**Software Design Document**

**Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu**

***Libraworks’ Virtual Library***

**10/30/2020 – 11/10/2020**

**Table of Contents:**

1. ***Purpose & Scope***
2. ***Definitions, Abbreviations and Acronyms***
3. ***Software and System Architectural Design***
4. ***System Components/Interfaces***
5. ***Usability Interfaces***
6. ***Requirement Traceability Matrix***

**Revisions:**

|  |  |  |
| --- | --- | --- |
| *Names* | *Details* | *Revision No.* |
| Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu | Began SDD | 0.1 |
| Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu | Continued SDD, restructured sections, added system components and updated diagram of high-level architecture, as well as created system overview diagram. | 0.2 |
| Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu | Inserted UI Screenshots | 0.3 |
| Jiayi Xu | Inserted traceability matrix | 0.4 |
| Grant Saylor, Kyle Smith, Anthony Tran, Jiayi Xu | Finished SDD for Sprint 3 | 1.0 |

***Introduction:***

1. **Purpose & Scope:**

This software design document will cover the overall system design of Libraworks’ Virtual Library, a library cataloguing and circulation creation system that will interface with existing Little Free Libraries, offering ease of access to books and libraries. By using Virtual Library there is the benefit of time saving by seeing nearby libraries on a map and only going to a Little Library when a book the user wants is available, it also provides a way for the Steward to easily catalog what is at their library at any given time. Within the software design document (SDD) an overview of the app architecture, data management, hardware requirements and definitions/acronyms. After reading this document there will be a clear goal of the project implementation.

1. **Definitions, Abbreviations and Acronyms:**

***SDD*** *– Software Design Document*

***REQ*** *– Requirements*

***LVL*** *– Libraworks Virtual Library*

***LL*** *– Little Library*

***OL*** *– Open Library*

***DB*** *– Database*

***API*** *– Application Programming Interface*

***XML*** *– Extensible Markup Language*

***JSON*** *– JavaScript Object Notation*

***Libraworks*** *– The software development team*

***Virtual Library*** *– The application*

***Little Library*** *– User established neighborhood libraries*

***Steward*** *– User who manages a Little Library*

***Internet Archives OpenLibrary*** *– The book API*

***Firebase –*** *Google’s consumer product database management system*

***OpenLibrary –*** *Internet Archive’s open source book information API*

1. **Software and System Architectural Design:**

Our client application will be using Kotlin with XML and JSON interfaces for the API calls. The client will communicate with both the Google Maps API as well as OpenLibrary API for functionality with our map activity and for book information respectively. The results of these API calls will be stored in a database hosted by Google Firebase, which will communicate with our client app to display items in the circulation and search activities.

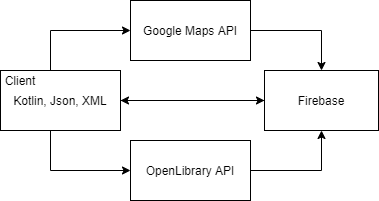


Figure 1.0

The architecture for the LVL system is unknown due to its design and source code unavailability. *(See figure 1.1 below for a larger overview)*

**References:**

Prior to reading this SDD, please reference the following:

|  |  |  |
| --- | --- | --- |
| *References* | *Location* | *Date* |
| Software Requirement Specification | https://bit.ly/3ea8X5q | 10/28/2020 |

1. **System Components:**

The client is written in Kotlin and XML, utilizing the Google Maps API and OpenLibrary API to display maps and book information, respectively. These in turn interact with our Firebase database to perform check-in/check-outs. The following diagram graphically demonstrates this:

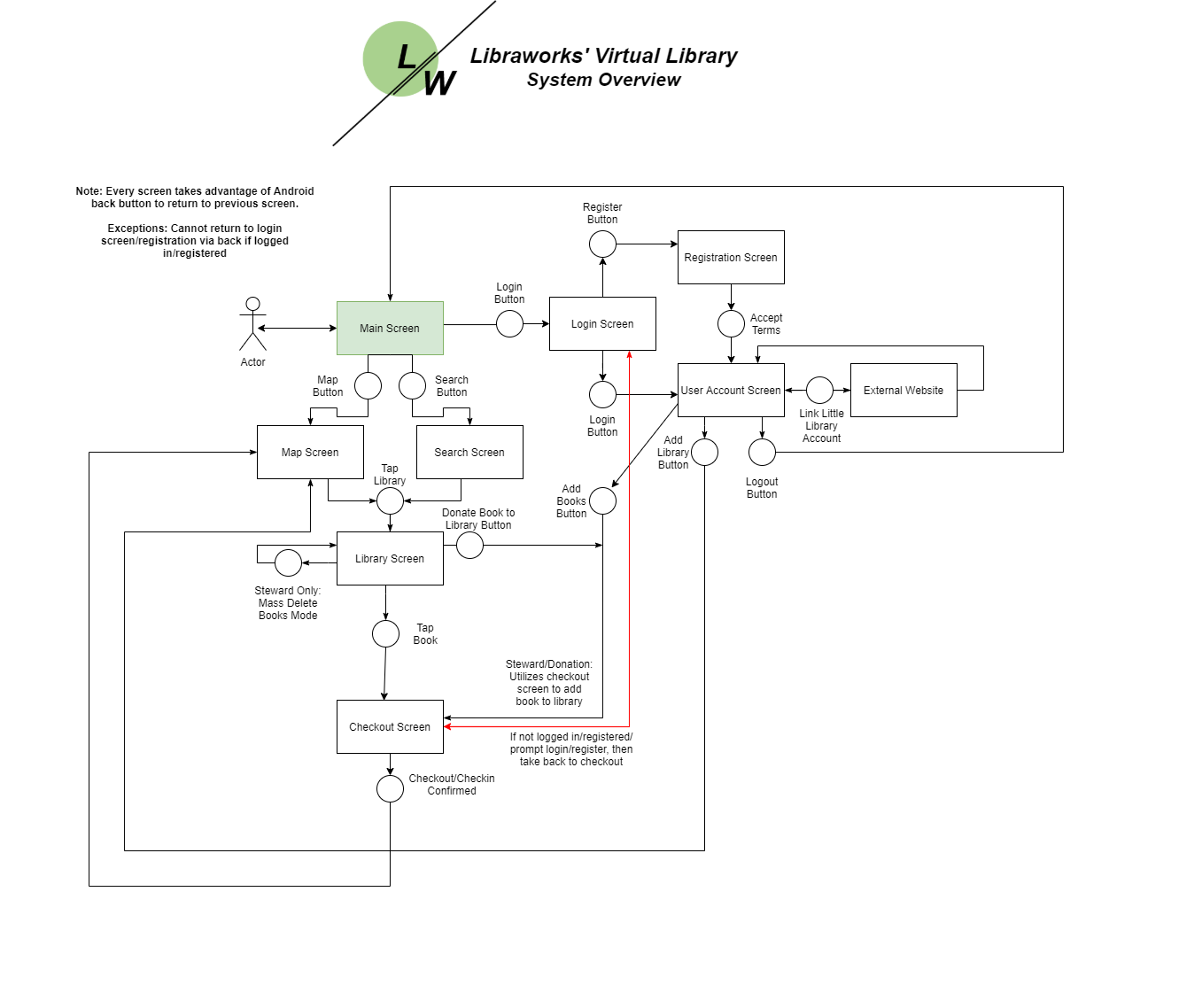


Figure 1.1

* 1. **Circulation Activity:**

This activity is what the user will see when they select a LL from the map screen or from the search screen. It will display the list of books available at that LL location. This activity will also link users to the Check In/Out activity page. The information displayed on this page comes directly from the Firebase DB, which is keeping track of the books associated with that LL, and which ones are currently available. There will be an additional button which will alter the layout of his page. The new layout will be more compact and allow the user to rapidly remove (checkout) books from that LL, or even to donate (add new book) to the LL, which will then update the DB.

* 1. **Map Activity:**

This activity displays nearby LL locations to the user. The Libraworks developers are undecided at this time as to how to define ‘nearby’ for users. Users will be able to select any of the LL locations visible on the map, and that will pop up a context menu with information such as the address and name of the LL, how many books are available currently, and a button to take the user to that LL circulation activity.

* 1. **Search Activity:**

The search activity will allow users to type in information about a book which will query the Firebase DB for which nearby LL have that book in stock. Because it is only interfacing with the DB and not the book API, it will only populate results if the book is found within our system. If the book has not been checked in at any LL location, the search result will be empty. From this search result, the user can click on a LL that has the book, and that will take them to the circulation activity page.

* 1. **Check In/Out Activity:**

This screen is the screen that users will arrive at when they decide to check in or out a book, or if they decide to donate to the LL. This screen will display the feed from the user’s video camera on their device. When the camera is pointed at a book ISBN, the camera will read the barcode and generate the ISBN associated with that barcode. If the user is not currently logged in, it will prompt the user to login with their credentials. Once they have logged in, that will bring them back to this screen. That ISBN will then be sent as an API call to grab information about the book from Internet Archives OpenLibrary to display relevant information to the user. Once the user confirms, the Firebase DB will be updated with the result. This activity can have a special case if the user is attempting to check in or donate multiple books at a time, where instead of redirecting back to the circulation activity, it will continue looping on the check in/out activity.

1. **Usability Interfaces:**

**A screen shot of a computer

Description automatically generatedMain Screen:**

**This is the main screen of the application.**

**Users can login via the “LOG IN” button.**

**Users can search for a book via a filtered search using the “SEARCH FOR BOOK” button or view the map with “VIEW LIBRARIES”.**

**Finally, users can view the map of their area with libraries via the “MAPS” button.**

****Login Screen:**

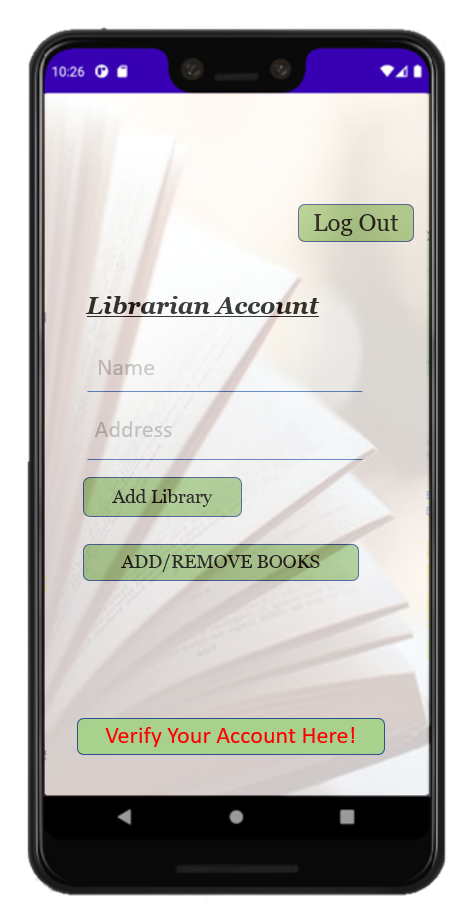
**This is login screen, reached via the “LOG IN” button on the main screen.**

**On this screen users will be asked to enter their email and password to login with an option for their credentials to be remembered.**

**Clicking “LOGIN” will take them to the main screen.**

**If an application user is not registered, they can click the “REGISTER” button to begin that process.**

**User Account Screen:**

**

**This is the user account screen.**

**The user is given the option to logout with the “LOG OUT” button, this will end their logged in session.**

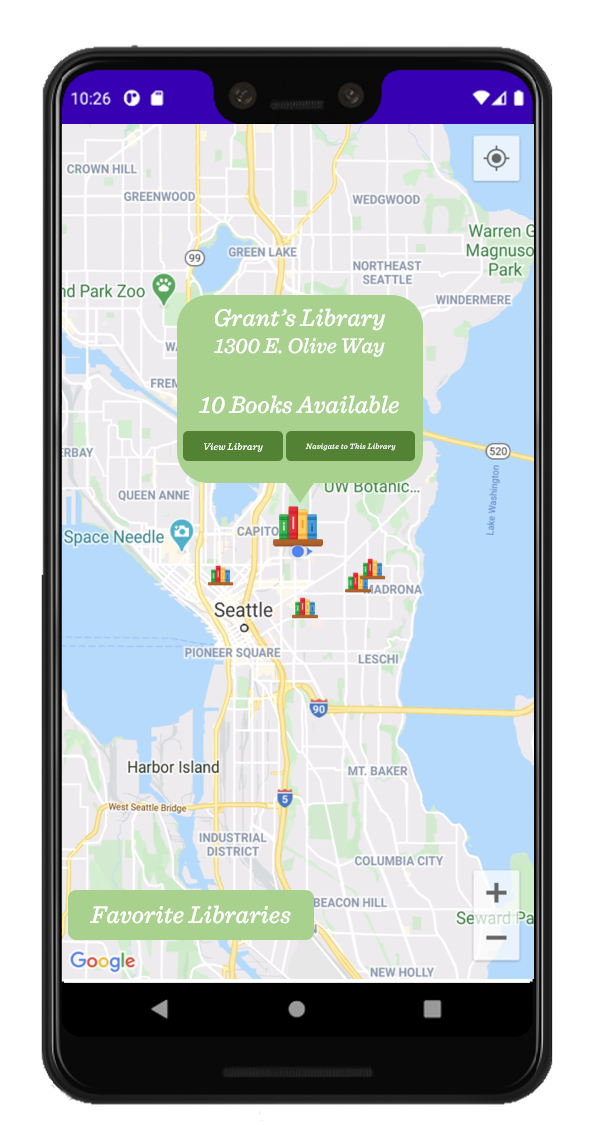
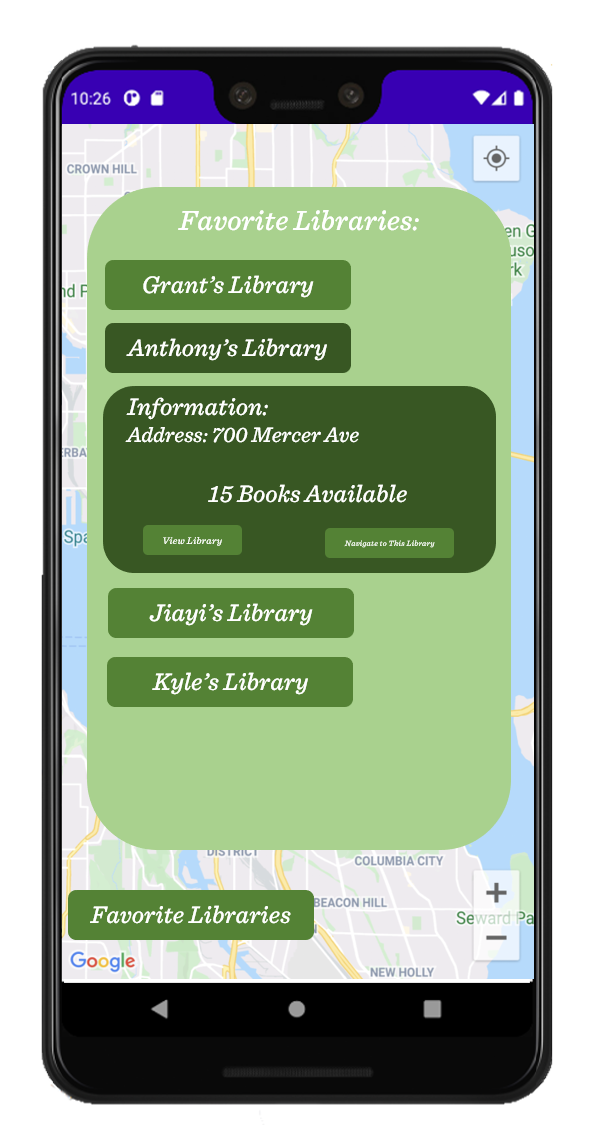
**Fields for Name and Address of their little library are provided for entry.**

**The “Add Library” button will add the library entered into the name and address fields into the system**

**“Add/Remove Books” will bring librarians to their respective library’s screen (showing all the books), from there they can remove books or add books using the ISBN camera scanner.**

**The large button at the bottom will verify that this person has an actual library and is not abusing the system with fake entries.**

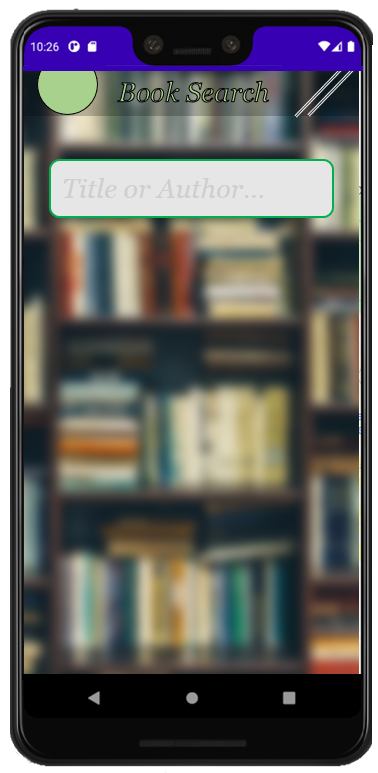
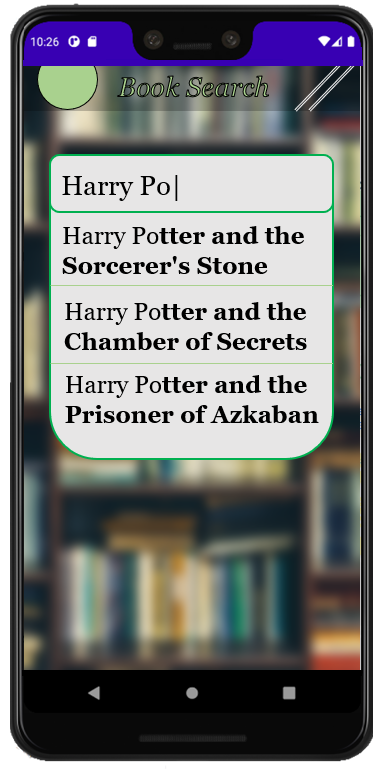
**Map Screen:**

**

**This is the map screen.**

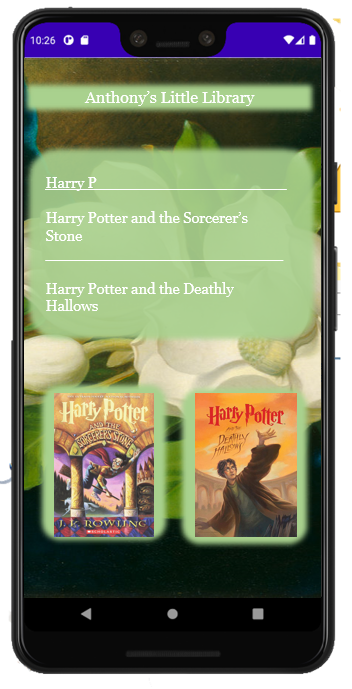
**Users can search an area close to their home for virtual libraries. Libraries will be displayed a pin on the map and users can favorite those libraries for quick access later.**

**Upon clicking the library, a separate screen with all the books is displayed.**

****Search Screen:**

**This is the search screen.**

**The user can search for a book and then the libraries that the book is held in will be displayed below.**

****Library Screen:**

**This is the library book screen.**

**Upon clicking a library on a map, or via the search screen, the circulation for that little library is shown.**

**On this screen users can manually scroll to browse or use a filtered search to see the book they want.**

**If users tap a book it will bring up the checkout screen.**

**Checkout Screen:**

**

**This is the checkout screen.**

**Users will point their camera at the EAN/ISBN barcode on the back of the book. The ISBN number will cross reference with the OpenLibrary API to ensure it is the correct book being checked out.**

**Upon checking out the book will be marked as unavailable for other users.**

**To check a book back in, the user will go back to their library and rescan the barcode to complete the process.**

1. **Requirement Traceability Matrix:**

***TBD Until Development Begins***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Req # | Req Desc | Design Specification | Test case | Test Status | Test Verification | Modification of Requirements | Remarks |
| 1 | Access to the device’s GPS |  |  |  | Yes |  | **TBD** |
| 2 | Internet connection on user’s device |  |  |  |  |  | **TBD** |
| 3 | Access to Virtual Library database(s) (User database, Inventory database, sorted databases for books/users) |  |  |  |  |  | **TBD** |
| 4 | Access to the map API |  |  |  | Yes |  | **TBD** |
| 5 | Access to privacy policy/Terms and conditions |  |  |  |  |  | **TBD** |
| 6 | Access to the device’s camera |  |  |  | Yes |  | **TBD** |
| 7 | Access to the book database API |  |  |  |  |  | **TBD** |