Task ##P/C – Spike: Musical Rental Instrument Application

Goals:

A key spike was to design and build a proof of concept Android app for a music studio's musical instrument rental system incorporating state-of-the-art Android app development techniques and designs. This was a living example of application development techniques for Android applications that encompass concepts of Parcelable transport of data structures, using Material Design widgets, multilocal support, and management of data objects in memory.

The following list outlines the goal broken down into more specific knowledge gaps involved in the goal:

- Advanced Android UI/UX Design with Material 3 Components
- Parcelable Data Transfer Between Activities
- Multi-language Implementation and Locale Management
- In-Memory Data Management and Singleton Patterns
- Comprehensive Input Validation and Error Handling
- Modern Android Architecture and Navigation Patterns
- Professional Grade User Experience Design

Tools and Resources Used

- Android Studio Narwhal 3 Feature Drop | 2025.1.3
- Kotlin Programming Language 1.9.0
- Android Material Design Components 1.11.0
- AndroidX AppCompat, Core, and Testing Libraries
- Android Developer Documentation: Parcelable Implementation
- Material Design 3 Guidelines and Component Library
- Android Testing Support Library (Espresso)
- Locale and Internationalization Best Practices

Model Design

All data is kept in memory for the app lifetime. A small repository singleton holds the list of items and booking state.

Data model (with Parcelable):

```kotlin
package com.example.assignment2.data.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

12 Usages
@Parcelize
data class InstrumentItem(
    val id: Int,
    val name: String,
    val imageRes: Int,
    val price: Int,
    val rating: Float,
    val category: String,
    val description: String,
    val specs: List<String>
) : Parcelable
```

Repository (singleton, in memory):

```kotlin
object InstrumentItemData {
    1 Usage
    private val instruments = listOf(
        InstrumentItem(
            id = 1,
            name = "Acoustic Guitar",
            imageRes = R.drawable.guitar,
            price = 150,
            rating = 4.5f,
            category = "Acoustic",
            description = "A beautiful classic guitar with nylon strings, perfect for beginners and intermediate players. Great sound qua
            specs = listOf("Nylon strings", "Spruce top", "Rosewood fingerboard")
        ),

        InstrumentItem(
            id = 2,
            name = "Yamaha Piano",
            imageRes = R.drawable.piano,
            price = 350,
            rating = 4.8f,
            category = "Keyboard",
            description = "Professional digital piano with weighted keys and authentic sound.",
            specs = listOf("88 keys", "Weighted action", "3 pedals")
        ),
        InstrumentItem(
            id = 3,
            name = "Fender Stratocaster",
            imageRes = R.drawable.electric_guitar,
            price = 200,
            rating = 4.3f,
            category = "Electric",
            description = "Classic electric guitar with versatile tone options.",
            specs = listOf("Maple neck", "3 single-coil pickups", "Tremolo system")
        )
    )
```

```
    1 Usage
    fun getInstruments(): List<InstrumentItem> = instruments
}
```

User Stories and Use Cases

- **User Story 1: The Hesitant First-Time Renter**

User Story: Ms. Elena Vance: "As a first-time renter with a limited budget, I want to easily compare the final rental costs of different electric guitars, including any insurance or delivery fees. Let's just say, I get anxious about hidden costs, so I need to see the full price breakdown and be sure I can afford it before I commit."

Use Case: Comparing Total Rental Cost.

Actor: First-time renter / Budget-conscious student / Price-sensitive user.

Precondition: The user has opened the app, completed the onboarding process, and is browsing the available instruments.

Postcondition: The user has a clear understanding of the total cost for one or more instruments and has either proceeded with a rental for the best option or exited the app without making a purchase.

- **User Story 2: The Busy Professional Musician**

User Story: Mr. Kenji Tanaka: "As a touring musician, my time is limited. I need to find and rent a high quality Fender Stratocaster that is available for my specific tour dates immediately. I don't want to browse; I want to filter, see what's available now, and secure it with one click."

Use Case: Quick Filtering and Reservation of a Specific Instrument.

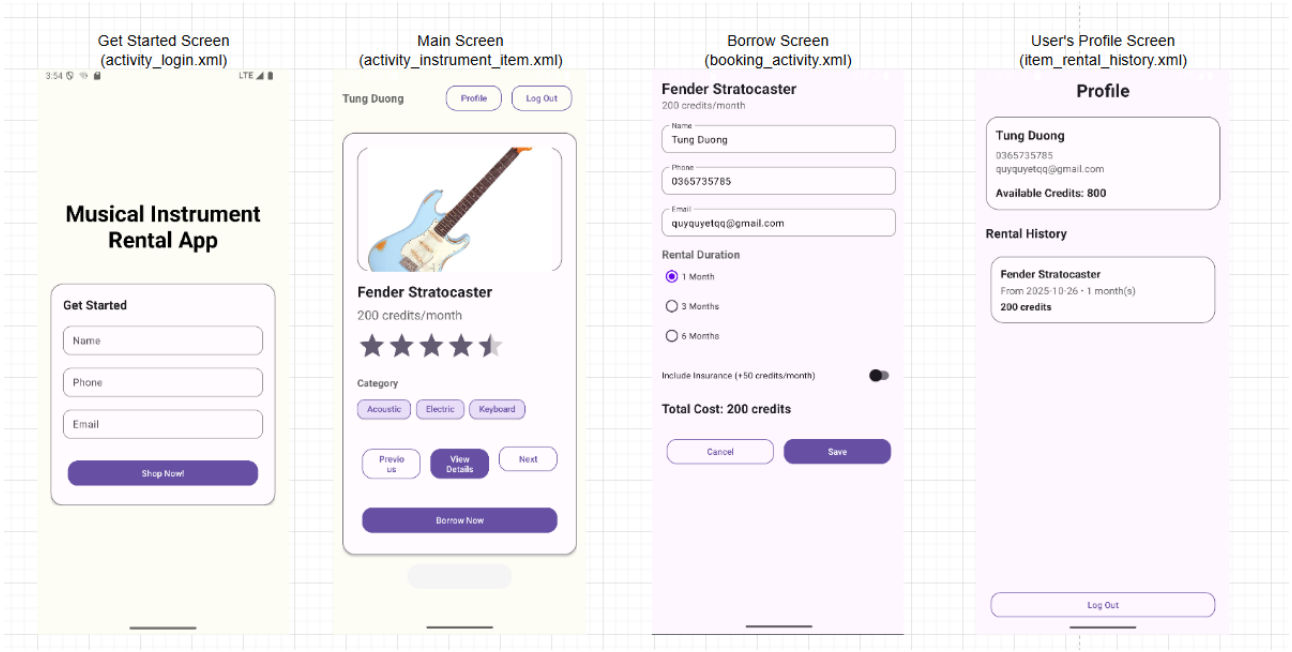
Actor: Professional musician / Frequent renter / Time-sensitive user.

Precondition: The user is logged into the app and is on the instrument catalog screen.

Postcondition: A specific instrument is successfully reserved for the user's selected dates, and a confirmation screen is displayed.

UI Sketches

Musical Rental Instrument App UI Sketches



- **Layout 1: Get Started Screen(activity_login.xml)**

LinearLayout(Contents inside box)
TextView(App's name)
TextInputLayout(Name, Phone, Email)
Button(Shop now!)

- **Layout 2: Main Screen(activity_instrument_item.xml)**

LinearLayout(Header, navigation buttons)
TextView(Name display, instruments details, pricing, categories)
ChipGroup(Categories filtering)
Button(Navigation, borrow instrument)

- **Layout 3: Borrow Screen(booking_activity.xml)**

LinearLayout(Layout)
TextView(Total cost, instrument name, rental cost/month)
TextInputLayout(Name, Phone, Email)
RadioGroup(Rental Durations)
SwitchMaterial(Enables whether user wants to include insurance)

- **Layout 4: User's Profile Screen(item_rental_history.xml)**

LinearLayout(Layout)
TextView(Instrument name, user's profile, rental history)

Knowledge Gaps and Solutions

Below are the knowledge Gaps and Solutions:

Colours and Styles

The app's visual design follows Material Design 3 guidelines, emphasizing simplicity, accessibility, and brand consistency. A cohesive colour palette and reusable style definitions were implemented to ensure professional, consistent appearance across all screens.

- Primary color: used on buttons and toolbar for key actions.
- Surface/background: white cards on a light lavender-gray background for readable contrast.
- Accent tones: chips and icons reuse the same purple family to keep a cohesive look.

Text and UI components share reusable styles:

- HeadingText for titles (bold, dark).
- BodyText for regular content (medium gray).
- PrimaryButton and SecondaryButton ensure consistent size, rounded corners, and spacing.

1. Colours

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="white">#FFFFFFFF</color>
    <color name="black">#FF000000</color>
    <color name="dark_gray">#FF666666</color>
    <color name="light_gray">#FFEEEEEE</color>
    <color name="chip_background">#FFEFEFEF</color>
    <color name="primary">#FF6750A4</color>
    <color name="primary_dark">#FF4A356A</color>
    <color name="primary_light">#FFE8DEF8</color>
    <color name="secondary">#FF7D5260</color>
    <color name="surface">#FFFFFBFE</color>
    <color name="background">#FFFDFDF6</color>
    <color name="on_primary">#FFFFFFFF</color>
    <color name="on_surface">#FF1C1B1F</color>
    <color name="outline">#FF79747E</color>
</resources>
```

2. Styles

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Modern Text Styles -->
    <style name="HeadingText">
        <item name="android:textColor">@color/on_surface</item>
        <item name="android:textSize">24sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="BodyText">
        <item name="android:textColor">@color/dark_gray</item>
        <item name="android:textSize">16sp</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="CaptionText">
        <item name="android:textColor">@color/outline</item>
        <item name="android:textSize">14sp</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="EditText">
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/black</item>
        <item name="android:padding">12dp</item>
    </style>

    <!-- Modern Buttons - Use shape appearance instead of cornerRadius -->
    <style name="PrimaryButton" parent="Widget.Material3.Button">
        <item name="android:layout_height">56dp</item>
        <item name="android:layout_margin">8dp</item>
        <item name="android:textColor">@color/on_primary</item>
        <item name="android:backgroundTint">@color/primary</item>
        <item name="shapeAppearance">@style/ShapeAppearanceMedium</item>
        <item name="android:elevation">2dp</item>
    </style>

    <style name="SecondaryButton" parent="Widget.Material3.Button.OutlinedButton">
        <item name="android:layout_height">56dp</item>
        <item name="android:layout_margin">8dp</item>
```

```xml
<resources>
    <style name="SecondaryButton" parent="Widget.Material3.Button.OutlinedButton">
        <item name="android:textColor">@color/primary</item>
        <item name="strokeColor">@color/primary</item>
        <item name="strokeWidth">1dp</item>
        <item name="shapeAppearance">@style/ShapeAppearanceMedium</item>
    </style>

    <!-- Enhanced Chip Style - Use chipCornerRadius -->
    <style name="Chip" parent="Widget.Material3.Chip.Assist.Elevated">
        <item name="chipBackgroundColor">@color/primary_light</item>
        <item name="chipStrokeColor">@color/primary</item>
        <item name="chipStrokeWidth">1dp</item>
        <item name="chipCornerRadius">12dp</item>
        <item name="android:textColor">@color/primary</item>
    </style>

    <style name="TextInputLayout" parent="Widget.Material3.TextInputLayout.OutlinedBox">
        <item name="boxBackgroundColor">@color/surface</item>
        <item name="boxBackgroundMode">outline</item>
        <item name="boxStrokeColor">@color/outline</item>
        <item name="shapeAppearance">@style/ShapeAppearanceSmall</item>
        <item name="hintTextColor">@color/outline</item>
    </style>

    <style name="ShapeAppearanceSmall" parent="ShapeAppearance.Material3.SmallComponent">
        <item name="cornerFamily">rounded</item>
        <item name="cornerSize">12dp</item>
    </style>

    <!-- Card Style -->
    <style name="Card">
        <item name="android:background">@drawable/card_background</item>
        <item name="android:elevation">2dp</item>
    </style>

    <!-- Shape Appearances -->
    <style name="ShapeAppearanceMedium" parent="ShapeAppearance.Material3.MediumComponent">
        <item name="cornerFamily">rounded</item>
        <item name="cornerSize">16dp</item>
    </style>
```

Together, these choices create a modern, professional look that's easy to read, visually balanced, and consistent across all screens.

Parcelable and Intent

What is Parcelable and Intent?

An Intent is a messaging object used to request an action from another app component, like navigating between activities. A Parcelable is an Android-specific interface that allows custom objects to be serialized (flattened) into a Parcel so they can be passed efficiently between components via an Intent.

Parcelable and Intents are crucial for this app, therefore, there are several Intents in the implementation:

1. LoginActivity to InstrumentItemActivity

```
btnLogin.setOnClickListener {
    if (validateInput()) {
        val user = User(
            name = etName.text.toString(),
            phone = etPhone.text.toString(),
            email = etEmail.text.toString()
        )
        UserManager.setUser(user)
        startActivity(Intent(this, InstrumentItemActivity::class.java))
        finish()
    }
}
```

2. InstrumentItemActivity to ItemDetailsActivity

```
btnViewDetails.setOnClickListener {
    val intent = Intent(this, ItemDetailsActivity::class.java).apply {
        putExtra("INSTRUMENT", instruments[currentIndex])
    }
    startActivity(intent)
}
```

3. InstrumentItemActivity to BookingActivity

```
btnBorrow.setOnClickListener {
        val intent = Intent(this, BookingActivity::class.java).apply {
            putExtra("INSTRUMENT", instruments[currentIndex])
        }
        startActivity(intent)
}
```

4. ProfileActivity and Change User Navigation

```
1 Usage
private fun setupListeners() {
    btnChangeUser.setOnClickListener {
        startActivity(Intent(this, LoginActivity::class.java))
        finish()
    }
}
```

5. ItemDetailsActivity to BookingActivity

```
        btnBorrow.setOnClickListener {
                val intent = Intent(this, BookingActivity::class.java).apply {
                    putExtra("INSTRUMENT", instrument)
                }
                startActivity(intent)
        }
    }
```

Intent Components with Parcelable:

What's in the Intent:

```
        btnBorrow.setOnClickListener {
                val intent = Intent(this, BookingActivity::class.java).apply {
                    putExtra("INSTRUMENT", instrument)
                }
                startActivity(intent)
        }
    }
```

Receiving the Parcelable in BookingActivity:

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

    instrument = intent.getParcelableExtra("INSTRUMENT")!!

    initializeViews()
    setupUI()
    setupListeners()
    calculateTotalCost()
}
```

Advantages of Using Parcelable:
1. Performance
- Parcelable is faster than Serializable for Android
- Direct memory access vs reflection in Serializable

2. Android-Optimized
- Designed specifically for Android inter-process communication
- More efficient for Intent data transfer

3. Type Safety
- Compile-time checking vs runtime errors with Serializable

4. Manual Control
- You control exactly what gets serialized/deserialized

5. Kotlin Support
- With @Parcelize, minimal code required

Validation and Error Handling

Validation and Error Handling is very important, not only in this assignment but it's important for improving security.

1. Real-Time Field Validation

```
1 Usage
private fun validateInput(): Boolean {
    val name = etName.text.toString()
    val phone = etPhone.text.toString()
    val email = etEmail.text.toString()

    if (name.isBlank()) {
        etName.error = "Name is required"
        return false
    }

    if (phone.isBlank()) {
        etPhone.error = "Phone is required"
        return false
    }

    if (email.isBlank() || !Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        etEmail.error = "Valid email is required"
        return false
    }

    return true
}
```

2. Credit System Validation

```
1 Usage
private fun validateCredits(): Boolean {
    val user = UserManager.getUser()
    val totalCost = calculateTotalCostValue()

    if (user == null) {
        showError("User not found")
        return false
    }

    if (user.credits < totalCost) {
        showError("Insufficient credits. You have ${user.credits} credits but need $totalCost")
        return false
    }

    return true
}

2 Usages
private fun calculateTotalCostValue(): Int {
    var total = instrument.price * duration
    if (insurance) {
        total += 50 * duration
    }
    return total
}
```

```
3 Usages
private fun calculateTotalCost() {
    var total = instrument.price * duration
    if (insurance) {
        total += 50 * duration
    }
    tvTotalCost.text = "Total Cost: $total credits"
}
```

3. Booking Flow Validation Chain

```
1 Usage
private fun validateInput(): Boolean {
    val name = etName.text.toString()
    val phone = etPhone.text.toString()
    val email = etEmail.text.toString()

    if (name.isBlank()) {
        etName.error = "Name is required"
        return false
    }

    if (phone.isBlank()) {
        etPhone.error = "Phone is required"
        return false
    }

    if (email.isBlank() || !Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        etEmail.error = "Valid email is required"
        return false
    }

    return true
}
```

This guarantees that the data remains consistent while making the UI responsive. The application only keeps all data in memory (no files or databases applied). This method is appropriate for a proof-of-concept where persistence isn't required. When the application closes, the data simply resets.

Feedback and User Interaction

A Snackbar is displayed upon successful booking:

```
1 Usage
private fun showRecentBookingNotification() {
    val recentBooking = BookingHistoryManager.getBookings().lastOrNull()
    if (recentBooking != null) {
        val rootView = findViewById<View>(android.R.id.content)
        Snackbar.make(rootView,
            "Booked successfully! Recently booked: ${recentBooking.instrument.name}",
            Snackbar.LENGTH_LONG
        ).show()
    }
}
```

Toasts are used for minor messages:

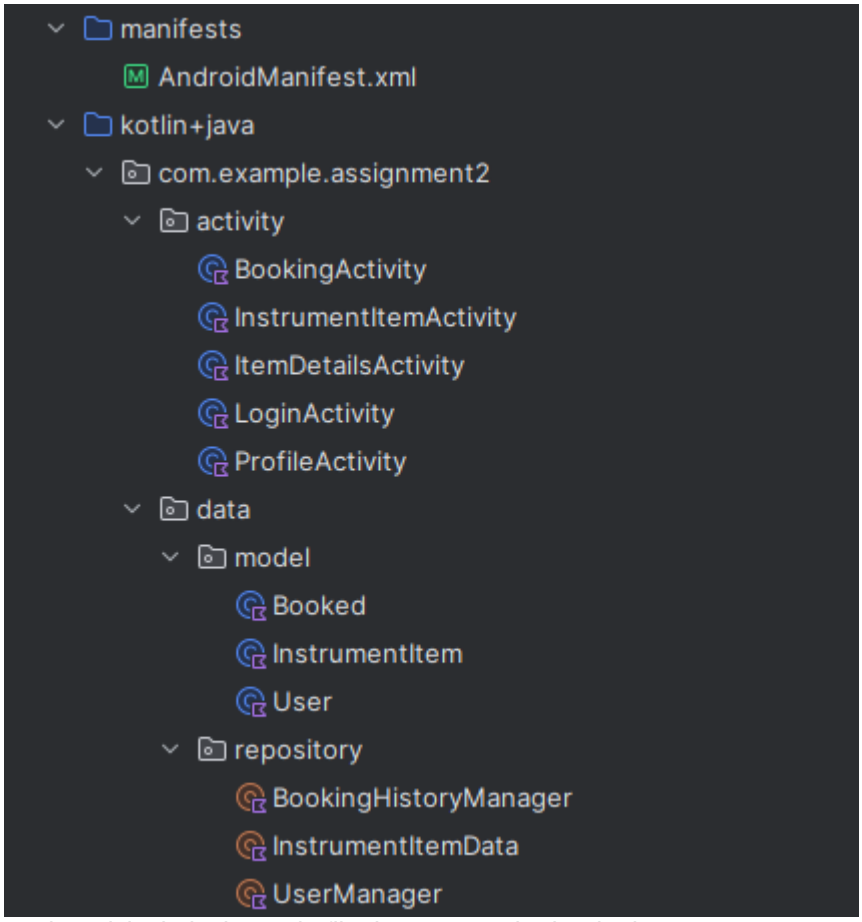
```
1 Usage
private fun showCancellationToast() {
    Toast.makeText(
        this,
        "Booking cancelled",
        Toast.LENGTH_SHORT
    ).show()
}

2 Usages
private fun showError(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_LONG).show()
}
```

```
3 Usages
private fun jumpToFirstInstrumentOfCategory(category: String) {
    val index = instruments.indexOfFirst { it.category == category }
    if (index != -1) {
        currentIndex = index
        displayCurrentInstrument()
        Toast.makeText(this, "Jumped to $category", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "No $category instruments", Toast.LENGTH_SHORT).show()
    }
}
```

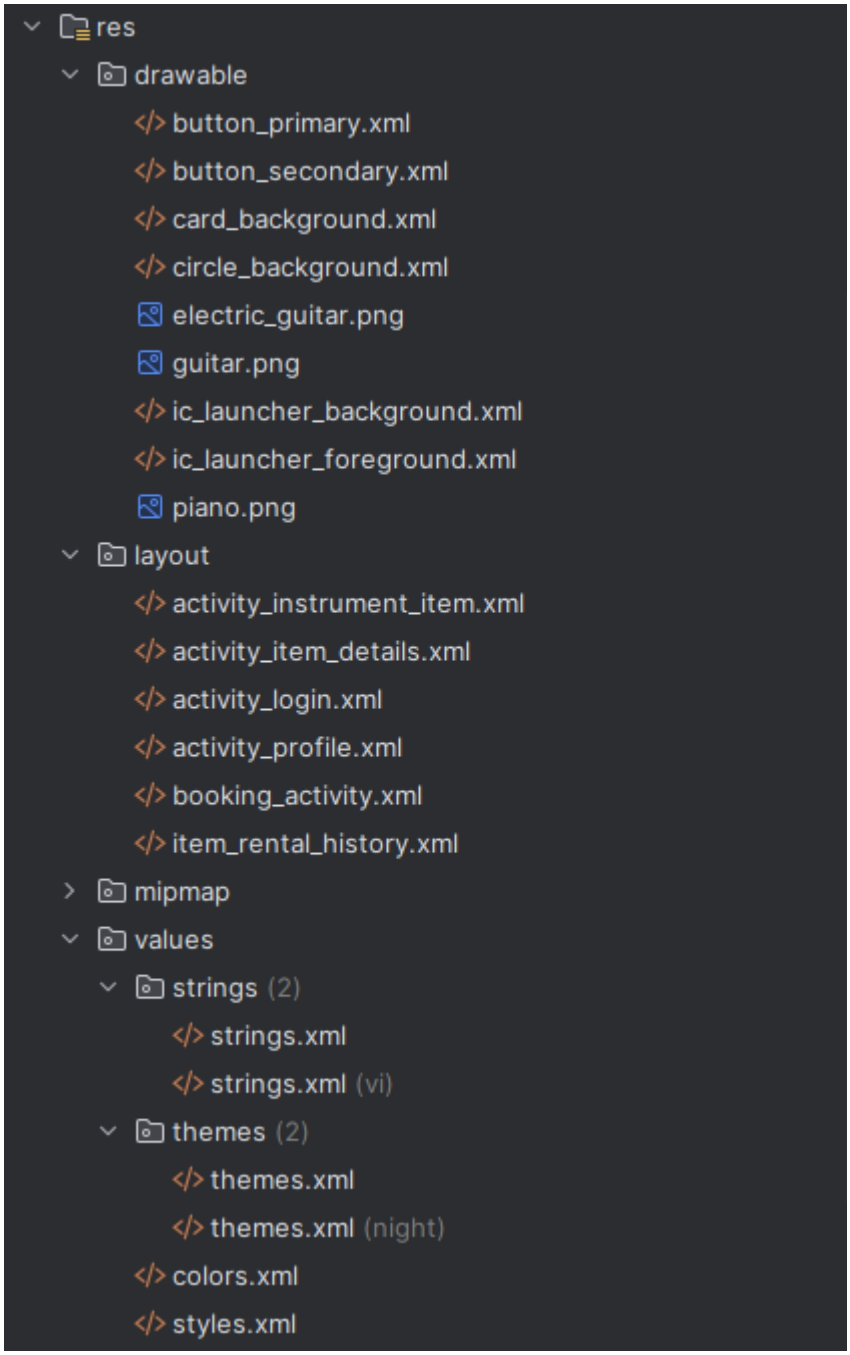
Snackbars were chosen for success feedback because they blend with Material Design and maintain visibility without interrupting user flow.

Code Structure and Design System

Here is the structure of the app's code:



LoginActivity.kt is the main file that runs at the beginning.



This is the xml files (files used to design the application).

Development Process

The application was developed using a systematic, phased approach:
Phase 1: Requirements Analysis & Planning (Days 1-2)

- Thorough analysis of assignment requirements
- Creation of detailed specification document
- UI/UX planning and wireframing
- Architecture design and technology selection

Phase 2: Core Implementation (Days 3-7)

- Data model creation (InstrumentItem, User, Booked)
- Repository pattern implementation
- Basic activity navigation flow
- Parcelable data transfer implementation

Phase 3: UI/UX Enhancement (Days 8-10)
- Material Design 3 component integration
- Professional styling and theming
- Validation and error handling systems
- User experience refinements

Phase 4: Testing & Polish (Days 11-12)
- Comprehensive functionality testing
- Bug fixes and performance optimization
- Final documentation and report preparation

Commit History and Version Control

While the project was developed using local version control with frequent incremental saves, no commits were pushed to GitHub until the final completion stage. This approach allowed for:
- Focused development without premature repository management
- Rapid prototyping and experimentation
- Clean commit history with meaningful, atomic changes
- Comprehensive final push with well-structured commit messages

Time Management
- Daily development sessions: 3-4 hours focused work
- Regular testing: After each feature implementation
- Incremental progress: Small, testable features built sequentially
- Documentation: Continuous documentation alongside development

Quality Assurance
- Systematic testing of each feature before proceeding
- Code review through self-testing and validation
- User experience testing on multiple device configurations
- Edge case consideration in validation and error handling

Final Commit Structure (When pushed to GitHub):
- Final bug fixes and documentation

Open Issues and Recommendations

This rental application for a musical instrument successfully illustrates a functional proof-of-concept that fulfills all specified requirements. The application was developed using a clear and easy-to-use user experience to browse and rent instruments. The application is organized around a simple activity-based architecture.

References

Ye, X., Ruan, Y., Xia, S., & Gu, L. (2025). Adoption of digital intangible cultural heritage: a configurational study integrating UTAUT2 and immersion theory. *Humanities and Social Sciences Communications*, *12*(1). https://doi.org/10.1057/s41599-024-04222-8

Google. (2023). *Material Design*. Material Design. https://m3.material.io/

Dev, A. (n.d.). *Parcelable*. Android Developers. https://developer.android.com/reference/android/os/Parcelable

```kotlin
package com.example.assignment2.data.repository

import com.example.assignment2.data.model.User

object UserManager {
    private var currentUser: User? = null
    fun setUser(user: User) {
        currentUser = user.copy()
    }

    fun getUser(): User? {
        return currentUser
    }

    fun updateCredits(credits: Int) {
        currentUser = currentUser?.copy(credits = credits)
    }

    fun clearUser() {
        currentUser = null
    }
}
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/
  res/android">
3     <solid android:color="@color/surface" />
4     <stroke android:width="1dp" android:color="@color
  /outline" />
5     <corners android:radius="20dp" />
6     <padding android:left="16dp" android:top="16dp"
  android:right="16dp" android:bottom="16dp" />
7 </shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com
/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    android:padding="24dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_gravity="center_vertical"
        android:gravity="center_horizontal">


        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="48dp"
            android:gravity="center"
            android:text="@string/app_name"
            android:textColor="@color/black"
            android:textSize="36sp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            style="@style/Card"
            android:padding="20dp">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/get_started"
                android:textSize="20sp"
                android:textStyle="bold"
                android:textColor="@color/on_surface"
                android:layout_marginBottom="16dp" />
```

```
41
42              <com.google.android.material.textfield.
    TextInputLayout
43                  android:layout_width="match_parent"
44                  android:layout_height="wrap_content"
45                  android:layout_marginBottom="16dp"
46                  style="@style/TextInputLayout">
47
48                  <com.google.android.material.
    textfield.TextInputEditText
49                      android:id="@+id/etName"
50                      android:layout_width="
    match_parent"
51                      android:layout_height="
    wrap_content"
52                      android:hint="@string/name"
53                      style="@style/EditText" />
54
55              </com.google.android.material.textfield.
    TextInputLayout>
56
57              <com.google.android.material.textfield.
    TextInputLayout
58                  android:layout_width="match_parent"
59                  android:layout_height="wrap_content"
60                  android:layout_marginBottom="16dp"
61                  style="@style/TextInputLayout">
62
63                  <com.google.android.material.
    textfield.TextInputEditText
64                      android:id="@+id/etPhone"
65                      android:layout_width="
    match_parent"
66                      android:layout_height="
    wrap_content"
67                      android:hint="@string/phone"
68                      android:inputType="phone"
69                      style="@style/EditText" />
70
71              </com.google.android.material.textfield.
    TextInputLayout>
```

```
72
73              <com.google.android.material.textfield.
    TextInputLayout
74                  android:layout_width="match_parent"
75                  android:layout_height="wrap_content"
76                  android:layout_marginBottom="24dp"
77                  style="@style/TextInputLayout">
78
79                  <com.google.android.material.
    textfield.TextInputEditText
80                      android:id="@+id/etEmail"
81                      android:layout_width="
    match_parent"
82                      android:layout_height="
    wrap_content"
83                      android:hint="@string/email"
84                      android:inputType="
    textEmailAddress"
85                      style="@style/EditText" />
86
87              </com.google.android.material.textfield.
    TextInputLayout>
88
89              <Button
90                  android:id="@+id/btnLogin"
91                  android:layout_width="match_parent"
92                  android:layout_height="wrap_content"
93                  android:text="@string/shop_now"
94                  style="@style/PrimaryButton" />
95
96          </LinearLayout>
97
98      </LinearLayout>
99 </ScrollView>
```

```kotlin
package com.example.assignment2.data.repository

import com.example.assignment2.data.model.InstrumentItem
import com.example.assignment2.R

object InstrumentItemData {
    private val instruments = listOf(
        InstrumentItem(
            id = 1,
            name = "Acoustic Guitar",
            imageRes = R.drawable.guitar,
            price = 150,
            rating = 4.5f,
            category = "Acoustic",
            description = "A beautiful classic guitar with nylon strings, perfect for beginners and intermediate players. Great sound quality and comfortable to play.",
            specs = listOf("Nylon strings", "Spruce top", "Rosewood fingerboard")
        ),

        InstrumentItem(
            id = 2,
            name = "Yamaha Piano",
            imageRes = R.drawable.piano,
            price = 350,
            rating = 4.8f,
            category = "Keyboard",
            description = "Professional digital piano with weighted keys and authentic sound.",
            specs = listOf("88 keys", "Weighted action", "3 pedals")
        ),
        InstrumentItem(
            id = 3,
            name = "Fender Stratocaster",
            imageRes = R.drawable.electric_guitar,
            price = 200,
            rating = 4.3f,
```

```
35              category = "Electric",
36              description = "Classic electric guitar
   with versatile tone options.",
37              specs = listOf("Maple neck", "3 single-
   coil pickups", "Tremolo system")
38          )
39      )
40
41      fun getInstruments(): List<InstrumentItem> =
   instruments
42  }
```

```kotlin
package com.example.assignment2.data.repository

import com.example.assignment2.data.model.Booked

object BookingHistoryManager {
    private val bookings = mutableListOf<Booked>()

    fun addBooking(booking: Booked) {
        bookings.add(booking)
    }

    fun getBookings(): List<Booked> = bookings.toList()
}
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/background"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center_vertical"
        android:layout_marginBottom="24dp">

        <TextView
            android:id="@+id/tvUserName"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:textSize="18sp"
            android:textStyle="bold"
            style="@style/BodyText" />

        <Button
            android:id="@+id/btnProfile"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/profile"
            style="@style/SecondaryButton"
            android:layout_marginEnd="8dp" />

        <Button
            android:id="@+id/btnChangeUser"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/log_out"
            style="@style/SecondaryButton" />

    </LinearLayout>
```

```
41
42      <!-- Instrument Card -->
43      <LinearLayout
44          android:layout_width="match_parent"
45          android:layout_height="wrap_content"
46          android:orientation="vertical"
47          style="@style/Card"
48          android:padding="24dp">
49
50          <!-- Instrument Image -->
51          <ImageView
52              android:id="@+id/ivInstrument"
53              android:layout_width="match_parent"
54              android:layout_height="200dp"
55              android:layout_marginBottom="16dp"
56              android:scaleType="centerCrop"
57              android:background="@drawable/
    card_background"
58              android:contentDescription="@string/
    instrument_image" />
59
60          <!-- Instrument Details -->
61          <TextView
62              android:id="@+id/tvInstrumentName"
63              android:layout_width="match_parent"
64              android:layout_height="wrap_content"
65              android:textSize="24sp"
66              android:textStyle="bold"
67              android:layout_marginBottom="8dp"
68              style="@style/HeadingText" />
69
70          <TextView
71              android:id="@+id/tvPrice"
72              android:layout_width="match_parent"
73              android:layout_height="wrap_content"
74              android:textSize="20sp"
75              android:layout_marginBottom="12dp"
76              style="@style/BodyText" />
77
78          <RatingBar
79              android:id="@+id/ratingBar"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:numStars="5"
            android:stepSize="0.5"
            android:rating="0"
            android:isIndicator="true" />

        <!-- Categories -->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/category"
            android:textSize="16sp"
            android:textStyle="bold"
            android:layout_marginBottom="8dp"
            style="@style/BodyText" />

        <com.google.android.material.chip.ChipGroup
            android:id="@+id/chipGroupCategory"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp">

            <com.google.android.material.chip.Chip
                android:id="@+id/chipAcoustic"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Acoustic"
                style="@style/Chip" />

            <com.google.android.material.chip.Chip
                android:id="@+id/chipElectric"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Electric"
                style="@style/Chip" />

            <com.google.android.material.chip.Chip
                android:id="@+id/chipKeyboard"
                android:layout_width="wrap_content"
```

```
121                 android:layout_height="wrap_content"
122                 android:text="Keyboard"
123                 style="@style/Chip" />
124
125         </com.google.android.material.chip.ChipGroup
    >
126
127         <!-- Navigation Buttons -->
128         <LinearLayout
129             android:layout_width="match_parent"
130             android:layout_height="83dp"
131             android:layout_marginBottom="16dp"
132             android:orientation="horizontal">
133
134             <Button
135                 android:id="@+id/btnPrevious"
136                 style="@style/SecondaryButton"
137                 android:layout_width="0dp"
138                 android:layout_height="wrap_content"
139                 android:layout_weight="1"
140                 android:text="@string/previous" />
141
142             <Button
143                 android:id="@+id/btnViewDetails"
144                 style="@style/PrimaryButton"
145                 android:layout_width="0dp"
146                 android:layout_height="wrap_content"
147                 android:layout_marginStart="8dp"
148                 android:layout_marginEnd="8dp"
149                 android:layout_weight="1"
150                 android:text="@string/view_details"
    />
151
152             <Button
153                 android:id="@+id/btnNext"
154                 style="@style/SecondaryButton"
155                 android:layout_width="0dp"
156                 android:layout_height="wrap_content"
157                 android:layout_weight="1"
158                 android:padding="0dp"
159                 android:text="@string/next" />
```

```
160
161         </LinearLayout>
162
163         <Button
164             android:id="@+id/btnBorrow"
165             android:layout_width="match_parent"
166             android:layout_height="wrap_content"
167             android:text="@string/borrow_now"
168             style="@style/PrimaryButton" />
169
170     </LinearLayout>
171
172 </LinearLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@android:color/transparent" />
    <stroke android:width="1dp" android:color="@color/primary" />
    <corners android:radius="16dp" />
</shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvInstrumentName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:textStyle="bold"
            style="@style/HeadingText" />

        <TextView
            android:id="@+id/tvPrice"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            style="@style/BodyText" />

        <com.google.android.material.textfield.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            style="@style/TextInputLayout">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/etName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="@string/name"
                style="@style/EditText" />
```

```
39
40          </com.google.android.material.textfield.
    TextInputLayout>
41
42          <com.google.android.material.textfield.
    TextInputLayout
43              android:layout_width="match_parent"
44              android:layout_height="wrap_content"
45              android:layout_marginBottom="16dp"
46              style="@style/TextInputLayout">
47
48              <com.google.android.material.textfield.
    TextInputEditText
49                  android:id="@+id/etPhone"
50                  android:layout_width="match_parent"
51                  android:layout_height="wrap_content"
52                  android:hint="@string/phone"
53                  android:inputType="phone"
54                  style="@style/EditText" />
55
56          </com.google.android.material.textfield.
    TextInputLayout>
57
58          <com.google.android.material.textfield.
    TextInputLayout
59              android:layout_width="match_parent"
60              android:layout_height="wrap_content"
61              android:layout_marginBottom="16dp"
62              style="@style/TextInputLayout">
63
64              <com.google.android.material.textfield.
    TextInputEditText
65                  android:id="@+id/etEmail"
66                  android:layout_width="match_parent"
67                  android:layout_height="wrap_content"
68                  android:hint="@string/email"
69                  android:inputType="textEmailAddress"
70                  style="@style/EditText" />
71
72          </com.google.android.material.textfield.
    TextInputLayout>
```

```
 73
 74          <TextView
 75              android:layout_width="match_parent"
 76              android:layout_height="wrap_content"
 77              android:text="@string/rental_duration"
 78              android:textSize="18sp"
 79              android:textStyle="bold"
 80              style="@style/BodyText" />
 81
 82          <RadioGroup
 83              android:id="@+id/radioGroupDuration"
 84              android:layout_width="match_parent"
 85              android:layout_height="wrap_content"
 86              android:layout_marginBottom="16dp"
 87              android:orientation="vertical">
 88
 89              <RadioButton
 90                  android:id="@+id/radio1Month"
 91                  android:layout_width="match_parent"
 92                  android:layout_height="wrap_content"
 93                  android:text="@string/one_month" />
 94
 95              <RadioButton
 96                  android:id="@+id/radio3Months"
 97                  android:layout_width="match_parent"
 98                  android:layout_height="wrap_content"
 99                  android:text="@string/three_months"
    />
100
101              <RadioButton
102                  android:id="@+id/radio6Months"
103                  android:layout_width="match_parent"
104                  android:layout_height="wrap_content"
105                  android:text="@string/six_months" />
106
107          </RadioGroup>
108
109          <com.google.android.material.switchmaterial.
    SwitchMaterial
110              android:id="@+id/switchInsurance"
111              android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:text="@string/include_insurance"
            android:layout_marginBottom="16dp" />

        <TextView
            android:id="@+id/tvTotalCost"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            style="@style/HeadingText" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="24dp">

            <Button
                android:id="@+id/btnCancel"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/cancel"
                style="@style/SecondaryButton" />

            <Button
                android:id="@+id/btnSave"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:text="@string/save"
                style="@style/PrimaryButton" />

        </LinearLayout>

    </LinearLayout>
</ScrollView>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/ivInstrument"
            android:layout_width="match_parent"
            android:layout_height="371dp"
            android:layout_marginBottom="16dp"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/tvInstrumentName"
            style="@style/HeadingText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:textStyle="bold" />

        <RatingBar
            android:id="@+id/ratingBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginVertical="8dp"
            android:isIndicator="true"
            android:numStars="5"
            android:rating="0"
            android:stepSize="0.5" />

        <TextView
            android:id="@+id/tvPrice"
            style="@style/BodyText"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp" />

        <TextView
            style="@style/HeadingText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/description"
            android:textSize="18sp"
            android:textStyle="bold" />

        <TextView
            android:id="@+id/tvDescription"
            style="@style/BodyText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp" />

        <TextView
            style="@style/HeadingText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/specifications"
            android:textSize="18sp"
            android:textStyle="bold" />

        <LinearLayout
            android:id="@+id/llSpecs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="24dp"
            android:orientation="vertical" />

        <Button
            android:id="@+id/btnBorrow"
            style="@style/PrimaryButton"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/borrow_now" />

    </LinearLayout>
```

```
82 </ScrollView>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/primary" />
    <corners android:radius="16dp" />
</shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:background="@drawable/card_background"
    android:padding="16dp"
    android:layout_margin="8dp">

    <TextView
        android:id="@+id/tvInstrumentName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        style="@style/HeadingText" />

    <TextView
        android:id="@+id/tvRentalPeriod"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        style="@style/BodyText" />

    <TextView
        android:id="@+id/tvCost"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:textSize="16sp"
        android:textStyle="bold"
        style="@style/HeadingText" />

</LinearLayout>
```

```kotlin
package com.example.assignment2.data.model

data class Booked(
    val instrument: InstrumentItem,
    val duration: Int,
    val insurance: Boolean,
    val totalCost: Int
)
```

File - D:\Android Projects\Assignment2\app\src\main\res\drawable\circle_background.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="@color/primary_light" />
    <stroke android:width="2dp" android:color="@color/primary" />
</shape>
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/profile"
        android:textSize="28sp"
        android:textStyle="bold"
        android:gravity="center"
        android:layout_marginBottom="24dp"
        style="@style/HeadingText" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:background="@drawable/card_background"
        android:padding="16dp"
        android:layout_marginBottom="24dp">

        <TextView
            android:id="@+id/tvUserName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            style="@style/HeadingText" />

        <TextView
            android:id="@+id/tvUserPhone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            style="@style/BodyText" />
```

```
40
41          <TextView
42              android:id="@+id/tvUserEmail"
43              android:layout_width="match_parent"
44              android:layout_height="wrap_content"
45              style="@style/BodyText" />
46
47          <TextView
48              android:id="@+id/tvCredits"
49              android:layout_width="match_parent"
50              android:layout_height="wrap_content"
51              android:layout_marginTop="16dp"
52              android:textSize="18sp"
53              android:textStyle="bold"
54              style="@style/HeadingText" />
55
56      </LinearLayout>
57
58      <TextView
59          android:layout_width="match_parent"
60          android:layout_height="wrap_content"
61          android:text="@string/rental_history"
62          android:textSize="20sp"
63          android:textStyle="bold"
64          android:layout_marginBottom="16dp"
65          style="@style/HeadingText" />
66
67      <ScrollView
68          android:layout_width="match_parent"
69          android:layout_height="0dp"
70          android:layout_weight="1">
71
72          <LinearLayout
73              android:id="@+id/llHistoryContainer"
74              android:layout_width="match_parent"
75              android:layout_height="wrap_content"
76              android:orientation="vertical" />
77
78      </ScrollView>
79
80      <Button
```

```
        android:id="@+id/btnChangeUser"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/log_out"
        style="@style/SecondaryButton" />

</LinearLayout>
```

```kotlin
package com.example.assignment2.data.model

data class User(
    val name: String,
    val phone: String,
    val email: String,
    var credits: Int = 1000
)

```

```kotlin
package com.example.assignment2.data.model

import android.os.Parcelable
import kotlinx.parcelize.Parcelize

@Parcelize
data class InstrumentItem(
    val id: Int,
    val name: String,
    val imageRes: Int,
    val price: Int,
    val rating: Float,
    val category: String,
    val description: String,
    val specs: List<String>
) : Parcelable

```

```kotlin
package com.example.assignment2.activity

import android.content.Intent
import android.content.res.Resources
import android.os.Bundle
import android.view.LayoutInflater
import android.widget.Button
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.example.assignment2.R
import com.example.assignment2.data.repository.BookingHistoryManager
import com.example.assignment2.data.repository.UserManager
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

class ProfileActivity : AppCompatActivity() {

    private lateinit var tvUserName: TextView
    private lateinit var tvUserPhone: TextView
    private lateinit var tvUserEmail: TextView
    private lateinit var tvCredits: TextView
    private lateinit var llHistoryContainer: LinearLayout
    private lateinit var btnChangeUser: Button

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

        initializeViews()
        setupUserInfo()
        setupRentalHistory()
        setupListeners()
    }

    private fun initializeViews() {
```

```
38          tvUserName = findViewById(R.id.tvUserName)
39          tvUserPhone = findViewById(R.id.tvUserPhone)
40          tvUserEmail = findViewById(R.id.tvUserEmail)
41          tvCredits = findViewById(R.id.tvCredits)
42          llHistoryContainer = findViewById(R.id.
    llHistoryContainer)
43          btnChangeUser = findViewById(R.id.
    btnChangeUser)
44      }
45
46      private fun setupUserInfo() {
47          val user = UserManager.getUser()
48          if (user != null) {
49              tvUserName.text = user.name
50              tvUserPhone.text = user.phone
51              tvUserEmail.text = user.email
52              tvCredits.text = "Available Credits: ${
    user.credits}"
53          }
54      }
55
56      private fun setupRentalHistory() {
57          val history = BookingHistoryManager.
    getBookings()
58
59          if (history.isEmpty()) {
60              val textView = TextView(this).apply {
61                  text = "No rental history"
62                  setTextAppearance(android.R.style.
    TextAppearance_Medium)
63                  gravity = android.view.Gravity.CENTER
64                  setPadding(0, 32.dpToPx(), 0, 32.
    dpToPx())
65              }
66              llHistoryContainer.addView(textView)
67          } else {
68              history.forEach { booking ->
69                  val historyItem = LayoutInflater.from
    (this)
70                      .inflate(R.layout.
    item_rental_history, llHistoryContainer, false)
```

```
71
72                  val tvInstrumentName = historyItem.
   findViewById<TextView>(R.id.tvInstrumentName)
73                  val tvRentalPeriod = historyItem.
   findViewById<TextView>(R.id.tvRentalPeriod)
74                  val tvCost = historyItem.
   findViewById<TextView>(R.id.tvCost)
75
76                  tvInstrumentName.text = booking.
   instrument.name
77                  tvRentalPeriod.text =
78                      "From ${getCurrentDate()} • ${
   booking.duration} month(s)" +
79                              if (booking.insurance)
   " • With Insurance" else ""
80                  tvCost.text = "${booking.totalCost}
   credits"
81
82                  llHistoryContainer.addView(
   historyItem)
83              }
84          }
85      }
86
87      private fun setupListeners() {
88          btnChangeUser.setOnClickListener {
89              startActivity(Intent(this, LoginActivity
   ::class.java))
90              finish()
91          }
92      }
93
94      private fun getCurrentDate(): String {
95          val sdf = SimpleDateFormat("yyyy-MM-dd",
   Locale.getDefault())
96          return sdf.format(Date())
97      }
98
99      private fun Int.dpToPx(): Int = (this *
   Resources.getSystem().displayMetrics.density).toInt
   ()
```

```
100 }
```

```kotlin
package com.example.assignment2.activity

import androidx.activity.enableEdgeToEdge
import android.content.Intent
import android.os.Bundle
import android.util.Patterns
import android.widget.Button
import android.widget.EditText
import androidx.appcompat.app.AppCompatActivity
import com.example.assignment2.R
import com.example.assignment2.data.model.User
import com.example.assignment2.data.repository.
UserManager

class LoginActivity : AppCompatActivity() {

    private lateinit var etName: EditText
    private lateinit var etPhone: EditText
    private lateinit var etEmail: EditText
    private lateinit var btnLogin: Button

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


        etName = findViewById(R.id.etName)
        etPhone = findViewById(R.id.etPhone)
        etEmail = findViewById(R.id.etEmail)
        btnLogin = findViewById(R.id.btnLogin)

        btnLogin.setOnClickListener {
            if (validateInput()) {
                val user = User(
                    name = etName.text.toString(),
                    phone = etPhone.text.toString(),
                    email = etEmail.text.toString()
                )
                UserManager.setUser(user)
```

```
40                  startActivity(Intent(this,
   InstrumentItemActivity::class.java))
41                  finish()
42              }
43          }
44      }
45
46      private fun validateInput(): Boolean {
47          val name = etName.text.toString()
48          val phone = etPhone.text.toString()
49          val email = etEmail.text.toString()
50
51          if (name.isBlank()) {
52              etName.error = "Name is required"
53              return false
54          }
55
56          if (phone.isBlank()) {
57              etPhone.error = "Phone is required"
58              return false
59          }
60
61          if (email.isBlank() || !Patterns.
   EMAIL_ADDRESS.matcher(email).matches()) {
62              etEmail.error = "Valid email is required"
63              return false
64          }
65
66          return true
67      }
68  }
```

```kotlin
package com.example.assignment2.activity

import android.content.Intent
import android.content.res.Resources
import android.os.Build
import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import com.example.assignment2.R
import com.example.assignment2.data.model.InstrumentItem
import com.example.assignment2.data.repository.UserManager

class ItemDetailsActivity : AppCompatActivity() {

    private lateinit var instrument: InstrumentItem
    private lateinit var ivInstrument: ImageView
    private lateinit var tvInstrumentName: TextView
    private lateinit var ratingBar: RatingBar
    private lateinit var tvPrice: TextView
    private lateinit var tvDescription: TextView
    private lateinit var llSpecs: LinearLayout
    private lateinit var btnBorrow: Button

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

        // Get instrument from intent
        instrument = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            intent.getParcelableExtra("INSTRUMENT", InstrumentItem::class.java)!!
        } else {
            @Suppress("DEPRECATION")
            intent.getParcelableExtra("INSTRUMENT")!!
        }

```

```
36          initializeViews()
37          setupUI()
38      }
39
40      private fun initializeViews() {
41          ivInstrument = findViewById(R.id.ivInstrument
    )
42          tvInstrumentName = findViewById(R.id.
    tvInstrumentName)
43          ratingBar = findViewById(R.id.ratingBar)
44          tvPrice = findViewById(R.id.tvPrice)
45          tvDescription = findViewById(R.id.
    tvDescription)
46          llSpecs = findViewById(R.id.llSpecs)
47          btnBorrow = findViewById(R.id.btnBorrow)
48      }
49
50      private fun setupUI() {
51          ivInstrument.setImageResource(instrument.
    imageRes)
52          tvInstrumentName.text = instrument.name
53          ratingBar.rating = instrument.rating
54          tvPrice.text = "${instrument.price} credits/
    month"
55          tvDescription.text = instrument.description
56
57          // Add specs
58          instrument.specs.forEach { spec ->
59              val textView = TextView(this).apply {
60                  text = "• $spec"
61                  setTextAppearance(android.R.style.
    TextAppearance_Medium)
62                  setPadding(0, 4.dpToPx(), 0, 4.dpToPx
    ())
63              }
64              llSpecs.addView(textView)
65          }
66
67          btnBorrow.setOnClickListener {
68                  val intent = Intent(this,
    BookingActivity::class.java).apply {
```

```
69                      putExtra("INSTRUMENT",
   instrument)
70                  }
71                  startActivity(intent)
72              }
73          }
74
75          private fun Int.dpToPx(): Int = (this *
   Resources.getSystem().displayMetrics.density).toInt
   ()
76      }
```

```
 1 package com.example.assignment2.activity
 2
 3 import android.content.Intent
 4 import android.os.Bundle
 5 import android.widget.*
 6 import androidx.appcompat.app.AppCompatActivity
 7 import com.example.assignment2.R
 8 import android.view.View
 9 import com.example.assignment2.data.repository.
   InstrumentItemData
10 import com.example.assignment2.data.repository.
   UserManager
11 import com.google.android.material.chip.Chip
12 import com.google.android.material.chip.ChipGroup
13 import com.example.assignment2.data.repository.
   BookingHistoryManager
14 import com.google.android.material.snackbar.Snackbar
15
16 class InstrumentItemActivity : AppCompatActivity() {
17
18     private lateinit var tvUserName: TextView
19     private lateinit var btnProfile: Button
20     private lateinit var btnChangeUser: Button
21     private lateinit var ivInstrument: ImageView
22     private lateinit var tvInstrumentName: TextView
23     private lateinit var tvPrice: TextView
24     private lateinit var ratingBar: RatingBar
25     private lateinit var chipGroupCategory: ChipGroup
26     private lateinit var chipAcoustic: Chip
27     private lateinit var chipElectric: Chip
28     private lateinit var chipKeyboard: Chip
29     private lateinit var btnPrevious: Button
30     private lateinit var btnNext: Button
31     private lateinit var btnViewDetails: Button
32     private lateinit var btnBorrow: Button
33
34     private val instruments = InstrumentItemData.
   getInstruments()
35     private var currentIndex = 0
36
37     override fun onCreate(savedInstanceState: Bundle
```

```
37 ?) {
38         super.onCreate(savedInstanceState)
39         setContentView(R.layout.
   activity_instrument_item)
40 
41         initializeViews()
42         setupUserInfo()
43         setupNavigation()
44         displayCurrentInstrument()
45     }
46 
47     private fun initializeViews() {
48         tvUserName = findViewById(R.id.tvUserName)
49         btnProfile = findViewById(R.id.btnProfile)
50         btnChangeUser = findViewById(R.id.
   btnChangeUser)
51         ivInstrument = findViewById(R.id.ivInstrument
   )
52         tvInstrumentName = findViewById(R.id.
   tvInstrumentName)
53         tvPrice = findViewById(R.id.tvPrice)
54         ratingBar = findViewById(R.id.ratingBar)
55         chipGroupCategory = findViewById(R.id.
   chipGroupCategory)
56         chipAcoustic = findViewById(R.id.chipAcoustic
   )
57         chipElectric = findViewById(R.id.chipElectric
   )
58         chipKeyboard = findViewById(R.id.chipKeyboard
   )
59         btnPrevious = findViewById(R.id.btnPrevious)
60         btnNext = findViewById(R.id.btnNext)
61         btnViewDetails = findViewById(R.id.
   btnViewDetails)
62         btnBorrow = findViewById(R.id.btnBorrow)
63     }
64 
65     private fun setupUserInfo() {
66         val user = UserManager.getUser()
67         tvUserName.text = user?.name ?: "Guest"
68 
```

```
 69          btnProfile.setOnClickListener {
 70              startActivity(Intent(this,
    ProfileActivity::class.java))
 71          }
 72
 73          btnChangeUser.setOnClickListener {
 74              startActivity(Intent(this, LoginActivity
    ::class.java))
 75              finish()
 76          }
 77      }
 78
 79      override fun onResume() {
 80          super.onResume()
 81          showRecentBookingNotification()
 82      }
 83
 84      private fun showRecentBookingNotification() {
 85          val recentBooking = BookingHistoryManager.
    getBookings().lastOrNull()
 86          if (recentBooking != null) {
 87              val rootView = findViewById<View>(
    android.R.id.content)
 88              Snackbar.make(rootView,
 89                  "Booked successfully! Recently
    booked: ${recentBooking.instrument.name}",
 90                  Snackbar.LENGTH_LONG
 91              ).show()
 92          }
 93      }
 94      private fun setupNavigation() {
 95          btnPrevious.setOnClickListener {
 96              if (currentIndex > 0) {
 97                  currentIndex--
 98                  displayCurrentInstrument()
 99              }
100          }
101
102          btnNext.setOnClickListener {
103              if (currentIndex < instruments.size - 1
    ) {
```

```
104                 currentIndex++
105                 displayCurrentInstrument()
106             }
107         }
108 
109         btnViewDetails.setOnClickListener {
110             val intent = Intent(this,
    ItemDetailsActivity::class.java).apply {
111                 putExtra("INSTRUMENT", instruments[
    currentIndex])
112             }
113             startActivity(intent)
114         }
115 
116         btnBorrow.setOnClickListener {
117                 val intent = Intent(this,
    BookingActivity::class.java).apply {
118                     putExtra("INSTRUMENT",
    instruments[currentIndex])
119                 }
120                 startActivity(intent)
121         }
122 
123         chipAcoustic.setOnClickListener {
124             jumpToFirstInstrumentOfCategory("
    Acoustic")
125         }
126 
127         chipElectric.setOnClickListener {
128             jumpToFirstInstrumentOfCategory("
    Electric")
129         }
130 
131         chipKeyboard.setOnClickListener {
132             jumpToFirstInstrumentOfCategory("
    Keyboard")
133         }
134     }
135 
136     private fun jumpToFirstInstrumentOfCategory(
    category: String) {
```

```
137         val index = instruments.indexOfFirst { it.
    category == category }
138         if (index != -1) {
139             currentIndex = index
140             displayCurrentInstrument()
141             Toast.makeText(this, "Jumped to $
    category", Toast.LENGTH_SHORT).show()
142         } else {
143             Toast.makeText(this, "No $category
    instruments", Toast.LENGTH_SHORT).show()
144         }
145     }
146 
147     private fun displayCurrentInstrument() {
148         val instrument = instruments[currentIndex]
149 
150         ivInstrument.setImageResource(instrument.
    imageRes)
151         tvInstrumentName.text = instrument.name
152         tvPrice.text = "${instrument.price} credits/
    month"
153         ratingBar.rating = instrument.rating
154 
155         chipGroupCategory.clearCheck()
156         when (instrument.category) {
157             "Acoustic" -> chipAcoustic.isChecked =
    true
158             "Electric" -> chipElectric.isChecked =
    true
159             "Keyboard" -> chipKeyboard.isChecked =
    true
160         }
161 
162         btnPrevious.isEnabled = currentIndex > 0
163         btnNext.isEnabled = currentIndex <
    instruments.size - 1
164     }
165 }
```

```
 1 package com.example.assignment2.activity
 2
 3 import android.os.Bundle
 4 import android.util.Patterns
 5 import android.widget.*
 6 import androidx.appcompat.app.AppCompatActivity
 7 import com.example.assignment2.R
 8 import com.example.assignment2.data.model.Booked
 9 import com.example.assignment2.data.model.
   InstrumentItem
10 import com.example.assignment2.data.repository.
   BookingHistoryManager
11 import com.example.assignment2.data.repository.
   UserManager
12 import com.google.android.material.switchmaterial.
   SwitchMaterial
13
14 class BookingActivity : AppCompatActivity() {
15
16     private lateinit var instrument: InstrumentItem
17     private lateinit var tvInstrumentName: TextView
18     private lateinit var tvPrice: TextView
19     private lateinit var etName: EditText
20     private lateinit var etPhone: EditText
21     private lateinit var etEmail: EditText
22     private lateinit var radioGroupDuration:
   RadioGroup
23     private lateinit var radio1Month: RadioButton
24     private lateinit var radio3Months: RadioButton
25     private lateinit var radio6Months: RadioButton
26     private lateinit var switchInsurance:
   SwitchMaterial
27     private lateinit var tvTotalCost: TextView
28     private lateinit var btnCancel: Button
29     private lateinit var btnSave: Button
30
31     private var duration = 1
32     private var insurance = false
33
34     override fun onCreate(savedInstanceState: Bundle
   ?) {
```

```
35          super.onCreate(savedInstanceState)
36          setContentView(R.layout.booking_activity)
37
38          instrument = intent.getParcelableExtra("
    INSTRUMENT")!!
39
40          initializeViews()
41          setupUI()
42          setupListeners()
43          calculateTotalCost()
44      }
45
46      private fun initializeViews() {
47          tvInstrumentName = findViewById(R.id.
    tvInstrumentName)
48          tvPrice = findViewById(R.id.tvPrice)
49          etName = findViewById(R.id.etName)
50          etPhone = findViewById(R.id.etPhone)
51          etEmail = findViewById(R.id.etEmail)
52          radioGroupDuration = findViewById(R.id.
    radioGroupDuration)
53          radio1Month = findViewById(R.id.radio1Month)
54          radio3Months = findViewById(R.id.radio3Months
    )
55          radio6Months = findViewById(R.id.radio6Months
    )
56          switchInsurance = findViewById(R.id.
    switchInsurance)
57          tvTotalCost = findViewById(R.id.tvTotalCost)
58          btnCancel = findViewById(R.id.btnCancel)
59          btnSave = findViewById(R.id.btnSave)
60      }
61
62      private fun setupUI() {
63          tvInstrumentName.text = instrument.name
64          tvPrice.text = "${instrument.price} credits/
    month"
65
66          // Pre-fill user info
67          val user = UserManager.getUser()
68          etName.setText(user?.name ?: "")
```

```
 69          etPhone.setText(user?.phone ?: "")
 70          etEmail.setText(user?.email ?: "")
 71
 72          // Set default duration
 73          radio1Month.isChecked = true
 74      }
 75
 76      private fun setupListeners() {
 77          radioGroupDuration.
    setOnCheckedChangeListener { _, checkedId ->
 78              duration = when (checkedId) {
 79                  R.id.radio1Month -> 1
 80                  R.id.radio3Months -> 3
 81                  R.id.radio6Months -> 6
 82                  else -> 1
 83              }
 84              calculateTotalCost()
 85          }
 86
 87          switchInsurance.setOnCheckedChangeListener
     { _, isChecked ->
 88              insurance = isChecked
 89              calculateTotalCost()
 90          }
 91
 92          btnCancel.setOnClickListener {
 93              showCancellationToast()
 94              finish()
 95          }
 96
 97          btnSave.setOnClickListener {
 98              if (validateInput() && validateCredits
    ()) {
 99                  saveBooking()
100              }
101          }
102      }
103
104      private fun calculateTotalCost() {
105          var total = instrument.price * duration
106          if (insurance) {
```

```
107             total += 50 * duration
108         }
109         tvTotalCost.text = "Total Cost: $total
    credits"
110     }
111
112     private fun validateInput(): Boolean {
113         val name = etName.text.toString()
114         val phone = etPhone.text.toString()
115         val email = etEmail.text.toString()
116
117         if (name.isBlank()) {
118             etName.error = "Name is required"
119             return false
120         }
121
122         if (phone.isBlank()) {
123             etPhone.error = "Phone is required"
124             return false
125         }
126
127         if (email.isBlank() || !Patterns.
    EMAIL_ADDRESS.matcher(email).matches()) {
128             etEmail.error = "Valid email is required
    "
129             return false
130         }
131
132         return true
133     }
134
135     private fun validateCredits(): Boolean {
136         val user = UserManager.getUser()
137         val totalCost = calculateTotalCostValue()
138
139         if (user == null) {
140             showError("User not found")
141             return false
142         }
143
144         if (user.credits < totalCost) {
```

```
145             showError("Insufficient credits. You
    have ${user.credits} credits but need $totalCost")
146             return false
147         }
148
149         return true
150     }
151
152     private fun calculateTotalCostValue(): Int {
153         var total = instrument.price * duration
154         if (insurance) {
155             total += 50 * duration
156         }
157         return total
158     }
159
160     private fun saveBooking() {
161         val totalCost = calculateTotalCostValue()
162         val booking = Booked(instrument, duration,
    insurance, totalCost)
163
164         val user = UserManager.getUser()!!
165         UserManager.updateCredits(user.credits -
    totalCost)
166
167         BookingHistoryManager.addBooking(booking)
168
169         setResult(RESULT_OK)
170         finish()
171     }
172
173     private fun showCancellationToast() {
174         Toast.makeText(
175             this,
176             "Booking cancelled",
177             Toast.LENGTH_SHORT
178         ).show()
179     }
180
181     private fun showError(message: String) {
182         Toast.makeText(this, message, Toast.
```

```
182 LENGTH_LONG).show()
183     }
184 }
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Musical Instrument Rental App</string>
    <string name="get_started">Get Started</string>
    <string name="shop_now">Shop Now!</string>
    <string name="name">Name</string>
    <string name="phone">Phone</string>
    <string name="email">Email</string>
    <string name="language">Language</string>
    <string name="english">English</string>
    <string name="vietnamese">Vietnamese</string>
    <string name="profile">Profile</string>
    <string name="log_out">Log Out</string>
    <string name="category">Category</string>
    <string name="previous">Previous</string>
    <string name="next">Next</string>
    <string name="view_details">View Details</string>
    <string name="borrow_now">Borrow Now</string>
    <string name="rental_duration">Rental Duration</string>
    <string name="one_month">1 Month</string>
    <string name="three_months">3 Months</string>
    <string name="six_months">6 Months</string>
    <string name="include_insurance">Include Insurance (+50 credits/month)</string>
    <string name="cancel">Cancel</string>
    <string name="save">Save</string>
    <string name="rental_history">Rental History</string>
    <string name="description">Description</string>
    <string name="specifications">Specifications</string>
    <string name="instrument_image">Instrument Image</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ứng Dụng Thuê Nhạc Cụ</string>
    <string name="get_started">Bắt Đầu Ngay!</string>
    <string name="shop_now">Mua Sắm Ngay!</string>
    <string name="name">Tên</string>
    <string name="phone">Điện Thoại</string>
    <string name="email">Email</string>
    <string name="language">Ngôn Ngữ</string>
    <string name="english">Tiếng Anh</string>
    <string name="vietnamese">Tiếng Việt</string>
    <string name="profile">Hồ Sơ</string>
    <string name="log_out">Đăng Xuất</string>
    <string name="category">Danh Mục</string>
    <string name="previous">Trước</string>
    <string name="next">Tiếp</string>
    <string name="view_details">Xem Chi Tiết</string>
    <string name="borrow_now">Thuê Ngay</string>
    <string name="rental_duration">Thời Gian Thuê</string>
    <string name="one_month">1 Tháng</string>
    <string name="three_months">3 Tháng</string>
    <string name="six_months">6 Tháng</string>
    <string name="include_insurance">Bao gồm Bảo hiểm (+50 tín dụng/tháng)</string>
    <string name="cancel">Hủy</string>
    <string name="save">Lưu</string>
    <string name="rental_history">Lịch Sử Thuê</string>
    <string name="description">Mô Tả</string>
    <string name="specifications">Thông Số Kỹ Thuật</string>
    <string name="instrument_image">Hình Ảnh Nhạc Cụ</string>
</resources>
```

```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Assignment2" parent="Theme.Material3.DayNight.NoActionBar">
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
    </style>

    <style name="Theme.Assignment2" parent="Base.Theme.Assignment2" />
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="white">#FFFFFFFF</color>
    <color name="black">#FF000000</color>
    <color name="dark_gray">#FF666666</color>
    <color name="light_gray">#FFEEEEEE</color>
    <color name="chip_background">#FFEFEFEF</color>
    <color name="primary">#FF6750A4</color>
    <color name="primary_dark">#FF4A356A</color>
    <color name="primary_light">#FFE8DEF8</color>
    <color name="secondary">#FF7D5260</color>
    <color name="surface">#FFFFFBFE</color>
    <color name="background">#FFFDFDF6</color>
    <color name="on_primary">#FFFFFFFF</color>
    <color name="on_surface">#FF1C1B1F</color>
    <color name="outline">#FF79747E</color>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Modern Text Styles -->
    <style name="HeadingText">
        <item name="android:textColor">@color/on_surface</item>
        <item name="android:textSize">24sp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="BodyText">
        <item name="android:textColor">@color/dark_gray</item>
        <item name="android:textSize">16sp</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="CaptionText">
        <item name="android:textColor">@color/outline</item>
        <item name="android:textSize">14sp</item>
        <item name="android:letterSpacing">0.01</item>
    </style>

    <style name="EditText">
        <item name="android:textSize">16sp</item>
        <item name="android:textColor">@color/black</item>
        <item name="android:padding">12dp</item>
    </style>

    <!-- Modern Buttons - Use shape appearance instead of cornerRadius -->
    <style name="PrimaryButton" parent="Widget.Material3.Button">
        <item name="android:layout_height">56dp</item>
```

```
32          <item name="android:layout_margin">8dp</item>
33          <item name="android:textColor">@color/
    on_primary</item>
34          <item name="android:backgroundTint">@color/
    primary</item>
35          <item name="shapeAppearance">@style/
    ShapeAppearanceMedium</item>
36          <item name="android:elevation">2dp</item>
37      </style>
38
39      <style name="SecondaryButton" parent="Widget.
    Material3.Button.OutlinedButton">
40          <item name="android:layout_height">56dp</item
    >
41          <item name="android:layout_margin">8dp</item>
42          <item name="android:textColor">@color/primary
    </item>
43          <item name="strokeColor">@color/primary</item
    >
44          <item name="strokeWidth">1dp</item>
45          <item name="shapeAppearance">@style/
    ShapeAppearanceMedium</item>
46      </style>
47
48      <!-- Enhanced Chip Style - Use chipCornerRadius
     -->
49      <style name="Chip" parent="Widget.Material3.Chip.
    Assist.Elevated">
50          <item name="chipBackgroundColor">@color/
    primary_light</item>
51          <item name="chipStrokeColor">@color/primary</
    item>
52          <item name="chipStrokeWidth">1dp</item>
53          <item name="chipCornerRadius">12dp</item>
54          <item name="android:textColor">@color/primary
    </item>
55      </style>
56
57      <style name="TextInputLayout" parent="Widget.
    Material3.TextInputLayout.OutlinedBox">
58          <item name="boxBackgroundColor">@color/
```

```
58 surface</item>
59         <item name="boxBackgroundMode">outline</item>
60         <item name="boxStrokeColor">@color/outline</
   item>
61         <item name="shapeAppearance">@style/
   ShapeAppearanceSmall</item>
62         <item name="hintTextColor">@color/outline</
   item>
63     </style>
64 
65     <style name="ShapeAppearanceSmall" parent="
   ShapeAppearance.Material3.SmallComponent">
66         <item name="cornerFamily">rounded</item>
67         <item name="cornerSize">12dp</item>
68     </style>
69 
70     <!-- Card Style -->
71     <style name="Card">
72         <item name="android:background">@drawable/
   card_background</item>
73         <item name="android:elevation">2dp</item>
74     </style>
75 
76     <!-- Shape Appearances -->
77     <style name="ShapeAppearanceMedium" parent="
   ShapeAppearance.Material3.MediumComponent">
78         <item name="cornerFamily">rounded</item>
79         <item name="cornerSize">16dp</item>
80     </style>
81 </resources>
```