Swinburne University of Technology

Software Development for Mobile Devices (COS30017)

Assignment 2

Name: Mallika Arachchillage Nidula Sanketh Mallikarachchi

Year 2 Semester 2

Swinburne ID: 104756611

Email: 104756611@student.swin.edu.au

Contact Number: +61 401 876 063

Contents

Introduction

The Android Application was done as a project for a Musical Instrument Rental Store that lets users rent musical instruments for their personal wants. My application provides an easy-to-use interface for browsing through instruments, checking instrument details, renting instruments, and managing rented items.

Even though we are requested to build a basic application for the assignment, I went ahead to create a more realistic version of this application that could be deployed by a real musical rental service. In the initial request of the assignment only one instrument of the store can be viewed by the user at a time. But this can be a deal breaker for a Real-world application. But using a Grid layout which can represent many items for users to view at once would be more of a realistic use case scenario. So, I have modified the basic application to a grid view for my assignment which gives it a unique and functional interface.

This report covers the application's features, technical implementations, file structure and logic behind the key components implemented.

GitHub Link: https://github.com/SoftDevMobDev-2024-Classrooms/assignment02-nidulamallikarachchi

Application Overview

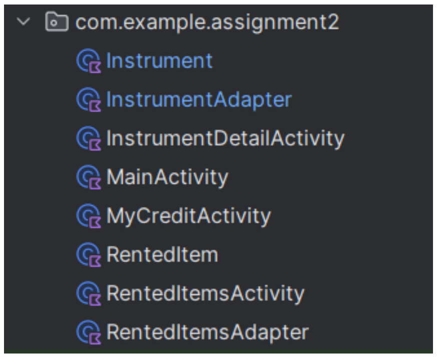
The main goal of this project was to create a functional, user-friendly android application for renting musical instruments. The application includes the following key functionalities:

- **Instrument Selection:** Users can browse and select instruments.
- **Instrument Details:** Users can view detailed information about an instrument.
- **Category Filtering:** Users can filter Music Instruments by categories using ChipGroups.
- **Real-time Renting:** Users can rent instruments instantly. Which correspondingly deducts users' credit for a month in advance.
- **Manage Rented Items:** Users can view a list of rented instruments and check the quantity of items rented.
- Store Credit Management: Users can view and add more credit if needed.

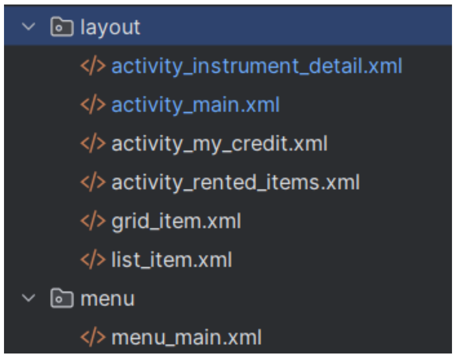
Since our application is a prototype, the payments, Rentals and Items doesn't happen in real life but only in our code. And since the application does not use any persistent data techniques, the application memory will reset when the application dies.

Application Structure

for better understanding of the applications structure and its classes, I have attached the overview of the project directories here. I will talk about the technical details and implementations done in each of these files in detail in the “Implementation Review” section.



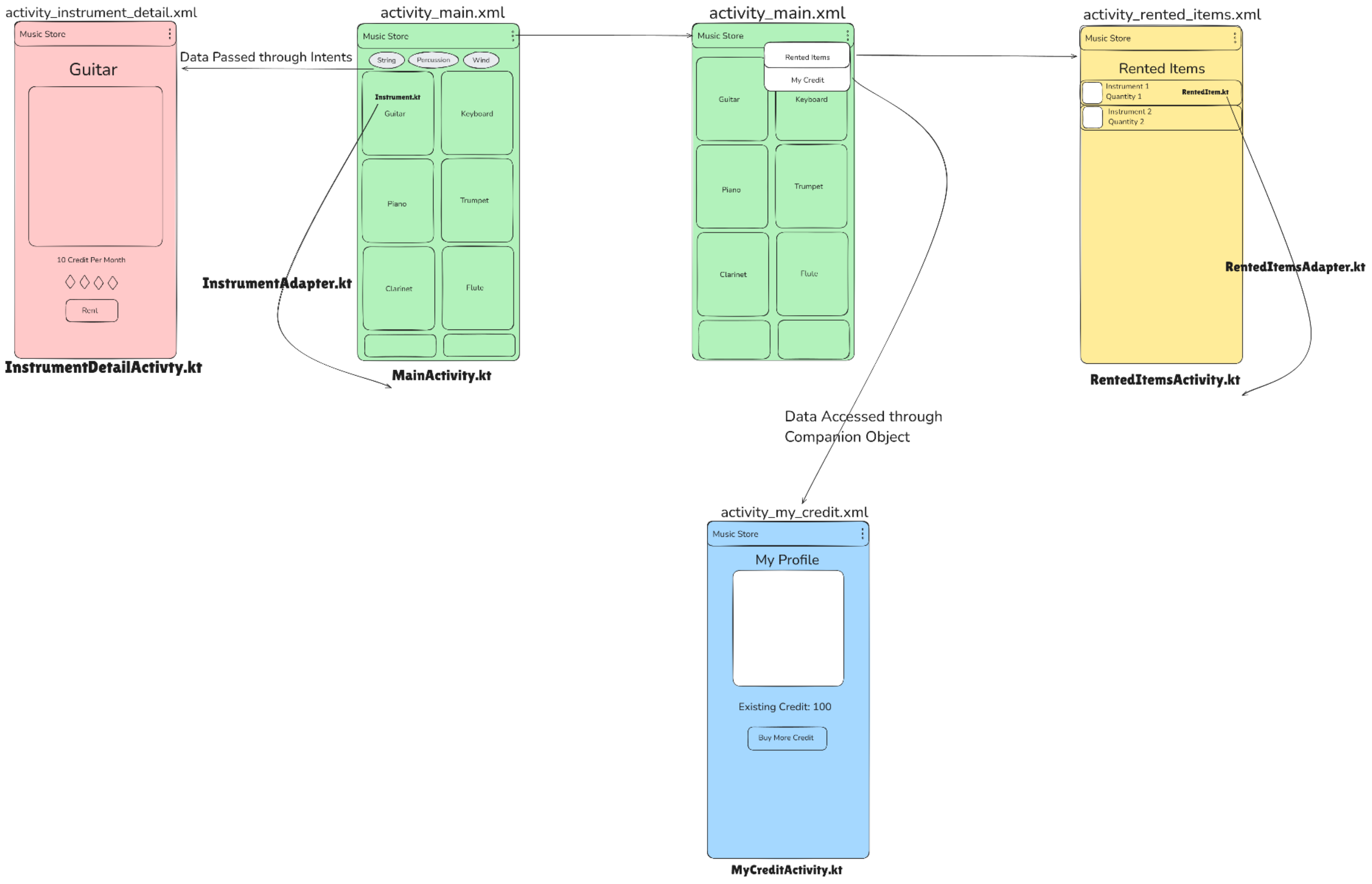
```
com.example.assignment2
    Instrument
    InstrumentAdapter
    InstrumentDetailActivity
    MainActivity
    MyCreditActivity
    RentedItem
    RentedItemsActivity
    RentedItemsAdapter
```



```
layout
    activity_instrument_detail.xml
    activity_main.xml
    activity_my_credit.xml
    activity_rented_items.xml
    grid_item.xml
    list_item.xml
menu
    menu_main.xml
```

App Prototype

This image shows a detailed prototype of how the classes are connected and how the app will function. Each activity is colored with a different color for clarity.



User Stories

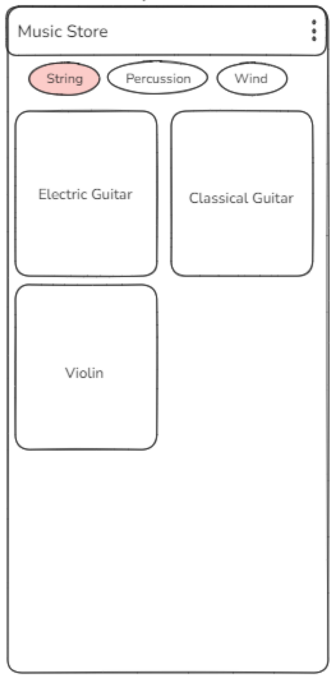
User Story 1:

*"As a music enthusiast, I want to **filter instruments** by category so that I can quickly find the specific instrument I want to rent."*

Use Case: Renting an Instrument

- **Actors**: User, System
- **Preconditions**: User is logged in and has available credit.
- **Steps**:
    1. User opens the app and views the grid of available instruments.
    2. User filters the instruments by category using the ChipGroup.
    3. User clicks on an instrument to view details.
    4. User clicks the "Rent" button.
    5. System prompts the user with a confirmation dialog.
    6. User confirms the rental.
    7. System deducts the relevant credit and updates the stock.
    8. System redirects the user to the main screen, displaying a success message.

activity_main.xml



MainActivity.kt

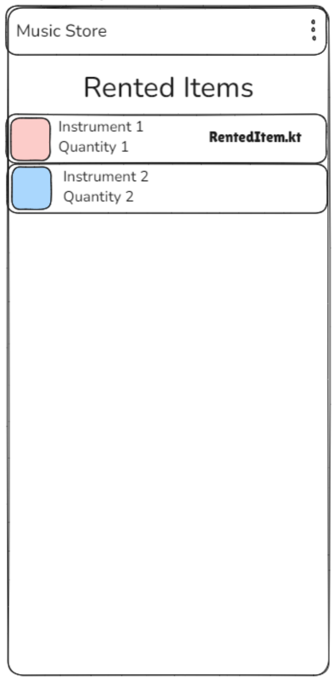
User Story 2:

*"As a store customer, I want to see my rented instruments in one place so that I can easily manage them and **track my rentals**."*

Use Case: Adding Store Credit

- **Actors**: User, System
- **Preconditions**: User is logged in and has less than 100 credits.
- **Steps**:
    - The user opens the app and navigates to the "My Credit" section via the menu.
    - User clicks the "Buy Credit" button.
    - System displays a confirmation dialog.
    - User confirms the purchase.
    - System adds 20 credits to the user's account.
    - System displays the updated credit balance.

activity_rented_items.xml



RentedItemsActivity.kt

Implementation Review

Section 1: Understanding how MainActivity & Item Binding works

`Instrument.kt`

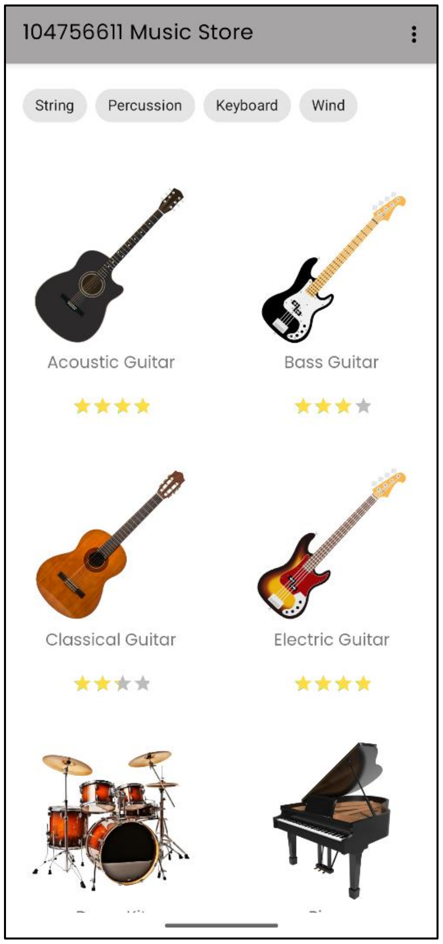
Instrument Class is a data model that is responsible for creating instrument objects to hold their values. An instrument has the following attributes

- ***imageId:*** Reference to the instrument image in res/drawable
- ***instrumentName:*** Name of the Instrument
- ***stock:*** Number of available instruments for rent
- ***instrumentCredit:*** Monthly rental credit per instrument
- ***instrumentRating:*** User Ratings for the instrument (This is not a rated by user real users)
- ***instrumentCategory:*** Category which the instrument belongs to

`grid_item.xml`

This layout file defines the UI for displaying an instrument in a grid view. This xml file uses *LinearLayout* as its root element and it includes the following UI Elements:

- **ImageView:** for the instrument image
- **TextView:** for the instrument name
- **RatingBar:** for displaying the instrument rating



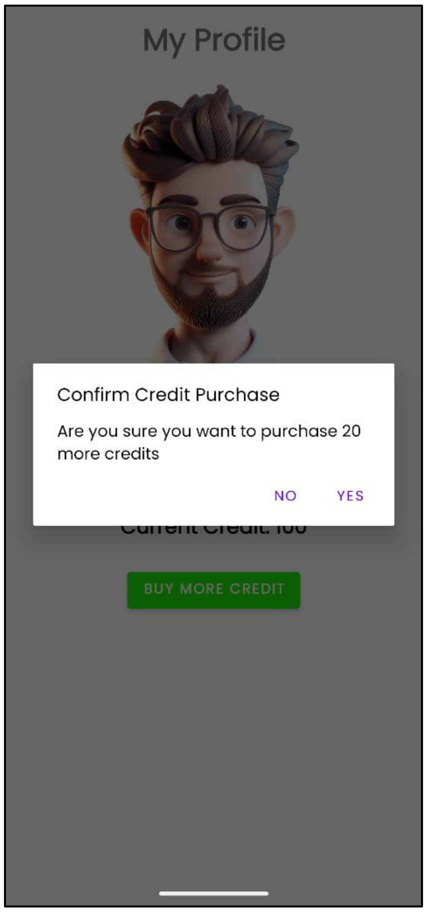
`MainActivity.kt`

The MainActivity class is the entry point of the application. Its key features are:

- Companion Objects
  - A list of instrument objects *"instruments"*
  - Default user credit of 100 *"userCredit"*
  - A mutable list to hold rented items *"rentedItems"*
- Toolbar Setup
  - The toolbar includes menu options such as viewing rented items and checking store credit.
  - Even though I am using my own theme in this app, the parent theme of the application is a NoActionBar theme. Because of this I have to setup Toolbar in order to customize it's style
- Adapter Setup
  - *InstrumentAdapter* is used to display instrument data
- Category Filtering

- ChipGroups are dynamically generated by iterating through all the object categories in the array so that the UI Elements don't need to be added manually.
- When a chip is selected it triggers a the filtering function defined in the code

- GidView Handling: When the user clicks on an instrument, the page is directed to the *InstrumentDetailActivity.kt* Activity
- Menu Options: The Menu allows users to navigate to the "Rented Items" and "My Credit" sections of the app



`InstrumentAdapter.kt`

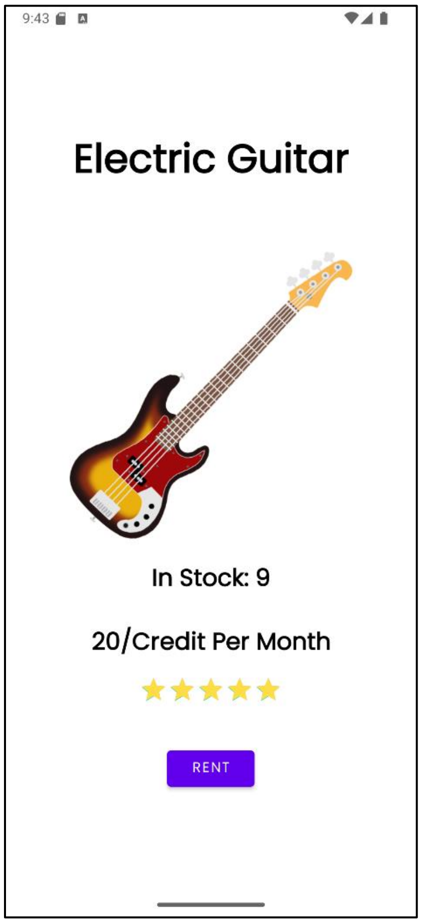
- This class is responsible for the binding of the list of instrument objects to the grid view in *MainActivity.kt* with the layout defined by *grid_item.xml*
- This is an extension of the BaseAdapter() class therefore all the common adapter methods such as getCount(), getItem(), getItemId() and getView() are overridden with the necessary functionalities.
- Also a updateInstruments() method is defined by me to use for filtering with ChipGroups

Section 2: Understanding How the Credit System Works

`MyCreditActivity.kt`

This activity lets the user buy more credit if they run out of credit. Since this is a prototype, the functionality is not actually implemented.

- Buy Credit Functionality: Users can add 20 credits at a time.
- Dialog Management: The *DialogBuilder* is used to create and display the confirmation dialog to ask the user to confirm the purchase or cancel.



Section 3: Understanding How the Renting Process Works

`RentedItem.kt`

The *RentedItem* class is used to represent the instruments user has rented. It has the following attributes

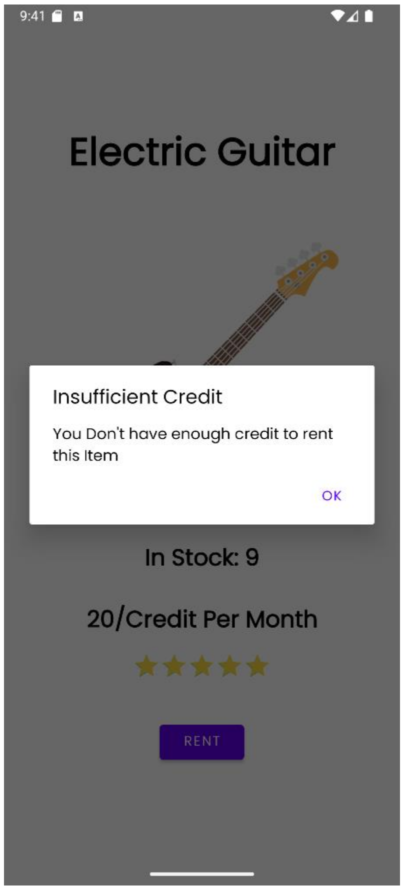
- imageId: Reference to the instrument Image
- instrumentName: Name of the Instrument
- quantity: Number of Units Rented from a single instrument

InstrumentDetailActivity.kt

This activity is responsible for displaying detailed information about the instrument and allows the user to rent the instrument.

- All the data is passed to this activity through ***Intents***

- The Content of this file is in the *activity_instrument_detail.xml* file which contain the following Views:
    - Text View: for instrument name
    - Image View: for instrument Image
    - Text View: for instrument Stock
    - Text View: for instrument Credit Per Month
    - Rating Bar: for instrument Rating
    - Button: for renting the instrument
- The rental process is handled by this activity.
- Before renting begins, the system will check if the user has enough credit to proceed renting. If not, an alert message will inform the user that they have no sufficient credit for this purchase.
- When the user presses the Rent Button a dialog will pop up asking the user to confirm the rental. If the user confirms "Yes" the following will happen
    - *instrumentStock* of the relevant instrument will reduce by 1
    - *userCredit* is deducted according to the amount of the instrument credit
    - The Item is added to the rentedItems mutable list in the MainActivity.kt which is a companion object
    - A Toast Message will be displayed saying "Your Item is ready for Pick Up".
    - The user will be redirected to the Main Activity.
- If the user cancels the rental
    - Then a toast message will display to show cancellation, and the user will be redirected to the main activity.


9:41

Electric Guitar

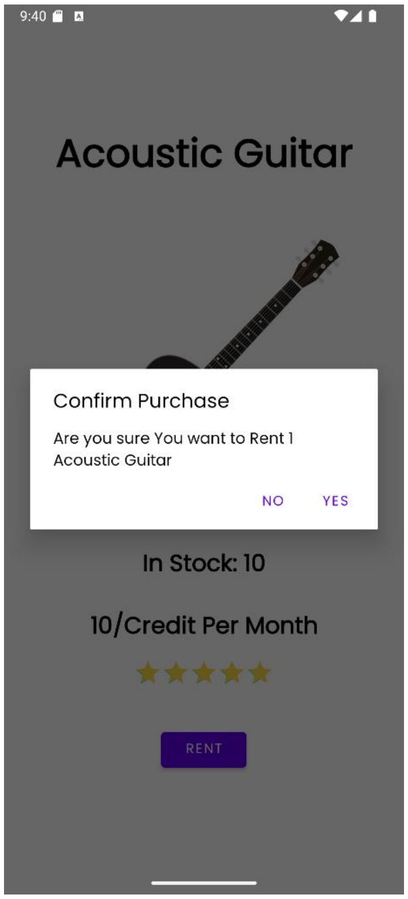
Insufficient Credit

You Don't have enough credit to rent this Item

OK

In Stock: 9

20/Credit Per Month


9:40

Acoustic Guitar

Confirm Purchase

Are you sure You want to Rent 1 Acoustic Guitar

NO YES

In Stock: 10

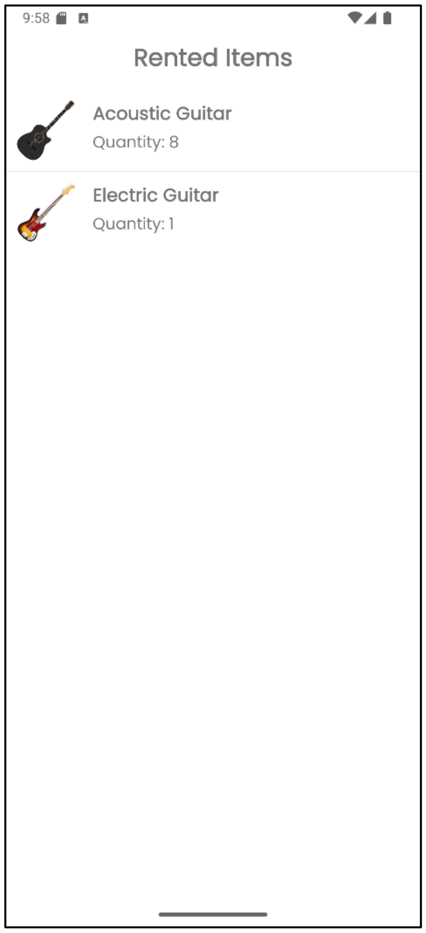
10/Credit Per Month

Intents from MainActivity.kt to InstrumentDetailActivity.kt

```
intent.putExtra("INSTRUMENT_IMAGE_ID",
selectedInstrument.imageId)
intent.putExtra("INSTRUMENT_NAME",
selectedInstrument.instrumentName)
intent.putExtra("INSTRUMENT_STOCK", selectedInstrument.stock)
intent.putExtra("INSTRUMENT_CREDIT",
selectedInstrument.instrumentCredit)
intent.putExtra("INSTRUMENT_INDEX", position)
intent.putExtra("INSTRUMENT_RATING",
selectedInstrument.instrumentRating)
```

Section 4: Understanding Rented Items Activity

RentedItemsActivity.kt

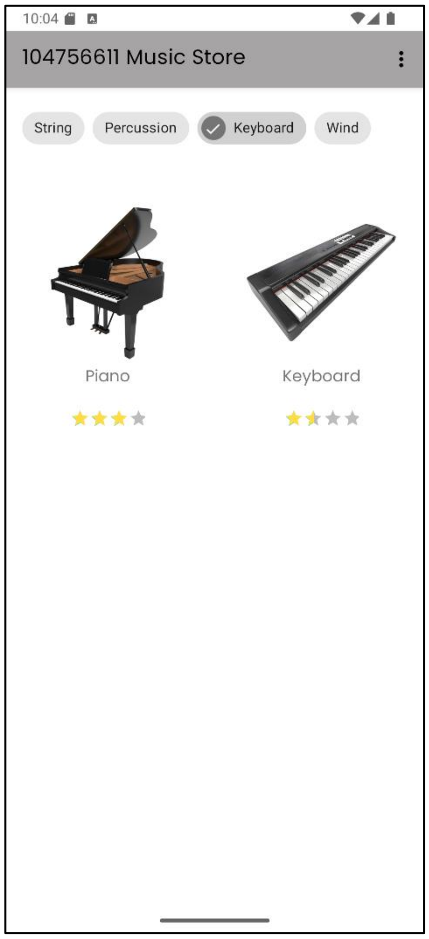

Rented Items

Acoustic Guitar
Quantity: 8

Electric Guitar
Quantity: 1

This class is responsible for viewing the items that the user has already rented.

- The file uses *"activity_rented_items.xml"* content file
- This activity uses a list view to represent the items.
- The list items on the list view of this xml file uses the *"list_item.xml"* file as the layout for a single item.
- The "list_item.xml" uses a linear layout as the root layout and it has an image view to show the item image, a text view for instrument name, and another text view for the item quantity.
- The "RentedItemsAdapter.kt" is used as the adapter for this activity to bind the object to the activity. This adapter uses the rented items list companion object in the main as its list to initialize objects.


104756611 Music Store

Challenge Task

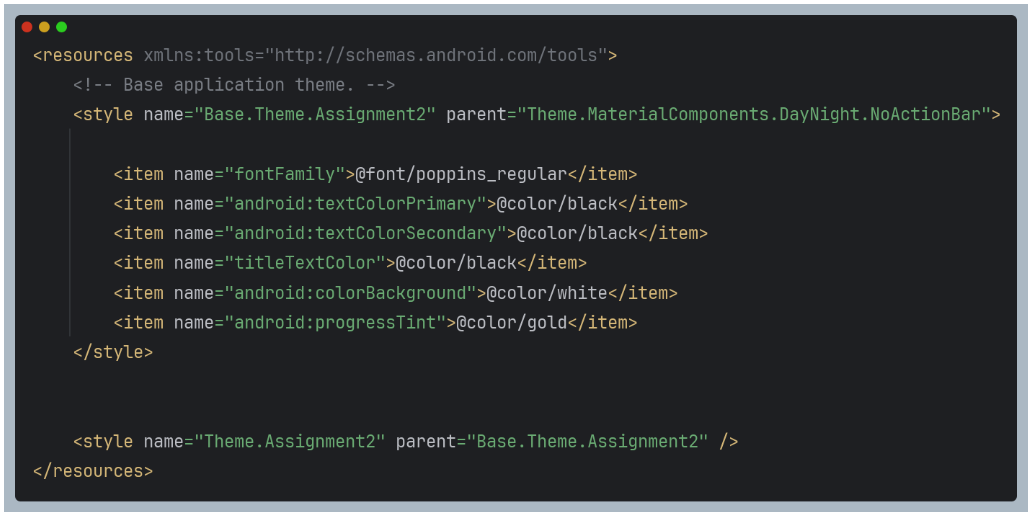
- This implementation is one of the most common user-friendly approaches that can be implemented in an app. This makes the searching process quick and easy for users. Thus, making it easy to navigate.
- The list of chips is dynamically updated as new instruments are added to the instruments array, this is done so that if the store owner wants to add many other instruments, it will not be a problem to sort the application in the future.

  Code:

```
val categories = instruments.map
{ it.instrumentCategory }.distinct()
val chipGroup = findViewById<ChipGroup>(R.id.chipGroup)

for (category in categories) {
    val chip = Chip(this).apply {
        text = category
        isCheckable = true
    }
    chipGroup.addView(chip)
}
```

Themes

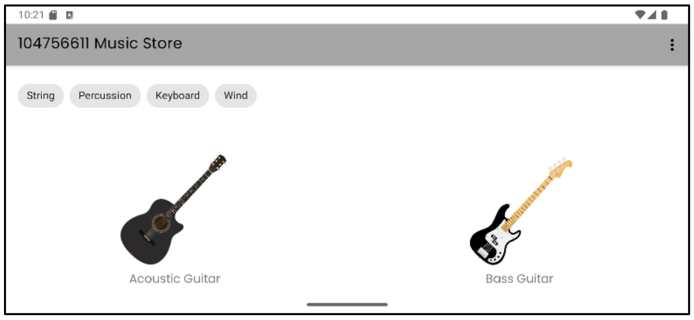
The app uses a custom minimalist theme that was designed by me. The themes do not have a lot of changes and it uses the Day Night Theme as the Parent Theme so that the only UI elements that the theme applies to are the once that I chose.

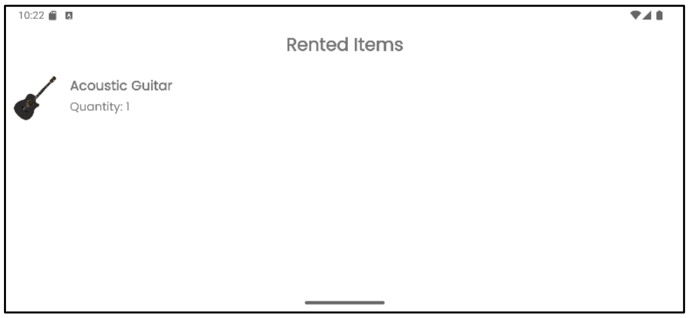


```xml
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Assignment2" parent="Theme.MaterialComponents.DayNight.NoActionBar">

        <item name="fontFamily">@font/poppins_regular</item>
        <item name="android:textColorPrimary">@color/black</item>
        <item name="android:textColorSecondary">@color/black</item>
        <item name="titleTextColor">@color/black</item>
        <item name="android:colorBackground">@color/white</item>
        <item name="android:progressTint">@color/gold</item>
    </style>


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

Landscape Mode

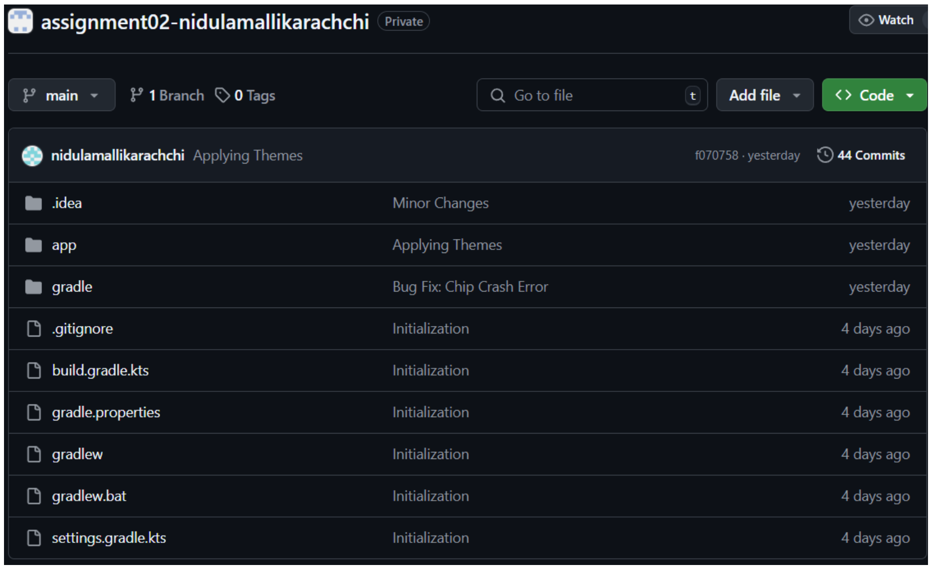
The app performs well in the landscape mode and also preserves its state





GitHub

The project took more than 4 days for me to completely figure out. But since the first few projects were not successful, the app was built from scratch again. The last attempt took only 1 day to complete with a few bugs but they are fixed as of now. The project is committed to GitHub with timely details.



assignment02-nidulamallikarachchi Private

nidulamallikarachchi Applying Themes f070758 · yesterday 44 Commits

.idea	Minor Changes	yesterday
app	Applying Themes	yesterday
gradle	Bug Fix: Chip Crash Error	yesterday
.gitignore	Initialization	4 days ago
build.gradle.kts	Initialization	4 days ago
gradle.properties	Initialization	4 days ago
gradlew	Initialization	4 days ago
gradlew.bat	Initialization	4 days ago
settings.gradle.kts	Initialization	4 days ago

Use of Generative AI

ChatGPT or any other Generative AI tool has not been used for this report whatsoever (Everything is handwritten by me)

ChatGPT has been used to gain knowledge about new concepts that I have not studied before about android development like Adapters, Intents, Companion Objects and other.

The assistance of ChatGPT is taken in debugging and troubleshooting the actual code mistakes but the code is written by myself with appropriate comments

References

Material Design - Version 2. (n.d.). Material Design. https://m2.material.io/components

Education is Life. (2020, December 23). *01 Android Studio Installation - Building your first Android app in Kotlin | Android App Development* [Video]. YouTube. https://www.youtube.com/watch?v=7pcKH0cQE6Y