System design document for book reselling application

Pajam Khoshnam, Alexander Jyborn, Albin Landgren, Oscar Bennet, Tarik Porobic

2020-09-30
Version 2.0

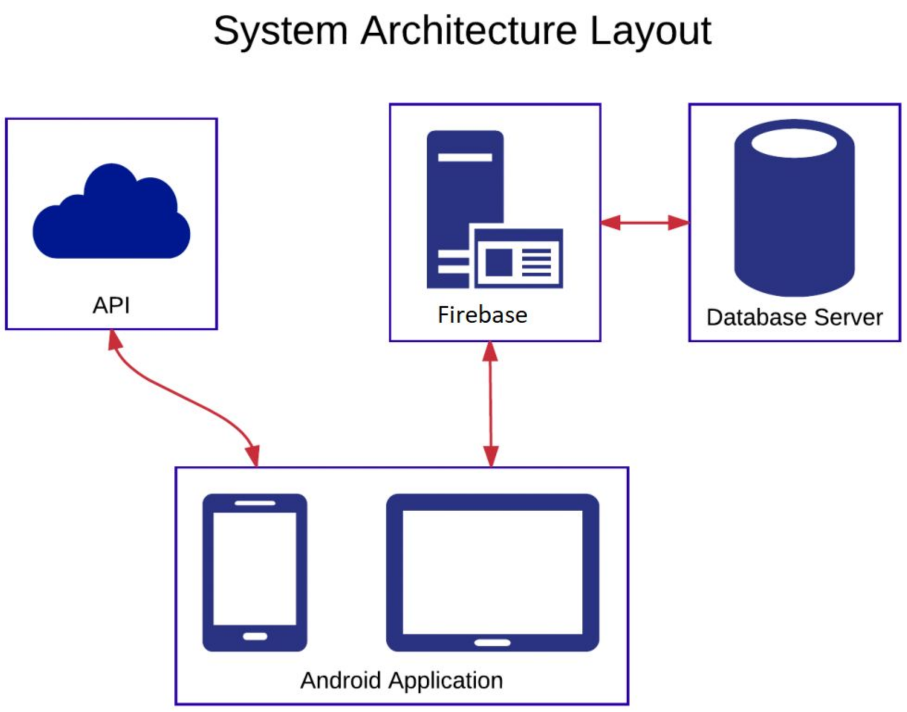
1 Introduction

This document describes the system design of the application Boket. Boket is an application that makes it easier for students to find and sell their used course literature. Students will be able to subscribe to books and get a notification when someone is selling the book. It will be easier to create an ad since the student will have the option to scan the barcode of the book to auto generate an ad.

1.1 Definitions, acronyms, and abbreviations

- app = the application
- user = an account in the application
- book = an book object used by the
- ad = an advertisement published by a user to sell a book
- database = data storage of the user, book and ad models
- firebase = handles the backend and data storage

1.2 System Architecture



The system consists of 3 different parts. The android application which is the part which the user will directly access. The Android Application will communicate with Firebase which will be used as backend and database. There are also few other API:s which the Android Application will communicate with, such as the search engine Algolia.

1.2.2 Design Constraints

The main design constraint of the system architecture is Firebase. Since Firebase will be used as both backend and database the features will be limited to what Firebase can do and offer. However, this allows for rapid development.

1.2.3 Future Contingencies

Firebase might not be able to provide all the features needed as a backend and database. There is also the possibility for them to shut down their service at any moment.

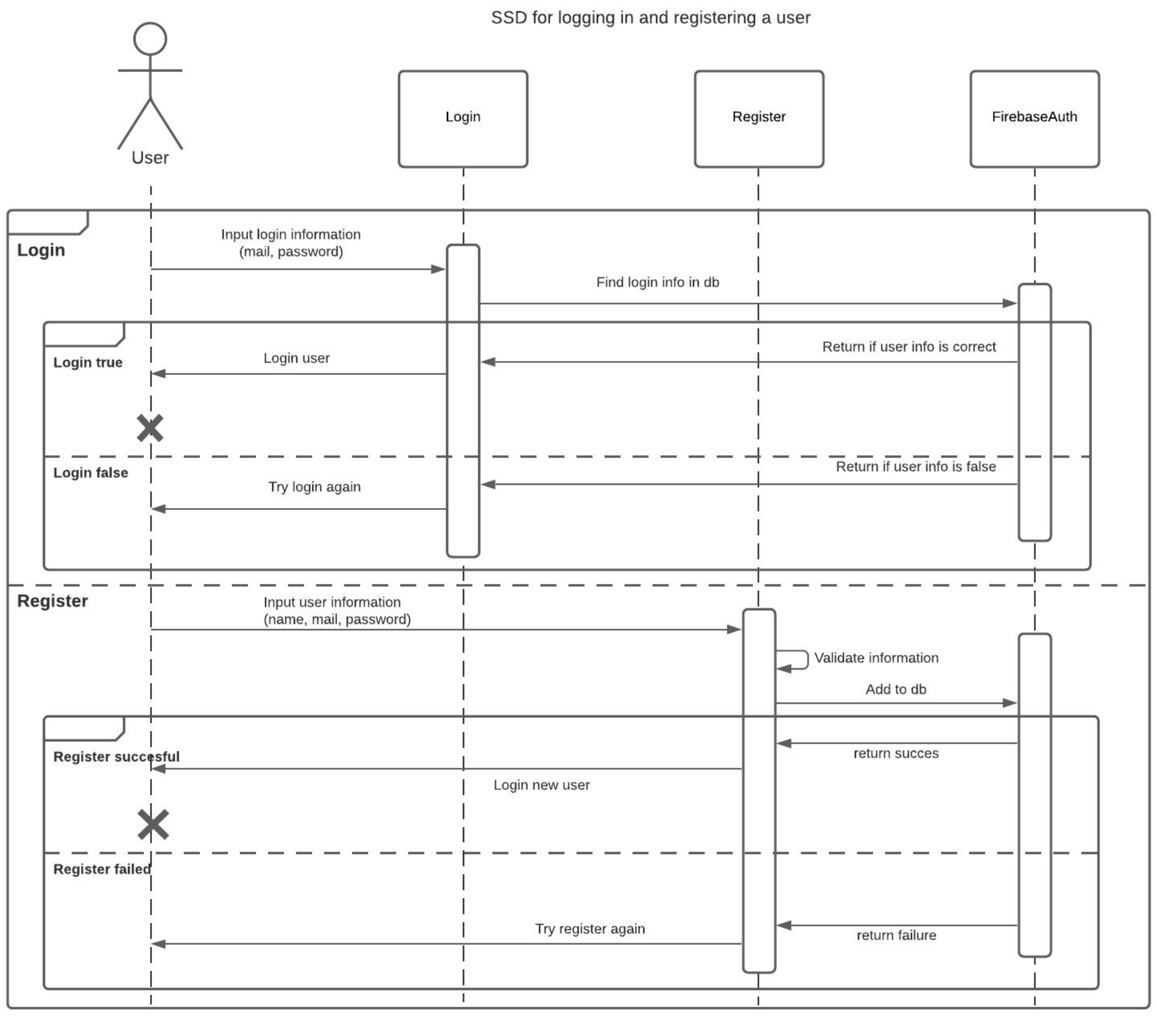
Possible workarounds is to eventually move to a fully owned and self-managed backend/database.

2 System architecture

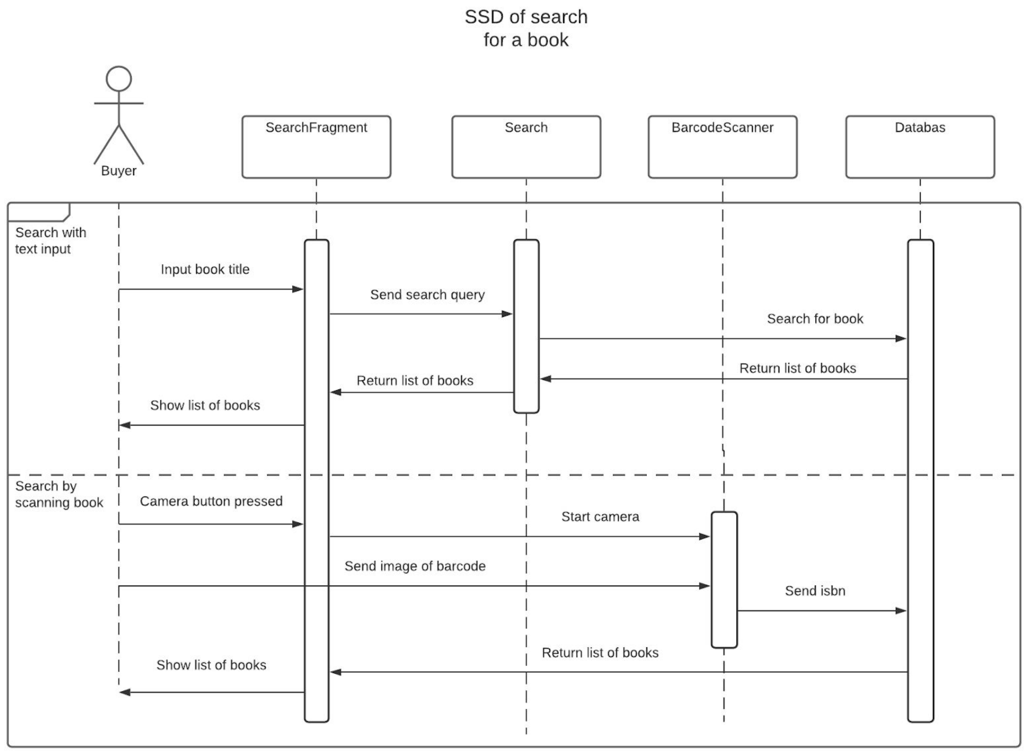
This section describes how the software architecture and design is planned to fulfill the requirements.

2.1 System Sequence Diagram

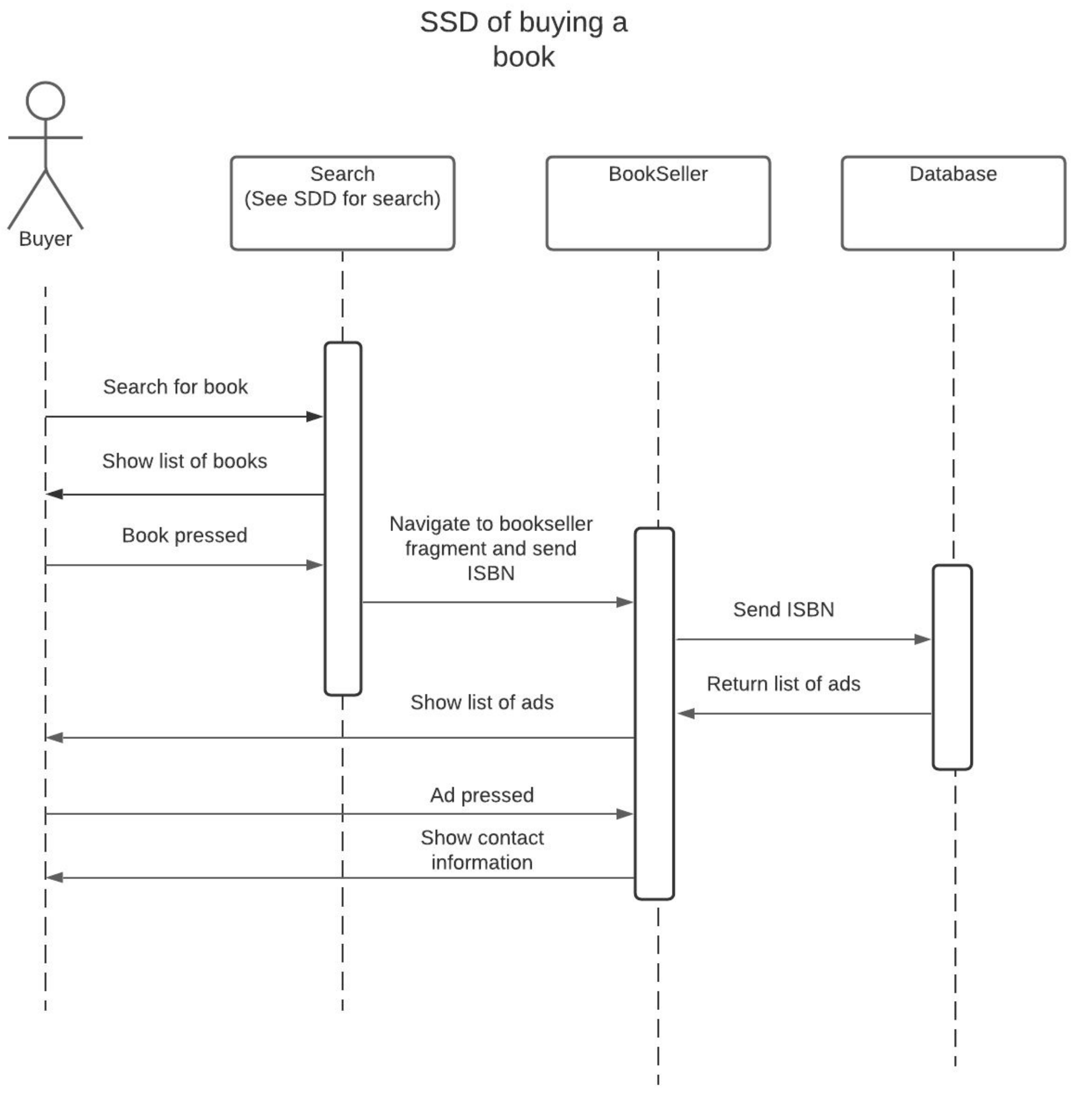
The SSD below illustrates the process of logging in to the application or registering to the application.



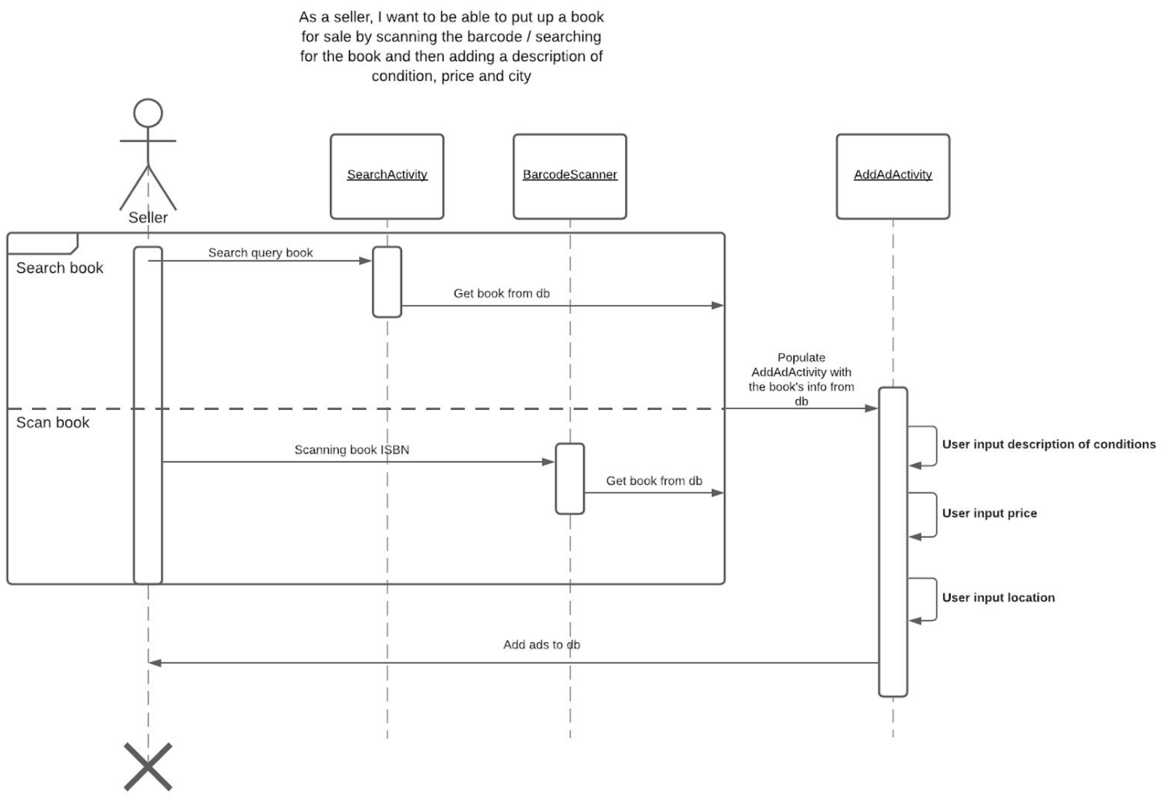
The following SSD shows the action of a buyer searching for a book. When searching for a book the user has two options, either with text input or scanning the barcode of the book.



The SSD below shows the act of a user buying a book.

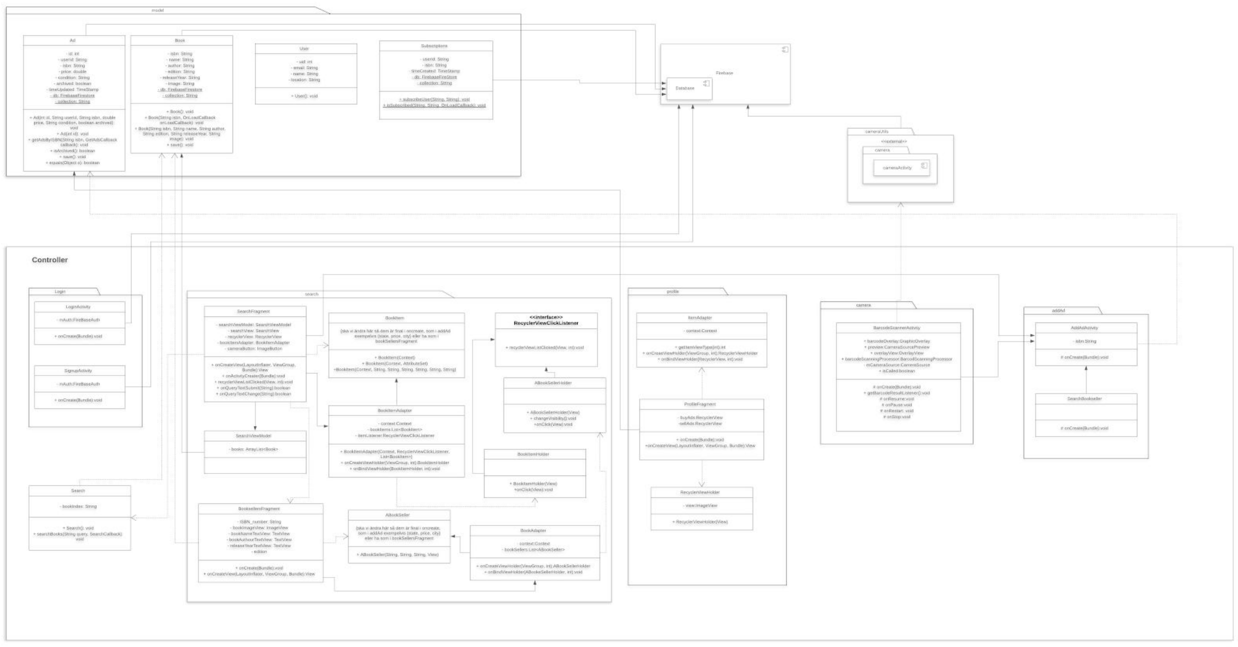


The SSD below illustrates the general process of publishing an Ad to the platform. The seller can either search for a book they want to sell or scan the ISBN number from the book with the phone camera. When the book has been found in the database, the book info such as title, author, edition and isbn are added to the AddAdActivity where the seller can complete the ad by inputting a description of the book's condition, price and location of sale.


As a seller, I want to be able to put up a book for sale by scanning the barcode / searching for the book and then adding a description of condition, price and city
Seller
SearchActivity
BarcodeScanner
AddAdActivity
Search book
Search query book
Get book from db
Populate AddAdActivity with the book's info from db
Scan book
Scanning book ISBN
Get book from db
User input description of conditions
User input price
User input location
Add ads to db

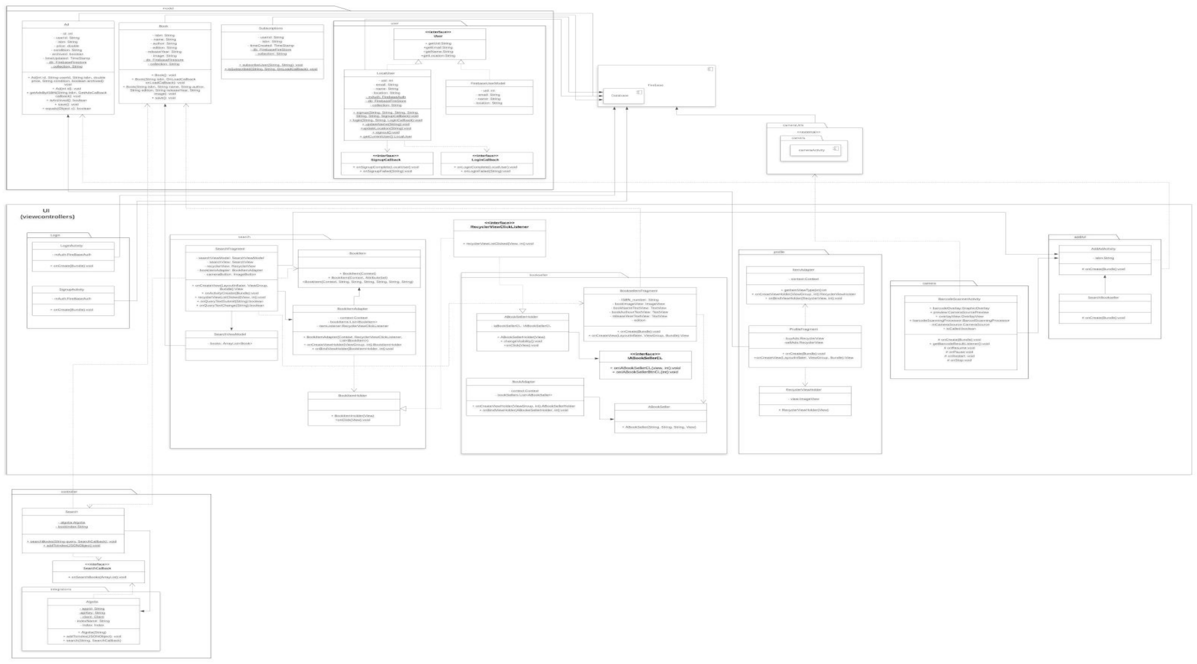
2.2 UML class diagram

Iteration 1:



Attached UML -Epic project 1337.pdf

Iteration 2:



Attached UML - Epic project 1337 - Class UML 2.0.pdf

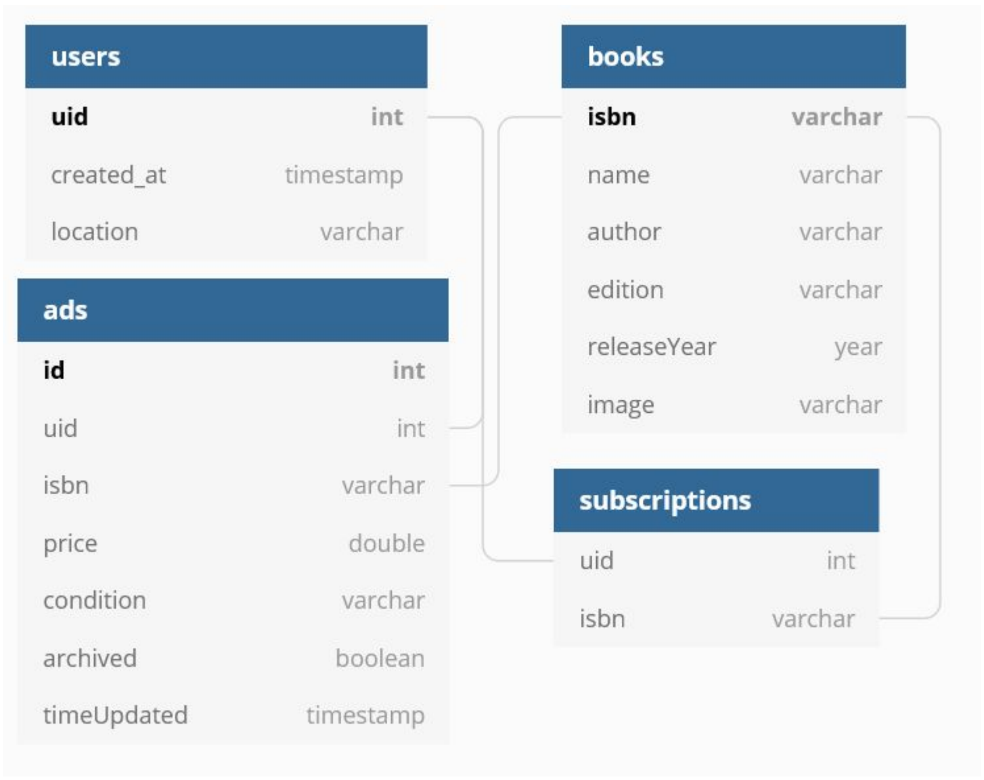
See also Attached UML - Epic project 1337 - Class UML 2.0 no attributes.pdf

3 File and Database Design

3.1 Database Management System Files

Firebase Firestore will be used as the database for the application to store persistent data such as user data, ads and books. Firebase Firestore is based on a NoSQL database.

The database will consist of 4 main documents. Users, Ads, Books and Subscriptions. Since it is a NoSQL database there are no relations built into the database structure. The diagram below will show the different documents and how they are related to each other in the application.



3.2 Design patterns

3.2.1 Adapter pattern

Adapter pattern is used every time an item is added to a recycler view. The reason for this is that the type of the item needs to be changed to fit the recycler view.

3.2.2 Singleton

Is mainly used in the LocalUser class since there can only be **one** local user at any given time and this instance is required in almost every part of the application.

3.2.3 MVC

The entire project is based on a Model View Controller structure.

3.2.4 Command pattern

Android's inbuilt fragments use the command pattern to swap between multiple fragments by using a fragment manager.

Command pattern is also used by the ad model to create an ad and then save it.

3.2.5 Builder pattern

Firebase SDK uses the builder pattern for certain requests which is used in our application.

3.2.6 Facade pattern

Facade pattern is used in basically all model classes. You can for example retrieve a book from the Book object without understanding the inner workings of how the book is stored, fetched etc.

When you save a book it you can also do easily with the Book class without understanding how the book is stored, how it is added to the search index, background requests etc.

The same applies for all other model classes.

3.2.7 Observer Pattern

Observer pattern is used in a lot of different places in our UI controllers.

4 Quality

4.1 Testing

The application is tested by testing the functionality in the android emulator as well as on a physical device. Functionality that can be tested fairly easy by unit testing is tested that way.

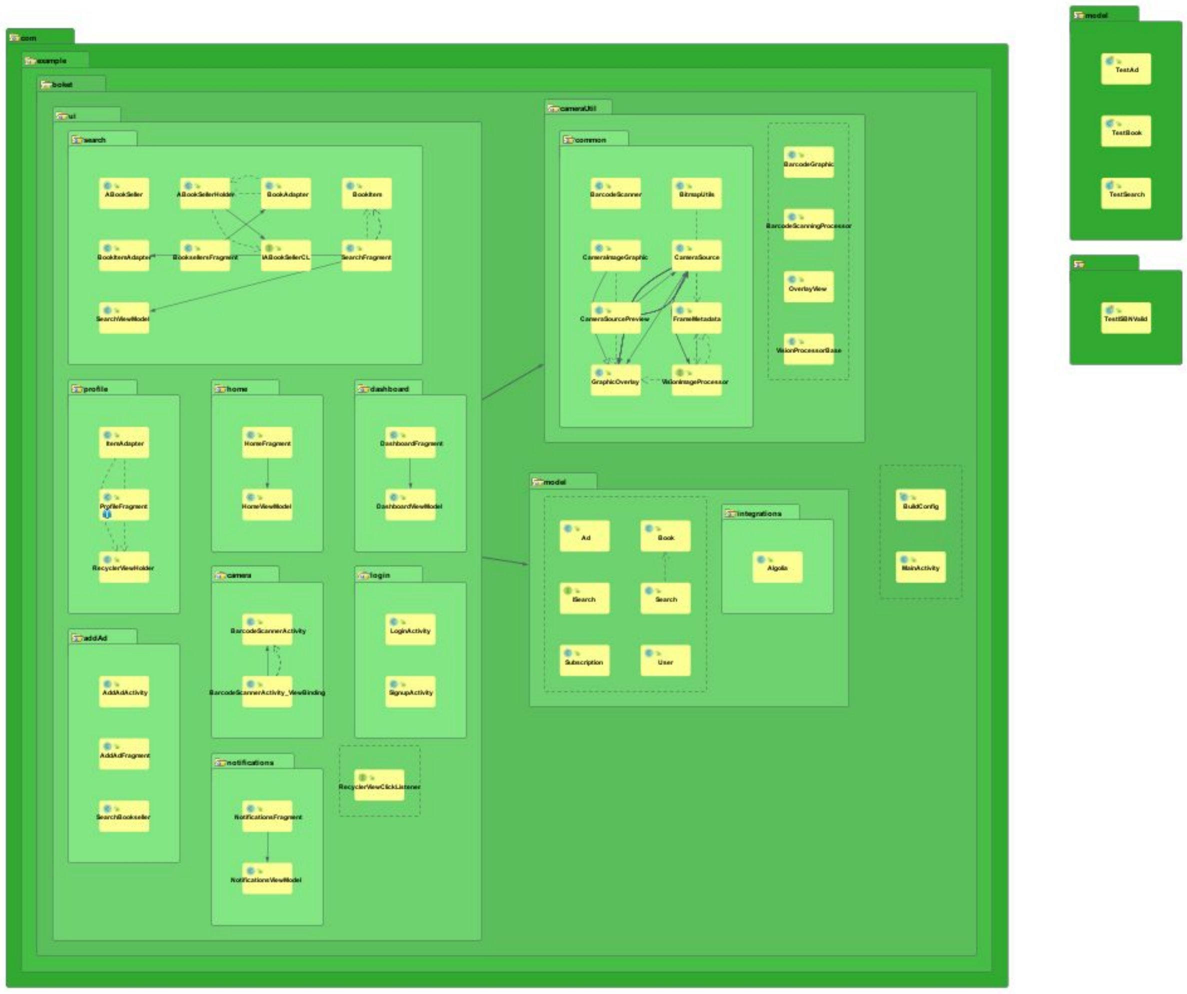
Unit testing can be found in the androidTest folder in the source code. For some of these tests to work you have to be logged into the app already.

4.2 Known issues

- Camera is not performance optimised
- Navigation backstack has a few issues
- Styling is not fully implemented
- Can not edit published ads (not implemented)
- When in profile and choosing a book and then pressing back button, you end up on the search fragment instead of profile fragment.
- You can’t retrieve your password if forgotten
- When you press the contact seller button and just go back to the app (without back arrow) and later choose another seller the text doesn't update in the gmail app. You have to either send the message or press any back button.
- It handles wrong information filled when trying to log in, but not as well as we wanted(Not fully implemented)

4.3 Results from analytical tools

Module dependencies analysed by Code Iris:



In this diagram we can see that there are few dependencies between packages and modules which means the code base has high cohesion and low coupling.

4.4 Access control and security

To use the application you have to register an account and be logged in. This is all managed with the Firebase Authentication library. Access control is managed by setting up different Firebase access rules.

This route allows for very rapid development, however, it also makes the application highly dependent on Firebase services.

5 References

Firebase (https://firebase.google.com)
Glide (https://github.com/bumptech/glide)
Algolia (https://www.algolia.com/)