  |
|  | fun insertOrder(order: Order) { |
|  | val db = writableDatabase |
|  | val values = ContentValues() |
|  | values.put(COLUMN\_QUANTITY, order.quantity) |
|  | values.put(COLUMN\_ADDRESS, order.address) |
|  | db.insert(TABLE\_NAME, null, values) |
|  | db.close() |
|  | } |
|  |  |
|  |  |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getOrderByQuantity(quantity: String): Order? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_QUANTITY = ?", arrayOf(quantity)) |
|  | var order: Order? = null |
|  | if (cursor.moveToFirst()) { |
|  | order = Order( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | address = cursor.getString(cursor.getColumnIndex(COLUMN\_ADDRESS)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return order |
|  | } |
|  | @SuppressLint("Range") |
|  | fun getOrderById(id: Int): Order? { |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME WHERE $COLUMN\_ID = ?", arrayOf(id.toString())) |
|  | var order: Order? = null |
|  | if (cursor.moveToFirst()) { |
|  | order = Order( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | address = cursor.getString(cursor.getColumnIndex(COLUMN\_ADDRESS)), |
|  | ) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return order |
|  | } |
|  |  |
|  | @SuppressLint("Range") |
|  | fun getAllOrders(): List<Order> { |
|  | val orders = mutableListOf<Order>() |
|  | val db = readableDatabase |
|  | val cursor: Cursor = db.rawQuery("SELECT \* FROM $TABLE\_NAME", null) |
|  | if (cursor.moveToFirst()) { |
|  | do { |
|  | val order = Order( |
|  | id = cursor.getInt(cursor.getColumnIndex(COLUMN\_ID)), |
|  | quantity = cursor.getString(cursor.getColumnIndex(COLUMN\_QUANTITY)), |
|  | address = cursor.getString(cursor.getColumnIndex(COLUMN\_ADDRESS)), |
|  | ) |
|  | orders.add(order) |
|  | } while (cursor.moveToNext()) |
|  | } |
|  | cursor.close() |
|  | db.close() |
|  | return orders |
|  | } |
|  |  |
|  | } |

LoginActivity.kt

|  |
| --- |
|  |
|  | import android.content.Intent  import android.content.Context |
|  | import android.os.Bundle |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.text.font.FontFamily |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import androidx.core.content.ContextCompat |
|  | import com.example.snackordering.ui.theme.SnackOrderingTheme |
|  |  |
|  | class LoginActivity : ComponentActivity() { |
|  | private lateinit var databaseHelper: UserDatabaseHelper |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | databaseHelper = UserDatabaseHelper(this) |
|  | setContent { |
|  | SnackOrderingTheme { |
|  |  |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  | LoginScreen(this, databaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | @Composable |
|  | fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) { |
|  |  |
|  | Image(painterResource(id = R.drawable.order), contentDescription = "", |
|  | alpha =0.3F, |
|  | contentScale = ContentScale.FillHeight, |
|  |  |
|  | ) |
|  |  |
|  | var username by remember { mutableStateOf("") } |
|  | var password by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Column( |
|  | modifier = Modifier.fillMaxSize(), |
|  | horizontalAlignment = Alignment.CenterHorizontally, |
|  | verticalArrangement = Arrangement.Center |
|  | ) { |
|  |  |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | color = Color.White, |
|  | text = "Login" |
|  | ) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | TextField( |
|  | value = username, |
|  | onValueChange = { username = it }, |
|  | label = { Text("Username") }, |
|  | modifier = Modifier.padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  | TextField( |
|  | value = password, |
|  | onValueChange = { password = it }, |
|  | label = { Text("Password") }, |
|  | modifier = Modifier.padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button( |
|  | onClick = { |
|  | if (username.isNotEmpty() && password.isNotEmpty()) { |
|  | val user = databaseHelper.getUserByUsername(username) |
|  | if (user != null && user.password == password) { |
|  | error = "Successfully log in" |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | MainPage::class.java |
|  | ) |
|  | ) |
|  |  |
|  | } |
|  | if (user != null && user.password == "admin") { |
|  | error = "Successfully log in" |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | AdminActivity::class.java |
|  | ) |
|  | ) |
|  | } |
|  | else { |
|  | error = "Invalid username or password" |
|  | } |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | modifier = Modifier.padding(top = 16.dp) |
|  | ) { |
|  | Text(text = "Login") |
|  | } |
|  | Row { |
|  | TextButton(onClick = {context.startActivity( |
|  | Intent( |
|  | context, |
|  | MainActivity::class.java |
|  | ) |
|  | )} |
|  | ) |
|  | { Text(color = Color.White,text = "Sign up") } |
|  | TextButton(onClick = { |
|  | }) |
|  |  |
|  | { |
|  | Spacer(modifier = Modifier.width(60.dp)) |
|  | Text(color = Color.White,text = "Forget password?") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startMainPage(context: Context) { |
|  | val intent = Intent(context, MainPage::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |

|  |
| --- |
|  |
|  | RegisterActivity.kt  Import android.content.Context import android.content.Intent |
|  | import android.os.Bundle |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.runtime.\* |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.text.font.FontFamily |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import androidx.core.content.ContextCompat |
|  | import com.example.snackordering.ui.theme.SnackOrderingTheme |
|  |  |
|  | class MainActivity : ComponentActivity() { |
|  | private lateinit var databaseHelper: UserDatabaseHelper |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | databaseHelper = UserDatabaseHelper(this) |
|  | setContent { |
|  | SnackOrderingTheme { |
|  |  |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  |  |
|  | RegistrationScreen(this,databaseHelper) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @Composable |
|  | fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) { |
|  |  |
|  | Image( |
|  | painterResource(id = R.drawable.order), contentDescription = "", |
|  | alpha =0.3F, |
|  | contentScale = ContentScale.FillHeight, |
|  |  |
|  | ) |
|  |  |
|  | var username by remember { mutableStateOf("") } |
|  | var password by remember { mutableStateOf("") } |
|  | var email by remember { mutableStateOf("") } |
|  | var error by remember { mutableStateOf("") } |
|  |  |
|  | Column( |
|  | modifier = Modifier.fillMaxSize(), |
|  | horizontalAlignment = Alignment.CenterHorizontally, |
|  | verticalArrangement = Arrangement.Center |
|  | ) { |
|  |  |
|  | Text( |
|  | fontSize = 36.sp, |
|  | fontWeight = FontWeight.ExtraBold, |
|  | fontFamily = FontFamily.Cursive, |
|  | color = Color.White, |
|  | text = "Register" |
|  | ) |
|  |  |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  | TextField( |
|  | value = username, |
|  | onValueChange = { username = it }, |
|  | label = { Text("Username") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp) |
|  |  |
|  | ) |
|  |  |
|  | TextField( |
|  | value = email, |
|  | onValueChange = { email = it }, |
|  | label = { Text("Email") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  | TextField( |
|  | value = password, |
|  | onValueChange = { password = it }, |
|  | label = { Text("Password") }, |
|  | modifier = Modifier |
|  | .padding(10.dp) |
|  | .width(280.dp) |
|  | ) |
|  |  |
|  |  |
|  | if (error.isNotEmpty()) { |
|  | Text( |
|  | text = error, |
|  | color = MaterialTheme.colors.error, |
|  | modifier = Modifier.padding(vertical = 16.dp) |
|  | ) |
|  | } |
|  |  |
|  | Button( |
|  | onClick = { |
|  | if (username.isNotEmpty() && password.isNotEmpty() && email.isNotEmpty()) { |
|  | val user = User( |
|  | id = null, |
|  | firstName = username, |
|  | lastName = null, |
|  | email = email, |
|  | password = password |
|  | ) |
|  | databaseHelper.insertUser(user) |
|  | error = "User registered successfully" |
|  |  |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | LoginActivity::class.java |
|  | ) |
|  | ) |
|  |  |
|  | } else { |
|  | error = "Please fill all fields" |
|  | } |
|  | }, |
|  | modifier = Modifier.padding(top = 16.dp) |
|  | ) { |
|  | Text(text = "Register") |
|  | } |
|  | Spacer(modifier = Modifier.width(10.dp)) |
|  | Spacer(modifier = Modifier.height(10.dp)) |
|  |  |
|  | Row() { |
|  | Text( |
|  | modifier = Modifier.padding(top = 14.dp), text = "Have an account?" |
|  | ) |
|  | TextButton(onClick = { |
|  | context.startActivity( |
|  | Intent( |
|  | context, |
|  | LoginActivity::class.java |
|  | ) |
|  | ) |
|  | }) |
|  |  |
|  | { |
|  | Spacer(modifier = Modifier.width(10.dp)) |
|  | Text(text = "Log in") |
|  | } |
|  | } |
|  | } |
|  | } |
|  | private fun startLoginActivity(context: Context) { |
|  | val intent = Intent(context, LoginActivity::class.java) |
|  | ContextCompat.startActivity(context, intent, null) |
|  | } |

|  |
| --- |
|  |
|  | MainPage.kt Import android.annotation.SuppressLint  import android.content.Context |
|  | import android.os.Bundle |
|  | import android.widget.Toast |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.annotation.DrawableRes |
|  | import androidx.annotation.StringRes |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.background |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.foundation.shape.CircleShape |
|  | import androidx.compose.foundation.shape.RoundedCornerShape |
|  | import androidx.compose.material.\* |
|  | import androidx.compose.material.icons.Icons |
|  | import androidx.compose.material.icons.filled.\* |
|  | import androidx.compose.runtime.Composable |
|  | import androidx.compose.ui.Alignment |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.draw.clip |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.foundation.lazy.LazyColumn |
|  | import androidx.compose.foundation.lazy.items |
|  | import androidx.compose.material.Text |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.graphics.RectangleShape |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.platform.LocalContext |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.res.stringResource |
|  | import androidx.compose.ui.text.font.FontWeight |
|  | import androidx.compose.ui.unit.sp |
|  | import androidx.core.content.ContextCompat.startActivity |
|  | import com.example.snackordering.ui.theme.SnackOrderingTheme |
|  |  |
|  | import android.content.Intent as Intent1 |
|  |  |
|  |  |
|  | class MainPage : ComponentActivity() { |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | setContent { |
|  | SnackOrderingTheme { |
|  |  |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  | FinalView(this) |
|  | val context = LocalContext.current |
|  | //PopularFoodColumn(context) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @Composable |
|  | fun TopPart() { |
|  |  |
|  | Row( |
|  | modifier = Modifier |
|  | .fillMaxWidth() |
|  | .background(Color(0xffeceef0)), Arrangement.SpaceBetween |
|  | ) { |
|  | Icon( |
|  | imageVector = Icons.Default.Add, contentDescription = "Menu Icon", |
|  | Modifier |
|  |  |
|  | .clip(CircleShape) |
|  | .size(40.dp), |
|  | tint = Color.Black, |
|  | ) |
|  | Column(horizontalAlignment = Alignment.CenterHorizontally) { |
|  | Text(text = "Location", style = MaterialTheme.typography.subtitle1, color = Color.Black) |
|  | Row { |
|  | Icon( |
|  | imageVector = Icons.Default.LocationOn, |
|  | contentDescription = "Location", |
|  | tint = Color.Red, |
|  | ) |
|  | Text(text = "Accra" , color = Color.Black) |
|  | } |
|  |  |
|  | } |
|  | Icon( |
|  | imageVector = Icons.Default.Notifications, contentDescription = "Notification Icon", |
|  |  |
|  | Modifier |
|  | .size(45.dp), |
|  | tint = Color.Black, |
|  | ) |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun CardPart() { |
|  | Card(modifier = Modifier.size(width = 310.dp, height = 150.dp), RoundedCornerShape(20.dp)) { |
|  | Row(modifier = Modifier.padding(10.dp), Arrangement.SpaceBetween) { |
|  | Column(verticalArrangement = Arrangement.spacedBy(12.dp)) { |
|  | Text(text = "Get Special Discounts") |
|  | Text(text = "up to 85%", style = MaterialTheme.typography.h5) |
|  | Button(onClick = {}, colors = ButtonDefaults.buttonColors(Color.White)) { |
|  | Text(text = "Claim voucher", color = MaterialTheme.colors.surface) |
|  | } |
|  | } |
|  | Image( |
|  | painter = painterResource(id = R.drawable.food\_tip\_im), |
|  | contentDescription = "Food Image", Modifier.size(width = 100.dp, height = 200.dp) |
|  | ) |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | @Composable |
|  | fun PopularFood( |
|  | @DrawableRes drawable: Int, |
|  | @StringRes text1: Int, |
|  | context: Context |
|  | ) { |
|  | Card( |
|  | modifier = Modifier |
|  | .padding(top=20.dp, bottom = 20.dp, start = 65.dp) |
|  | .width(250.dp) |
|  |  |
|  | ) { |
|  | Column( |
|  | verticalArrangement = Arrangement.Top, |
|  | horizontalAlignment = Alignment.CenterHorizontally |
|  | ) { |
|  | Spacer(modifier = Modifier.padding(vertical = 5.dp)) |
|  | Row( |
|  | modifier = Modifier |
|  | .fillMaxWidth(0.7f), Arrangement.End |
|  | ) { |
|  | Icon( |
|  | imageVector = Icons.Default.Star, |
|  | contentDescription = "Star Icon", |
|  | tint = Color.Yellow |
|  | ) |
|  | Text(text = "4.3", fontWeight = FontWeight.Black) |
|  | } |
|  | Image( |
|  | painter = painterResource(id = drawable), |
|  | contentDescription = "Food Image", |
|  | contentScale = ContentScale.Crop, |
|  | modifier = Modifier |
|  | .size(100.dp) |
|  | .clip(CircleShape) |
|  | ) |
|  | Text(text = stringResource(id = text1), fontWeight = FontWeight.Bold) |
|  | Row(modifier = Modifier.fillMaxWidth(0.7f), Arrangement.SpaceBetween) { |
|  | /\*TODO Implement Prices for each card\*/ |
|  | Text( |
|  | text = "$50", |
|  | style = MaterialTheme.typography.h6, |
|  | fontWeight = FontWeight.Bold, |
|  | fontSize = 18.sp |
|  | ) |
|  |  |
|  | IconButton(onClick = { |
|  |  |
|  |  |
|  |  |
|  | val intent = Intent1(context, TargetActivity::class.java) |
|  | context.startActivity(intent) |
|  |  |
|  | }) { |
|  | Icon( |
|  | imageVector = Icons.Default.ShoppingCart, |
|  | contentDescription = "shopping cart", |
|  | ) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | private val FoodList = listOf( |
|  | R.drawable.sandwish to R.string.sandwich, |
|  | R.drawable.sandwish to R.string.burgers, |
|  | R.drawable.pack to R.string.pack, |
|  | R.drawable.pasta to R.string.pasta, |
|  | R.drawable.tequila to R.string.tequila, |
|  | R.drawable.wine to R.string.wine, |
|  | R.drawable.salad to R.string.salad, |
|  | R.drawable.pop to R.string.popcorn |
|  | ).map { DrawableStringPair(it.first, it.second) } |
|  |  |
|  | private data class DrawableStringPair( |
|  | @DrawableRes val drawable: Int, |
|  | @StringRes val text1: Int |
|  | ) |
|  |  |
|  |  |
|  | @Composable |
|  | fun App(context: Context) { |
|  |  |
|  | Column( |
|  | modifier = Modifier |
|  | .fillMaxSize() |
|  | .background(Color(0xffeceef0)) |
|  | .padding(10.dp), |
|  | verticalArrangement = Arrangement.Top, |
|  | horizontalAlignment = Alignment.CenterHorizontally |
|  | ) { |
|  | Surface(modifier = Modifier, elevation = 5.dp) { |
|  | TopPart() |
|  | } |
|  | Spacer(modifier = Modifier.padding(10.dp)) |
|  | CardPart() |
|  |  |
|  | Spacer(modifier = Modifier.padding(10.dp)) |
|  | Row(modifier = Modifier.fillMaxWidth(), Arrangement.SpaceBetween) { |
|  | Text(text = "Popular Food", style = MaterialTheme.typography.h5, color = Color.Black) |
|  | Text(text = "view all", style = MaterialTheme.typography.subtitle1, color = Color.Black) |
|  | } |
|  | Spacer(modifier = Modifier.padding(10.dp)) |
|  | PopularFoodColumn(context) // <- call the function with parentheses |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | @Composable |
|  | fun PopularFoodColumn(context: Context) { |
|  |  |
|  | LazyColumn( |
|  | modifier = Modifier.fillMaxSize(), |
|  |  |
|  | content = { |
|  | items(FoodList) { item -> |
|  | PopularFood(context = context,drawable = item.drawable, text1 = item.text1) |
|  | abstract class Context |
|  | } |
|  | }, |
|  | verticalArrangement = Arrangement.spacedBy(16.dp)) |
|  | } |
|  |  |
|  |  |
|  | @SuppressLint("UnusedMaterialScaffoldPaddingParameter") |
|  | @Composable |
|  | fun FinalView(mainPage: MainPage) { |
|  | SnackOrderingTheme { |
|  | Scaffold() { |
|  | val context = LocalContext.current |
|  | App(context) |
|  | } |
|  | } |
|  | } |

AdminActivity.kt

|  |
| --- |
|  |
|  | Import android.icu.text.SimpleDateFormat  import android.os.Bundle |
|  | import android.util.Log |
|  | import androidx.activity.ComponentActivity |
|  | import androidx.activity.compose.setContent |
|  | import androidx.compose.foundation.Image |
|  | import androidx.compose.foundation.layout.\* |
|  | import androidx.compose.foundation.lazy.LazyColumn |
|  | import androidx.compose.foundation.lazy.LazyRow |
|  | import androidx.compose.foundation.lazy.items |
|  | import androidx.compose.material.MaterialTheme |
|  | import androidx.compose.material.Surface |
|  | import androidx.compose.material.Text |
|  | import androidx.compose.runtime.Composable |
|  | import androidx.compose.ui.Modifier |
|  | import androidx.compose.ui.graphics.Color |
|  | import androidx.compose.ui.layout.ContentScale |
|  | import androidx.compose.ui.res.painterResource |
|  | import androidx.compose.ui.unit.dp |
|  | import androidx.compose.ui.unit.sp |
|  | import com.example.snackordering.ui.theme.SnackOrderingTheme |
|  | import java.util.\* |
|  |  |
|  | class AdminActivity : ComponentActivity() { |
|  | private lateinit var orderDatabaseHelper: OrderDatabaseHelper |
|  | override fun onCreate(savedInstanceState: Bundle?) { |
|  | super.onCreate(savedInstanceState) |
|  | orderDatabaseHelper = OrderDatabaseHelper(this) |
|  | setContent { |
|  | SnackOrderingTheme { |
|  |  |
|  | Surface( |
|  | modifier = Modifier.fillMaxSize(), |
|  | color = MaterialTheme.colors.background |
|  | ) { |
|  | val data=orderDatabaseHelper.getAllOrders(); |
|  | Log.d("swathi" ,data.toString()) |
|  | val order = orderDatabaseHelper.getAllOrders() |
|  | ListListScopeSample(order) |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | @Composable |
|  | fun ListListScopeSample(order: List<Order>) { |
|  | Image( |
|  | painterResource(id = R.drawable.order), contentDescription = "", |
|  | alpha =0.5F, |
|  | contentScale = ContentScale.FillHeight) |
|  | Text(text = "Order Tracking", modifier = Modifier.padding(top = 24.dp, start = 106.dp, bottom = 24.dp ), color = Color.White, fontSize = 30.sp) |
|  | Spacer(modifier = Modifier.height(30.dp)) |
|  | LazyRow( |
|  | modifier = Modifier |
|  | .fillMaxSize() |
|  | .padding(top = 80.dp), |
|  |  |
|  | horizontalArrangement = Arrangement.SpaceBetween |
|  | ){ |
|  | item { |
|  |  |
|  | LazyColumn { |
|  | items(order) { order -> |
|  | Column(modifier = Modifier.padding(top = 16.dp, start = 48.dp, bottom = 20.dp)) { |
|  | Text("Quantity: ${order.quantity}") |
|  | Text("Address: ${order.address}") |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |

|  |
| --- |
|  |
|  | AndoridManifext.xml  <?xml version=”1.0” encouding =”utf-8”?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" |
|  | xmlns:tools="http://schemas.android.com/tools"> |
|  |  |
|  | <application |
|  | android:allowBackup="true" |
|  | android:dataExtractionRules="@xml/data\_extraction\_rules" |
|  | android:fullBackupContent="@xml/backup\_rules" |
|  | android:icon="@drawable/fast\_food" |
|  | android:label="@string/app\_name" |
|  | android:supportsRtl="true" |
|  | android:theme="@style/Theme.SnackOrdering" |
|  | tools:targetApi="31"> |
|  | <activity |
|  | android:name=".AdminActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_admin" |
|  | android:theme="@style/Theme.SnackOrdering" /> |
|  | <activity |
|  | android:name=".LoginActivity" |
|  | android:exported="true" |
|  | android:label="SnackSquad" |
|  | android:theme="@style/Theme.SnackOrdering"> |
|  | <intent-filter> |
|  | <action android:name="android.intent.action.MAIN" /> |
|  |  |
|  | <category android:name="android.intent.category.LAUNCHER" /> |
|  | </intent-filter> |
|  | </activity> |
|  | <activity |
|  | android:name=".TargetActivity" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_target" |
|  | android:theme="@style/Theme.SnackOrdering" /> |
|  | <activity |
|  | android:name=".MainPage" |
|  | android:exported="false" |
|  | android:label="@string/title\_activity\_main\_page" |
|  | android:theme="@style/Theme.SnackOrdering" /> |
|  | <activity |
|  | android:name=".MainActivity" |
|  | android:exported="false" |
|  | android:label="MainActivity" |
|  | android:theme="@style/Theme.SnackOrdering" /> |
|  | </application> |
|  |  |
|  | </manifest> |