OCR A-Level Computer Science Programming Project

Palvinder Singh Sander

CANDIDATE NUMBER : 2212
CENTER NUMBER : 20866

Contents

Chapter 1

Analysis

1.1 Problem Definition

Over the course of this development life cycle I will be creating an android application, that is principally a platform for users to share books, within the school community. Many students have spare books at home that will not be used again; these books could be rented to others. A utility that conveniently allows for this exchange does not exist for three primary reasons, it must be: intuitive, secure and fair. To overcome these issues, I plan implementing features that allows users to: create an account, add their books by scanning them or otherwise, search for and request a book to rent and rate others based on their quality. This will come hand in hand with rigorous policies; these will ensure that the user has an incentive to add more books and be compliant with book return deadlines.

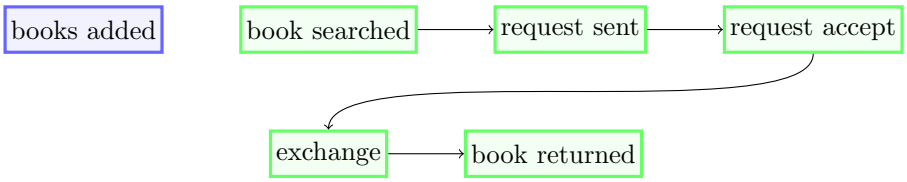


Figure 1.1: Proposed System

Throughout the development process I will document the initial stages of defining the problem to the design, development and testing of prototypes that will iteratively be built up to a final product. This process is reminiscent of the Rapid Application Development methodology. I will undoubtedly face problems that I will have to overcome to create a stable piece of software. I will attack these problems with a logical mindset, supported by the use of computational methods. This process will also be documented and analysed throughout. Although this may be a very technical process, I must be able to abstract away such details in order to consult my stakeholders about the key facts and motives behind my decisions.

1.2 Stakeholders

Stakeholders are individuals, part of your target demographic, that have an active involvement and interest in my project. They will be the ones who finally test the end program, but also will test prototypes to identify errors and requirements of the application in order to improve it.

1.2.1 Students - Upper Years

An important subset of the overall school population are upper year students (year-11 and above). These students have important assessments and therefore academic books are of most importance to them. These students will often only need revision guides just before and during exam season; the process of finding appropriate resources is both time consuming and expensive.Therefore it would be beneficial to them to be able to access this material for free. The added benefit of books being supplied by other students means that students can have confidence that they have been effective in others' revision. I expect that users of this age bracket will want a dependable and strict system in order to reliably use the renting services. My nominated stakeholders of this category are: *Lewis Jones*, *Shashi Balla* and *Jeevon Grewal*; all of which are studying A-Level subjects.

1.2.2 Students - Lower Years

Lower years (year-7 to year-10) often have spare time to read fictional books. With high demand for popular books, such as the *Harry Potter* series, it can often be hard to find these books in the existing school library. This usually results in students buying their own expensive copy that will be read and then not used again. Therefore, my application could be used to stop the perpetuation of this problem by circulating these books around the school. For this reason the expectations of my application will be focused on having a large variety of available books. I will gauge with a year-10 student, *Joe Millerchip*, as a stakeholder.

1.2.3 Teachers

Although teachers may not use my application directly they may be potentially affected by students' use. If my solution is not reliable or secure enough, there may be resulting issues between students; this will ultimately become the staffs' responsibility to handle. This would therefore be a nuisance to both students and teachers. For this reason consulting staff throughout will allow me to implement the necessary requirements to avoid such situations occurring. I will be consulting *Mr Jones*, as a stakeholder. *Mr Jones* is an English teacher and can therefore also help me with deciding on relevant details about books etc.

1.3 Computational Methods

My project will be created primarily using the programming language Java, due to its native support on android devices, and the Integrated-Development-Environment Android Studio, as it is the officially supported development platform. The nature of my application means that there will be a heavy focus on the Graphical-User-Interface. In android each activity (page displayed on the screen), is comprised of an XML layout file and a corresponding class, inherited from a parent AppCompat class. This is a example of modularization that will have to exist within my code. This will come hand in hand with the cornerstones of computational thinking: abstraction, decomposition, computational methods and algorithms. A computer would suit my project, as opposed to asking around and finding a book you can borrow, because such a process as this process complex and time consuming and may not account to anything, for example if you find a book but a person is not willing to lend it. A computer can also be used to quickly identify books as well as enforce organization, something that would otherwise be prone to human error.

1.3.1 Abstraction and Visualization

Abstraction is the process of separating concrete instance of a solution from ideas that were used to solve them. In the case of development, I can use abstraction to take details away from reality, leaving the necessary solution to the problem I am solving. There 3 main details I can abstract away when creating my application:

1. Adding and Searching for Books
    - By implementing an ISBN scanner the need for the user to interpret and input various details about the book is removed, thus making the process much simpler. This also extends across to searching for books as they can simply search for the book and immediately see the results. If this was a real-life scenario, the user would most likely have to traverse a Library to find the book; thus this digital process can be abstracted away from the user.
2. Exchange Management
    - The actual transaction of the book does not need to occur under supervision of the application , rather it is the events leading up and proceeding the exchange that should be moderated. These events include: submitting a request for the book, setting up a time and place to meet.
3. User Rating
    - Users must be held recognized for the service they provide, good or bad. Users could do this in person or by discussing it with others. This is a hit or miss way of dealing with this system. Given its importance, in maintaining a fair system, I must implement this system within the application itself. I expect this do be done in a simple and easy to understand way, possibly a quantifiable rating (1-5).

Overall, considerable aspects of the application can be seen as an abstraction of how this process would function in real-life. This will in turn make the problem of renting books much more simple and intuitive.

1.3.2 Logical Approach

Logic will be of great importance throughout this process. There will be clear stages of decision making, for example handling and validating user inputs appropriately, navigation through different screens and searching for data. By doing this I will be able to create an efficient and purposeful solution.

1.3.3 Thinking Ahead

Thinking ahead involves identifying the inputs and making the correct outputs are displayed. The main inputs for my application will vary from user details to search parameters or a bar code

image. The outputs from this would be displayed to the user in the form of an activity change, confirmation message, notification or otherwise.

Table Of Input Requirements		
Input	Stage	Requirement
User Details	Account Creation	Details about the user such as email, password, year-group and date-of-birth will be needed for account creation
ISBN/Camera	Adding Books	ISBN must be scanned by device's camera in order to find the book
Book Details	Adding Books	As a fail-safe, data about the book must be entered manually
Book Details	Book Search	ISBN, book title or author can be input to search for a book
Renting Details	Rent Request	Details such as period of renting and where the exchange should occur must be communicated between users
User Rating	Experience Rating	Given a certain scale, the user should rate the experience of renting with the user providing or using the book

1.3.4 Algorithms

Algorithms will make up a huge part of my project, so that all of the decomposed sub-procedures can be designed in pseudo code/flowcharts during the planning and design stage. In computational thinking, algorithms are simple steps that fit together to create the whole program. I will most likely use a sorting algorithm (bubble/insertion/merge/quick sort) to sort books into alphabetical or ranked order, a searching algorithm (linear/binary search) to search for specific details from an array of books and also other algorithms for different more discrete tasks such as fetching ISBN data.

1.3.5 Software Development Methodology

I will follow the Rapid Application Development methodology when creating my project. This is the best methodology for my program because it involves making lots of prototypes and relies heavily upon testing once all the analysis and design administration is complete, in order to get a high quality, functioning program in the end. The speed with which programming takes place is high so there will be little pressure on ensuring the actual project gets complete within the deadline. The model will also incorporate some of the Agile Methodology in that it may require me to do some more research before creating new prototypes to be implemented and tested, resulting in a very high quality solution, combined with the excellent client usability from using RAD. This way I get the most out of two of the quickest low risk methods for a small project like my own.

1.3.6 Decomposition

Functional decomposition is when a problem is broken down into its various sub-parts till a difficult problem turns into lots of solvable problems. My application can be broken down into distinct chunks; these could later be represented as different classes, functions or methods. These areas include:

1. Account Creation
2. Adding a Book
3. Searching a Book
4. Requesting a Book
5. Renting a Book
6. Rating a User

These distinct stages bring their own challenges thus can be decomposed even further. This will ultimately make the process of visualizing the layout of the application along with the background processes much more apparent; this will be particularly useful in developmental stages. Decomposing the problem will also present itself to the end-user as it will result in a clearer and more intuitive experience.

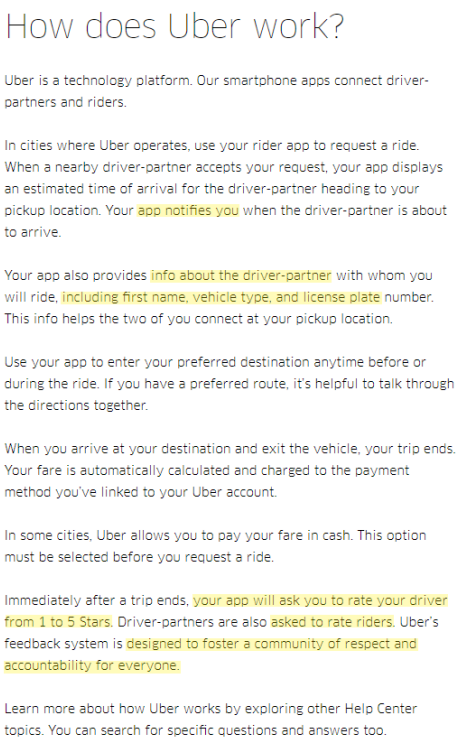
1.3.7 Heuristics

The underlying premise of promoting sharing between users is principally based upon trust between users. Although my application is targeted at legitimising this and using it to share books, there is always room for a situation where things may not go according to plan; a user may not return a book or attend an exchange. This makes my solution inherently heuristic. However, it is still possible to overcome these issues, e.g. implementing a user rating system.

1.4 Similar Products

1.4.1 Uber

Although completely unrelated to books, Uber is a prime example for understanding how a trust based system operates along with other features such as a user rating system. The way in this is presented to the user, user interface, is also useful. On Uber's main website is an article[1] briefly outlining it's functionality.



How does Uber work?

Uber is a technology platform. Our smartphone apps connect driver-partners and riders.

In cities where Uber operates, use your rider app to request a ride. When a nearby driver-partner accepts your request, your app displays an estimated time of arrival for the driver-partner heading to your pickup location. Your app notifies you when the driver-partner is about to arrive.

Your app also provides info about the driver-partner with whom you will ride, including first name, vehicle type, and license plate number. This info helps the two of you connect at your pickup location.

Use your app to enter your preferred destination anytime before or during the ride. If you have a preferred route, it's helpful to talk through the directions together.

When you arrive at your destination and exit the vehicle, your trip ends. Your fare is automatically calculated and charged to the payment method you've linked to your Uber account.

In some cities, Uber allows you to pay your fare in cash. This option must be selected before you request a ride.

Immediately after a trip ends, your app will ask you to rate your driver from 1 to 5 Stars. Driver-partners are also asked to rate riders. Uber's feedback system is designed to foster a community of respect and accountability for everyone.

Learn more about how Uber works by exploring other Help Center topics. You can search for specific questions and answers too.

Figure 1.2: Official *'How does Uber Work?'* Article

The highlighted yellow text brought important features to my attention:

1. Use of notifications
    - The app provides a clear to communicate between the user and the app. I could adapt this in my project, to notify the user about requests, period of renting remaining and deadlines.
2. User information
    - Users are able to see details about the user they are interacting with, in this case this is *First Name, Vehicle Type and License Plate*. These are essentially identifiers for both users. I could therefore allow information such as *Name, Year Group and Form* to be revealed between users. An important thing to note is that, users cannot actively search for users rather details are revealed by searching another service. This means that users should be able to search for books and then from there user details should be revealed.
3. User rating
    - User rating is actively enforced to, as stated, *foster a community of respect and accountability for everyone*. I believe that away from the service provided, this is the most important aspect of the app. It ensures that users have confidence with the service they are receiving and can also have an impact themselves by highlighting a poor experience. The user of *1-5 stars* is something that I could easily implement and would similarly be easy for the user to interact with. The article also makes clear that the rating is done at the end of the entire process by both parties.

User Interface is a large importance to any mobile app. Uber's main UI can be seen below:

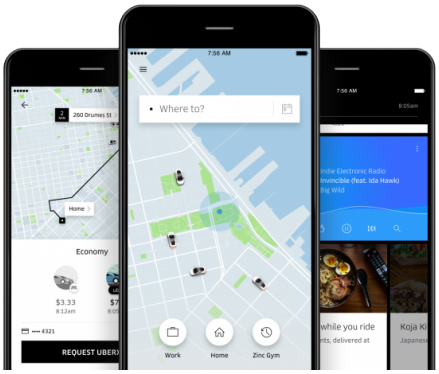


Figure 1.3: Uber User Interface

Uber's focus is on providing as much relevant information to user with the least amount of navigation. It can also be seen how the details are shown amongst the backdrop of the main screen thus making the experience feel connected to the main service. I could implement a similar design perhaps by creating a main screen with a book scanner or cards detailing rented books.

1.4.2 Amazon

Amazons android application has a useful bar code scanner[2] along with other features that I can study. With Amazon's success I can have confidence that these features are effective and beneficial to the user.

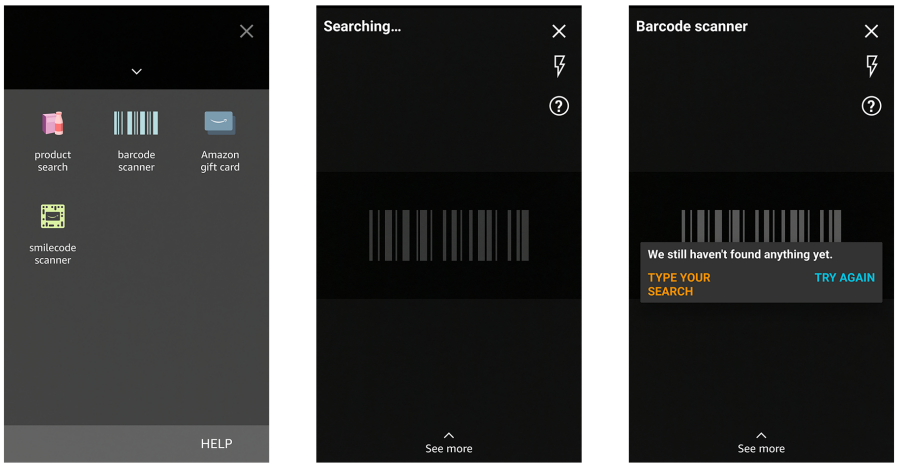


Figure 1.4: Amazon ISBN Scanner

There are 3 stages to this ISBN scanner:

1. Option selection
    - Multiple options are presented to the user before the ISBN scanner is actually opened. These include: *product search, bar code scanner, gift-card and smile-code*; the last two options are proprietary features. Product Search can be adjusted, to my use, as manual input. This is an important feature as it gives the user freedom to choose what to do rather than following a discrete path.
2. ISBN scanner

- The actual scanner is a full-screen camera-view with various options on the side and a bar code watermark in the center. These seem like particularly useful features; this means that I should also implement a flash-light toggle and a watermark. This will be useful for lowlight situations and directing the user where to place the bar code.

3. Fail safe system

- Amazon has implemented an additional pop-up boxes that gives the user two branches of decisions: to carry on scanning or manual input. This is useful as it intuitively gives the user an alternative option, after a set time or number of failed attempts.

Figure 1.5: Amazon Search

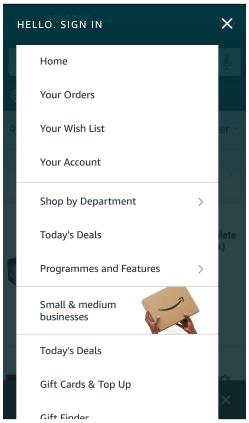


Figure 1.6: Amazon Navigation

The Amazon app also allows user to search for products. This is a feature that I will most likely implement to allow users to search for books that can be rented; thus the way in which this is displayed to the user is particularly important. As can be seen, books are ordered in boxes/containers; which holds details such as Title, Author, Rating and Format. These can be filtered accordingly. There is also a large search bar along the top of the screen. It is important to note that this is a full-screen independent page but still allows for interaction with other pages, as can been seen by the navigation icon in the top left. This navigation bar presents a simple set of options that can be clicked to navigate between pages. The navigation bar is present thought all pages and thus allows the user to easily move between pages easily.

1.4.3 Cloud Services

My app will have be dependent on a database stored in the cloud. Whilst researching various options, all of which are widely used in most android application, I have been able to narrow my options down to 3 provides:

1. Firebase - hosted by Google
2. AWS mobile - hosted by Amazon
3. Azure mobile - hosted by Microsoft

The option that seems the most beneficial is Firebase because it is developed by Google and thus has immediate support with android applications. Furthermore it offers many other features other than a cloud database: authentication, file storage and a Machine-Learning Kit whilst also being free to use. AWS mobile and Azure mobile on the other hand has less to offer and also enforces a strict pricing strategy. Firebase is an open-source project thus there are several example applications available on Github. Firebase documentation[3] details two database options: Firestore and a Realtime Database. Both seem to provide the same persistent organised structure however the underlying difference is that Firestore adopts a NoSQL, document-oriented approach.

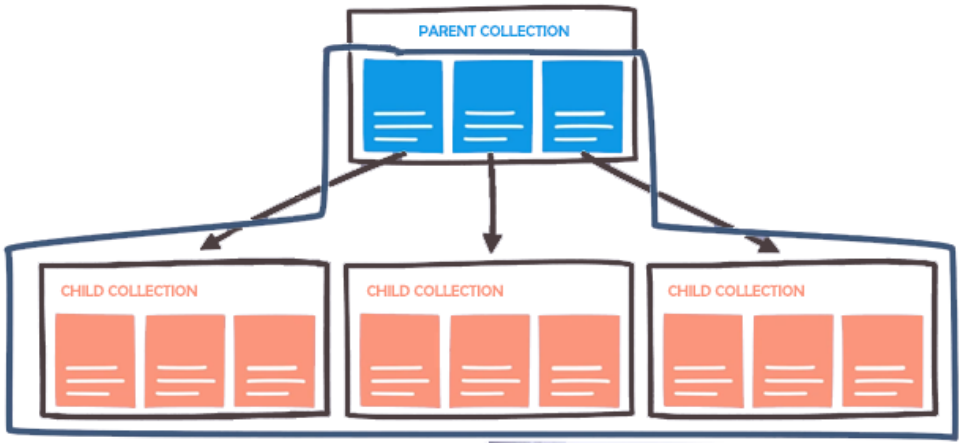


Figure 1.7: Firesore Data Model

As can be seen above, this approach means that rather than traditional tables, data is held in documents, each document containing attributes with an overall maximum size limit of 1MB. Documents are then stored within collections, there can exist several independent collections or collections that are children of others. This quickly resembles a file-directory structure. In the case of storing data for an individual then for each book and transaction associated to them, Firestore seems to provide a suitable and efficient structure.

1.5 Stakeholder Research

To build upon the research that I have conducted, I plan on distributing questionnaires amongst my stakeholders. This will allow me to gain a deeper understanding of their needs and features that they want to be implemented. Based off of the feedback from each stakeholder, I will then conduct interviews to go into more detail about which ideas I can refine. After analysing the results from these interviews, I will create mock-up graphic designs to present; this will give me an idea if I have interpreted their opinions correctly. By following these three steps I should be able to consult wider and more discrete areas of my idea and form a concrete understanding of the features and design elements that I should implement.

1.5.1 Questionnaire

The questionnaire I created can be seen on the following pages...

Palvinder Sander - Mobile Book Sharing Application Questionnaire

I am developing an android application that is principally a platform for users to share books, within the school community. I plan implementing features that allows users to: create an account, add their books by scanning them or otherwise, search for and request a book to rent, message users to plan for the exchange and rate others based on their quality.

I would like to have your thoughts and opinions on this system and thus created this questionnaire. It should take no more than five minutes to complete. Thank You.

1. **How would you like to sign up with the application? (Pick One)**
   - School Username
   - Email
   - Personal Username

2. **Which system would you prefer to add books? (Pick One)**
   - ISBN scanner
   - Manual Input

3. **Do you own an android device? If Yes, does it have a camera? (Pick One)**
   - Yes
     - Yes
     - No
   - No

4. **How would you like to search for a book? (Pick One or More)**
   - Title
   - ISBN
   - Author
   - Genre

5. **Do you have confidence in trusting others to provide quality books? If No, why? (Pick One)**
   - Yes
   - No

……
……
……
……
……..

6. **Do you believe a user rating system can enforce the previous question? (Pick One)**
    - Yes
    - No

7. H**ow often would you need books? (Pick One)**
    - Less than 1 week
    - 1 week
    - 2 weeks
    - Over 2 weeks

8. **What books would you request? (Pick One or More)**
    - Fictional
    - Non-fictional
    - Educational

9. Any other ideas?

……
……
……
……
……
……
……
……

Results: After giving out the questionnaires and receiving the the following week, I was able to make the following analysis and conclusions.

- Question 1
  - 1 person: personal user name
  - 4 people: email
- Question 2
  - 5 people: ISBN scanner
- Question 3
  - 5 people: yes, yes
- Question 4
  - 3 people: genre
  - 4 people: author
  - 5 people: IBSN
  - 5 people: title
- Question 5
  - 4 people: yes
  - 1 person: no - some people could be late to return books, have the wrong book or just not respond
- Question 6
  - 2 people: no
  - 3 people: yes
- Question 7
  - 1 person: more than 2 weeks
  - 1 person: 2 weeks
  - 2 people: 1 week
  - 1 person: less than 1 week
- Question 8
  - 3 people: educational
  - 2 people: fictional

From the results I was able to see the wide array of uses that people may have for my app, for example the varying time period of renting and genres of books. An interesting prospect was out forwards in *Question 9* by *Lewis Jones*; how the app will function without internet connectivity. When I collectively interview the Upper Year stakeholders, I will delve deeper into this feature. I also found that, as i presumed, the Lower Year stakeholder would rather rent non-fictional books and Upper Year stakeholders, academic books. This means that I will have to create a robust system for both parties. From this data I have created a set of interview specific questions that I will be asking each stakeholder set.

1.5.2 Upper Year Feedback

I collectively interviewed *Lewis Jones*, *Shashi Balla* and *Jeevon Grewal* with the following questions and responses:

1. When and what would you use the app primarily for?
    - Lewis Jones *I would probably use the app mainly during and months leading up to exam season.*
    - Shashi Balla *Perhaps over the year-12 summer, reading books is a good thing to put on UCAS so most people buy 2 or 3 books to read in June/July.*
    - Jeevon Grewal *Studying Philosophy means that I have to quite a lot of reading around the subject, its annoying to buy books all the time so I would probably use the app for that purpose. I would also use around Christmas time or half-terms because I have free time them to read.*
2. Do you think a user rating system would be useful?
    - Shashi Balla *I think it is a great security protocol but if someone really wanted to mess around they still easily could.*
    - Lewis Jones *I can see people just giving 5 stars to themselves if they have a bad score; you need to prevent that from happening.*
    - Jeevon Grewal *I think it is a feature that definitely needs to be in there, I want to be confident that the person giving me the book is reliable. How would you start the user off though? Will the user start with 5 stars? I think that everyone that just joined should start with 3.5 stars just so people know that they have not actually done any exchanges yet.*
3. What extra features would you like to see?
    - Lewis Jones *As I mentioned in my questionnaire, I think you should be able to use the app without internet so that you can see the books that you have scanned or books you are renting. That may make some internet dependent features redundant, I think that will need to be taken into account. For most applications I have used their solution to this is natural I have never really even thought about how it works or what to do.*
    - Jeevon Grewal *The foundations are definitely there, its all about refining the UI and making sure it flows nicely.*
    - Shashi Balla *To build upon what Lewis said, I think that the system should allow the user to access the app but also notify them that they have no internet and then allow the user to access blocked features when the phone is reconnected without having to reload the app.*
4. How do you think navigation should work?
    - Shashi Balla: *With most applications there is usually a bottom navigation drawer or side panel to use the app. If there are a lot of pages then I think a side panel would be better as it can allow for more options but if not a bottom panel should be fine.*
    - Jeevon Grewal: *I would like to see icons that are associated with each main page and also a title bar at the top of each page so the user knows what to do. The title bar should allow for a back arrow to navigate to the previous page, this is really important as it will allow the user to transition between pages quickly.*
    - Lewis Jones: *Although animations between scenes can look nice, it can also slow the app down and thus ruin the experience so not including that should improve performance.*
5. What color scheme should I use?
    - Shashi Balla: *Something that isn't too bright and flashy, consistency between colors is important.*
    - Lewis Jones: *A white and green combination could look nice as long as text can be seen clearly.*

6. What do you think about scanning books?

    - Jeevon Grewal: *It sounds interesting but depends very much on how quick it is and accurate.*
    - Shashi Balla: *I don't want to have to scan a book 3 or 4 times for it to recognize the bar code so reliability is very important. I also think that there should be a fail safe system, one for entering the ISBN and one for entering all the details manually.*

1.5.3 Lower Year Feedback

I interviewed *Joe Millerchip*, with the focus on what books younger years would want.

1. What books would you use?

    - *I think that most people including me would want story books and maybe academic books, I say maybe because our school text books cover a lot of the content. On a side note, I think it is important that textbooks from school are not put on the system, that would be really bad.*

2. Do you feel comfortable with meeting with older years to swap books?

    - *I did not really think of that before but now you bring it up, a little bit. But that is probably inevitable even for people in the same year that don't talk to each other*

3. Joe asks me 'How will you organise meet ups?'

    - My response *I will make it completely secure by adding in a simple interface where you can agree on a time and place within school*
    - Joe to me *I think that would work but you could make it easier by allowing people to put in their form room as user details; I think more people would be comfortable with that.*

1.5.4 Teacher Feedback

I interviewed *Mr Jones*, I wanted to get more details about user security.

1. What is your opinions on exchanges?

    - *Your idea on meeting up in set rooms is great but I think that you have to be careful about certain areas such as the Canteen which are already busy*
    - My response *It was brought to my attention that it would be good to keep it within form rooms as a first choice*
    - Mr Jones to me *I think that is certainly a good idea and will save staff a lot of potential hassle*

2. What other policies do you think I should have?

    - *Although not directly related to policies, making sure that the layout is good is important as if it was complex it could lead to errors which could then lead to arguments about lost or not returned books and what not. Also make sure that app is not used during school lessons hours; you could set the time of exchange at lunch time and break time.*

1.5.5 Feedback Analysis

The three interviews I conducted where extremely useful as I gained a better understanding of what actual potential users want from my product. I was also made aware of features and security policies that are valid and will be very important for the applications reliable use. These features included:

1. Long term renting periods - The renting periods should span over a time of 1 week to 4 weeks (a month). This should allow sufficient time for my stakeholders and their representative demographics to use the books sufficiently. Any longer than 4 weeks may also risk users forgetting about the books.
2. New user score - A user should be initialised with a rating of 3.5/5.
3. Self transaction prevention - Users should not be able to request their own books.
4. Easy to use UI - Interface should include a universal toolbar that allows back navigation to the home page.
5. Data security - Data should be handled securely and responsibly.
6. Form Room user details - The app should request users details on sign up to be used for others to evaluate renting a book.
7. Network management - The application should take into account whether internet is available.
8. Break and Lunch transaction periods - There should be 2 time periods for an exchange, lunch and break time. These times are 11:00 - 11:30 and 1:00 to 2:00.

1.6 Application Features

1.6.1 Cloud Application

All services should be supported by cloud storage. This means that all data should be stored in a database on the cloud. This should include authentication, user and book data. This should allow for users to log in from any device and will therefore support for cross-device communication. Cloud support is vital for all services of the application.

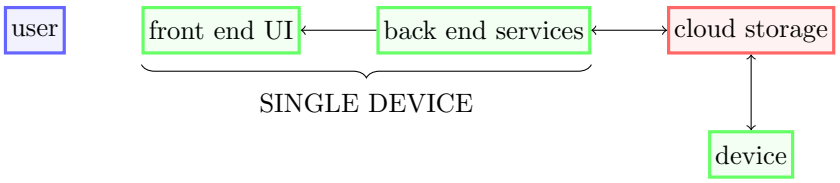


Figure 1.8: cloud services

1.6.2 Account Creation and Log In

My application should allow users to create an account, taking email and password as log-in parameters. It should also request details such as name, date-of-birth, form room and year group. This data should then appropriately stored in the cloud. The app should ideally allow for password changes however this is not an entirely required feature. The user should be presented with a link to a user policy agreement outlining that my application stores data, that they have input about themselves. Users should remain logged even when the app is closed. In order to log out the user must go into a settings page and click a log out button. Once this occurs the app should take the user to the login page. By default the user should also have a user rating score of 3.5.

1.6.3 Adding Books

An interface that allows users to both scan the ISBN of books and accurately fetch meta data along with a fail-safe manual input option should be incorporated. The ISBN scanner should be able to scan all ISBN-10 and ISBN-13 bar codes and return the correct value (e.g.. a bar code may return *0552172359*). From this an API should be used to fetch meta data on the book; this data should then be shown to the user so they can see that the correct book has been found. The meta data returned should include Author, Title and Genre details; an image of the book would also be useful but is not necessaries. Alternatively the user should be given the option of inputting Title, Author, Genre and ISBN. These inputs should be checked if they are valid. These input should then be used as attributes for a book class or data-structure to be stored in the cloud.

1.6.4 Searching Books

Users should be allowed to enter a Title, Author, Genre or ISBN parameters into a search function. These inputs should then be used to traverse the cloud storage in order to find the correct books. The data on the books found, if any, should be returned and displayed in a card format; as can be seen below:

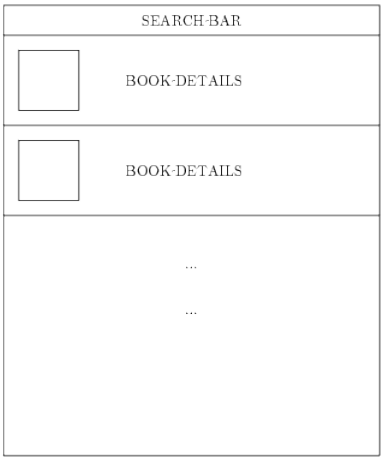


Figure 1.9: Search User Interface

As part of *BOOK DETAILS* should be all of it's meta data along with associated user data such as name, form and user rating. Clicking on a book should reveal a new interface that allows for renting procedures.

1.6.5 Requesting Books

As discussed in *Searching Books*, upon clicking a book the user should be presented with renting options along with all the books user data. The user should be able to enter a time and area for an exchange along with the renting period and submit the request to the other user. The time for the exchange should be limited to lunch and break times. The option for a book exchange location should be selected from a list containing either the requester's form room or the renter's form room. Furthermore, the user should not be able to submit a request of their own book or request a book given they have not met the requirement to rent multiple books (*discussed in User Rating*); an error should be shown appropriately. Once the user has requested the book, the other should be notified. This request should be shown in a *loaned books* page whereby the user can select the request and accept or decline. Whilst the request is in process, the book should be taken off the market and marked as pending. Once the period of renting has expired the book should be removed from the users book list; this will mean that the user has to put it on the market again under their own accord instead of it accidentally being automatically added. Whilst the book is on the market, the option to take it off should be easily accessible.

1.6.6 User Rating

Upon confirmation that the rented book has been returned the user should be able to rate the user on a scale of 1 to 5. The mean of the users total scores and number of scores should be displayed as their current score. If a user has a rating less than 1 they should be limited to one book at any one time, as a penalty.

1.6.7 Limitations

I have decided to develop my application for Android devices, this will mean that I cannot release it for iPhones or otherwise. This is because although Android devices run a Linux kernel, written in C, Java is the supported language for application development. This is partially due to JVM (Java Virtual Machine), where high-level code is compiled into byte code, that means code can be run on as many devices that support the JVM, without having to change the code. This is particularly useful for android devices where there are thousands of brands and hardware sets. On the other hand, iPhone devices have a more proprietary outlook to it's operating system, where applications can only be developed by an in-house language, Swift. Despite my app potentially not reaching many people, I still believe that developing it for Android will have a significant impact; despite Apple's mainstream popularity the majority of people still use android devices.

1.7 Hardware and Software Prerequisites

Any android device, ideally with a camera, network connection would be suitable. The minimum operating system required for this system will be Android Lollipop 5.0. I have chosen this version of Android because it should support all the packages and google service APIs I intend on using furthermore this will approximately support 85% of all devices.

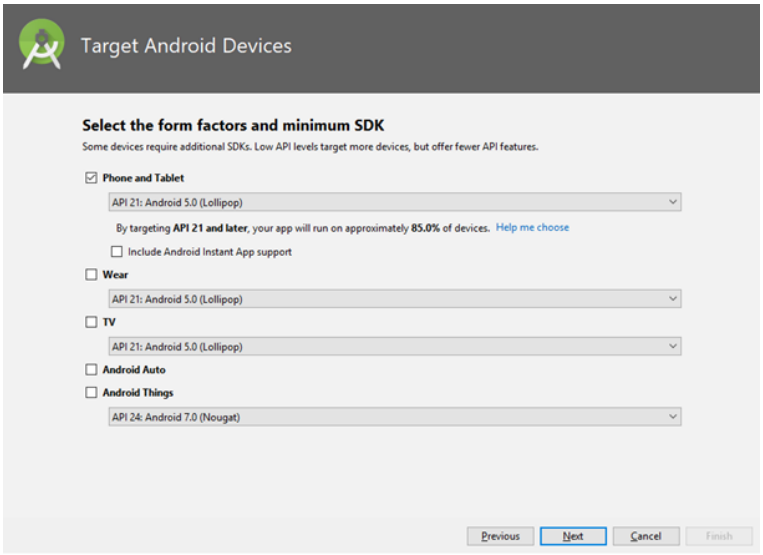


Figure 1.10: Android Studio SDK selection screen

1.8 Success Criteria

Table Of Success Criteria	
Criteria	Justification
Completed Project - fully functioning cloud-based application that allows users to exchange books	The stakeholders must be able to use the app effectively in order to solve the initial problem.
Account Management - Create an account, log in and out capability on multiple devices, user data storage (date of birth, year group, form room)	The end user should be able to log in from any where and access their data, this is important for identification purposes and for other services such as book exchanges when each user's form room is required. This should also allow for greater dexterity of use for the user.
Book Input - ISBN scanner, ISBN manual input, meta data source, complete data manual input	This data is required for the basic functioning of the app; users need to be able to see which books they are interacting with. This should also allow the user to easily add data to their account without the need for long input processes, although this is still available, reducing human error.
Book Management - Alter book data	Allows the user to delete books in the case that they may not want that data to be held any longer. Books should be able transition between states in order to signify changes in it's life cycle e.g. scanned, on the market or rented.
Book Search - Criteria for searching, ranked results	User can find options that are appropriate for what they want or if they are unsure they can find somewhat relevant books which may still be helpful. Ranked results will allow users to find the best quality providers.
Book Request Management - time place time input, accept decline dialogue, clear deadline details	Provides a clear interface for users to formally request a book based on clear parameter. This data is then relayed to the other user thus making it an instantaneous process. Deadline enforcement will ensure users are aware of when to return the book, reducing needless complications.
User Rating - averaged over time on a 1-5 star basis	Ensure users can rate each other based on their reliability and quality of service. Ultimately forms confidence that books can be requested, exchanged and returned with honesty and without problems. Measure in which appropriate penalties can be applied to poorly behaving users.
Error Free	It would not be acceptable to present a program containing glitches or bugs, especially any logic errors which can mean the application runs but not as expected.
Graphics Styling	The client should be impressed by the material theme but it should not be a processor intensive task.
Navigation Layout - bottom navigation, back navigation	Client should be able to feel a flow in the app whereby they can open activities and return to previous ones quickly. Should also prevent users from navigating to expired activities such as to the log in screen once logged in.
Suitable Complexity	The app must live up to the standards of an A-Level project for .g. database management, file handling, network tasks, multi-threaded operations , searching and sorting algorithms etc.

Chapter 2

Design

2.1 Plan Overview

To design this application, the problem needs to be decomposed into its various components and corresponding digital structure. These digital structures are confined to the de facto standards that must be met when developing an android application; from programming paradigm restraints to class structures. This stage of design must also be consulted with my stakeholders before development begins, this is because any features that I may have overlooked or designed incorrectly can be highlighted. A considerable part of design is functional decomposition, breaking a problem into smaller sub-problems. This will in turn aid me with deciding on a development time-line, based around how I will split up iterations according to the Rapid Application Development life cycle.

2.1.1 Procedural vs Object-Orientated

Programs are made up of modules, which are parts of a program that can be coded and tested separately and then assembled to form a complete program. In procedural languages, these modules are procedures, where a procedure is a sequence of statements. In object-oriented programming, the main modules in a program are classes, rather than procedures. This approach lets you create classes and objects that model real-world objects. Furthermore object-orientated programming is much better at security and abstraction, in terms of encapsulation of attributes and methods. Procedural programs on the other had can be much simpler to manage and code. In my case of development, I have to use Java, and object-orientated programming language, due to it's native support for android development. This however comes had in hand with the advantages of encapsulation, previously discussed. This is because each class can extend (concept of inheritance in Java) from a parent activity class. This means that the underlying mechanics of the android kernel, activity life-cycle management and threading scheme is appropriately abstracted away.

2.1.2 Stakeholder Feedback Plan

My five stakeholders will play an instrumental role in determining the quality of my product and whether it fits the requirements indicated in the analysis stage. I will be meeting with my stakeholders at the end of the design stage to verify my work and ask if anything other ideas or issues have arisen since the last time we met. This should bring me in good stead to continue onto the development stage. I think that it will also provide me with a good opportunity to showcase any developments on my part that I think should be discussed before I continue with development. I will show the users any function decomposition trees and user interface designs I create in this process of design. These are diagrams that should give my stakeholders a good idea off the vision I have for the application whilst also no overcomplicating things between us,i.e showing pseudocode could confuse some of my stakeholders.

2.1.3 Android Development Structure

Each page[1] in an android application consists of two elements, an XML file and a child class that extends an Appcompat parent. The XML file is used to declare the user-interface components and their corresponding identification tags. The corresponding class is used to interact with the activity, through override methods, as defined by an activity life-cycle.

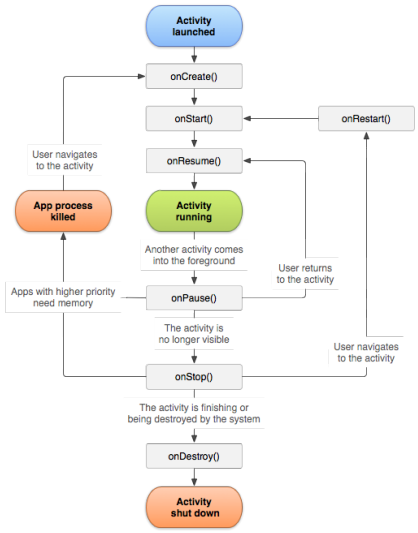


Figure 2.1: A simplified illustration of the activity life cycle.

2.2 Functional Decomposition

Functional Decomposition is the process of breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain; this process can be represented by a structure diagram. This is not only a computational method to further development but also a means of consolidating previous knowledge and ensure that all areas have been looked at and analysed. This is especially important for my application as the user will most likely interact at some point with every interface that I create. For this reason it is vital that I plan a robust system.

2.2.1 Structure Diagram

I first set about abstracting away ideas from reality; leaving me with the essential 2 objects that could later be represented as classes, a user and a book. The attributes that are associated with these two objects are most important at this stage, as they will play a large part in designing the collection of this data. For the user this should include: email, password, name, date of birth, form room, rating, scanned/borrowed/loaned books. For a book this should include: authors, genre, image, ISBN, owner and title. These are abstractions as they do not represent every detail of their real life entities, only the data that is relevant to end user and processes that occur within the app. Using the computational method of thinking ahead, I was able to declare how inputs, outputs and other processes can be separated into different stages; such as user account creation, displaying user profiles to fetching data and storing it behind the scenes of the user-interface. Thinking logically involved me managing user-interface interactivity, such as with navigation components where a switch case selection statement may be required to change between activities. This process also extends across to judging where I can apply concurrent processing, such as when fetching data, using the internet, whereby the main ui thread should not be blocked. Using these processes I have been able to break down the overall problems into layers of first abstraction then specific details of interactivity, in chronological order.

[1] A page or scene within an android application is formally known as an Activity

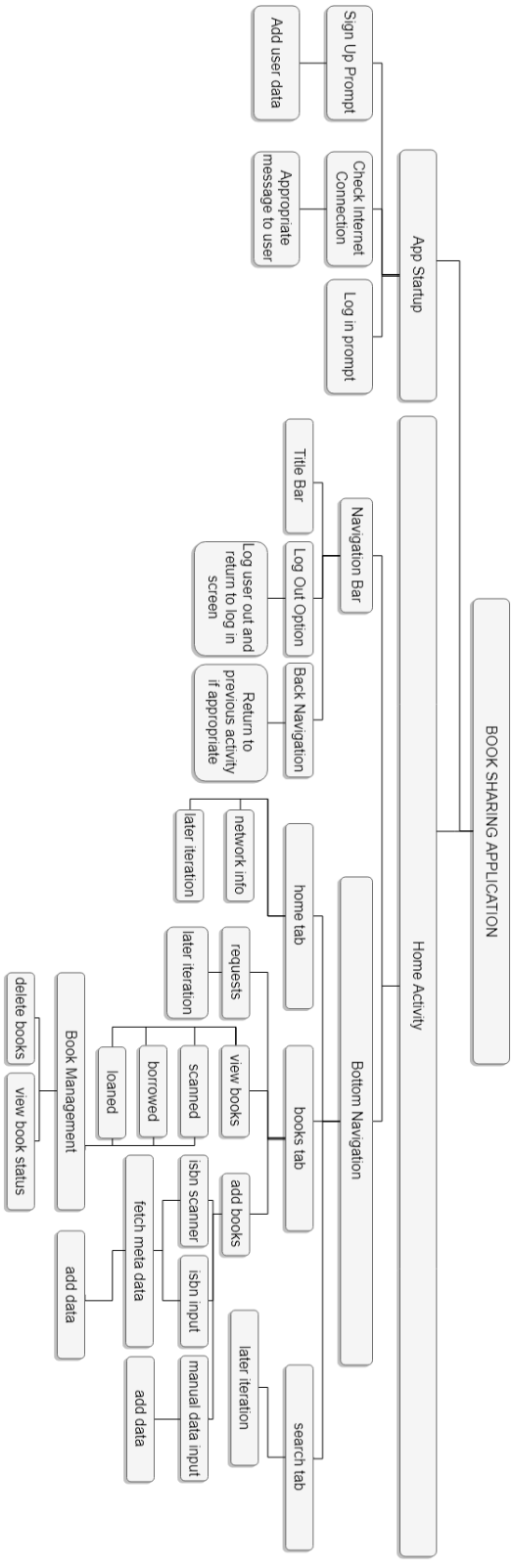


Figure 2.2: Application Decomposition

2.2.2 Iterative Development

Following the Rapid Application Development methodology means that I will be creating multiple prototypes of my app, each iteration fixing previous problems that may arise and adding new features. As part of my first iteration I will be following the first 2 branches of my structure diagram, these tasks will include:

1. Managing network connection
2. Account Creation
3. Log In
4. Adding user data
5. Navigation Bar
6. Bottom Navigation
7. Books Tab
8. Adding books
9. Scanning books or otherwise
10. Fetching ISBN data

I plan on creating further iterations as part of creating the final product. The features that I will develop there will consist of:

1. Viewing Books
2. Book Management
3. Book Searching
4. Book Requesting
5. Any other unforeseen changes in requirements

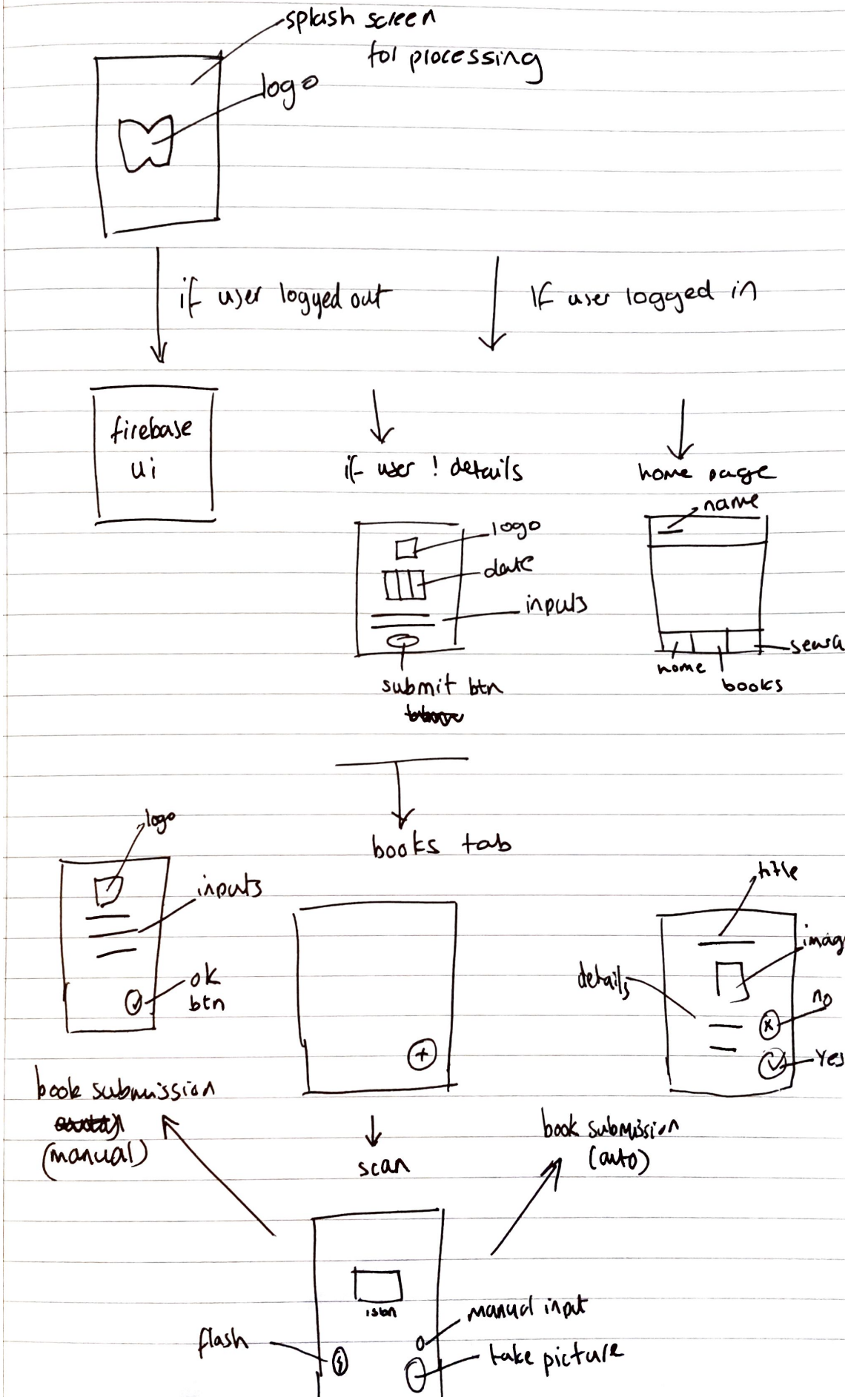
The list for this prototype may seem to be a lot more extensive than what I have to do as part of other iterations. This is because there is a plentiful amount of foundation work to be done first, i.e building a layout that can be built upon with more features and setting up a basic way of creating book object etc.

2.3 Testing

For all prototypes I will include deep testing in order to ensure that the code I am producing is stable and performs the correct tasks. This is important as going forward, one error could propagate throughout the app. In order to display my tests I will use the console to output data, use screen shots of the app in action, screen shots of debugging tools and video recordings of the app. Videos will be provided on a CD, all videos will be named. If I later state video evidence is used, I will add it's file name in the footer of that page. The videos will be stored on the CD in the directory recordings/tests.

2.4 Usability

I have designed mock ups of the user-interface and how each component links to each other. This should help me to create the layouts in the development stages but also consult my stakeholders to see how it fairs with their expectations.


splash screen
for processing
logo
if user logged out
If user logged in
firebase ui
if user ! details
home page
name
logo
date
inputs
submit btn
search
home
books
books tab
logo
inputs
ok btn
book submission (manual)
title
image
details
no
Yes
scan
book submission (auto)
vision
manual input
flash
take picture

2.5 Development Prerequisites

2.5.1 Google Books API

An important stage in the app, that the end-user will most probably use, is the function of fetching meta data for a scanned book. A You Tube video titled *Google Books API Example - Book Search Application* drew my attention to the use of the Google Books Repository[2]. At first the video seemed to be useless because the individual was explaining how to develop a web application in JavaScript. However, the individual briefly explained that there is a specific link that can be used to fetch data; this link being *https://www.googleapis.com/books/v1/volumes?q=*. The end of the link has the query *?q=*, here an ISBN should be appended. After further investigation, I accessed the web domain using the following ISBN, from a Harry Potter and the Deathly Hallows, 9780545010221. The website produced the following results:


```
{
 "kind": "books#volumes",
 "totalItems": 5661,
 "items": [
  {
   "kind": "books#volume",
   "id": "GZAoAQAAIAAJ",
   "etag": "u98UmxjSESM",
   "selfLink": "https://www.googleapis.com/books/v1/volumes/GZAoAQAAIAAJ",
   "volumeInfo": {
    "title": "Harry Potter and the Deathly Hallows",
    "authors": [
     "J. K. Rowling"
    ],
    "publisher": "Arthur a Levine",
    "publishedDate": "2007",
    "description": "The magnificent final book in J. K. Rowling's seven-part saga comes to readers July 21, 2007. You'll find out July 21!",
    "industryIdentifiers": [
     {
      "type": "OTHER",
      "identifier": "UCSC:32106019703807"
     }
    ],
    "readingModes": {
     "text": false,
     "image": false
    },
    "pageCount": 759,
    "printType": "BOOK",
    "categories": [
     "Juvenile Fiction"
    ],
    "averageRating": 4.5,
    "ratingsCount": 3446,
    "maturityRating": "NOT_MATURE",
    "allowAnonLogging": false,
    "contentVersion": "1.0.1.0.preview.0",
    "imageLinks": {
     "smallThumbnail": "http://books.google.com/books/content?id=GZAoAQAAIAAJ&printsec=frontcover&img=1&zoom=5&source=gbs_api",
     "thumbnail": "http://books.google.com/books/content?id=GZAoAQAAIAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api"
    },
    "language": "en",
    "previewLink": "http://books.google.co.uk/books?id=GZAoAQAAIAAJ&q=9780545010221&dq=9780545010221&hl=&cd=1&source=gbs_api",
    "infoLink": "http://books.google.co.uk/books?id=GZAoAQAAIAAJ&dq=9780545010221&hl=&source=gbs_api",
    "canonicalVolumeLink": "https://books.google.com/books/about/Harry_Potter_and_the_Deathly_Hallows.html?hl=&id=GZAoAQAAIAAJ"
   },
   "saleInfo": {
    "country": "GB",
    "saleability": "NOT_FOR_SALE",
    "isEbook": false
   },
   "accessInfo": {
    "country": "GB",
    "viewability": "NO_PAGES"
```


Figure 2.3: Google Books API example

The data at first glance seemed to be correct however after the first correct result, other different books with seemingly similar ISBN values were appearing; this can be taken into account when parsing this data. As can be seen the data seems to be in the format of a dictionary embedded with the arrays etc.

[2]Database of books hosted and maintained by Google

Entering this data into a JSON parsing website[3] produced much clearer results:


```
{
    "kind":"books#volumes",
    "totalItems":5661,
    "items":[
        {
            "kind":"books#volume",
            "id":"GZAoAQAAIAAJ",
            "etag":"u98UmxjSESM",
            "selfLink":"https://www.googleapis.com/books/v1/volumes/GZAoAQAAIAAJ",
            "volumeInfo":{
                "title":"Harry Potter and the Deathly Hallows",
                "authors":[
                    "J. K. Rowling"
                ],
                "publisher":"Arthur a Levine",
                "publishedDate":"2007",
                "description":"The magnificent final book in J. K. Rowling's seven-part saga comes to readers July 21, 2007
                You'll find out July 21!",
                "industryIdentifiers":[1],
                "readingModes":{
                    "text":false,
                    "image":false
                },
                "pageCount":759,
                "printType":"BOOK",
                "categories":[1],
                "averageRating":4.5,
                "ratingsCount":3446,
                "maturityRating":"NOT_MATURE",
                "allowAnonLogging":false,
                "contentVersion":"1.0.1.0.preview.0",
                "imageLinks":{
                    "smallThumbnail":"http://books.google.com/books/content?
                    id=GZAoAQAAIAAJ&printsec=frontcover&img=1&zoom=5&source=gbs_api",
                    "thumbnail":"http://books.google.com/books/content?
                    id=GZAoAQAAIAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api"
                },
                "language":"en",
                "previewLink":"http://books.google.co.uk/books?
                id=GZAoAQAAIAAJ&q=9780545010221&dq=9780545010221&hl=&cd=1&source=gbs_api",
                "infoLink":"http://books.google.co.uk/books?id=GZAoAQAAIAAJ&dq=9780545010221&hl=&source=gbs_api",
                "canonicalVolumeLink":"https://books.google.com/books/about/Harry_Potter_and_the_Deathly_Hallows.html?
                hl=&id=GZAoAQAAIAAJ"
            },
            "saleInfo":{…},
            "accessInfo":{…},
            "searchInfo":{…}
        },
        {…},
```


Figure 2.4: Google Books API example parsed

The data to fetched, as decided in the analysis stage, is the title, genre, image and author. All relevant data is provided in the first item in the array *items*. The data title, and author are both stored within the sub-array *volumeInfo*. The title is simply a string with the associated token of *title*, the author is contained within the array *Authors*, this is because a book could potentially have multiple authors. Genre is also stored in the array *categories*, a child of the *items* array. The image thumbnail is stored within the the dictionary *imageLinks*; there are two values in this dictionary, *smallThumbnail* and *Thumbnail*. This is presumably done so that there is a lower resolution image available to minimize network usage for API users; the *smallThumbnail* is therefore may be better for my use. This is because network usage can be both taxing on the battery, processor and the users phone bill if they are not connected to WI-FI. However an image of higher resolution may provide the user with a better method of identifying their book. Furthermore the difference in thumbnail is often minimal or not at all. For example with the isbn *1782944141* both the small and larger thumbnail have a resolution of *128*x*182* and a file size of *5KB*. For this reason

[3]Format of the data resembled JSON an open-standard file format that uses human-readable text to transmit data objects consisting of attribute value pairs and array data types.

I will be using the larger thumbnail. With this task involving accessing data from a url, it will involve some type of asynchronous implementation in order to prevent suspension of the main user interface. Asynchronous programming is a means of parallel programming in which a unit of work runs separately from the main application thread and notifies the calling thread of its completion, failure or progress. I expect that this will involve instantiating an object that inherits from an asynchronous task and from there overrides certain parent methods. I gathered this from research I did using stack-overflow[4]. They represent the use of a class that looks like:

```
1  public class className extends AsyncTask<String, Void, [DATA]>{
2      @Override
3      protected [DATA] doInBackground(String... extraData) {
4          //call a method to execute
5          return [DATA];
6      }
7
8      @Override
9      protected void onProgressUpdate(Void... values) {
10         super.onProgressUpdate(values);
11     }
12
13     @Override
14     protected void onPostExecute([DATA] DATA){
15         //do stuff
16     }
17 }
18
19 ...
20
21 new className().execute(DATA);
```

As can be seen the class can be instantiated then run using the method *.execute()*. I think that I should be able to use the construct in order to fetch data in the *doInBackground* method and then implement the results in the *onPostExecute* method.

2.5.2 Firebase Authentication

As previously discussed Firebase is an open source library that offers several feature from cloud storage to database. Firebase also includes an authentication library that provides back end services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to an app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more. My stakeholders informed me that sign-up should include the use of an email thus other forms of authentication can be ignored. A feature that is focused on in documentation is the FirebaseUI solution. It provides a drop-in auth solution that handles the UI flows for signing users with email addresses and passwords. The UI component also handles edge cases like account recovery and account linking that can be security sensitive and error-prone to handle correctly. Behind the scenes, when a user provides details or an OAuth token[4] it is passed onto the Authentication SDK; here back end services verify the credentials and return a response to the client. Implementing this functionality using Firebase is beneficial in abstracting away lengthy, time consuming processes of error handling, which if not enforced correctly can lead to further errors throughout the application, such as redundant users accessing data.

[4]Signature created from federated identity providers

2.5.3 Firestore

Cloud Firestore is a flexible, scalable database for mobile development; it also keeps data in sync across all clients through real time listeners and offers offline support. I will be using Firestore to hold 3 collections of documents[5]: users, books, transactions.

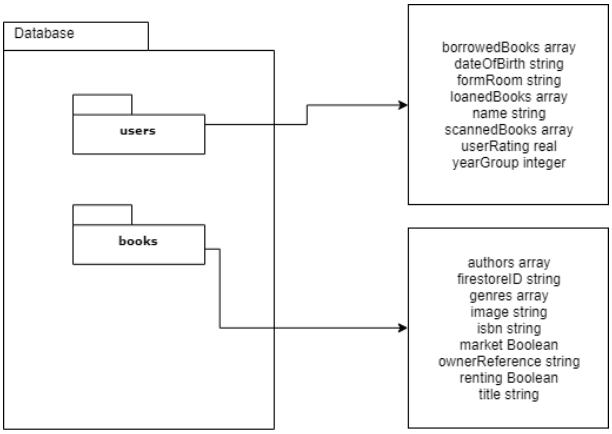


Figure 2.5: Firestore Data Structure

When adding data, it must be in the form of a hash-map or a defined class. A hash-map is a structure that is able to map certain keys to assigned values. The keys and values can be any primitive type[6] or defined object. Hash-maps are used because they are much faster for retrieving than arrays and linked lists. A sorted array in an average case scenario would be able to find a particular value in $O(\log n)$ with a binary search. On the other hand a hash-map is able to check if it contains a value in $O(1)$. $O(1)$ being a better time complexity than $O(\log n)$ thus means a hash-map is faster. On the other hand I could create a class for each collection, which can be instantiated using given data and then added to the cloud. Using a class means that I can declare given attributes and create a constructor that ensure that they will all be given a value. Using this construct, I should be able to remove potential errors in adding data with incorrect field names or missing fields out completely. Furthermore I will be able to easily add methods that could be used to process this data in later stages. After doing some research into the syntax of declaring a hash-map in Java, it is obvious that the process is lengthy and generates a lot of boilerplate code, this can be abstracted away by simply instantiating a class with one line rather than several. For this reason I will be creating classes for each of my collections and using Firestore's ability to interpret the attributes stored by them to add data to the cloud. Firestore also enforces its cousin authentication library by implementing access rules. I will be using the access rule that allows for read and write access to data if the user is able to provide an authentication token, i.e. they are logged in. I have created a structure diagram to illustrate how collections and documents will be organised and the data to be stored within them. Cloud Firestore also supports offline data persistence. This feature caches a copy of the Cloud Firestore data that my app is actively using, so my app can access the data when the device is offline. I can write, read, listen to, and query the cached data. When the device comes back online, Cloud Firestore synchronizes any local changes made by my app to the data stored remotely in Cloud Firestore. It should be noted that all the data added can be viewed from an admin perspective on the online web app firebase-console.

[5]Firestore structure discussed in analysis
[6]Primitive types are the most basic data types available within Java

2.5.4 Firebase Machine Learning Kit

ML Kit is a mobile SDK that brings Google's machine learning expertise to Android in a powerful yet easy-to-use package. ML Kit comes with a set of ready-to-use APIs for common mobile use cases, including barcode scanning. ML Kit's selection of APIs run on-device or in the cloud. On-device APIs can process your data quickly and work even when there's no network connection. Firebase ML Kit has four key features for my use:

1. Reads most standard formats, including EAN-8 and EAN-13
2. Extracts structured data using one of the supported formats, automatically parsing the image
3. Barcodes are recognized and scanned regardless of their orientation.

Although seemingly complicated, Firebase have abstracted away the complications of dealing with neural networks and such technologies to simply instantiating an image, detector and passing one into the other to get a result.

2.5.5 Otalia-Studios Camera View

Camera View is a well documented, high-level library that makes capturing pictures and videos easy. The alternative would be implementing the Camera2 API[7] myself, this is a tedious process of creating camera manager classes, managing thread usage and the limits of the applications context(making sure threads do not execute when the phone screen is off as this can result in overheating due to excessive camera access). Instead this library, offered by Otalia-Studios, provides an easy to use way of interacting with hardware. Furthermore, interacting with the camera is as easy as instantiating a camera listener to take a picture when a button is clicked:

```
camera.addCameraListener(new CameraListener() {
    @Override
    public void onPictureTaken(byte[] picture) {
        // Create a bitmap or a file...
        // CameraUtils will read orientation for you, in a worker thread.
        CameraUtils.decodeBitmap(picture, ...);
    }
});

camera.capturePicture();
```

I will use the library in the stage of taking a picture to pass into a firebase ml-kit detector to find barcodes. However, I see a problem with the data that Camera View returns and the data ml-kit requires. As can be seen in the code, Camera View returns a byte array. In ml-kit documentation it states that a bitmap object is required for the detector to function. A byte array is used to store a collection of binary data (this data is stored contiguous in memory). At bitmap is an image that is broken into the small units (pixels) and then the color information of each pixel (color depth) is stored in bits that are mapped out in rows and columns. I will have to take this process into account in my solution by implementing a method to convert a byte array into a bitmap.

[7] Official android api to interact with the devices camera

2.5.6 Bottom Navigation

I will be implementing a bottom navigation system to be used to switch between the main pages. This will look something like: Official Android documentation explains that such an interface is

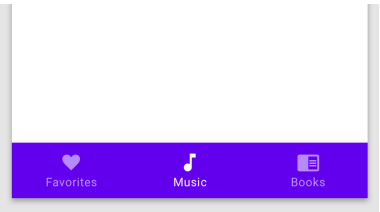


Figure 2.6: Bottom Navigation

implemented using *Fragments*. A fragment is a modular section of an activity, which has its own lifecycle, receives its own input events. Here I will have to declare three fragments for home, books and search. Each of these fragments will then be display using the bottom navigation and its corresponding buttons (within an main activity). This is done by declaring an attribute that is an instantiated BottomNavigationView interface:

```
1  private final BottomNavigationView.OnNavigationItemSelectedListener ATTR
2              = new BottomNavigationView.OnNavigationItemSelectedListener() {
3          @Override
4          public boolean onNavigationItemSelected(@NonNull MenuItem item) {
5              switch (item.getItemId()) {
6                  case R.id.PAGE1:
7                      //show PAGE1 Fragment
8                      return true;
9                  case R.id.PAGE2:
10                     //show PAGE2 Fragment
11                     return true;
12                 case R.id.action_search:
13                     //show PAGE3 Fragment
14                     return true;
15             }
16             return false;
17         }
18     };
```

2.5.7 Tool Bar

The Tool Bar will be instrumental in providing a clear interface to the user, not only will it provide clear information on what page the user is on but also provide back navigation to go back to previous pages and an option to logout. Implementing back navigation comes from editing a manifest file (*.xml* file). The option to logout must be implemented using an override method:

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case BTN1:
                //do stuff
                return true;

            case BTN2:
                //do stuff
                return true

            default:
                return super.onOptionsItemSelected(item);

        }
    }
```

2.5.8 Floating Action Button

As illustrated in my user interface designs, I will be using a button known as a floating action button in order for the user to do specific tasks. After creating my decomposition tree, it was brought to my attention that the availability of a network connection will be integral to the function of my app. In the case that there is not a connection, I must be able to handle these cases so that run-time errors do not arise. For example one of my interfaces will require an internet connection in order to fetch meta data. The way in which I have designed my interface means that the user has to click on a floating action button to get to these activities. Going to such a page, without an internet connection, may result in errors or needless complexities in the user interface. For this reason I need to find a way of preventing the user from viewing certain pages if a connection is not available.

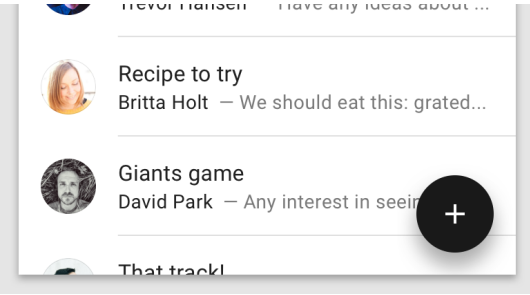


Figure 2.7: Floating Action Button

Android Development documentation provides clear examples of how to implement a floating action button:

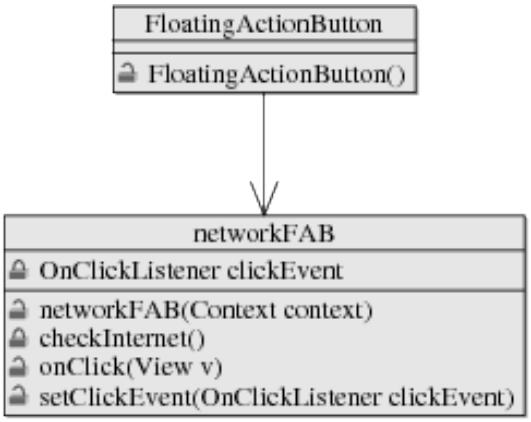
```
FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //do stuff
    }
});
```

A variable that is a casting[8] of a user interface component is used to set an *OnClickListener*. An *OnClickListener* is a class that in Java can be instantiated from an interface. An interface is an abstract type that is used to specify a behaviour that classes must implement. Here the *OnClickListener* must implement the override method *onClick* where any instructions to be executed when the button is clicked is defined. A thing to note is that the scope of this override method is within the activity or fragment class it is defined in. This means that I could potentially create a variable that instantiates the listener instead of directly in the method *setOnClickListener*. Now that I have established how a floating action button works, I can return to the original problem: checking for internet before the button is clicked. My immediate response would be to define a method that checks for an internet connection and then call that in each *onClick* override method. However this is extremely inefficient. I would have to repeat this process each time I want a floating action button with this property. This will result in the same *checkInternet* method being defined across multiple classes. Furthermore, it makes it easy for me to forget to implement this feature in development. Instead I think that I should implement a class that inherits from the *FloatingActionButton* class. I would also have to implement the *onClickListener* interface. As a result I will have to implement three distinct constructors, the *onClick* override method, a *checkInternet* method and another method to call *setOnClickListener*. Doing all of this will result in a class that can be used as a floating action button replacement, however it still will not complete it's intended task as I have not defined what should occur when the button is clicked. This definition will be unique for each button therefore I cannot define it directly in the *onClick* method. Instead, I will have to define an attribute of type *onClickListener*. After I have instantiated the class, I will have to call a setter method to pass a pre-defined *onClickListener* to the object (as shown possible previously). Within the *onClick* method I can then call the *checkInternet* method to check if a connection is available then accordingly call the *onClick* method defined in the *onClickListener* method.

[8]Use of polymorphism in object oriented programming to set one variable type to another

2.6 Algorithm Design

2.6.1 Floating Action Button Class



Algorithm: networkFAB(Context context)

```
1 call parent constructor with arguments;
2 initialise onClick listener;
```

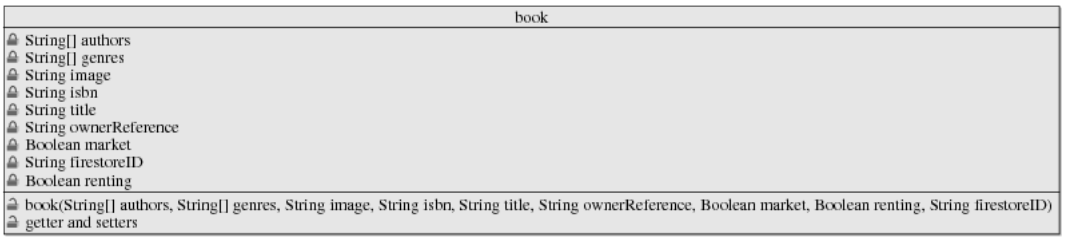
Algorithm: checkInternet()

```
1 if mobile data is connected or wi-fi is connected then
2 |   return true;
3 else
4 |   return false;
5 end
```

Algorithm: onClick(View v)

```
1 if checkInternet() then
2 |   clickEvent.onClick(v);
3 else
4 |   prompt user internet is not available;
5 end
```

2.6.2 Book Class



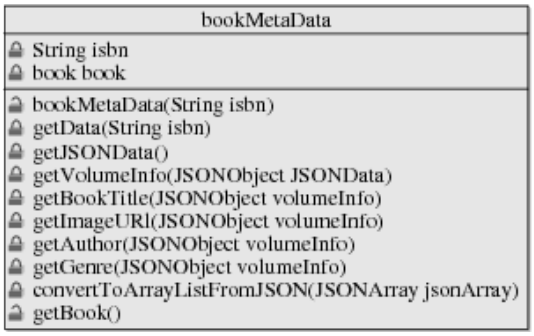
Algorithm: book(List authors, List genres, String image, String isbn, String title, String ownerReference, Boolean market, Boolean renting, String rentingID)

```
1 this.authors = authors;
2 this.genres = genres;
3 this.image = image;
4 this.isbn = isbn;
5 this.title = title;
6 this.ownerReference = ownerReference;
7 this.market = market;
8 this.renting = renting;
9 this.rentingID = rentingID;
```

This class will be serialised when saving it to the firestore database. Serialisation is the process of process of translating data structures or object state into a format that can be stored and then reconstructed later. I have created a table of attributes to indicate the reason for their existence.

Attribute	Justification
authors	Array of all the authors that contributed to the writing of the book. This data can be null if there are no authors.
genres	Array of all the genres that are associated with the book. This data can be null if there are no authors.
image	String of the thumbnail url. This can later be used to fetch the image.
isbn	String of the isbn. Used to identify the book.
title	String of the title. Main title of the book.
ownerReference	Firebase user-id reference of the user that added the book. This can be used to identify the book from as their own. Will also be used to fetch user's data if another individual searches the book.
market	Boolean value to indicate whether the user has put the book in a 'pool' of other books where by others can find it by searching it's details.
firestoreId	String value of the documents id where this class serialised is saved to.
renting	Boolean value as to indicate whether the book is loaned to another person. This can be used to prevent the user from deleting the book or it appearing when others search for it.

2.6.3 Fetch Meta Data Class



Algorithm: bookMetaData(String isbn)

```
1 this.isbn = isbn;
2 this.mapData = getData();
```

Algorithm: getData()

```
1  JSONObject JSONData = getJSONData();
2  JSONObject volumeInfo = getVolumeInfo(JSONData);
3  if volumeInfo == null then
4  |   return null;
5  else
6  |   String title = getBookTitle(volumeInfo);
7  |   String image = getImageURl(volumeInfo);
8  |   List authors = getAuthor(volumeInfo);
9  |   List genres = getGenre(volumeInfo;
10 |   book book = new book(authors, genres, image, isbn, title, null, false, false, null);
11 |   return book;
12 end
```

Algorithm: getJSONData()

```
1  try:
2  |   String urlString = "https://www.googleapis.com/books/v1/volumes?q=" + this.isbn;
3  |   BufferedReader reader = new BufferedReader(get url data);
4  |   String data = "";
5  |   String line = "";
6  |   while line != null do
7  |   |   line = reader.readline();
8  |   |   data += line;
9  |   end
10 |   return new JSONObject(data)
11 catch:
12 |   return null;
13 end
```

I have not discussed the *BufferedReader* object instantiated here previously. This is because it is a Java specific implementation of accessing a website, something that I was made aware by from StackOverflow[5]. The buffered reader is just an object that accesses a website and gets the raw data displayed on the screen and then allows the programmer to access this data line by line. As a result I should be able to get a string of data that is the contents of the website. I should then be able to convert this to a JSONObject by simply instantiating it with the string of data as a parameter.

Algorithm: getVolumeInfo(JSONObject JSONData)

```
try:
    JSONArray items = JSONData.get("items");
    JSONObject firstItem = items.get(0);
    return firstItem.get("volumeInfo");
catch:
    return null;
end
```

Algorithm: getBookTitle(JSONObject volumeInfo)

```
try:
    return volumeInfo.get("title");
catch:
    return null;
end
```

Algorithm: getImageURL(JSONObject volumeInfo)

```
try:
    JSONObject urls = volumeinfo.get("imageLinks");
    return urls.get("smallThumbnail")
catch:
    return null;
end
```

Algorithm: getAuthor(JSONObject volumeInfo)

```
try:
    JSONArray authors volumeInfo.get("authors");
    return convertToArrayFromJSON(authors);
catch:
    return null;
end
```

Algorithm: getGenre(JSONObject volumeInfo)

```
try:
    JSONArray categories volumeInfo.get("categories");
    return convertToArrayFromJSON(categories);
catch:
    return null;
end
```

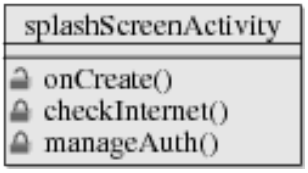
Algorithm: convertToArrayListFromJSON(JSONArray jsonArray)

```
Array data = new Array[jsonArray.length()];
for ( i = 0; i < jsonArray.length(); i++ ) {
    String item = jsonArray[i];
    data = data + item;
}
return data;
```

The method *convertToArrayListFromJSON(...)* may seem like a crude solution to a crude problem. The problem being that the data type of the genre and author arrays must be a primitive,in order for the book class to be correctly serialised and stored in firestore[9]. Thus the JSONArray must be converted to an a standard array. There is no method within the JSONArray class to do this. Thus the reason why this method exists.

[9]Attribute restrictions are detailed within the firestore documentation.

2.6.4 Splash Screen Activity



Algorithm: checkInternet()

```
1  if mobile data is connected or wi-fi is connected then
2  |  return true;
3  else
4  |  return false;
5  end
```

Algorithm: manageAuth()

```
1  FirebaseFirestore initialise-firestore;
2  FirebaseUser current-user = initialise current user;
3  Boolean internet-connection = getInternetConnection();
4  if !internet-connection then
5  |  output no network message to user;
6  if current-user == null then
7  |  if !internetConnection then
8  |  |  go to no network activity;
9  |  else
10 |  |  go to log in activity;
11 |  end
12 else
13 |  DocumentReference document get current-user document;
14 |  String date-of-birth = document.get("dateOfBirth");
15 |  String form-room = document.get("formRoom");
16 |  String year-group = document.get("yearGroup");
17 |  ArrayList data;
18 |  data.add(date-of-birth);
19 |  data.add(form-room);
20 |  data.add(year-group);
21 |  if data[0] != "" and data[1] != "" and data[2] != "" then
22 |  |  go to home activity;
23 |  else
24 |  |  if !internet-connection then
25 |  |  |  go to no network activity;
26 |  |  else
27 |  |  |  go to details activity;
28 |  |  end
29 |  end
30 end
```

Algorithm: onCreate()

```
1  initialise-activity;
2  Integer SPLASH-TIME-OUT = 250;
3  wait(SPLASH-TIME-OUT);
4  manageAuth();
```

2.6.5 No Network Activity

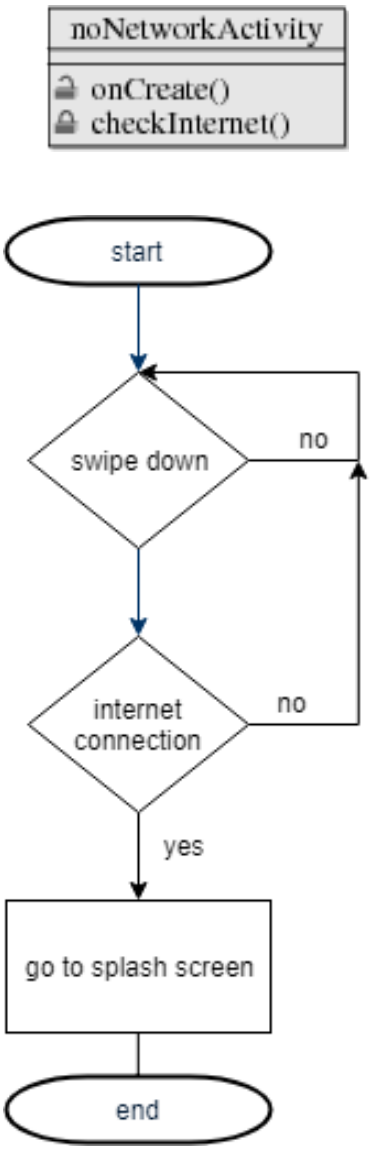
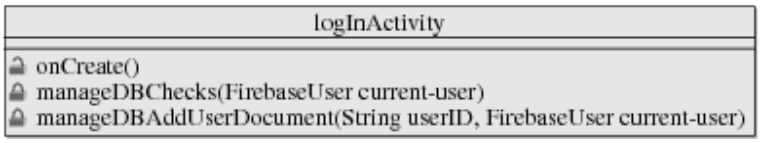


Figure 2.8: onCreate()

Algorithm: checkInternet()

```
1 if mobile data is connected or wi-fi is connected then
2 |   return true;
3 else
4 |   return false;
5 end
```

2.6.6 Log in Activity



Algorithm: manageDBChecks(FirebaseUser current-user)

```
1  String userID = get current-user ID;
2  DocumentReference document = get current-user document;
3  if document exists then
4  |  manageDBAddUserDocument(userID, user);
5  |  go to details activity;
6  else
7  |  go to home activity;
8  end
```

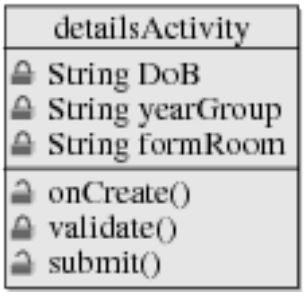
Algorithm: manageDBAddUserDocument(String userID, FirebaseUser current-user)

```
1  HashMap userData;
2  String name = get current-user name;
3  Double rating = 3.5;
4  userData.add("name", name);
5  userData.add("scannedBooks", []);
6  userData.add("loanedBooks", []);
7  userData.add("borrowedBooks", []);
8  userData.add("ratings", [rating]);
9  userData.add("transactionHistory", []);
10 userData.add("dateOfBirth", "");
11 userData.add("yearGroup", "");
12 userData.add("formRoom", "");
13 initialise new document with ID=userID and data=userData;
```

Algorithm: onCreate()

```
1  initialise-activity;
2  load-firebaseUI;
3  if sign up is successful then
4  |  FirebaseUser current-user = initialise current user;
5  |  manageDBChecks();
6  else
7  |  output error message to user;
8  end
```

2.6.7 Details Activity



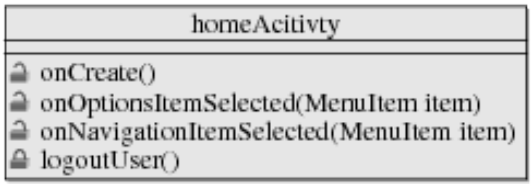
Algorithm: validate()

```
1  Array[2] result;
2  if year group is integer then
3  |  if year group is between 7 and 13 then
4  |  |  if form room length is between 2 and 4 then
5  |  |  |  result.add("t");
6  |  |  |  result.add("valid");
7  |  |  else
8  |  |  |  result.add("f");
9  |  |  |  result.add("form room invalid")
10 |  |  end
11 |  else
12 |  |  result.add("f");
13 |  |  result.add("year group must be between 7 and 13")
14 |  end
15 else
16 |  result.add("f");
17 |  result.add("year group must be an integer");
18 end
19 return result;
```

Algorithm: submit(View v)

```
1  this.DoB = get input date of birth;
2  this.yearGroup = get input year group;
3  this.formRoom = get input year group;
4  String[2] validate = validate();
5  if validate[0] == "f" then
6  |  output validate[1];
7  else
8  |  add data to firebase;
9  end
```

2.6.8 Home Activity



Algorithm: logoutUser()

```
1 FirebaseUser current-user = initialise current user;
2 log current-user out;
3 if log out successful then
4 |   output log out message to user;
5 |   go to log in activity;
```

Algorithm: onCreate()

```
1 initialise-activity;
2 initialise-toolbar;
3 initialise-bottom-navigation;
4 show home fragment;
```

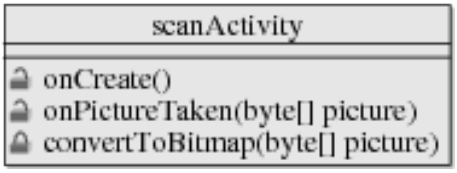
Algorithm: onOptionsItemSelected(MenuItem Item)

```
1 switch Item id do
2 |   case item do
3 |   |   logoutUser();
4 |   |   return True;
5 |   end
6 |   otherwise do
7 |   |   return null;
8 |   end
9 end
```

Algorithm: onNavigationItemSelected(MenuItem Item)

```
1 switch Item id do
2 |   case home do
3 |   |   show home fragment;
4 |   |   return True;
5 |   end
6 |   case books do
7 |   |   show books fragment;
8 |   |   return True;
9 |   end
10|   case search do
11|   |   show search fragment;
12|   |   return True;
13|   end
14|   return False;
15 end
```

2.6.9 ISBN Scanner Activity



Algorithm: onCreate()

```
initialise-activity;
initialise-toolbar;
if scan button clicked then
    capture image;
if flash button clicked then
    if flash is on then
        turn flash off;
    else
        turn flash on;
    end
if manual button clicked then
    go to manual input activity;
```

Algorithm: convertToBitmap(byte[] picture)

```
ByteArrayInputStream arrayStream = new ByteArrayInputStream(picture);
Bitmap image = convert arrayStream to bitmap;
return image;
```

Algorithm: onPictureTaken(byte[] picture)

```
Bitmap bitmap = convertToBitmap(picture);
FirebaseBarcodeOptions options = set options to EAN-13 and EAN-8;
FirebaseVisionImage image = FirebaseVisionImage using bitmap;
FirebaseVisionBarcodeDetector detector = FirebaseVisionBarcodeDetector using options;
List barcodes = find bar codes in image using detector;
if barcodes is successful in declaration then
    if barcodes.length > 1 then
        output too many barcodes message to user;
    else
        if barcodes.length == 1 then
            attempts = 0;
            String rawValue = barcodes[0].getRawValue();
            set isbn text on screen to rawValue;
            output barcode found message to user;
            go to add book activity, passing rawValue too;
        else
            attempts += 1;
            output barcode not found message to user;
            if attempts > 3 then
                show alert box with text input and OK, Cancel buttons;
                if OK button clicked then
                    String rawValue = get value entered;
                    go to add book activity, passing rawValue too;
                if Cancel button clicked then
                    attempts = 1;
                    go back to activity;
        end
    end
```

I took the liberty here to add a system that my stakeholders have not advised me about and nor have I asked them about, I will discuss the feature with my next meeting. Never the less I have implemented it anyway; the features checks if the user has scanned a book wrong 3 times at which point it will present a dialogue of manual isbn input and then continue to fetch the data. This is a simple but potentially nifty feature that will save the end user a lot of time and effort.

2.7 Software Development Life Cycle

A traditional approach to the stages of the Software Development Life Cycle for analysis and design will be taken. However, once development begins, an iterative approach will be taken which includes continuous user involvement and testing. I will do it this way because it allows me to make multiple prototypes and show the stakeholder as I develop the program further to find any errors and compare it to my initial requirements. It is also useful for making high quality code quickly, which is necessary as this project has a limited deadline, and other SDLC methods like waterfall take a longer time to go through so that I would have no time to iterate back and adapt the program as per the users' requirements.

2.8 Stakeholder Feedback

I showed the initial idea for the screen designs and the general inner workings of the application to some of my stakeholders, Lewis and Shashi. Their responses was that it is exactly how they imagined it to be, the navigation and usability features. A reminder from lewis was to ensure that this is reciprocated in further iterations and also that data is kept private throughout. When I discussed the minor extra feature I added, manual isbn input given failed scanning attempts. They seemed to be fully in support and thought it was a good idea however they informed me that the number of times before it is activated should be around 2 not 3. I also discussed a colour scheme, we agreed on white and green. I will take these details forward and implement them to the best of my ability. I will remain in contact so that I can ensure that no changes in requirements have occurred.

2.9 Test Data

Test No	Test Description	Input	Expected Output	Input Type
1	Splash Screen - no user is logged in	Load Activity	Goes to log in activity	Valid
2	Splash Screen - what happens if user is logged in with recorded details	Load Activity	Goes to home activity	Valid
3	Splash Screen - what happens if user logged in with no details recorded	Load Activity	Goes to details activity	Valid
4	Splash Screen - what happens if no internet available	Load Activity	Goes to no network activity	Valid
5	No Network - what happens if no internet is available	Swipe Down on activity	Toast output stating no internet is available	Valid
6	No Network - what happens if internet is available	Swipe Down on activity	Toast output stating internet is available. Goes to splash screen activity	Valid
7	Sign Up (New User) - user email validation	"a random string"	Error message stating email is invalid	Invalid

8	Sign Up (New User) - user email validation	“test@user.com”	No error message	Valid
10	Sign Up (New User) - user name validation	“Test User”	No error message	Valid
11	Sign Up (New User) - password validation	“test”	Error message stating password is too short	Invalid
12	Sign Up (New User) - password validation	“test123”	No error message	Valid
13	Sign Up (New User) - submit	Submit button click	Goes to details activity	Valid
14	Log In (Existing User) - email	“a random string”	Error message stating email is invalid	Invalid
15	Log In (Existing User) - email	“test@user.com”	No error message	Valid
16	Log In (Existing User) - password	“WrongPassword”	Error message stating password is incorrect	Invalid
17	Log In (Existing User) - password	“test123”	No error message	Valid
18	Log In (Existing User with not details recorded) - submit	Submit button click	Goes to details activity	Valid
19	Log In (Existing User with details recorded) - submit	Submit button click	Goes to home activity	Valid
20	Details - date of birth validation	“17/08/2001”	No error message	Valid
21	Details - year validation	“11”	No error message	Valid
22	Details - year validation	“ ”	Error message year must be an int	Invalid
23	Details - year validation	“99”	Error message year must be between 7 and 13	Invalid
24	Details - form validation	“L1”	No error message	Valid
25	Details - form validation	“notfm”	Error message stating form is invalid	Invalid
26	Details - submit	Submit button click	Goes to home activity	Valid
27	Home - home fragment	Home button click	Goes to home fragment	Valid
28	Home - books fragment	Books button click	Goes to books fragment	Valid
29	Home - search fragment	Search button click	Goes to search fragment	Valid
30	Home - logout	Logout button click	Logs user out. Goes to login activity	Valid
31	Home - restart	Restart button click	Restarts app. Goes to splash screen activity	Valid
32	Scan - click scan button with no internet	Scan button floating action button click (in books fragment)	Error message stating internet is not available	Extreme
33	Scan - click scan button with internet	Scan button floating action button click (in books fragment)	Goes to scan activity	Valid

34	Scan - flash on	Flash floating action button click	Turns flash on	Valid
35	Scan - flash off	Flash floating action button click	Turns flash off	Valid
36	Scan - capture button click with no internet	Capture floating action button click	Error message stating internet is not available	Extreme
37	Scan - capture button click with internet and no barcode in sight	Capture floating action button click	Error message stating barcode not found	Valid
38	Scan - capture button click with internet and no barcode in sight x3	Capture floating action button click	Manual input box appears	Valid
39	Scan - Manual input box	Ok click. Barcode Input is 9781108412742	Goes to add book activity	Valid
40	Scan - Manual input box	Cancel click	Goes back to scan activity	Valid
41	Scan - manual input button click	Manual input floating action button click	Goes to manual add book activity	Valid
42	Scan - capture button click with internet and barcode in sight	Capture floating action button click. Barcode is 9781108412742	Barcode found message. Goes to add book activity	Valid
43	Scan - capture button click with internet and fake barcode in sight	Capture floating action button	No barcode found message.	Extreme
44	Add Book - accept button click	Accept floating action button click	Goes to books fragment. Adds books to firebase	Valid
45	Add Book - decline button click	Decline floating action button click	Goes to manual add book fragment	Valid
46	Add book - book fetched correctly	Activity load	Image shown with title, author and ISBN of book with barcode of 9781108412742, (OCR Compsci A-Level book)	Valid
47	Manual Add Book - ISBN validation	“123”	Error message stating input is invalid	Invalid
48	Manual Add Book - title validation	“a”	Error message stating input is invalid	Invalid
49	Manual Add Book - author validation	“a”	Error message stating input is invalid	Invalid
50	Manual Add Book - genre validation	“a”	Error message stating input is invalid	Invalid
51	Manual Add Book - ISBN validation	“9781108412742”	No error message	Valid
52	Manual Add Book - title validation	“OCR GCSE COMPSCI”	No error message	Valid
53	Manual Add Book - author validation	“Cambridge Press”	No error message	Valid
54	Manual Add Book - genre validation	“Education”	No error message	Valid

55	Manual Add Book - submit button	Submit floating action button click	Goes to books fragment. Adds books to firebase	Valid

Chapter 3

Implementation One

3.1 Integrated Development Environment

Throughout this development process I will be working with the IDE Android Studio. This IDE is the recommended platform for android development. It is developed jetBrains, a private company that creates software development tools, but endorsed by Google. For this reason, it has all the features that I require:

1. Intelligent Code Editor - The code editor offers advanced code completion, refactoring, and code analysis.
2. Fast Emulator - Android Emulator installs and starts apps efficiently thus allowing fast prototyping and a means to test my app on various Android device configurations.
3. Testing tools and Frameworks - Android Studio provides extensive tools to help test your Android apps including Espresso Test Recorder to capture screen recordings of UI tests.
4. Robust Build System - Android Studio offers build automation, dependency management, and customizable build configurations. I can configure my project to include local and hosted libraries, and define build variants that include different code and resources, and apply different code shrinking and app signing configurations.
5. Firebase Integration - The Firebase Assistant will allow me to connect my app to Firebase and add services such as Analytics, Authentication, Notifications and more with step-by-step procedures right inside Android Studio.
6. Layout Editor - When working with XML layout files, Android Studio provides a drag-and-drop visual editor that makes it easier than ever to create a new layout.
7. APK Analyser - The APK Analyzer can easily inspect the contents of your APK. It reveals the size of each component so you can identify ways to reduce the overall APK size.
8. Vector Asset Studio - Android Studio makes it easy to create a new image asset for every density size. With Vector Asset Studio, you can select from Google-provided material design icons or import an SVG or PSD file.

3.2 Development Prerequisite Knowledge

As I have previously discussed, I will be using a plethora of non-native libraries for various different uses. These libraries all have their own way of being implemented and associated rules. They also have extensive documentation which I will be studying and then extracting the relevant technologies that apply to me.

3.2.1 Android Manifest and Gradle

An android application consisting of app resources, source code and packages (libraries) into an executable APK that can be run, tested and distributed. Android uses *Gradle*, an advanced build toolkit, to automate and manage the build process. The part that I should be concerned with,

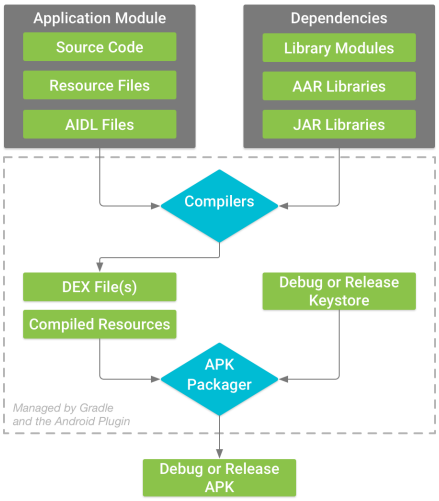


Figure 3.1: Typical Build Process

is interacting with a Gradle file. When writing code it is common to call the import function for example to call in a library to use. The Gradle file offers similar functionality, I can simply call what library modules I want and from there own they will be universally available throughout my code. The key part here is that they will also be compiled along with my code to create the final APK. A Gradle file can be expected to look something like:

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.bookstore.palvindersingh"
        minSdkVersion 23
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support:design:27.1.1'
```

```
23      implementation 'com.android.support.constraint:constraint-layout:1.1.3'
24      implementation 'com.android.support:support-v4:27.1.1'
25      testImplementation 'junit:junit:4.12'
26      androidTestImplementation 'com.android.support.test:runner:1.0.2'
27      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
28      implementation 'com.google.firebase:firebase-core:16.0.4'
29      implementation 'com.google.firebase:firebase-auth:16.0.5'
30      implementation 'com.firebaseui:firebase-ui-auth:4.2.0'
31      implementation 'com.google.firebase:firebase-firestore:17.1.1'
32      implementation 'com.squareup.picasso:picasso:2.71828'
33      implementation 'android.arch.navigation:navigation-fragment:1.0.0-alpha06'
34      implementation 'com.google.firebase:firebase-ml-vision:18.0.1'
35      implementation 'com.otaliastudios:cameraview:1.6.0'
36  }
37
38  apply plugin: 'com.google.gms.google-services'
```

This is my Gradle file for my android application. It is automatically generated when creating a new project in Android Studio. As can be seen, I have set *minSdkVersion* to 23. This should cover roughly 75% of all android devices to date. Another key piece of information is the *applicationId*, this is the unique package name of my application. I have decided to call it *com.bookstore.palvindersingh*. The android project can be thought as a package of library itself. This identifier is the libraries name. By using this I will be able to store multiple classes in separate files thus decomposing the development process. The *dependencies* definition is where I will declare any universal libraries I want to use, I have already added all the libraries I will use for this iteration. Android development also requires a manifest file, This can be thought of a file that is more of a contents page, i.e it defines each activity and can be used to add meta data about them.

3.2.2 Firebase

I will be using 3 features of Firebase: Authentication, Firestore and ML-kit. Firebase has extensively documented these features on their website. I will be referring to this in order to learn how to interact with their services.

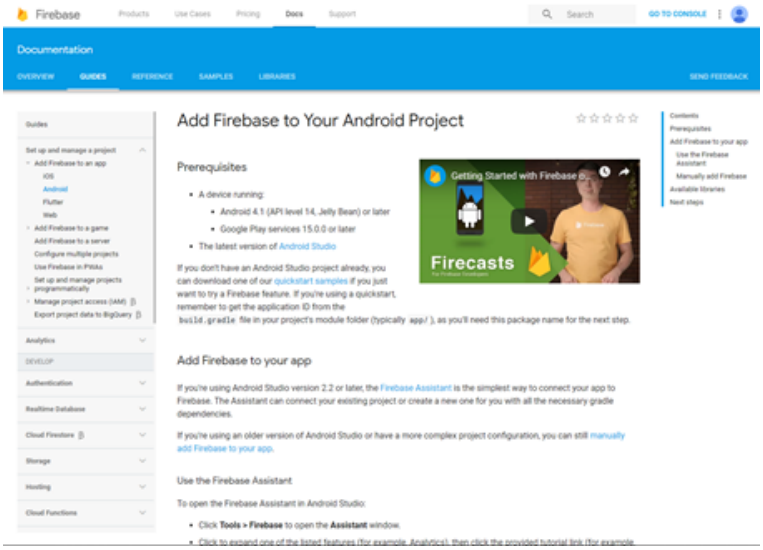


Figure 3.2: Firebase Documentation

On the Firebase website there is also a feature called Firebase Console. This is where I can create an instance of these features and then connected it to my android project. The Console also allows me to look at any data that has passed through Firebase, most importantly users and Firestore data. It does not let me look at the users passwords; this is excellent as it given me confidence that the system has been developed with a sense of security in mind. I have set up Firebase as follows:

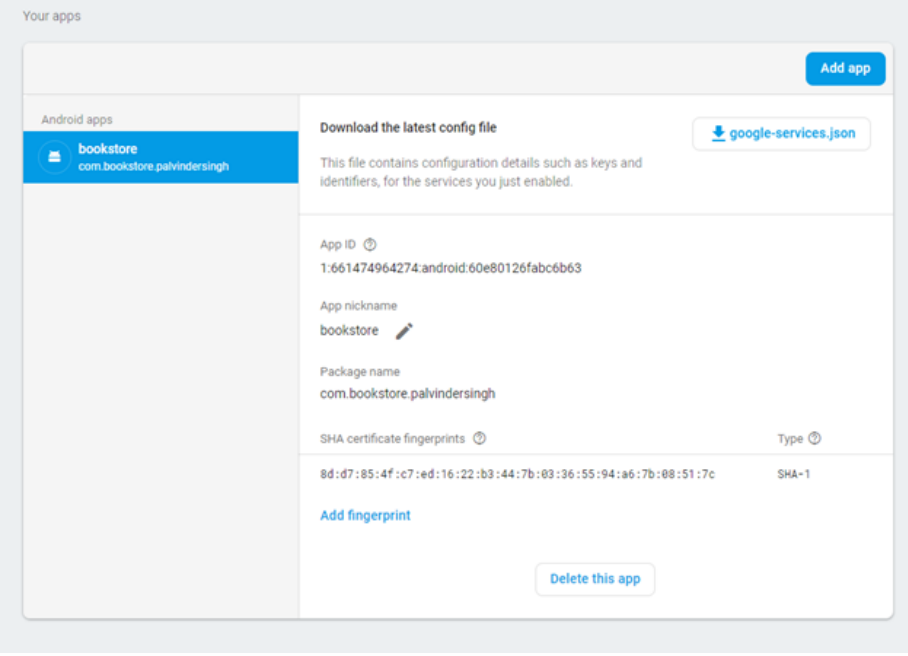


Figure 3.3: Firebase Console

In order to add Firebase to my android application I have to download the google-services.json file and place it in the directory of my gradle file.

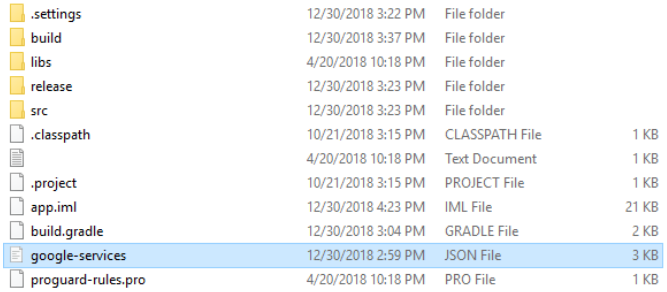


Figure 3.4: google-services.json file

Now that I have Firebase set-up along with all the other prerequisites, I should be in good stead to start development.

3.2.3 File Layout and Colouring

Within my IDE, on a side panel, I am presented with the file structure layout of the application. The areas to note here are manifest, com.bookstore.palvindersingh, drawable, layout and values.

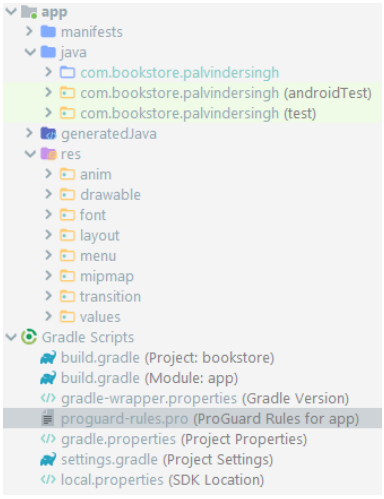


Figure 3.5: Files

The com.bookstore.palvindersingh directory stores all classes. The manifest directory holds the manifest file. The drawable directory holds all images. The layout directory holds all XML activity layouts. Finally, the values directory holds all constants. These constants are composed of colors and strings. Whenever I add text to an activity, in an XML layout, my IDE will automatically added to the strings file. It is the colors file that I must define myself. I have declared the following:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#1abc9c</color>
    <color name="colorPrimaryDark">#16a085</color>
    <color name="colorAccent">#00574B</color>
    <color name="white">#FFFFFF</color>
</resources>
```

I have also designed the following logo that will be used for the application. I created this logo within Photoshop, using the colors described above, thus maintaining the color scheme of the application. It should also be noted that android has it's own set of icons that can be used within

Figure 3.6: Logo

the application. I will use these pre made icons within my app, this this logo is the only asset/image I need to create. I showed this image to my stakeholders Jeevon and Lewis, they told me "the logo is unique and the color scheme is clear". This is positive as I want my color scheme to not hinder the use of those with lower resolution screen, use in low light conditions and also users who may have some visionary medical condition.

3.3 Floating Action Button

The implementation of this Floating Action Button will be quite abstract. This is because, although being its own entity, it cannot be tested independent of other classes as it is a interface component. I will need to have this button coded before I can complete other stages of development. For this reason, I will be implementing the *networkFAB* class now and testing it as soon as it is implemented in actual activity. After referring back to my design application, I can see three things that I must do before implementing it. First off, I need to find a way of dealing with finding whether or no internet is available. Secondly, I must find how to display an output message to the user without implementing any special interface; i.e independent of any layout. Finally, I must find what constructors I must declare as part of the parent class.

After reading through the Android Development Documentation[6], I discovered a simple, native way to check for internet. It is slightly over-complicated and can be simplified for my use. Their code reads:

```
1 ConnectivityManager cm =
  ↪ (ConnectivityManager)context.getSystemService(Context.CONNECTIVITY_SERVICE);
2 NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
3 boolean isConnected = activeNetwork != null && activeNetwork.isConnectedOrConnecting();
```

Before I get onto how I will simplify this method, I must first consider the variable associated with this code. It seems as though the procedure is dependent on the variable *context*. From prior research, I know that it is an abstract class that is used as an interface to global information about the application environment; in other words, it allows the programmer to interface with the device. I am not quite sure however how to derive this class as it can not be immediately instantiated. With this variable being so fundamental to the application, I think that it is reasonably logical to suggest that it may be passed into the parent *FloatingActionButton* class via it's constructor. I will now research what constructor methods I must implement (this will handily solve my third issue). Fortunately, my IDE provides a *generation* feature that can implement parent constructors itself without the need for me identify what they are prior. As can be seen, the constructors which

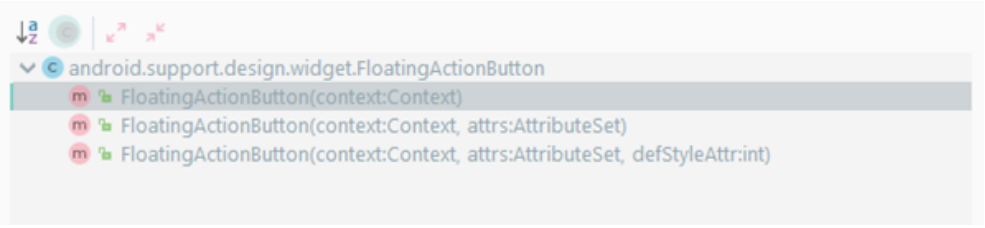


Figure 3.7: networkFAB constructors

I must implement all have *context* as a parameter. This means that I should be able to simply use that attribute of the parent class to check for internet. Now I can discuss how I will simplify the procedure. I will simply return the *isConnected* declaration rather than creating the variable itself. Now onto the second issue, outputting an error message. In using an android device, I have seen a UI component that is commonly used in various applications for the purpose that I describe. That UI component is called a *toast*.

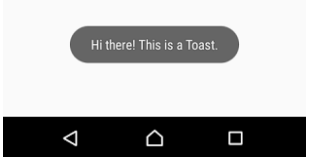


Figure 3.8: Toast

Again, Android documentation details that it can be implemented by doing:

```
Toast.makeText(<CONTEXT>.getApplicationContext(), "<MESSAGE>", Toast.LENGTH_SHORT).show();
```

Context is required here, similar to how it is required to check for internet. Given that these requirements both occur in the same *onClick* method I will declare a local variable within that can then be passed to the *checkInternet* method and used to make a toast. Given these solutions, I have now coded the final (to-be tested) class:

```
package com.bookstore.palvindersingh;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.design.widget.FloatingActionButton;
import android.util.AttributeSet;
import android.view.View;
import android.widget.Toast;
//class to inherit form floating action button and implement an on click listener interface in order to automate internet checking
public class networkFAB extends FloatingActionButton implements View.OnClickListener {
    //initialise attributes
    OnClickListener clickEvent;
    //initialise parent constructors
    public networkFAB(Context context) {
        super(context);
        init();
    }
    public networkFAB(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }
    public networkFAB(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }
    //method to set an on click listener
    private void init() {
        setOnClickListener(this);
    }
    //setter
    public void setClickEvent(OnClickListener l) {
        this.clickEvent = l;
    }
    //method to check for internet
    private Boolean checkInternet(Context c) {
        ConnectivityManager connMgr = (ConnectivityManager)
        ↪  c.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        return (networkInfo != null && networkInfo.isConnected());
    }
    //override method for on click event
    @Override
    public void onClick(View v) {
        Context c = super.getContext();
        //if there is internet
        if (checkInternet(c)) {
            //perform clickEvent instructions
            clickEvent.onClick(v);
        } else {
            //prompt the user that there is no internet available
            Toast.makeText(c.getApplicationContext(), "no internet connection",
            ↪  Toast.LENGTH_SHORT).show();
```

```
51          }
52      }
53  }
```

3.4 Book Class

This was a fairly simple implementation given that its more of a data store than a way of manipulating data. Instead implementing this class into an activity, I will instead create a Java main class to test this class.

```
1  package com.bookstore.palvindersingh;
2  import java.util.ArrayList;
3  public class book {
4      //initialise attributes
5      private ArrayList<String> authors;
6      private ArrayList<String> genres;
7      private String image;
8      private String isbn;
9      private String title;
10     private String ownerReference;
11     private Boolean market;
12     private String firestoreID;
13     private Boolean renting;
14     private String rentingID;
15     //constructor
16     public book() {
17     }
18     //constructor
19     public book(ArrayList<String> authors, ArrayList<String> genres, String image, String isbn,
       ↪  String title, String ownerReference, Boolean market, Boolean renting, String rentingID) {
20         this.authors = authors;
21         this.genres = genres;
22         this.image = image;
23         this.isbn = isbn;
24         this.title = title;
25         this.ownerReference = ownerReference;
26         this.market = market;
27         this.renting = renting;
28         this.rentingID = rentingID;
29     }
30     //getter and setters
31     public String getFirestoreID() {
32         return firestoreID;
33     }
34     public void setFirestoreID(String firestoreID) {
35         this.firestoreID = firestoreID;
36     }
37     public ArrayList<String> getAuthors() {
38         return authors;
39     }
40     public ArrayList<String> getGenres() {
41         return genres;
42     }
43     public String getImage() {
44         return image;
45     }
46     public String getIsbn() {
47         return isbn;
48     }
49     public String getTitle() {
50         return title;
51     }
52     public String getOwnerReference() {
53         return ownerReference;
54     }
```

```
55      public void setOwnerReference(String ownerReference) {
56          this.ownerReference = ownerReference;
57      }
58      public Boolean getMarket() {
59          return market;
60      }
61      public void setMarket(Boolean market) {
62          this.market = market;
63      }
64      public Boolean getRenting() {
65          return renting;
66      }
67      public void setRenting(Boolean renting) {
68          this.renting = renting;
69      }
70      public String getRentingID() {
71          return rentingID;
72      }
73      public void setRentingID(String rentingID) {
74          this.rentingID = rentingID;
75      }
76  }
```

After coding this I then went ahead and completed the following tests:

```
1  ArrayList authors = new ArrayList();
2  authors.add("bob");
3  authors.add("billy");
4  ArrayList genres = new ArrayList();
5  authors.add("education");
6  book b = new book(authors, genres, "image", "1234567890", "this is the title", "abcOwnerRef123",
   ↪  False, False, "123rentingIDABC");
7  System.out.println(b.getAuthors());
```

The following was correctly output with no runtime errors:

```
1  ["bob", "billy"]
```

3.5 Fetch Meta Data

As part of this implementation I do not know how to read websites raw display data. This is something that I will now research how to do. Before I do so, I will implement all that I can, commenting the areas I must develop further. This should make the process simpler for what I expect to be a difficult implementation. The method that I must now build upon looks like so:

```
1  private JSONObject getJSONData() {
2          try {
3
4              //TODO reading web HERE
5
6              return new JSONObject(data);
7          } catch (IOException | JSONException e) {
8              e.printStackTrace();
9              return null;
10         }
11     }
```

I briefly mentioned before ,in my design of this class, a buffered reader. This is essentially a line by line reader of a website that will ultimately return a string. Knowing this I went a did some research into the area and found some useful, but abstract documentation[7] that fits my needs. The code within it is:

```
1  import java.net.*;
2  import java.io.*;
3  public class URLReader {
4      public static void main(String[] args) throws Exception {
5          URL oracle = new URL("http://www.oracle.com/");
6          BufferedReader in = new BufferedReader(
7          new InputStreamReader(oracle.openStream()));
8          String inputLine;
9          while ((inputLine = in.readLine()) != null)
10             System.out.println(inputLine);
11         in.close();
12     }
13 }
```

This code seems to be instantiating a URL object with the parameter being a string that is url. A buffered reader is then declared to read the url. After inputting this into my IDE, I was returned with a message detailing this code included redundant method implementations. Fortunately, my IDE's intelligent code completion software also managed to solve this problem by implementing the correct alternative methods. The finished class now looks like:

```
1  package com.bookstore.palvindersingh;
2
3  import org.json.JSONArray;
4  import org.json.JSONException;
5  import org.json.JSONObject;
6  import java.io.BufferedReader;
7  import java.io.IOException;
8  import java.io.InputStream;
9  import java.io.InputStreamReader;
10 import java.net.HttpURLConnection;
11 import java.net.URL;
12 import java.util.ArrayList;
13 class bookMetaData {
14     //initialise attributes
```

```
15      private final String isbn;
16      private book book;
17      //constructor
18      bookMetaData(String isbn) {
19          this.isbn = isbn;
20          this.book = getData();
21      }
22      //method to get book meta data
23      private book getData() {
24          //get data
25          JSONObject JSONData = getJSONData();
26          JSONObject volumeInfo = getVolumeInfo(JSONData);
27          //check if data exists
28          if (volumeInfo == null) {
29              //return null to indicate a result does not exist
30              return null;
31          } else {
32              //extract meta data
33              String title = getBookTitle(volumeInfo);
34              String image = getImageURl(volumeInfo);
35              ArrayList<String> authors = getAuthor(volumeInfo);
36              ArrayList<String> genres = getGenre(volumeInfo);
37              //return a book object from data
38              book book = new book(authors, genres, image, this.isbn, title, null, false, false,
                ↪  null);
39              return book;
40          }
41      }
42      //method to pull main data from google books api
43      private JSONObject getJSONData() {
44          try {
45              //buffer read the url
46              URL url = new URL("https://www.googleapis.com/books/v1/volumes?q=isbn:" + this.isbn);
47              HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
48              InputStream inputStream = httpURLConnection.getInputStream();
49              BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
50              String data = "";
51              String line = "";
52              while (line != null) {
53                  line = bufferedReader.readLine();
54                  data += line;
55              }
56              return new JSONObject(data);
57          } catch (IOException | JSONException e) {
58              e.printStackTrace();
59              return null;
60          }
61      }
62      //method to parse the first layer of the overall data
63      private JSONObject getVolumeInfo(JSONObject JSONData) {
64          try {
65              JSONArray items = (JSONArray) JSONData.get("items");
66              JSONObject firstItem = (JSONObject) items.get(0);
67              return (JSONObject) firstItem.get("volumeInfo");
68          } catch (JSONException e) {
69              e.printStackTrace();
70              return null;
71          }
72      }
73      //method to extract book title
74      private String getBookTitle(JSONObject volumeInfo) {
```

```
75          try {
76              return (String) volumeInfo.get("title");
77          } catch (JSONException e) {
78              e.printStackTrace();
79              return null;
80          }
81      }
82      //method to extract image url
83      private String getImageURl(JSONObject volumeInfo) {
84          try {
85              JSONObject urls = (JSONObject) volumeInfo.get("imageLinks");
86              return (String) urls.get("thumbnail");
87          } catch (JSONException e) {
88              e.printStackTrace();
89              return null;
90          }
91      }
92      //method to extract author array
93      private ArrayList<String> getAuthor(JSONObject volumeInfo) {
94          try {
95              JSONArray authors = (JSONArray) volumeInfo.get("authors");
96              return convertToArrayListFromJSON(authors);
97          } catch (JSONException e) {
98              e.printStackTrace();
99              return null;
100         }
101     }
102     //method to extract genre array
103     private ArrayList<String> getGenre(JSONObject volumeInfo) {
104         try{
105             JSONArray categories = (JSONArray) volumeInfo.get("categories");
106             return convertToArrayListFromJSON(categories);
107         }catch (JSONException e){
108             e.printStackTrace();
109             return null;
110         }
111     }
112     //method to convert a json array to array list for generics
113     private ArrayList<String> convertToArrayListFromJSON(JSONArray jsonArray) throws JSONException {
114         ArrayList<String> data = new ArrayList<>();
115         for (int i = 0; i < jsonArray.length(); i++) {
116             String item = jsonArray.get(i).toString();
117             data.add(item);
118         }
119         return data;
120     }
121     //getter
122     public book getBook() {
123         return book;
124     }
125 }
```

My code here slightly differs from my initial pseudocode, in that I have had to declare specifically what exception I am catching. The reason I have to implement try/catch statements in the first place is that the data may or may not exist in which case my code should be able to handle this scenario or any other similar error. This is also why I implemented specific exceptions rather than catching all errors; these were suggested to me by my IDE. I have also implemented a feature of Java called *Generics*. Generic Programming is the process in which the programmer can take an object of type X, for example, and then declare a data structure also with the type X. This will ultimately mean that all of the data structures methods have been changed to return or take an object of type X. This can then be adapted to other objects thus creating allowing for a versatile

data structure that can be used with the data it is holding in mind. Applying this abstract definition to code meant that I could have confidence that the data my *ArrayList* structures are holding are always consistent and that I can always expect to get an object of the correct type. This is especially important when an error in one area could potentially propagate throughout the rest of the methods. After I finished coding this class, I proceeded to complete a series of tests (using the isbn: 9780545010221 from Harry Potter and the Deathly Hallows).

```
bookMetaData data = new bookMetaData("9780545010221");
book book = (book) data.getBook();
System.out.println("title is: " + book.getTitle());
```

This test is deliberately only checks the book title. This is because I first want to see if the code runs correctly without returning any syntax or runtime errors. The results were as follows:

```
java.net.UnknownHostException: www.googleapis.com
        at com.company.bookMetaData.getJSONData(bookMetaData.java:47)
```

As you can see an error was returned, at line 47, this line being:

```
InputStream inputStream = httpURLConnection.getInputStream();
```

Looking at the error, it seems as though it is caused when trying to connect to the web-page as this is the task of the *getInputStream()* method. I soon realised that it was because at run time the internet on my device was off. This would clearly pose a problem for a class whose sole primary purpose is to access the internet. This did make me think however, how I will deal with such an error when it is being run on android device. For this reason I did some research into how abstracted network management is provided in android. I found that it should not create a problem as such an error is handled internally by the kernel. Once I had reconnected my device, I compiled and ran the code again and got successful results:

```
>>> title is: Harry Potter and the Deathly Hallows
>>> Process finished with exit code 0
```

The console output the title correctly, no errors were returned and to confirm this the and exit code of 0 was output (meaning no issues occurred). This test just printed a single attribute so to test the class fully, I then printed every attribute and yielded the following successful results:

```
>>> title is: Harry Potter and the Deathly Hallows
>>> authors is: [J. K. Rowling]
>>> genres is: [Juvenile Fiction]
>>> image is: http://books.google.com/books/content?
↪  id=GZAoAQAAIAAJ&printsec=frontcover&img=1&zoom=1&source=gbs_api
>>> isbn is: 9780545010221
>>> ownerref is: null
>>> market is: false
>>> firestoreID is: null
>>> rentingID is: null
>>> renting is: false
>>> Process finished with exit code 0
```

To see how to class handles with extreme data I repeated the process but instead with an ISBN of *"abc123"* that is not a registered ISBN. The test code is:

```
bookMetaData data = new bookMetaData("abc123");
book book = (book) data.getBook();
```

```
3  if (book == null){
4      System.out.println("no book exists");
5  }else{
6      System.out.println("title is: " + book.getTitle());
7      System.out.println("authors is: " + book.getAuthors());
8      System.out.println("genres is: " + book.getGenres());
9      System.out.println("image is: " + book.getImage());
10     System.out.println("isbn is: " + book.getIsbn());
11     System.out.println("ownerref is: " + book.getOwnerReference());
12     System.out.println("market is: " + book.getMarket());
13     System.out.println("firestoreID is: " + book.getFirestoreID());
14     System.out.println("rentingID is: " + book.getRentingID());
15     System.out.println("renting is: " + book.getRenting());
16 }
```

I have added the if statement that checks if the book is null as this is what is expected to be returned by the bookMetaData class if the search query is void. Furthermore, this is how I will deal with using the class in its actual implementation in the app. The following results were returned:

```
1  >>> no book exists
2  >>> org.json.JSONException: JSONObject["items"] not found.
3          at org.json.JSONObject.get(JSONObject.java:454)
4          at com.company.bookMetaData.getVolumeInfo(bookMetaData.java:64)
5          at com.company.bookMetaData.getData(bookMetaData.java:25)
6          at com.company.bookMetaData.<init>(bookMetaData.java:19)
7          at com.company.Main.main(Main.java:6)
8
9  >>> Process finished with exit code 0
```

As expected *no book exists* was printed. In addition what seems to be an error was output, in fact this is an error that was correctly caught and handled by my try catch statement. I could choose not to print it however it makes the debugging process easier so I chose not to do so.

3.6 Splash Screen

For this section, I will be implementing my first activity/user-interface. To do this, I will call on the designs I previously created and the logo I designed. The activity will be visually simple, the logo behind in the center of the screen. It is the behind the scenes computation that will be of most importance. As with any activity, I must first create the UI. Android development requires this to be done in XML however Android Studio has a handy drag-and-drop system that allows me to graphically create the UI, whilst generating the XML for me. I chose to do use this method for design and then editting the XML manually to add id-tags or other unique attributes. I decided to stray from my class name of splashScreenActivity and rather use the name mainActivity. I have read that this is common practice so that over developers know the first activity that is executed.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:gravity="center"
7      tools:context=".mainActivity">
8
9      <ImageView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:contentDescription="@string/app_name"
13         android:scaleType="center"
14         android:scaleX="2"
15         android:scaleY="2"
16         android:src="@mipmap/ic_launcher_foreground" />
17
18 </LinearLayout>
```

As seen in my IDE:

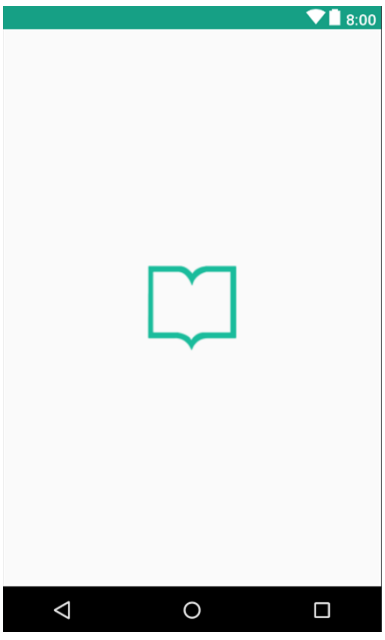


Figure 3.9: Main Activity

There are four implementations that I must research before I start to code this activity: how to delay a method's execution, how to initialise Firestore, how to add a document to Firestore and how to get a document from Firestore. I am already aware that Firebase documentation covers

the last 3 requirements; these I will research as I code as they seem quite straight forward. It is the first that I may find difficultly in solving, and I must do before, as I do not want to use any third part library to complete this task, using standard libraries are a lot more efficient and stable due to their vast use.

I think that delaying a method's execution will have a reference to time or some sort if timer. After doing some research into this, I can confirm that Java does have a native Timer class. This class is described as a facility for threads to schedule tasks for future execution in a background thread. This sounds similar to what I require. However, after reading some comments about the class from other developers, It has come to my attention that it is possible that the class is not dealt with properly and could result to leaked threads. This means that their are dormant threads running but executing nothing and can also not switch to any other task. This is especially a problem for mobile devices as computing power can often be limited an thus any optimisations that are possible should be made. I have however discovered a more relevant way to manage this process, a Handler[8] class. This class is described to have 2 main uses:

1. To schedule messages and runnables to be executed at some point in the future
2. To enqueue an action to be performed on a different thread than your own

The first is what applies to me; documentation as to how to implement this class is also provided.

```
int SPLASH_TIME_OUT = 250;
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        // call method here
    }
}, SPLASH_TIME_OUT);
```

The implementation is rather simple. A variable must be declared to state the duration, in milliseconds. Then a Handler class is defined and a method called. Within this method a class is instantiated with the override method *run()*. It is here that I shall call whatever method I need to. Another thing that I have learnt from this is that instead instantiating a class and then passing it as a parameter, it can be done directly; this is useful as I can start to make my code less cluttered and more understandable. I can now proceed to implement this knowledge.

```
package com.bookstore.palvindersingh;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreSettings;
import java.util.ArrayList;
import java.util.Objects;
public class mainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26          //initialise splash screen
27          int SPLASH_TIME_OUT = 250;
28          new Handler().postDelayed(new Runnable() {
29              @Override
30              public void run() {
31                  manageAuth();
32              }
33          }, SPLASH_TIME_OUT);
34      }
35      //method to check for internet connection
36      private Boolean checkInternet() {
37          ConnectivityManager connMgr = (ConnectivityManager)
            ↪  getSystemService(Context.CONNECTIVITY_SERVICE);
38          NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
39          return (networkInfo != null && networkInfo.isConnected());
40      }
41      //method to manage authentication
42      private void manageAuth() {
43          //initialise firestore connection
44          FirebaseFirestore firestore = FirebaseFirestore.getInstance();
45          FirebaseFirestoreSettings settings = new FirebaseFirestoreSettings.Builder()
46                  .setTimestampsInSnapshotsEnabled(true)
47                  .build();
48          firestore.setFirestoreSettings(settings);
49          //check if there is an internet connection
50          final Boolean internetConnection = checkInternet();
51          FirebaseAuth auth = FirebaseAuth.getInstance();
52          if (!internetConnection){
53              //prompt the user there is no connection
54              Toast.makeText(getApplicationContext(), "no internet connection",
                ↪  Toast.LENGTH_SHORT).show();
55          }
56          //if the user is not logged in
57          if (auth.getCurrentUser() == null) {
58              //if there is no internet
59              if (!internetConnection) {
60                  //go to no network activity
61                  Intent intent = new Intent(mainActivity.this, noNetworkActivity.class);
62                  startActivity(intent);
63                  finish();
64              } else {
65                  //go to login activity
66                  Intent intent = new Intent(mainActivity.this, logInActivity.class);
67                  startActivity(intent);
68                  finish();
69              }
70          } else {
71              //initialise firestore connection
72              FirebaseUser user = auth.getCurrentUser();
73              FirebaseFirestore db = FirebaseFirestore.getInstance();
74              final String userID = user.getUid();
75              DocumentReference userDocument = db.collection("users").document(userID);
76              //get users document
77              userDocument.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
78                  @Override
79                  public void onComplete(@NonNull Task<DocumentSnapshot> task) {
80                      //check if extra details have been input before
81                      ArrayList<String> array = new ArrayList<>();
82                      array.add(task.getResult().get("dateOfBirth").toString());
```

```
83                      array.add(task.getResult().get("formRoom").toString());
84                      array.add(task.getResult().get("yearGroup").toString());
85                      //if they have been input
86                      if (!Objects.equals(array.get(0), "") && array.get(1) != "" && array.get(2) !=
                        ↪  "") {
87                          //go to home activity
88                          Intent intent = new Intent(mainActivity.this, homeActivity.class);
89                          startActivity(intent);
90                          finish();
91                      } else {
92                          //if there is no internet
93                          if (!internetConnection) {
94                              //go to no network activity
95                              Intent intent = new Intent(mainActivity.this, noNetworkActivity.class);
96                              startActivity(intent);
97                              finish();
98                          } else {
99                              //go to details activity
100                             Intent intent = new Intent(mainActivity.this, detailsActivity.class);
101                             startActivity(intent);
102                             finish();
103                         }
104                     }
105                 }
106             });
107         }
108     }
109 }
```

Before I continue to test this code, I must first discuss some of the challenges and learning curves I have encountered whilst coding. The first is to do with changing activities. This is a relatively simple process but nevertheless I had to research how to implement the process. It is done by instantiating an Intent class. This class is an abstract definition of an operation to be executed. The parameters that I pass are detailed to call a constructor that has the following effect: *"create an intent for a specific component"*. In other words the Intent class hold data on what is intended to be operated on. It is is the *startActivity()* method, which is from the abstract Context class, that performs the operation of changing activities. The other difficulties I faced was in relation to the Firebase. The first was to set it up, programmatically stating settings, given this is the first time the app will access the framework in it's life cycle. Doing this was quite simple as all I had to do was instantiate a *FirebaseFirestoreSettings* class. From this I set the use of Timestamps when accessing data to true. This just allows me to see when certain data is accessed, this may be helpful in the future and is generally a good feature to enable for development purposes as If I need to I will able to track when data is accessed. After passing it to a declared Firestore initialistion object, preliminary settings were complete. After this I could begin to access Firebase services. The two services that I use here is Firebase-Auth and Firestore. I use Firebase auth to check if a user is logged in or not, as per the requirements of my pseudocode. This was just a case of declaring an instance of a FirebaseAuth object and then checking if the current user was null or not using the *getCurrentUser()* method. Using Firestore on the other hand was a bit more challenging. In order to access a user's data, to check if they have filled in appropriate details or not, I must first initialise various variables. The first is a FirebaseUser object that uses the previously declared FirebaseAuth object to get the current user. An instance of Firestore is then also declared by instantiating a FirebaseFirstore object and calling *getInstance()*. Behind the scenes this looks at a locally stored cache for meta data about the database and then seeks to check if internet is available to decide whether operations should occur locally or with the cloud. This check is also made before any interaction with Firestore is made to ensure that the optimal option is always selected. The user's document can then be loaded by searching for the document with the id that is the same as the user's id. This process is done internally (abstracted away from me). This is where I got a bit stuck. I thought that since I have got the document I could then proceed to use its methods to access it's data. This was not the case; I realised that I had declared the documents reference and

not yet "got" it. To this I had call the *get()* method. Upon this I added an on complete listener, as instructed in the Firestore documentation. This is because this process takes time and thus I must wait for the data to be completely retrieved before accessing it. Furthermore, what if the document is corrupted or is not successful in retrieving it? This would result in numerous errors. An on complete listener also ensures that the document is retrieved correctly therefore solving this problem.

I can now test this code. However, I have not yet created the other activities, therefore for testing purposes I will change each Intent to load the same activity again and instead print the intended location. It must also be noted that given I have not completed the login stages of development it is certain that no users documents will exist, for this reason a branch of the program should be guaranteed never to be reached, meaning I cannot test it. I will have to test this again later. Given this there are two external variables that can effect the flow of my program, whether there is internet or not. I will turn of my device internet for the first test. It is expected that the console should print that it has gone to "noNetwork Activity". The altered portion of code looks like:

```
...
System.out.println("I have gone to noNetActivity")
Intent intent = new Intent(mainActivity.this, mainActivity.class);
startActivity(intent);
finish();
...
...
System.out.println("I have gone to loginActivty")
Intent intent = new Intent(mainActivity.this, mainActivity.class);
startActivity(intent);
finish();
```

The results are successful (note that the other messages displayed are from the emulator not the application.):

```
>>> D/FA: Connected to remote service
>>> V/FA: Processing queued up service tasks: 4
>>> I/System.out: I have gone to noNetActivity
```

Doing the same for the activity, with internet, correctly yields:

```
>>> D/FA: Connected to remote service
>>> V/FA: Processing queued up service tasks: 2
>>> I/System.out: I have gone to loginActivty
>>> D/EGL_emulation: eglMakeCurrent: 0xe6ce57c0: ver 3 1 (tinfo 0xe6c0ed10)
```

After implementing login activity - I have returned to this branch of documentation after completing the login activity, as I can now test all branches of the code.

Given that I have logged in before but not provided the required user details yet (date of birth, form room and year group) my program should go to the details activity. As I have done before, I will print this out instead since the details activity is not actually yet created. My edited code looks like so:

```
...
System.out.println("I have gone to detailsActivity")
Intent intent = new Intent(mainActivity.this, mainActivity.class);
startActivity(intent);
finish();
...
```

The following successful results were output:

```
1 >>> I/System.out: I have gone to detailsActivity
```

3.7 Log In

As I have previously discussed, I will be using Firebase-UI to generate the login interface. Thus the XML file for this activity does not need to contain anything except for the general boilerplate code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".logInActivity">

</LinearLayout>
```

Implementing this class was relatively straight forward, the only required knowledge was how to call the login-UI, which after some research I found to be quite simple. The question that was posed to me was how do I want users to be registers, with the options of email, google mail, twitter, facebook or github account. I chose email as this is the simplest and the most stable, the other formats for login require other APIs to be implemented furthermore my target audience is around 11-17; half of these people are under the age limit to use twitter and facebook thus its implementation would be fruitless. After deciding this, it was just a matter if following documentation to implement the correct code:

```
package com.bookstore.palvindersingh;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;
import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.auth.IdpResponse;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class logInActivity extends AppCompatActivity {
    //initialise attributes
    private static final int RC_SIGN_IN = 123;
    private final List<AuthUI.IdpConfig> providers = Collections.singletonList(
            new AuthUI.IdpConfig.EmailBuilder().build());
    private FirebaseFirestore db;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log_in);
        //initialise firestore connection
        db = FirebaseFirestore.getInstance();
        //initialise start sign in/up activity
```

```
35          startActivityForResult(AuthUI.getInstance()
36                  .createSignInIntentBuilder()
37                  .setAvailableProviders(providers)
38                  .setLogo(R.mipmap.ic_launcher)
39                  .build(), RC_SIGN_IN);
40      }
41      //override method to manage the results of sign up
42      @Override
43      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
44          super.onActivityResult(requestCode, resultCode, data);
45          if (requestCode == RC_SIGN_IN) {
46              IdpResponse response = IdpResponse.fromResultIntent(data);
47              if (resultCode == RESULT_OK) {
48                  //sign in is successful
49                  FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
50                  Toast.makeText(getApplicationContext(), "sign in successful",
                    ↪  Toast.LENGTH_SHORT).show();
51                  manageDBChecks(user);
52              } else {
53                  //sign in failed
54                  Toast.makeText(getApplicationContext(), "sign in failed",
                    ↪  Toast.LENGTH_SHORT).show();
55                  finish();
56              }
57          }
58      }
59      //method to check if the user has input extra data about themselves before
60      private void manageDBChecks(final FirebaseUser user) {
61          final String userID = user.getUid();
62          //get the users document
63          DocumentReference userDocument = db.collection("users").document(userID);
64          userDocument.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
65              @Override
66              public void onComplete(@NonNull Task<DocumentSnapshot> task) {
67                  //check if user exists in firestore
68                  if (!Objects.requireNonNull(task.getResult()).exists()) {
69                      //make the users document
70                      manageDBAddUserDocument(userID, user);
71                      //go to details activity
72                      Intent intent = new Intent(logInActivity.this, detailsActivity.class);
73                      startActivity(intent);
74                      finish();
75                  } else {
76                      //go to home activity
77                      Intent intent = new Intent(logInActivity.this, homeActivity.class);
78                      startActivity(intent);
79                      finish();
80                  }
81              }
82          });
83      }
84      //method to make a new document for a user
85      private void manageDBAddUserDocument(String userID, FirebaseUser user) {
86          //initialise hashmap
87          Map<String, Object> userData = new HashMap<>();
88          String name = user.getDisplayName();
89          ArrayList <Double> ratings = new ArrayList<>();
90          ratings.add(3.5);
91          userData.put("name", name);
92          userData.put("scannedBooks", Collections.emptyList());
93          userData.put("loanedBooks", Collections.emptyList());
```

```
94          userData.put("borrowedBooks", Collections.emptyList());
95          userData.put("readingList", Collections.emptyList());
96          userData.put("userRating", ratings);
97          userData.put("dateOfBirth", "");
98          userData.put("yearGroup", "");
99          userData.put("formRoom", "");
100         userData.put("transactionHistory", Collections.emptyList());
101         //add user data
102         db.collection("users").document(userID)
103                 .set(userData)
104                 .addOnCompleteListener(new OnCompleteListener<Void>() {
105                     @Override
106                     public void onComplete(@NonNull Task<Void> task) {
107                     }
108                 })
109                 .addOnFailureListener(new OnFailureListener() {
110                     @Override
111                     public void onFailure(@NonNull Exception e) {
112                     }
113                 });
114     }
115 }
```

For the purposes of testing I will print the activity names "homeActivity" or "detailsActivity" when an intent is made is to go to them, this is because those activities are not yet made. I will also provide a link[1] to the recording in the footer. Given that the process of sign-up is quite linear, i.e there are no branches of selection besides where to go after login, I will be evaluating whether the activity is stable and if data is added to Firebase, displayed in the Firebase Console.

```
1 >>> V/FA: onActivityCreated
2 >>> V/FA: Connecting to remote service
3 >>> I/System.out: I have gone to detailsActivity
```

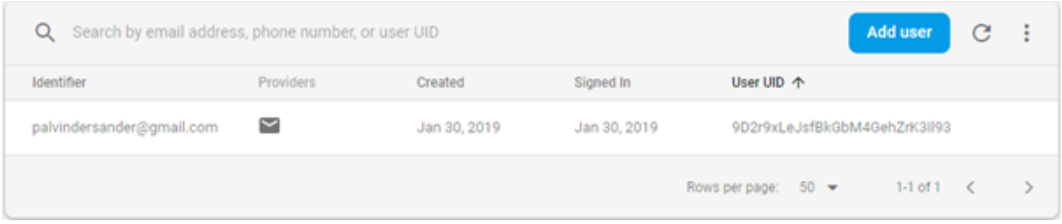


Figure 3.10: Firebase User

[1] $imp1TEST14 - 18.webm$ and $imp1TEST7 - 13.webm$

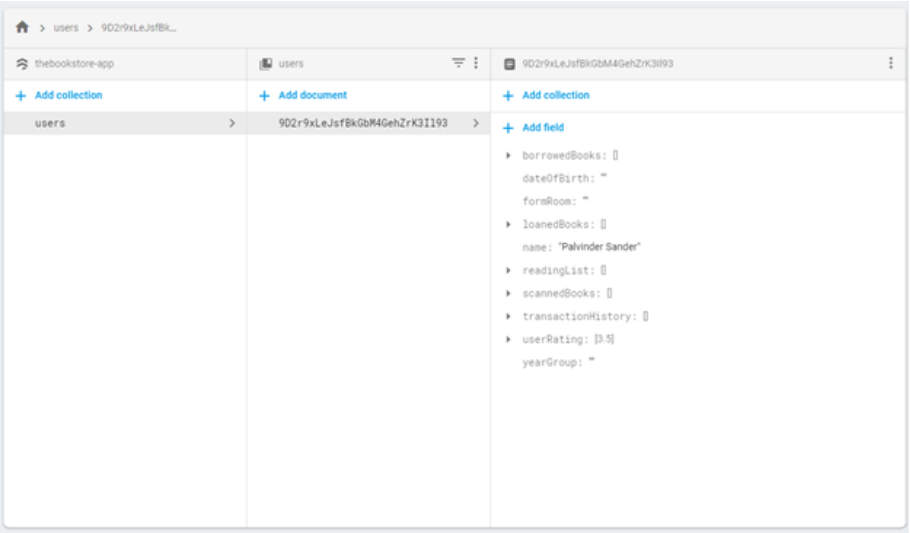


Figure 3.11: Firebase Console

With these successful results, I can now continue to the next stage of the log-in implementation: details activity.

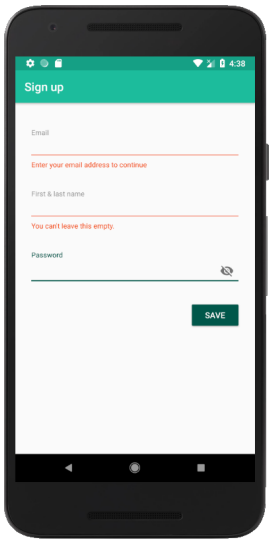


Figure 3.12: Log-In Interface

As can be seen above, the personal data that is input in the log-in UI is name, email and password. I require more data about the user and therefore I must create my own activity to collect this. I must first set about creating the UI. To do this I must consider what data I want the user to input and how I will store it:

1. form-room, String, text input
2. year-group, Integer, text input
3. date-of-birth, String, date input mm/dd/yyyy

The first two should be simple, it is the third that I may find trouble implementing. Within the IDE there is the option of adding a datePicker object that allows for the user to use 3 scroll wheels to select a date. Each value in the wheel can then be queried. I would then have to add "/" (slashes) between each value to create a string that is the input date. The corressponding XML code is:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:padding="5dp"
    tools:context=".detailsActivity">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:contentDescription="@string/app_name"
        android:scaleType="center"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@mipmap/ic_launcher_foreground" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:layout_weight="0.5"
        android:text="@string/enter_your_date_of_birth"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary" />
    <DatePicker
        android:id="@+id/datePicker"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:layout_weight="0.5"
        android:calendarViewShown="false"
        android:datePickerMode="spinner" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:text="@string/enter_these_details"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary" />
    <EditText
        android:id="@+id/yearGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginRight="100dp"
        android:layout_weight="0.5"
        android:autofillHints="@string/year_group"
        android:hint="@string/year_group"
        android:inputType="number"
        android:textAlignment="center"
        android:textSize="7.5pt" />
    <EditText
        android:id="@+id/formRoom"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:layout_marginLeft="100dp"
        android:layout_marginRight="100dp"
        android:layout_weight="0.5"
        android:autofillHints="@string/form_room"
        android:hint="@string/form_room"
        android:inputType="text"
        android:maxLength="5"
        android:textAlignment="center"
        android:textSize="7.5pt" />
    <Button
        android:id="@+id/submitData"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="75dp"
        android:layout_marginRight="75dp"
        android:background="@color/colorPrimary"
        android:onClick="submitData"
        android:text="@string/submit"
        android:textColor="@android:color/white" />
</LinearLayout>
```

The XML viewer displays the following preview:

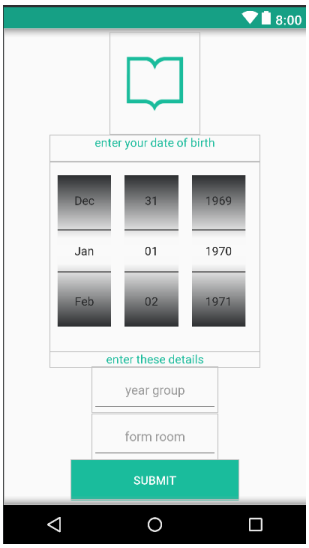


Figure 3.13: Details Activity

I can now proceed to implementing the code:

```
package com.bookstore.palvindersingh;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
```

```
16 import java.util.ArrayList;
17 public class detailsActivity extends AppCompatActivity {
18     //initialise attributes
19     private DatePicker datePicker;
20     private EditText yearGroup;
21     private EditText formRoom;
22     private FirebaseFirestore db;
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_details);
27         //initialise firestore connection
28         db = FirebaseFirestore.getInstance();
29         //initialise ui components
30         datePicker = findViewById(R.id.datePicker);
31         yearGroup = findViewById(R.id.yearGroup);
32         formRoom = findViewById(R.id.formRoom);
33     }
34     //method to submit input data
35     public void submitData(View v) {
36         //validate data
37         final String date = datePicker.getDayOfMonth() + "/" + datePicker.getMonth() + "/" +
           ↪  datePicker.getYear();
38         ArrayList validation = validate();
39         if (validation.get(0) == "f") {
40             Toast.makeText(getApplicationContext(), validation.get(1).toString(),
               ↪  Toast.LENGTH_SHORT).show();
41             return;
42         }
43         final String yg = yearGroup.getText().toString();
44         final String fr = formRoom.getText().toString();
45         FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
46         assert user != null;
47         String userID = user.getUid();
48         //add data to firestore
49         final DocumentReference userData = db.collection("users").document(userID);
50         userData.update("dateOfBirth", date)
51                 .addOnSuccessListener(new OnSuccessListener<Void>() {
52                     @Override
53                     public void onSuccess(Void aVoid) {
54                         userData.update("yearGroup", yg)
55                                 .addOnSuccessListener(new OnSuccessListener<Void>() {
56                                     @Override
57                                     public void onSuccess(Void aVoid) {
58                                         userData.update("formRoom", fr)
59                                                 .addOnSuccessListener(new OnSuccessListener<Void>()
                                                   ↪  {
60                                                     @Override
61                                                     public void onSuccess(Void aVoid) {
62                                                         Intent intent = new
                                                           ↪  Intent(detailsActivity.this,
                                                           ↪  homeActivity.class);
63                                                         startActivity(intent);
64                                                         finish();
65                                                     }
66                                                 })
67                                                 .addOnFailureListener(new OnFailureListener() {
68                                                     @Override
69                                                     public void onFailure(@NonNull Exception e) {
70                                                     }
71                                                 });
```

```
                                    }
                                })
                                .addOnFailureListener(new OnFailureListener() {
                                    @Override
                                    public void onFailure(@NonNull Exception e) {
                                    }
                                });
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                    }
                });
    }
    //method to validate data
    private ArrayList validate() {
        ArrayList result = new ArrayList<>();
        try {
            Integer yg = Integer.parseInt(yearGroup.getText().toString());
            if (yg > 6 && yg < 14) {
                String fr = formRoom.getText().toString();
                if ((fr.length() >2) && (fr.length() < 4)){
                    result.add("t");
                    result.add("valid");
                    return result;
                }else {
                    result.add("f");
                    result.add("form room invalid");
                    return result;
                }
            } else {
                result.add("f");
                result.add("year group must be between 7 and 13");
                return result;
            }
        } catch (Exception e) {
            result.add("f");
            result.add("year group must be an int");
            return result;
        }
    }
```

Now I can test the whole process, start till finish. I will provide a video link[2] in the footer for this. As I have done before, I will output "I am going to homeActivity", in the form of a Toast, instead of actually running the intent as the home activity screen has not yet been created. When I ran the test initially I was given the unexpected result of a toast that gave me the error message "form room invalid", for a valid piece of data. No errors were output to the console thus I can confirm that It is not a syntax error, as this would raise an error when in the program was in the Lexical Analysis stages of compilation. This must therefore mean there is a logical error in my code, somewhere in the *validate()* method. I will step through the code manually to see If i can find any errors there. With the input of "L1" the first check related to this input is it's size. The size of "L1" is 2, I can confirm this by printing the length out of the string.

```
String x = "L1";
Integer s = x.size();
System.out.println(s);
...
```

[2] $imp1TEST20 - 26.webm$

```
...
>>> 2
```

The code in *validate()* then checks if it's size is greater than 2 or less than 4. I have found the error. I was meant to accept values of length 2 or 3, i.e more than or equal to 2 and less than 4, instead the only valid length is 3. A quick addition of equal sign should fix this issue:

```
if ((fr.length() >=2) && (fr.length() < 4)){
```

As can now be seen in the footage and the screen-shots below, my application was able to successfully take the inputs, parse those necessary and add it to Firebase. It should be noted that my app goes directly to the details activity as I have already created an account using the app but without any added details thus the app carries out the correct selective statements in the splash screen to take me to details activity.

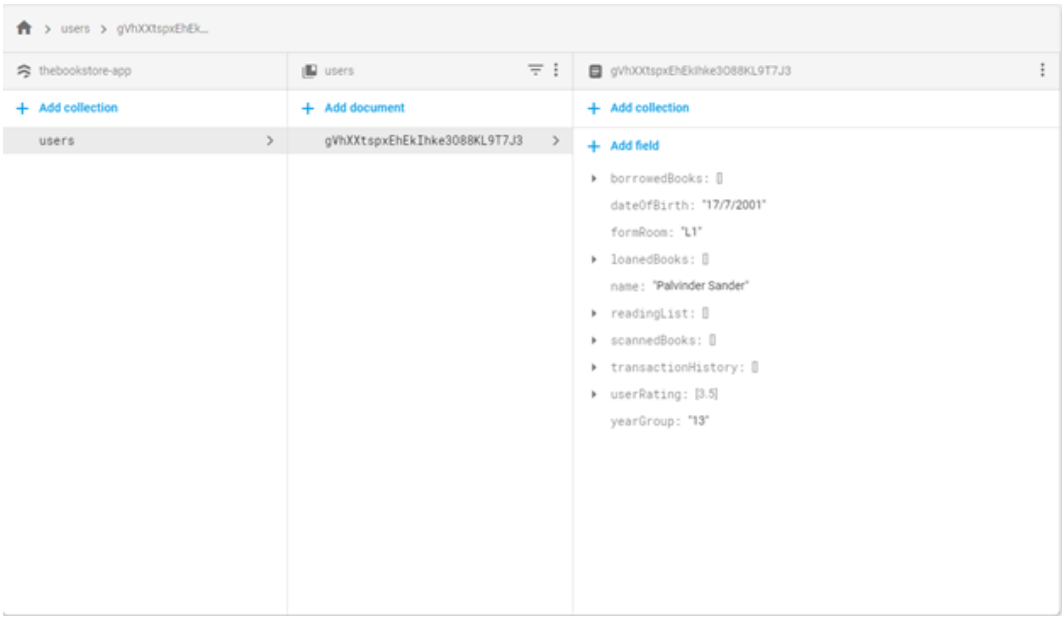


Figure 3.14: Firebase Console

3.8 No Network

As part of the specification for this activity, I must be able swipe down in order to refresh (run a method) the page. This is a UI component that I am completely unfamiliar with but fortunately is very easy to implement. It required me to add a *SwipeRefreshLayout* and then interact with this by instantiating a class with type *SwipeRefreshLayout* then calling a method *setOnRefreshListener()* in which I pass a *OnRefreshListener* into. This parameter requires an override method called *onRefresh()* to be declared in which the code to be executed should be input. The completed XML file and Java class implement these features:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/swipeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".noNetworkActivity">
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:contentDescription="@string/app_name"
            android:scaleType="center"
            android:scaleX="3"
            android:scaleY="3"
            android:src="@drawable/ic_signal_wifi_off_24dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="32dp"
            android:layout_marginEnd="8dp"
            android:fontFamily="@font/arimo"
            android:text="@string/no_network"
            android:textColor="@color/colorPrimary"
            android:textSize="25sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/imageView2" />
        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
```

```
            android:fontFamily="@font/arimo"
            android:text="@string/swipe_down_to_refresh"
            android:textColor="@color/colorPrimary"
            android:textSize="12sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView2" />
    </android.support.constraint.ConstraintLayout>
</android.support.v4.widget.SwipeRefreshLayout>
```

The user-interface for this activity is displayed to look like:

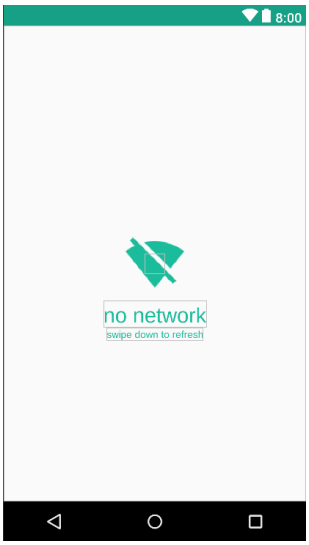


Figure 3.15: No Network Activity

```
package com.bookstore.palvindersingh;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;
public class noNetworkActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_no_network);
        //initialise ui components
        final SwipeRefreshLayout mySwipeRefreshLayout = findViewById(R.id.swipeLayout);
        //on page swipe listener
        mySwipeRefreshLayout.setOnRefreshListener(
                new SwipeRefreshLayout.OnRefreshListener() {
                    @Override
                    public void onRefresh() {
                        //check for internet
                        if (checkInternet()) {
                            //go to main activity
                            Intent intent = new Intent(noNetworkActivity.this, mainActivity.class);
                            startActivity(intent);
                            finish();
```

```
28                          } else {
29                              //prompt the user of no connection
30                              Toast.makeText(getApplicationContext(), "no internet connection",
                                ↪  Toast.LENGTH_SHORT).show();
31                              mySwipeRefreshLayout.setRefreshing(false);
32                          }
33                      }
34                  }
35          );
36      }
37      //method to check for internet
38      private Boolean checkInternet() {
39          ConnectivityManager connMgr = (ConnectivityManager)
            ↪  getSystemService(Context.CONNECTIVITY_SERVICE);
40          NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
41          return (networkInfo != null && networkInfo.isConnected());
42      }
43  }
```

To test this screen, I cleared the data for the app on the emulated device so that when I load it up, there will be no record of a user that is signed in. Given that my internet is off, I should then be taken to this activity where I will then turn on my internet refresh and be taken to the splash screen. A video recording of this is provided in the link[3] in the footer. Throughout the process no errors were returned and nor did any unexpected events occur.

[3] $imp1TEST5 - 6.webm$

3.9 Home

As part of my design, I discussed how to implement a navigation bar[9] and bottom navigation[10] in terms of methods and classes not how it is done in XML. Research into their corresponding documentation led me through this process.

Navigation Bar - For this I must declare an action bar XML layout in a new directory called menu:

```
<?xml version="1.0" encoding="utf-8"?>

<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_logout"
        android:title="@string/logoutActionBar"
        app:showAsAction="never" />
    <item
        android:id="@+id/action_restart"
        android:title="@string/restart"
        app:showAsAction="never" />
</menu>
```

I decided to add the restart button as a development feature. It is will make it easier for me to reload the app without having to leave it and close it down from the multi-tasking interface on the phone. The XML output the following preview:

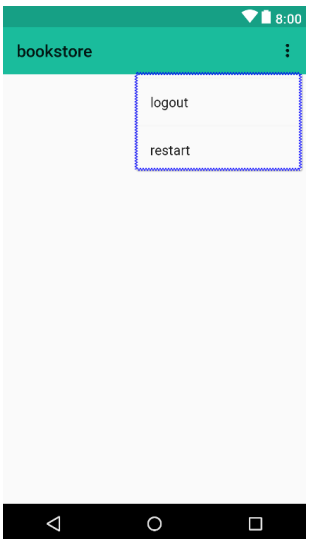


Figure 3.16: Menu Navigation

The only other thing to do here is to program what actions I want to do and add it into the home activity XML as a Toolbar component.

Bottom Navigation - This UI element is a bit more challenging to implement. As before I must declare an XML layout for in the menu directory:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/action_home"
        android:icon="@drawable/ic_home_24dp"
```

```
        android:title="@string/home" />
    <item
        android:id="@+id/action_books"
        android:icon="@drawable/ic_library_books_24dp"
        android:title="@string/books" />
    <item
        android:id="@+id/action_search"
        android:icon="@drawable/ic_search_24dp"
        android:title="@string/search" />
</menu>
```

I must then declare this in the home activity XML file as a BottomNavigationView component. I previously discussed in design stages, this form of navigation uses fragments, modular section of an activity, which has its own lifecycle and receives its own input events. Fragments have their own classes and layouts as do activities. Fortunately, my IDE can generate these for me. I did this for 3 fragments: home, books and search. After doing that I must then declare the home Activity class as implementing the 3 fragments. This technical term "implements" means the use of an interface. An interface is a class that defines methods and attributes but does not actually add content to them, it can be thought of a structure that can be 'implemented' to remind the programmer to declare certain methods and attributes. In this case the implementation of the fragments is actually an implementation of a pre-defined attribute of the fragment, which is an interface. I am guessing this is for the abstracted system behind android applications to deal with such features. My resulting code is as follows:

Fragment XML - For the sake of being able to read this documentation, I have only added one of the layouts. The other two are exactly the same but with the names *.booksFragment* and *.searchFragment*.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/AppTheme"
    tools:context=".homeFragment">

</FrameLayout>
```

Fragment Classes - Again, for the sake of being able to read this documentation, I have only added one of the classes. The other two are exactly the same but with the class names *booksFragment* and *searchFragment*.

```
package com.bookstore.palvindersingh;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;

public class homeFragment extends Fragment {

```

```
17      //initialise boilerplate attributes and methods
18      private static final String ARG_PARAM1 = "param1";
19      private static final String ARG_PARAM2 = "param2";
20
21      private OnFragmentInteractionListener mListener;
22      public homeFragment() {
23      }
24      public static homeFragment newInstance(String param1, String param2) {
25          homeFragment fragment = new homeFragment();
26          Bundle args = new Bundle();
27          args.putString(ARG_PARAM1, param1);
28          args.putString(ARG_PARAM2, param2);
29          fragment.setArguments(args);
30          Log.e("home", "fragment newInstance");
31          return fragment;
32      }
33      @Override
34      public void onCreate(Bundle savedInstanceState) {
35          super.onCreate(savedInstanceState);
36          if (getArguments() != null) {
37              String mParam1 = getArguments().getString(ARG_PARAM1);
38              String mParam2 = getArguments().getString(ARG_PARAM2);
39          }
40      }
41      @Override
42      public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
43                               Bundle savedInstanceState) {
44          Log.e("home", "fragment onCreateView");
45          View view = inflater.inflate(R.layout.fragment_home, container, false);
46                  return view;
47      }
48      public void onButtonPressed(Uri uri) {
49          if (mListener != null) {
50              mListener.onFragmentInteraction(uri);
51          }
52      }
53      @Override
54      public void onAttach(Context context) {
55          super.onAttach(context);
56          if (context instanceof OnFragmentInteractionListener) {
57              mListener = (OnFragmentInteractionListener) context;
58          } else {
59              throw new RuntimeException(context.toString()
60                      + " must implement OnFragmentInteractionListener");
61          }
62      }
63      @Override
64      public void onDetach() {
65          super.onDetach();
66          mListener = null;
67      }
68      public interface OnFragmentInteractionListener {
69          void onFragmentInteraction(Uri uri);
70      }
71  }
```

Home Activity XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".homeActivity">
    <android.support.v7.widget.Toolbar
        android:id="@+id/my_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />
    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="2"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />
    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_weight="0"
        android:background="?android:attr/windowBackground"
        app:menu="@menu/bottom_nav" />
</LinearLayout>
```

Home Activity Class

```
package com.bookstore.palvindersingh;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import com.firebase.ui.auth.AuthUI;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
public class homeActivity extends AppCompatActivity implements
↪  homeFragment.OnFragmentInteractionListener, booksFragment.OnFragmentInteractionListener,
↪  searchFragment.OnFragmentInteractionListener {
    //method to manage fragments for bottom nav
    private final BottomNavigationView.OnNavigationItemSelectedListener
    ↪  mOnNavigationItemSelectedListener
            = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            Fragment selectedFragment;
            switch (item.getItemId()) {
                case R.id.action_home:
```

```
26                      FragmentTransaction transaction =
                        ↪  getSupportFragmentManager().beginTransaction();
27                      selectedFragment = homeFragment.newInstance("home", "fragment");
28                      transaction.replace(R.id.content, selectedFragment);
29                      transaction.commit();
30                      return true;
31                  case R.id.action_books:
32                      FragmentTransaction transaction2 =
                        ↪  getSupportFragmentManager().beginTransaction();
33                      selectedFragment = booksFragment.newInstance("books", "fragment");
34                      transaction2.replace(R.id.content, selectedFragment);
35                      transaction2.commit();
36                      return true;
37                  case R.id.action_search:
38                      FragmentTransaction transaction3 =
                        ↪  getSupportFragmentManager().beginTransaction();
39                      selectedFragment = searchFragment.newInstance("search", "fragment");
40                      transaction3.replace(R.id.content, selectedFragment);
41                      transaction3.commit();
42                      return true;
43              }
44              return false;
45          }
46      };
47      @Override
48      protected void onCreate(Bundle savedInstanceState) {
49          super.onCreate(savedInstanceState);
50          setContentView(R.layout.activity_home);
51          //initialise toolbar
52          Toolbar myToolbar = findViewById(R.id.my_toolbar);
53          setSupportActionBar(myToolbar);
54          //set up first fragment
55          FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
56          transaction.replace(R.id.content, booksFragment.newInstance("first", "fragment"));
57          transaction.commit();
58          //initialise bottom navigation
59          BottomNavigationView navigation = findViewById(R.id.navigation);
60          navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
61      }
62      //initialise toolbar options
63      public boolean onCreateOptionsMenu(Menu menu) {
64          getMenuInflater().inflate(R.menu.action_bar, menu);
65          return true;
66      }
67      //initialise toolbar actions
68      @Override
69      public boolean onOptionsItemSelected(MenuItem item) {
70          switch (item.getItemId()) {
71              case R.id.action_logout:
72                  logoutUser();
73                  return true;
74              case R.id.action_restart:
75                  Intent intent = new Intent(homeActivity.this, mainActivity.class);
76                  startActivity(intent);
77                  finish();
78              default:
79                  return super.onOptionsItemSelected(item);
80
81          }
82      }
83      //method to log current user out
```

```
84      private void logoutUser() {
85          AuthUI.getInstance()
86                  .signOut(this)
87                  .addOnCompleteListener(new OnCompleteListener<Void>() {
88                      public void onComplete(@NonNull Task<Void> task) {
89                          Toast.makeText(getApplicationContext(), "logged out",
                            ↪  Toast.LENGTH_SHORT).show();
90                          Intent intent = new Intent(homeActivity.this, mainActivity.class);
91                          startActivity(intent);
92                          finish();
93                      }
94                  });
95      }
96  }
```

Floating Action Button - I must now edit the home fragment to implement my previously designed floating action button. This is fairly simple as it only requires the addition of some XML and 2 lines of code the in *onCreateView()* method in it's class.

```
1   <com.bookstore.palvindersingh.networkFAB
2       android:id="@+id/floating_action_button"
3       android:layout_width="wrap_content"
4       android:layout_height="wrap_content"
5       android:layout_gravity="bottom|end"
6       android:layout_margin="16dp"
7       app:fabSize="normal"
8       app:srcCompat="@drawable/ic_add_to_photos_white_24dp" />
```

```
1   //initialise add book floating action button
2   networkFAB addBookFAB = view.findViewById(R.id.floating_action_button);
3   //on click listener
4   View.OnClickListener addClick = new View.OnClickListener() {
5       @Override
6       public void onClick(View v) {
7           Intent intent = new Intent(getActivity(), scanBookActivity.class);
8           startActivity(intent);
9       }
10  };
```

Testing - Now that I have implemented all aspects of the class I can now proceed to testing it. I will first load the application, logged in, to see if the activity loads. The following error was returned:

```
1   >>> Class 'homeActivity' must either be declared abstract or implement abstract method
    ↪  'onFragmentInteraction(Uri)' in 'OnFragmentInteractionListener'
```

I have no idea what this means. It does not mention any specific lines and thus I doubt is due to a syntax error. Given that it does not compile in the first place, it cannot be runtime or logical error. It must therefore be some sort of implementation issue. I Googled this error and found a relevant article[11]. The question posed on this forum has absolutely nothing to do with my implementation but I still very helpful. The problem detailed within it is the use of a button without the definition of an override method required to interact with it. I am guessing that my error message means something similar, I have note implemented the method described. Given that I have declared the class as implementing various interfaces, it has become polymorphic in nature. This means that I can declare methods that replace or add to methods in parent/interface

classes. I think that the method *onFragmentInteraction(Uri)* should also be polymorphic, i.e be an override method. I consequently added this method to homeActivty. Although, I still do not see the point in having to do this, the method will be empty and not return or execute anything; it is as good as not existing at all. Perhaps it is a requirement of the back-end system.

```
1 @Override
2 public void onFragmentInteraction(Uri uri) {
3 }
```

My program now functions correctly. All of the menu buttons work and correctly transition, the toolbar buttons all function and the floating action button correctly works in reference to my networkFAB class. The link[4] in the footer displays this.

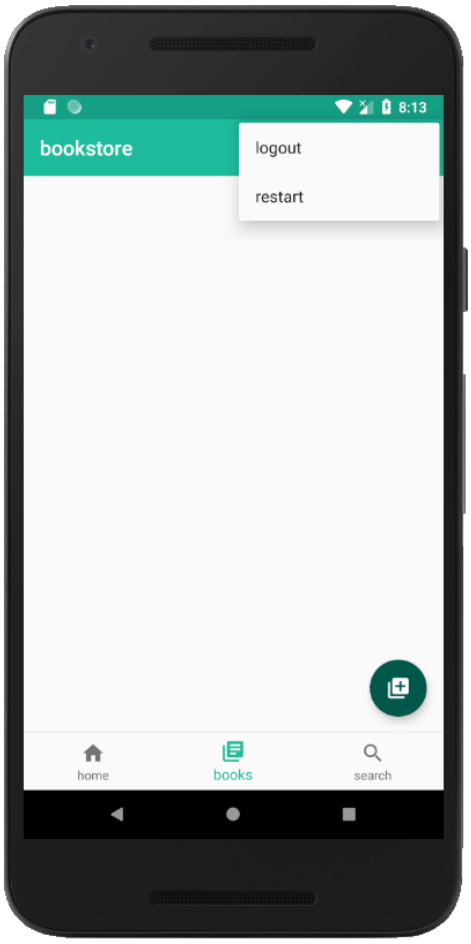


Figure 3.17: Home Activity

[4] $imp1TEST27 - 31.webm$

3.10 ISBN Scanner

One of the fundamental features of this application is to be able to scan books and given that they exist within the Google Books database, add them to their account. This is a complicated procedure that I can fortunately use the library Firebase Ml-Kit to abstract complicated machine learning structures away. However, instead of blindly implementing it I will first study how it works behind the scenes.

Firebase ML-Kit First off, Why use Machine Learning? Before even considering scanning a Barcode, lets take the process of scanning a single digit. To humans this seems easy, we see a number, we recall what it corresponds to and there you go, the number. This has been made possible of millions of years of evolution and natural optimisations. Now let's take a linear approach to breaking this down for a computer to interpret. We could convert an image of each number into binary and then save this, then whenever we have to identify a number compare it with the saved binaries. If it matches one of them, that is the number. But of course the way in which we write numbers varies drastically thus this process cannot be used. It quickly becomes apparent that simple intuitions about how we recognise digits, such as an 8 being two circle on top of each other, to be difficult the implement algorithmically. Machine Learning approaches the problem in a different way. The use of Machine Learning or otherwise known as artificial intelligence is built upon a structure called a *Neural Network*. The origin of such structures date back to the 1940s, with the goal of emulating the human brain, as the name infers. By this I do not mean completely recreating a digital brain but rather creating a statistical model that has the ability to produce good predictions of solutions given data of on previously solved problems. The process of optimisation by feeding the system data is known as training. The most fundamental part of a neural network is a neuron or node. There are two main types or neurons, Perceptrons and Sigmoids. Both take multiple inputs and produce a single output. Both also have weights associated with each input that determine the importance of it. The whole process of "learning" involves adjusting these weights so that they produce a more accurate outcome. The problem comes with changing these weights, a change in weight for Perceptron can cause a large change in the output. Sigmoids use a function called the *sigmoid function,* $\delta(z) = \frac{1}{1+e^{-x}}$, to ensure that a small change in weights results in a small change in the output. In this manner, Sigmoids can be used to improve the network while maintaining it's original integrity. As a result this is the type of neuron that is most commonly used to date. This, simplified, definition of neural networks can then be built upon further to create a form that is able to classify images, *Convolution Neural Networks*. These networks have neurons that receive an input as a matrix and then performs various mathematical operations to it, in the end outputting a probability of the image being a certain class. It essentially does this by decomposing an image and identifying the most important or valuable parts then repeating this process until only key features that match a classifier is found. In the case of Barcode Scanning, the neural network powering the features is able to successfully find the region of the image in which the Barcode exists, if it does, then identify features and interpret them to produce the correct corresponding value. This process can be quite computationally expensive but fortunately vasts amounts of data is available for training and thus the system has been optimised to be extremely effective and efficient.

Implementation I can now start to implement this knowledge. For this activity, I will be using 2 main libraries, Camera API and Firebase, both of which I have documented in design stages. To recap, The Camera API will be implemented by creating a UI element that is of the type Camera View. I will use the default settings as described in it's documentation[12]. I will then use various associated methods to capture an image. After converting it to a bitmap I will then declare an options, image and detector class that will work in conjunction with one another to find a Barcode. The XML for this activity is:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.constraint.ConstraintLayout
  ↪ xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
```

```
6       android:layout_height="match_parent"
7       tools:context=".scanBookActivity">
8       <android.support.v7.widget.Toolbar
9           android:id="@+id/my_child_toolbar"
10          style="@style/toolbarTheme"
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:theme="@style/AppTheme"
14          app:titleTextColor="@color/white" />
15      <com.otaliastudios.cameraview.CameraView
16          android:id="@+id/camera"
17          android:layout_width="match_parent"
18          android:layout_height="747dp"
19          android:keepScreenOn="true"
20          app:cameraAudio="off"
21          app:cameraJpegQuality="25"
22          app:cameraVideoQuality="max480p"
23          app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar" />
24      <TextView
25          android:id="@+id/isbn"
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:layout_marginStart="8dp"
29          android:layout_marginTop="8dp"
30          android:layout_marginEnd="8dp"
31          android:padding="10dp"
32          android:text="@string/isbn"
33          android:textColor="@color/white"
34          app:layout_constraintEnd_toEndOf="parent"
35          app:layout_constraintStart_toStartOf="parent"
36          app:layout_constraintTop_toBottomOf="@+id/view" />
37      <com.bookstore.palvindersingh.networkFAB
38          android:id="@+id/scan"
39          android:layout_width="wrap_content"
40          android:layout_height="wrap_content"
41          android:layout_margin="16dp"
42          android:layout_marginEnd="8dp"
43          android:layout_marginBottom="8dp"
44          android:clickable="true"
45          android:focusable="true"
46          app:fabSize="normal"
47          app:layout_constraintBottom_toBottomOf="parent"
48          app:layout_constraintEnd_toEndOf="parent"
49          app:srcCompat="@drawable/ic_add_a_photo_24dp" />
50      <android.support.design.widget.FloatingActionButton
51          android:id="@+id/testButton"
52          android:layout_width="wrap_content"
53          android:layout_height="wrap_content"
54          android:layout_margin="16dp"
55          android:layout_marginBottom="8dp"
56          android:clickable="true"
57          android:focusable="true"
58          app:backgroundTint="@color/fui_bgTwitter"
59          app:fabSize="mini"
60          app:layout_constraintBottom_toTopOf="@+id/flash"
61          app:layout_constraintEnd_toEndOf="@+id/flash"
62          app:layout_constraintStart_toStartOf="@+id/flash"
63          app:srcCompat="@android:drawable/alert_light_frame" />
64      <android.support.design.widget.FloatingActionButton
65          android:id="@+id/manual"
66          android:layout_width="wrap_content"
```

```
67          android:layout_height="wrap_content"
68          android:layout_margin="16dp"
69          android:layout_marginBottom="8dp"
70          android:clickable="true"
71          android:focusable="true"
72          app:backgroundTint="@color/browser_actions_title_color"
73          app:fabSize="mini"
74          app:layout_constraintBottom_toTopOf="@+id/scan"
75          app:layout_constraintEnd_toEndOf="@+id/scan"
76          app:layout_constraintStart_toStartOf="@+id/scan"
77          app:srcCompat="@drawable/ic_edit_24dp" />
78      <android.support.design.widget.FloatingActionButton
79          android:id="@+id/flash"
80          android:layout_width="wrap_content"
81          android:layout_height="wrap_content"
82          android:layout_margin="16dp"
83          android:layout_marginStart="8dp"
84          android:layout_marginBottom="8dp"
85          android:clickable="true"
86          android:focusable="true"
87          app:backgroundTint="@color/fui_bgAnonymous"
88          app:fabSize="normal"
89          app:layout_constraintBottom_toBottomOf="parent"
90          app:layout_constraintStart_toStartOf="parent"
91          app:srcCompat="@drawable/ic_flash_on_24dp" />
92      <View
93          android:id="@+id/view"
94          android:layout_width="match_parent"
95          android:layout_height="150dp"
96          android:layout_marginStart="32dp"
97          android:layout_marginTop="8dp"
98          android:layout_marginEnd="32dp"
99          android:layout_marginBottom="8dp"
100         android:background="@drawable/isbn_bounds"
101         app:layout_constraintBottom_toBottomOf="parent"
102         app:layout_constraintEnd_toEndOf="parent"
103         app:layout_constraintStart_toStartOf="parent"
104         app:layout_constraintTop_toTopOf="@+id/my_child_toolbar" />
105     <TextView
106         android:layout_width="wrap_content"
107         android:layout_height="wrap_content"
108         android:layout_marginEnd="8dp"
109         android:fontFamily="@font/arimo"
110         android:padding="10dp"
111         android:text="Manual"
112         android:textColor="@color/white"
113         android:textSize="15sp"
114         app:layout_constraintBottom_toBottomOf="@+id/manual"
115         app:layout_constraintEnd_toStartOf="@+id/manual"
116         app:layout_constraintTop_toTopOf="@+id/manual" />
117     <TextView
118         android:id="@+id/textView3"
119         android:layout_width="wrap_content"
120         android:layout_height="wrap_content"
121         android:layout_marginEnd="8dp"
122         android:fontFamily="@font/arimo"
123         android:padding="10dp"
124         android:text="Scan"
125         android:textColor="@color/white"
126         android:textSize="15sp"
127         app:layout_constraintBottom_toBottomOf="@+id/scan"
```

```
128         app:layout_constraintEnd_toStartOf="@+id/scan"
129         app:layout_constraintTop_toTopOf="@+id/scan" />
130     <TextView
131         android:layout_width="wrap_content"
132         android:layout_height="wrap_content"
133         android:layout_marginStart="8dp"
134         android:fontFamily="@font/arimo"
135         android:padding="10dp"
136         android:text="Flash"
137         android:textColor="@color/white"
138         android:textSize="15sp"
139         app:layout_constraintBottom_toBottomOf="@+id/flash"
140         app:layout_constraintStart_toEndOf="@+id/flash"
141         app:layout_constraintTop_toTopOf="@+id/flash" />
142 </android.support.constraint.ConstraintLayout>
```

Generating the following UI:

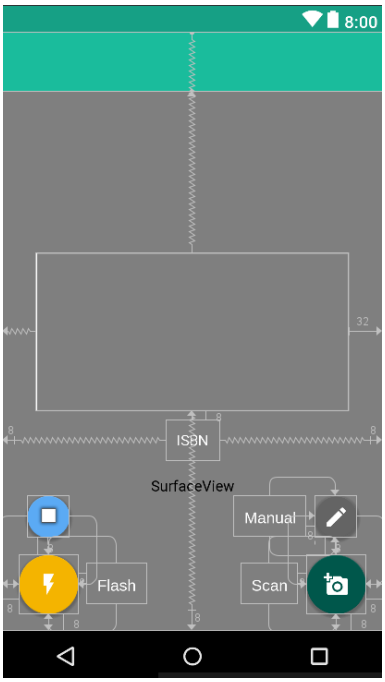


Figure 3.18: Scan Activity

I have taken the liberty here to add the small blue button for testing purposes. It will be tedious scan a book each time I want to test the following submission activity, the blue button will be used to take a pre defined ISBN and use it to pass onto the next activity, thus meaning I do not have to scan a book.

```
1  package com.bookstore.palvindersingh;
2  import android.content.DialogInterface;
3  import android.content.Intent;
4  import android.graphics.Bitmap;
5  import android.graphics.BitmapFactory;
6  import android.os.Bundle;
7  import android.support.annotation.NonNull;
8  import android.support.design.widget.FloatingActionButton;
9  import android.support.v7.app.ActionBar;
10 import android.support.v7.app.AlertDialog;
11 import android.support.v7.app.AppCompatActivity;
12 import android.support.v7.widget.Toolbar;
```

```
13 import android.text.InputType;
14 import android.view.View;
15 import android.widget.EditText;
16 import android.widget.TextView;
17 import android.widget.Toast;
18 import com.google.android.gms.tasks.OnFailureListener;
19 import com.google.android.gms.tasks.OnSuccessListener;
20 import com.google.android.gms.tasks.Task;
21 import com.google.firebase.ml.vision.FirebaseVision;
22 import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcode;
23 import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcodeDetector;
24 import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcodeDetectorOptions;
25 import com.google.firebase.ml.vision.common.FirebaseVisionImage;
26 import com.otaliastudios.cameraview.CameraListener;
27 import com.otaliastudios.cameraview.CameraView;
28 import com.otaliastudios.cameraview.Flash;
29 import java.io.ByteArrayInputStream;
30 import java.util.List;
31 public class scanBookActivity extends AppCompatActivity {
32     //initialise attributes
33     private int attempts = 0;
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_scan_book);
38         //initialise toolbar
39         Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
40         myChildToolbar.setTitle("scan a book");
41         setSupportActionBar(myChildToolbar);
42         ActionBar ab = getSupportActionBar();
43         assert ab != null;
44         ab.setDisplayHomeAsUpEnabled(true);
45         //initialise ui components
46         final TextView isbnData = findViewById(R.id.isbn);
47         final networkFAB scan = findViewById(R.id.scan);
48         FloatingActionButton flash = findViewById(R.id.flash);
49         final CameraView camera = findViewById(R.id.camera);
50         //initialise camera
51         camera.setLifecycleOwner(this);
52         //add camera picture taken listener
53         camera.addCameraListener(new CameraListener() {
54             @Override
55             public void onPictureTaken(byte[] picture) {
56                 //convert image to a bitmap
57                 Bitmap bitmap = convertToBitmap(picture);
58                 //instantiate options class
59                 FirebaseVisionBarcodeDetectorOptions options =
60                         new FirebaseVisionBarcodeDetectorOptions.Builder()
61                                 .setBarcodeFormats(
62                                         FirebaseVisionBarcode.FORMAT_EAN_13,
63                                         FirebaseVisionBarcode.FORMAT_EAN_8)
64                                 .build();
65                 //convert bitmap to firebase image
66                 FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(bitmap);
67                 //instantiate detector class
68                 FirebaseVisionBarcodeDetector detector =
                    ↪  FirebaseVision.getInstance().getVisionBarcodeDetector(options);
69                 //check for barcodes in the image
70                 Task<List<FirebaseVisionBarcode>> result = detector.detectInImage(image)
71                         .addOnSuccessListener(new OnSuccessListener<List<FirebaseVisionBarcode>>() {
72                             @Override
```

```
73                          public void onSuccess(List<FirebaseVisionBarcode> barcodes) {
74                              //check if there is more than one
75                              if (barcodes.size() > 1) {
76                                  //prompt the user that there are too many in the frame
77                                  Toast.makeText(getApplicationContext(), "too many barcodes",
                                    ↪  Toast.LENGTH_SHORT).show();
78                              } else {
79                                  //if there is only one barcode
80                                  if (barcodes.size() == 1) {
81                                      //set attempts to zero
82                                      attempts = 0;
83                                      //extract isbn value
84                                      String rawValue =
                                        ↪  String.valueOf(barcodes.get(0).getRawValue());
85                                      isbnData.setText(rawValue);
86                                      //prompt the user a barcode is found
87                                      Toast.makeText(getApplicationContext(), "barcode found",
                                        ↪  Toast.LENGTH_SHORT).show();
88                                      //go to add book activity
89                                      Intent intent = new Intent(scanBookActivity.this,
                                        ↪  addBookActivity.class);
90                                      //pass isbn value into intent
91                                      intent.putExtra("isbn", rawValue);
92                                      startActivity(intent);
93                                  } else {
94                                      //increment attempts
95                                      attempts += 1;
96                                      //prompt the user no barcode is found
97                                      Toast.makeText(getApplicationContext(), "no barcode found",
                                        ↪  Toast.LENGTH_SHORT).show();
98                                      if (attempts > 2) {
99                                          //create alert dialog for manual isbn input
100                                         AlertDialog.Builder builder = new
                                            ↪  AlertDialog.Builder(scanBookActivity.this);
101                                         final EditText input = new
                                            ↪  EditText(scanBookActivity.this);
102                                         input.setHint("enter an isbn");
103                                         input.setInputType(InputType.TYPE_CLASS_NUMBER);
104                                         builder.setView(input);
105                                         builder.setPositiveButton("OK", new
                                            ↪  DialogInterface.OnClickListener() {
106                                             @Override
107                                             public void onClick(DialogInterface dialog, int
                                                ↪  which) {
108                                                 String rawValue = input.getText().toString();
109                                                 Intent intent = new
                                                    ↪  Intent(scanBookActivity.this,
                                                    ↪  addBookActivity.class);
110                                                 intent.putExtra("isbn", rawValue);
111                                                 startActivity(intent);
112                                                 finish();
113                                             }
114                                         });
115                                         builder.setNegativeButton("Cancel", new
                                            ↪  DialogInterface.OnClickListener() {
116                                             @Override
117                                             public void onClick(DialogInterface dialog, int
                                                ↪  which) {
118                                                 attempts = 1;
119                                                 dialog.cancel();
120                                             }
```

```
121                                         });
122                                         builder.show();
123                                     }
124                                 }
125                             }
126                         }
127                     })
128                     .addOnFailureListener(new OnFailureListener() {
129                         @Override
130                         public void onFailure(@NonNull Exception e) {
131                             Toast.makeText(getApplicationContext(), "eRrOr OcCuRrEd",
                                ↪  Toast.LENGTH_SHORT).show();
132                         }
133                     });
134             }
135         });
136         //on click listener to take a picture
137         View.OnClickListener clickEvent = new View.OnClickListener() {
138             @Override
139             public void onClick(View v) {
140                 camera.capturePicture();
141             }
142         };
143         scan.setClickEvent(clickEvent);
144         //on click listener to turn flash on and off
145         flash.setOnClickListener(new View.OnClickListener() {
146             @Override
147             public void onClick(View v) {
148                 if (camera.getFlash() == Flash.TORCH) {
149                     camera.setFlash(Flash.OFF);
150                 } else {
151                     camera.setFlash(Flash.TORCH);
152                 }
153             }
154         });
155         //on click listener
156         FloatingActionButton manual = findViewById(R.id.manual);
157         manual.setOnClickListener(new View.OnClickListener() {
158             @Override
159             public void onClick(View v) {
160                 //go to add book manual activity
161                 Intent intent = new Intent(scanBookActivity.this, addBookManualActivity.class);
162                 startActivity(intent);
163             }
164         });
165         FloatingActionButton testButton = findViewById(R.id.testButton);
166         testButton.setOnClickListener(new View.OnClickListener() {
167             @Override
168             public void onClick(View v) {
169                 String rawValue = "1782944141";
170                 Intent intent = new Intent(scanBookActivity.this, addBookActivity.class);
171                 intent.putExtra("isbn", rawValue);
172                 startActivity(intent);
173             }
174         });
175     }
176     //method to convert byte array into a bitmap
177     private Bitmap convertToBitmap(byte picture) {
178         ByteArrayInputStream arrayInputStream = new ByteArrayInputStream(picture);
179         return BitmapFactory.decodeStream(arrayInputStream);
180     }
```

```
181 }
```

I then implemented the code, as seen above, running into a few road blocks along the way. Reviewing my algorithm design, I immediately saw two big gaps in the implementation, the first being, *how do I create a dialog box?* and the second *How do I transfer data between classes?*. The first was fairly easy gap to fill as a useful article[13] on the *AlertDialog* clearly explained the foundation of such a component. In short, I declared an AlertDialog object, populated it's interface with a label,, input and an OK/CANCEL button. After doing this I assigned the class an *onClickListener* to handle both the OK and CANCEL button clicks. This was all done programmatically, meaning no XML was required. Now onto passing data between classes. There were a few requirements I had to decide upon before finding a solution. The first being, I did not want to use a method that used any form of static attributes or methods. This would result in '*messy*' and not very android-friendly code as I would be going about a non-typical method and essentially ignoring the android activity life-cycle. Therefore, I had to find a way of transferring data via the intent itself as that is class that links the two activities. After reading some of the intent class documentation[14], I found a method called *putExtra*. It takes two parameters, on being the value name (similar to how a dictionary or hash-map stores data) and the value itself. It does not say what type of data is allowed thus I believed the primitive type string should be allowed at very least. As I previously mentioned, I have implemented a blue button for testing purposes. In doing this, I had to instantiate an *onClickListener* class for the button. All the object does is take a predefined ISBN and pass it into an intent, as discussed. The ISBN is from one of my own old GCSE books, a CGP English Revision Guide. I also how do find out how I can convert an array of bytes to a bitmap image. After researching the topic I found a simple implementation that solves the issue. It just uses native libraries in Java to convert it directly to a bitmap. Now that I have implemented the code I can begin to test it.

When my IDE started to compile the application the following error was returned indicating a syntax error occurred in the lexical analysis stages of compilation.

```
1 >>> error: incompatible types: byte[] cannot be converted to byte
2 >>> error: incompatible types: byte cannot be converted to byte[]
```

After some proof-reading of my code, I found that I incorrectly defined the parameter for the *convertToBitmap* method as a byte, when it should be a byte array. Thus the altered syntax yields:

```
1 private Bitmap convertToBitmap(byte[] picture) {...}
```

Now my application successfully compiles. I will not continue to test X parts of the activity:

1. Does it scan an ISBN correctly?
2. Does it allow for a Dialog?

Starting with the first, I will install the app on my phone and scan an isbn. In order to see whether it identifies an ISBN but also identify its data I will output the ISBN to the console. I will scan a book with the ISBN: 9781108412742. The following was output in the console (with the altered code below):

```
1 if (barcodes.size() == 1) {
2     //set attempts to zero
3     attempts = 0;
4     //extract isbn value
5     String rawValue = String.valueOf(barcodes.get(0).getRawValue());
6     isbnData.setText(rawValue);
7     System.out.println("BARCODE IS: " + rawValue);
8     //prompt the user a barcode is found
```

```
9       Toast.makeText(getApplicationContext(), "barcode found", Toast.LENGTH_SHORT).show();
10      //go to add book activity
11      Intent intent = new Intent(scanBookActivity.this, addBookActivity.class);
12      //pass isbn value into intent
13      intent.putExtra("isbn", rawValue);
14      startActivity(intent);
```

```
1   >>> BARCODE IS: 9781108412742
```

As can be seen, the output was as expected. Now testing whether a Dialog appears after 3 incorrect tries. I can do this from within the emulator. Again outputting the ISBN to the console once input. The Dialog screen correctly appears (excuse the awkward looking background, it is the emulators 'emulation' of a camera).

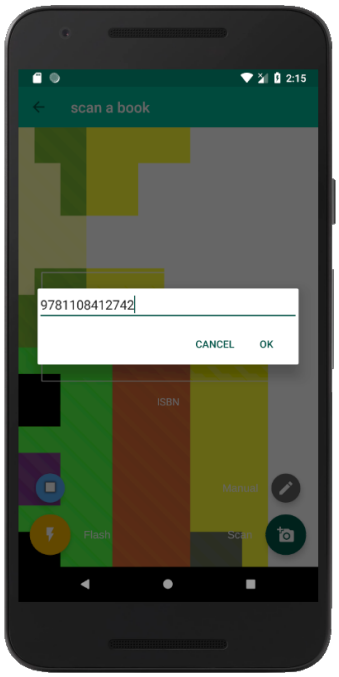


Figure 3.19: Scan Dialog

When clicking OK the following was output in the console (with the altered code):

```
1   public void onClick(DialogInterface dialog, int which) {
2       String rawValue = input.getText().toString();
3       System.out.println("BARCODE IS: " + rawValue);
4       Intent intent = new Intent(scanBookActivity.this, addBookActivity.class);
5       intent.putExtra("isbn", rawValue);
6       startActivity(intent);
7       finish();
8   }
```

```
1   >>> D/EGL_emulation: eglMakeCurrent: 0xe6ca2220: ver 3 1 (tinfo 0xe6cecb40)
2   >>> D/EGL_emulation: eglMakeCurrent: 0xe6ca2220: ver 3 1 (tinfo 0xe6cecb40)
3   >>> BARCODE IS: 9781108412742
```

It would be expected for me to test the navigation components but given that I have no created the linked activities yet, I will have to delay this till the end of prototype tests.

3.11 Book Submission

For this stage of implementation there will be 2 activities to implement, an automatic submission using the fetch meta data class and the manual submission activity. I will therefore split this implementation up into these two activities, starting with development on the automatic submission one.

3.11.1 Automatic Submission

Before I begin development there are a few areas of should first research. As we discussed in the scan activity, I am passing the value of the captured ISBN as a string via the Intent class. Thus I must research how this is done, specifically how do I retrieve the data that has been passed into the class. Secondly, I must research how to implement the fetch meta data class. In design stages I had briefly discussed an asynchronous task class (all computations relating to that class are done on a separate thread). Finally, It has come to my attention that I should implement some form of ACID rules or Transaction Processing given that on this stage I am writing to multiple documents.

First, I will find how to receive the passed ISBN value. The documentation that details how to pass it also described how to receive it, with the simple following syntax:

```
final String extraData = getIntent().getStringExtra("DATA NAME");
```

So for my implementation I should replace "DATA NAME" with "isbn".

Now onto the implementation of an async task. If we recall the following code from design:

```
public class className extends AsyncTask<String, Void, [DATA]>{
    @Override
    protected [DATA] doInBackground(String... extraData) {
        //call a method to execute
        return [DATA];
    }

    @Override
    protected void onProgressUpdate(Void... values) {
        super.onProgressUpdate(values);
    }

    @Override
    protected void onPostExecute([DATA] DATA){
        //do stuff
    }
}

...

new className().execute(DATA);
```

I have drawn a simple diagram to display how this will work for my application.

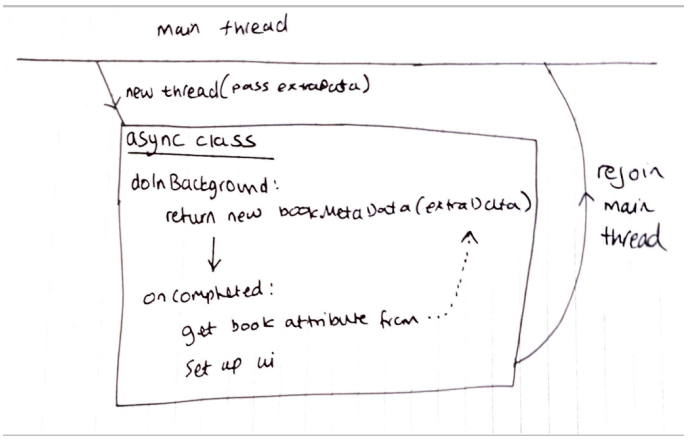


Figure 3.20: Async Task

As you can see my application will be running in the main thread. When the activity is started the async class should be instantiated using the parameter extraData, the data fetched from the intent i.e. the ISBN. Within the class I should return a bookMetaData class to the onPostExecute method, instantiating the class will in turn prompt the fetching of the books data from the web. After displaying this data within the UI the thread can then rejoin with the main, thus ending the process of fetching data.

Transaction processing is the process of taking multiple smaller operations and grouping the together in a single 'transaction' operation. If any of the smaller operations fails then the transaction as a whole must fail. The aim of transaction processing is to ensure that complex operations are not left partially complete by hardware or software failures. There are four properties of a successful transactions. Atomicity - Either every part of the transaction is successful or none is. Consistency - All parts of the transactions must be in line with the database rules; for example they do not create duplicate keys. Isolation - Ensure that even if different transactions are happening at the same time, the results are the same as if they were happening one after the other. Durability - Once a transaction has been successfully completed, its actions are not undone. The main aspect that I am interested in is Atomicity, as given that I will be changing multiple pieces of data, the users array of books and creating a new book document, it is important that if the process fails for any reason either all the data is written or none. If parts of the data were written and not all of it, this would lead to redundant or missing data, something that is a haven for bugs. Luckily after some research I found that Firebase[15] allows for this process. The syntax is simple but as we have seen before, very long thus I will leave it out of the documentation and implement directly into my code.

Now that I have researched and discussed these areas I should be good to start developing the UI and implementing the code. The layout designer created the following XML and displayed this UI (note that I have hidden some UI components as they will be made visible when data is found/ if data is found for them).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
   ↪  xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".addBookActivity">
8      <android.support.v7.widget.Toolbar
9          android:id="@+id/my_child_toolbar"
10         style="@style/toolbarTheme"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:theme="@style/AppTheme"
14         app:titleTextColor="@color/white" />
```

```
15      <ScrollView
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18          app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">
19          <LinearLayout
20              android:layout_width="match_parent"
21              android:layout_height="wrap_content"
22              android:orientation="vertical"
23              android:paddingLeft="50dp"
24              android:paddingRight="50dp">
25              <TextView
26                  android:layout_width="wrap_content"
27                  android:layout_height="wrap_content"
28                  android:layout_gravity="center"
29                  android:fontFamily="@font/arimo"
30                  android:padding="20dp"
31                  android:text="@string/is_this_your_book"
32                  android:textAllCaps="true"
33                  android:textColor="@color/colorPrimary"
34                  android:textSize="15sp"
35                  android:textStyle="bold" />
36              <ImageView
37                  android:id="@+id/bookImage"
38                  android:layout_width="125dp"
39                  android:layout_height="wrap_content"
40                  android:layout_gravity="center"
41                  android:adjustViewBounds="true"
42                  android:contentDescription="@string/books"
43                  android:scaleType="fitXY"
44                  android:visibility="gone" />
45              <LinearLayout
46                  android:layout_width="match_parent"
47                  android:layout_height="wrap_content"
48                  android:layout_marginTop="10dp"
49                  android:background="@color/browser_actions_bg_grey"
50                  android:divider="?android:listDivider"
51                  android:orientation="vertical"
52                  android:showDividers="middle">
53                  <TextView
54                      android:layout_width="match_parent"
55                      android:layout_height="wrap_content"
56                      android:elevation="10dp"
57                      android:fontFamily="@font/arimo"
58                      android:padding="10dp"
59                      android:text="About"
60                      android:textColor="#000"
61                      android:textSize="15sp" />
62                  <TextView
63                      android:id="@+id/isbn"
64                      android:layout_width="match_parent"
65                      android:layout_height="wrap_content"
66                      android:elevation="10dp"
67                      android:fontFamily="@font/arimo"
68                      android:padding="10dp"
69                      android:text="@string/isbn"
70                      android:textColor="@color/colorPrimaryDark"
71                      android:textSize="15sp"
72                      android:visibility="gone" />
73                  <TextView
74                      android:id="@+id/bookTitle"
75                      android:layout_width="match_parent"
```

```
76                      android:layout_height="wrap_content"
77                      android:fontFamily="@font/arimo"
78                      android:padding="10dp"
79                      android:text="@string/title"
80                      android:textAllCaps="true"
81                      android:textColor="@color/colorPrimaryDark"
82                      android:textSize="15sp"
83                      android:visibility="gone" />
84                  <TextView
85                      android:id="@+id/bookAuthors"
86                      android:layout_width="match_parent"
87                      android:layout_height="wrap_content"
88                      android:fontFamily="@font/arimo"
89                      android:padding="10dp"
90                      android:text="@string/authors"
91                      android:textAllCaps="true"
92                      android:textColor="@color/colorPrimaryDark"
93                      android:textSize="15sp"
94                      android:visibility="gone" />
95                  <TextView
96                      android:id="@+id/bookGenres"
97                      android:layout_width="match_parent"
98                      android:layout_height="wrap_content"
99                      android:fontFamily="@font/arimo"
100                     android:padding="10dp"
101                     android:text="@string/genres"
102                     android:textAllCaps="true"
103                     android:textColor="@color/colorPrimaryDark"
104                     android:textSize="15sp"
105                     android:visibility="gone" />
106             </LinearLayout>
107         </LinearLayout>
108     </ScrollView>
109     <com.bookstore.palvindersingh.networkFAB
110         android:id="@+id/correct"
111         android:layout_width="wrap_content"
112         android:layout_height="wrap_content"
113         android:layout_margin="16dp"
114         android:layout_marginEnd="8dp"
115         android:layout_marginBottom="8dp"
116         android:clickable="true"
117         android:focusable="true"
118         android:src="@drawable/ic_done_24dp"
119         app:backgroundTint="@color/fui_bgPhone"
120         app:fabSize="normal"
121         app:layout_constraintBottom_toBottomOf="parent"
122         app:layout_constraintEnd_toEndOf="parent" />
123     <android.support.design.widget.FloatingActionButton
124         android:id="@+id/wrong"
125         android:layout_width="wrap_content"
126         android:layout_height="wrap_content"
127         android:layout_margin="16dp"
128         android:layout_marginEnd="8dp"
129         android:layout_marginBottom="8dp"
130         android:clickable="true"
131         android:focusable="true"
132         android:src="@drawable/ic_close_24dp"
133         app:backgroundTint="@color/fui_bgEmail"
134         app:fabSize="normal"
135         app:layout_constraintBottom_toTopOf="@+id/correct"
136         app:layout_constraintEnd_toEndOf="parent" />
```

```
137     <TextView
138         android:layout_width="wrap_content"
139         android:layout_height="wrap_content"
140         android:layout_marginEnd="8dp"
141         android:fontFamily="@font/arimo"
142         android:padding="10dp"
143         android:text="No"
144         android:textColor="#000"
145         android:textSize="15sp"
146         app:layout_constraintBottom_toBottomOf="@+id/wrong"
147         app:layout_constraintEnd_toStartOf="@+id/wrong"
148         app:layout_constraintTop_toTopOf="@+id/wrong" />
149     <TextView
150         android:layout_width="wrap_content"
151         android:layout_height="wrap_content"
152         android:layout_marginEnd="8dp"
153         android:layout_marginBottom="8dp"
154         android:fontFamily="@font/arimo"
155         android:padding="10dp"
156         android:text="Yes"
157         android:textColor="#000"
158         android:textSize="15sp"
159         app:layout_constraintBottom_toBottomOf="@+id/correct"
160         app:layout_constraintEnd_toStartOf="@+id/correct" />
161 </android.support.constraint.ConstraintLayout>
```

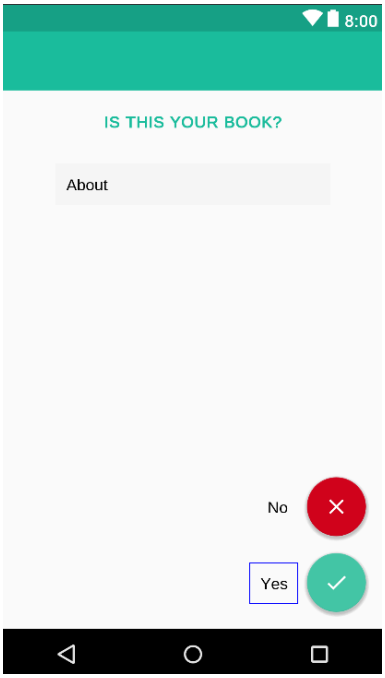


Figure 3.21: Submission UI

```
1  package com.bookstore.palvindersingh;
2  import android.content.Intent;
3  import android.os.AsyncTask;
4  import android.os.Bundle;
5  import android.support.annotation.NonNull;
6  import android.support.annotation.Nullable;
7  import android.support.design.widget.FloatingActionButton;
8  import android.support.v7.app.ActionBar;
9  import android.support.v7.app.AppCompatActivity;
10 import android.support.v7.widget.Toolbar;
11 import android.view.View;
12 import android.widget.ImageView;
13 import android.widget.TextView;
14 import android.widget.Toast;
15 import com.google.android.gms.tasks.OnFailureListener;
16 import com.google.android.gms.tasks.OnSuccessListener;
17 import com.google.firebase.auth.FirebaseAuth;
18 import com.google.firebase.auth.FirebaseUser;
19 import com.google.firebase.firestore.DocumentReference;
20 import com.google.firebase.firestore.FieldValue;
21 import com.google.firebase.firestore.FirebaseFirestore;
22 import com.google.firebase.firestore.Transaction;
23 import com.squareup.picasso.Picasso;
24 import java.util.ArrayList;
25 public class addBookActivity extends AppCompatActivity {
26     //book attribute
27     book book;
28     //convert array into a string with spaces in between each value
29     private String arrayToDisplayString(ArrayList data) {
30         String string = "";
31         for (int i = 0; i < data.size(); i++) {
32             string = string + data.get(i);
33             if (i != data.size() - 1) {
34                 string = string + " ";
35             }
36         }
37         return string;
38     }
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_add_book);
43         //initialise toolbar
44         Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
45         myChildToolbar.setTitle("submit a book");
46         setSupportActionBar(myChildToolbar);
47         ActionBar ab = getSupportActionBar();
48         assert ab != null;
49         ab.setDisplayHomeAsUpEnabled(true);
50         //initialise ui components
51         final TextView isbnText = findViewById(R.id.isbn);
52         final String extraData = getIntent().getStringExtra("isbn");
53         isbnText.setText(extraData);
54         //start async task to get metadata
55         new setData().execute(extraData);
56         //initialise wrong floating action button
57         FloatingActionButton wrong = findViewById(R.id.wrong);
58         //on click listener
59         wrong.setOnClickListener(new View.OnClickListener() {
60             @Override
```

```
61              public void onClick(View v) {
62                  //go to manual input activity
63                  Intent intent = new Intent(addBookActivity.this, addBookManualActivity.class);
64                  startActivity(intent);
65                  finish();
66              }
67          });
68          //initialise correct floating action button
69          networkFAB correct = findViewById(R.id.correct);
70          //on click listener
71          View.OnClickListener correctClick = new View.OnClickListener() {
72              @Override
73              public void onClick(View v) {
74                  //initialise ui component
75                  TextView bookTitle = findViewById(R.id.bookTitle);
76                  //check if data has loaded
77                  if (bookTitle.getText().equals("title")) {
78                      Toast.makeText(getApplicationContext(), "wait for data to load",
                        ↪  Toast.LENGTH_SHORT).show();
79                  } else {
80                      //add data to firestore
81                      final FirebaseFirestore db = FirebaseFirestore.getInstance();
82                      final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
83                      book.setOwnerReference(user.getUid());
84                      //transaction processing
85                      db.runTransaction(new Transaction.Function<Void>() {
86                          @Nullable
87                          @Override
88                          public Void apply(@NonNull Transaction transaction) {
89                              //add book to collection
90                              db.collection("books").add(book).addOnSuccessListener(new
                                ↪  OnSuccessListener<DocumentReference>() {
91                                  @Override
92                                  public void onSuccess(DocumentReference documentReference) {
93                                      //update user data
94                                      DocumentReference docRef =
                                        ↪  db.collection("users").document(user.getUid());
95                                      docRef.update("scannedBooks",
                                        ↪  FieldValue.arrayUnion(documentReference.getId()));
96                                  }
97                              });
98                              return null;
99                          }
100                     }).addOnSuccessListener(new OnSuccessListener<Void>() {
101                         @Override
102                         public void onSuccess(Void aVoid) {
103                             //go to home activity if successful
104                             Toast.makeText(getApplicationContext(), "book added",
                                ↪  Toast.LENGTH_SHORT).show();
105                             Intent intent = new Intent(addBookActivity.this, homeActivity.class);
106                             startActivity(intent);
107                             finish();
108                         }
109                     }).addOnFailureListener(new OnFailureListener() {
110                         @Override
111                         public void onFailure(@NonNull Exception e) {
112                             //go to scan book activity if not successful
113                             Toast.makeText(getApplicationContext(), "something unexpected occurred",
                                ↪  Toast.LENGTH_SHORT).show();
114                             Intent intent = new Intent(addBookActivity.this,
                                ↪  scanBookActivity.class);
```

```
115                             startActivity(intent);
116                             finish();
117                         }
118                     });
119                 }
120             }
121         };
122         //pass correctClick to correct class
123         correct.setClickEvent(correctClick);
124     }
125     //define setData async task
126     private class setData extends AsyncTask<String, Void, bookMetaData> {
127         @Override
128         protected bookMetaData doInBackground(String... extraData) {
129             //run bookMetaData class
130             return new bookMetaData(extraData);
131         }
132         @Override
133         protected void onProgressUpdate(Void... values) {
134             super.onProgressUpdate(values);
135         }
136         @Override
137         protected void onPostExecute(bookMetaData bookMetaData) {
138             //get book object
139             book meta = bookMetaData.getBook();
140             //set book attribute to meta
141             book = meta;
142             //check a book object exists
143             if (meta == null) {
144                 //go to home activity
145                 Toast.makeText(getApplicationContext(), "book does not exist",
                    ↪  Toast.LENGTH_SHORT).show();
146                 Intent intent = new Intent(addBookActivity.this, homeActivity.class);
147                 startActivity(intent);
148                 finish();
149                 return;
150             }
151             //set ui components to visible and change according text if it exists
152             TextView bookISBN = findViewById(R.id.isbn);
153             bookISBN.setVisibility(View.VISIBLE);
154             if (meta.getTitle() != null) {
155                 TextView bookTitle = findViewById(R.id.bookTitle);
156                 bookTitle.setText(meta.getTitle());
157                 bookTitle.setVisibility(View.VISIBLE);
158             }
159             if (meta.getAuthors() != null) {
160                 TextView bookAuthors = findViewById(R.id.bookAuthors);
161                 bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
162                 bookAuthors.setVisibility(View.VISIBLE);
163             }
164             if (meta.getGenres() != null) {
165                 TextView bookGenres = findViewById(R.id.bookGenres);
166                 bookGenres.setText(arrayToDisplayString(meta.getGenres()));
167                 bookGenres.setVisibility(View.VISIBLE);
168             }
169             if (meta.getImage() != null) {
170                 ImageView bookImage = findViewById(R.id.bookImage);
171                 Picasso.get().load(meta.getImage()).into(bookImage);
172                 bookImage.setContentDescription(meta.getImage());
173                 bookImage.setVisibility(View.VISIBLE);
174             }
```

```
175         }
176     }
177 }
```

After compilation, the following syntax error was returned:

```
1 >>> error line-144: incompatible types: String[] cannot be converted to String
```

As a result this must be in relation to this line:

```
1 return new bookMetaData(extraData);
```

Given that the error is in relation to a string it must be to do with the parameter extraData. To my knowledge I have no idea what I have done wrong. It does not seem to be a syntax error like a missing colon or something thus it must be something to do with my implementation. After searching the issue I found someone that has had a similar issue on StackOverflow[16]. It seems as though the class takes the parameter and then initialises it as an array, with my string being the first value in the array. Thus I should change the code to:

```
1 return new bookMetaData(extraData[0]);
```

Now that I have fixed this I shall recompile my code again. It worked, the error has gone however a new error has presented itself:

```
1 >>> method Intent.putExtra(String,Parcelable[]) is not applicable
2 >>> (argument mismatch; book cannot be converted to Parcelable[])
3 >>> method Intent.putExtra(String,Serializable) is not applicable
4 >>> (argument mismatch; book cannot be converted to Serializable)
```

It seems to be in relation to the book class. I have seen this terminology of before Parcelable and Serializable before. Both are a means of compressing a data structure for transmission. In my class I pass the book class between the client a Firebase however the error has not originated from such that library. The other place where I pass data between classes is the async class and the activity. This would explain why no specific library has been mentioned in the error log, async class is a native package. Another StackOverflow[17] forum seems to have solved my issue. I think that all I need to do is declare my book class as implementing the interface Serializable so that it can be passed between classes:

```
1 public class book implements Serializable {
```

I have now fixed all my compilation errors. I will no use the blue button implemented in the scan activity to test this activity. The book that should appear is a CGP GCSE English Revision Guide. The emulator correctly displays:

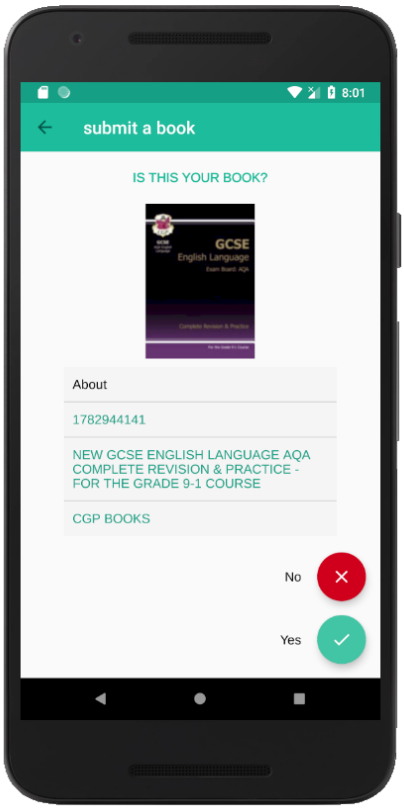


Figure 3.22: Submission UI

I have not created the manual submission activity yet thus the "No" button is not expected to function, as when clicked it should take the user to the manual submission activity. I can however test the Yes button. I should see Firestore Console update with the correct data. Before I check Firebase, I can confirm that no errors were returned when I clicked the button and the following was output in the console, implying that the transaction was successful.

```
>>> V/FA: Activity paused, time: 12845155
>>> D/FA: Logging event (FE): user_engagement(_e), Bundle[{firebase_event_origin(_o)=auto,
↪  engagement_time_msec(_et)=2850, firebase_screen_class(_sc)=addBookActivity,
↪  firebase_screen_id(_si)=2158628408263557096}]
```

Firebase Console also update correctly.

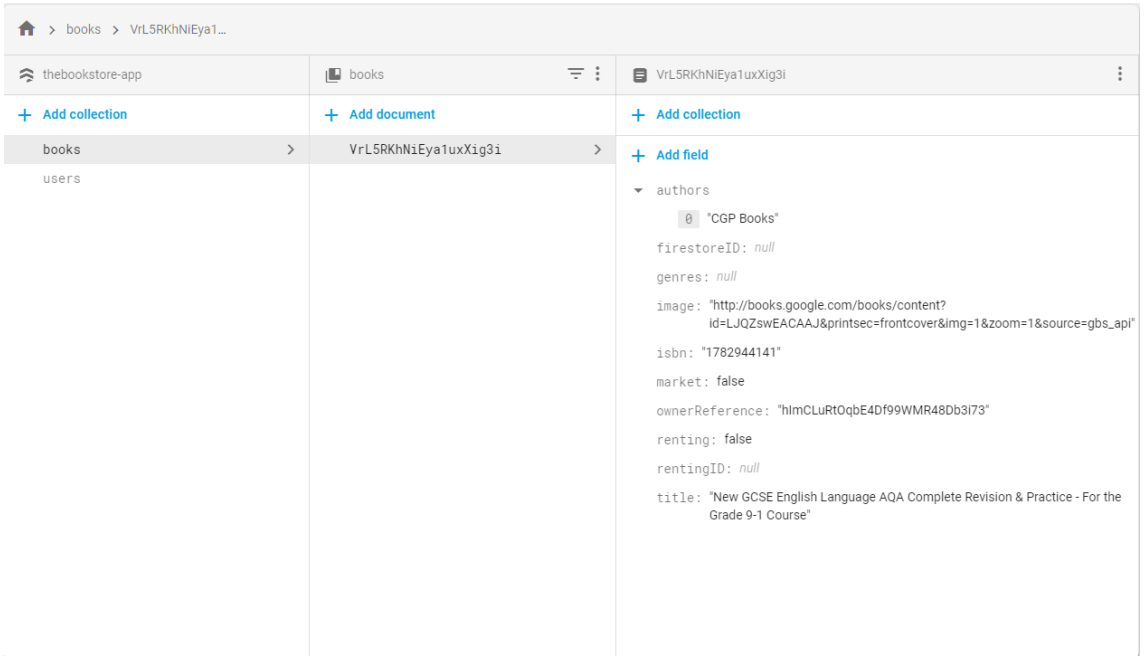


Figure 3.23: Firebase Console

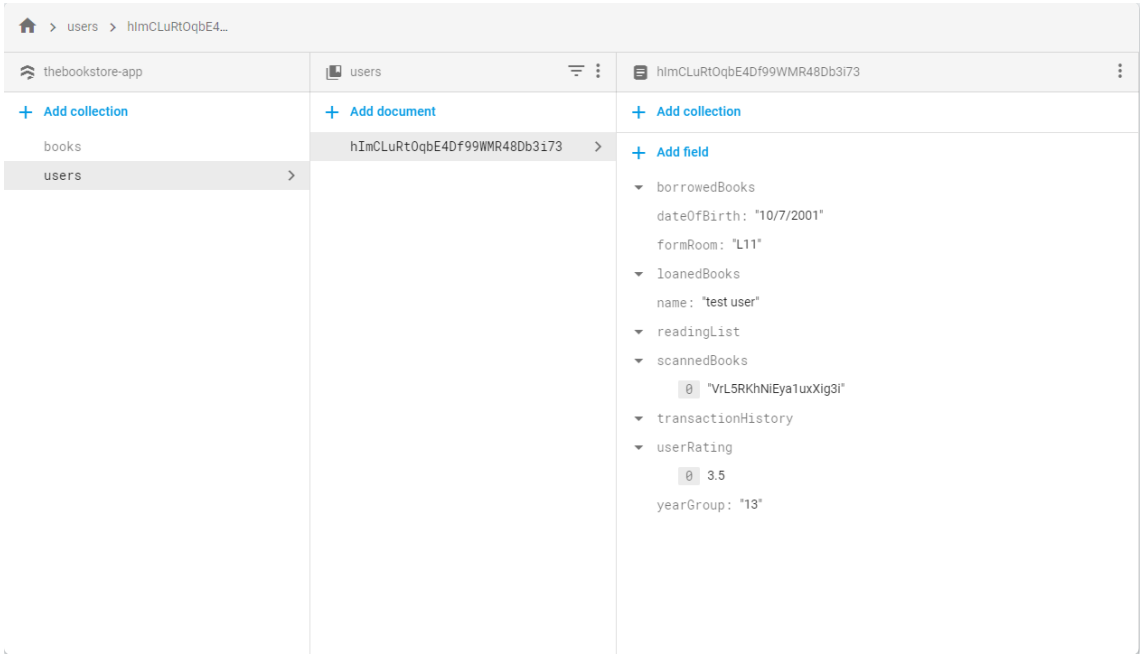


Figure 3.24: Firebase Console

3.11.2 Manual Submission

The implementation for this class is pretty straight forward. The UI is just a group of fields with a button. The code to submit the data just takes the field inputs, creates a book class from it then adds that via a transaction.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".addBookManualActivity">
    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_margin="25dp"
            android:background="@color/browser_actions_bg_grey"
            android:orientation="vertical"
            android:padding="10dp">
            <ImageView
                android:id="@+id/imageView3"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:contentDescription="@string/app_name"
                app:srcCompat="@mipmap/ic_launcher_foreground" />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/isbn"
                android:textColor="@color/colorPrimary"
                android:textSize="15sp" />
            <EditText
                android:id="@+id/isbn"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginBottom="10dp"
                android:autofillHints="@string/isbn"
                android:ems="10"
                android:inputType="number"
                android:textColor="@color/colorAccent" />
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/title_caps"
                android:textColor="@color/colorPrimary"
                android:textSize="15sp" />
```

```
55              <EditText
56                  android:id="@+id/title"
57                  android:layout_width="match_parent"
58                  android:layout_height="wrap_content"
59                  android:layout_gravity="center"
60                  android:layout_marginBottom="10dp"
61                  android:autofillHints="@string/title_caps"
62                  android:ems="10"
63                  android:inputType="text"
64                  android:maxLength="25"
65                  android:textAllCaps="true"
66                  android:textColor="@color/colorAccent" />
67              <TextView
68                  android:layout_width="match_parent"
69                  android:layout_height="wrap_content"
70                  android:text="@string/author_caps"
71                  android:textColor="@color/colorPrimary"
72                  android:textSize="15sp" />
73              <EditText
74                  android:id="@+id/author"
75                  android:layout_width="match_parent"
76                  android:layout_height="wrap_content"
77                  android:layout_gravity="center"
78                  android:layout_marginBottom="10dp"
79                  android:autofillHints="@string/author_caps"
80                  android:ems="10"
81                  android:inputType="text"
82                  android:maxLength="35"
83                  android:textAllCaps="true"
84                  android:textColor="@color/colorAccent" />
85              <TextView
86                  android:layout_width="match_parent"
87                  android:layout_height="wrap_content"
88                  android:text="@string/genre_caps"
89                  android:textColor="@color/colorPrimary"
90                  android:textSize="15sp" />
91              <EditText
92                  android:id="@+id/genre"
93                  android:layout_width="match_parent"
94                  android:layout_height="wrap_content"
95                  android:layout_gravity="center"
96                  android:autofillHints="@string/genre_caps"
97                  android:ems="10"
98                  android:inputType="text"
99                  android:maxLength="10"
100                 android:textAllCaps="true"
101                 android:textColor="@color/colorAccent" />
102         </LinearLayout>
103     </ScrollView>
104     <com.bookstore.palvindersingh.networkFAB
105         android:id="@+id/correct"
106         android:layout_width="wrap_content"
107         android:layout_height="wrap_content"
108         android:layout_margin="16dp"
109         android:layout_marginEnd="8dp"
110         android:layout_marginBottom="8dp"
111         android:clickable="true"
112         android:focusable="true"
113         android:src="@drawable/ic_done_24dp"
114         app:backgroundTint="@color/fui_bgPhone"
115         app:fabSize="normal"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:fontFamily="@font/arimo"
        android:padding="10dp"
        android:text="Submit"
        android:textColor="#000"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="@+id/correct"
        app:layout_constraintEnd_toStartOf="@+id/correct" />
</android.support.constraint.ConstraintLayout>
```

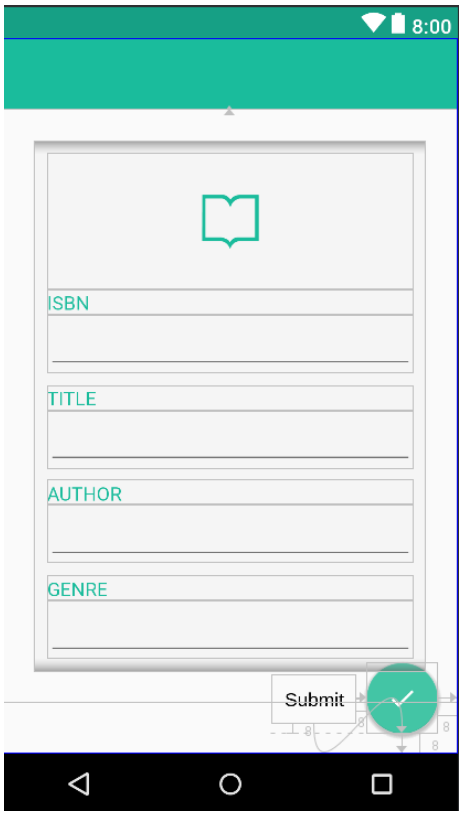


Figure 3.25: Submission UI

```
package com.bookstore.palvindersingh;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
```

```
14 import com.google.firebase.auth.FirebaseAuth;
15 import com.google.firebase.auth.FirebaseUser;
16 import com.google.firebase.firestore.DocumentReference;
17 import com.google.firebase.firestore.FieldValue;
18 import com.google.firebase.firestore.FirebaseFirestore;
19 import com.google.firebase.firestore.Transaction;
20 import java.util.ArrayList;
21 public class addBookManualActivity extends AppCompatActivity {
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_add_book_manual);
26         //initialise toolbar
27         Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
28         myChildToolbar.setTitle("submit a book");
29         setSupportActionBar(myChildToolbar);
30         ActionBar ab = getSupportActionBar();
31         ab.setDisplayHomeAsUpEnabled(true);
32         //initialise correct floating action button
33         networkFAB correct = findViewById(R.id.correct);
34         //on click listener
35         View.OnClickListener correctClick = new View.OnClickListener() {
36             @Override
37             public void onClick(View v) {
38                 //initialise ui components
39                 EditText isbn = findViewById(R.id.isbn);
40                 EditText title = findViewById(R.id.title);
41                 EditText author = findViewById(R.id.author);
42                 EditText genre = findViewById(R.id.genre);
43                 //check if input data is valid
44                 if (validateInputs(isbn.getText().toString(), title.getText().toString(),
                    ↪  author.getText().toString(), genre.getText().toString())) {
45                     //add data to firestore
46                     final FirebaseFirestore db = FirebaseFirestore.getInstance();
47                     final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
48                     ArrayList<String> authorList = new ArrayList<>();
49                     authorList.add(author.getText().toString());
50                     ArrayList<String> genreList = new ArrayList<>();
51                     genreList.add(genre.getText().toString());
52                     //create book class
53                     final book book = new book(authorList, genreList, null,
                        ↪  isbn.getText().toString(), title.getText().toString(), user.getUid(), false,
                        ↪  false, null);
54                     //transaction processing
55                     db.runTransaction(new Transaction.Function<Void>() {
56                         @Nullable
57                         @Override
58                         public Void apply(@NonNull Transaction transaction) {
59                             //add book to collection
60                             db.collection("books").add(book).addOnSuccessListener(new
                                ↪  OnSuccessListener<DocumentReference>() {
61                                 @Override
62                                 public void onSuccess(DocumentReference documentReference) {
63                                     //update user data
64                                     DocumentReference docRef =
                                        ↪  db.collection("users").document(user.getUid());
65                                     docRef.update("scannedBooks",
                                        ↪  FieldValue.arrayUnion(documentReference.getId()));
66                                 }
67                             });
68                             return null;
```

```
69                      }
70                  }).addOnSuccessListener(new OnSuccessListener<Void>() {
71                      @Override
72                      public void onSuccess(Void aVoid) {
73                          //go to home activity if successful
74                          Toast.makeText(getApplicationContext(), "book added",
                            ↪  Toast.LENGTH_SHORT).show();
75                          Intent intent = new Intent(addBookManualActivity.this,
                            ↪  homeActivity.class);
76                          startActivity(intent);
77                          finish();
78                      }
79                  }).addOnFailureListener(new OnFailureListener() {
80                      @Override
81                      public void onFailure(@NonNull Exception e) {
82                          //go to scan activity if not successful
83                          Toast.makeText(getApplicationContext(), "something unexpected occurred",
                            ↪  Toast.LENGTH_SHORT).show();
84                          Intent intent = new Intent(addBookManualActivity.this,
                            ↪  scanBookActivity.class);
85                          startActivity(intent);
86                          finish();
87                      }
88                  });
89              } else {
90                  //prompt user to input valid data
91                  Toast.makeText(getApplicationContext(), "enter valid data",
                    ↪  Toast.LENGTH_SHORT).show();
92              }
93          }
94      };
95      //pass correctClick into correct class
96      correct.setClickEvent(correctClick);
97  }
98  //validate user inputs
99  private Boolean validateInputs(String isbn, String title, String author, String genre) {
100     //check if isbn is valid
101     if (!(isbn.length() == 10 || isbn.length() == 13)) {
102         return false;
103     }
104     //check if other data lengths are appropriate
105     return title.length() >= 3 && author.length() >= 3 && genre.length() >= 3;
106 }
107 }
```

To test this activity, I navigated to it (confirming that all navigation components function correctly) then added the following data then submit it. Everything executed as expected.

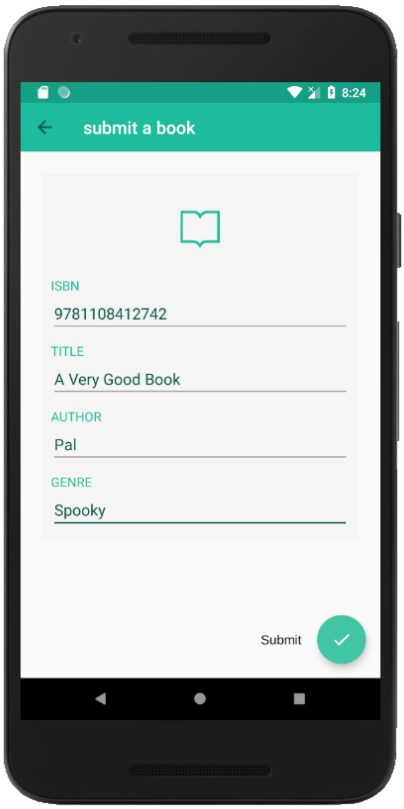


Figure 3.26: Submission UI

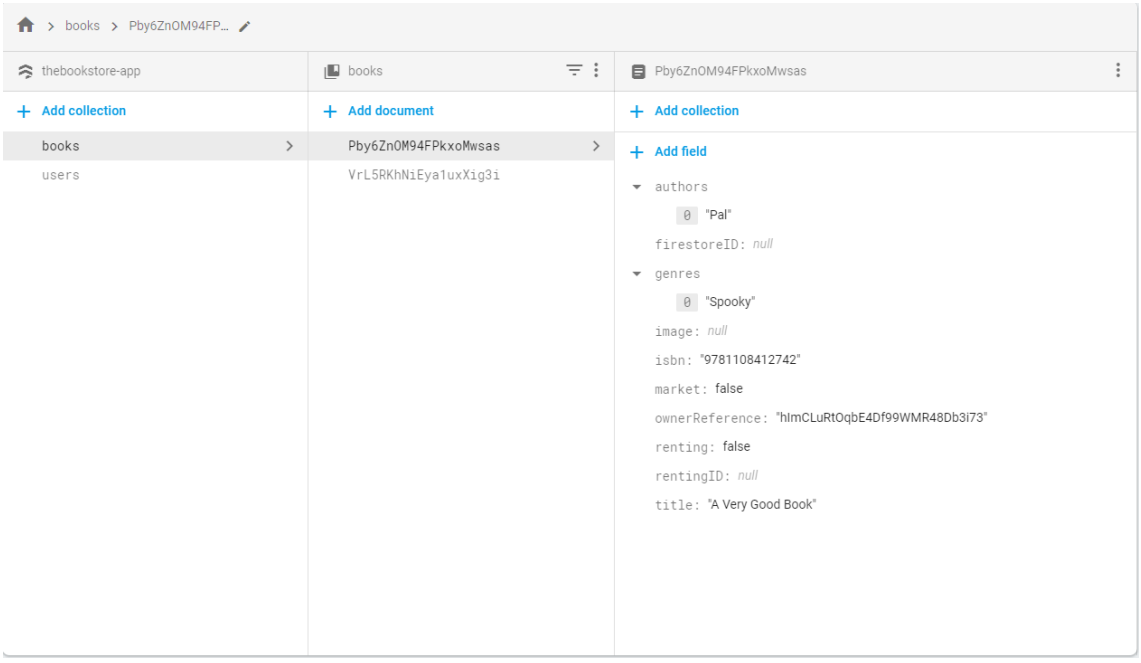


Figure 3.27: Firebase Console

3.12 Prototype Specification Testing

To thoroughly test my finished prototype, I will refer to the 55 tests I detailed in the design of the this implementation. I will provide/list recordings for them all as media evidence. I will test my app using the black-box testing method. Also known as Behavioral Testing, Black Box testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. In other-words I will be looking for whether the desired outputs are met.

1. Tests 1-4: Splash Screen Tests
    - $imp1TEST1.webm$
    - $imp1TEST2.webm$
    - $imp1TEST3.webm$
    - $imp1TEST4.webm$
    - The outputs were exactly as expected
    - I did notice that the processing time for this activity varied quite significantly. I believe this is dependent on the quality of an internet connection (if one exists) or the speed at which the device is able to read cached Firestore data from it's secondary storage.
2. Tests 5-6: No Network Tests
    - $imp1TEST5-6.webm$
    - The outputs were exactly as expected
    - Something that I did not previously spot was the automatic implementation of the color scheme in certain elements such as the swipe down loading ring. This is a nice addition.
3. Tests 7-13: Sign Up Tests
    - $imp1TEST7-13.webm$
    - Generally outputs were exactly as expected
    - Although all tests were successful, the format of the login UI brought some potential usability issues to mind. When the user is prompted to login an email input field appears, however if the user does not have an account an email should still be input, from which the interface takes the input and passes it onto a new user activity. Although this is quite intuitive, I think that some people may have difficulty identifying this is how you sign up. For that reason, I will discuss this topic with my stakeholders.
4. Tests 14-19: Log In Tests
    - $imp1TEST14-18.webm$
    - $imp1TEST19.webm$
    - The outputs were exactly as expected
    - Again, loading times varied depending on internet connection speeds but were generally quite quick
5. Tests 20-26: Details Tests
    - $imp1TEST20-26.webm$
    - The outputs were exactly as expected
6. Tests 27-31: Home Tests
    - $imp1TEST27-31.webm$
    - The outputs were exactly as expected
    - I initially added the restart option button for potential testing purposes but I think that it would be wise to keep it in the final application as it allows the users an automated way of essentially closing and re-opening the app

7. Tests 32-43: Scan Tests
    - $imp1TEST32 - 33.webm$
    - $imp1TEST36 - 41.webm$
    - $imp1TEST42 - 43.mp4$
        - this recording also includes the tests 34 and 35.
    - The outputs were exactly as expected
    - Some initial difficulties with scanning books due to bad lighting conditions and glare on the book from the flash light however this was an extreme case
    - A nifty additional feature of my camera implementation that I completely overlooked was the means of requesting permission to use the camera. I believe this is a clear highlight of using widely supported open-source libraries as they often solve issues that are overlooked by the general development community thus providing more in depth and through features across the board.
8. Tests 44-46: Add Book Tests
    - $imp1TEST44 - 46.webm$
    - The outputs were exactly as expected
9. Tests 47-55: Add Book Manual Tests
    - $imp1TEST47 - 55.webm$
    - The outputs were exactly as expected

Table Of Success Criteria		
Criteria	Justification	Achieved
Account Management - Create an account, log in and out capability on multiple devices, user data storage (date of birth, year group, form room)	The end user should be able to log in from any where and access their data, this is important for identification purposes and for other services such as book exchanges when each user's form room is required. This should also allow for greater dexterity of use for the user.	Yes
Book Input - ISBN scanner, ISBN manual input, meta data source, complete data manual input	This data is required for the basic functioning of the app; users need to be able to see which books they are interacting with. This should also allow the user to easily add data to their account without the need for long input processes, although this is still available, reducing human error.	Yes
Error Free	It would not be acceptable to present a program containing glitches or bugs, especially any logic errors which can mean the application runs but not as expected.	Yes
Graphics Styling	The client should be impressed by the material theme but it should not be a processor intensive task.	Yes
Navigation Layout - bottom navigation, back navigation	Client should be able to feel a flow in the app whereby they can open activities and return to previous ones quickly. Should also prevent users from navigating to expired activities such as to the log in screen once logged in.	Yes
Suitable Complexity	The app must live up to the standards of an A-Level project for .g. database management, file handling, network tasks, multi-threaded operations , searching and sorting algorithms etc.	Yes

3.13 Stakeholder Feedback

After having showed my main stakeholders, Jeevon Shashi and Lewis, the current prototype, they were immediately impressed by the striking user interface and the suitable colour scheme that had been used to make it look professional even in this early first prototype stage. They found the entire experience satisfactory but were concerned with the varying time taken to scan books. Given this is out of my hands to optimise I ensured them that the the tools available, flash and manual input, should supplement for such short comings. Over this implementation process, the time taken for learning the content and implementation has been alarming as it has taken longer than expected. For this reason I discussed the possibility of removing less prominent/essential features from future prototypes. The main feature we discussed and agreed upon removing was the rating features and potentially simplifying the requesting features. Despite the rating systems advantages of bringing a fairness to the system it will take a long time to implement. Thus in time I will present to them my adjusted solution that will make a comprise between time and functionality.

Chapter 4

Implementation Two

4.1 Requirements Revision

In the previous implementation I discussed the possibility of the removing certain features for the sake of a timely completion of the application. In this section I will flesh my proposed revisions out and then present it to my stakeholders.

4.1.1 Rating

The first area that I will remove is the rating feature. Instead of depending on users to rate others, I will create an automated way of determining the users rating. To create this system I must first consider the most important aspects of a users data in relation to book transactions. This would be the number of books they have rented out to other or the number of books they themselves have rented, the ratio of these numbers. The number of books is important as it provides an insight into how active the user is within the community. The ratio of these numbers is significant since it gives a quantitative value for how the user uses the platform, does he/she rent more or lease more? If the user rents and leases books you could say that they have wider and more positive impact on the community than someone who only rents books. Given that I have already incorporated unpopulated arrays for user's rented and borrowed books, in their documents, I should be able to access this data by simply fetching the array and querying it's size. The problem with this solution is that there is no limit to the rating of a user. I would like to keep the requirement of a 0-5 scale for a users rating. Thus I need a way to convert any number to a number between 0 and 5. If we think back to a brief, unrelated, paragraph I wrote in the previous implementation in regards to neural networks; you may remember me mentioning the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

This function takes any real number and plots it between 0 and 1, as can be seen by the graph below: The function is particularly useful as it becomes increasingly harder and pretty impossible

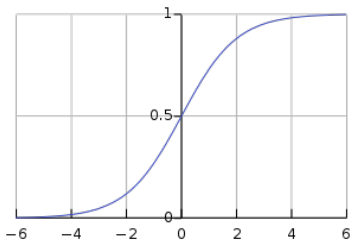


Figure 4.1: Sigmoid Function

to reach a maximum value of 1. On the other hand, it is quite easy to get a lower rating. To scale this function up to a 0-5 range all you have to do is multiply by 5 and round to the nearest integer. Now in context, this should make a pretty concrete automated way of determining a users score whilst ensuring that users have an incentive to maintain their credibility. I consider various special

cases such as dividing by 0, as this would result in a math error. To get around this problem I believe adding one to the loaned and borrowed counters could be the solution as we are calculating a ratio thus if both a greater than 1 there will be minimal influence on the overall rating.

4.1.2 Requesting

The next area I must consider is the renting/requesting aspect of the application. When I first thought of this app, I had a grand idea of a pooling system where transactions between users are stored in which various variables are agreed upon by both parties, time and location. I now think that this is both infeasible and naïve to implement. By infeasible I mean there is a lot to implement here and not a huge deal of time; I would rather create an adequate and working solution than one that could be better but is not finished and thus has no benefit; this is also what my stakeholders believe. I say that is naïve because adding such rules and restrictions on a user's interaction with the platform can be good on paper but in reality would be a tiresome system that I myself would probably not want to use. For this reason, I think that is in my best interest to create a easier and simpler way of requesting books. Given that this is an app to be used internally within the school, all students are pretty familiar with all the classrooms and year groups thus I think that providing this information to a requester should be enough for them to go to the person in reality and request the book. This therefore makes the app a resource for finding books rather than absolute moderation of it's exchange. Furthermore, such an exchange is an unpredictable event and thus any unforeseen changes would have to be accounted for in my original solution, adding extra unnecessary detail. In terms of the actual implementation, I think it should be as simple as searching for a book, viewing details of the book and user and then having the option of renting the book. This action will change the state of the book in Firebase to 'renting' and thus the owners app will mark it as being rented out and part of the rented ui component. On the person's app who is renting the book it will come under the rented books ui component where by it can be removed from its renting state.

4.1.3 Stakeholder Opinions

I will now refer back to my stakeholders for their opinions on the matter. I will refer to all my stakeholder: Joe, Jeevon, Lewis, Shashi and Mr Jones.

I proposed my solution to Jeevon, Lewis and Shashi and how it may affect them in terms of reliability of renting important books (being older than most of the school they want to rent revision books). They detailed to me that they do not mind this new requesting system as it is seems like less hassle and provides the same services. They also agreed that they would want such a system over the previous. Jeevon had his doubts in regards to the rating system due to its automated fashion. He asked how I can guarantee it is always accurate. I then detailed the maths behind it and he seemed satisfied with the solution.

I then asked Joe, a younger year student, on his opinions. He seemed to gauge with the concepts better than previous meetings. This gives me the indication it is a simpler system to use. I asked if this would hinder his use of the platform for story books. He told me that it would not and would provide all the requirements of the original solution.

Finally, I asked Mr Jones whether the solution is viable with school rules. He told me that the simplicity of it provides a greater transparency in the system so that if a problem does occur it cannot be blamed on technology or other possible faults. He also stated that the rating system should provide a better non-bias way of rating where everyone is effectively on a level playing field and no manipulation of the platform can occur.

On this positive feedback I will begin the design of the prototype.

4.2 Design

4.2.1 Prototype Requirements

As part of this final prototype I will focusing on 6 main areas:

1. Rating Function
2. Scanned Books View
3. Specific Scanned Book View and Management
4. Search Book View
5. Searched Book View and Requesting
6. Currently Rented Book View and Withdrawal of Renting

There a various pre-requisite areas of knowledge I will have to have before implementation, this includes:

- Recycle Views
- Math in Java
- Firestore Queries

4.2.2 Pre-Requisite Areas of Knowledge

Recycle View

I will start by explaining what Recycle Views are and how I can use them. Whenever there is an array of books to tbe displayed to the user I can use a Recycle View ui component to represent them: As we can see, I want to have a Recycle View where each book is it's own "tab". Each tab

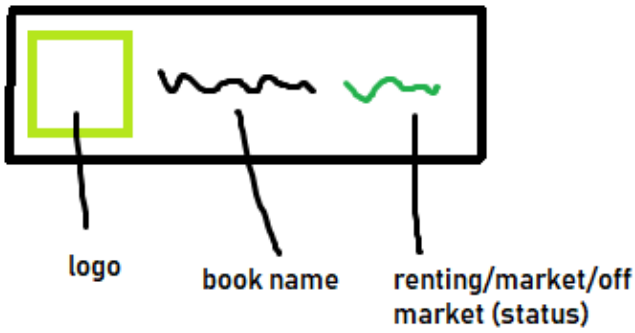


Figure 4.2: Recycle View

should have the app logo indicating that is is a book along with the book name and status of the book. The should be a clickable element that takes the user to a page that allows one to view the book. The implementation of this can be seen below: A dataset, in my case an array of books, will

Figure 4.3: Recycle View Implementation

be passed to an adapter. This adapter will have to be a class that I must code. Most of the code is boilerplate however there are few important polymorphic override methods where I will be able to add my own code for custom function. This class inherits from a parent Recycle View class,

this is what makes this level of abstracted customisation possible. I will also need to create the ui in XML, as I did for the bottom navigation components in the previous prototype, this should not be difficult. I will have three implementations of the class, once for all scanned books, for rented books and searched books.

Math

Math in Java is not that complicated of an implementation as it is part of the standard library packages. I will be using the Math library. Here is an example of an implementation of the sigmoid function.

```
import java.lang.Math.*;
int x = //some value
double y = Math.round((1 / (1 + Math.exp(-x))) * 5);
```

Firestore Queries

The syntax here is quite straight forward. First the package must be imported. The package essentially provides a static class (class that has not been/does not need to instantiated instead it acts a global data structure). This means that to access certain functions, Math itself just needs to be called with the appropriate method. Here .round(...) is used to round to the nearest integer and .exp(...) is used to raise e to the power of the arguments passed.

As I have found before, becoming familiar with various Firestore capabilities is easy than other technologies due to its well documented source code and online documentation[18]. There are several techniques I have picked up on whilst reading their documentation. Before I get onto queries I will detail various ways I can access and update data. At the time of writing the design stages of the first prototype firestore was in beta however now it is has been officially launched and with that a plethora of new features (older ones still work fine).The first is updating a document. All I have to do now is create a HashMap with the relevant attributes and I can merge it with the document stored in Firebase. For example with a book:

```
String bookID = //firestore document ID
Map<String, Object> data = new HashMap<>();
dataBook.put("renting", false);
dataBook.put("rentingID", "");
db.collection("books").document(bookID).set(data, SetOptions.merge());
```

It is equally easy to delete a document:

```
String bookID = //firestore document ID
db.collection("books").document(bookID).delete();
```

Querying for books is also a simple. Here is an example that I will have to use later on for searching.

```
db.collection("books").whereEqualTo("isbn", input).whereEqualTo("market", true).get()
```

I have now covered all areas I believe need to be researched. I can begin with decomposing this prototype down into more manageable tasks.

4.2.3 Decomposition

I have already detailed the requirements of this prototpye. I will now go into more detail discussing and providing Structure Diagram for each aspect.

Rating Function

I have discussed the inner workings of this features before. It is the implementation that I am concerned with as there are a few changes to be made to already written code. I am unsure as to when the computation of calculating a users rating should be done. I can confirm it will be done on a users device, not by a cloud function or other device. I think that process should be run each time the app is opened i.e each time the app is at the main activity. This will ensure that it is update at a decently timed interval. If it was every time a user interacts with firebase or adds a book the users rating may fluctuate too much and thus causes issues for people searching for their books at the same time. The application will probably have this flow:

$$\textit{Load App} \rightarrow \textit{Get User Data} \rightarrow \textit{Compute Rating} \rightarrow \textit{Update Rating if Changed} \rightarrow \ldots$$

Books Fragment Recycle Views

On the books fragment there will be 2 recycle views. One for all a user scanned books and one for all a users rented books. When there are not books for either an empty tab detailing either "add a book" or "rent a book" should be added. For each tab it should detail whether the book is "on rent" or "market" or "off market" or "being rented". In order to do this I must load the users data each time the fragment is loaded and query any books with the user's id as a renting id attribute:

$$\textit{Load Fragment} \rightarrow \textit{Get Books} \rightarrow \textit{Show all Books} \rightarrow \textit{Query Rented Books} \rightarrow \textit{Show rented books}$$

View a Specific Book

For the recycle view in books fragment that displays all books I should be able to click on it and go to an activity where details of the book are show. I should also provide Floating Action Buttons to delete or put the book on the market. By that I mean changing the books market boolean value to true so it can be searched. The details should also state the books "state" in bold. If the book is on rent the delete and market buttons should be disabled/not visible.

Searching a Book

The Search Fragment should have a bar, that looks like the diagram below, at the bottom of the screen with an input for an isbn and a search button. There should also be a text label to show how many results there are. All results should be ranked in order of the users rating. For this I must use an appropriate sorting algorithm. Each book should also be clickable whereby you are

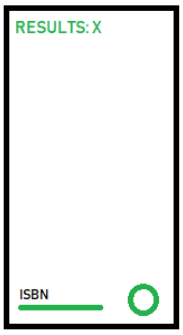


Figure 4.4: Search UI

taken to a screen that displays book details. For the search, I will query all books that have market set to true and have a corresponding ISBN. To sort the data I think that it would be best to use an algorithm with a space complexity of $O(1)$. This is because I will be moving book objects around in an array. They can be large data structures that take up a lot of memory thus any non in-place

algorithms may use already limited RAM resources. This eliminates both quick and merge sorts. I must therefore choose between an insertion and bubble sort. Given that the queries are returned randomly, It can be said that over the course of the applications use the searches will be distributed in it's worst or average case scenario; this more likely for larger queries. Thus comparing insertion and bubble sort leaves no apparent clarity as to which one to choose as both have a complexity of $O(n^2)$. I googled the problem and found a useful blog[19] discussing the issue and why insertion sort seems to generally be faster despite the same complexities. It essentially discussed that overall per iteration of the data set the insertion sort does less operations and overall the same number of passes of the data. To support my decision to exclude merge and quick sorts I have also found that both insertion sort and bubble sort are typically faster than the O(nlogn) alternatives for data sets less than 20. This will most definitely be the case (less than 20 data items), I doubt 20 different users will scan the same book. Based on this research and comparison I will use an insertion sort as it has the most appropriate time and space complexities. It is also to be noted that books will be ranked in terms of rating. This data is not part of a book collection thus a users data will be need to be queried for each book to fetch the rating.

Search Book Recycle View

The final recycle view will be one to view results of a searched isbn. Resulting books should be output here. Each tab should have the same layout as before except not status should be shown as it is not appropriate/has not use to be shown. On clicking on a tab a user should be taken to another page to view the book in greater depth.

Renting Book Acitvity

This activity should be the one that comes up when a searched book is clicked. It should have details about the books owner (name, year, form and rating) along with the book. A floating action button should be visible that allows the user to rent the book. This should essentially take the book of the market, at it to the array of the renters borrowedBooks array and also added to the book owners loanedBooks array. Upon thinking about this layout it can be said that it will be almost identical to the page that will appear when a tab is clicked on the rented books recycle view. I think that it would be efficient to therefore reuse the activity. I would have to implement another floating action that does the opposite to the first except it keep the book off the market and sets the book renting value to false. This should be hidden if the book is being searched (i.e renting is false). The rent request button should therefore also be hidden if the book is being rented (i.e renting is true).

Structure Diagram

I have created a structure diagram to illustrate all these features.

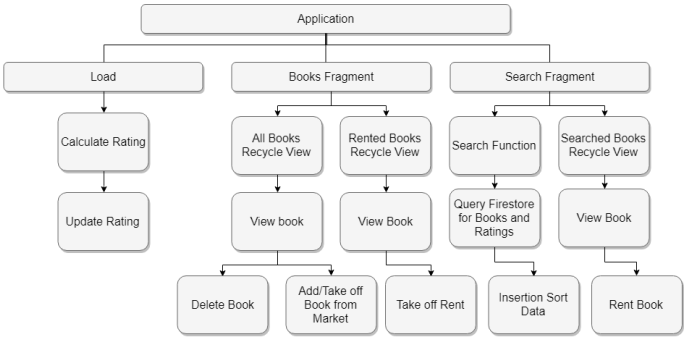
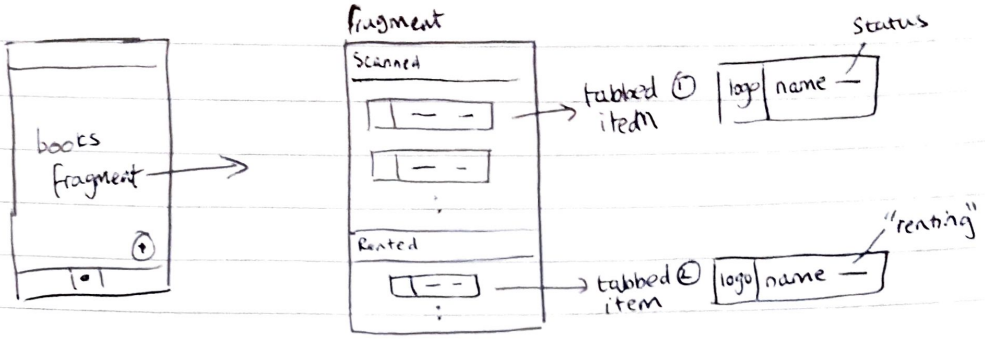
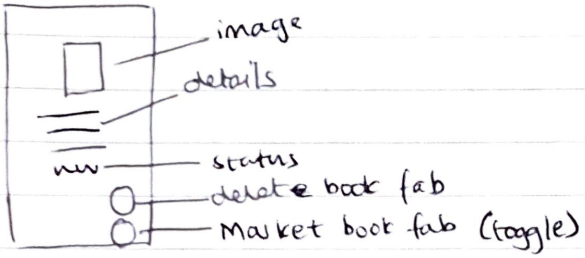


Figure 4.5: Feature Decomposition

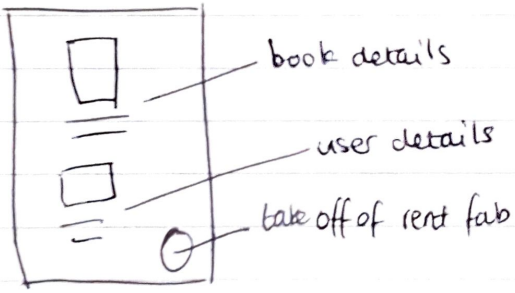
Usability Diagram

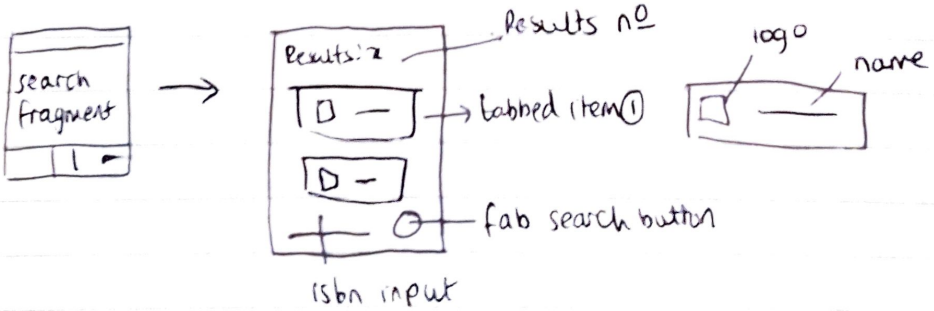


tabbed item ① click

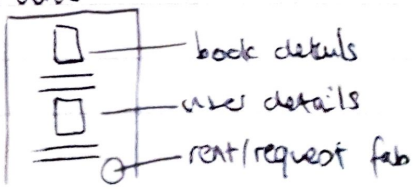


tabbed item ② click





tabbed item ① click



4.2.4 Algorithm Design

Rating System

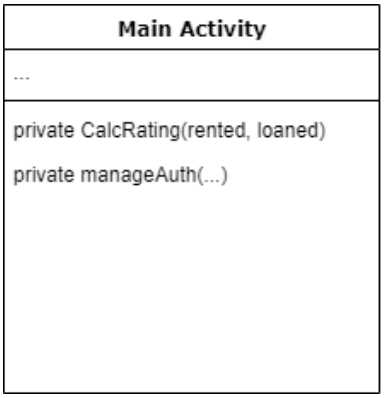


Figure 4.6: Class Diagram

Algorithm: calcRating(double rented, double loaned)

```
1  try:
2  |   rented = rented + 1;
3  |   loaned = loaned + 1;
4  |   int x = (int) loaned/rented;
5  |   double rating = double rating = Math.round((1 / (1 + Math.exp(-x))) * 5);
6  |   return rating;
7  catch:
8  |   return-1;
9  end
```

Algorithm: manageAuth()

```
1  ... amongst already coded class ..;
2  if !Objects.equals(array.get(0), "") and array.get(1) != "" and array.get(2) != "" then
3  |   ArrayList currentlyRented = get borrowed books;
4  |   double currentlyRentedSize = currentlyRented.size();
5  |   ArrayList loanedBooks = get loaned books;
6  |   double loanedBooksSize = loanedBooks.size();
7  |   Integer rating = (int) calcRating(currentlyRentedSize, loanedBooksSize);
8  |   Integer userRating = (int) get user rating;
9  |   if rating != userRating and rating != -1 then
10 |   |   Update user document userRating with rating;
11 |   end
12 |   ... amongst already coded class ..;
13 end
```

As we can see, in the calcRating(rented, loaned) method I have surrounded the majority of the code with a try catch statement. This is because data is coming from Firestore, error may occur in which data is collected which is inaccurate or of the wrong type. Thus using the Math package on this data would result in an error. Catching this error and returning -1 is a way for me to later identify in the manageAuth() method if the operation has been successful. Furthermore if the user has no books rented the value of x will be calculated by dividing by 0. This operation is not possible and would return an error. For this reason I have added 1 to both values of rented and loaned.

Books Fragment

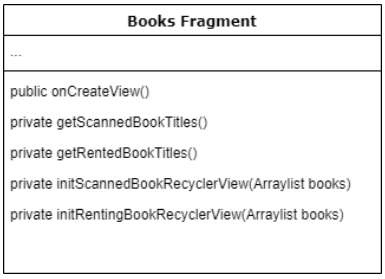


Figure 4.7: Class Diagram

Algorithm: onCreateView()

```
1 getScannedBookTitles();
2 getRentedBookTitles();
```

Algorithm: getScannedBookTitles()

```
1  initialise Firestore;
2  String userID = get user's firestore id;
3  task = firestore.collection("books").whereEqualTo("ownerReference", userID).get();
4  if task is successful then
5  |   ArrayList books;
6  |   for ( book document in task ) {
7  |   |   book book = get book from document;
8  |   |   book.setFirestoreID(document.getID);
9  |   |   books.add(book);
10 |   }
11 end
12 if books.size() == 0 then
13 |   books.add(new book(null, null, null, null, "add a book", null, false, false, null));
14 end
15 initScannedBooksRecyclerView(books);
```

Algorithm: getRentedBookTitles()

```
1  initialise Firestore;
2  String userID = get user's firestore id;
3  task = firestore.collection("books").whereEqualTo("rentingID", userID).get();
4  if task is successful then
5  |   ArrayList books;
6  |   for ( book document in task ) {
7  |   |   book book = get book from document;
8  |   |   book.setFirestoreID(document.getID);
9  |   |   books.add(book);
10 |   }
11 end
12 if books.size() == 0 then
13 |   books.add(new book(null, null, null, null, "rent a book", null, false, false, null));
14 end
15 initRentingBooksRecylerView(books);
```

Algorithm: initScannedBooksRecyclerView(ArrayList books)
1 initialise scanned books recycle view class;

Algorithm: initRentingBooksRecylerView(ArrayList books)
1 initialise rented books recycle view class;

I have not included the recycleAdapter classes as these mostly contain boilerplate code that can be autogenerated by my IDE. I will implement these classes directly when coding this stage.

View Book

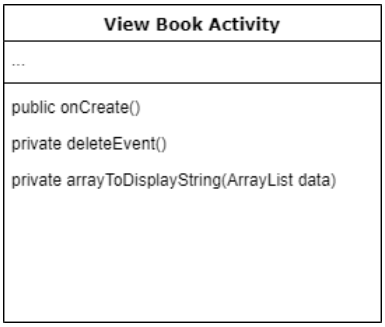


Figure 4.8: Class Diagram

Algorithm: deleteEvent

```
1 initialise Firestore;
2 task = firestore.collection("books").document(meta.getFirestoreID()).delete();
3 if task is successful then
4 |   String firestoreID = meta.getFirestoreID();
5 |   Delete firestoreID from user's scannedBook document field;
6 |   Go to home activity;
7 end
```

Algorithm: clickEventMarket

```
1  initialise Firestore;
2  if !meta.getMarket() then
3  |   if !meta.getRenting() then
4  |   |   task =
   |   |    firestore.collection("books").document(meta.getFirestoreID()).update("market",
   |   |    true);
5  |   |   if task is successful then
6  |   |   |   Go to home activity;
7  |   |   end
8  |   end
9  else
10 |   task =
   |    firestore.collection("books").document(meta.getFirestoreID()).update("market",
   |    false);
11 |   if task is successful then
12 |   |   Go to home activity;
13 |   end
14 end
```

Algorithm: arrayToDisplayString(ArrayList data)

```
1 String string = "";
2 for ( i = 0; i < data.length(); i++ ) {
3 |   string = string + data.get(i);
4 |   if i != data.size() - 1 then
5 |   |   string = string + " ";
6 |   end
7 }
8 return string;
```

This method is necessary as I have stored certain data in Firestore as arrays thus to output this data in a readable manner I must convert it to a string where each value is separated by a space.

Algorithm: onCreate()

```
initialise toolbar;
book meta = get data from intent;
networkFAB deleteBook = initialise delete floating action button;
networkFAB market = initialise market floating action button;
if meta.getIsbn() != null then
    TextView bookISBN = initialise bookISBN text field;
    bookISBN.setText(meta.getIsbn());
    bookISBN.setVisibility(VISIBLE);
end
if meta.getTitle() != null then
    TextView bookTitle = initialise bookTitle text field;
    bookTitle.setText(meta.getTitle());
    bookTitle.setVisibility(VISIBLE);
end
if meta.getAuthors() != null then
    TextView bookAuthors = initialise bookAuthors text field;
    bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
    bookAuthors.setVisibility(VISIBLE);
end
if meta.getGenres() != null then
    TextView bookGenres = initialise bookGenres text field;
    bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
    bookGenres.setVisibility(VISIBLE);
end
if meta.getImage() != null then
    ImageView bookImage = initialise bookImage image ui;
    Picasso.get().load(meta.getImage()).into(bookImage);
    bookImage.setContentDescription(meta.getImage());
    bookImage.setVisibility(VISIBLE);
end
TextView bookMarket = initialise bookMarket text field;
TextView marketname = initialise marketName text field;
TextView deletename = initialise deletename text field;
bookMarket.setVisibility(VISIBLE);
if !meta.getMarket() then
    if meta.getRenting() then
        bookMarket.setText("on rent");
        deleteBook.setVisibility(INVISIBLE);
        market.setVisibility(INVISIBLE);
        marketname.setVisibility(INVISIBLE);
        deletename.setVisibility(INVISIBLE);
    else
        bookMarket.setText("not on the market");
        deleteBook.setVisibility(VISIBLE);
        market.setVisibility(VISIBLE);
    end
else
    bookMarket.setText("on the market");
    market.setVisibility(VISIBLE);
end
if !meta.getRenting then
    deleteBook.setVisibility(VISIBLE);
end
deleteBook.setClickEvent(deleteEvent);
market.setClickEvent(clickEventMarket);
```

Search Fragment

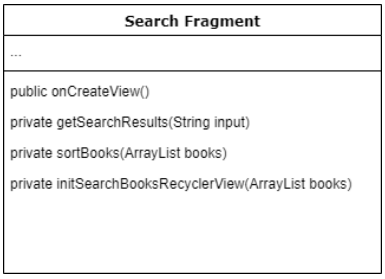


Figure 4.9: Class Diagram

Algorithm: onCreateView()

```
networkFAB searchFAB = initialise searchFAB floating action button;
EditText searchInput = initialise searchInput input field;
clickEvent = new OnClickListener(getSearchResults(searchInput.getText().toString()));
searchFAB.setClickEvent(clickEvent);
```

Algorithm: getSearchResults(String input)

```
initialise Firestore;
task = firestore.collection("books").whereEqualTo("isbn", input).whereEqualTo("market",
  true).get();
if task is successful then
    ArrayList books;
    String userID = get user's firestore id;
    for ( document document in task ) {
        book book = document;
        book.setFirestoreID(document.getID());
        if !book.getOwnerReference().equals(userID) then
            books.add(book);
        end
    }
    if books.size() == 0 then
        books.add(new book(null, null, null, null, "no books found", null, false, false, null));
    end
    if books.get(0).getTitle != "no books found" then
        sortBook(books);
    end
    TextView results = initialise result text field;
    result.setText("RESULTS: " + books.size());
end
```

Algorithm: sortBooks(final ArrayList books)

```
initialise Firestore;
ArrayList ratings;
for ( book book in books ) {
    String owner = book.getOwnerReference();
    ArrayList data;
    data.add(book);
    task = firestore.collection("users").document(owner).get();
    if task is successful then
        data.add(document.get("userRating")) ratings.add(data)
    else
        data.add(0);
        ratings.add(data);
    end
    if ratings.size() == books.size() then
        int n = ratings.size();
        for ( i = 1; i < n; i++ ) {
            ArrayList temp = ratings.get(i);
            int j = i - 1;
            ArrayList nextArray = ratings(j);
            Double tempRating = temp.get(1);
            Double nextRating = nextArray.get(1);
            while j >= 0 and nextRating > tempRating do
                ratings[j+1] = ratings.get(j);
                j = j - 1;
            end
            ratings[j+1] = temp;
        }
        ArrayList sorted;
        for ( i = ratings.size() - 1; i > -1; i-- ) {
            sorted.add(ratings.get(i).get(0));
        }
        initSearchBookRecycleView(bookList);
    end
}
```

My implementation of an insertion sort may look unfamiliar in comparison to a standard implemenation. This is because I have to add various extra feature in order to manage data from Firestore. The main task is to sort books into their owners' rating (highest to smallest). This data is not included within the book class, it must be fetched from Firestore. Thus I will have a resultant 2 sets of data, ratings and books. I decided to convert this into a 2D ArrayList, where each sub-array contains a book and the corresponding rating. I can then perform an insertion sort on that data. I must then create another array containing only books from the 2D sorted array.

Algorithm: initSearchBooksRecyclerView(ArrayList books)

```
initialise searched books recycle view class;
```

View Searched Book

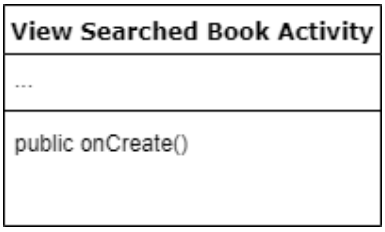


Figure 4.10: Class Diagram

Algorithm: stopRentClick

```
1 initialise firestore;
2 HashMap dataBook;
3 dataBook.put("renting", false);
4 dataBook.put("rentingID", "");
5 firestore.collection("books").document(meta.getFirestoreID()).set(dataBook,
    SetOptions.merge());
6 firestore.collection("users").document(user.getUid()).update("borrowedBooks",
    FieldValue.arrayRemove(meta.getFirestoreID()));
7 firestore.collection("users").document(meta.getOwnerReference()).update("loanedBooks",
    FieldValue.arrayRemove(meta.getFirestoreID()));
8 Go to home activity;
```

Algorithm: rentEvent

```
1 initialise firestore;
2 String userID = user's firestore id;
3 HashMap dataBook;
4 dataBook.put("market", false); dataBook.put("renting", true); dataBook.put("rentingID",
    userID); firestore.collection("books").document(meta.getFirestoreID()).set(dataBook,
    SetOptions.merge());
5 firestore.collection("users").document(userID).update("borrowedBooks",
    FieldValue.arrayUnion(meta.getFirestoreID()));
6 String bookOwnerID = meta.getOwnerReference();
7 firestore.collection("users").document(bookOwnerID).update("loanedBooks",
    FieldValue.arrayUnion(meta.getFirestoreID()));
8 Go to home activity;
```

Algorithm: onCreate()

```
1  initialise toolbar;
2  book meta = get data from intent;
3  networkFAB requestBook = initialise request floating action button;
4  TextView rentingText = initialise rentingText text field;
5  networkFAB rentBookStop = initialise renting request floating action button;
6  TextView rentBookStopText = initialise stop renting text field;
7  rentBookStop.setClickEvent(stopRentClick);
8  requestBook.setClickEvent(rentEvent);
9  if meta.getIsbn() != null then
10 |   TextView bookISBN = initialise bookISBN text field;
11 |   bookISBN.setText(meta.getIsbn());
12 |   bookISBN.setVisibility(VISIBLE);
13 end
14 if meta.getTitle() != null then
15 |   TextView bookTitle = initialise bookTitle text field;
16 |   bookTitle.setText(meta.getTitle());
17 |   bookTitle.setVisibility(VISIBLE);
18 end
19 if meta.getAuthors() != null then
20 |   TextView bookAuthors = initialise bookAuthors text field;
21 |   bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
22 |   bookAuthors.setVisibility(VISIBLE);
23 end
24 if meta.getGenres() != null then
25 |   TextView bookGenres = initialise bookGenres text field;
26 |   bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
27 |   bookGenres.setVisibility(VISIBLE);
28 end
29 if meta.getImage() != null then
30 |   ImageView bookImage = initialise bookImage image ui;
31 |   Picasso.get().load(meta.getImage()).into(bookImage);
32 |   bookImage.setContentDescription(meta.getImage());
33 |   bookImage.setVisibility(VISIBLE);
34 end
35 initialise firestore;
36 String userID = user's firestore id;
37 task = firestore.collection("users").document(meta.getOwnerReference()).get();
38 if task is successful then
39 |   TextView name = initialise name text field;
40 |   TextView year = initialise year text field;
41 |   TextView form = initialise form text field;
42 |   TextView rating = initialise rating text field;
43 |   name.setText(document.get("name").toString();
44 |   year.setText(document.get("yearGroup").toString());
45 |   form.setText(document.get("formRoom").toString());
46 |   rating.setText(document.get("userRating").toString());
47 |   name.setVisibility(VISIBLE);
48 |   year.setVisibility(VISIBLE);
49 |   form.setVisibility(VISIBLE);
50 |   rating.setVisibility(VISIBLE);
51 end
```

4.2.5 Stakeholder Feedback

As part of the Rapid Application Development methodology, I am required to show the results of my designs to my stakeholders. I particularly want to gauge whether or not my stakeholders think that the more indepth design of my revised rating feature and requesting features is in line with their original requests.

Jeevon informed my that after further research he realized that my function for determining a rating has a bottom limit of 2. As substituting 0 into the sigmoid function gives a number round able to 2. I told him my original plan of having a scale between 1 and 5. He suggested simply leaving it as is, the difference between a rating of 1 and 2 is not that significant as both denote poor usage. I understand and agree with his opinion my rating system is hueuristic and not meant to be completely accurate rather a ball park figure, something that is better than nothing. I then asked Shashi and Lewis their opinions on my planned renting system. They seemed happy with the implementation and though that it seemed simple enough for anybody form year 7 to 13 to use.

4.2.6 Testing Table

Note that for all tests I will use the same book: ISBN is 1782944141 (testing book).

Test No	Test Description	Input	Expected Output	Input Type
1	Update User Rating - 2 books rented to others and 0 borrowed	on app load	4.0	Valid
2	Update User Rating - 0 books rented and 0 loaned	on app load	3.0	Extreme
3	View Scanned Books Recycle View (no books scanned) - correct tabbed item - on click go to scan book activity	on load/on click	Displays scan a book and on click goes to scan activity	Extreme
4	View Rented Books Recycle View (no books rented) - correct tabbed item - on click go to search fragment	on load/on click	Displays rent a book and on click goes to search fragment	Extreme
5	View Scanned Book (rented, on market and scanned books)	on load	Correctly loads data	Valid
6	View Scanned Book Floating Action Buttons	on load	Correctly shows relevant floating action button	Valid
7	View Scanned Book Delete Books	delete floating action button click	Deletes books	Valid
8	View Scanned Book Market Toggle	market floating action button click	Toggles market status	Valid
9	View Rented Book	on load	Correct book data and user	Valid
10	View Rented Book Off Rent	on off rent floating action button click	Takes book off rent	Valid
11	Search Fragment	Invalid isbn or isbn that does not exist and search button click	String says results is 0	Invalid

12	Search Fragment	1782944141 input and search button click	Shows books in rating order(high-low)	Valid
13	View Searched Book	on load	Book data output and owner details output	Valid
14	View Searched Book Rent	Rent button click	Books begins to be rented	Valid

4.3 Implementation

4.3.1 Rating System

I first set about the simple task of adding the following method:

```
private double calcRating(double rented, double loaned) {
    try {
        rented = rented + 1;
        loaned = loaned + 1;
        int x = (int) (loaned / rented);
        double rating = Math.round((1 / (1 + Math.exp(-x))) * 5);
        return rating;
    } catch (Exception e) {
        return -1;
    }
}
```

I also added to following into the manageAuth() method:

```
ArrayList currentlyRented = (ArrayList) task.getResult().get("borrowedBooks");
double currentlyRentedSize = currentlyRented.size();
ArrayList loanedBooks = (ArrayList) task.getResult().get("loanedBooks");
double loanedBooksSize = loanedBooks.size();
Integer rating = (int) calcRating(currentlyRentedSize, loanedBooksSize);
Integer userRating = (int) task.getResult().get("userRating");
if (rating != userRating && rating != -1) {
    Map<String, Object> data = new HashMap<>();
    data.put("userRating", rating);
    db.collection("users").document(userID).set(data, SetOptions.merge());
}
```

Testing this should be fairly easy, all I must do is load the application and output the calculated rating. After successful compilation the following error was returned:

```
/AndroidRuntime: Shutting down VM
E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.bookstore.palvindersingh, PID: 29813
    java.lang.ClassCastException: java.lang.Double cannot be cast to java.lang.Integer
        at com.bookstore.palvindersingh.mainActivity$2.onComplete(mainActivity.java:115)
        at com.google.android.gms.tasks.zzj.run(Unknown Source:4)
```

Line 115 is in respect to: *Integer userRating = (int) task.getResult().get("userRating");*. The exception means that it is not possible to convert the double numerical value returned from Firestore for an integer. I googled this error with no avail. I solution that I think may work can be found by inspected how a number is stored in Firestore. A number is stored as a decimal pointed value i.e the value of 3 would be stored as "3.0". Given that the rating is ensured to always be an integer I can have full faith that the rating value will never have a decimal value. Thus I can convert the the double to an Integer, get the first value then convert that to an integer. The fixed solution is:

```
Integer userRating = Integer.valueOf(task.getResult().get("userRating").toString().charAt(0));
```

Upon implementing this and successful compilation no errors where returned. The account logged in has no books rented and no books scanned. The console successfully output:

```
I/System.out: rating is: 4
```

4.3.2 Books Fragment

I first created the blueprint ui for a tabbed item. This is neccessary for all future RecycleViews. The XML for this is:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/parentLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:background="@color/cardview_light_background"
    android:clickable="true"
    android:focusable="true">
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">
        <ImageView
            android:id="@+id/icon"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="clip_vertical"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp"
            android:src="@mipmap/ic_launcher"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
        <TextView
            android:id="@+id/bookTitle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:padding="10dp"
            android:text="@string/title"
            android:textColor="@color/colorPrimary"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/bookStatus"
            app:layout_constraintStart_toEndOf="@+id/icon"
            app:layout_constraintTop_toTopOf="parent" />
        <TextView
            android:id="@+id/bookStatus"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
            android:layout_gravity="center_vertical"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:padding="10dp"
            android:text="status"
            android:textColor="@color/colorPrimary"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    </android.support.constraint.ConstraintLayout>
</android.support.v7.widget.CardView>
```

This generates the following ui in my ide: As we can see, I have added a the app logo to the right

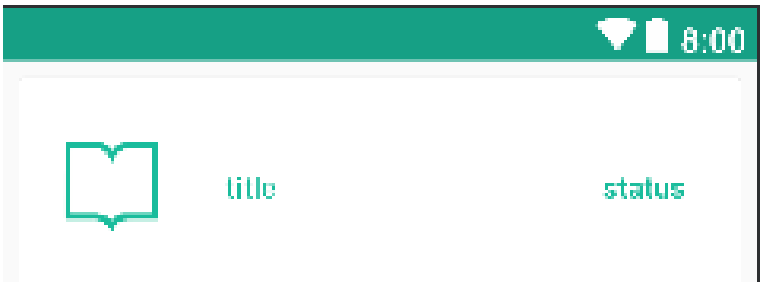


Figure 4.11: Tabbed Item

and a title text view and status text view. These will be later populated with he relavant details. I then added the following to the onCreateView method in the booksFragment class.

```
getScannedBookTitles(view);
getRentedBookTitles(view);
```

I must now implement these methods. The pre-requisites for these 2 methods are the initScanned-BooksRecyclerView and initRentingBooksRecylerView methods. To further this the pre-requisites for these 2 methods are the rentingBooksRecyclerViewAdapter and scannedBooksRecyclerViewAdapter classes. I will therefore begin by implementing these classes methods in that order.

scannedBooksRecyclerViewAdapter class This class as detailed in android documention[20] must contain 3 attributes and 3 override method and a constructor method. The first attribute should be an array of data. The second, an attribute to store a context object and the third is a class itself called ViewHolder. This class can be thought of as a blueprint for the defined tabbed item.

```
public class scannedBooksRecyclerViewAdapter extends
↪   RecyclerView.Adapter<scannedBooksRecyclerViewAdapter.ViewHolder> {
    //initialise attributes
    private final ArrayList<book> books;
    private Context mContext;
}
```

I will now add the boilerplate class attribute:

```
class ViewHolder extends RecyclerView.ViewHolder {
    TextView bookTitle;
    CardView parentLayout;
```

```
4       TextView bookStatus;
5       ViewHolder(View itemView) {
6           super(itemView);
7           bookTitle = itemView.findViewById(R.id.bookTitle);
8           parentLayout = itemView.findViewById(R.id.parentLayout);
9           bookStatus = itemView.findViewById(R.id.bookStatus);
10      }
11  }
```

I then added the remaining methods, creating the following completed class:

```
1   package com.bookstore.palvindersingh;
2   import android.app.Activity;
3   import android.content.Context;
4   import android.content.Intent;
5   import android.support.annotation.NonNull;
6   import android.support.v7.widget.CardView;
7   import android.support.v7.widget.RecyclerView;
8   import android.view.LayoutInflater;
9   import android.view.View;
10  import android.view.ViewGroup;
11  import android.widget.TextView;
12  import java.util.ArrayList;
13  public class scannedBooksRecyclerViewAdapter extends
    ↪  RecyclerView.Adapter<scannedBooksRecyclerViewAdapter.ViewHolder> {
14      //initialise attributes
15      private final ArrayList<book> books;
16      private Context mContext;
17      //boilerplate methods
18      scannedBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
19          this.books = books;
20          this.mContext = mContext;
21      }
22      @NonNull
23      @Override
24      public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
25          //inflate layout
26          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item,
            ↪  parent, false);
27          return new ViewHolder(view);
28      }
29      @Override
30      public void onBindViewHolder(@NonNull ViewHolder holder, final int position) {
31          book book = books.get(position);
32
33          //check and output book state
34          if (!book.getMarket()) {
35              if (book.getRenting()) {
36                  holder.bookStatus.setText("on rent");
37              } else {
38                  if (book.getTitle() != "add a book") {
39                      holder.bookStatus.setText("off market");
40                  } else {
41                      holder.bookStatus.setVisibility(View.INVISIBLE);
42                  }
43              }
44          } else {
45              holder.bookStatus.setText("on market");
46          }
47          //set title
48          holder.bookTitle.setText(books.get(position).getTitle());
```

```
49          //initialise on click listener
50          holder.parentLayout.setOnClickListener(new View.OnClickListener() {
51              @Override
52              public void onClick(View v) {
53                  book book = books.get(position);
54                  if (book.getTitle() == "add a book") {
55                      //if no books added yet perform add book floating action button click
56                      networkFAB btn = ((Activity)
                        ↪  mContext).findViewById(R.id.floating_action_button);
57                      btn.performClick();
58                  } else {
59                      //go to view book class
60                      Intent intent = new Intent(mContext, viewBookActivity.class);
61                      intent.putExtra("book", book);
62                      mContext.startActivity(intent);
63                  }
64              }
65          });
66      }
67      //method to return array size
68      @Override
69      public int getItemCount() {
70          return books.size();
71      }
72      //boilerplate class to represent layout
73      class ViewHolder extends RecyclerView.ViewHolder {
74          TextView bookTitle;
75          CardView parentLayout;
76          TextView bookStatus;
77          ViewHolder(View itemView) {
78              super(itemView);
79              bookTitle = itemView.findViewById(R.id.bookTitle);
80              parentLayout = itemView.findViewById(R.id.parentLayout);
81              bookStatus = itemView.findViewById(R.id.bookStatus);
82          }
83      }
84  }
```

scannedBooksRecyclerViewAdapter class This class is exactly the same as the previous with a few minor changes. It sets the tabbed item's status to "renting".

```
1   package com.bookstore.palvindersingh;
2   import android.app.Activity;
3   import android.content.Context;
4   import android.content.Intent;
5   import android.support.annotation.NonNull;
6   import android.support.design.widget.BottomNavigationView;
7   import android.support.v7.widget.CardView;
8   import android.support.v7.widget.RecyclerView;
9   import android.view.LayoutInflater;
10  import android.view.View;
11  import android.view.ViewGroup;
12  import android.widget.TextView;
13  import java.util.ArrayList;
14  public class rentingBooksRecyclerViewAdapter extends
    ↪  RecyclerView.Adapter<rentingBooksRecyclerViewAdapter.ViewHolder> {
15      //initialise attributes
16      private final ArrayList<book> books;
17      private Context mContext;
18      //boilerplate methods
```

```
19      rentingBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
20          this.books = books;
21          this.mContext = mContext;
22      }
23      @NonNull
24      @Override
25      public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
26          //inflate layout
27          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item,
            ↪  parent, false);
28          return new ViewHolder(view);
29      }
30      @Override
31      public void onBindViewHolder(@NonNull ViewHolder holder, final int position) {
32          book book = books.get(position);
33          holder.bookStatus.setVisibility(View.GONE);
34          if (book.getTitle() != "rent a book") {
35              holder.bookStatus.setVisibility(View.VISIBLE);
36              holder.bookStatus.setText("renting");
37          }
38          //set title
39          holder.bookTitle.setText(books.get(position).getTitle());
40          //initialise on click listener
41          holder.parentLayout.setOnClickListener(new View.OnClickListener() {
42              @Override
43              public void onClick(View v) {
44                  book book = books.get(position);
45                  if (book.getTitle() == "rent a book") {
46                      //if no books added yet perform add book floating action button click
47                      BottomNavigationView navigation = ((Activity)
                        ↪  mContext).findViewById(R.id.navigation);
48                      navigation.setSelectedItemId(R.id.action_search);
49                  } else {
50                      //go to view book class
51                      Intent intent = new Intent(mContext, viewSearchBookActivity.class);
52                      intent.putExtra("book", book);
53                      mContext.startActivity(intent);
54                  }
55              }
56          });
57      }
58      //method to return array size
59      @Override
60      public int getItemCount() {
61          return books.size();
62      }
63      //boilerplate class to represent layout
64      class ViewHolder extends RecyclerView.ViewHolder {
65          TextView bookTitle;
66          CardView parentLayout;
67          TextView bookStatus;
68          ViewHolder(View itemView) {
69              super(itemView);
70              bookTitle = itemView.findViewById(R.id.bookTitle);
71              parentLayout = itemView.findViewById(R.id.parentLayout);
72              bookStatus = itemView.findViewById(R.id.bookStatus);
73          }
74      }
75  }
```

books Fragment I can now return to this class to implement the initScannedBooksRecyclerView and initRentingBooksRecylerView methods.

```
//method to initialise recycle view
private void initScannedBooksRecyclerView(View view, ArrayList<book> books) {
    RecyclerView recyclerView = view.findViewById(R.id.scannedBooksView);
    scannedBooksRecyclerViewAdapter adapter = new scannedBooksRecyclerViewAdapter(getContext(),
    ↪  books);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
}

private void initRentingBooksRecylerView(View view, ArrayList<book> books) {
    RecyclerView recyclerView = view.findViewById(R.id.rentingBooksView);
    rentingBooksRecyclerViewAdapter adapter = new rentingBooksRecyclerViewAdapter(getContext(),
    ↪  books);
    recyclerView.setAdapter(adapter);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
}
```

Now that I have set up these class and methods the recycleView should be ready to be populated from data fetched within the getScannedBookTitles and getRentedBookTitles methods:

```
private void getRentedBookTitles(final View view) {
    final FirebaseFirestore db = FirebaseFirestore.getInstance();
    final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    final String userID = user.getUid();
    db.collection("books").whereEqualTo("rentingID", userID).get().addOnCompleteListener(new
    ↪  OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                ArrayList<book> books = new ArrayList<>();
                for (QueryDocumentSnapshot document : task.getResult()) {
                    book book = document.toObject(com.bookstore.palvindersingh.book.class);
                    book.setFirestoreID(document.getId());
                    books.add(book);
                }
                if (books.size() == 0) {
                    books.add(new book(null, null, null, null, "rent a book", null, false, false,
                    ↪  null));
                }
                initRentingBooksRecylerView(view, books);
            }
        }
    });
}
//method to get all books
private void getScannedBookTitles(final View view) {
    //initialise firestore connection
    final FirebaseFirestore db = FirebaseFirestore.getInstance();
    final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    final String userID = user.getUid();
    //get all associated books
    db.collection("books").whereEqualTo("ownerReference", userID).get().addOnCompleteListener(new
    ↪  OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                ArrayList<book> books = new ArrayList<>();
```

```
35                  for (QueryDocumentSnapshot document : task.getResult()) {
36                      book book = document.toObject(book.class);
37                      book.setFirestoreID(document.getId());
38                      books.add(book);
39                  }
40                  //single book class to indicate no books have been added yet
41                  if (books.size() == 0) {
42                      books.add(new book(null, null, null, null, "add a book", null, false, false,
                        ↪  null));
43                  }
44                  //initialise recycle view
45                  initScannedBooksRecyclerView(view, books);
46              }
47          }
48      });
49  }
```

As we can see I have Firestore's query functionalities. I have also included in both methods the implementation of the case in which no book has been found whereby an empty/default tabbed item is displayed. I can now test whether the recycle views work. However I cannot click on them as I have not yet implemented the view book activities. I will use an account with a single book scanned. I will also manually add a book to firebase that meets the requirements of a rented book (change the attributes of a book and added to the user borrowed books array). It should therefore appear in the rented books recycle view. I have to do this because I have not yet created the interfaces that allow for requesting books. The results are successful.

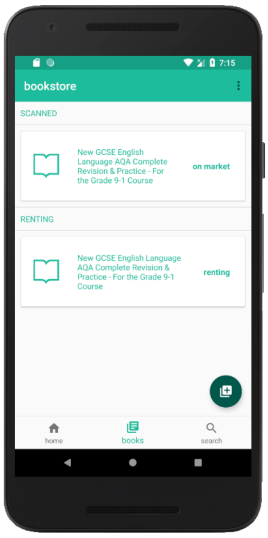


Figure 4.12: Recycle Views

4.3.3 View Book Activity

I must first create the ui for this activity. It resembles quite closely the ui that I create for displaying meta data about a book. For this reason I have replicated to ui design. This should make it familiar and consistent amongst users.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
   ↪  xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
```

```
6       android:layout_height="match_parent"
7       tools:context=".viewBookActivity">
8       <android.support.v7.widget.Toolbar
9           android:id="@+id/my_child_toolbar"
10          style="@style/toolbarTheme"
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:theme="@style/AppTheme"
14          app:titleTextColor="@color/white" />
15      <ScrollView
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18          app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">
19          <LinearLayout
20              android:layout_width="match_parent"
21              android:layout_height="wrap_content"
22              android:orientation="vertical"
23              android:paddingLeft="50dp"
24              android:paddingTop="50dp"
25              android:paddingRight="50dp">
26              <ImageView
27                  android:id="@+id/bookImage"
28                  android:layout_width="125dp"
29                  android:layout_height="wrap_content"
30                  android:layout_gravity="center"
31                  android:adjustViewBounds="true"
32                  android:contentDescription="@string/books"
33                  android:scaleType="fitXY"
34                  android:visibility="gone" />
35              <LinearLayout
36                  android:layout_width="match_parent"
37                  android:layout_height="wrap_content"
38                  android:layout_marginTop="50dp"
39                  android:background="@color/browser_actions_bg_grey"
40                  android:divider="?android:listDivider"
41                  android:orientation="vertical"
42                  android:showDividers="middle">
43                  <TextView
44                      android:layout_width="match_parent"
45                      android:layout_height="wrap_content"
46                      android:elevation="10dp"
47                      android:fontFamily="@font/arimo"
48                      android:padding="10dp"
49                      android:text="About"
50                      android:textColor="#000"
51                      android:textSize="15sp" />
52                  <TextView
53                      android:id="@+id/isbn"
54                      android:layout_width="match_parent"
55                      android:layout_height="wrap_content"
56                      android:elevation="10dp"
57                      android:fontFamily="@font/arimo"
58                      android:padding="10dp"
59                      android:text="@string/isbn"
60                      android:textColor="@color/colorPrimaryDark"
61                      android:textSize="15sp"
62                      android:visibility="gone" />
63                  <TextView
64                      android:id="@+id/bookTitle"
65                      android:layout_width="match_parent"
66                      android:layout_height="wrap_content"
```

```
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="@string/title"
                    android:textAllCaps="true"
                    android:textColor="@color/colorPrimaryDark"
                    android:textSize="15sp"
                    android:visibility="gone" />
                <TextView
                    android:id="@+id/bookAuthors"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="@string/authors"
                    android:textAllCaps="true"
                    android:textColor="@color/colorPrimaryDark"
                    android:textSize="15sp"
                    android:visibility="gone" />
                <TextView
                    android:id="@+id/bookGenres"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="@string/genres"
                    android:textAllCaps="true"
                    android:textColor="@color/colorPrimaryDark"
                    android:textSize="15sp"
                    android:visibility="gone" />
                <TextView
                    android:id="@+id/bookMarket"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="@string/market"
                    android:textAllCaps="true"
                    android:textColor="@color/colorPrimaryDark"
                    android:textSize="15sp"
                    android:textStyle="bold"
                    android:visibility="gone" />
            </LinearLayout>
        </LinearLayout>
    </ScrollView>
    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:clickable="true"
        android:focusable="true"
        android:src="@drawable/ic_delete_24dp"
        android:visibility="visible"
        app:backgroundTint="@color/fui_bgEmail"
        app:fabSize="normal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/market"
```

```
128         android:layout_width="wrap_content"
129         android:layout_height="wrap_content"
130         android:layout_margin="16dp"
131         android:layout_marginEnd="8dp"
132         android:layout_marginBottom="8dp"
133         android:clickable="true"
134         android:focusable="true"
135         android:src="@drawable/ic_local_grocery_store_24dp"
136         android:visibility="visible"
137         app:backgroundTint="@color/white"
138         app:fabSize="normal"
139         app:layout_constraintBottom_toTopOf="@id/delete"
140         app:layout_constraintEnd_toEndOf="parent" />
141     <TextView
142         android:id="@+id/deletename"
143         android:layout_width="wrap_content"
144         android:layout_height="wrap_content"
145         android:layout_marginEnd="8dp"
146         android:fontFamily="@font/arimo"
147         android:padding="10dp"
148         android:text="Delete"
149         android:textColor="#000"
150         android:textSize="15sp"
151         app:layout_constraintBottom_toBottomOf="@+id/delete"
152         app:layout_constraintEnd_toStartOf="@+id/delete"
153         app:layout_constraintTop_toTopOf="@+id/delete" />
154     <TextView
155         android:id="@+id/marketname"
156         android:layout_width="wrap_content"
157         android:layout_height="wrap_content"
158         android:layout_marginEnd="8dp"
159         android:fontFamily="@font/arimo"
160         android:padding="10dp"
161         android:text="Market"
162         android:textColor="#000"
163         android:textSize="15sp"
164         app:layout_constraintBottom_toBottomOf="@+id/market"
165         app:layout_constraintEnd_toStartOf="@+id/market"
166         app:layout_constraintTop_toTopOf="@+id/market" />
167 </android.support.constraint.ConstraintLayout>
```

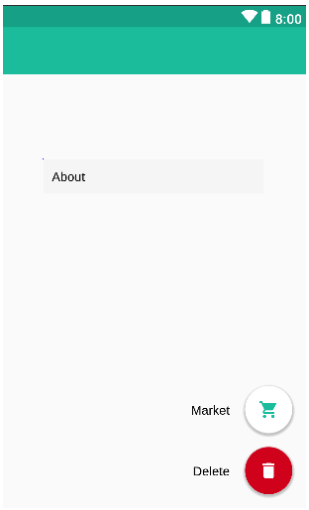


Figure 4.13: Recycle Views

As you can see I have hidden various aspects of the ui. These will be made visible as appropriate when the activitiy is run depending on whether the book's data is null or not. Upon generating this activity my ide also created an activity class for me. I then implemented the code I have previously designed.

```
1  package com.bookstore.palvindersingh;
2  import android.annotation.SuppressLint;
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.support.annotation.NonNull;
6  import android.support.v7.app.ActionBar;
7  import android.support.v7.app.AppCompatActivity;
8  import android.support.v7.widget.Toolbar;
9  import android.view.View;
10 import android.widget.ImageView;
11 import android.widget.TextView;
12 import android.widget.Toast;
13 import com.google.android.gms.tasks.OnFailureListener;
14 import com.google.android.gms.tasks.OnSuccessListener;
15 import com.google.firebase.auth.FirebaseAuth;
16 import com.google.firebase.auth.FirebaseUser;
17 import com.google.firebase.firestore.DocumentReference;
18 import com.google.firebase.firestore.FieldValue;
19 import com.google.firebase.firestore.FirebaseFirestore;
20 import com.squareup.picasso.Picasso;
21 import java.util.ArrayList;
22 public class viewBookActivity extends AppCompatActivity {
23     @SuppressLint("SetTextI18n")
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_view_book);
28         //initialise toolbar
29         Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
30         myChildToolbar.setTitle("book details");
31         setSupportActionBar(myChildToolbar);
32         ActionBar ab = getSupportActionBar();
33         assert ab != null;
34         ab.setDisplayHomeAsUpEnabled(true);
35         //initialise book object serialised within the intent
36         final book meta = (book) getIntent().getSerializableExtra("book");
```

```
37          networkFAB deleteBook = findViewById(R.id.delete);
38          //display all data if it exists
39          if (meta.getIsbn() != null) {
40              TextView bookISBN = findViewById(R.id.isbn);
41              bookISBN.setText(meta.getIsbn());
42              bookISBN.setVisibility(View.VISIBLE);
43          }
44          if (meta.getTitle() != null) {
45              TextView bookTitle = findViewById(R.id.bookTitle);
46              bookTitle.setText(meta.getTitle());
47              bookTitle.setVisibility(View.VISIBLE);
48          }
49          if (meta.getAuthors() != null) {
50              TextView bookAuthors = findViewById(R.id.bookAuthors);
51              bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
52              bookAuthors.setVisibility(View.VISIBLE);
53          }
54          if (meta.getGenres() != null) {
55              TextView bookGenres = findViewById(R.id.bookGenres);
56              bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
57              bookGenres.setVisibility(View.VISIBLE);
58          }
59          if (meta.getImage() != null) {
60              ImageView bookImage = findViewById(R.id.bookImage);
61              Picasso.get().load(meta.getImage()).into(bookImage);
62              bookImage.setContentDescription(meta.getImage());
63              bookImage.setVisibility(View.VISIBLE);
64          }
65          networkFAB market = findViewById(R.id.market);
66          //display book status
67          TextView bookMarket = findViewById(R.id.bookMarket);
68          TextView marketname = findViewById(R.id.marketname);
69          TextView deletename = findViewById(R.id.deletename);
70          bookMarket.setVisibility(View.VISIBLE);
71          if (!meta.getMarket()) {
72              if (meta.getRenting()) {
73                  bookMarket.setText("on rent");
74                  deleteBook.setVisibility(View.INVISIBLE);
75                  market.setVisibility(View.INVISIBLE);
76                  marketname.setVisibility(View.INVISIBLE);
77                  deletename.setVisibility(View.INVISIBLE);
78              } else {
79                  bookMarket.setText("not on the market");
80                  deleteBook.setVisibility(View.VISIBLE);
81                  market.setVisibility(View.VISIBLE);
82              }
83          } else {
84              bookMarket.setText("on the market");
85              market.setVisibility(View.VISIBLE);
86          }
87          //display delete button if it is not on rent
88          if (!meta.getRenting()) {
89              deleteBook.setVisibility(View.VISIBLE);
90          }
91          //initialise on click listener
92          View.OnClickListener deleteEvent = new View.OnClickListener() {
93              @Override
94              public void onClick(View v) {
95                  //initialise firestore connection
96                  final FirebaseFirestore db = FirebaseFirestore.getInstance();
97                  final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
```

```
98                  final String userID = user.getUid();
99                  //delete document
100                 db.collection("books").document(meta.getFirestoreID())
101                         .delete()
102                         .addOnSuccessListener(new OnSuccessListener<Void>() {
103                             @Override
104                             public void onSuccess(Void aVoid) {
105                                 //delete data from user document
106                                 DocumentReference docRef =
                                    ↪  db.collection("users").document(user.getUid());
107                                 docRef.update("scannedBooks",
                                    ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
108                                 //go to home activity
109                                 Toast.makeText(getApplicationContext(), "deleted " +
                                    ↪  meta.getTitle(), Toast.LENGTH_SHORT).show();
110                                 Intent intent = new Intent(viewBookActivity.this,
                                    ↪  homeActivity.class);
111                                 startActivity(intent);
112                                 finish();
113                             }
114                         })
115                         .addOnFailureListener(new OnFailureListener() {
116                             @Override
117                             public void onFailure(@NonNull Exception e) {
118                                 Toast.makeText(getApplicationContext(), "failed to delete " +
                                    ↪  meta.getTitle(), Toast.LENGTH_SHORT).show();
119                             }
120                         });
121             }
122         };
123         deleteBook.setClickEvent(deleteEvent);
124         //initialise on click listener
125         View.OnClickListener clickEventMarket = new View.OnClickListener() {
126             @Override
127             public void onClick(View v) {
128                 //identify book status
129                 if (!meta.getMarket()) {
130                     if (!meta.getRenting()) {
131                         //put on market
132                         final FirebaseFirestore db = FirebaseFirestore.getInstance();
133                         final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
134                         final String userID = user.getUid();
135                         db.collection("books").document(meta.getFirestoreID())
136                                 .update("market", true)
137                                 .addOnSuccessListener(new OnSuccessListener<Void>() {
138                                     @Override
139                                     public void onSuccess(Void aVoid) {
140                                         Toast.makeText(getApplicationContext(), "on the market",
                                            ↪  Toast.LENGTH_SHORT).show();
141                                         Intent intent = new Intent(viewBookActivity.this,
                                            ↪  homeActivity.class);
142                                         startActivity(intent);
143                                         finish();
144                                     }
145                                 })
146                                 .addOnFailureListener(new OnFailureListener() {
147                                     @Override
148                                     public void onFailure(@NonNull Exception e) {
149                                         //failed
150                                         Toast.makeText(getApplicationContext(),
                                            ↪  "something went wrong", Toast.LENGTH_SHORT).show();
```

```
151                                     }
152                                 });
153                     }
154                 } else {
155                     //take off market
156                     final FirebaseFirestore db = FirebaseFirestore.getInstance();
157                     final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
158                     final String userID = user.getUid();
159                     db.collection("books").document(meta.getFirestoreID())
160                             .update("market", false)
161                             .addOnSuccessListener(new OnSuccessListener<Void>() {
162                                 @Override
163                                 public void onSuccess(Void aVoid) {
164                                     Toast.makeText(getApplicationContext(), "off the market",
                                        ↪  Toast.LENGTH_SHORT).show();
165                                     Intent intent = new Intent(viewBookActivity.this,
                                        ↪  homeActivity.class);
166                                     startActivity(intent);
167                                     finish();
168                                 }
169                             })
170                             .addOnFailureListener(new OnFailureListener() {
171                                 @Override
172                                 public void onFailure(@NonNull Exception e) {
173                                     //failed
174                                     Toast.makeText(getApplicationContext(), "something went wrong",
                                        ↪  Toast.LENGTH_SHORT).show();
175                                 }
176                             });
177                 }
178             }
179         };
180         market.setClickEvent(clickEventMarket);
181     }
182     //convert array to readable string
183     private String arrayToDisplayString(ArrayList data) {
184         String string = "";
185         for (int i = 0; i < data.size(); i++) {
186             string = string + data.get(i);
187             if (i != data.size() - 1) {
188                 string = string + " ";
189             }
190         }
191         return string;
192     }
193 }
```

In term of loading ui there is nothing significantly special here, as with the xml it was a case of replicating a previously designed activity. I did add a ui element which declares what state the book is in. I then implemented this with simple selection statements. I have also instantiated a delete onClickListener object that is passed as an attribute to my previously designed floating action button. The Firestore methods that were called are the same as those I discussed before. I also added a Toast to confirm to the user the book has indeed been deleted or if not an error occured. I did the same for the market floating action button. This method simply toggles data about the book between 2 states, true and false, whilst updating user's details. Again a Toast is ouput signifying success or failure in this action. I have ensured that both floating action buttons are made invisible when the book is on rent. In order to test this I have provided a recording[1], this can be found in the footer. At first when I ran the application, it failed to compile. I soon realized this was due to misplaced semi-colons, thus resulting in the syntax error. When selecting

[1] $imp2TEST3 - 4.webm$

a book in the scanned book recycle view, the following error was received:

```
E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.bookstore.palvindersingh, PID: 511
    java.lang.RuntimeException: Unable to start activity
    ↪  ComponentInfo{com.bookstore.palvindersingh/com.bookstore.palvindersingh.viewBookActivity}:
    ↪  java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.String
    ↪  com.bookstore.palvindersingh.book.getIsbn()' on a null object reference
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2778)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2856)
        at android.app.ActivityThread.-wrap11(Unknown Source:0)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1589)
        at android.os.Handler.dispatchMessage(Handler.java:106)
        at android.os.Looper.loop(Looper.java:164)
        at android.app.ActivityThread.main(ActivityThread.java:6494)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:438)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:807)
     Caused by: java.lang.NullPointerException: Attempt to invoke virtual method 'java.lang.String
     ↪  com.bookstore.palvindersingh.book.getIsbn()' on a null object reference
        at com.bookstore.palvindersingh.viewBookActivity.onCreate(viewBookActivity.java:47)
        at android.app.Activity.performCreate(Activity.java:7009)
        at android.app.Activity.performCreate(Activity.java:7000)
        at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1214)
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2731)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2856)
        at android.app.ActivityThread.-wrap11(Unknown Source:0)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1589)
        at android.os.Handler.dispatchMessage(Handler.java:106)
        at android.os.Looper.loop(Looper.java:164)
        at android.app.ActivityThread.main(ActivityThread.java:6494)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:438)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:807)
```

This is a lengthy error that at first looked intimidating as it inferred that I had done something disastrously wrong. Reading the errors closely, I found that this was because of the abstracted nature of android development. The classes that I am using to illustrate activities or other ui components all have their own associated packages. Thus an error will propagate throughout all of these abstractions, in turn producing the lengthy error we have above. The error is essentially telling me that my instruction to get a book object from an intent which called the activity is void, i.e it does not exist. I went back an checked if indeed passed the book class into the intent within the scannedBookRecycleViewAdapter class. I had not; the addition of the line below fixed my issue.

```
intent.putExtra("book", book);
```

I then ran the application with successful compilation. As we can see from the recording the activity runs as expected. I can also checked Firestore console to confirm this.

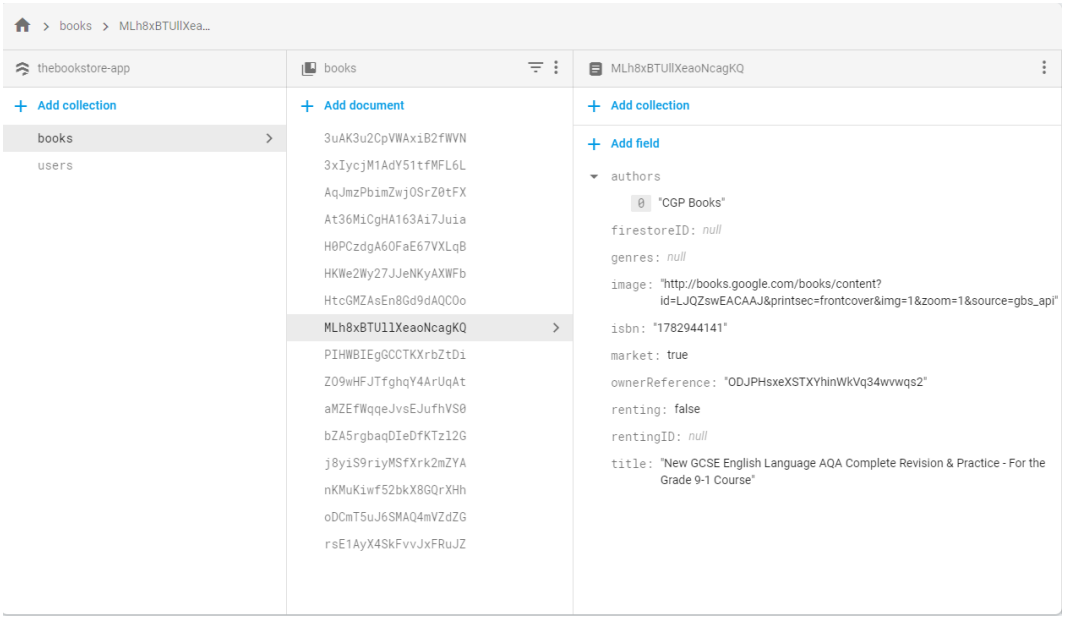


Figure 4.14: Firestore Console

4.3.4 Search Fragment

This activity is a large task thus I will break it's implementation into stages. First I will create a recycleview adapter for the recyle view. Then I will create the ui and implement the recycle view. Once doing this I can create sortBooks and getSearchResults methods. After this is done I can implement these methods in the fragment's onCreateView override mehod, by connected them to an onClickListener assigned to a floating action button.

searchBooksRecyclerViewAdapter Implementing this class is yet again similar to what I have already done. The only difference between this class and previous implementations of adapters is that I want the tabbed item status text field not to be visible. Otherwise outputting the title of the book is the same requirement. Given the similarities I should be able to implement this class directly:

```
1  package com.bookstore.palvindersingh;
2  import android.content.Context;
3  import android.content.Intent;
4  import android.support.annotation.NonNull;
5  import android.support.v7.widget.CardView;
6  import android.support.v7.widget.RecyclerView;
7  import android.view.LayoutInflater;
8  import android.view.View;
9  import android.view.ViewGroup;
10 import android.widget.TextView;
11 import java.util.ArrayList;
12 public class searchBooksRecyclerViewAdapter extends
   ↪  RecyclerView.Adapter<searchBooksRecyclerViewAdapter.ViewHolder> {
13     //initialise attributes
14     private final ArrayList<book> books;
15     private Context mContext;
16     //boilerplate methods
17     searchBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
18         this.books = books;
19         this.mContext = mContext;
20     }
21     @NonNull
```

```
22      @Override
23      public searchBooksRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
        ↪  int viewType) {
24          //inflate layout
25          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item,
            ↪  parent, false);
26          return new searchBooksRecyclerViewAdapter.ViewHolder(view);
27      }
28      @Override
29      public void onBindViewHolder(@NonNull searchBooksRecyclerViewAdapter.ViewHolder holder, final
        ↪  int position) {
30          book book = books.get(position);
31          //set book data
32          holder.bookTitle.setText(book.getTitle());
33          holder.bookStatus.setVisibility(View.GONE);
34          //initialise on click listener
35          holder.parentLayout.setOnClickListener(new View.OnClickListener() {
36              @Override
37              public void onClick(View v) {
38                  book book = books.get(position);
39                  //go to view searched book activity
40                  Intent intent = new Intent(mContext, viewSearchBookActivity.class);
41                  intent.putExtra("book", book);
42                  mContext.startActivity(intent);
43              }
44          });
45      }
46      //method to return array size
47      @Override
48      public int getItemCount() {
49          return books.size();
50      }
51      //boilerplate class to represent layout
52      class ViewHolder extends RecyclerView.ViewHolder {
53          TextView bookTitle;
54          CardView parentLayout;
55          TextView bookStatus;
56
57          ViewHolder(View itemView) {
58              super(itemView);
59              bookTitle = itemView.findViewById(R.id.bookTitle);
60              parentLayout = itemView.findViewById(R.id.parentLayout);
61              bookStatus = itemView.findViewById(R.id.bookStatus);
62          }
63      }
64  }
```

I then added the following method to the Search Fragment which allows interaction with the ui.

```
1   private void initSearchBooksRecyclerView(View view, ArrayList<book> books) {
2       RecyclerView recyclerView = view.findViewById(R.id.searchBooksView);
3       searchBooksRecyclerViewAdapter adapter = new searchBooksRecyclerViewAdapter(getContext(),
        ↪  books);
4       recyclerView.setAdapter(adapter);
5       recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
6   }
```

Search Fragment I started this stage by first building the necessary ui components. From the design drawings I created the following ui.

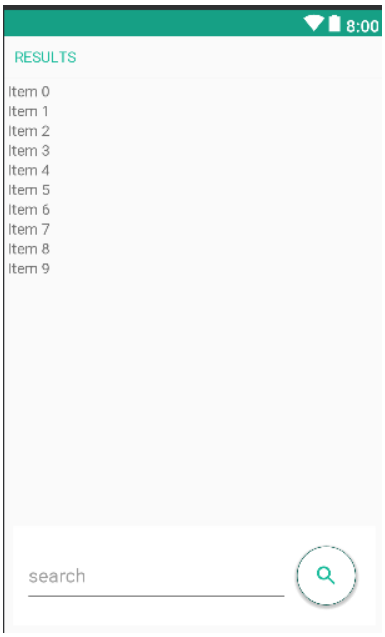


Figure 4.15: Search Fragment UI

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
   ↪  xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".searchFragment">
8      <ScrollView
9          android:layout_width="match_parent"
10         android:layout_height="wrap_content">
11         <LinearLayout
12             android:layout_width="match_parent"
13             android:layout_height="wrap_content"
14             android:divider="?android:listDivider"
15             android:orientation="vertical"
16             android:showDividers="middle">
17             <TextView
18                 android:id="@+id/results"
19                 android:layout_width="match_parent"
20                 android:layout_height="wrap_content"
21                 android:padding="10dp"
22                 android:text="@string/results"
23                 android:textAllCaps="true"
24                 android:textColor="@color/colorPrimary" />
25             <android.support.v7.widget.RecyclerView
26                 android:id="@+id/searchBooksView"
27                 android:layout_width="match_parent"
28                 android:layout_height="wrap_content"
29                 android:layout_marginBottom="100dp"
30                 android:padding="3dp" />
31         </LinearLayout>
32     </ScrollView>
33     <android.support.constraint.ConstraintLayout
34         android:layout_width="match_parent"
35         android:layout_height="wrap_content"
```

```
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:background="@color/white"
        android:orientation="horizontal"
        android:padding="3dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">
        <com.bookstore.palvindersingh.networkFAB
            android:id="@+id/floating_action_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:backgroundTint="@color/white"
            android:src="@drawable/ic_search_24dp"
            app:fabSize="normal"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
        <EditText
            android:id="@+id/search"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_gravity="bottom|end"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:hint="@string/search"
            app:layout_constraintBottom_toBottomOf="@+id/floating_action_button"
            app:layout_constraintEnd_toStartOf="@+id/floating_action_button"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="@+id/floating_action_button" />
    </android.support.constraint.ConstraintLayout>
</android.support.constraint.ConstraintLayout>
```

I can now use this as a foundation to begin impelemting my sortBooks and getSearchResults methods. I can then incorportate these into the onCreateView override method. The first method to implement is getSearchResults. This method takes a string called input as a parameter. Within this class I will ultimately need to call sortBooks in order to sort the array of books that I ultimately want to query in this method. Within the sortBook class once the books have been sorted, I will call the initSearchBooksRecyclerView class. This class requires a view object to be passed as a parameter. This will allow the searchBooksRecyclerViewAdapter class to interact with the activity, i.e it brings the instantiated class in scope of the activity. Thus I need to add a parameter of a view object into the getSearchResults class. As we have done before, we will query the books collection, this time for the string parameter input (which should be an isbn). Only books with a market value of true should be queried. This compound query can be easily done in Firestore. Once complete I will perform simple checks to see if the books are present or not and accordingly sort the data or add an empty book class.

```
private void getSearchResults(String input, final View view) {
    //initialise firestore connection
    final FirebaseFirestore db = FirebaseFirestore.getInstance();
    final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    //query all documents on the market with the input isbn
    db.collection("books").whereEqualTo("isbn", input).whereEqualTo("market",
    ↪  true).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @SuppressLint("SetTextI18n")
```

```
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                ArrayList<book> books = new ArrayList<>();
                String userid = user.getUid();
                for (QueryDocumentSnapshot document : task.getResult()) {
                    book book = document.toObject(book.class);
                    book.setFirestoreID(document.getId());
                    if (!book.getOwnerReference().equals(userid)) {
                        //if the user does not own the book add it to an array
                        books.add(book);
                    }
                }
                //if no books are found
                if (books.size() == 0) {
                    //add a boilerplate book class to indicate no books found
                    books.add(new book(null, null, null, null, "no books found", null, false, false,
                    ↪  null));
                }
                if (books.get(0).getTitle() != "no books found") {
                    sortBooks(books, view);
                }
                TextView results = view.findViewById(R.id.results);
                results.setText("RESULTS: " + books.size());
            }
        }
    });
}
```

I can now move onto the implementation of the sortBooks method. This method seems far more challenging than the rest as it has come to my attention that Firestore tasks occur asynchnorously meaning that instructions that are declared after sortBook method may/most probably will be executed. This leaves me the challenge of incorporating an insertion sort into the scope of the async task. Furthermore, as we have seen before when making a Firestore query an onSuccessListener is required. This is essentially a paramter that is a class it self. This class can be declared directly as part of an argument, whereby code to be executed after the Firestore task is successful is input within an override method called onComplete. I also need to continue passing the the view object into the initSearchBooksRecyclerView method. I have also decided to declare an attribute called bookList. This will store the latest sorted list of books.

```
private void sortBooks(final ArrayList<book> books, final View view) {
        final ArrayList ratings = new ArrayList();
        final FirebaseFirestore db = FirebaseFirestore.getInstance();
        for (book book : books) {
            String owner = book.getOwnerReference();
            final ArrayList data = new ArrayList();
            data.add(book);
            db.collection("users").document(owner).get().addOnCompleteListener(new
            ↪  OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    if (task.isSuccessful()) {
                        DocumentSnapshot document = task.getResult();
                        if (document.exists()) {
                            data.add(document.get("userRating"));
                            ratings.add(data);
                        }
                    } else {
                        data.add(0);
                        ratings.add(data);
```

```
20                      }
21                      if (ratings.size() == books.size()) {
22                          int n = ratings.size();
23                          for (int i = 1; i < n; i++) {
24                              ArrayList temp = (ArrayList) ratings.get(i);
25                              int j = i + 1;
26                              ArrayList nextArray = (ArrayList) ratings.get(j);
27                              Double tempRating = Double.valueOf(temp.get(1).toString());
28                              Double nextRating = Double.valueOf(temp.get(1).toString());
29                              while (j >= 0 && nextRating > tempRating) {
30                                  ratings.set(j + 1, ratings.get(j));
31                                  j = j - 1;
32                              }
33                              ratings.set(j + 1, temp);
34                          }
35                          ArrayList<book> sorted = new ArrayList<>();
36                          for (int i = ratings.size() - 1; i > -1; i--) {
37                              sorted.add((book) ((ArrayList) ratings.get(i)).get(0));
38                              System.out.println(((String)((ArrayList) ratings.get(i)).get(1)));
39                          }
40                          bookList = sorted;
41                          initSearchBooksRecyclerView(view, bookList);
42                      }
43                  }
44              });
45          }
```

As you can see I have added an output statement on line 38; this should print the ratings of the books in the new order so that I can verify that books have been ordered in terms of ratings. Now all I must do is implement the onCreateView method. This is straight forward, all I must do is declare an onClickListener that gets the value within the input text field then passes it into getSearchResults method

```
1   @Override
2   public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
3                            Bundle savedInstanceState) {
4       Log.e("books", "fragment newInstance");
5       final View view = inflater.inflate(R.layout.fragment_search, container, false);
6       //initialise ui components
7       networkFAB searchFAB = view.findViewById(R.id.floating_action_button);
8       final EditText searchInput = view.findViewById(R.id.search);
9       //initialise on click event
10      View.OnClickListener clickEvent = new View.OnClickListener() {
11          @Override
12          public void onClick(View v) {
13              //call get search results method
14              getSearchResults(searchInput.getText().toString(), view);
15          }
16      };
17      searchFAB.setClickEvent(clickEvent);
18      return view;
19  }
```

To test this class I ensure there were at least 3 books from 2 different users of ratings 5 and 3. The isbn of the book was 1782944141. The compilation was successful however when it came to searching the book an error was returned and the app crashed:

```
1   E/AndroidRuntime: FATAL EXCEPTION: main
2       Process: com.bookstore.palvindersingh, PID: 2572
```

```
    java.lang.IndexOutOfBoundsException: Index: 2, Size: 2
        at java.util.ArrayList.get(ArrayList.java:437)
        at com.bookstore.palvindersingh.searchFragment$3.onComplete(searchFragment.java:144)
        at com.google.android.gms.tasks.zzj.run(Unknown Source:4)
        at android.os.Handler.handleCallback(Handler.java:790)
        at android.os.Handler.dispatchMessage(Handler.java:99)
        at android.os.Looper.loop(Looper.java:164)
        at android.app.ActivityThread.main(ActivityThread.java:6494)
        at java.lang.reflect.Method.invoke(Native Method)
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:438)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:807)
```

```
ArrayList nextArray = (ArrayList) ratings.get(j);
```

I can track the error down to the above line. The error states that there is an out of bounds exception meaning that the value of j is bigger than the size of the list. As can be seen in the error this occurs on the second run then the j is equal to 2 and the size of the array is also equal to 2. Of course the for an array of 2 value the indexes can only be 0 and 1. Thus 2 would logically give an out of bounds error. In review of my code I spotted that I was incrementing j incorrectly. Most sorting algorithms take a value and compare it with the next/one to the right. Insertion sort works the opposite way round. It takes the first value and places it into an unsorted (in place) array. It then takes iterates through the rest of the values (unsorted array), moving each one two the left as necessary in order to place it directly into it's correct position. Thus the value of j should be one less than the value of i, i have incremented it. The altered code on line 25 is:

```
int j = i - 1;
```

I ran the application again and repeated the process. The values was 4 and 5 were output to the console. I ran the program again to verify the result but this time 5 and 4 were output. This means that a logical error has occurred in my code.

```
I/System.out: 4
 I/zygote: Background concurrent copying GC freed 42307(1997KB) AllocSpace objects, 5(228KB) LOS
  ↪  objects, 50 free, 3MB/7MB, paused 1.067ms total 113.075ms
I/System.out: 5
```

```
I/System.out: 5
    4
```

I soon realized that I had made a mistake of duplicating the same variable value twice for tempRating and nextRating. This meant that the conditions for the while loop (nextRating > tempRating) were never met so no sorting occurs and thus the array is essentially no sorted at all/ retains it original order. I can however take the positive that the code for displaying the recycle view works correctly. After my final fix:

```
Double nextRating = Double.valueOf(nextArray.get(1).toString());
```

I ran the code. It compiled successfuly and upon searching the isbn. The console output 5 and 4 correctly. Books were also shown in the Recycle View as expected.

```
I/System.out: 5
I/System.out: 4
```

I added another book for the user with a rating of 5 and repated the procces to ensure that it indeed sorts the books correctly. Again it correctly output 5,5,4 and generated the recycle view correctly also and update the results string correctly.

```
I/System.out: 5
E/RecyclerView: No adapter attached; skipping layout
I/System.out: 5
I/System.out: 4
```

Alongside output statements, I used the debugging tool breakpoints to monitor the attribute bookList. To do this I placed a breakpoint on the line where the attribute is declared. I then ran the application once so that the breakpoint was met. Then whenever that variable is changed it appears in the debugging console. As you can see the debugging tool successfully picked up the

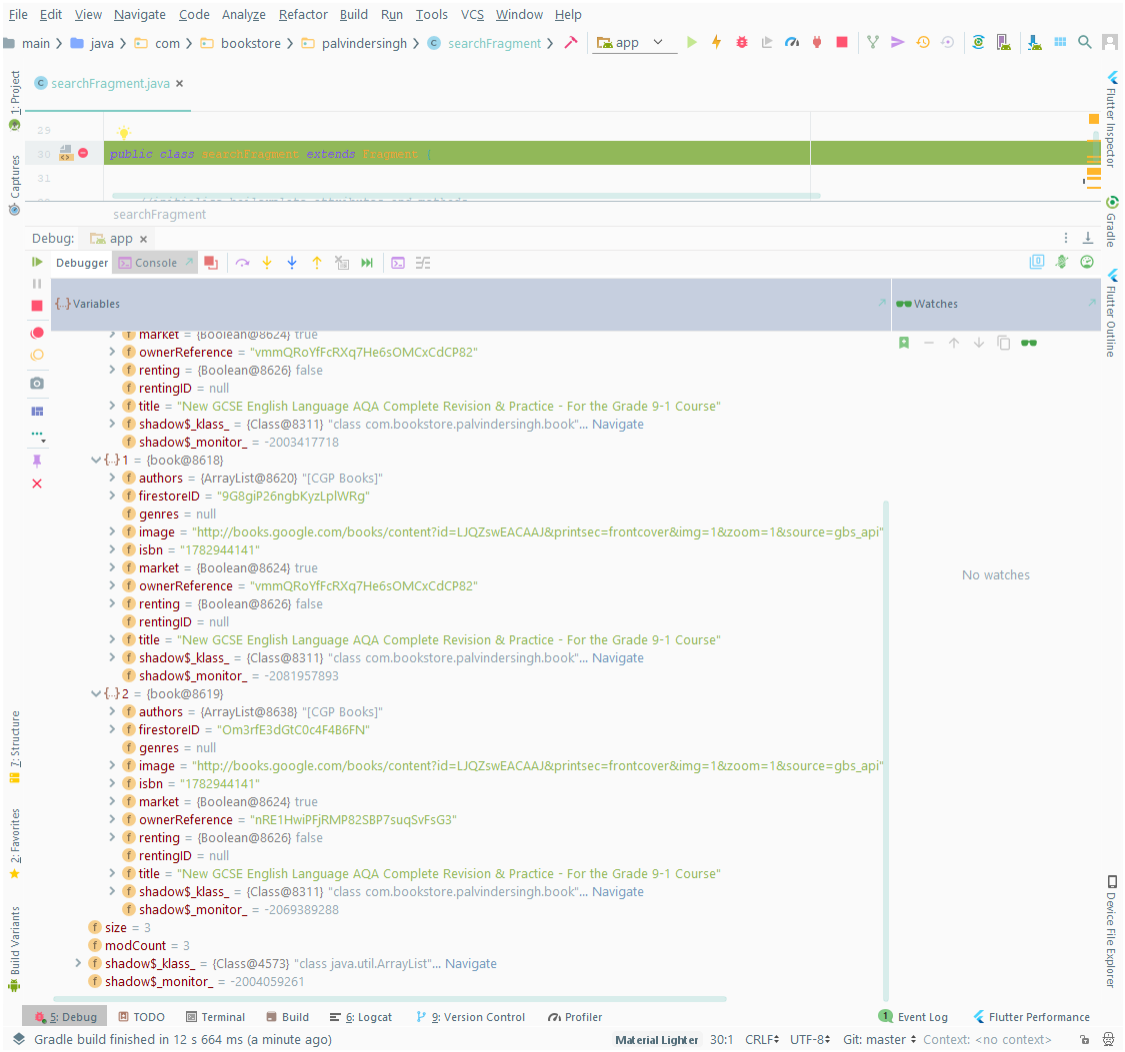


Figure 4.16: Debugger

breakpoint and cached the array correctly. Inspecting the array, I found the books were indeed correctly stored.

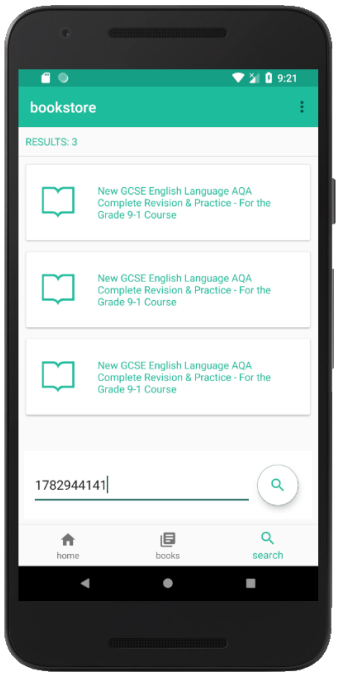


Figure 4.17: Search Fragment UI

4.3.5 View Searched Book Activity

This activity will play a role in two areas of my application. It will serve as a way for renting books via the search feature, it will also allow for books to be removed from renting. Given that a book will be passed in via an intent; as I have indicated within my pseudocode, I will change the state of the activity depending on the book's renting value. If it is false I know that I must show the ui that allows for renting. If it is true I know to show the ui that allows to stop renting. The ui that displays book details and owner details will remain unchanged. It is however the visibility of 2 floating action buttons that will change. I can start to create this ui, following my previously created drawn designs. Given that this is similar to the ui for the viewSearchBook activity all I have to do is replicate that ui and add an extra floating action button.

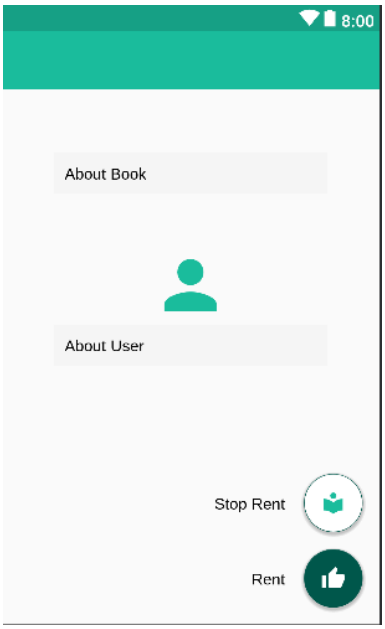


Figure 4.18: View Searched Book UI

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".viewSearchBookActivity">
    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingLeft="50dp"
            android:paddingTop="50dp"
            android:paddingRight="50dp">
            <ImageView
                android:id="@+id/bookImage"
                android:layout_width="125dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:adjustViewBounds="true"
                android:contentDescription="@string/books"
                android:scaleType="fitXY"
                android:visibility="gone" />
            <LinearLayout
                android:layout_width="match_parent"
```

```
37                  android:layout_height="wrap_content"
38                  android:layout_marginTop="10dp"
39                  android:background="@color/browser_actions_bg_grey"
40                  android:divider="?android:listDivider"
41                  android:orientation="vertical"
42                  android:showDividers="middle">
43                  <TextView
44                      android:layout_width="match_parent"
45                      android:layout_height="wrap_content"
46                      android:elevation="10dp"
47                      android:fontFamily="@font/arimo"
48                      android:padding="10dp"
49                      android:text="About Book"
50                      android:textColor="#000"
51                      android:textSize="15sp" />
52                  <TextView
53                      android:id="@+id/isbn"
54                      android:layout_width="match_parent"
55                      android:layout_height="wrap_content"
56                      android:elevation="10dp"
57                      android:fontFamily="@font/arimo"
58                      android:padding="10dp"
59                      android:text="@string/isbn"
60                      android:textColor="@color/colorPrimaryDark"
61                      android:textSize="15sp"
62                      android:visibility="gone" />
63                  <TextView
64                      android:id="@+id/bookTitle"
65                      android:layout_width="match_parent"
66                      android:layout_height="wrap_content"
67                      android:fontFamily="@font/arimo"
68                      android:padding="10dp"
69                      android:text="@string/title"
70                      android:textAllCaps="true"
71                      android:textColor="@color/colorPrimaryDark"
72                      android:textSize="15sp"
73                      android:visibility="gone" />
74                  <TextView
75                      android:id="@+id/bookAuthors"
76                      android:layout_width="match_parent"
77                      android:layout_height="wrap_content"
78                      android:fontFamily="@font/arimo"
79                      android:padding="10dp"
80                      android:text="@string/authors"
81                      android:textAllCaps="true"
82                      android:textColor="@color/colorPrimaryDark"
83                      android:textSize="15sp"
84                      android:visibility="gone" />
85                  <TextView
86                      android:id="@+id/bookGenres"
87                      android:layout_width="match_parent"
88                      android:layout_height="wrap_content"
89                      android:fontFamily="@font/arimo"
90                      android:padding="10dp"
91                      android:text="@string/genres"
92                      android:textAllCaps="true"
93                      android:textColor="@color/colorPrimaryDark"
94                      android:textSize="15sp"
95                      android:visibility="gone" />
96              </LinearLayout>
97              <ImageView
```

```
98                  android:layout_width="75dp"
99                  android:layout_height="wrap_content"
100                 android:layout_gravity="center"
101                 android:layout_marginTop="50dp"
102                 android:adjustViewBounds="true"
103                 android:contentDescription="@string/books"
104                 android:scaleType="fitXY"
105                 android:src="@drawable/ic_person_24dp" />
106             <LinearLayout
107                 android:layout_width="match_parent"
108                 android:layout_height="wrap_content"
109                 android:layout_marginBottom="100dp"
110                 android:background="@color/browser_actions_bg_grey"
111                 android:divider="?android:listDivider"
112                 android:orientation="vertical"
113                 android:showDividers="middle">
114                 <TextView
115                     android:layout_width="match_parent"
116                     android:layout_height="wrap_content"
117                     android:elevation="10dp"
118                     android:fontFamily="@font/arimo"
119                     android:padding="10dp"
120                     android:text="About User"
121                     android:textColor="#000"
122                     android:textSize="15sp" />
123                 <TextView
124                     android:id="@+id/name"
125                     android:layout_width="match_parent"
126                     android:layout_height="wrap_content"
127                     android:elevation="10dp"
128                     android:fontFamily="@font/arimo"
129                     android:padding="10dp"
130                     android:text="name"
131                     android:textColor="@color/colorPrimaryDark"
132                     android:textSize="15sp"
133                     android:visibility="gone" />
134                 <TextView
135                     android:id="@+id/year"
136                     android:layout_width="match_parent"
137                     android:layout_height="wrap_content"
138                     android:fontFamily="@font/arimo"
139                     android:padding="10dp"
140                     android:text="@string/year_group"
141                     android:textAllCaps="true"
142                     android:textColor="@color/colorPrimaryDark"
143                     android:textSize="15sp"
144                     android:visibility="gone" />
145                 <TextView
146                     android:id="@+id/form"
147                     android:layout_width="match_parent"
148                     android:layout_height="wrap_content"
149                     android:fontFamily="@font/arimo"
150                     android:padding="10dp"
151                     android:text="@string/form_room"
152                     android:textAllCaps="true"
153                     android:textColor="@color/colorPrimaryDark"
154                     android:textSize="15sp"
155                     android:visibility="gone" />
156                 <TextView
157                     android:id="@+id/rating"
158                     android:layout_width="match_parent"
```

```
159                     android:layout_height="wrap_content"
160                     android:fontFamily="@font/arimo"
161                     android:padding="10dp"
162                     android:text="rating"
163                     android:textAllCaps="true"
164                     android:textColor="@color/colorPrimaryDark"
165                     android:textSize="15sp"
166                     android:visibility="gone" />
167             </LinearLayout>
168         </LinearLayout>
169     </ScrollView>
170     <com.bookstore.palvindersingh.networkFAB
171         android:id="@+id/request"
172         android:layout_width="wrap_content"
173         android:layout_height="wrap_content"
174         android:layout_margin="16dp"
175         android:layout_marginEnd="8dp"
176         android:layout_marginBottom="8dp"
177         android:clickable="true"
178         android:focusable="true"
179         android:src="@drawable/ic_thumb_up_24dp"
180         android:visibility="visible"
181         app:fabSize="normal"
182         app:layout_constraintBottom_toBottomOf="parent"
183         app:layout_constraintEnd_toEndOf="parent" />
184     <com.bookstore.palvindersingh.networkFAB
185         android:id="@+id/rent"
186         android:layout_width="wrap_content"
187         android:layout_height="wrap_content"
188         android:layout_margin="16dp"
189         android:layout_marginBottom="8dp"
190         android:backgroundTint="@color/white"
191         android:clickable="true"
192         android:focusable="true"
193         android:src="@drawable/ic_local_library"
194         android:visibility="visible"
195         app:fabSize="normal"
196         app:layout_constraintBottom_toTopOf="@+id/request"
197         app:layout_constraintEnd_toEndOf="@+id/request"
198         app:layout_constraintStart_toStartOf="@+id/request" />
199     <TextView
200         android:id="@+id/rentingtext"
201         android:layout_width="wrap_content"
202         android:layout_height="wrap_content"
203         android:layout_marginEnd="8dp"
204         android:fontFamily="@font/arimo"
205         android:padding="10dp"
206         android:text="Rent"
207         android:textColor="#000"
208         android:textSize="15sp"
209         app:layout_constraintBottom_toBottomOf="@+id/request"
210         app:layout_constraintEnd_toStartOf="@+id/request"
211         app:layout_constraintTop_toTopOf="@+id/request" />
212     <TextView
213         android:id="@+id/rentingtext2"
214         android:layout_width="wrap_content"
215         android:layout_height="wrap_content"
216         android:layout_marginEnd="8dp"
217         android:fontFamily="@font/arimo"
218         android:padding="10dp"
219         android:text="Stop Rent"
```

```
        android:textColor="#000"
        android:textSize="15sp"
        android:visibility="visible"
        app:layout_constraintBottom_toBottomOf="@+id/rent"
        app:layout_constraintEnd_toStartOf="@+id/rent"
        app:layout_constraintTop_toTopOf="@+id/rent" />
</android.support.constraint.ConstraintLayout>
```

For the above code I made the stop rent floating action button and text view visible so that it can be seen within the layout editor. I have changed it to invisible so that it can only be seen if certain conditions are met. As part of this implementation I will need the method arrayToDisplayString again. This is because certain attributes of the book class, genre and author, are arrays thus I need to convert it to a readable string.

```
private String arrayToDisplayString(ArrayList data) {
    String string = "";
    for (int i = 0; i < data.size(); i++) {
        string = string + data.get(i);
        if (i != data.size() - 1) {
            string = string + " ";
        }
    }
    return string;
}
```

The rest of my code will be stored within the onCreateView method. I must first initialise the toolbar.

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_search_book);
        //initialise toolbar
        Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
        myChildToolbar.setTitle("book details");
        setSupportActionBar(myChildToolbar);
        ActionBar ab = getSupportActionBar();
        assert ab != null;
        ab.setDisplayHomeAsUpEnabled(true);
}
```

I can now get the book object from the intent that instantiated the activity class. Once doing this I will initialise the ui components defined in my xml layout then populate them accordingly. As with many aspects of this activity; I have done this process before so all I have to do is copy the code to set ui text values from viewSearchBook activity.

```
//get book class serialised in the intent
final book meta = (book) getIntent().getSerializableExtra("book");
//initialise ui components
networkFAB requestBook = findViewById(R.id.request);
TextView rentingText = findViewById(R.id.rentingtext);
networkFAB rentBookStop = findViewById(R.id.rent);
TextView rentBookStopText = findViewById(R.id.rentingtext2);
if (meta.getRenting()) {
    requestBook.setVisibility(View.INVISIBLE);
    rentingText.setVisibility(View.INVISIBLE);
    rentBookStop.setVisibility(View.VISIBLE);
```

```
12      rentBookStopText.setVisibility(View.VISIBLE);
13  }
14  //display all data is it exists
15  if (meta.getIsbn() != null) {
16      TextView bookISBN = findViewById(R.id.isbn);
17      bookISBN.setText(meta.getIsbn());
18      bookISBN.setVisibility(View.VISIBLE);
19  }
20  if (meta.getTitle() != null) {
21      TextView bookTitle = findViewById(R.id.bookTitle);
22      bookTitle.setText(meta.getTitle());
23      bookTitle.setVisibility(View.VISIBLE);
24  }
25  if (meta.getAuthors() != null) {
26      TextView bookAuthors = findViewById(R.id.bookAuthors);
27      bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
28      bookAuthors.setVisibility(View.VISIBLE);
29  }
30  if (meta.getGenres() != null) {
31      TextView bookGenres = findViewById(R.id.bookGenres);
32      bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
33      bookGenres.setVisibility(View.VISIBLE);
34  }
35  if (meta.getImage() != null) {
36      ImageView bookImage = findViewById(R.id.bookImage);
37      Picasso.get().load(meta.getImage()).into(bookImage);
38      bookImage.setContentDescription(meta.getImage());
39      bookImage.setVisibility(View.VISIBLE);
40  }
```

This has only displayed the book details, I also want to show owner details. To do this I must use the getOwnerReference method of the book class and use this string to get the user document from Firestore. Once this has completed it should display the data via the activity.

```
1   //initialise firestore connection
2   final FirebaseFirestore db = FirebaseFirestore.getInstance();
3   final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
4   final String userID = user.getUid();
5   //get user who owns the book data
6   db.collection("users").document(meta.getOwnerReference()).get().addOnCompleteListener(new
    ↪  OnCompleteListener<DocumentSnapshot>() {
7       @Override
8       public void onComplete(@NonNull Task<DocumentSnapshot> task) {
9           if (task.isSuccessful()) {
10              DocumentSnapshot document = task.getResult();
11              if (document.exists()) {
12                  //display data if it exists
13                  TextView name = findViewById(R.id.name);
14                  TextView year = findViewById(R.id.year);
15                  TextView form = findViewById(R.id.form);
16                  TextView rating = findViewById(R.id.rating);
17                  name.setText(document.get("name").toString());
18                  year.setText(document.get("yearGroup").toString());
19                  form.setText(document.get("formRoom").toString());
20                  rating.setText(document.get("userRating").toString());
21                  name.setVisibility(View.VISIBLE);
22                  year.setVisibility(View.VISIBLE);
23                  form.setVisibility(View.VISIBLE);
24                  rating.setVisibility(View.VISIBLE);
25              } else {
```

```
                //prompt the user doc does not exist
                Toast.makeText(getApplicationContext(), "error occurred",
                ↪  Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(getApplicationContext(), "error occurred", Toast.LENGTH_SHORT).show();
        }
    }
});
```

I will now add functionality to the stopRent floating action button. This is quite straight forward as it just involves editing Firestore values, something that I have gotten used to now. I use my own networkFAB class here as I want the floating action button to only trigger if there is internet available. I can do the same for the beginEvent floating action button. The code that will execute here is essentially the opposite to that which takes a book off rent. Again this is fairly simple.

```
View.OnClickListener stopRentClick = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final FirebaseFirestore db = FirebaseFirestore.getInstance();
                final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                Map<String, Object> dataBook = new HashMap<>();
                dataBook.put("renting", false);
                dataBook.put("rentingID", "");
                db.collection("books").document(meta.getFirestoreID()).set(dataBook,
                ↪  SetOptions.merge());
                db.collection("users").document(user.getUid()).update("borrowedBooks",
                ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
                db.collection("users").document(meta.getOwnerReference()).update("loanedBooks",
                ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
                Toast.makeText(getApplicationContext(), "returned book", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(viewSearchBookActivity.this, homeActivity.class);
                startActivity(intent);
                finish();
            }
        };
View.OnClickListener rentEvent = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Map<String, Object> dataBook = new HashMap<>();
        dataBook.put("market", false);
        dataBook.put("renting", true);
        dataBk.put("rentingID", userID);
        db.collection("books").document(meta.getFirestoreID()).set(dataBook, SetOptions.merge());
        DocumentReference docRef = db.collection("users").document(userID);
        docRef.update("borrowedBooks", FieldValue.arrayUnion(meta.getFirestoreID()));
        String bookOwnerID = meta.getOwnerReference();
        db.collection("users").document(bookOwnerID).update("loanedBooks",
        ↪  FieldValue.arrayUnion(meta.getFirestoreID()));
        Toast.makeText(getApplicationContext(), "renting book", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(viewSearchBookActivity.this, homeActivity.class);
        startActivity(intent);
        finish();
    }
};
requestBook.setClickEvent(rentEvent);
rentBookStop.setClickEvent(stopRentClick);
```

Note that the above code for managing ui components is all sequentially stored within the onCreate method.

I have added all required methods and attributes to the class, thus I will no begin to test it. As before I searched books with the isbn: 1782944141. At first my application failied to compile, this was due to a syntax error, I had in misspelt a variable name:

```
1 dataBook.put("rentingID", userID);
```

After the above fix, my program successfully compiled. I have provided a link to the recording of the test[2] in the footer. As you can see the test was successful, books were added and removed from both the search and rented book recycle views.

[2] $imp2TEST9 - 10.webm$ and $imp2TEST13 - 14.webm$

4.3.6 Prototype Specification Testing

I will refer to the 14 tests I detailed in the design of the this implementation to test my final implemenation. I will provide/list recordings for them all as media evidence. I will test my app using the black-box testing method. Also known as Behavioral Testing, Black Box testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. In other-words I will be looking for whether the desired outputs are met.

1. Tests 1 - 2
    - $imp2TEST1.png$
    - $imp2TEST2.png$
    - The ouputs were as expected
    - I have provided photos of the firestore console in both of the states. This is because the user's rating is not visible to the user thus I cannot displays the outputs from within the app.
2. Tests 3 - 4
    - $imp2TEST3-4.webm$
    - The ouputs were better than expected
    - As can be seen within the footage, I also tested both when no internet was available, the application followed the correct decisions involving restricting access to the scan activity if no internet is available. This is something that I had not originally planned for or though about, it comes as a pleasant surprise that the abstraction involved in creating this application has spread far through its implementation to create a stable experience across the board.
3. Tests 5 - 8
    - $imp2TEST5-8.webm$
    - The ouputs were as expected
    - Although not shown in the footage, Firestore console also updated accordingly, this can be inferred from the successful results as if this was not the case the activity would not display the expected outputs.
4. Tests 9 - 10
    - $imp2TEST9-10.webm$
    - The ouputs were as expected
    - It should be noted that this is the same activity as the one used to see search results, thus I can confirm that it's dynamic nature performs correctly.
5. Tests 11 - 12
    - $imp2TEST11-12.webm$
    - The ouputs were partially expected
    - Overall these tests were definitely a success but sorting times varied with internet connection, this is because I am accessing the cloud for user data, this requires internet a service that is not always consistent.
6. Tests 13 - 14
    - $imp2TEST13-14.webm$
    - The ouputs were as expected
    - I have also shown that a rented book is added to the rented books list on the books fragment, this confirms that the process is working correctly.

Table Of Success Criteria		
Criteria	Justification	Acheived
Completed Project - fully functioning cloud-based application that allows users to exchange books	The stakeholders must be able to use the app effectively in order to solve the initial problem.	Yes
Book Management - Alter book data	Allows the user to delete books in the case that they may not want that data to be held any longer. Books should be able transition between states in order to signify changes in it's life cycle e.g. scanned, on the market or rented.	Yes
Book Search - Criteria for searching, ranked results	User can find options that are appropriate for what they want or if they are unsure they can find somewhat relevant books which may still be helpful. Ranked results will allow users to find the best quality providers.	Yes
Book Request Management - time place time input, accept decline dialogue, clear deadline details	Provides a clear interface for users to formally request a book based on clear parameter. This data is then relayed to the other user thus making it an instantaneous process. Deadline enforcement will ensure users are aware of when to return the book, reducing needless complications.	Partially - In light of changes in requirements I had to realign my goals for this requirement, on this basis I have completed the updated requirements
User Rating - averaged over time on a 1-5 star basis	Ensure users can rate each other based on their reliability and quality of service. Ultimately forms confidence that books can be requested, exchanged and returned with honesty and without problems. Measure in which appropriate penalties can be applied to poorly behaving users.	Partially - Changes in requirements led me take an automated approach to this feature, this I have successful achieved
Error Free	It would not be acceptable to present a program containing glitches or bugs, especially any logic errors which can mean the application runs but not as expected.	Yes
Graphics Styling	The client should be impressed by the material theme but it should not be a processor intensive task.	Yes
Navigation Layout - bottom navigation, back navigation	Client should be able to feel a flow in the app whereby they can open activities and return to previous ones quickly. Should also prevent users from navigating to expired activities such as to the log in screen once logged in.	Yes
Suitable Complexity	The app must live up to the standards of an A-Level project for .g. database management, file handling, network tasks, multi-threaded operations , searching and sorting algorithms etc.	Yes

4.3.7 Stakeholder Feedback

With the completion of this final prototype, I will present this final solution to my stakeholders. I have prepared a set of questions I will ask my stakeholders, all of which should give me an indication of how my solution solves the problem of sharing books within school. I started by showing my upper school stakehodlers, Jeevon, Lewis and Shashi, the achieved criteria table detailed in the previous section. They agreed with two of the section being partially complete but also understood how this was necessary due to changes in requirements. In order to confirm that the app was error free, I showed them the recordings of various tests and also let them use the app themselves. I then asked them a series of questions to see what they thought of the app. After this I went to my other stakeholders, Joe and Mr Jones, to ask them the same questions. I have summarized their responses below.

Question	Yes/No	Comments
Where all the features we discussed in the beginning implemented?	Yes	All stakeholders agreed that features were implemented as necessary. Jeevon and Mr Jones both said that in hind-sight the new solution is easy to use and now that it is actually implemented feels like better experience than the original requirements may of yielded.
Are all book management features working correctly?	Yes	All agreed that the app worked fluently like it was proposed, it has no issues in the logic of managing books or displaying data.
Can you rent books easily?	Yes	Shashi displayed his concern that you can only search using ISBNs which may cause a problem for minority of users as this process could be tedious. I then suggested that maybe such a searching feature is too general and the original idea of the app was to search for specific books. Other agreed with this statement but also added that such a feature is definitely not necessary but still would be pleasant.
Do you think that the rating system is helpful?	Yes	Mr Jones re-called his first interview with myself and said that the system's automated implementation avoids student bias or rigging of the system: this inline with the principles of the school. He then added that he would certainly endorse the app with his students. Joe and Lewis both said similar things, the rating system is fair, present but not hindering, it is discrete and thus makes the app easier to use.
Is navigation easy/fluid?	Yes	All agreed that the bottom navigation and tool bar makes the experience especially smooth. The floating action buttons custom usage was also said to be useful and consistent in preventing unauthorised actions - by Shashi.
Did you realise books are sorted in terms of rating?	No	All stakeholders said that they only realised until they selected multiple books from a given set of search results I then asked them all if I should change anything to make it more explicit. All said that it was not needed as the ordering feature should be discrete and hidden, no one should want to rent from a low quality user and given people will always select the first result they should always select the best user.
Did the interface look appealing?	Yes	I got similar responses the question about navigation. Everyone said that it was fluid and easy to use.

Chapter 5

Evaluation

5.1 Success Criteria

The first step to evaluating my project is to see how many of the initial requirements were met: As you can see the majority of my criteria was met, apart from those involving lengthy implementation time.

Table Of Success Criteria		
Criteria	Justification	Achieved
Completed Project - fully functioning cloud-based application that allows users to exchange books	The stakeholders must be able to use the app effectively in order to solve the initial problem.	Yes
Account Management - Create an account, log in and out capability on multiple devices, user data storage (date of birth, year group, form room)	The end user should be able to log in from any where and access their data, this is important for identification purposes and for other services such as book exchanges when each user's form room is required. This should also allow for greater dexterity of use for the user.	Yes
Book Input - ISBN scanner, ISBN manual input, meta data source, complete data manual input	This data is required for the basic functioning of the app; users need to be able to see which books they are interacting with. This should also allow the user to easily add data to their account without the need for long input processes, although this is still available, reducing human error.	Yes
Book Management - Alter book data	Allows the user to delete books in the case that they may not want that data to be held any longer. Books should be able transition between states in order to signify changes in it's life cycle e.g. scanned, on the market or rented.	Yes
Book Search - Criteria for searching, ranked results	User can find options that are appropriate for what they want or if they are unsure they can find somewhat relevant books which may still be helpful. Ranked results will allow users to find the best quality providers.	Yes

Book Request Management - time place time input, accept decline dialogue, clear deadline details	Provides a clear interface for users to formally request a book based on clear parameter. This data is then relayed to the other user thus making it an instantaneous process. Deadline enforcement will ensure users are aware of when to return the book, reducing needless complications.	Partially - In light of changes in requirements I had to realign my goals for this requirement, on this basis I have completed the updated requirements
User Rating - averaged over time on a 1-5 star basis	Ensure users can rate each other based on their reliability and quality of service. Ultimately forms confidence that books can be requested, exchanged and returned with honesty and without problems. Measure in which appropriate penalties can be applied to poorly behaving users.	Partially - Changes in requirements led me take an automated approach to this feature, this I have successful achieved
Error Free	It would not be acceptable to present a program containing glitches or bugs, especially any logic errors which can mean the application runs but not as expected.	Yes
Graphics Styling	The client should be impressed by the material theme but it should not be a processor intensive task.	Yes
Navigation Layout - bottom navigation, back navigation	Client should be able to feel a flow in the app whereby they can open activities and return to previous ones quickly. Should also prevent users from navigating to expired activities such as to the log in screen once logged in.	Yes
Suitable Complexity	The app must live up to the standards of an A-Level project for .g. database management, file handling, network tasks, multi-threaded operations , searching and sorting algorithms etc.	Yes

5.2 Failures and Improvements

5.2.1 Failures

- Throughout developmental testing I found issues with my code's logic, runtime errors and in correct usage of androids highly abstracted development eco-system. All of these errors I was able to research and remove. Those that I cannot for example runtime error with Firestore or internet services (which are dependent on external factors) I ensured have been handled with exception statements and error messages.

- The barcode scanning is not as fast as I would like and can often fail if the lighting conditions are less than ideal. I have implemented features such as the flash light toggle to over come this but it does not always work. This is a failure that is beyond my control but I hope that my measures to reduce such occurrences have made it's usage better.

- Changes in requirements due to time-constraints can be seen as a failure. Although my stakeholders agree that the changes are actually better than the originally proposed ones, it still came about due to poor planning and false time-line. I did not expect development for android to be so long.

5.2.2 Developmental Improvements

- The rating system has become much easier to use and was simple to implement. It is hidden from the user and provides advantage without the user even noticing. I believe this is the best sort of improvement to an app as it provides an experience that does not bog the user down or make them concerned with the reliability of the app or even how it works; to the user it simply does what it is meant to.
- The requesting feature has been simplified for the better. It is now simpler and easier to use. Both parties do not have to get that involved, i.e app usage is kept to a minimum. The problem I set out to solve was to find books one can borrow. The requesting feature (and the app as a whole) provides this platform and information and leaves the actual exchange to occur between the people themselves. An app that regulates everything would not be suitable for a school where there are random events that prevent events from occurring, further more this system would be bothersome to user compare to mine as they would have to constantly check the app.
- My apps' capability to handle a lack of internet is universal thus providing a stable experience. This is an improvement as it required me to edit an existing android ui component for my own use. In my opinion my implementation is better than the native android one as it remove the need to needless repetitive code.
- My final improvement comes with optimization using object-oriented-programming. Yes, using java means that I have to use the paradigm, but a developer could still force procedural thinking onto it making it's advantages redundant. I think that I have completely embraced the paradigm making use of interfaces, inheritance, polymorphism(overriding and overloading) and it's abstractive form to create an efficiently coded solution.

5.3 Changes to Requirements and Overall Impact

For the most part, I stuck to the design plan as I had intended to. But when it came to the second prototype which including user rating and book management, realising the size of this task I decided to alter the features. This is because my changes solution was more doable in the timeframe and still provided me with sufficient solutions. It was more beneficial to me to make progress on a good enough feature set than be slow in developing a longer and unnecessary one. At least this way I would know I have an app to present completed to my stakeholders and I could come back for further development given extra time. I think that my revised solution for renting and rating is more than sufficient so not implementing the original criteria is not devastating. I should also recall one of my requirements that measures whether my app is complex enough. I think that although I removed features my application is still very complex, it deals with internet connections(floating action buttons), ui components(bottom navigation), cloud storage(Firestore), hardware usage(camera and flash), abstracted machine learning (barcode scanning), user authentication(Firebase Auth) and much more, all efficiently implemented.

5.4 Limitations and Potential for Further Development

One area that I have expressed concern for is the barcode scanner due to its variable success. Whilst research Firebase Machine Learning I saw a functionality set that allows for barcodes to be scanned and then processed on the cloud via Googles Servers. These servers would obviously be more powerful than a mobile phone thus a more complex neural network could be used on the image which could make the accuracy of it better. Perhaps if I was given more time I would implement this as it would no doubt improve scanning.

My stakeholders previously discussed concerns with the search functionality and it's limitation to an ISBN's. I definitely agreed with this and would of implemented a system where a user can search using an author or a genre. This would not be too hard an would add a great deal to the app.

A final limitation is that the user must be part of the school to use such a system. My app could be used in other schools or communities to share books. I think that a features that could

allow for a unique code to be input that joins a user to a virtual community of books and users could solve this problem. I think this would add greater depth and potential clients to the app.

As I have found with my development, problems will definitely arise with such potential further iterations of the app. The first is time scale; the features discussed could take an extremely long time to implement and research, time that could be spent making the app better as it is rather than adding to it. Also such features may make original ones redundant thus more testing is required on top of that for the new features to ensure that rest of the originally coded app still functions as expected.

5.5 Maintenance

The app as a whole should follow the same logic every time, so not much will change usage to usage. Therefore when it comes to errors in the code, the testing stage showed that all my features run correctly in all circumstances. This means that the maintenance in terms of error should not be needed due to it's robustness.

However, often in computer science, maintenance does not involve correcting errors, but enhancing and tweaking the program to optimise it in some way. This may not involving a vast amount of extra features but instead refining those that already exist. For example I think that I could add more error or success messages so the user knows what is occurring. It is little tweaks like that these that are needed to improve the usability of the app, in terms of the interface, for the user. Hence, the code can be maintained by checking for errors every so often with the client, and improving the interface every so often, as is seen with most social media/branded applications which introduce a whole brand new interface every few months with lots of new features.

5.6 Development Hurdles and Opinions

One of the biggest challenges I faced was to do with the ui. I underestimated the challenge of implementing such components in android. In fact it required more work and research than some of the most complex and 'behind the scenes' features such as barcode recognition. I often had to create multiple classes to set up a component. However this meant a single component could be used in multiple instances. For example, the network floating action button could be used across the app. I also think that xml was difficult to grasp due to the vast array of options, the IDE's layout creater features helped with this. Overall I think that my choices of colors, styling, components and implementation of ui components and activities were good as it provided an abstracted way of implementing them so that I could reuse certain things, sparing my time and effort.

Using Firebase was also a huge challenge as it is like nothing else I have used before. Despite it's advantages, I had to create a numerous amount fail safes due to it's dependence on internet. It's usage was 100% necessary however in order to create an app that can be used across multiple phone and users. It was also helpful for the authentication stages as all I had to do was set up the activity, Firebase handled the rest if the login/sign up. This system is not that complex but is very time consuming to implement thus it saved me a tremendous amount of time.

The final development hurdle was using OOP with Android as at time it seemed easier to use procedural methodologies, such as with displaying data. But I soon realised, that such hardships bought great advantages. Such as with the book-class, how else would I transfer important organised data? In a procedural language I would have to use an array, a structure that simply does not have the dexterity of use like a class. Furthermore, OOP allowed me to use it's features of inheritance and polymorphism, both of which made other areas of development easier.

5.7 Additional Features

I collated a list of features I could implement later on:

- Firebase Cloud Barcode Scanning

- Multi-school app usage
- Multiple Search terms
- Search ordering (most popular books/genre)
- Permanent book renting i.e buying books (e-commerce)
- Teacher Administration features
- User-user rating system
- Book Recommendation System

There are clearly many other features that my app could have had which were not thought of in the design stage, nor were they suggested by the client, but are still very good ideas nonetheless. Perhaps in future development the app could have all of these features.

5.8 Evolving Future Requirements

The program in terms of adapting to changing requirements of stakeholders should bode well in the most part. This is because a lot of the additional features mentioned previously could be added in Java by instantiating already existing classes and adding to the xml. My abstracted code purposefully makes its easy to do this. Therefore, for small changes it can easily adapt to whatever the stakeholders require. The main issue with the programs' adaptability is larger more fundamental changes such as a new scanning system or multi-school system; it may require completely redesigning the app both in Firestore, the UI and the code itself. Hence, the adaptability on a small scale is very good for the program, but for bigger, more ambitious ideas, it would not be easy to carry out.

5.9 Stakeholder Final Feedback

I gave all the new ideas to my main stakeholders, Lewis, Jeevon and Shashi, and asked them to give final feedback on the project from the initial requirements, to development, to now (this was their collective statement):
"Thanks to Palvinder for creating this application. We asked for an app that could allow for students to share books easily and securely. The app provides this core functionality along with stability and extra features that enhance the experience. The overall requirements were clearly met this way, the app is well designed, the ui is polished and fun to use. We had originally expected a rating and requesting system of greater depth, we understood this was not feasible to implement, the new solution is perfectly fine and does all that we wanted. It is disappointing that some features such as searching for titles or authors is not included but this was never part of the original requirements and not that necessary. We think that the final app is great it meets what we had more or less specified. It should be noted that the app works consistently with no bugs, another good job. We really like the color scheme and how it is maintained throughout the app. When it comes to future development, there is a lot to think about, given it can be scaled to a much larger user base. The overcomplicated and overambitious aspects of the game were clearly not doable within the time frame, but in the future, since there is no restrictions we can see a lot of improvements and additional features. Overall, the app works well and consistently, it is a job well done!"

5.10 Developer Feedback

This project was full of moments where it seemed like a dead end and there was no way of coding it. Through perseverance and use of A Level computational thinking I managed to work around it and fulfil as many of Stakeholders' requirements and my success criteria as possible. At first missing certain criteria was disappointing but I simply could not do it so I moved on to something different, something that fortunately turned out to be better. There is a lot more that could be added so there is much space for development. I would like to add that this entire development

process has been a great experience, throughout I have faced problems I have had to overcome and find completely unique solutions to. I have learnt a great deal about cloud storage and android along the way. Overall in terms of development I think the success of my app is a testament to the power of abstraction and how it can break down a large problem into smaller more manageable chunks. I am glad my Stakeholders like the app and although the usability could be improved, the project has been completed.

Chapter 6

Appendices

6.1 *addBookActivity.java*

```
1  package com.bookstore.palvindersingh;
2
3  import android.content.Intent;
4  import android.os.AsyncTask;
5  import android.os.Bundle;
6  import android.support.annotation.NonNull;
7  import android.support.annotation.Nullable;
8  import android.support.design.widget.FloatingActionButton;
9  import android.support.v7.app.ActionBar;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.Toolbar;
12 import android.view.View;
13 import android.widget.ImageView;
14 import android.widget.TextView;
15 import android.widget.Toast;
16
17 import com.google.android.gms.tasks.OnFailureListener;
18 import com.google.android.gms.tasks.OnSuccessListener;
19 import com.google.firebase.auth.FirebaseAuth;
20 import com.google.firebase.auth.FirebaseUser;
21 import com.google.firebase.firestore.DocumentReference;
22 import com.google.firebase.firestore.FieldValue;
23 import com.google.firebase.firestore.FirebaseFirestore;
24 import com.google.firebase.firestore.Transaction;
25 import com.squareup.picasso.Picasso;
26
27 import java.util.ArrayList;
28
29 public class addBookActivity extends AppCompatActivity {
30
31     //book attribute
32     book book;
33
34     //convert array into a string with spaces in between each value
35     private String arrayToDisplayString(ArrayList data) {
36         String string = "";
37         for (int i = 0; i < data.size(); i++) {
38             string = string + data.get(i);
39             if (i != data.size() - 1) {
40                 string = string + " ";
41             }
42         }
```

```
43          return string;
44      }
45
46      @Override
47      protected void onCreate(Bundle savedInstanceState) {
48          super.onCreate(savedInstanceState);
49          setContentView(R.layout.activity_add_book);
50
51          //initialise toolbar
52          Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
53          myChildToolbar.setTitle("submit a book");
54          setSupportActionBar(myChildToolbar);
55          ActionBar ab = getSupportActionBar();
56          assert ab != null;
57          ab.setDisplayHomeAsUpEnabled(true);
58
59          //initialise ui components
60          final TextView isbnText = findViewById(R.id.isbn);
61          final String extraData = getIntent().getStringExtra("isbn");
62          isbnText.setText(extraData);
63
64          //start async task to get metadata
65          new setData().execute(extraData);
66
67          //initialise wrong floating action button
68          FloatingActionButton wrong = findViewById(R.id.wrong);
69          //on click listener
70          wrong.setOnClickListener(new View.OnClickListener() {
71              @Override
72              public void onClick(View v) {
73                  //go to manual input activity
74                  Intent intent = new Intent(addBookActivity.this, addBookManualActivity.class);
75                  startActivity(intent);
76                  finish();
77              }
78          });
79
80          //initialise correct floating action button
81          networkFAB correct = findViewById(R.id.correct);
82          //on click listener
83          View.OnClickListener correctClick = new View.OnClickListener() {
84              @Override
85              public void onClick(View v) {
86                  //initialise ui component
87                  TextView bookTitle = findViewById(R.id.bookTitle);
88                  //check if data has loaded
89                  if (bookTitle.getText().equals("title")) {
90                      Toast.makeText(getApplicationContext(), "wait for data to load",
                        ↪  Toast.LENGTH_SHORT).show();
91                  } else {
92                      //add data to firestore
93                      final FirebaseFirestore db = FirebaseFirestore.getInstance();
94                      final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
95                      book.setOwnerReference(user.getUid());
96                      //transaction processing
97                      db.runTransaction(new Transaction.Function<Void>() {
98                          @Nullable
99                          @Override
100                         public Void apply(@NonNull Transaction transaction) {
101                             //add book to collection
```

```
102                             db.collection("books").add(book).addOnSuccessListener(new
                                ↪  OnSuccessListener<DocumentReference>() {
103                                 @Override
104                                 public void onSuccess(DocumentReference documentReference) {
105                                     //update user data
106                                     DocumentReference docRef =
                                        ↪  db.collection("users").document(user.getUid());
107                                     docRef.update("scannedBooks",
                                        ↪  FieldValue.arrayUnion(documentReference.getId()));
108                                 }
109                             });
110                             return null;
111                         }
112                     }).addOnSuccessListener(new OnSuccessListener<Void>() {
113                         @Override
114                         public void onSuccess(Void aVoid) {
115                             //go to home activity if successful
116                             Toast.makeText(getApplicationContext(), "book added",
                                ↪  Toast.LENGTH_SHORT).show();
117                             Intent intent = new Intent(addBookActivity.this, homeActivity.class);
118                             startActivity(intent);
119                             finish();
120                         }
121                     }).addOnFailureListener(new OnFailureListener() {
122                         @Override
123                         public void onFailure(@NonNull Exception e) {
124                             //go to scan book activity if not successful
125                             Toast.makeText(getApplicationContext(), "something unexpected occurred",
                                ↪  Toast.LENGTH_SHORT).show();
126                             Intent intent = new Intent(addBookActivity.this,
                                ↪  scanBookActivity.class);
127                             startActivity(intent);
128                             finish();
129                         }
130                     });
131                 }
132             }
133         };
134         //pass correctClick to correct class
135         correct.setClickEvent(correctClick);
136     }
137
138     //define setData async task
139     private class setData extends AsyncTask<String, Void, bookMetaData> {
140
141         @Override
142         protected bookMetaData doInBackground(String... extraData) {
143             //run bookMetaData class
144             return new bookMetaData(extraData[0]);
145         }
146
147         @Override
148         protected void onProgressUpdate(Void... values) {
149             super.onProgressUpdate(values);
150         }
151
152         @Override
153         protected void onPostExecute(bookMetaData bookMetaData) {
154             //get book object
155             book meta = bookMetaData.getBook();
156             //set book attribute to meta
```

```
157             book = meta;
158             //check a book object exists
159             if (meta == null) {
160                 //go to home activity
161                 Toast.makeText(getApplicationContext(), "book does not exist",
                    ↪  Toast.LENGTH_SHORT).show();
162                 Intent intent = new Intent(addBookActivity.this, homeActivity.class);
163                 startActivity(intent);
164                 finish();
165                 return;
166             }
167             //set ui components to visible and change according text if it exists
168             TextView bookISBN = findViewById(R.id.isbn);
169             bookISBN.setVisibility(View.VISIBLE);
170             if (meta.getTitle() != null) {
171                 TextView bookTitle = findViewById(R.id.bookTitle);
172                 bookTitle.setText(meta.getTitle());
173                 bookTitle.setVisibility(View.VISIBLE);
174             }
175             if (meta.getAuthors() != null) {
176                 TextView bookAuthors = findViewById(R.id.bookAuthors);
177                 bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
178                 bookAuthors.setVisibility(View.VISIBLE);
179             }
180             if (meta.getGenres() != null) {
181                 TextView bookGenres = findViewById(R.id.bookGenres);
182                 bookGenres.setText(arrayToDisplayString(meta.getGenres()));
183                 bookGenres.setVisibility(View.VISIBLE);
184             }
185             if (meta.getImage() != null) {
186                 ImageView bookImage = findViewById(R.id.bookImage);
187                 Picasso.get().load(meta.getImage()).into(bookImage);
188                 bookImage.setContentDescription(meta.getImage());
189                 bookImage.setVisibility(View.VISIBLE);
190             }
191         }
192     }
193 }
```

6.2 *addBookManualActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Transaction;

import java.util.ArrayList;

public class addBookManualActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_book_manual);


        //initialise toolbar
        Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
        myChildToolbar.setTitle("submit a book");
        setSupportActionBar(myChildToolbar);
        ActionBar ab = getSupportActionBar();
        ab.setDisplayHomeAsUpEnabled(true);

        //initialise correct floating action button
        networkFAB correct = findViewById(R.id.correct);
        //on click listener
        View.OnClickListener correctClick = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //initialise ui components
                EditText isbn = findViewById(R.id.isbn);
                EditText title = findViewById(R.id.title);
                EditText author = findViewById(R.id.author);
                EditText genre = findViewById(R.id.genre);
                //check if input data is valid
                if (validateInputs(isbn.getText().toString(), title.getText().toString(),
                ↪  author.getText().toString(), genre.getText().toString())) {
                    //add data to firestore
                    final FirebaseFirestore db = FirebaseFirestore.getInstance();
                    final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                    ArrayList<String> authorList = new ArrayList<>();
                    authorList.add(author.getText().toString());
```

```
58                      ArrayList<String> genreList = new ArrayList<>();
59                      genreList.add(genre.getText().toString());
60                      //create book class
61                      final book book = new book(authorList, genreList, null,
                        ↪  isbn.getText().toString(), title.getText().toString(), user.getUid(), false,
                        ↪  false, null);
62                      //transaction processing
63                      db.runTransaction(new Transaction.Function<Void>() {
64                          @Nullable
65                          @Override
66                          public Void apply(@NonNull Transaction transaction) {
67                              //add book to collection
68                              db.collection("books").add(book).addOnSuccessListener(new
                                ↪  OnSuccessListener<DocumentReference>() {
69                                  @Override
70                                  public void onSuccess(DocumentReference documentReference) {
71                                      //update user data
72                                      DocumentReference docRef =
                                        ↪  db.collection("users").document(user.getUid());
73                                      docRef.update("scannedBooks",
                                        ↪  FieldValue.arrayUnion(documentReference.getId()));
74                                  }
75                              });
76                              return null;
77                          }
78                      }).addOnSuccessListener(new OnSuccessListener<Void>() {
79                          @Override
80                          public void onSuccess(Void aVoid) {
81                              //go to home activity if successful
82                              Toast.makeText(getApplicationContext(), "book added",
                                ↪  Toast.LENGTH_SHORT).show();
83                              Intent intent = new Intent(addBookManualActivity.this,
                                ↪  homeActivity.class);
84                              startActivity(intent);
85                              finish();
86                          }
87                      }).addOnFailureListener(new OnFailureListener() {
88                          @Override
89                          public void onFailure(@NonNull Exception e) {
90                              //go to scan activity if not successful
91                              Toast.makeText(getApplicationContext(), "something unexpected occurred",
                                ↪  Toast.LENGTH_SHORT).show();
92                              Intent intent = new Intent(addBookManualActivity.this,
                                ↪  scanBookActivity.class);
93                              startActivity(intent);
94                              finish();
95                          }
96                      });
97                  } else {
98                      //prompt user to input valid data
99                      Toast.makeText(getApplicationContext(), "enter valid data",
                        ↪  Toast.LENGTH_SHORT).show();
100                 }
101             }
102         };
103         //pass correctClick into correct class
104         correct.setClickEvent(correctClick);
105     }
106     //validate user inputs
107     private Boolean validateInputs(String isbn, String title, String author, String genre) {
108         //check if isbn is valid
```

```
109         if (!(isbn.length() == 10 || isbn.length() == 13)) {
110             return false;
111         }
112         //check if other data lengths are appropriate
113         return title.length() >= 3 && author.length() >= 3 && genre.length() >= 3;
114     }
115 }
```

6.3 *book.java*

```java
package com.bookstore.palvindersingh;

import java.io.Serializable;
import java.util.ArrayList;

//implement serializable in order to be passed between activities
public class book implements Serializable {

    //initialise attributes
    private ArrayList<String> authors;
    private ArrayList<String> genres;
    private String image;
    private String isbn;
    private String title;
    private String ownerReference;
    private Boolean market;
    private String firestoreID;
    private Boolean renting;
    private String rentingID;


    //constructor
    public book() {
    }

    //constructor
    public book(ArrayList<String> authors, ArrayList<String> genres, String image, String isbn,
    ↪  String title, String ownerReference, Boolean market, Boolean renting, String rentingID) {
        this.authors = authors;
        this.genres = genres;
        this.image = image;
        this.isbn = isbn;
        this.title = title;
        this.ownerReference = ownerReference;
        this.market = market;
        this.renting = renting;
        this.rentingID = rentingID;
    }

    //getter and setters
    public String getFirestoreID() {
        return firestoreID;
    }

    public void setFirestoreID(String firestoreID) {
        this.firestoreID = firestoreID;
    }

    public ArrayList<String> getAuthors() {
        return authors;
    }

    public ArrayList<String> getGenres() {
        return genres;
    }

    public String getImage() {
        return image;
```

```
58      }
59
60      public String getIsbn() {
61          return isbn;
62      }
63
64      public String getTitle() {
65          return title;
66      }
67
68      public String getOwnerReference() {
69          return ownerReference;
70      }
71
72      public void setOwnerReference(String ownerReference) {
73          this.ownerReference = ownerReference;
74      }
75
76      public Boolean getMarket() {
77          return market;
78      }
79
80      public void setMarket(Boolean market) {
81          this.market = market;
82      }
83
84      public Boolean getRenting() {
85          return renting;
86      }
87
88      public void setRenting(Boolean renting) {
89          this.renting = renting;
90      }
91
92      public String getRentingID() {
93          return rentingID;
94      }
95
96      public void setRentingID(String rentingID) {
97          this.rentingID = rentingID;
98      }
99  }
```

6.4 *bookMetaData.java*

```java
package com.bookstore.palvindersingh;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;

class bookMetaData {

    //initialise attributes
    private final String isbn;
    private book book;

    //constructor
    bookMetaData(String isbn) {
        this.isbn = isbn;
        this.book = getData();
    }

    //method to get book meta data
    private book getData() {
        //get data
        JSONObject JSONData = getJSONData();
        JSONObject volumeInfo = getVolumeInfo(JSONData);
        //check if data exists
        if (volumeInfo == null) {
            //return null to indicate a result does not exist
            return null;
        } else {
            //extract meta data
            String title = getBookTitle(volumeInfo);
            String image = getImageURl(volumeInfo);
            ArrayList<String> authors = getAuthor(volumeInfo);
            ArrayList<String> genres = getGenre(volumeInfo);
            //return a book object from data
            book book = new book(authors, genres, image, this.isbn, title, null, false, false,
            ↪  null);
            return book;
        }
    }

    //method to pull main data from google books api
    private JSONObject getJSONData() {
        try {
            //buffer read the url
            URL url = new URL("https://www.googleapis.com/books/v1/volumes?q=" + this.isbn);
            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
            InputStream inputStream = httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
            String data = "";
            String line = "";
```

```
58              while (line != null) {
59                  line = bufferedReader.readLine();
60                  data += line;
61              }
62              return new JSONObject(data);
63          } catch (IOException | JSONException e) {
64              e.printStackTrace();
65              return null;
66          }
67      }
68
69      //method to parse the first layer of the overall data
70      private JSONObject getVolumeInfo(JSONObject JSONData) {
71          try {
72              JSONArray items = (JSONArray) JSONData.get("items");
73              JSONObject firstItem = (JSONObject) items.get(0);
74              return (JSONObject) firstItem.get("volumeInfo");
75          } catch (JSONException e) {
76              e.printStackTrace();
77              return null;
78          }
79      }
80
81      //method to extract book title
82      private String getBookTitle(JSONObject volumeInfo) {
83          try {
84              return (String) volumeInfo.get("title");
85          } catch (JSONException e) {
86              e.printStackTrace();
87              return null;
88          }
89      }
90
91      //method to extract image url
92      private String getImageURl(JSONObject volumeInfo) {
93          try {
94              JSONObject urls = (JSONObject) volumeInfo.get("imageLinks");
95              return (String) urls.get("thumbnail");
96          } catch (JSONException e) {
97              e.printStackTrace();
98              return null;
99          }
100     }
101
102     //method to extract author array
103     private ArrayList<String> getAuthor(JSONObject volumeInfo) {
104         try {
105             JSONArray authors = (JSONArray) volumeInfo.get("authors");
106             return convertToArrayListFromJSON(authors);
107         } catch (JSONException e) {
108             e.printStackTrace();
109             return null;
110         }
111     }
112
113     //method to extract genre array
114     private ArrayList<String> getGenre(JSONObject volumeInfo) {
115         try {
116             JSONArray categories = (JSONArray) volumeInfo.get("categories");
117             return convertToArrayListFromJSON(categories);
118         } catch (JSONException e) {
```

```
119             e.printStackTrace();
120             return null;
121         }
122     }
123
124     //method to convert a json array to array list for generics
125     private ArrayList<String> convertToArrayListFromJSON(JSONArray jsonArray) throws JSONException {
126         ArrayList<String> data = new ArrayList<>();
127         for (int i = 0; i < jsonArray.length(); i++) {
128             String item = jsonArray.get(i).toString();
129             data.add(item);
130         }
131         return data;
132     }
133
134     //getter
135     public book getBook() {
136         return book;
137     }
138 }
```

6.5 *booksFragment.java*

```
1  package com.bookstore.palvindersingh;
2
3  import android.content.Context;
4  import android.content.Intent;
5  import android.net.Uri;
6  import android.os.Bundle;
7  import android.support.annotation.NonNull;
8  import android.support.v4.app.Fragment;
9  import android.support.v7.widget.LinearLayoutManager;
10 import android.support.v7.widget.RecyclerView;
11 import android.util.Log;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15
16 import com.google.android.gms.tasks.OnCompleteListener;
17 import com.google.android.gms.tasks.Task;
18 import com.google.firebase.auth.FirebaseAuth;
19 import com.google.firebase.auth.FirebaseUser;
20 import com.google.firebase.firestore.FirebaseFirestore;
21 import com.google.firebase.firestore.QueryDocumentSnapshot;
22 import com.google.firebase.firestore.QuerySnapshot;
23
24 import java.util.ArrayList;
25
26 public class booksFragment extends Fragment {
27
28     //boilerplate attributes and methods
29     private static final String ARG_PARAM1 = "param1";
30     private static final String ARG_PARAM2 = "param2";
31
32     private OnFragmentInteractionListener mListener;
33
34     //constructor
35     public booksFragment() {
36     }
37
38     public static booksFragment newInstance(String param1, String param2) {
39         booksFragment fragment = new booksFragment();
40         Bundle args = new Bundle();
41         args.putString(ARG_PARAM1, param1);
42         args.putString(ARG_PARAM2, param2);
43         fragment.setArguments(args);
44         Log.e("books", "fragment newInstance");
45         return fragment;
46     }
47
48     @Override
49     public void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         if (getArguments() != null) {
52             String mParam1 = getArguments().getString(ARG_PARAM1);
53             String mParam2 = getArguments().getString(ARG_PARAM2);
54         }
55     }
56
57     @Override
58     public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
```

```
                            Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_books, container, false);
        Log.e("books", "fragment onCreateView");
        //initialise add book floating action button
        networkFAB addBookFAB = view.findViewById(R.id.floating_action_button);
        //on click listener
        View.OnClickListener addClick = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity(), scanBookActivity.class);
                startActivity(intent);
            }
        };
        addBookFAB.setClickEvent(addClick);
        //initialise recycle view
        getScannedBookTitles(view);
        getRentedBookTitles(view);
        return view;
    }

    private void getRentedBookTitles(final View view) {
        final FirebaseFirestore db = FirebaseFirestore.getInstance();
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        final String userID = user.getUid();
        db.collection("books").whereEqualTo("rentingID", userID).get().addOnCompleteListener(new
        ↪  OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    ArrayList<book> books = new ArrayList<>();
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        book book = document.toObject(com.bookstore.palvindersingh.book.class);
                        book.setFirestoreID(document.getId());
                        books.add(book);
                    }
                    if (books.size() == 0) {
                        books.add(new book(null, null, null, null, "rent a book", null, false,
                        ↪  false, null));
                    }
                    initRentingBooksRecylerView(view, books);
                }
            }
        });
    }

    //method to get all books
    private void getScannedBookTitles(final View view) {
        //initialise firestore connection
        final FirebaseFirestore db = FirebaseFirestore.getInstance();
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        final String userID = user.getUid();
        //get all associated books
        db.collection("books").whereEqualTo("ownerReference",
        ↪  userID).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    ArrayList<book> books = new ArrayList<>();
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        book book = document.toObject(book.class);
                        book.setFirestoreID(document.getId());
```

```
117                         books.add(book);
118                     }
119                     //single book class to indicate no books have been added yet
120                     if (books.size() == 0) {
121                         books.add(new book(null, null, null, null, "add a book", null, false, false,
                            ↪  null));
122                     }
123                     //initialise recycle view
124                     initScannedBooksRecyclerView(view, books);
125                 }
126             }
127         });
128     }
129
130     //method to initialise recycle view
131     private void initScannedBooksRecyclerView(View view, ArrayList<book> books) {
132         RecyclerView recyclerView = view.findViewById(R.id.scannedBooksView);
133         scannedBooksRecyclerViewAdapter adapter = new scannedBooksRecyclerViewAdapter(getContext(),
            ↪  books);
134         recyclerView.setAdapter(adapter);
135         recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
136     }
137
138     private void initRentingBooksRecylerView(View view, ArrayList<book> books) {
139         RecyclerView recyclerView = view.findViewById(R.id.rentingBooksView);
140         rentingBooksRecyclerViewAdapter adapter = new rentingBooksRecyclerViewAdapter(getContext(),
            ↪  books);
141         recyclerView.setAdapter(adapter);
142         recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
143     }
144
145     public void onButtonPressed(Uri uri) {
146         if (mListener != null) {
147             mListener.onFragmentInteraction(uri);
148         }
149     }
150
151     @Override
152     public void onAttach(Context context) {
153         super.onAttach(context);
154         if (context instanceof OnFragmentInteractionListener) {
155             mListener = (OnFragmentInteractionListener) context;
156         } else {
157             throw new RuntimeException(context.toString()
158                     + " must implement OnFragmentInteractionListener");
159         }
160     }
161
162     @Override
163     public void onDetach() {
164         super.onDetach();
165         mListener = null;
166     }
167
168     public interface OnFragmentInteractionListener {
169         void onFragmentInteraction(Uri uri);
170     }
171 }
```

6.6 *detailsActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.ArrayList;

public class detailsActivity extends AppCompatActivity {

    //initialise attributes
    private DatePicker datePicker;
    private EditText yearGroup;
    private EditText formRoom;
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_details);

        //initialise firestore connection
        db = FirebaseFirestore.getInstance();

        //initialise ui components
        datePicker = findViewById(R.id.datePicker);
        yearGroup = findViewById(R.id.yearGroup);
        formRoom = findViewById(R.id.formRoom);
    }

    //method to submit input data
    public void submitData(View v) {
        //validate data
        final String date = datePicker.getDayOfMonth() + "/" + datePicker.getMonth() + "/" +
        ↪  datePicker.getYear();
        ArrayList validation = validate();
        if (validation.get(0) == "f") {
            Toast.makeText(getApplicationContext(), validation.get(1).toString(),
            ↪  Toast.LENGTH_SHORT).show();
            return;
        }
        final String yg = yearGroup.getText().toString();
        final String fr = formRoom.getText().toString();
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        assert user != null;
        String userID = user.getUid();
```

```
57          //add data to firestore
58          final DocumentReference userData = db.collection("users").document(userID);
59          userData.update("dateOfBirth", date)
60                  .addOnSuccessListener(new OnSuccessListener<Void>() {
61                      @Override
62                      public void onSuccess(Void aVoid) {
63                          userData.update("yearGroup", yg)
64                                  .addOnSuccessListener(new OnSuccessListener<Void>() {
65                                      @Override
66                                      public void onSuccess(Void aVoid) {
67                                          userData.update("formRoom", fr)
68                                                  .addOnSuccessListener(new OnSuccessListener<Void>()
                                                    ↪ {
69                                                      @Override
70                                                      public void onSuccess(Void aVoid) {
71                                                          Intent intent = new
                                                            ↪  Intent(detailsActivity.this,
                                                            ↪  homeActivity.class);
72                                                          startActivity(intent);
73                                                          finish();
74                                                      }
75                                                  })
76                                                  .addOnFailureListener(new OnFailureListener() {
77                                                      @Override
78                                                      public void onFailure(@NonNull Exception e) {
79                                                      }
80                                                  });
81                                      }
82                                  })
83                                  .addOnFailureListener(new OnFailureListener() {
84                                      @Override
85                                      public void onFailure(@NonNull Exception e) {
86                                      }
87                                  });
88                      }
89                  })
90                  .addOnFailureListener(new OnFailureListener() {
91                      @Override
92                      public void onFailure(@NonNull Exception e) {
93                      }
94                  });
95      }
96
97      //method to validate data
98      private ArrayList validate() {
99          ArrayList result = new ArrayList<>();
100         try {
101             Integer yg = Integer.parseInt(yearGroup.getText().toString());
102             if (yg > 6 && yg < 14) {
103                 String fr = formRoom.getText().toString();
104                 if ((fr.length() >= 2) && (fr.length() < 4)) {
105                     result.add("t");
106                     result.add("valid");
107                     return result;
108                 } else {
109                     result.add("f");
110                     result.add("form room invalid");
111                     return result;
112                 }
113             } else {
114                 result.add("f");
```

```
115                 result.add("year group must be between 7 and 13");
116                 return result;
117             }
118         } catch (Exception e) {
119             result.add("f");
120             result.add("year group must be an int");
121             return result;
122         }
123     }
124 }
```

6.7 *homeActivity.java*

```
1  package com.bookstore.palvindersingh;
2
3  import android.content.Intent;
4  import android.net.Uri;
5  import android.os.Bundle;
6  import android.support.annotation.NonNull;
7  import android.support.design.widget.BottomNavigationView;
8  import android.support.v4.app.Fragment;
9  import android.support.v4.app.FragmentTransaction;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.Toolbar;
12 import android.view.Menu;
13 import android.view.MenuItem;
14 import android.widget.Toast;
15
16 import com.firebase.ui.auth.AuthUI;
17 import com.google.android.gms.tasks.OnCompleteListener;
18 import com.google.android.gms.tasks.Task;
19
20
21 public class homeActivity extends AppCompatActivity implements
   ↪ homeFragment.OnFragmentInteractionListener, booksFragment.OnFragmentInteractionListener,
   ↪ searchFragment.OnFragmentInteractionListener {
22
23     //method to manage fragments for bottom nav
24     private final BottomNavigationView.OnNavigationItemSelectedListener
       ↪ mOnNavigationItemSelectedListener
25             = new BottomNavigationView.OnNavigationItemSelectedListener() {
26         @Override
27         public boolean onNavigationItemSelected(@NonNull MenuItem item) {
28             Fragment selectedFragment;
29             switch (item.getItemId()) {
30                 case R.id.action_home:
31                     FragmentTransaction transaction =
                       ↪ getSupportFragmentManager().beginTransaction();
32                     selectedFragment = homeFragment.newInstance("home", "fragment");
33                     transaction.replace(R.id.content, selectedFragment);
34                     transaction.commit();
35                     return true;
36                 case R.id.action_books:
37                     FragmentTransaction transaction2 =
                       ↪ getSupportFragmentManager().beginTransaction();
38                     selectedFragment = booksFragment.newInstance("books", "fragment");
39                     transaction2.replace(R.id.content, selectedFragment);
40                     transaction2.commit();
41                     return true;
42                 case R.id.action_search:
43                     FragmentTransaction transaction3 =
                       ↪ getSupportFragmentManager().beginTransaction();
44                     selectedFragment = searchFragment.newInstance("search", "fragment");
45                     transaction3.replace(R.id.content, selectedFragment);
46                     transaction3.commit();
47                     return true;
48             }
49             return false;
50         }
51     };
52
```

```
53      @Override
54      protected void onCreate(Bundle savedInstanceState) {
55          super.onCreate(savedInstanceState);
56          setContentView(R.layout.activity_home);
57
58          //initialise toolbar
59          Toolbar myToolbar = findViewById(R.id.my_toolbar);
60          setSupportActionBar(myToolbar);
61
62
63          //set up first fragment
64          FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
65          transaction.replace(R.id.content, homeFragment.newInstance("first", "fragment"));
66          transaction.commit();
67
68          //initialise bottom navigation
69          BottomNavigationView navigation = findViewById(R.id.navigation);
70          navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
71          navigation.setSelectedItemId(R.id.action_books);
72      }
73
74      //initialise toolbar options
75      public boolean onCreateOptionsMenu(Menu menu) {
76          getMenuInflater().inflate(R.menu.action_bar, menu);
77          return true;
78      }
79
80      //initialise toolbar actions
81      @Override
82      public boolean onOptionsItemSelected(MenuItem item) {
83          switch (item.getItemId()) {
84              case R.id.action_logout:
85                  logoutUser();
86                  return true;
87
88              case R.id.action_restart:
89                  Intent intent = new Intent(homeActivity.this, mainActivity.class);
90                  startActivity(intent);
91                  finish();
92
93              default:
94                  return super.onOptionsItemSelected(item);
95
96          }
97      }
98
99      //method to log current user out
100     private void logoutUser() {
101         AuthUI.getInstance()
102                 .signOut(this)
103                 .addOnCompleteListener(new OnCompleteListener<Void>() {
104                     public void onComplete(@NonNull Task<Void> task) {
105                         Toast.makeText(getApplicationContext(), "logged out",
                            ↪  Toast.LENGTH_SHORT).show();
106                         Intent intent = new Intent(homeActivity.this, mainActivity.class);
107                         startActivity(intent);
108                         finish();
109                     }
110                 });
111     }
112
```

```
113     @Override
114     public void onFragmentInteraction(Uri uri) {
115     }
116 }
```

6.8 *homeFragment.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;

public class homeFragment extends Fragment {

    //initialise boilerplate attributes and methods
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    private OnFragmentInteractionListener mListener;

    public homeFragment() {
    }

    public static homeFragment newInstance(String param1, String param2) {
        homeFragment fragment = new homeFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        Log.e("home", "fragment newInstance");
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            String mParam1 = getArguments().getString(ARG_PARAM1);
            String mParam2 = getArguments().getString(ARG_PARAM2);
        }
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Log.e("home", "fragment onCreateView");
        View view = inflater.inflate(R.layout.fragment_home, container, false);
        //initialise ui components
        LinearLayout networkAvailability = view.findViewById(R.id.networkAvailability);
        //check internet availability
        if (checkInternet()) {
            networkAvailability.setVisibility(View.GONE);
        } else {
            networkAvailability.setVisibility(View.VISIBLE);
        }
```

```
59          return view;
60      }
61
62      public void onButtonPressed(Uri uri) {
63          if (mListener != null) {
64              mListener.onFragmentInteraction(uri);
65          }
66      }
67
68      @Override
69      public void onAttach(Context context) {
70          super.onAttach(context);
71          if (context instanceof OnFragmentInteractionListener) {
72              mListener = (OnFragmentInteractionListener) context;
73          } else {
74              throw new RuntimeException(context.toString()
75                      + " must implement OnFragmentInteractionListener");
76          }
77      }
78
79      @Override
80      public void onDetach() {
81          super.onDetach();
82          mListener = null;
83      }
84
85      //method to check for internet connection
86      private Boolean checkInternet() {
87          ConnectivityManager connMgr = (ConnectivityManager)
            ↪  getActivity().getSystemService(Context.CONNECTIVITY_SERVICE);
88          NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
89          return (networkInfo != null && networkInfo.isConnected());
90      }
91
92      public interface OnFragmentInteractionListener {
93          void onFragmentInteraction(Uri uri);
94      }
95  }
```

6.9 *logInActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.auth.IdpResponse;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Objects;

public class logInActivity extends AppCompatActivity {

    //initialise attributes
    private static final int RC_SIGN_IN = 123;
    private final List<AuthUI.IdpConfig> providers = Collections.singletonList(
            new AuthUI.IdpConfig.EmailBuilder().build());
    private FirebaseFirestore db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log_in);

        //initialise firestore connection
        db = FirebaseFirestore.getInstance();

        //initialise start sign in/up activity
        startActivityForResult(AuthUI.getInstance()
                .createSignInIntentBuilder()
                .setAvailableProviders(providers)
                .setLogo(R.mipmap.ic_launcher)
                .build(), RC_SIGN_IN);
    }

    //override method to manage the results of sign up
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == RC_SIGN_IN) {
            IdpResponse response = IdpResponse.fromResultIntent(data);
            if (resultCode == RESULT_OK) {
                //sign in is successful
```

```
59                  FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
60                  Toast.makeText(getApplicationContext(), "sign in successful",
                    ↪  Toast.LENGTH_SHORT).show();
61                  manageDBChecks(user);
62              } else {
63                  //sign in failed
64                  Toast.makeText(getApplicationContext(), "sign in failed",
                    ↪  Toast.LENGTH_SHORT).show();
65                  finish();
66              }
67          }
68      }
69
70      //method to check if the user has input extra data about themselves before
71      private void manageDBChecks(final FirebaseUser user) {
72          final String userID = user.getUid();
73          //get the users document
74          DocumentReference userDocument = db.collection("users").document(userID);
75          userDocument.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
76              @Override
77              public void onComplete(@NonNull Task<DocumentSnapshot> task) {
78                  //check if user exists in firestore
79                  if (!Objects.requireNonNull(task.getResult()).exists()) {
80                      //make the users document
81                      manageDBAddUserDocument(userID, user);
82                      //go to details activity
83                      Intent intent = new Intent(logInActivity.this, detailsActivity.class);
84                      startActivity(intent);
85                      finish();
86                  } else {
87                      ArrayList<String> array = new ArrayList<>();
88                      array.add(task.getResult().get("dateOfBirth").toString());
89                      array.add(task.getResult().get("formRoom").toString());
90                      array.add(task.getResult().get("yearGroup").toString());
91                      if (Objects.equals(array.get(0), "") && array.get(1) == "" && array.get(2) ==
                        ↪  "") {
92                          //go to details activity
93                          Intent intent = new Intent(logInActivity.this, detailsActivity.class);
94                          startActivity(intent);
95                          finish();
96                      }else{
97                          //go to home activity
98                          Intent intent = new Intent(logInActivity.this, homeActivity.class);
99                          startActivity(intent);
100                         finish();
101                     }
102                 }
103             }
104         });
105
106     }
107
108     //method to make a new document for a user
109     private void manageDBAddUserDocument(String userID, FirebaseUser user) {
110         //initialise hashmap
111         Map<String, Object> userData = new HashMap<>();
112         String name = user.getDisplayName();
113         double rating = 3;
114         double bookNumber = 0;
115         userData.put("name", name);
116         userData.put("scannedBooks", Collections.emptyList());
```

```
117         userData.put("borrowedBooks", Collections.emptyList());
118         userData.put("loanedBooks", Collections.emptyList());
119         userData.put("userRating", rating);
120         userData.put("dateOfBirth", "");
121         userData.put("yearGroup", "");
122         userData.put("formRoom", "");
123         //add user data
124         db.collection("users").document(userID)
125                 .set(userData)
126                 .addOnCompleteListener(new OnCompleteListener<Void>() {
127                     @Override
128                     public void onComplete(@NonNull Task<Void> task) {
129                     }
130                 })
131                 .addOnFailureListener(new OnFailureListener() {
132                     @Override
133                     public void onFailure(@NonNull Exception e) {
134                     }
135                 });
136     }
137 }
```

6.10 *mainActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreSettings;
import com.google.firebase.firestore.SetOptions;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

public class mainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //initialise splash screen
        int SPLASH_TIME_OUT = 250;
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                manageAuth();
            }
        }, SPLASH_TIME_OUT);
    }

    //method to check for internet connection
    private Boolean checkInternet() {
        ConnectivityManager connMgr = (ConnectivityManager)
        ↪  getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        return (networkInfo != null && networkInfo.isConnected());
    }

    private double calcRating(double rented, double loaned) {
        try {
            rented = rented + 1;
            loaned = loaned + 1;
            int x = (int) (loaned / rented);
            double rating = Math.round((1 / (1 + Math.exp(-x))) * 5);
            return rating;
```

```
58          } catch (Exception e) {
59              return -1;
60          }
61      }
62
63      //method to manage authentication
64      private void manageAuth() {
65          //initialise firestore connection
66          FirebaseFirestore firestore = FirebaseFirestore.getInstance();
67          FirebaseFirestoreSettings settings = new FirebaseFirestoreSettings.Builder()
68                  .setTimestampsInSnapshotsEnabled(true)
69                  .build();
70          firestore.setFirestoreSettings(settings);
71          //check if there is an internet connection
72          final Boolean internetConnection = checkInternet();
73          FirebaseAuth auth = FirebaseAuth.getInstance();
74          if (!internetConnection) {
75              //prompt the user there is no connection
76              Toast.makeText(getApplicationContext(), "no internet connection",
                ↪  Toast.LENGTH_SHORT).show();
77          }
78          //if the user is not logged in
79          if (auth.getCurrentUser() == null) {
80              //if there is no internet
81              if (!internetConnection) {
82                  //go to no network activity
83                  Intent intent = new Intent(mainActivity.this, noNetworkActivity.class);
84                  startActivity(intent);
85                  finish();
86              } else {
87                  //go to login activity
88                  Intent intent = new Intent(mainActivity.this, logInActivity.class);
89                  startActivity(intent);
90                  finish();
91              }
92          } else {
93              //initialise firestore connection
94              final FirebaseUser user = auth.getCurrentUser();
95              final FirebaseFirestore db = FirebaseFirestore.getInstance();
96              final String userID = user.getUid();
97              DocumentReference userDocument = db.collection("users").document(userID);
98              //get users document
99              userDocument.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
100                 @Override
101                 public void onComplete(@NonNull Task<DocumentSnapshot> task) {
102                     //check if extra details have been input before
103                     ArrayList<String> array = new ArrayList<>();
104                     array.add(task.getResult().get("dateOfBirth").toString());
105                     array.add(task.getResult().get("formRoom").toString());
106                     array.add(task.getResult().get("yearGroup").toString());
107                     //if they have been input
108                     if (!Objects.equals(array.get(0), "") && array.get(1) != "" && array.get(2) !=
                        ↪  "") {
109                         //update rating
110                         ArrayList currentlyRented = (ArrayList)
                            ↪  task.getResult().get("borrowedBooks");
111                         double currentlyRentedSize = currentlyRented.size();
112                         ArrayList loanedBooks = (ArrayList) task.getResult().get("loanedBooks");
113                         double loanedBooksSize = loanedBooks.size();
114                         Integer rating = (int) calcRating(currentlyRentedSize, loanedBooksSize);
```

```
115                         Integer userRating =
                            ↪  Integer.valueOf(task.getResult().get("userRating").toString().charAt(0));
116                         if (rating != userRating && rating != -1) {
117                             Map<String, Object> data = new HashMap<>();
118                             data.put("userRating", rating);
119                             db.collection("users").document(userID).set(data, SetOptions.merge());
120                         }
121
122                         //go to home activity
123                         Intent intent = new Intent(mainActivity.this, homeActivity.class);
124                         startActivity(intent);
125                         finish();
126                     } else {
127                         //if there is no internet
128                         if (!internetConnection) {
129                             //go to no network activity
130                             Intent intent = new Intent(mainActivity.this, noNetworkActivity.class);
131                             startActivity(intent);
132                             finish();
133                         } else {
134                             //go to details activity
135                             Intent intent = new Intent(mainActivity.this, detailsActivity.class);
136                             startActivity(intent);
137                             finish();
138                         }
139                     }
140                 }
141             });
142         }
143     }
144 }
```

6.11 *networkFAB.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.design.widget.FloatingActionButton;
import android.util.AttributeSet;
import android.view.View;
import android.widget.Toast;

//class to inherit form floating action button and implement an on click listener interface in order to automate internet checking
public class networkFAB extends FloatingActionButton implements View.OnClickListener {

    //initialise attributes
    OnClickListener clickEvent;

    //initialise parent constructors
    public networkFAB(Context context) {
        super(context);
        init();
    }

    public networkFAB(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public networkFAB(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    //method to set an on click listener
    private void init() {
        setOnClickListener(this);
    }

    //setter
    public void setClickEvent(OnClickListener l) {
        this.clickEvent = l;
    }

    //method to check for internet
    private Boolean checkInternet(Context c) {
        ConnectivityManager connMgr = (ConnectivityManager) c.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        return (networkInfo != null && networkInfo.isConnected());
    }

    //override method for on click event
    @Override
    public void onClick(View v) {
        Context c = super.getContext();
        //if there is internet
        if (checkInternet(c)) {
            //perform clickEvent instructions
```

```
57              clickEvent.onClick(v);
58          } else {
59              //prompt the user that there is no internet available
60              Toast.makeText(c.getApplicationContext(), "no internet connection",
                ↪  Toast.LENGTH_SHORT).show();
61          }
62      }
63  }
```

6.12 *noNetworkActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.v4.widget.SwipeRefreshLayout;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

public class noNetworkActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_no_network);

        //initialise ui components
        final SwipeRefreshLayout mySwipeRefreshLayout = findViewById(R.id.swipeLayout);
        //on page swipe listener
        mySwipeRefreshLayout.setOnRefreshListener(
                new SwipeRefreshLayout.OnRefreshListener() {
                    @Override
                    public void onRefresh() {
                        //check for internet
                        if (checkInternet()) {
                            //go to main activity
                            Intent intent = new Intent(noNetworkActivity.this, mainActivity.class);
                            startActivity(intent);
                            finish();
                        } else {
                            //prompt the user of no connection
                            Toast.makeText(getApplicationContext(), "no internet connection",
                            ↪  Toast.LENGTH_SHORT).show();
                            mySwipeRefreshLayout.setRefreshing(false);
                        }
                    }
                }
        );
    }

    //method to check for internet
    private Boolean checkInternet() {
        ConnectivityManager connMgr = (ConnectivityManager)
        ↪  getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        return (networkInfo != null && networkInfo.isConnected());
    }
}
```

6.13 *rentingBooksRecyclerViewAdapter.java*

```
package com.bookstore.palvindersingh;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.ArrayList;

public class rentingBooksRecyclerViewAdapter extends
↪  RecyclerView.Adapter<rentingBooksRecyclerViewAdapter.ViewHolder> {

    //initialise attributes
    private final ArrayList<book> books;
    private Context mContext;

    //boilerplate methods
    rentingBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
        this.books = books;
        this.mContext = mContext;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        //inflate layout
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item,
        ↪  parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, final int position) {
        book book = books.get(position);

        holder.bookStatus.setVisibility(View.GONE);
        if (book.getTitle() != "rent a book") {
            holder.bookStatus.setVisibility(View.VISIBLE);
            holder.bookStatus.setText("renting");
        }

        //set title
        holder.bookTitle.setText(books.get(position).getTitle());
        //initialise on click listener
        holder.parentLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                book book = books.get(position);
                if (book.getTitle() == "rent a book") {
                    //if no books added yet perform add book floating action button click
                    BottomNavigationView navigation = ((Activity)
                    ↪  mContext).findViewById(R.id.navigation);
```

```
57                      navigation.setSelectedItemId(R.id.action_search);
58                  } else {
59                      //go to view book class
60                      Intent intent = new Intent(mContext, viewSearchBookActivity.class);
61                      intent.putExtra("book", book);
62                      mContext.startActivity(intent);
63                  }
64              }
65          });
66      }
67
68      //method to return array size
69      @Override
70      public int getItemCount() {
71          return books.size();
72      }
73
74      //boilerplate class to represent layout
75      class ViewHolder extends RecyclerView.ViewHolder {
76          TextView bookTitle;
77          CardView parentLayout;
78          TextView bookStatus;
79
80          ViewHolder(View itemView) {
81              super(itemView);
82              bookTitle = itemView.findViewById(R.id.bookTitle);
83              parentLayout = itemView.findViewById(R.id.parentLayout);
84              bookStatus = itemView.findViewById(R.id.bookStatus);
85          }
86      }
87  }
```

6.14 *scanBookActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.ml.vision.FirebaseVision;
import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcode;
import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcodeDetector;
import com.google.firebase.ml.vision.barcode.FirebaseVisionBarcodeDetectorOptions;
import com.google.firebase.ml.vision.common.FirebaseVisionImage;
import com.otaliastudios.cameraview.CameraListener;
import com.otaliastudios.cameraview.CameraView;
import com.otaliastudios.cameraview.Flash;

import java.io.ByteArrayInputStream;
import java.util.List;


public class scanBookActivity extends AppCompatActivity {

    //initialise attributes
    private int attempts = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scan_book);

        //initialise toolbar
        Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
        myChildToolbar.setTitle("scan a book");
        setSupportActionBar(myChildToolbar);
        ActionBar ab = getSupportActionBar();
        assert ab != null;
        ab.setDisplayHomeAsUpEnabled(true);

        //initialise ui components
        final TextView isbnData = findViewById(R.id.isbn);
        final networkFAB scan = findViewById(R.id.scan);
        FloatingActionButton flash = findViewById(R.id.flash);
        final CameraView camera = findViewById(R.id.camera);
```

```
59          //initialise camera
60          camera.setLifecycleOwner(this);
61          //add camera picture taken listener
62          camera.addCameraListener(new CameraListener() {
63              @Override
64              public void onPictureTaken(byte[] picture) {
65                  //convert image to a bitmap
66                  Bitmap bitmap = convertToBitmap(picture);
67                  //instantiate options class
68                  FirebaseVisionBarcodeDetectorOptions options =
69                          new FirebaseVisionBarcodeDetectorOptions.Builder()
70                                  .setBarcodeFormats(
71                                          FirebaseVisionBarcode.FORMAT_EAN_13,
72                                          FirebaseVisionBarcode.FORMAT_EAN_8)
73                                  .build();
74                  //convert bitmap to firebase image
75                  FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(bitmap);
76                  //instantiate detector class
77                  FirebaseVisionBarcodeDetector detector =
                    ↪  FirebaseVision.getInstance().getVisionBarcodeDetector(options);
78                  //check for barcodes in the image
79                  Task<List<FirebaseVisionBarcode>> result = detector.detectInImage(image)
80                          .addOnSuccessListener(new OnSuccessListener<List<FirebaseVisionBarcode>>() {
81                              @Override
82                              public void onSuccess(List<FirebaseVisionBarcode> barcodes) {
83                                  //check if there is more than one
84                                  if (barcodes.size() > 1) {
85                                      //prompt the user that there are too many in the frame
86                                      Toast.makeText(getApplicationContext(), "too many barcodes",
                                        ↪  Toast.LENGTH_SHORT).show();
87                                  } else {
88                                      //if there is only one barcode
89                                      if (barcodes.size() == 1) {
90                                          //set attempts to zero
91                                          attempts = 0;
92                                          //extract isbn value
93                                          String rawValue =
                                            ↪  String.valueOf(barcodes.get(0).getRawValue());
94                                          isbnData.setText(rawValue);
95                                          //prompt the user a barcode is found
96                                          Toast.makeText(getApplicationContext(), "barcode found",
                                            ↪  Toast.LENGTH_SHORT).show();
97                                          //go to add book activity
98                                          Intent intent = new Intent(scanBookActivity.this,
                                            ↪  addBookActivity.class);
99                                          //pass isbn value into intent
100                                         intent.putExtra("isbn", rawValue);
101                                         startActivity(intent);
102                                     } else {
103                                         //increment attempts
104                                         attempts += 1;
105                                         //prompt the user no barcode is found
106                                         Toast.makeText(getApplicationContext(), "no barcode found",
                                            ↪  Toast.LENGTH_SHORT).show();
107                                         if (attempts > 2) {
108                                             //create alert dialog for manual isbn input
109                                             AlertDialog.Builder builder = new
                                                ↪  AlertDialog.Builder(scanBookActivity.this);
110                                             final EditText input = new
                                                ↪  EditText(scanBookActivity.this);
111                                             input.setHint("enter an isbn");
```

```
                                        input.setInputType(InputType.TYPE_CLASS_NUMBER);
                                        builder.setView(input);
                                        builder.setPositiveButton("OK", new
                                        ↪  DialogInterface.OnClickListener() {
                                            @Override
                                            public void onClick(DialogInterface dialog, int
                                            ↪  which) {
                                                String rawValue = input.getText().toString();
                                                Intent intent = new
                                                ↪  Intent(scanBookActivity.this,
                                                ↪  addBookActivity.class);
                                                intent.putExtra("isbn", rawValue);
                                                startActivity(intent);
                                                finish();
                                            }
                                        });
                                        builder.setNegativeButton("Cancel", new
                                        ↪  DialogInterface.OnClickListener() {
                                            @Override
                                            public void onClick(DialogInterface dialog, int
                                            ↪  which) {
                                                attempts = 1;
                                                dialog.cancel();
                                            }
                                        });
                                        builder.show();
                                    }
                                }
                            }
                        }
                    })
                    .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(getApplicationContext(), "eRrOr OcCuRrEd",
                            ↪  Toast.LENGTH_SHORT).show();
                        }
                    });
        }
    });

    //on click listener to take a picture
    View.OnClickListener clickEvent = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            camera.capturePicture();
        }
    };
    scan.setClickEvent(clickEvent);

    //on click listener to turn flash on and off
    flash.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (camera.getFlash() == Flash.TORCH) {
                camera.setFlash(Flash.OFF);
            } else {
                camera.setFlash(Flash.TORCH);
            }
        }
    });
```

```
166
167         //on click listener
168         FloatingActionButton manual = findViewById(R.id.manual);
169         manual.setOnClickListener(new View.OnClickListener() {
170             @Override
171             public void onClick(View v) {
172                 //go to add book manual activity
173                 Intent intent = new Intent(scanBookActivity.this, addBookManualActivity.class);
174                 startActivity(intent);
175             }
176         });
177
178         FloatingActionButton testButton = findViewById(R.id.testButton);
179         testButton.setOnClickListener(new View.OnClickListener() {
180             @Override
181             public void onClick(View v) {
182                 String rawValue = "1782944141";
183                 Intent intent = new Intent(scanBookActivity.this, addBookActivity.class);
184                 intent.putExtra("isbn", rawValue);
185                 startActivity(intent);
186             }
187         });
188
189     }
190
191     //method to convert byte array into a bitmap
192     private Bitmap convertToBitmap(byte[] picture) {
193         ByteArrayInputStream arrayInputStream = new ByteArrayInputStream(picture);
194         return BitmapFactory.decodeStream(arrayInputStream);
195     }
196 }
```

6.15 *scannedBooksRecyclerViewAdapter.java*

```java
package com.bookstore.palvindersingh;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.ArrayList;

public class scannedBooksRecyclerViewAdapter extends
↪  RecyclerView.Adapter<scannedBooksRecyclerViewAdapter.ViewHolder> {

    //initialise attributes
    private final ArrayList<book> books;
    private Context mContext;

    //boilerplate methods
    scannedBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
        this.books = books;
        this.mContext = mContext;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        //inflate layout
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item,
        ↪  parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, final int position) {
        book book = books.get(position);

        //check and output book state
        if (!book.getMarket()) {
            if (book.getRenting()) {
                holder.bookStatus.setText("on rent");
            } else {
                if (book.getTitle() != "add a book") {
                    holder.bookStatus.setText("off market");
                } else {
                    holder.bookStatus.setVisibility(View.INVISIBLE);
                }
            }
        } else {
            holder.bookStatus.setText("on market");
        }

        //set title
        holder.bookTitle.setText(books.get(position).getTitle());
```

```
57          //initialise on click listener
58          holder.parentLayout.setOnClickListener(new View.OnClickListener() {
59              @Override
60              public void onClick(View v) {
61                  book book = books.get(position);
62                  if (book.getTitle() == "add a book") {
63                      //if no books added yet perform add book floating action button click
64                      networkFAB btn = ((Activity)
                        ↪  mContext).findViewById(R.id.floating_action_button);
65                      btn.performClick();
66                  } else {
67                      //go to view book class
68                      Intent intent = new Intent(mContext, viewBookActivity.class);
69                      intent.putExtra("book", book);
70                      mContext.startActivity(intent);
71                  }
72              }
73          });
74      }
75
76      //method to return array size
77      @Override
78      public int getItemCount() {
79          return books.size();
80      }
81
82      //boilerplate class to represent layout
83      class ViewHolder extends RecyclerView.ViewHolder {
84          TextView bookTitle;
85          CardView parentLayout;
86          TextView bookStatus;
87
88          ViewHolder(View itemView) {
89              super(itemView);
90              bookTitle = itemView.findViewById(R.id.bookTitle);
91              parentLayout = itemView.findViewById(R.id.parentLayout);
92              bookStatus = itemView.findViewById(R.id.bookStatus);
93          }
94      }
95  }
```

6.16 *searchBooksRecyclerViewAdapter.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.widget.CardView;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.ArrayList;

public class searchBooksRecyclerViewAdapter extends RecyclerView.Adapter<searchBooksRecyclerViewAdapter.ViewHolder> {

    //initialise attributes
    private final ArrayList<book> books;
    private Context mContext;

    //boilerplate methods
    searchBooksRecyclerViewAdapter(Context mContext, ArrayList<book> books) {
        this.books = books;
        this.mContext = mContext;
    }

    @NonNull
    @Override
    public searchBooksRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        //inflate layout
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.scanned_book_item, parent, false);
        return new searchBooksRecyclerViewAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull searchBooksRecyclerViewAdapter.ViewHolder holder, final int position) {
        book book = books.get(position);

        //set book data
        holder.bookTitle.setText(book.getTitle());

        holder.bookStatus.setVisibility(View.GONE);

        //initialise on click listener
        holder.parentLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                book book = books.get(position);
                //go to view searched book activity
                Intent intent = new Intent(mContext, viewSearchBookActivity.class);
                intent.putExtra("book", book);
                mContext.startActivity(intent);
            }
        });
```

```
55      }
56
57      //method to return array size
58      @Override
59      public int getItemCount() {
60          return books.size();
61      }
62
63      //boilerplate class to represent layout
64      class ViewHolder extends RecyclerView.ViewHolder {
65          TextView bookTitle;
66          CardView parentLayout;
67          TextView bookStatus;
68
69          ViewHolder(View itemView) {
70              super(itemView);
71              bookTitle = itemView.findViewById(R.id.bookTitle);
72              parentLayout = itemView.findViewById(R.id.parentLayout);
73              bookStatus = itemView.findViewById(R.id.bookStatus);
74          }
75      }
76  }
```

6.17 *searchFragment.java*

```
1  package com.bookstore.palvindersingh;
2
3  import android.annotation.SuppressLint;
4  import android.content.Context;
5  import android.net.Uri;
6  import android.os.Bundle;
7  import android.support.annotation.NonNull;
8  import android.support.v4.app.Fragment;
9  import android.support.v7.widget.LinearLayoutManager;
10 import android.support.v7.widget.RecyclerView;
11 import android.util.Log;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.widget.EditText;
16 import android.widget.TextView;
17
18 import com.google.android.gms.tasks.OnCompleteListener;
19 import com.google.android.gms.tasks.Task;
20 import com.google.firebase.auth.FirebaseAuth;
21 import com.google.firebase.auth.FirebaseUser;
22 import com.google.firebase.firestore.DocumentSnapshot;
23 import com.google.firebase.firestore.FirebaseFirestore;
24 import com.google.firebase.firestore.QueryDocumentSnapshot;
25 import com.google.firebase.firestore.QuerySnapshot;
26
27 import java.util.ArrayList;
28
29
30 public class searchFragment extends Fragment {
31
32     //initialise boilerplate attributes and methods
33     private static final String ARG_PARAM1 = "param1";
34     private static final String ARG_PARAM2 = "param2";
35     private ArrayList<book> bookList = new ArrayList<>();
36     private OnFragmentInteractionListener mListener;
37
38     //constructor
39     public searchFragment() {
40     }
41
42     public static searchFragment newInstance(String param1, String param2) {
43         searchFragment fragment = new searchFragment();
44         Bundle args = new Bundle();
45         args.putString(ARG_PARAM1, param1);
46         args.putString(ARG_PARAM2, param2);
47         fragment.setArguments(args);
48         Log.e("search", "fragment newInstance");
49         return fragment;
50     }
51
52     @Override
53     public void onCreate(Bundle savedInstanceState) {
54         super.onCreate(savedInstanceState);
55         if (getArguments() != null) {
56             String mParam1 = getArguments().getString(ARG_PARAM1);
57             String mParam2 = getArguments().getString(ARG_PARAM2);
58         }
```

```
59      }
60
61      @Override
62      public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
63                               Bundle savedInstanceState) {
64          Log.e("books", "fragment newInstance");
65          final View view = inflater.inflate(R.layout.fragment_search, container, false);
66          //initialise ui components
67          networkFAB searchFAB = view.findViewById(R.id.floating_action_button);
68          final EditText searchInput = view.findViewById(R.id.search);
69
70          //initialise on click event
71          View.OnClickListener clickEvent = new View.OnClickListener() {
72              @Override
73              public void onClick(View v) {
74                  //call get search results method
75                  getSearchResults(searchInput.getText().toString(), view);
76              }
77          };
78          searchFAB.setClickEvent(clickEvent);
79
80          return view;
81      }
82
83      //method to get and display search results
84      private void getSearchResults(String input, final View view) {
85          //initialise firestore connection
86          final FirebaseFirestore db = FirebaseFirestore.getInstance();
87          final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
88          //query all documents on the market with the input isbn
89          db.collection("books").whereEqualTo("isbn", input).whereEqualTo("market",
            ↪  true).get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
90              @SuppressLint("SetTextI18n")
91              @Override
92              public void onComplete(@NonNull Task<QuerySnapshot> task) {
93                  if (task.isSuccessful()) {
94                      ArrayList<book> books = new ArrayList<>();
95                      String userid = user.getUid();
96                      for (QueryDocumentSnapshot document : task.getResult()) {
97                          book book = document.toObject(book.class);
98                          book.setFirestoreID(document.getId());
99                          if (!book.getOwnerReference().equals(userid)) {
100                             //if the user does not own the book add it to an array
101                             books.add(book);
102                         }
103                     }
104                     //if no books are found
105                     if (books.size() == 0) {
106                         //add a boilerplate book class to indicate no books found
107                         books.add(new book(null, null, null, null, "no books found", null, false,
                            ↪  false, null));
108                     }
109                     if (books.get(0).getTitle() != "no books found") {
110                         sortBooks(books, view);
111                     }
112                     TextView results = view.findViewById(R.id.results);
113                     results.setText("RESULTS: " + books.size());
114                 }
115             }
116         });
117     }
```

```
118
119     private void sortBooks(final ArrayList<book> books, final View view) {
120         final ArrayList ratings = new ArrayList();
121         final FirebaseFirestore db = FirebaseFirestore.getInstance();
122         for (book book : books) {
123             String owner = book.getOwnerReference();
124             final ArrayList data = new ArrayList();
125             data.add(book);
126             db.collection("users").document(owner).get().addOnCompleteListener(new
                ↪  OnCompleteListener<DocumentSnapshot>() {
127                 @Override
128                 public void onComplete(@NonNull Task<DocumentSnapshot> task) {
129                     if (task.isSuccessful()) {
130                         DocumentSnapshot document = task.getResult();
131                         if (document.exists()) {
132                             data.add(document.get("userRating"));
133                             ratings.add(data);
134                         }
135                     } else {
136                         data.add(0);
137                         ratings.add(data);
138                     }
139                     if (ratings.size() == books.size()) {
140                         int n = ratings.size();
141                         for (int i = 1; i < n; i++) {
142                             ArrayList temp = (ArrayList) ratings.get(i);
143                             int j = i - 1;
144                             ArrayList nextArray = (ArrayList) ratings.get(j);
145                             Double tempRating = Double.valueOf(temp.get(1).toString());
146                             Double nextRating = Double.valueOf(nextArray.get(1).toString());
147                             while (j >= 0 && nextRating > tempRating) {
148                                 ratings.set(j + 1, ratings.get(j));
149                                 j = j - 1;
150                             }
151                             ratings.set(j + 1, temp);
152                         }
153                         ArrayList<book> sorted = new ArrayList<>();
154                         for (int i = ratings.size() - 1; i > -1; i--) {
155                             sorted.add((book) ((ArrayList) ratings.get(i)).get(0));
156                         }
157                         bookList = sorted;
158                         initSearchBooksRecyclerView(view, bookList);
159                     }
160                 }
161             });
162         }
163     }
164
165     //method to initialise recycle view
166     private void initSearchBooksRecyclerView(View view, ArrayList<book> books) {
167         RecyclerView recyclerView = view.findViewById(R.id.searchBooksView);
168         searchBooksRecyclerViewAdapter adapter = new searchBooksRecyclerViewAdapter(getContext(),
            ↪  books);
169         recyclerView.setAdapter(adapter);
170         recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
171     }
172
173     public void onButtonPressed(Uri uri) {
174         if (mListener != null) {
175             mListener.onFragmentInteraction(uri);
176         }
```

```
177     }
178
179     @Override
180     public void onAttach(Context context) {
181         super.onAttach(context);
182         if (context instanceof OnFragmentInteractionListener) {
183             mListener = (OnFragmentInteractionListener) context;
184         } else {
185             throw new RuntimeException(context.toString()
186                     + " must implement OnFragmentInteractionListener");
187         }
188     }
189
190     @Override
191     public void onDetach() {
192         super.onDetach();
193         mListener = null;
194     }
195
196     public interface OnFragmentInteractionListener {
197         void onFragmentInteraction(Uri uri);
198     }
199 }
```

6.18 *viewBookActivity.java*

```
1  package com.bookstore.palvindersingh;
2
3  import android.annotation.SuppressLint;
4  import android.content.Intent;
5  import android.os.Bundle;
6  import android.support.annotation.NonNull;
7  import android.support.v7.app.ActionBar;
8  import android.support.v7.app.AppCompatActivity;
9  import android.support.v7.widget.Toolbar;
10 import android.view.View;
11 import android.widget.ImageView;
12 import android.widget.TextView;
13 import android.widget.Toast;
14
15 import com.google.android.gms.tasks.OnFailureListener;
16 import com.google.android.gms.tasks.OnSuccessListener;
17 import com.google.firebase.auth.FirebaseAuth;
18 import com.google.firebase.auth.FirebaseUser;
19 import com.google.firebase.firestore.DocumentReference;
20 import com.google.firebase.firestore.FieldValue;
21 import com.google.firebase.firestore.FirebaseFirestore;
22 import com.squareup.picasso.Picasso;
23
24 import java.util.ArrayList;
25
26 public class viewBookActivity extends AppCompatActivity {
27
28     @SuppressLint("SetTextI18n")
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_view_book);
33
34         //initialise toolbar
35         Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
36         myChildToolbar.setTitle("book details");
37         setSupportActionBar(myChildToolbar);
38         ActionBar ab = getSupportActionBar();
39         assert ab != null;
40         ab.setDisplayHomeAsUpEnabled(true);
41
42         //initialise book object serialised within the intent
43         final book meta = (book) getIntent().getSerializableExtra("book");
44         networkFAB deleteBook = findViewById(R.id.delete);
45
46         //display all data if it exists
47         if (meta.getIsbn() != null) {
48             TextView bookISBN = findViewById(R.id.isbn);
49             bookISBN.setText(meta.getIsbn());
50             bookISBN.setVisibility(View.VISIBLE);
51         }
52         if (meta.getTitle() != null) {
53             TextView bookTitle = findViewById(R.id.bookTitle);
54             bookTitle.setText(meta.getTitle());
55             bookTitle.setVisibility(View.VISIBLE);
56         }
57         if (meta.getAuthors() != null) {
58             TextView bookAuthors = findViewById(R.id.bookAuthors);
```

```
59              bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
60              bookAuthors.setVisibility(View.VISIBLE);
61          }
62          if (meta.getGenres() != null) {
63              TextView bookGenres = findViewById(R.id.bookGenres);
64              bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
65              bookGenres.setVisibility(View.VISIBLE);
66          }
67          if (meta.getImage() != null) {
68              ImageView bookImage = findViewById(R.id.bookImage);
69              Picasso.get().load(meta.getImage()).into(bookImage);
70              bookImage.setContentDescription(meta.getImage());
71              bookImage.setVisibility(View.VISIBLE);
72          }
73
74          networkFAB market = findViewById(R.id.market);
75
76          //display book status
77          TextView bookMarket = findViewById(R.id.bookMarket);
78          TextView marketname = findViewById(R.id.marketname);
79          TextView deletename = findViewById(R.id.deletename);
80          bookMarket.setVisibility(View.VISIBLE);
81          if (!meta.getMarket()) {
82              if (meta.getRenting()) {
83                  bookMarket.setText("on rent");
84                  deleteBook.setVisibility(View.INVISIBLE);
85                  market.setVisibility(View.INVISIBLE);
86                  marketname.setVisibility(View.INVISIBLE);
87                  deletename.setVisibility(View.INVISIBLE);
88              } else {
89                  bookMarket.setText("not on the market");
90                  deleteBook.setVisibility(View.VISIBLE);
91                  market.setVisibility(View.VISIBLE);
92              }
93          } else {
94              bookMarket.setText("on the market");
95              market.setVisibility(View.VISIBLE);
96          }
97
98          //display delete button if it is not on rent
99          if (!meta.getRenting()) {
100             deleteBook.setVisibility(View.VISIBLE);
101         }
102
103         //initialise on click listener
104         View.OnClickListener deleteEvent = new View.OnClickListener() {
105             @Override
106             public void onClick(View v) {
107                 //initialise firestore connection
108                 final FirebaseFirestore db = FirebaseFirestore.getInstance();
109                 final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
110                 final String userID = user.getUid();
111                 //delete document
112                 db.collection("books").document(meta.getFirestoreID())
113                         .delete()
114                         .addOnSuccessListener(new OnSuccessListener<Void>() {
115                             @Override
116                             public void onSuccess(Void aVoid) {
117                                 //delete data from user document
118                                 DocumentReference docRef =
                                    ↪  db.collection("users").document(user.getUid());
```

```
                                docRef.update("scannedBooks",
                                ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
                                //go to home activity
                                Toast.makeText(getApplicationContext(), "deleted " +
                                ↪  meta.getTitle(), Toast.LENGTH_SHORT).show();
                                Intent intent = new Intent(viewBookActivity.this,
                                ↪  homeActivity.class);
                                startActivity(intent);
                                finish();
                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText(getApplicationContext(), "failed to delete " +
                                ↪  meta.getTitle(), Toast.LENGTH_SHORT).show();
                            }
                        });
            }
        };
        deleteBook.setClickEvent(deleteEvent);

        //initialise on click listener
        View.OnClickListener clickEventMarket = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //identify book status
                if (!meta.getMarket()) {
                    if (!meta.getRenting()) {
                        //put on market
                        final FirebaseFirestore db = FirebaseFirestore.getInstance();
                        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                        final String userID = user.getUid();
                        db.collection("books").document(meta.getFirestoreID())
                                .update("market", true)
                                .addOnSuccessListener(new OnSuccessListener<Void>() {
                                    @Override
                                    public void onSuccess(Void aVoid) {
                                        Toast.makeText(getApplicationContext(), "on the market",
                                        ↪  Toast.LENGTH_SHORT).show();
                                        Intent intent = new Intent(viewBookActivity.this,
                                        ↪  homeActivity.class);
                                        startActivity(intent);
                                        finish();
                                    }
                                })
                                .addOnFailureListener(new OnFailureListener() {
                                    @Override
                                    public void onFailure(@NonNull Exception e) {
                                        //failed
                                        Toast.makeText(getApplicationContext(),
                                        ↪  "something went wrong", Toast.LENGTH_SHORT).show();
                                    }
                                });
                    }
                } else {
                    //take off market
                    final FirebaseFirestore db = FirebaseFirestore.getInstance();
                    final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                    final String userID = user.getUid();
                    db.collection("books").document(meta.getFirestoreID())
```

```
173                             .update("market", false)
174                             .addOnSuccessListener(new OnSuccessListener<Void>() {
175                                 @Override
176                                 public void onSuccess(Void aVoid) {
177                                     Toast.makeText(getApplicationContext(), "off the market",
                                        ↪  Toast.LENGTH_SHORT).show();
178                                     Intent intent = new Intent(viewBookActivity.this,
                                        ↪  homeActivity.class);
179                                     startActivity(intent);
180                                     finish();
181                                 }
182                             })
183                             .addOnFailureListener(new OnFailureListener() {
184                                 @Override
185                                 public void onFailure(@NonNull Exception e) {
186                                     //failed
187                                     Toast.makeText(getApplicationContext(), "something went wrong",
                                        ↪  Toast.LENGTH_SHORT).show();
188                                 }
189                             });
190                 }
191             }
192         };
193         market.setClickEvent(clickEventMarket);
194     }
195
196     //convert array to readable string
197     private String arrayToDisplayString(ArrayList data) {
198         String string = "";
199         for (int i = 0; i < data.size(); i++) {
200             string = string + data.get(i);
201             if (i != data.size() - 1) {
202                 string = string + " ";
203             }
204         }
205         return string;
206     }
207 }
```

6.19 *viewSearchBookActivity.java*

```java
package com.bookstore.palvindersingh;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.SetOptions;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class viewSearchBookActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_search_book);

        //initialise toolbar
        Toolbar myChildToolbar = findViewById(R.id.my_child_toolbar);
        myChildToolbar.setTitle("book details");
        setSupportActionBar(myChildToolbar);
        ActionBar ab = getSupportActionBar();
        assert ab != null;
        ab.setDisplayHomeAsUpEnabled(true);

        //get book class serialised in the intent
        final book meta = (book) getIntent().getSerializableExtra("book");

        //initialise ui components
        networkFAB requestBook = findViewById(R.id.request);
        TextView rentingText = findViewById(R.id.rentingtext);
        networkFAB rentBookStop = findViewById(R.id.rent);
        TextView rentBookStopText = findViewById(R.id.rentingtext2);

        if (meta.getRenting()) {
            requestBook.setVisibility(View.INVISIBLE);
            rentingText.setVisibility(View.INVISIBLE);
```

```
59              rentBookStop.setVisibility(View.VISIBLE);
60              rentBookStopText.setVisibility(View.VISIBLE);
61          }
62
63          View.OnClickListener stopRentClick = new View.OnClickListener() {
64              @Override
65              public void onClick(View v) {
66                  final FirebaseFirestore db = FirebaseFirestore.getInstance();
67                  final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
68                  Map<String, Object> dataBook = new HashMap<>();
69                  dataBook.put("renting", false);
70                  dataBook.put("rentingID", "");
71                  db.collection("books").document(meta.getFirestoreID()).set(dataBook,
                    ↪  SetOptions.merge());
72                  db.collection("users").document(user.getUid()).update("borrowedBooks",
                    ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
73                  db.collection("users").document(meta.getOwnerReference()).update("loanedBooks",
                    ↪  FieldValue.arrayRemove(meta.getFirestoreID()));
74                  Toast.makeText(getApplicationContext(), "returned book", Toast.LENGTH_SHORT).show();
75                  Intent intent = new Intent(viewSearchBookActivity.this, homeActivity.class);
76                  startActivity(intent);
77                  finish();
78              }
79          };
80
81          rentBookStop.setClickEvent(stopRentClick);
82
83          //display all data is it exists
84          if (meta.getIsbn() != null) {
85              TextView bookISBN = findViewById(R.id.isbn);
86              bookISBN.setText(meta.getIsbn());
87              bookISBN.setVisibility(View.VISIBLE);
88          }
89          if (meta.getTitle() != null) {
90              TextView bookTitle = findViewById(R.id.bookTitle);
91              bookTitle.setText(meta.getTitle());
92              bookTitle.setVisibility(View.VISIBLE);
93          }
94          if (meta.getAuthors() != null) {
95              TextView bookAuthors = findViewById(R.id.bookAuthors);
96              bookAuthors.setText(arrayToDisplayString(meta.getAuthors()));
97              bookAuthors.setVisibility(View.VISIBLE);
98          }
99          if (meta.getGenres() != null) {
100             TextView bookGenres = findViewById(R.id.bookGenres);
101             bookGenres.setText(arrayToDisplayString(meta.getAuthors()));
102             bookGenres.setVisibility(View.VISIBLE);
103         }
104         if (meta.getImage() != null) {
105             ImageView bookImage = findViewById(R.id.bookImage);
106             Picasso.get().load(meta.getImage()).into(bookImage);
107             bookImage.setContentDescription(meta.getImage());
108             bookImage.setVisibility(View.VISIBLE);
109         }
110
111         //initialise firestore connection
112         final FirebaseFirestore db = FirebaseFirestore.getInstance();
113         final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
114         final String userID = user.getUid();
115         //get user who owns the book data
```

```
116         db.collection("users").document(meta.getOwnerReference()).get().addOnCompleteListener(new
            ↪  OnCompleteListener<DocumentSnapshot>() {
117             @Override
118             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
119                 if (task.isSuccessful()) {
120                     DocumentSnapshot document = task.getResult();
121                     if (document.exists()) {
122                         //display data if it exists
123                         TextView name = findViewById(R.id.name);
124                         TextView year = findViewById(R.id.year);
125                         TextView form = findViewById(R.id.form);
126                         TextView rating = findViewById(R.id.rating);
127                         name.setText(document.get("name").toString());
128                         year.setText(document.get("yearGroup").toString());
129                         form.setText(document.get("formRoom").toString());
130                         rating.setText(document.get("userRating").toString());
131                         name.setVisibility(View.VISIBLE);
132                         year.setVisibility(View.VISIBLE);
133                         form.setVisibility(View.VISIBLE);
134                         rating.setVisibility(View.VISIBLE);
135                     } else {
136                         //prompt the user doc does not exist
137                         Toast.makeText(getApplicationContext(), "error occurred",
                            ↪  Toast.LENGTH_SHORT).show();
138                     }
139                 } else {
140                     Toast.makeText(getApplicationContext(), "error occurred",
                        ↪  Toast.LENGTH_SHORT).show();
141                 }
142             }
143         });
144
145         View.OnClickListener rentEvent = new View.OnClickListener() {
146             @Override
147             public void onClick(View v) {
148                 Map<String, Object> dataBook = new HashMap<>();
149                 dataBook.put("market", false);
150                 dataBook.put("renting", true);
151                 dataBook.put("rentingID", userID);
152                 db.collection("books").document(meta.getFirestoreID()).set(dataBook,
                    ↪  SetOptions.merge());
153                 DocumentReference docRef = db.collection("users").document(userID);
154                 docRef.update("borrowedBooks", FieldValue.arrayUnion(meta.getFirestoreID()));
155                 String bookOwnerID = meta.getOwnerReference();
156                 db.collection("users").document(bookOwnerID).update("loanedBooks",
                    ↪  FieldValue.arrayUnion(meta.getFirestoreID()));
157                 Toast.makeText(getApplicationContext(), "renting book", Toast.LENGTH_SHORT).show();
158                 Intent intent = new Intent(viewSearchBookActivity.this, homeActivity.class);
159                 startActivity(intent);
160                 finish();
161             }
162         };
163         requestBook.setClickEvent(rentEvent);
164     }
165
166     //method to convert array into a readable string
167     private String arrayToDisplayString(ArrayList data) {
168         String string = "";
169         for (int i = 0; i < data.size(); i++) {
170             string = string + data.get(i);
171             if (i != data.size() - 1) {
```

```
172                 string = string + " ";
173             }
174         }
175         return string;
176     }
177 }
```

6.20 *activityaddbook.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".addBookActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingLeft="50dp"
            android:paddingRight="50dp">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:fontFamily="@font/arimo"
                android:padding="20dp"
                android:text="@string/is_this_your_book"
                android:textAllCaps="true"
                android:textColor="@color/colorPrimary"
                android:textSize="15sp"
                android:textStyle="bold" />

            <ImageView
                android:id="@+id/bookImage"
                android:layout_width="125dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:adjustViewBounds="true"
                android:contentDescription="@string/books"
                android:scaleType="fitXY"
                android:visibility="gone" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:background="@color/browser_actions_bg_grey"
                android:divider="?android:listDivider"
                android:orientation="vertical"
```

```
58                  android:showDividers="middle">
59
60                  <TextView
61                      android:layout_width="match_parent"
62                      android:layout_height="wrap_content"
63                      android:elevation="10dp"
64                      android:fontFamily="@font/arimo"
65                      android:padding="10dp"
66                      android:text="About"
67                      android:textColor="#000"
68                      android:textSize="15sp" />
69
70                  <TextView
71                      android:id="@+id/isbn"
72                      android:layout_width="match_parent"
73                      android:layout_height="wrap_content"
74                      android:elevation="10dp"
75                      android:fontFamily="@font/arimo"
76                      android:padding="10dp"
77                      android:text="@string/isbn"
78                      android:textColor="@color/colorPrimaryDark"
79                      android:textSize="15sp"
80                      android:visibility="gone" />
81
82                  <TextView
83                      android:id="@+id/bookTitle"
84                      android:layout_width="match_parent"
85                      android:layout_height="wrap_content"
86                      android:fontFamily="@font/arimo"
87                      android:padding="10dp"
88                      android:text="@string/title"
89                      android:textAllCaps="true"
90                      android:textColor="@color/colorPrimaryDark"
91                      android:textSize="15sp"
92                      android:visibility="gone" />
93
94                  <TextView
95                      android:id="@+id/bookAuthors"
96                      android:layout_width="match_parent"
97                      android:layout_height="wrap_content"
98                      android:fontFamily="@font/arimo"
99                      android:padding="10dp"
100                     android:text="@string/authors"
101                     android:textAllCaps="true"
102                     android:textColor="@color/colorPrimaryDark"
103                     android:textSize="15sp"
104                     android:visibility="gone" />
105
106                 <TextView
107                     android:id="@+id/bookGenres"
108                     android:layout_width="match_parent"
109                     android:layout_height="wrap_content"
110                     android:fontFamily="@font/arimo"
111                     android:padding="10dp"
112                     android:text="@string/genres"
113                     android:textAllCaps="true"
114                     android:textColor="@color/colorPrimaryDark"
115                     android:textSize="15sp"
116                     android:visibility="gone" />
117
118             </LinearLayout>
```

```
119
120             </LinearLayout>
121
122         </ScrollView>
123
124         <com.bookstore.palvindersingh.networkFAB
125             android:id="@+id/correct"
126             android:layout_width="wrap_content"
127             android:layout_height="wrap_content"
128             android:layout_margin="16dp"
129             android:layout_marginEnd="8dp"
130             android:layout_marginBottom="8dp"
131             android:clickable="true"
132             android:focusable="true"
133             android:src="@drawable/ic_done_24dp"
134             app:backgroundTint="@color/fui_bgPhone"
135             app:fabSize="normal"
136             app:layout_constraintBottom_toBottomOf="parent"
137             app:layout_constraintEnd_toEndOf="parent" />
138
139         <android.support.design.widget.FloatingActionButton
140             android:id="@+id/wrong"
141             android:layout_width="wrap_content"
142             android:layout_height="wrap_content"
143             android:layout_margin="16dp"
144             android:layout_marginEnd="8dp"
145             android:layout_marginBottom="8dp"
146             android:clickable="true"
147             android:focusable="true"
148             android:src="@drawable/ic_close_24dp"
149             app:backgroundTint="@color/fui_bgEmail"
150             app:fabSize="normal"
151             app:layout_constraintBottom_toTopOf="@+id/correct"
152             app:layout_constraintEnd_toEndOf="parent" />
153
154         <TextView
155             android:layout_width="wrap_content"
156             android:layout_height="wrap_content"
157             android:layout_marginEnd="8dp"
158             android:fontFamily="@font/arimo"
159             android:padding="10dp"
160             android:text="No"
161             android:textColor="#000"
162             android:textSize="15sp"
163             app:layout_constraintBottom_toBottomOf="@+id/wrong"
164             app:layout_constraintEnd_toStartOf="@+id/wrong"
165             app:layout_constraintTop_toTopOf="@+id/wrong" />
166
167         <TextView
168             android:layout_width="wrap_content"
169             android:layout_height="wrap_content"
170             android:layout_marginEnd="8dp"
171             android:layout_marginBottom="8dp"
172             android:fontFamily="@font/arimo"
173             android:padding="10dp"
174             android:text="Yes"
175             android:textColor="#000"
176             android:textSize="15sp"
177             app:layout_constraintBottom_toBottomOf="@+id/correct"
178             app:layout_constraintEnd_toStartOf="@+id/correct" />
179
```

```
180 </android.support.constraint.ConstraintLayout>
```

6.21 *activityaddbookmanual.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".addBookManualActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_margin="25dp"
            android:background="@color/browser_actions_bg_grey"
            android:orientation="vertical"
            android:padding="10dp">

            <ImageView
                android:id="@+id/imageView3"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:contentDescription="@string/app_name"
                app:srcCompat="@mipmap/ic_launcher_foreground" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/isbn"
                android:textColor="@color/colorPrimary"
                android:textSize="15sp" />

            <EditText
                android:id="@+id/isbn"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginBottom="10dp"
                android:autofillHints="@string/isbn"
                android:ems="10"
                android:inputType="number"
                android:textColor="@color/colorAccent" />

            <TextView
                android:layout_width="match_parent"
```

```
58                  android:layout_height="wrap_content"
59                  android:text="@string/title_caps"
60                  android:textColor="@color/colorPrimary"
61                  android:textSize="15sp" />
62
63              <EditText
64                  android:id="@+id/title"
65                  android:layout_width="match_parent"
66                  android:layout_height="wrap_content"
67                  android:layout_gravity="center"
68                  android:layout_marginBottom="10dp"
69                  android:autofillHints="@string/title_caps"
70                  android:ems="10"
71                  android:inputType="text"
72                  android:maxLength="25"
73                  android:textAllCaps="true"
74                  android:textColor="@color/colorAccent" />
75
76              <TextView
77                  android:layout_width="match_parent"
78                  android:layout_height="wrap_content"
79                  android:text="@string/author_caps"
80                  android:textColor="@color/colorPrimary"
81                  android:textSize="15sp" />
82
83              <EditText
84                  android:id="@+id/author"
85                  android:layout_width="match_parent"
86                  android:layout_height="wrap_content"
87                  android:layout_gravity="center"
88                  android:layout_marginBottom="10dp"
89                  android:autofillHints="@string/author_caps"
90                  android:ems="10"
91                  android:inputType="text"
92                  android:maxLength="35"
93                  android:textAllCaps="true"
94                  android:textColor="@color/colorAccent" />
95
96              <TextView
97                  android:layout_width="match_parent"
98                  android:layout_height="wrap_content"
99                  android:text="@string/genre_caps"
100                 android:textColor="@color/colorPrimary"
101                 android:textSize="15sp" />
102
103             <EditText
104                 android:id="@+id/genre"
105                 android:layout_width="match_parent"
106                 android:layout_height="wrap_content"
107                 android:layout_gravity="center"
108                 android:autofillHints="@string/genre_caps"
109                 android:ems="10"
110                 android:inputType="text"
111                 android:maxLength="10"
112                 android:textAllCaps="true"
113                 android:textColor="@color/colorAccent" />
114
115         </LinearLayout>
116
117     </ScrollView>
118
```

```
119     <com.bookstore.palvindersingh.networkFAB
120         android:id="@+id/correct"
121         android:layout_width="wrap_content"
122         android:layout_height="wrap_content"
123         android:layout_margin="16dp"
124         android:layout_marginEnd="8dp"
125         android:layout_marginBottom="8dp"
126         android:clickable="true"
127         android:focusable="true"
128         android:src="@drawable/ic_done_24dp"
129         app:backgroundTint="@color/fui_bgPhone"
130         app:fabSize="normal"
131         app:layout_constraintBottom_toBottomOf="parent"
132         app:layout_constraintEnd_toEndOf="parent" />
133
134     <TextView
135         android:layout_width="wrap_content"
136         android:layout_height="wrap_content"
137         android:layout_marginEnd="8dp"
138         android:layout_marginBottom="8dp"
139         android:fontFamily="@font/arimo"
140         android:padding="10dp"
141         android:text="Submit"
142         android:textColor="#000"
143         android:textSize="15sp"
144         app:layout_constraintBottom_toBottomOf="@+id/correct"
145         app:layout_constraintEnd_toStartOf="@+id/correct" />
146
147 </android.support.constraint.ConstraintLayout>
```

6.22 *activitydetails.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:padding="5dp"
    tools:context=".detailsActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0.5"
        android:contentDescription="@string/app_name"
        android:scaleType="center"
        android:scaleX="1.5"
        android:scaleY="1.5"
        android:src="@mipmap/ic_launcher_foreground" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:layout_weight="0.5"
        android:text="@string/enter_your_date_of_birth"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary" />

    <DatePicker
        android:id="@+id/datePicker"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:layout_weight="0.5"
        android:calendarViewShown="false"
        android:datePickerMode="spinner" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50dp"
        android:layout_marginRight="50dp"
        android:text="@string/enter_these_details"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary" />

    <EditText
        android:id="@+id/yearGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginRight="100dp"
        android:layout_weight="0.5"
        android:autofillHints="@string/year_group"
        android:hint="@string/year_group"
```

```
59          android:inputType="number"
60          android:textAlignment="center"
61          android:textSize="7.5pt" />
62
63      <EditText
64          android:id="@+id/formRoom"
65          android:layout_width="match_parent"
66          android:layout_height="wrap_content"
67          android:layout_marginLeft="100dp"
68          android:layout_marginRight="100dp"
69          android:layout_weight="0.5"
70          android:autofillHints="@string/form_room"
71          android:hint="@string/form_room"
72          android:inputType="text"
73          android:maxLength="5"
74          android:textAlignment="center"
75          android:textSize="7.5pt" />
76
77      <Button
78          android:id="@+id/submitData"
79          android:layout_width="match_parent"
80          android:layout_height="wrap_content"
81          android:layout_marginLeft="75dp"
82          android:layout_marginRight="75dp"
83          android:background="@color/colorPrimary"
84          android:onClick="submitData"
85          android:text="@string/submit"
86          android:textColor="@android:color/white" />
87
88  </LinearLayout>
```

6.23 *activityhome.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".homeActivity">


    <android.support.v7.widget.Toolbar
        android:id="@+id/my_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <FrameLayout
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="2"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

    <android.support.design.widget.BottomNavigationView
        android:id="@+id/navigation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:layout_weight="0"
        android:background="?android:attr/windowBackground"
        app:menu="@menu/bottom_nav" />

</LinearLayout>
```

6.24 *activitylogin.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".logInActivity">

</LinearLayout>
```

6.25 *activitymain.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    tools:context=".mainActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/app_name"
        android:scaleType="center"
        android:scaleX="2"
        android:scaleY="2"
        android:src="@mipmap/ic_launcher_foreground" />

</LinearLayout>
```

6.26 *activitynonetwork.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SwipeRefreshLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/swipeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".noNetworkActivity">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imageView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:contentDescription="@string/app_name"
            android:scaleType="center"
            android:scaleX="3"
            android:scaleY="3"
            android:src="@drawable/ic_signal_wifi_off_24dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="32dp"
            android:layout_marginEnd="8dp"
            android:fontFamily="@font/arimo"
            android:text="@string/no_network"
            android:textColor="@color/colorPrimary"
            android:textSize="25sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/imageView2" />

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:fontFamily="@font/arimo"
            android:text="@string/swipe_down_to_refresh"
            android:textColor="@color/colorPrimary"
```

```
58              android:textSize="12sp"
59              app:layout_constraintEnd_toEndOf="parent"
60              app:layout_constraintStart_toStartOf="parent"
61              app:layout_constraintTop_toBottomOf="@+id/textView2" />
62
63      </android.support.constraint.ConstraintLayout>
64
65  </android.support.v4.widget.SwipeRefreshLayout>
```

6.27 *activityscanbook.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".scanBookActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <com.otaliastudios.cameraview.CameraView
        android:id="@+id/camera"
        android:layout_width="match_parent"
        android:layout_height="747dp"
        android:keepScreenOn="true"
        app:cameraAudio="off"
        app:cameraJpegQuality="25"
        app:cameraVideoQuality="max480p"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar" />

    <TextView
        android:id="@+id/isbn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:padding="10dp"
        android:text="@string/isbn"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/view" />

    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/scan"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:clickable="true"
        android:focusable="true"
        app:fabSize="normal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@drawable/ic_add_a_photo_24dp" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/testButton"
        android:layout_width="wrap_content"
```

```
58          android:layout_height="wrap_content"
59          android:layout_margin="16dp"
60          android:layout_marginBottom="8dp"
61          android:clickable="true"
62          android:focusable="true"
63          app:backgroundTint="@color/fui_bgTwitter"
64          app:fabSize="mini"
65          app:layout_constraintBottom_toTopOf="@+id/flash"
66          app:layout_constraintEnd_toEndOf="@+id/flash"
67          app:layout_constraintStart_toStartOf="@+id/flash"
68          app:srcCompat="@android:drawable/alert_light_frame" />
69
70      <android.support.design.widget.FloatingActionButton
71          android:id="@+id/manual"
72          android:layout_width="wrap_content"
73          android:layout_height="wrap_content"
74          android:layout_margin="16dp"
75          android:layout_marginBottom="8dp"
76          android:clickable="true"
77          android:focusable="true"
78          app:backgroundTint="@color/browser_actions_title_color"
79          app:fabSize="mini"
80          app:layout_constraintBottom_toTopOf="@+id/scan"
81          app:layout_constraintEnd_toEndOf="@+id/scan"
82          app:layout_constraintStart_toStartOf="@+id/scan"
83          app:srcCompat="@drawable/ic_edit_24dp" />
84
85      <android.support.design.widget.FloatingActionButton
86          android:id="@+id/flash"
87          android:layout_width="wrap_content"
88          android:layout_height="wrap_content"
89          android:layout_margin="16dp"
90          android:layout_marginStart="8dp"
91          android:layout_marginBottom="8dp"
92          android:clickable="true"
93          android:focusable="true"
94          app:backgroundTint="@color/fui_bgAnonymous"
95          app:fabSize="normal"
96          app:layout_constraintBottom_toBottomOf="parent"
97          app:layout_constraintStart_toStartOf="parent"
98          app:srcCompat="@drawable/ic_flash_on_24dp" />
99
100     <View
101         android:id="@+id/view"
102         android:layout_width="match_parent"
103         android:layout_height="150dp"
104         android:layout_marginStart="32dp"
105         android:layout_marginTop="8dp"
106         android:layout_marginEnd="32dp"
107         android:layout_marginBottom="8dp"
108         android:background="@drawable/isbn_bounds"
109         app:layout_constraintBottom_toBottomOf="parent"
110         app:layout_constraintEnd_toEndOf="parent"
111         app:layout_constraintStart_toStartOf="parent"
112         app:layout_constraintTop_toTopOf="@+id/my_child_toolbar" />
113
114     <TextView
115         android:layout_width="wrap_content"
116         android:layout_height="wrap_content"
117         android:layout_marginEnd="8dp"
118         android:fontFamily="@font/arimo"
```

```
119         android:padding="10dp"
120         android:text="Manual"
121         android:textColor="@color/white"
122         android:textSize="15sp"
123         app:layout_constraintBottom_toBottomOf="@+id/manual"
124         app:layout_constraintEnd_toStartOf="@+id/manual"
125         app:layout_constraintTop_toTopOf="@+id/manual" />
126
127     <TextView
128         android:id="@+id/textView3"
129         android:layout_width="wrap_content"
130         android:layout_height="wrap_content"
131         android:layout_marginEnd="8dp"
132         android:fontFamily="@font/arimo"
133         android:padding="10dp"
134         android:text="Scan"
135         android:textColor="@color/white"
136         android:textSize="15sp"
137         app:layout_constraintBottom_toBottomOf="@+id/scan"
138         app:layout_constraintEnd_toStartOf="@+id/scan"
139         app:layout_constraintTop_toTopOf="@+id/scan" />
140
141     <TextView
142         android:layout_width="wrap_content"
143         android:layout_height="wrap_content"
144         android:layout_marginStart="8dp"
145         android:fontFamily="@font/arimo"
146         android:padding="10dp"
147         android:text="Flash"
148         android:textColor="@color/white"
149         android:textSize="15sp"
150         app:layout_constraintBottom_toBottomOf="@+id/flash"
151         app:layout_constraintStart_toEndOf="@+id/flash"
152         app:layout_constraintTop_toTopOf="@+id/flash" />
153
154 </android.support.constraint.ConstraintLayout>
```

6.28 *activityviewbook.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".viewBookActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingLeft="50dp"
            android:paddingTop="50dp"
            android:paddingRight="50dp">

            <ImageView
                android:id="@+id/bookImage"
                android:layout_width="125dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:adjustViewBounds="true"
                android:contentDescription="@string/books"
                android:scaleType="fitXY"
                android:visibility="gone" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="50dp"
                android:background="@color/browser_actions_bg_grey"
                android:divider="?android:listDivider"
                android:orientation="vertical"
                android:showDividers="middle">

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:elevation="10dp"
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="About"
                    android:textColor="#000"
                    android:textSize="15sp" />
```

```
58
59                  <TextView
60                      android:id="@+id/isbn"
61                      android:layout_width="match_parent"
62                      android:layout_height="wrap_content"
63                      android:elevation="10dp"
64                      android:fontFamily="@font/arimo"
65                      android:padding="10dp"
66                      android:text="@string/isbn"
67                      android:textColor="@color/colorPrimaryDark"
68                      android:textSize="15sp"
69                      android:visibility="gone" />
70
71                  <TextView
72                      android:id="@+id/bookTitle"
73                      android:layout_width="match_parent"
74                      android:layout_height="wrap_content"
75                      android:fontFamily="@font/arimo"
76                      android:padding="10dp"
77                      android:text="@string/title"
78                      android:textAllCaps="true"
79                      android:textColor="@color/colorPrimaryDark"
80                      android:textSize="15sp"
81                      android:visibility="gone" />
82
83                  <TextView
84                      android:id="@+id/bookAuthors"
85                      android:layout_width="match_parent"
86                      android:layout_height="wrap_content"
87                      android:fontFamily="@font/arimo"
88                      android:padding="10dp"
89                      android:text="@string/authors"
90                      android:textAllCaps="true"
91                      android:textColor="@color/colorPrimaryDark"
92                      android:textSize="15sp"
93                      android:visibility="gone" />
94
95                  <TextView
96                      android:id="@+id/bookGenres"
97                      android:layout_width="match_parent"
98                      android:layout_height="wrap_content"
99                      android:fontFamily="@font/arimo"
100                     android:padding="10dp"
101                     android:text="@string/genres"
102                     android:textAllCaps="true"
103                     android:textColor="@color/colorPrimaryDark"
104                     android:textSize="15sp"
105                     android:visibility="gone" />
106
107                 <TextView
108                     android:id="@+id/bookMarket"
109                     android:layout_width="match_parent"
110                     android:layout_height="wrap_content"
111                     android:fontFamily="@font/arimo"
112                     android:padding="10dp"
113                     android:text="@string/market"
114                     android:textAllCaps="true"
115                     android:textColor="@color/colorPrimaryDark"
116                     android:textSize="15sp"
117                     android:textStyle="bold"
118                     android:visibility="gone" />
```

```
119
120             </LinearLayout>
121
122         </LinearLayout>
123
124     </ScrollView>
125
126     <com.bookstore.palvindersingh.networkFAB
127         android:id="@+id/delete"
128         android:layout_width="wrap_content"
129         android:layout_height="wrap_content"
130         android:layout_margin="16dp"
131         android:layout_marginEnd="8dp"
132         android:layout_marginBottom="8dp"
133         android:clickable="true"
134         android:focusable="true"
135         android:src="@drawable/ic_delete_24dp"
136         android:visibility="visible"
137         app:backgroundTint="@color/fui_bgEmail"
138         app:fabSize="normal"
139         app:layout_constraintBottom_toBottomOf="parent"
140         app:layout_constraintEnd_toEndOf="parent" />
141
142     <com.bookstore.palvindersingh.networkFAB
143         android:id="@+id/market"
144         android:layout_width="wrap_content"
145         android:layout_height="wrap_content"
146         android:layout_margin="16dp"
147         android:layout_marginEnd="8dp"
148         android:layout_marginBottom="8dp"
149         android:clickable="true"
150         android:focusable="true"
151         android:src="@drawable/ic_local_grocery_store_24dp"
152         android:visibility="visible"
153         app:backgroundTint="@color/white"
154         app:fabSize="normal"
155         app:layout_constraintBottom_toTopOf="@id/delete"
156         app:layout_constraintEnd_toEndOf="parent" />
157
158     <TextView
159         android:id="@+id/deletename"
160         android:layout_width="wrap_content"
161         android:layout_height="wrap_content"
162         android:layout_marginEnd="8dp"
163         android:fontFamily="@font/arimo"
164         android:padding="10dp"
165         android:text="Delete"
166         android:textColor="#000"
167         android:textSize="15sp"
168         app:layout_constraintBottom_toBottomOf="@+id/delete"
169         app:layout_constraintEnd_toStartOf="@+id/delete"
170         app:layout_constraintTop_toTopOf="@+id/delete" />
171
172     <TextView
173         android:id="@+id/marketname"
174         android:layout_width="wrap_content"
175         android:layout_height="wrap_content"
176         android:layout_marginEnd="8dp"
177         android:fontFamily="@font/arimo"
178         android:padding="10dp"
179         android:text="Market"
```

```
180         android:textColor="#000"
181         android:textSize="15sp"
182         app:layout_constraintBottom_toBottomOf="@+id/market"
183         app:layout_constraintEnd_toStartOf="@+id/market"
184         app:layout_constraintTop_toTopOf="@+id/market" />
185
186 </android.support.constraint.ConstraintLayout>
```

6.29 *activityviewsearchbook.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".viewSearchBookActivity">

    <android.support.v7.widget.Toolbar
        android:id="@+id/my_child_toolbar"
        style="@style/toolbarTheme"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme"
        app:titleTextColor="@color/white" />

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/my_child_toolbar">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingLeft="50dp"
            android:paddingTop="50dp"
            android:paddingRight="50dp">

            <ImageView
                android:id="@+id/bookImage"
                android:layout_width="125dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:adjustViewBounds="true"
                android:contentDescription="@string/books"
                android:scaleType="fitXY"
                android:visibility="gone" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                android:background="@color/browser_actions_bg_grey"
                android:divider="?android:listDivider"
                android:orientation="vertical"
                android:showDividers="middle">

                <TextView
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:elevation="10dp"
                    android:fontFamily="@font/arimo"
                    android:padding="10dp"
                    android:text="About Book"
                    android:textColor="#000"
                    android:textSize="15sp" />
```

```
58
59                  <TextView
60                      android:id="@+id/isbn"
61                      android:layout_width="match_parent"
62                      android:layout_height="wrap_content"
63                      android:elevation="10dp"
64                      android:fontFamily="@font/arimo"
65                      android:padding="10dp"
66                      android:text="@string/isbn"
67                      android:textColor="@color/colorPrimaryDark"
68                      android:textSize="15sp"
69                      android:visibility="gone" />
70
71                  <TextView
72                      android:id="@+id/bookTitle"
73                      android:layout_width="match_parent"
74                      android:layout_height="wrap_content"
75                      android:fontFamily="@font/arimo"
76                      android:padding="10dp"
77                      android:text="@string/title"
78                      android:textAllCaps="true"
79                      android:textColor="@color/colorPrimaryDark"
80                      android:textSize="15sp"
81                      android:visibility="gone" />
82
83                  <TextView
84                      android:id="@+id/bookAuthors"
85                      android:layout_width="match_parent"
86                      android:layout_height="wrap_content"
87                      android:fontFamily="@font/arimo"
88                      android:padding="10dp"
89                      android:text="@string/authors"
90                      android:textAllCaps="true"
91                      android:textColor="@color/colorPrimaryDark"
92                      android:textSize="15sp"
93                      android:visibility="gone" />
94
95                  <TextView
96                      android:id="@+id/bookGenres"
97                      android:layout_width="match_parent"
98                      android:layout_height="wrap_content"
99                      android:fontFamily="@font/arimo"
100                     android:padding="10dp"
101                     android:text="@string/genres"
102                     android:textAllCaps="true"
103                     android:textColor="@color/colorPrimaryDark"
104                     android:textSize="15sp"
105                     android:visibility="gone" />
106
107             </LinearLayout>
108
109             <ImageView
110                 android:layout_width="75dp"
111                 android:layout_height="wrap_content"
112                 android:layout_gravity="center"
113                 android:layout_marginTop="50dp"
114                 android:adjustViewBounds="true"
115                 android:contentDescription="@string/books"
116                 android:scaleType="fitXY"
117                 android:src="@drawable/ic_person_24dp" />
118
```

```
119             <LinearLayout
120                 android:layout_width="match_parent"
121                 android:layout_height="wrap_content"
122                 android:layout_marginBottom="100dp"
123                 android:background="@color/browser_actions_bg_grey"
124                 android:divider="?android:listDivider"
125                 android:orientation="vertical"
126                 android:showDividers="middle">
127
128                 <TextView
129                     android:layout_width="match_parent"
130                     android:layout_height="wrap_content"
131                     android:elevation="10dp"
132                     android:fontFamily="@font/arimo"
133                     android:padding="10dp"
134                     android:text="About User"
135                     android:textColor="#000"
136                     android:textSize="15sp" />
137
138                 <TextView
139                     android:id="@+id/name"
140                     android:layout_width="match_parent"
141                     android:layout_height="wrap_content"
142                     android:elevation="10dp"
143                     android:fontFamily="@font/arimo"
144                     android:padding="10dp"
145                     android:text="name"
146                     android:textColor="@color/colorPrimaryDark"
147                     android:textSize="15sp"
148                     android:visibility="gone" />
149
150                 <TextView
151                     android:id="@+id/year"
152                     android:layout_width="match_parent"
153                     android:layout_height="wrap_content"
154                     android:fontFamily="@font/arimo"
155                     android:padding="10dp"
156                     android:text="@string/year_group"
157                     android:textAllCaps="true"
158                     android:textColor="@color/colorPrimaryDark"
159                     android:textSize="15sp"
160                     android:visibility="gone" />
161
162                 <TextView
163                     android:id="@+id/form"
164                     android:layout_width="match_parent"
165                     android:layout_height="wrap_content"
166                     android:fontFamily="@font/arimo"
167                     android:padding="10dp"
168                     android:text="@string/form_room"
169                     android:textAllCaps="true"
170                     android:textColor="@color/colorPrimaryDark"
171                     android:textSize="15sp"
172                     android:visibility="gone" />
173
174                 <TextView
175                     android:id="@+id/rating"
176                     android:layout_width="match_parent"
177                     android:layout_height="wrap_content"
178                     android:fontFamily="@font/arimo"
179                     android:padding="10dp"
```

```
                    android:text="rating"
                    android:textAllCaps="true"
                    android:textColor="@color/colorPrimaryDark"
                    android:textSize="15sp"
                    android:visibility="gone" />

            </LinearLayout>

        </LinearLayout>

    </ScrollView>

    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/request"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:clickable="true"
        android:focusable="true"
        android:src="@drawable/ic_thumb_up_24dp"
        android:visibility="visible"
        app:fabSize="normal"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/rent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:layout_marginBottom="8dp"
        android:backgroundTint="@color/white"
        android:clickable="true"
        android:focusable="true"
        android:src="@drawable/ic_local_library"
        android:visibility="gone"
        app:fabSize="normal"
        app:layout_constraintBottom_toTopOf="@+id/request"
        app:layout_constraintEnd_toEndOf="@+id/request"
        app:layout_constraintStart_toStartOf="@+id/request" />

    <TextView
        android:id="@+id/rentingtext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:fontFamily="@font/arimo"
        android:padding="10dp"
        android:text="Rent"
        android:textColor="#000"
        android:textSize="15sp"
        app:layout_constraintBottom_toBottomOf="@+id/request"
        app:layout_constraintEnd_toStartOf="@+id/request"
        app:layout_constraintTop_toTopOf="@+id/request" />

    <TextView
        android:id="@+id/rentingtext2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
241         android:layout_marginEnd="8dp"
242         android:fontFamily="@font/arimo"
243         android:padding="10dp"
244         android:text="Stop Rent"
245         android:textColor="#000"
246         android:textSize="15sp"
247         android:visibility="gone"
248         app:layout_constraintBottom_toBottomOf="@+id/rent"
249         app:layout_constraintEnd_toStartOf="@+id/rent"
250         app:layout_constraintTop_toTopOf="@+id/rent" />
251
252
253 </android.support.constraint.ConstraintLayout>
```

6.30 *fragmentbooks.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/AppTheme"
    tools:context=".booksFragment">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:divider="?android:listDivider"
            android:orientation="vertical"
            android:showDividers="middle">

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="10dp"
                android:text="@string/scanned"
                android:textAllCaps="true"
                android:textColor="@color/colorPrimary" />

            <android.support.v7.widget.RecyclerView
                android:id="@+id/scannedBooksView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="3dp" />

            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="10dp"
                android:text="@string/renting"
                android:textAllCaps="true"
                android:textColor="@color/colorPrimary" />

            <android.support.v7.widget.RecyclerView
                android:id="@+id/rentingBooksView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="3dp" />

        </LinearLayout>

    </ScrollView>

    <com.bookstore.palvindersingh.networkFAB
        android:id="@+id/floating_action_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
```

```
59          app:fabSize="normal"
60          app:srcCompat="@drawable/ic_add_to_photos_white_24dp" />
61
62  </FrameLayout>
```

6.31 *fragmenthome.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:theme="@style/AppTheme"
    tools:context=".homeFragment">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:divider="?android:listDivider"
            android:orientation="vertical"
            android:showDividers="middle">

            <LinearLayout
                android:id="@+id/networkAvailability"
                android:layout_width="match_parent"
                android:layout_height="100dp"
                android:background="@color/browser_actions_bg_grey"
                android:gravity="center"
                android:orientation="vertical"
                android:visibility="gone">

                <ImageView
                    android:id="@+id/imageView2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:contentDescription="@string/app_name"
                    android:scaleType="center"
                    android:scaleX="2"
                    android:scaleY="2"
                    android:src="@drawable/ic_signal_wifi_off_24dp" />

                <TextView
                    android:id="@+id/textView2"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="10dp"
                    android:fontFamily="@font/arimo"
                    android:text="@string/no_network"
                    android:textColor="@color/colorPrimary"
                    android:textSize="15sp" />

            </LinearLayout>

        </LinearLayout>

    </ScrollView>

</FrameLayout>
```

6.32 *fragmentsearch.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".searchFragment">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:divider="?android:listDivider"
            android:orientation="vertical"
            android:showDividers="middle">

            <TextView
                android:id="@+id/results"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:padding="10dp"
                android:text="@string/results"
                android:textAllCaps="true"
                android:textColor="@color/colorPrimary" />

            <android.support.v7.widget.RecyclerView
                android:id="@+id/searchBooksView"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="100dp"
                android:padding="3dp" />

        </LinearLayout>

    </ScrollView>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:background="@color/white"
        android:orientation="horizontal"
        android:padding="3dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">

        <com.bookstore.palvindersingh.networkFAB
            android:id="@+id/floating_action_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="16dp"
```

```
58              android:layout_marginTop="8dp"
59              android:layout_marginEnd="8dp"
60              android:layout_marginBottom="8dp"
61              android:backgroundTint="@color/white"
62              android:src="@drawable/ic_search_24dp"
63              app:fabSize="normal"
64              app:layout_constraintBottom_toBottomOf="parent"
65              app:layout_constraintEnd_toEndOf="parent"
66              app:layout_constraintTop_toTopOf="parent" />
67
68          <EditText
69              android:id="@+id/search"
70              android:layout_width="0dp"
71              android:layout_height="0dp"
72              android:layout_gravity="bottom|end"
73              android:layout_marginStart="8dp"
74              android:layout_marginEnd="8dp"
75              android:hint="@string/search"
76              app:layout_constraintBottom_toBottomOf="@+id/floating_action_button"
77              app:layout_constraintEnd_toStartOf="@+id/floating_action_button"
78              app:layout_constraintStart_toStartOf="parent"
79              app:layout_constraintTop_toTopOf="@+id/floating_action_button" />
80
81      </android.support.constraint.ConstraintLayout>
82
83  </android.support.constraint.ConstraintLayout>
```

6.33 *scannedbookitem.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/parentLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:background="@color/cardview_light_background"
    android:clickable="true"
    android:focusable="true">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="8dp">

        <ImageView
            android:id="@+id/icon"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="clip_vertical"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp"
            android:src="@mipmap/ic_launcher"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/bookTitle"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginStart="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:padding="10dp"
            android:text="@string/title"
            android:textColor="@color/colorPrimary"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toStartOf="@+id/bookStatus"
            app:layout_constraintStart_toEndOf="@+id/icon"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/bookStatus"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginBottom="8dp"
            android:padding="10dp"
            android:text="status"
            android:textColor="@color/colorPrimary"
            android:textStyle="bold"
```

```
59              app:layout_constraintBottom_toBottomOf="parent"
60              app:layout_constraintEnd_toEndOf="parent"
61              app:layout_constraintTop_toTopOf="parent" />
62
63      </android.support.constraint.ConstraintLayout>
64
65  </android.support.v7.widget.CardView>
```

Bibliography

[1] https://help.uber.com/h/738d1ff7-5fe0-4383-b34c-4a2480efd71e

[2] https://play.google.com/store/apps/details?id=com.amazon.mShop.android.shopping

[3] https://firebase.google.com/docs/firestore/data-model

[4] https://stackoverflow.com/questions/36272642/asynctask-equivalent-in-java

[5] https://stackoverflow.com/questions/12035316/reading-entire-html-file-to-string

[6] https://developer.android.com/training/monitoring-device-state/connectivity-monitoring

[7] https://docs.oracle.com/javase/tutorial/networking/urls/readingURL.html

[8] https://developer.android.com/reference/android/os/Handler

[9] https://developer.android.com/training/appbar

[10] https://developer.android.com/reference/android/support/design/widget/BottomNavigationView

[11] https://stackoverflow.com/questions/29789057/class-must-either-be-declared-abstract-or-implement-abstract-method-error

[12] https://github.com/natario1/CameraView

[13] https://www.javatpoint.com/android-alert-dialog-example

[14] https://developer.android.com/reference/android/content/Intent

[15] https://firebase.google.com/docs/firestore/manage-data/transactions

[16] https://stackoverflow.com/questions/17549042/android-asynctask-passing-a-single-string

[17] https://stackoverflow.com/questions/9893813/can-an-android-asynctask-doinbackground-be-synchronized-to-serialize-the-task-ex

[18] https://firebase.google.com/docs/firestore/query-data/queries

[19] https://www.quora.com/Why-is-insertion-sort-faster-than-bubble-sort-while-having-the-same-big-O-notation

[20] https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter