## 50.001 Team 2F

### FifthBro

#### Members

John Lim Jie Sheng 1004602

Saw Yi Xuan 1004655

Cheh Kang Ming 1005174

Alphonsus Tan Yong Shing 1005534

Tron Ng 1004298

Theresa Lam 1005477

Acqquilaa Bathumalai 1005337

[**GitHub Repository**](https://github.com/cymstx/InfoSys1D)

**Description**

Currently, communication between Fifth Rows and Student Interest Groups are limited to telegram groups and email chains – this can make resource sharing a challenge.

With FifthBro, we aim to promote organisational sharing by providing a hub for Fifth Row members to share and request items.

**Implementation**

Firebase

Authentication (Auth)

We used Firebase Authentication SDK to create and manage user instances. Each user instance will be created with their email address and password. Upon each successful creation, a unique userID will be generated by Auth and each user instance will be stored in Firebase Realtime Database under its own userID for future referencing.

Realtime Database (RTDB)

We used RTDB as our database. As RTDB does not allow us to retrieve a parent class with a child class inside, we decided to flatten our RTDB structure as much as possible. Instead of having each item instance store a list of booking instances, each item instance should just contain a list of bookingIDs, which are of type string. When retrieving the item instance from RTDB, we can then get the required bookingID from the list of bookingIDs and retrieve the corresponding booking instance stored in RTDB under another reference.

To retrieve an object from RTDB, an asynchronous listener is attached to the RTDB reference, the listener is triggered once for the initial state of the data and again anytime the data changes. Upon each trigger, a callback function receives a DataSnapshot, which is a snapshot of the data. The data is then marshalled into our desired class through the overloaded getValue() method of the DataSnapshot class.

Classes

Some of our Java classes are made to implement the Java Serializable interface to allow for the object to be passed to the next activity via explicit intent.

User Class

User Class is instantiated upon registering a new user. Each registered user is a user instance. Each user instance has a unique userID generated by Firebase Authentication. Each user instance has its own ‘bookings’ attribute that is an array list of bookingIDs.

Club Class

Club Class is instantiated upon registering a new user. There will be one club instance per Fifth Row. Each club instance has an array list of item objects and user objects, to keep track of its inventory and members respectively.

Item Class

Item Class is instantiated upon adding a new item to the club’s inventory. Each item instance has its own itemID generated by Java’s UUID class upon creation. Each item instance has its own ‘log’ attribute that is an array list of bookingIDs.

Booking Class

Booking Class is instantiated upon each successful booking made. Each booking instance has its unique bookingID generated by Java’s UUID class. Booking objects consists of importation information that is required for validation (which user made the booking, which item is this booking under, the club of this item, start and end time of the booking, etc).

Time Period Class

TimePeriod Class is instantiated upon each successful booking made. Each time period instance is tied to its corresponding bookingID. Consists of the start and end time of the booking stored in a long data type.

**Third Party Libraries Used**

Date Time Picker (com.github.florent37:singledateandtimepicker:2.2.7)

Using a highly customisable builder. It allows us to create a simple dialog that allows us to set its own minimum and maximum timestamps. We can also remove certain parameters such as the year, thereby removing redundant steps. It also gives us control over its appearance and allows us to match our theme. This simple dialog makes the date time selection simpler and quicker compared to the separate date and time pickers in the first party libraries.

QR Scanning/QR Generation (com.github.yuriy-budiyev:code-scanner:2.1.0) (com.github.kenglxn.QRGen:android:2.6.0)

The QR scanning and QR generation libraries used are both built on top of ZXing, a barcode scanning library for Java that allows Android devices with imaging hardware to scan barcodes and retrieve the information decoded. ZXing is able to scan QR codes, MaxiCode and EAN-13 1D barcodes.

QR Functionality: For QR generation of an item, the item’s uniqueID is fetched from Firebase in a string and stores it as a Bitmap object. This Bitmap object is then used inside an ImageView object to display the generated code on screen. The QR scanner class decodes the scanned QR code and stores it in a string. The string can be used in the validation process of checking in/checking out the item.

**Design Patterns**

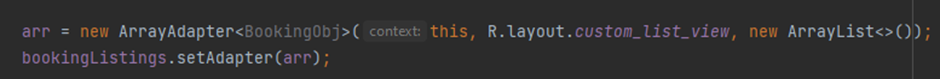
Single Responsibility Principle (SRP)

Our codebase is split into many Java classes, one for each feature. This allows for multiple members to work on different features without code overlap. Individual features can also be modified without breaking other features. SRP allows for easier debugging as bugs can be traced to a single feature/class.

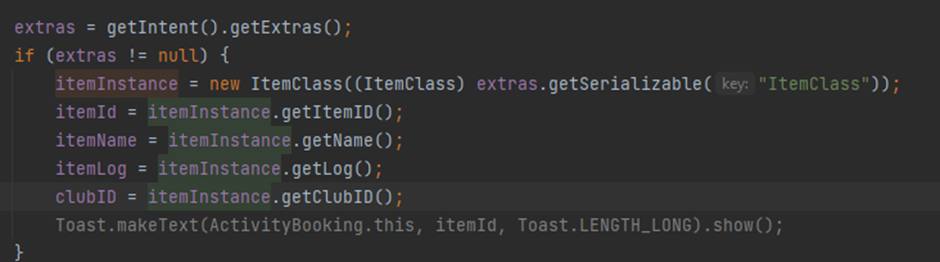
Adapter and Observer Design Patterns

ActivityBooking.java

An adapter was used for a ListView to show the bookings from the item log. Instantiated at the top of onCreate(), the custom\_list\_view is a simple text view widget for each list entry.

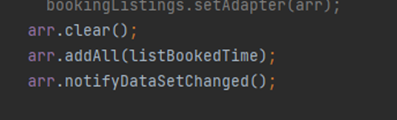


The item in question is brought in from the previous activity intent.



We are interested in the item log, which contains the bookingIDs of the booking objects we want to print.

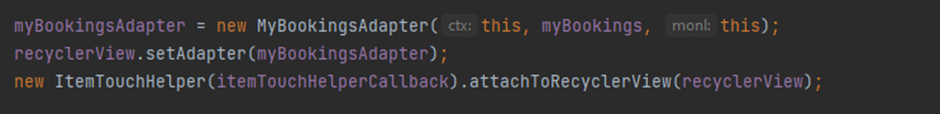
The printBookings() function is used to draw items in the ListView whenever we enter the activity or complete a new booking. We have an addValueListener on the referenceBookings instance and an anonymous onDataChange() to pull the booking objects from referenceBookings into a sorted list based on their start time. Using the ArrayAdapter, we clear it of its previous entries, then add the list and notify the ArrayAdapter that data has changed for the Adapter to redraw the ListView.



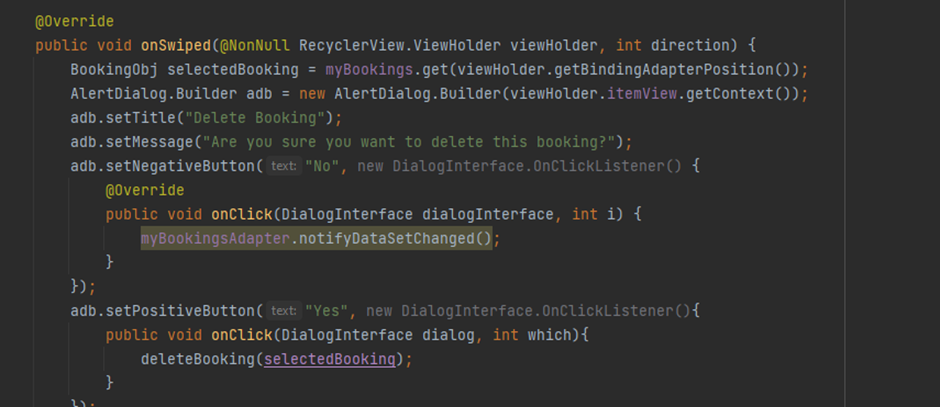
ActivityDisplayBookings.java

The activity gets the user instance from the intent of the previous activity.

We instantiated the RecyclerView adapter for the RecyclerView widget, and the ItemTouchHelper for user interactions.



Using the list of bookingIDs from the user instance, we call the booking instances into a list to be put into the Adapter. We then notify the Adapter that we have updated, to draw the objects. We use the onSwiped() function to delete a booking instance from firebase. Upon swiping, a simple alert dialog prompts for confirmation. Upon confirmation, the position of the booking object is passed to the list which returns the booking instance. The booking instance is then removed from firebase and the list, the RecyclerView is then updated to show the new booking list.

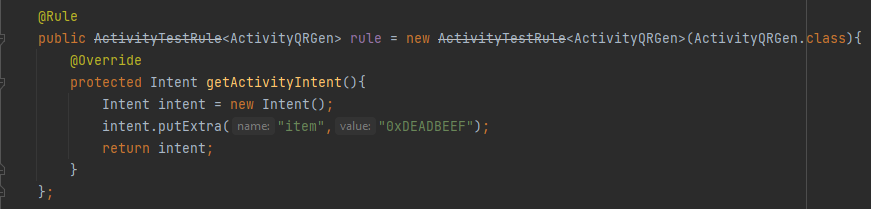


Builder Design Pattern

In our ActivityBooking.java, we used the Builder Design Pattern in the implementation of Calendar picker instances for setting the booking. The builder stores the custom characteristics of the dialog screen to be drawn.

**Unit and Instrumented Testing**

Unit tests using JUnit4 were written for BookingObj, ClubClass and UserClass to validate functionality in the face of continual code changes. We hope to improve test coverage in the future.

Instrumented tests were written for ActivityLoggedIn, ActivityQRGen and MainActivity to ensure elements of the User Interface were displayed correctly to the user when performing different functionalities. For activities that relied on extras from Intents, intent was overridden using ActivityTestRule. 

**Tasks**

John Lim : BookingActivity implementation

Tron Ng: QR Generation and Scanning, Testing, code debugging

Theresa Lam: Inventory implementation

Saw Yi Xuan: Generate and Scan QR Code + Connect to databases researches, Presentation Slides, Short Project Info Document

Cheh Kang Ming: Integrated classes required with firebase read and write queries, displaying objects retrieved from firebase to recycler view, code debugging

Acqquilaa Bathumalai: Poster, video and research

Alphonsus Tan: Integrated classes required with firebase read and write queries, logic for verification of checking in/out items, code debugging

**Appendix**

Firebase Structure

|  |  |  |
| --- | --- | --- |
|  |  |  |

System Architecture

|  |
| --- |

QR scanning/QR generation

| Screen of owner’s phone | Screen of lender’s phone |
| --- | --- |
|  |  |