UNIVERSITY OF ALASKA ANCHORAGE

CSCE A470

CAPSTONE PROJECT

The Android Textbook APP

Author:

Yu Jian Zhao

Supervisor:

Prof. Kenrick Mock, PhD

Anchorage AK, May 2016

Computer Science & Engineering Department
UNIVERSITY of ALASKA ANCHORAGE

yjzhao@alaska.edu

Version 0

Abstract

With the capacity and capability of our computing device increases every year, the mobile application market has been growing exponentially with millions of Android and IOS users. The goal of this project is to design and build an Android application that allows college students to trade their used textbooks which will save them a lot of time and money. This project includes front-end client which is the Android application itself that will be discussed in details in this report, and back-end server that will be implemented by my project coworker Gabriel Esposito which includes a MySQL databased and REST API implemented in Python that will be interfacing with the client.

Acknowledgement

During the development of the application I received helps from various people. Without them I would not have completed this project.

I would like to thank my project coworker Gabriel Esposito for this dedication on the server side and helping me bridging the Android client with the server.

I would like to thank my project supervisor Dr. Mock for supervising our project, provided server to host our client data, and helped us on numerous occasions.

I would like to thank my capstone professor Dr. Cavalcanti for cultivating us on the subject of capstone project and other interesting topics that he had talked about in the class.

I would also like to thank all UAA professors that taught me new skills and knowledges. Without them I would not be able to complete this project and this program.

Table of Contents

Chapter 1 Introduction **4**
1.1 Introduction 4
1.2 The Application 5
1.3 Motivation 8

Chapter 2 System Integration, Modelling and Methodology **9**
2.1 Introduction 9
2.2 The System Architecture 10
2.3 Android Client 10
2.4 Google Endpoint Services 11
2.5 Server and Data 11
2.6 Agile Methodology 12
2.7 Gantt Chart 14

Chapter 3 Design and Testing **15**
3.1 Introduction 15
3.2 Design 15
3.3 Testing 19

Chapter 4 User's manual **20**
4.1 Product Introduction 20
4.2 Installation 20
4.3 Use and Navigation 21
4.4 Services and Permissions 28
4.5 Data and Privacy 28
4.6 Help 28

Chapter 5 Conclusion **29**
5.1 Project Result 29
5.2 Future Development 29
5.3 Conclusion 30

References

Appendix A
UML Diagram

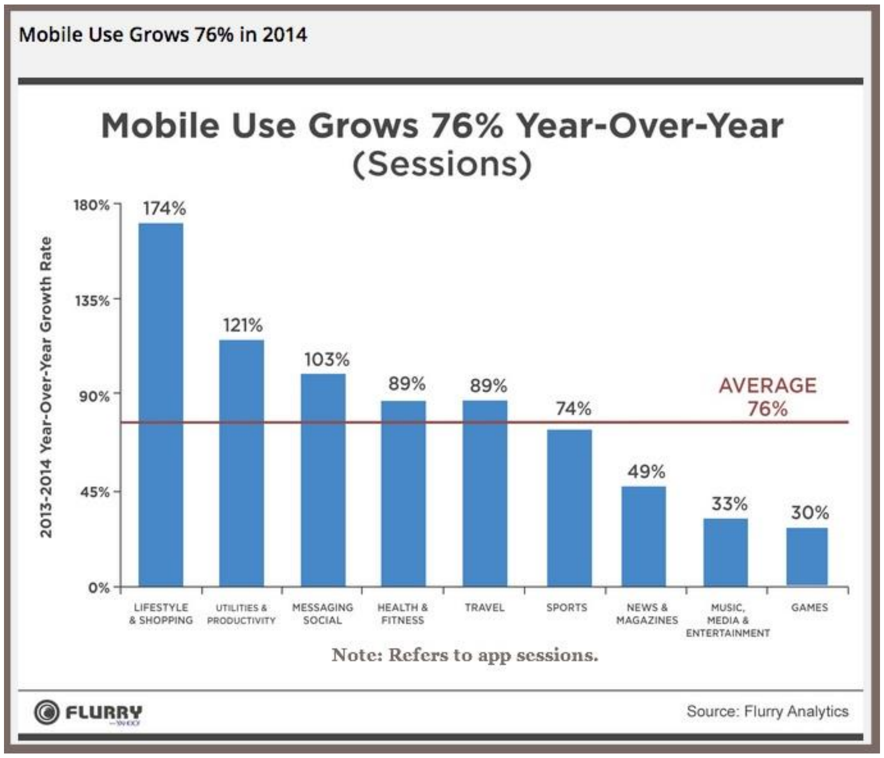
Appendix B
Initial Source Code

Chapter 1

Introduction

1.1 Introduction

The market for mobile apps are increasing exponentially. 178 million people in the US owned smartphones, 73.6% mobile market penetration according to comScore based on November 2014 data. Smartphones are a mature market in the US [1]. With the capacity and capability of our computing device increases every year, mobile apps are also having more and more functionalities. Mobile apps are now capable of doing so many things. It is important for businesses to consider and adapt mobile platform to their services. If Wells Fargo hadn't gone out mobile app five years ago, it would be out of business, said its CEO and Chairman John Stumpf in 2015 [2]. Particularly, mobile app market in android has boosted a lot, 5% more than IOS's. Android developers saw a 6% salary increase, but it was only a 1% hike for iOS developers, survey finds [3].



[1] Figure 1.1: Flurry Mobile App Trend

1.2 The Application

With mobile market growing and we see the need of a good mobile app for students to trade textbooks, we are purposing to make an Android App that will help college students to buy and sell their used textbooks between students.

The application will be similar to Craigslist but more powerful in terms of usability and user-friendly since we will be using barcode scanner to automatically populate book data. An in-app non-real time communication will also be possible. Users will be able to search nearby sales posted by other students with the ability to sort by college courses if it is provided by the seller. We will be using Agile Software development [4].

The following diagrams provides a good overview of the application infrastructure.

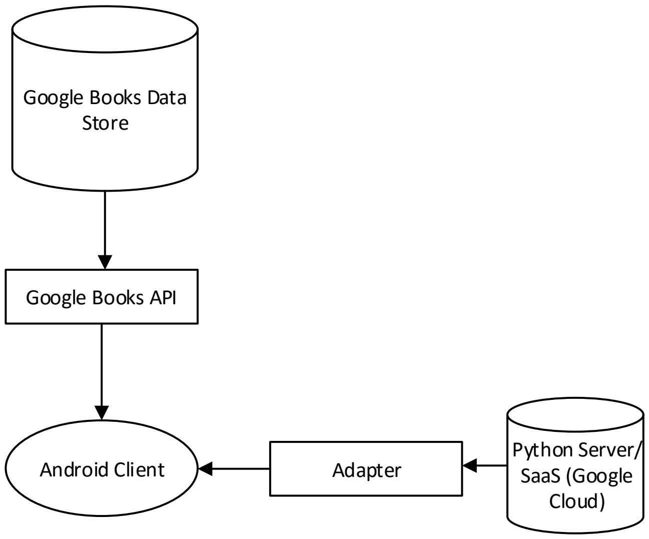


Figure 1.2: Proposed System Structure

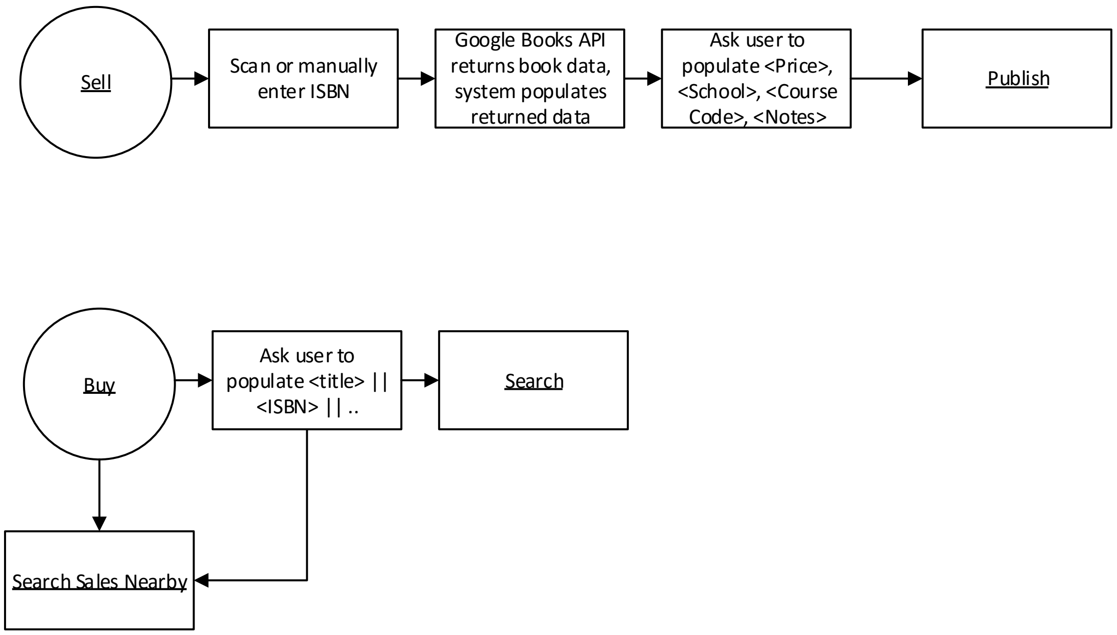

Sell
Scan or manually enter ISBN
Google Books API returns book data, system populates returned data
Ask user to populate <Price>, <School>, <Course Code>, <Notes>
Publish
Buy
Ask user to populate <title> || <ISBN> || ..
Search
Search Sales Nearby

Figure 1.3.1: Proposed Workflow

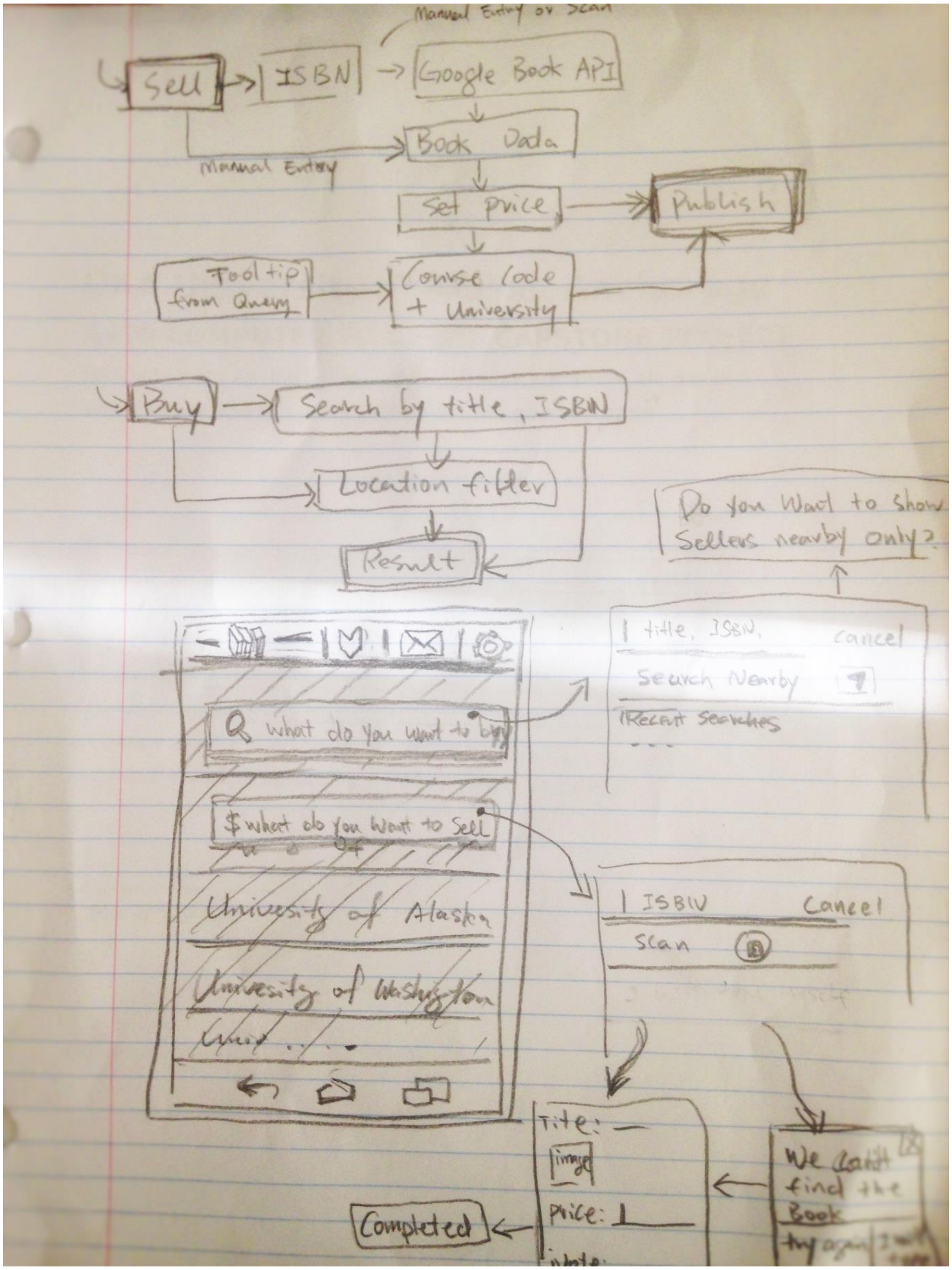
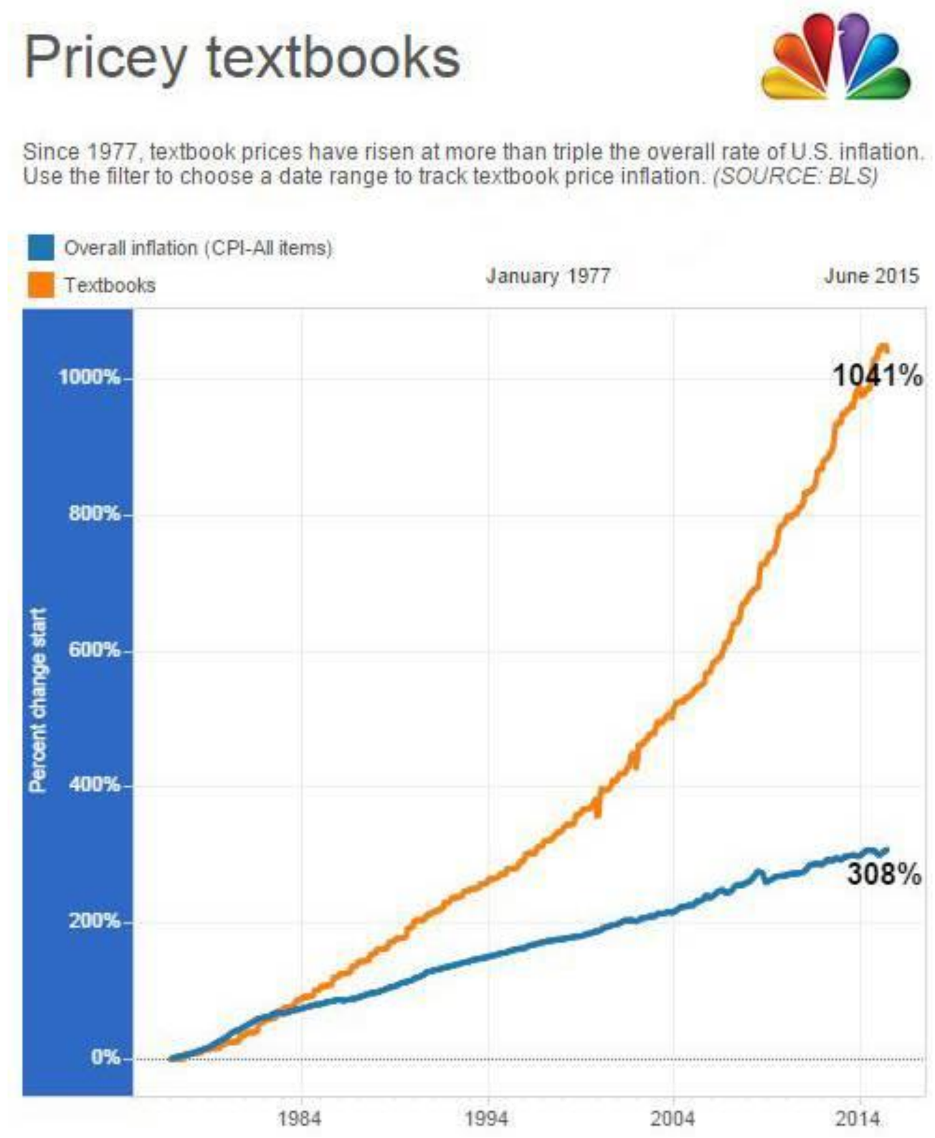


Figure 1.3.2: Proposed Workflow and GUI

1.3 Motivation

The motivation behind this application is that we as students are experiencing and seeing students paying high price for their textbooks and sell them back at a much lower price to bookstore or online retailers who then sell these textbooks back to other students at two or more times of the buyback price. I see this is just plain wrong that college students with no or part-time job carrying a stack of student loans paying unreasonable price for their textbooks, so I was thinking to make a mobile app to serve as a platform for students themselves to trade their textbooks. Mobile device is so common and frequently used today especially among students, so we believe it will be a good platform for the application. The application will be completely free to use but transaction of the textbooks will be between students the app itself does not handle payments of any type.



[5]Figure 1.4: Textbook prices have risen higher than overall inflation for the past three decades

Chapter 2
System Integration, Modelling and Methodology

2.1 Introduction

The Android application, namely, the client will be implemented using agile methodology allowing quick iterations of software development cycle, finding bugs and design flaws early on the process, hence improve programmer and customer satisfaction. HTTP protocol will be used as communication protocol between the client and server. Various Google endpoints services will be used to assist accomplishing the app functionality as well. All these components will be explained in details in next few sections.

Android Studio will be used to develop the android application. Dr. Mock has provided us a server for the back-end, and will be configured and used by the client. The android client requires minimum Android SDK version 15 and targeted at version 23.

2.2 The System Architecture

We will be using server-client model for the system. The android application will be the client and the server is what Dr. Mock provided for us which is a Linux server (Ubuntu) along with Apache to handle HTTP request, MySQL for database, Python for back-end scripting. Google Books API, Google Play Location Service, and OAuth (Google+) will be used in the application. The diagram below presents the overall system architecture:

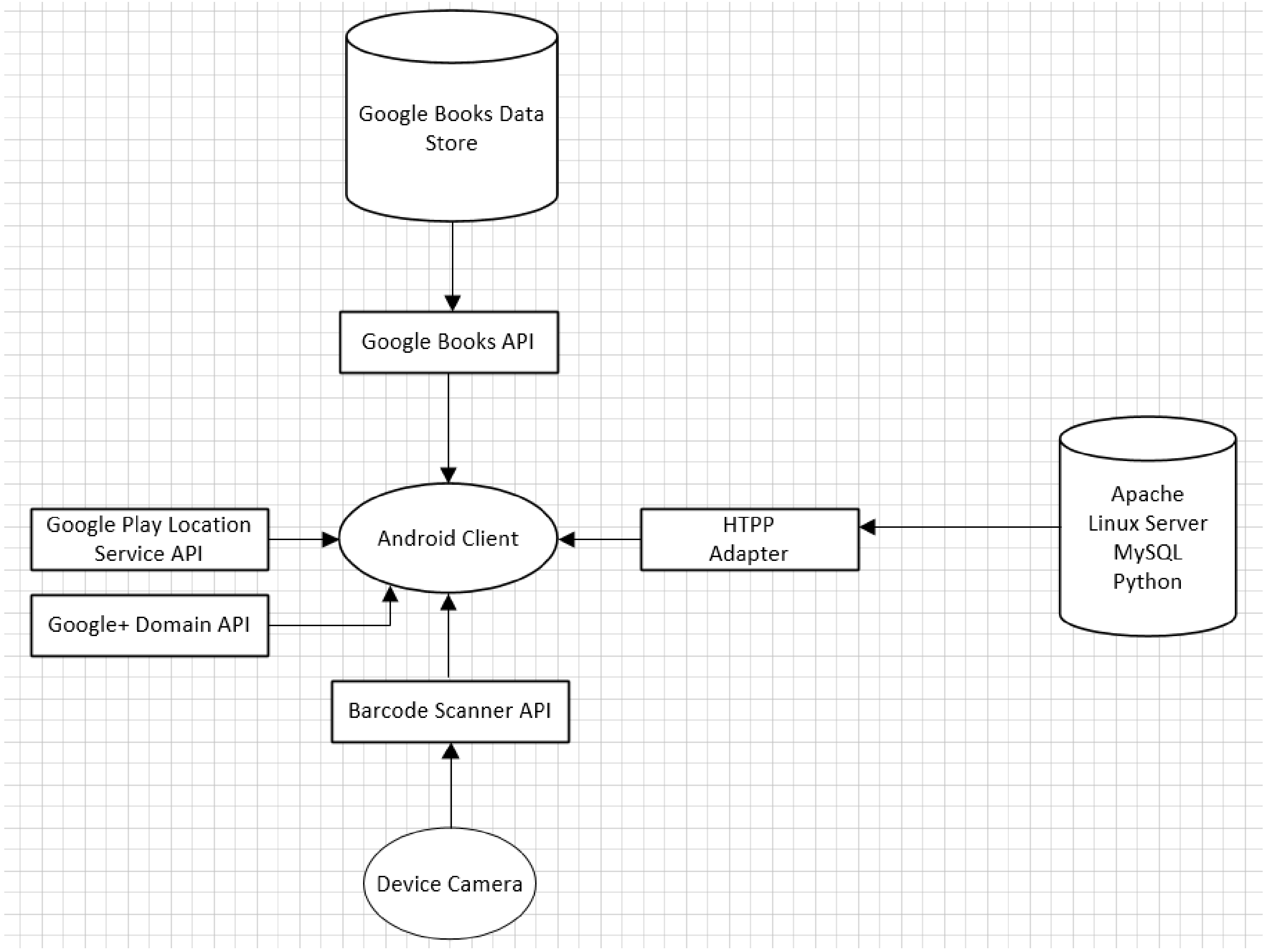


Figure 2.1 System Architecture

2.3 Android Client

The service that our android application provides, is very similar what Craigslist provides. However, our application is mobile and should provide 5-star user experience. Thus, a function that allows user to populate book information should as easy as possible. The solution is to use ISBN as a key to query public book library to retrieve book information. In this approach, we will need to use Barcode Scanner library and Google Books API.

Barcode Scanner is an open source project that inherit from other open source projects to utilize Android camera to scan and interpret barcode. We will show the users an option to scan ISBN barcode, and directly retrieve ISBN. The interpreted ISBN will then be passed into the data entry activity along with Google Books API it will auto-populate all the relevant information about the book, which will be discussed in the next section.

2.4 Google Endpoint Services

We will use three Google endpoint services: Google Books, Google Play Location Service, and OAuth with Google+. We may also use Google Map to show all sales in the map.

Google Books API powered by Google, will provide a nice and simple programming interface to the android client to query book information based on ISBN or other book attributes such as author and title. For our Android client, the API will only be used to query book data by ISBN.

One of the unique features of mobile applications is location awareness. Mobile users take their devices with them everywhere, and adding location awareness to your app offers users a more contextual experience [6]. The application will provide a function to search all nearby sales. To do that we will need to get and store user location data whenever they post a new book sale. Google Play Location Service will be used in our client side to fetch device's location data. It is preferred over Android framework location APIs as stated on Google Android Developer's website, but we will try to embed Android framework location APIs into the application as an alternative.

Since the application doesn't require precise location data. The fused location provider in Google's service provides the device's last known location. The fused location provider is one of the location APIs in Google Play services. It manages the underlying location technology and provides a simple API so that you can specify requirements at a high level, like high accuracy or low power. It also optimizes the device's use of battery power [7]. We may also use Google Map Service for Android to show all nearby sales by location just like how Yelp shows nearby restaurants.

2.5 Server and Data

As mentioned previously we will use MySQL for server database, configured with Apache and Linux server. For this project, MySQL is enough to handle all the data and queries we need. Both local and server database are needed to deliver 5-start user experience. For client data storage, we will use SQLite Database and store the database file in user's internal storage directory. SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine [8]. SQLite is a popular choice as embedded database because of its lightweight and easy-configuration.

One of the difficult parts of development of this system is the synchronization between the data in the local database and the server database. A check on the data and database schema is needed prior to publishing stored sale data. We are also purposing an off-line publishing function, that the data user populated in the form while off-line will be published online when their device is connected to internet.

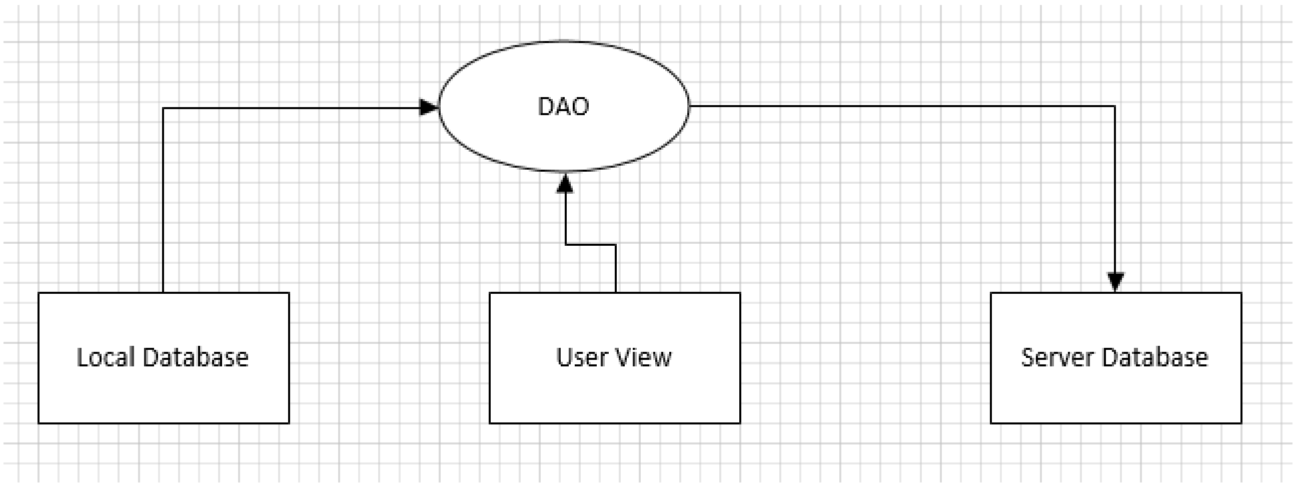
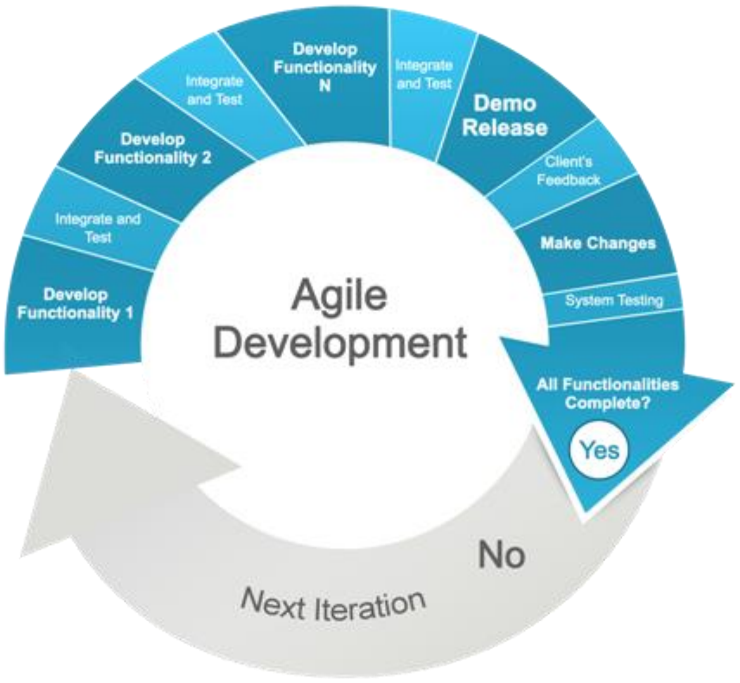
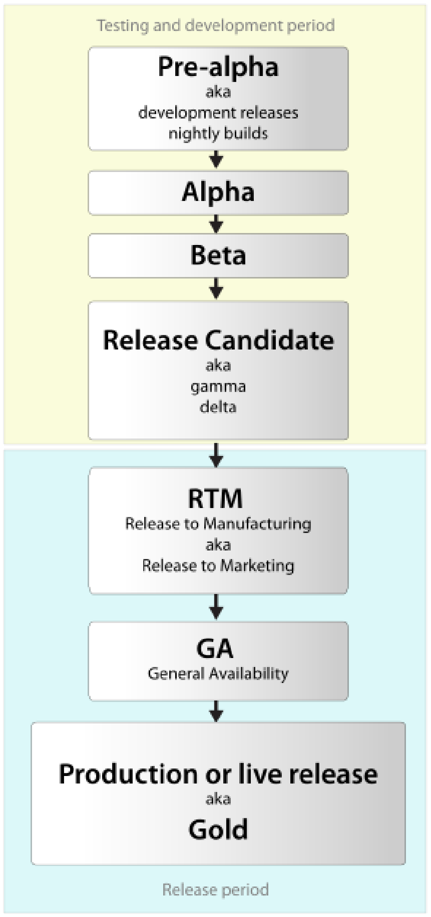


Figure 2.2 Data Access and Synchronization

2.6 Agile Methodology

Unlike other software that have clients to provide requirements and periodic feedback during the development process, software such Android application that will be published to the market for general public will not get reliable feedbacks until it is published, and for this projects the requirements are solely based on the idea and product requirements that developers created. Agile methodology is still a good choice this project, because the size of the project is not too big that does not require much effort into requirement analysis, and also because the application uses many third-party APIs and libraries it will be reasonable to use “horizontal” development process, in other words, went through the entire development process at first, and built more material on the foundation of the first iteration.



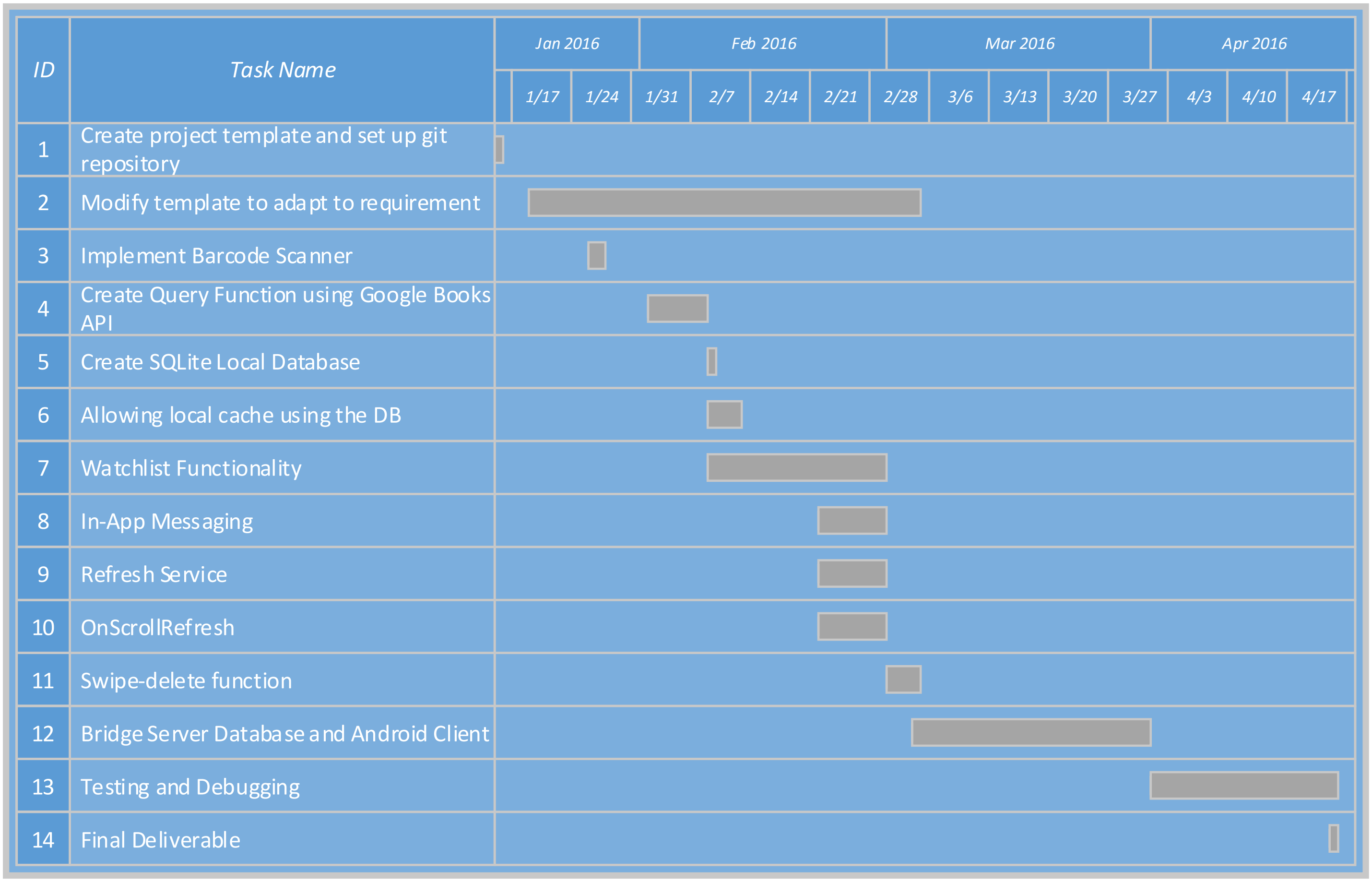


[9]Figure 2.3 Agile Development

[10]Figure 2.4 Software Release Life Cycle

2.7 Gantt Chart

The entire project is broken down to 14 different tasks, from the creation of the project template and Git repository to the final deliverable. The following Gantt chart shows estimated tasks timeline:



ID	Task Name
1	Create project template and set up git repository
2	Modify template to adapt to requirement
3	Implement Barcode Scanner
4	Create Query Function using Google Books API
5	Create SQLite Local Database
6	Allowing local cache using the DB
7	Watchlist Functionality
8	In-App Messaging
9	Refresh Service
10	OnScrollRefresh
11	Swipe-delete function
12	Bridge Server Database and Android Client
13	Testing and Debugging
14	Final Deliverable

Figure 2.5 Gantt chart

Chapter 3

Design and Testing

3.1 Introduction

In previous chapter, we talked about the system architecture and modeling. In this chapter we will talk about the design, implementation of the system, and software testing.

There are many existing applications in the market that already have fair amount of users. Our application is essentially competing with them to get the same group of clients. In order to succeed the competition, we have these ways in terms of product design to attract their or new users:

1. Provide better quality of service including easy use of the system and improvement of system performance in different devices.
2. Provide extra functionalities that will provide benefits to the users. For example, integration with social media, such as Facebook.
3. Supporting wider range of device and possibly platforms.

3.2 Design

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints [11]. The client application has a main panel with four tabs, one for the home section where you have action choices, to buy, to sell or show all nearby sales, one for watch list that lists all stared and your own sales, one for inbox that lists all messages from other users, the a setting tab for application settings.

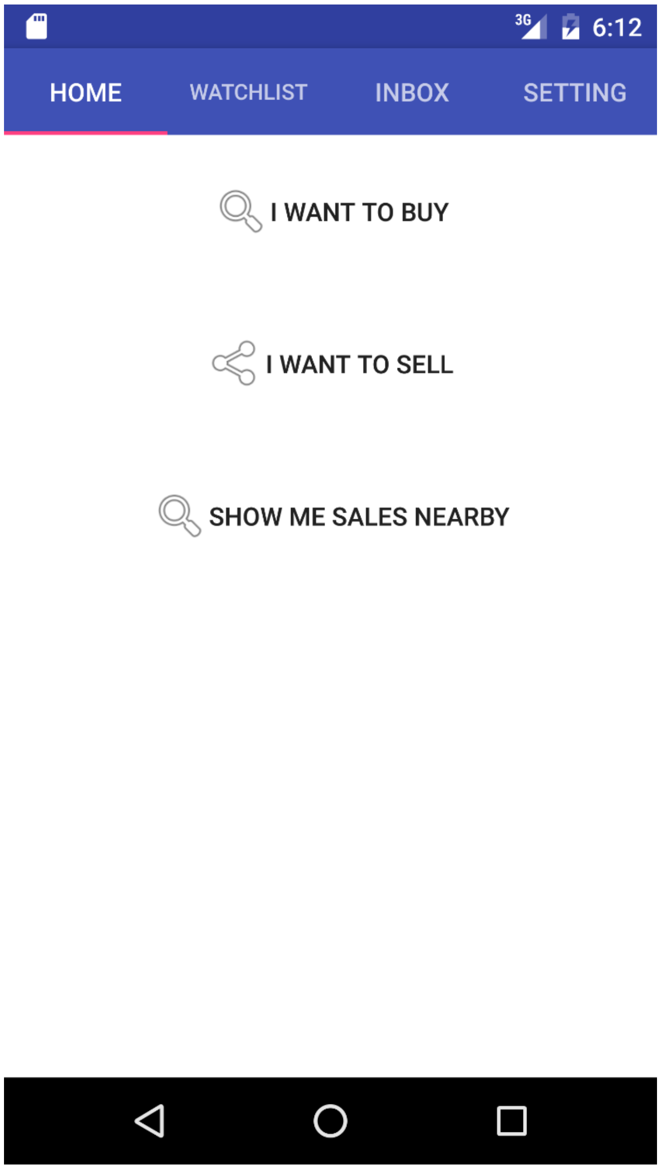
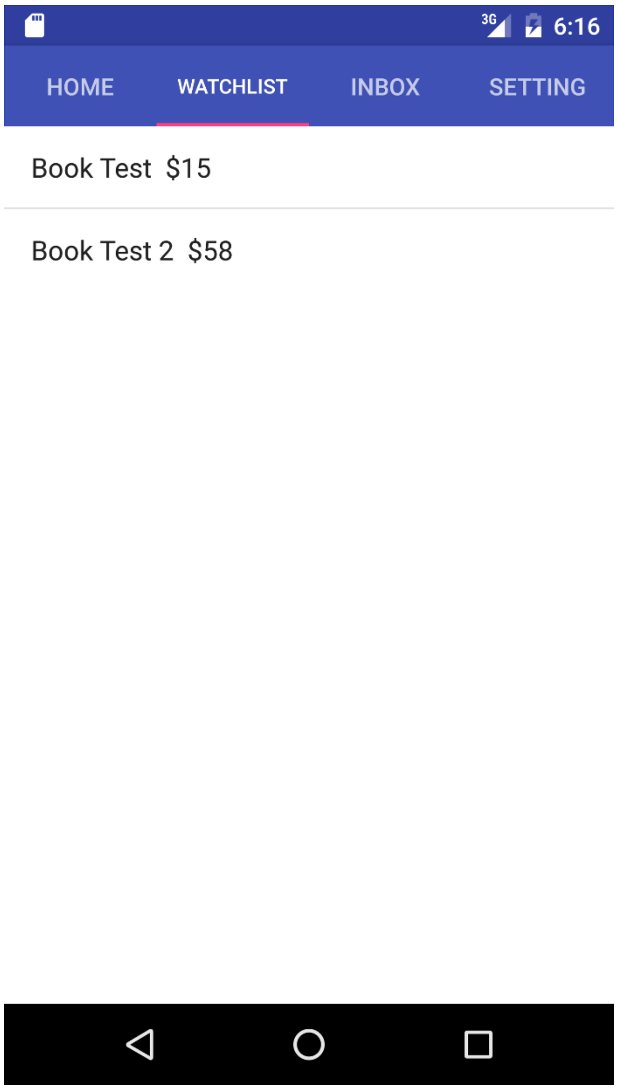


Figure 3.1 Application Main Panel Home



Book Test $15

Book Test 2 $58

Figure 3.2 Application Watch list

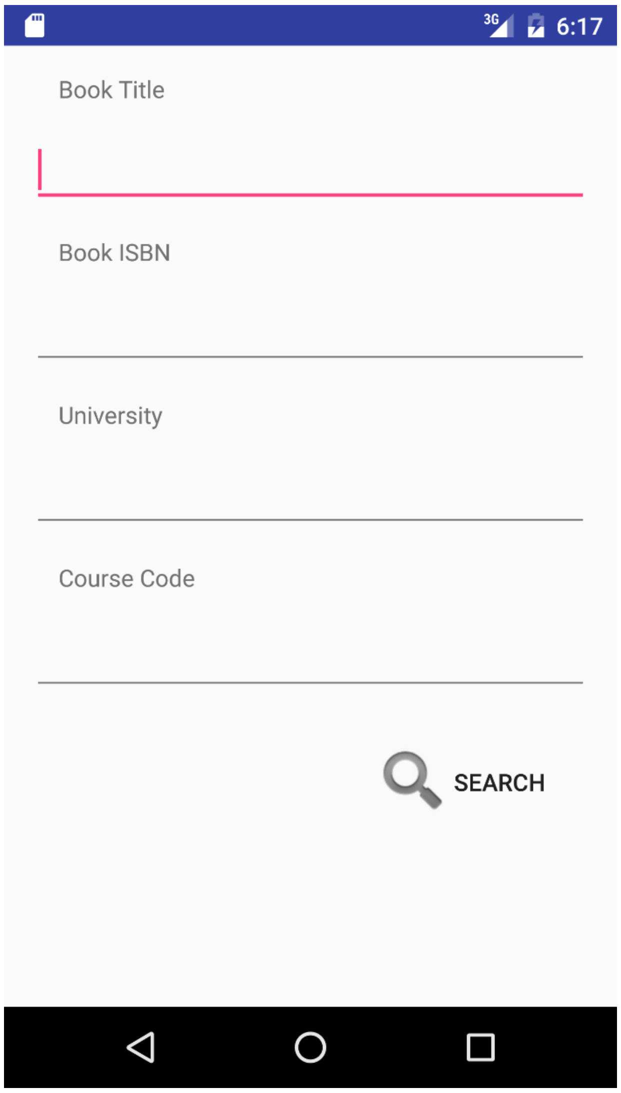


Figure 3.3 Application Sale Search

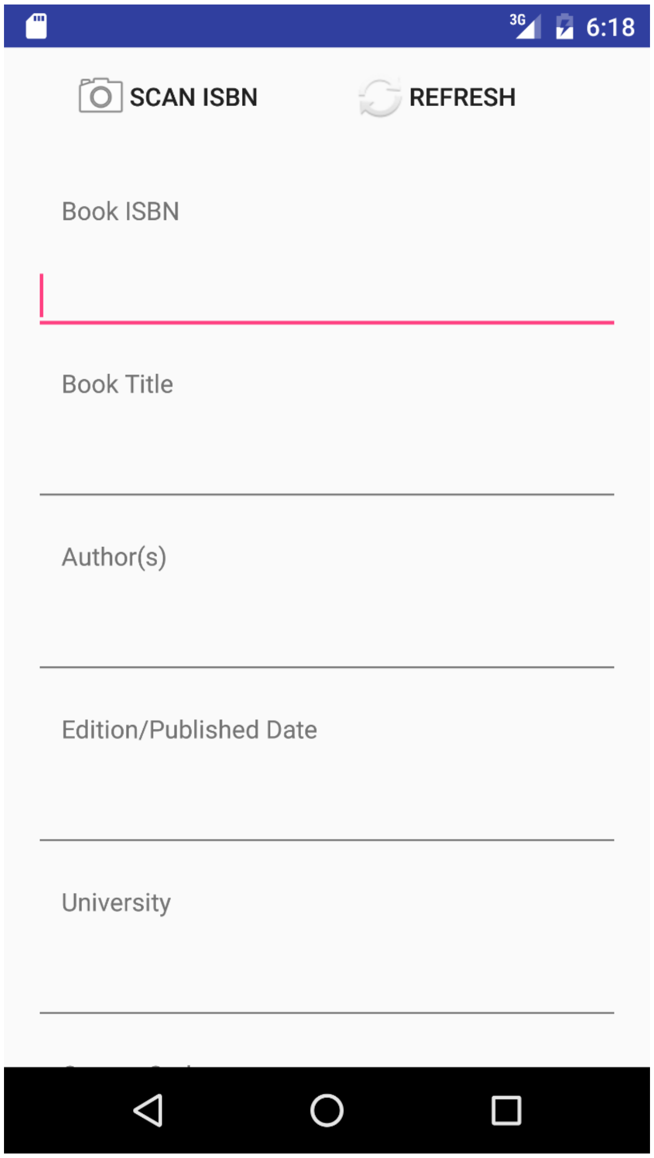
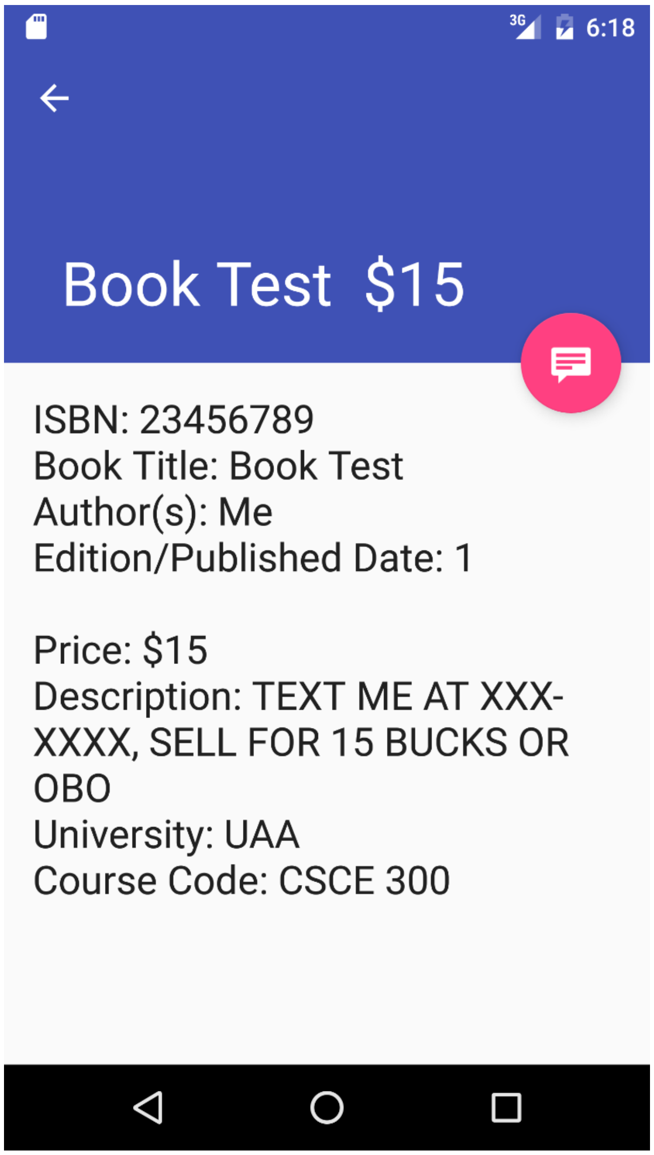


Figure 3.4 Application Publishing Sale Data Entry



Book Test $15

ISBN: 23456789
Book Title: Book Test
Author(s): Me
Edition/Published Date: 1

Price: $15
Description: TEXT ME AT XXX-XXXX, SELL FOR 15 BUCKS OR OBO
University: UAA
Course Code: CSCE 300

Figure 3.5 Application Sale Details

3.3 Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test [12]. Software testing is so important that it will literally save you lots of money but eliminating bugs. A study conducted by NIST in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually [13].

In application testing, there are two types of testing. The first one is Black-box testing. Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings [14]. The second one is White-box testing. White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality [15]. In our product, the testing phase will be broken into three segments:

1. Test of the corrections of the functions and modules within the system, also known as White-box testing.
2. Test of the application functionalities and GUI from user perspective, also known as Black-box testing.
3. User acceptance test, is quite special to our product since we do not get feedback from clients during development and testing phase, but we will rely on user feedbacks from our beta release. Since our application is not so complicated, we are expecting the beta and stable release will not have much difference.

Chapter 4

User's Manual

4.1 Product Introduction

This is an android application serves as a platform to allow college students to sell their used textbooks online. The application user interface is easy to navigate, configure, and use. The application gives you the functionality to retrieve book data from Google Books API by ISBN that is auto-populated by scanning your book's ISBN barcode, which saves students lots of time considering how much time they would have spent on entering all the information about their textbook. More functionalities of the application will be introduced in details in the next few sections of this manual.

4.2 Installation

You can install the android textbook app from Google Play when it is published or you may visit developer's website to download it manually and install it. It is recommended that you download it from Google Play as it is much safer. When downloading it will ask your machine to give few access permissions, such as internet and hard disk, simply allow it as the application will need these accesses to function properly.

4.3 Use and Navigation

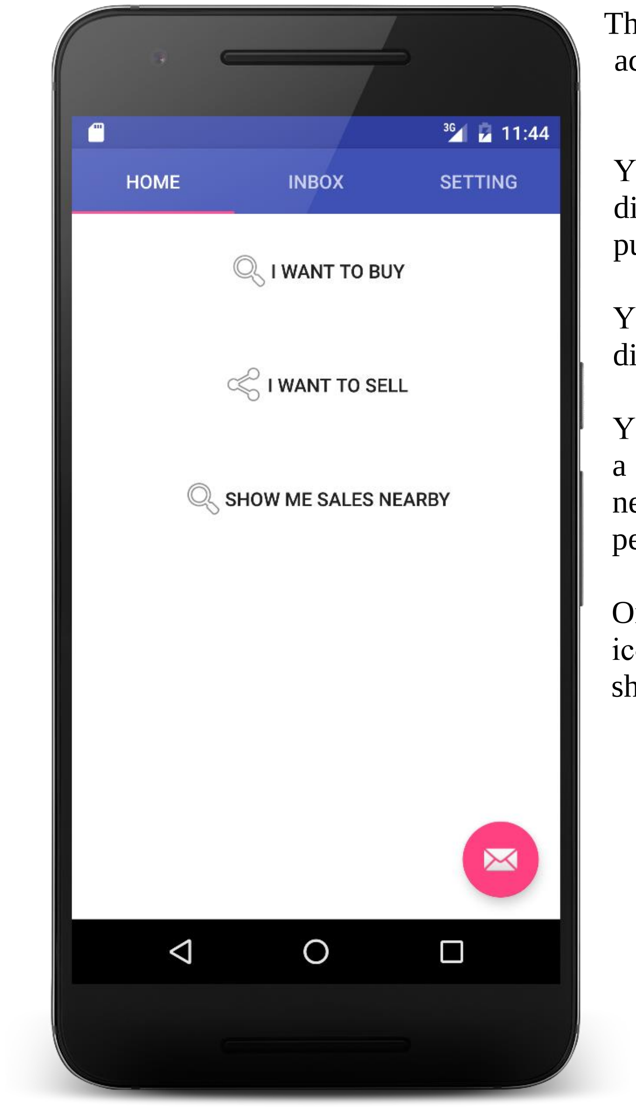


Figure 4.1 Application Home Page

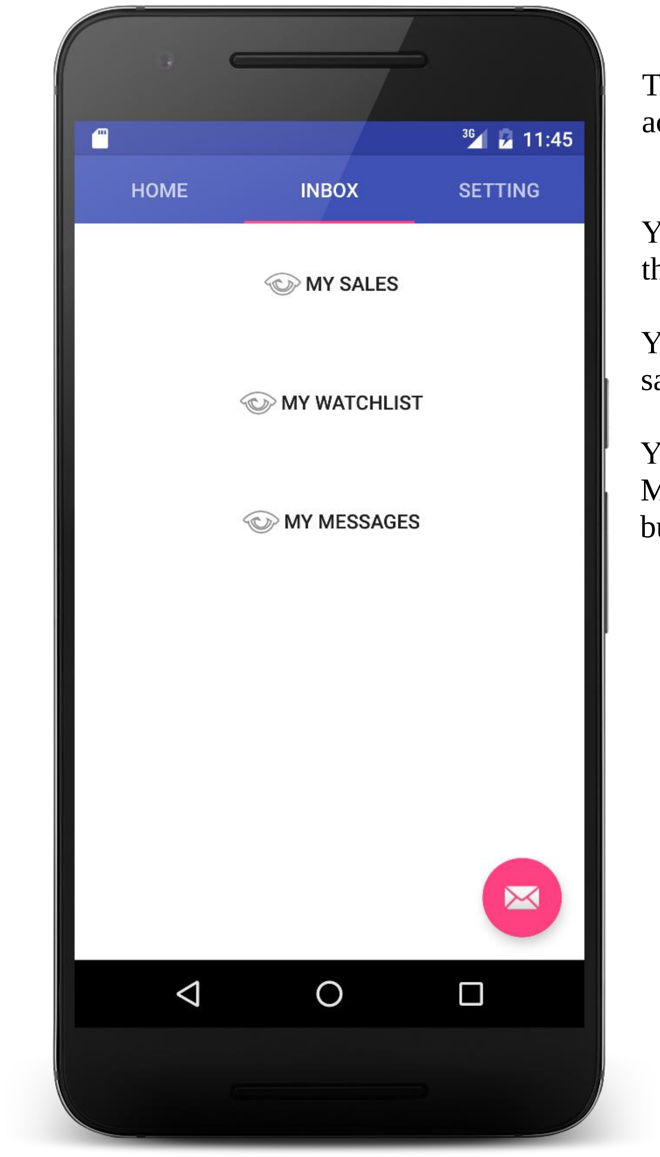
The home page allows you to navigate to different activities.

You can click on ‘I want to sell’ to navigate to a different page where you can enter book sale data and publish for sale.

You can click on ‘I want to buy’ to navigate to a different page where you can search for book sales.

You can click on ‘Show me sales nearby’ to navigate to a different page where you can find all book sales nearby your location. This function will need access permission to your device geolocation data.

On the right-lower corner of the home page, the ‘Email icon’ allow you access your ‘Message Inbox’ in shortcut.



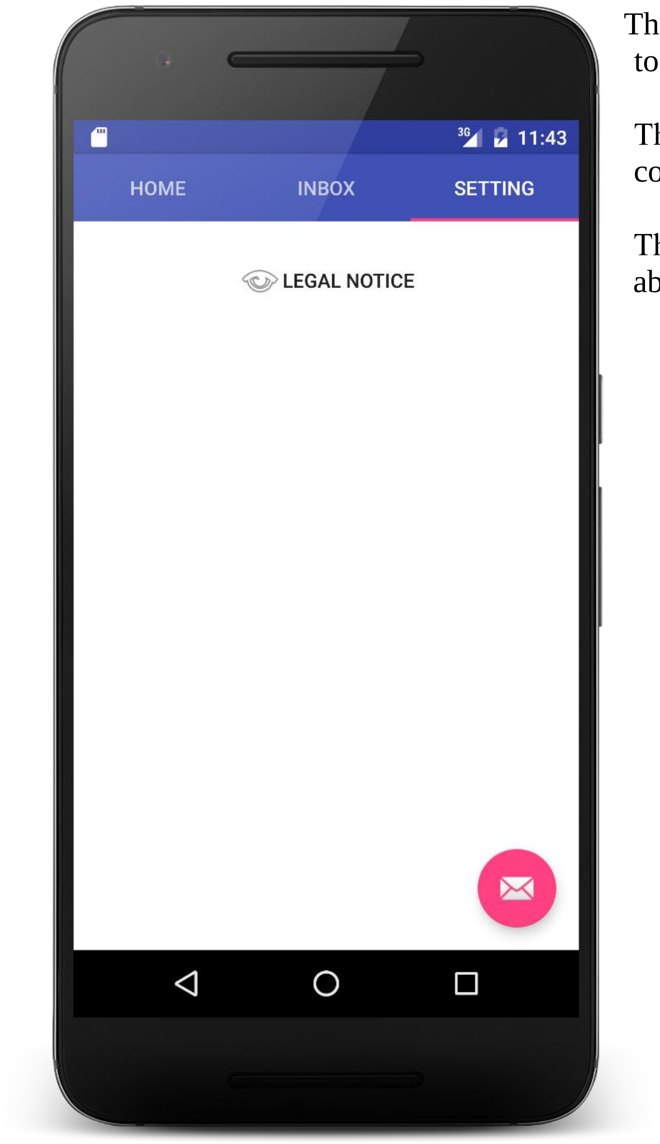
The inbox page allows you to navigate to different activities.

You can click on ‘My sales’ to bring up a list of sales that you published.

You can click on ‘My watchlist’ to bring up a list of sales that you starred (saved).

You can click on ‘My messages’ to access your Message Inbox, to see your messages from sellers and buyer about a particular sale.

Figure 4.2 Application Inbox Page

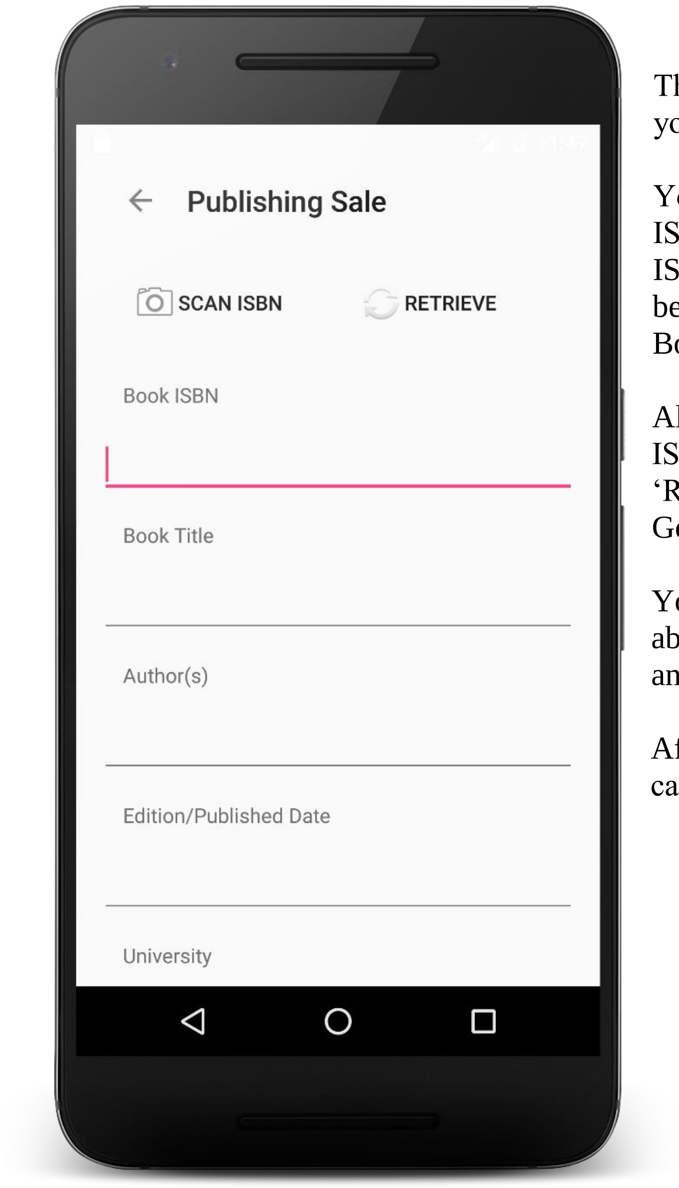


The setting page allows you to control some variables to customize the application appearance and behavior.

The ‘Privacy and Data’ section explains how data collected by this application are used.

The ‘Legal notice’ section explains legal information about the application in details.

Figure 4.3 Application Setting Page



This is 'Publishing Sale' page that shows up after you click 'I want to sell'.

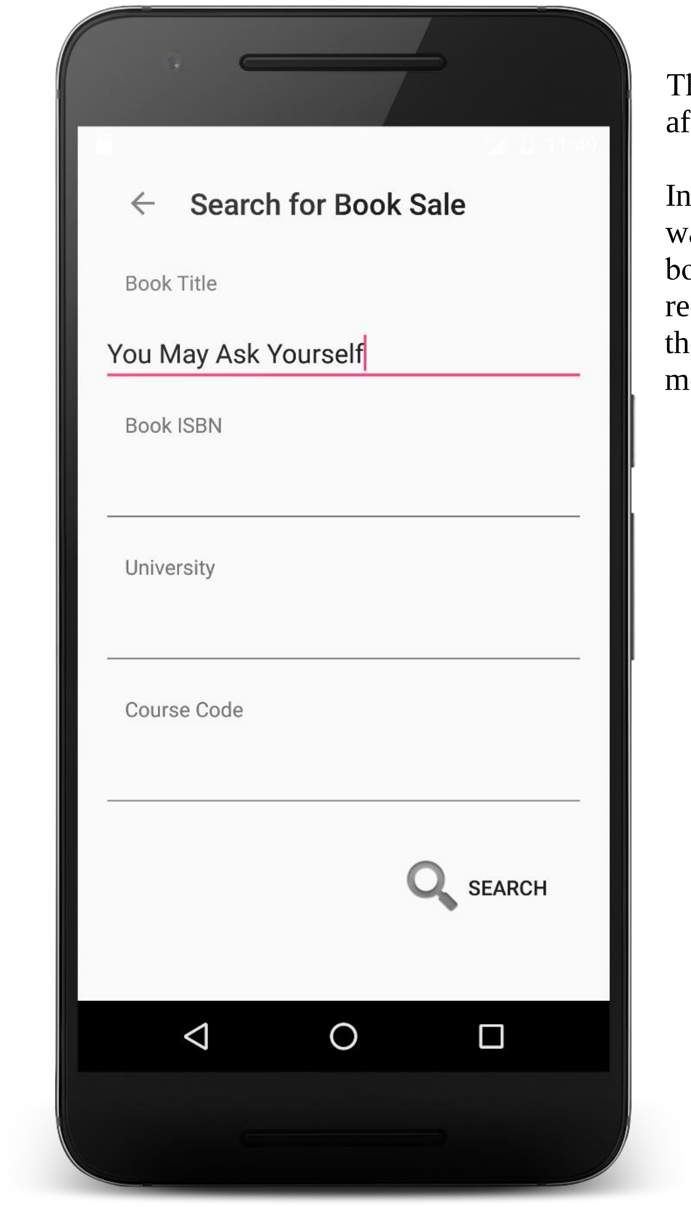
You can click on 'Scan ISBN' to scan your book ISBN bar code by using your device's camera. After ISBN is retrieved, the rest of book information will be automatically populated by querying Google Books API using the ISBN.

Alternatively, if you were not able to use 'Scan ISBN', you can enter ISBN manually and click 'Retrieve', and the application will try to query Google Books with the ISBN you entered by hand.

You can then continue to populate other information about the sale, such as price, university, course code, and sale description.

After you have entered all the sale information, you can then click 'Publish' to publish the sale.

Figure 4.4 Application Publishing Sale


Search for Book Sale

Book Title

You May Ask Yourself

Book ISBN

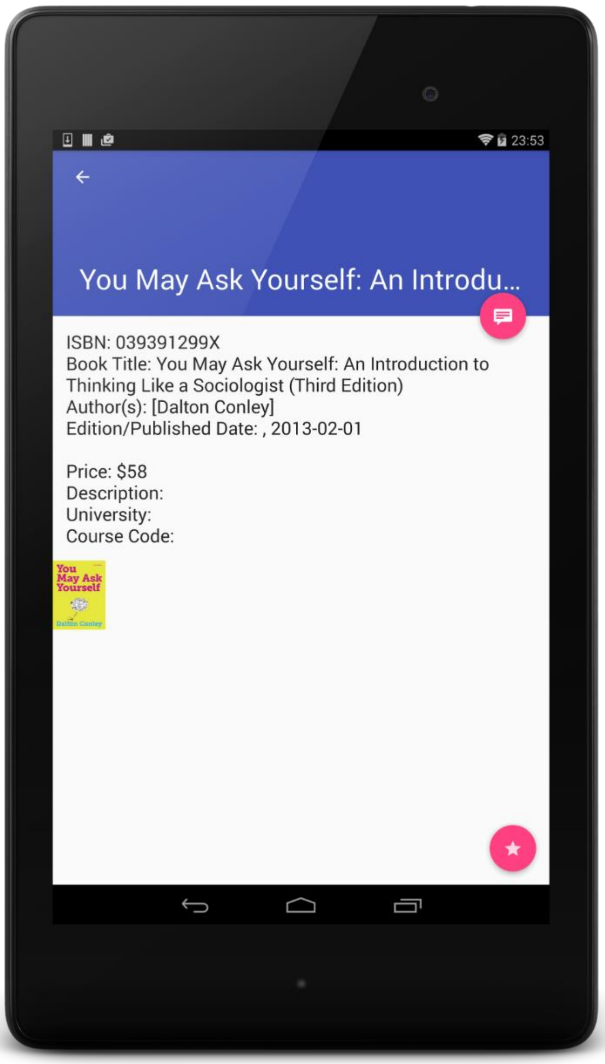
University

Course Code

This is ‘Search for Book Sale’ page that shows up after you click ‘I want to buy’.

In this page, you entered details about the book you want to buy. Details entered will be used to search book sales when you click ‘Search’. It is recommended that you entered 1 – 2 items about the book and search, this way you will likely get more search results.

Figure 4.5 Application Search for Sale



23:53

You May Ask Yourself: An Introdu...

ISBN: 039391299X
Book Title: You May Ask Yourself: An Introduction to Thinking Like a Sociologist (Third Edition)
Author(s): [Dalton Conley]
Edition/Published Date: , 2013-02-01

Price: $58
Description:
University:
Course Code:

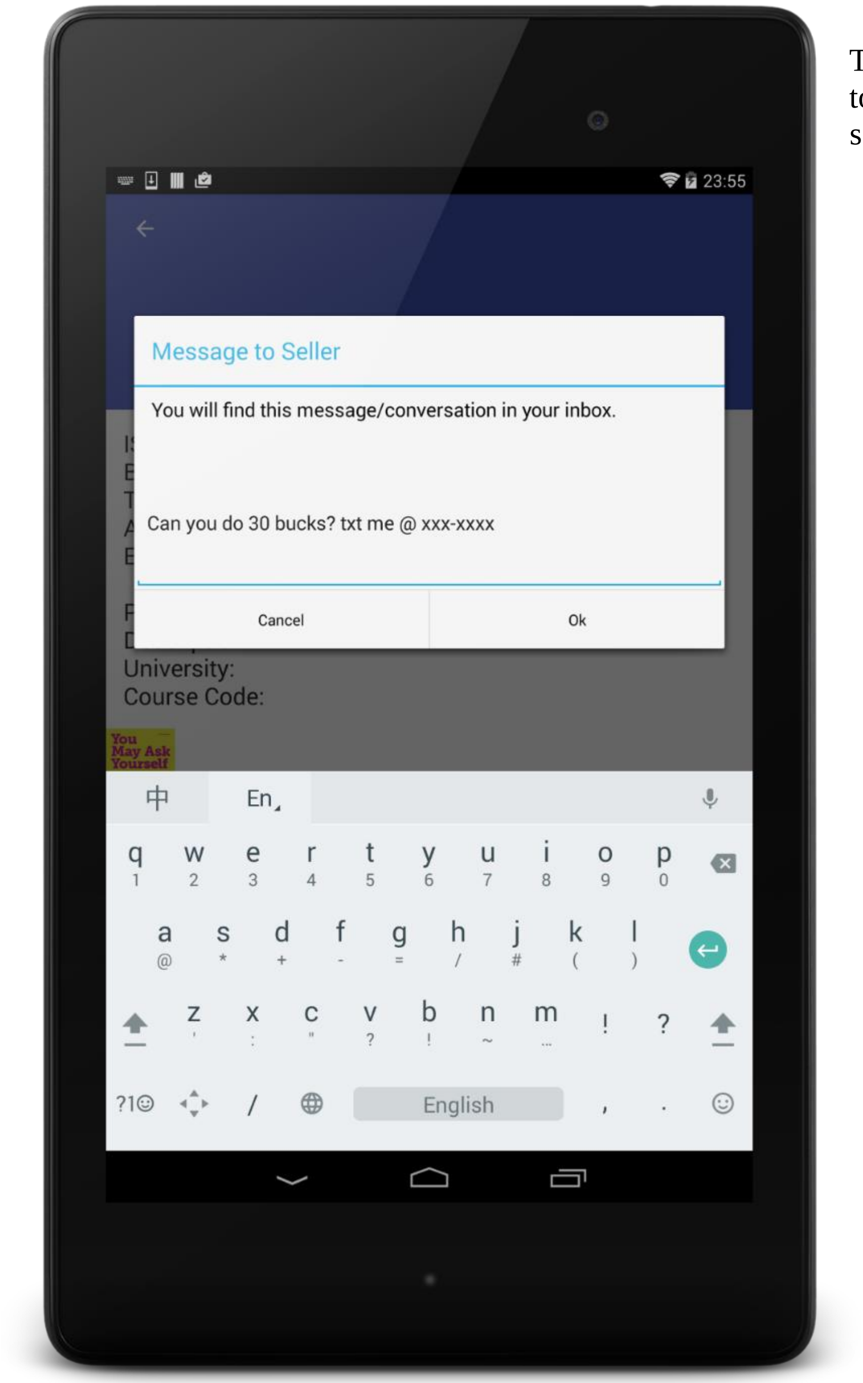
Figure 4.6 Application Sale Details

This is Book Sale details page that shows up after you click on one of the sales you found on the application.

This page shows details about the sale.

You can click on the ‘Conversion icon’ located on the right-upper corner of the screen, to send a message to the seller. You can find your conversion (messages) in your ‘My messages’ of your ‘Inbox’ page.

You can click on the ‘Star icon’ located on the right-lower corner of the screen, to star (save) the sale. You can find starred sales in your ‘My watchlist’ of your ‘Inbox’ page.



Message to Seller

You will find this message/conversation in your inbox.

Can you do 30 bucks? txt me @ xxx-xxxx

This is an example of how you want to send the message to the book seller.

Figure 4.7 Application Sending Seller Message

4.4 Services and Permissions

The application will ask to access your camera for scanning ISBN, your local device disk to store database file, your device geolocation data to enable ‘search nearby’ function, your Google+ profile to authenticate service, your device network to connect to the internet.

The application uses three Google endpoint services: Google Books, Google Play Location Service, and OAuth with Google+.

Google Books API powered by Google, will provide a nice and simple programming interface to the android client to query book information based on ISBN or other book attributes such as author and title. For our Android client, the API will only be used to query book data by ISBN.

The application will provide a function to search all nearby sales. Google Play Location Service will be used to fetch device's location data. Since the application doesn't require precise location data. The fused location provider in Google's service provides the device's last known location. The fused location provider is one of the location APIs in Google Play services. It manages the underlying location technology and provides a simple API so that you can specify requirements at a high level, like high accuracy or low power.

4.5 Data and Privacy

The data collected by the application are solely used for textbook sales between users. We do not use your data for any other purpose, other than providing textbook sale information on the application.

4.6 Help

Contact application developer for additional help. You can find developer’s contact information of their website and Google Play application page.

Chapter 5

Conclusion

5.1 Project Result

At the end of the project, we completed most of the functions we planned both on the server and client side. The entire project lasted 12 weeks but we still have 2 weeks to do some testing and debugging for the final presentation on April 25th.

The growing mobile market and the convenience of mobile application makes the platform perfect for developing some agile and useful applications. We as students understand how expensive college education is so decided to develop an Android application that allow students to trade their used textbooks thus save some money for them. When this application is published to the market place, hundreds of thousands of students will be able to download and use it to their convenience and benefit, and it likely will save them a fair amount of money and time.

5.2 Future Development

There are two aspects in regards to future development of this project, one is features, and another is code base.

The features of this application needs to be polished and extended:

1. Swipe View List used in My Sale and Watch List Activity can be polished that the icon of the item can be the thumbnail image of the book.
2. On the Main Activity the user should be allowed to sign out and switch to different account.
3. On Setting panel there should be option to change application permission access.
4. The view of Message Activity can be enhanced.
5. Search for textbook sales (“I want to buy” option) function can be more intuitive and user-friendly

6. An introduction animation can be included.
7. Integration with social media function.

The code base and the performance of the application can be optimized:

1. Many unreferenced classes and library can be eliminated.
2. Some custom libraries can classes can be simplified
3. Better handling of tasks and services
4. Better handling of threads and exceptions, which includes more clear separation of UI threads and non-UI threads to improve user experience.
5. Better handling of debugging and testing
6. The server REST API needs to be able to handle image URL link in the request or we may to develop Android API instead of REST.

Short Presentation Video: https://www.youtube.com/watch?v=DpWqz-uhsrY
Android client GitHub link: https://github.com/Aaron-Zhao/Textbookbns

Figure 5.1 Android Bot

Figure 5.2 GitHub Octocat

5.3 Conclusion

This project has been a great experience for both me and my project coworker. I touched many areas in Android development where I was not exposed to before. I am glad that I made my decision to make this Android application as it practiced lots of my programming skills in the course of the development. I also learned many new things on the server side from my project coworker as he is the person in charge of the server side of this application. I honestly believe this Android application can make huge contribution to the community of college students once it is published to the market and people started using it. The future development of this project can also be a help to the community.

References

[1] Heidi Cohen (2015). Mobile Apps In 2015: Not Just Social Media Anymore. Retrieved from: http://heidicohen.com/mobile-app-trends-2015/

[2] Biz Carson (2015). What two 100-year-old companies — IBM and Wells Fargo — have learned about staying relevant. Retrieved from: http://www.businessinsider.com/ibm-ceo-ginni-rometty-wells-fargo-ceo-john-stumpf-you-cant-protect-the-past-2015-11

[3] Patrick Thibodeau (2014). The top 10 paying IT jobs: IT careers can prove fruitful. Retrieved from: http://www.computerworld.com/article/2489809/it-careers/the-top-10-paying-it-jobs--it-careers-can-prove-fruitful.html

[4] Wiki (2016). Agile software development. Retrieved from: https://en.wikipedia.org/wiki/Agile_software_development

[5] Ben Popken (2015). College Textbook Prices Have Risen 1,041 Percent Since 1977. Retrieved from: http://www.nbcnews.com/feature/freshman-year/college-textbook-prices-have-risen-812-percent-1978-n399926

[6] Google Android Developer (2016). Making Your App Location-Aware. Retrieved from: http://developer.android.com/training/location/index.html

[7] Google Android Developer (2016). Getting the Last Known Location. Retrieved from: http://developer.android.com/training/location/retrieve-current.html

[8] SQLite Org (2016). SQLite Home Page. Retrieved from: https://www.sqlite.org/

[9] Strategybeach (2016). Agile Development Methodology. Retrieved from: http://strategybeach.com/our-agile-development-methodology/

[10] Wikipedia (2016). Software release life cycle. Retrieved from: https://en.wikipedia.org/wiki/Software_release_life_cycle

[11] Wiki (2016). Software design. Retrieved from: https://en.wikipedia.org/wiki/Software_design

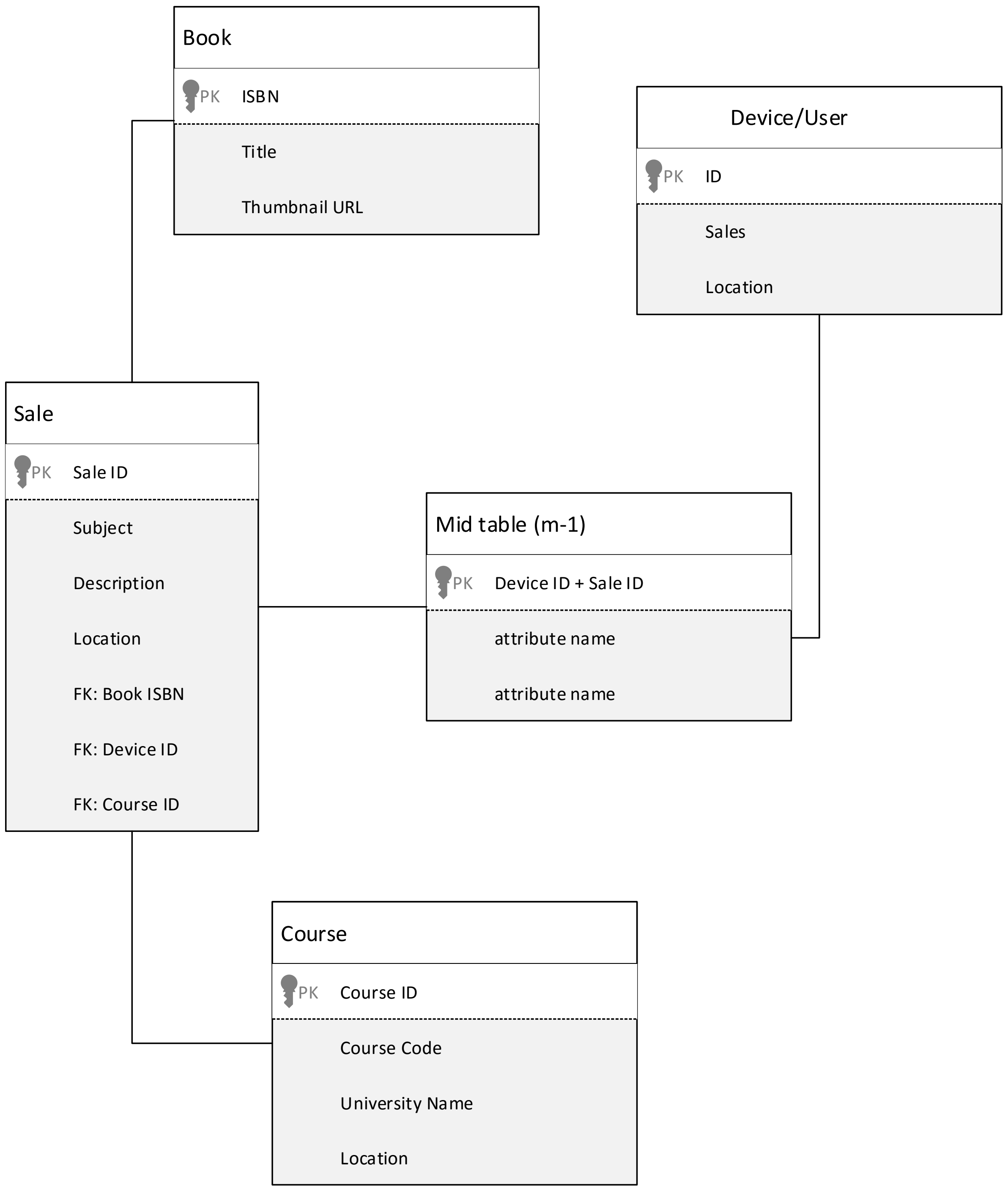
[12] Wiki (2016). Software Testing. Retrieved from: https://en.wikipedia.org/wiki/Software_testing

[13] NIST Report (2016). The economic impacts of inadequate infrastructure for software testing. Retrieved from: http://www.nist.gov/director/planning/upload/report02-3.pdf

[14] Wiki (2016). Black-box testing. Retrieved from: https://en.wikipedia.org/wiki/Black-box_testing

[15] Wiki (2016). White-box testing. Retrieved from: https://en.wikipedia.org/wiki/White-box_testing

Appendix A: UML Diagram



Appendix B: Initial Source Code

See Details: https://github.com/Aaron-Zhao/Textbookbns

```
/**
 * An activity representing a list of SaleItems. This activity
 * has different presentations for handset and tablet-size devices. On
 * handsets, the activity presents a list of items, which when touched,
 * lead to a {@link SaleItemDetailActivity} representing
 * item details. On tablets, the activity presents the list of items and
 * item details side-by-side using two vertical panes.
 * <p/>
 * The activity makes heavy use of fragments. The list of items is a
 * {@link SaleItemListFragment} and the item details
 * (if present) is a {@link SaleItemDetailFragment}.
 * <p/>
 * This activity also implements the required
 * {@link SaleItemListFragment.Callbacks} interface
 * to listen for item selections.
 */
public class SaleItemListActivity extends AppCompatActivity
        implements SaleItemListFragment.Callbacks {


    /**
     * Whether or not the activity is in two-pane mode, i.e. running on a tablet
     * device.
     * Customized to always false
     */
    private static final boolean TWO_PANEL_MODE = false;


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


        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        toolbar.setTitle(getTitle());


        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
```

```
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
```