LiBusy

Make The Most Out of your Study Time!

Dillon Wastrack, Kevin Shannon, Alex Heath, Keith Waggoner

Change History

Summary	**Author**	**Date**
Started working on the initial document format and introduction section.	Dillon Wastrack	9/6/2016
Added definitions, goals, and high level overview	Dillon Wastrack	9/7/2016
Functional and nonfunctional requirements	Dillon Wastrack	9/12/2016
Review and Additions to first 3 sections	Kevin Shannon	9/15/2016
Added Diagram section including diagram images and descriptions	Keith Waggoner	9/21/2016
Added third activity diagram and description, added description for class diagram. Added title page, page numbers.	Dillon Wastrack	9/21/2016
Added Mockup section, mockup images, and discussion of mockups.	Alex Heath	9/21/2016
Design Requirements Added	Kevin Shannon	10/18/2016
Designed and inserted Sequence Diagrams	Keith Waggoner	10/18/2016
Added app screenshots	Dillon Wastrack	10/18/2016
Added Sequence Descriptions	Keith Waggoner	10/19/2016

Altered headings for better readability, added updated class diagram	Alex Heath	10/19/2016
Added description of updated class diagram	Alex Heath	10/20/2016

Table of Contents

1. Introduction

- ## 1.1 Motivation/Purpose
    - The motivation of this app is to give University of Alabama students real time information on libraries/study areas to help them quickly locate the best place to study depending on nearness and busyness level. Other motivations include providing students with more information on library programs and initiatives, while also benefitting the libraries/other study areas with more efficient space utilization.

- ## 1.2 Scope
    - Initially, this app will focus on the 4 main University sanctioned libraries (Gorgas, McLure, Rodgers, Bruno) to track overall busyness level, and provide specific information on available seats/tables/study rooms. Future developments could involve branching into non-library study areas, such as those in the Ferguson Center. By using people as a resource, students will be able to provide valuable information to each other across the app to save them time and energy in making the most of their time studying. We plan to have a working prototype with a functional database and server upon the conclusion of this project for the 4 main libraries.

- ## 1.3 Definitions
    - **Busyness**

        - Busyness is how full a study location is, and how noisy an area may be.
    - **Verified User**
        - Someone (usually a library employee) that has been verified as being a trusted source of information about the study area.
    - **Study Area**
        - An area on campus open for students to study, but not necessarily sanctioned by The University as a library

- ### 1.4 Goals
    - To allow students to find a place to study faster by giving them real-time data regarding nearest libraries, their busyness levels, and currently open seats, tables, and rooms.
    - To help UA address their growing problem of lack of study space by giving them a low-cost, maintainable method of utilizing space more efficiently.
    - To expose students to more library and campus resources, by providing a convenient hub on the app that students will be more likely to use because of the useful service being provided to them.

2. Project Description

- ### 2.1 High Level Overview
    - As The University of Alabama continues to grow at a rapid pace, the problem of overcrowding is severe. In no other area is this overcrowding more prevalent than in finding a place on campus to study. Our app will provide students crucial information regarding the layout of available study areas on campus, how busy they are, descriptive characteristics of these areas, and even the location of individual open seats/ tables/ study rooms in real-time.
    - The reporting of data will be accomplished through a very simplistic interface to provide a low learning curve and to encourage user-participation.

- ## 2.2 Features
    - View map of currently open campus libraries (color coded busyness level)
    - View individual library's details and associated busyness ranking
    - Perform advanced "find a library" search
    - Access library resource hub
    - Check in at library
        - Location-based, time-delayed notification
        - Report busyness level
        - Report current floor to begin marking open seats/ tables/ rooms
    - Mark Available
        - Seat
        - Table
        - Study Room

3. Requirements

- ## 3.1 Functional Requirements
    - Detailed mapping of open seats, tables, and study rooms available to students
    - It should take a user less than 10 seconds to give general busyness input on the status of the study area.
    - Provide a menu showing open study locations across campus, with the ability to sort based on various criteria
    - Provide an interactive map with pins on the various study areas throughout campus, pins will contain useful information
    - A one click handoff to Google Maps to provide directions to any study area the user desires.
    - Location based, time-delayed notification for check-in activity

- ## 3.2 Nonfunctional Requirements
    - All menus should be navigated quickly and efficiently
    - Simplicity and reliability are the highest priorities for this system.

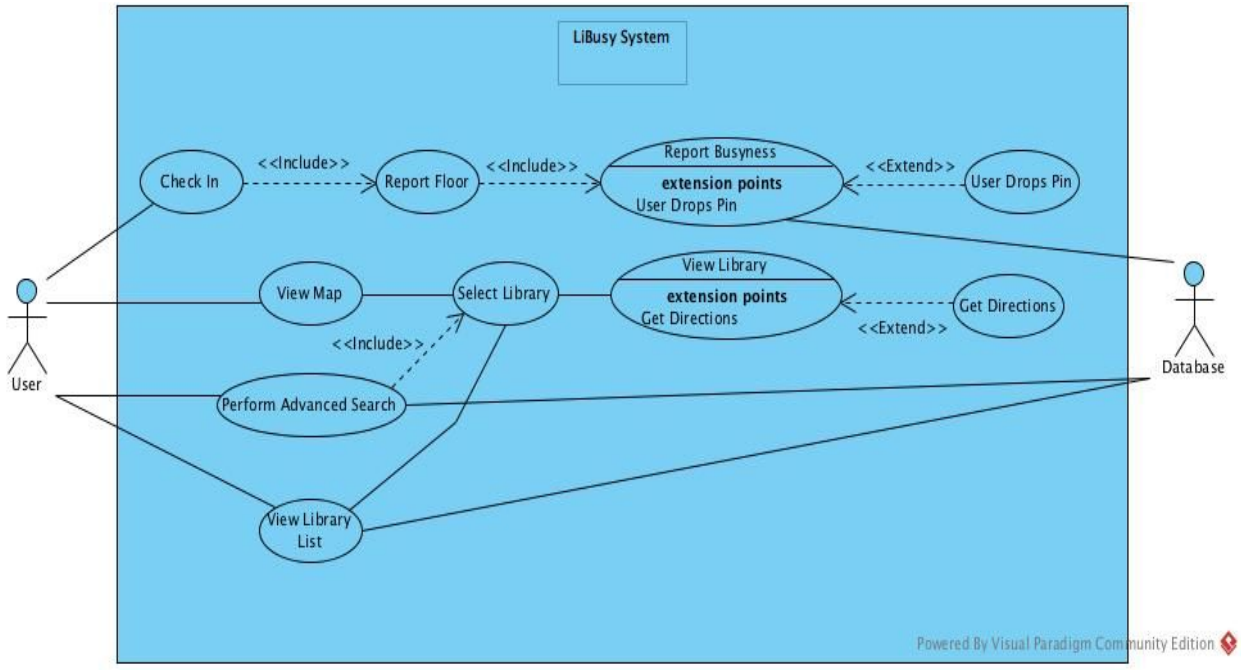
- The quality of our data depends on user satisfaction and retention.
- Users should be able to quickly and easily provide information on the app

○ Security is also a high priority

- Users should remain completely anonymous, giving only details that they are comfortable giving
- No personal information on users should be stored by any of our databases to lower responsibility in the event of a security breach.

4. Requirements Diagrams

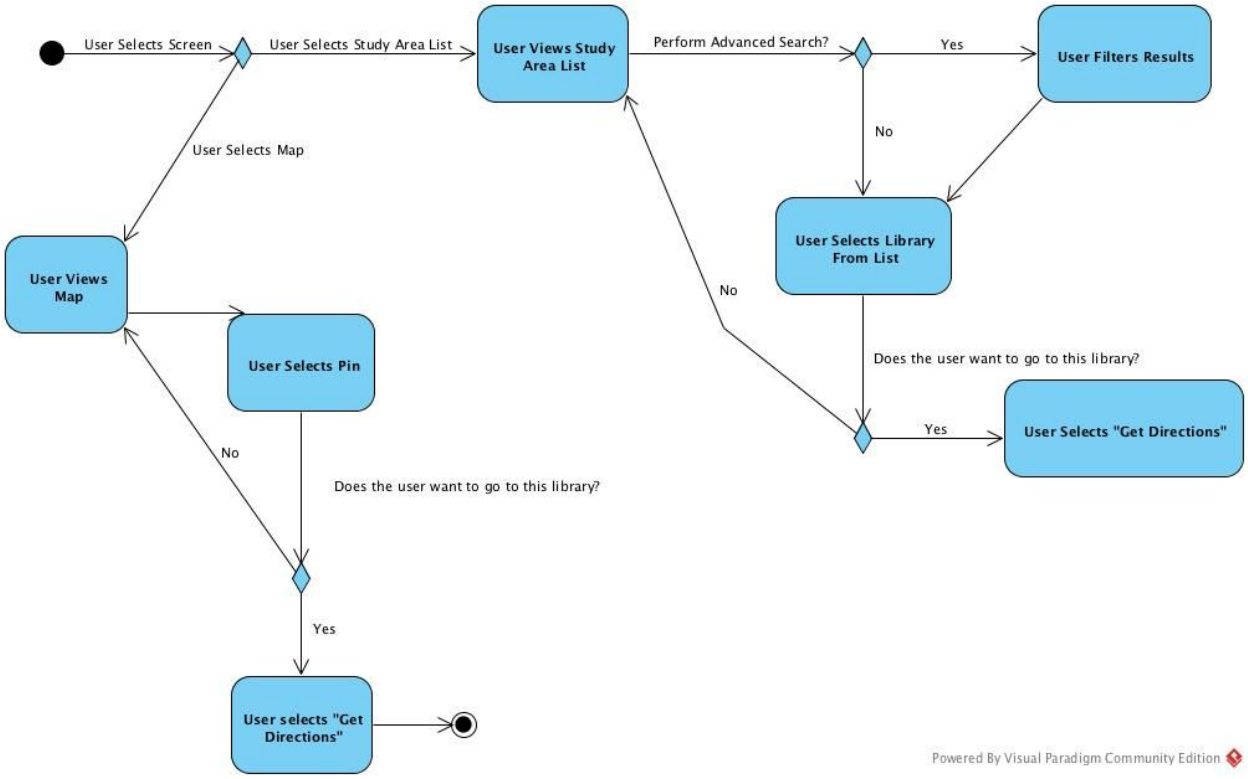
● 4.1 Use Case Diagram



Our use case diagram describes the overview of the usage requirements for our library app system. The first actor labeled “User” has four options when entering our app. He may choose to check in, view the map, perform an advanced search, or view the library list. The diagram also describes the role of the database in our system. The database actor provides

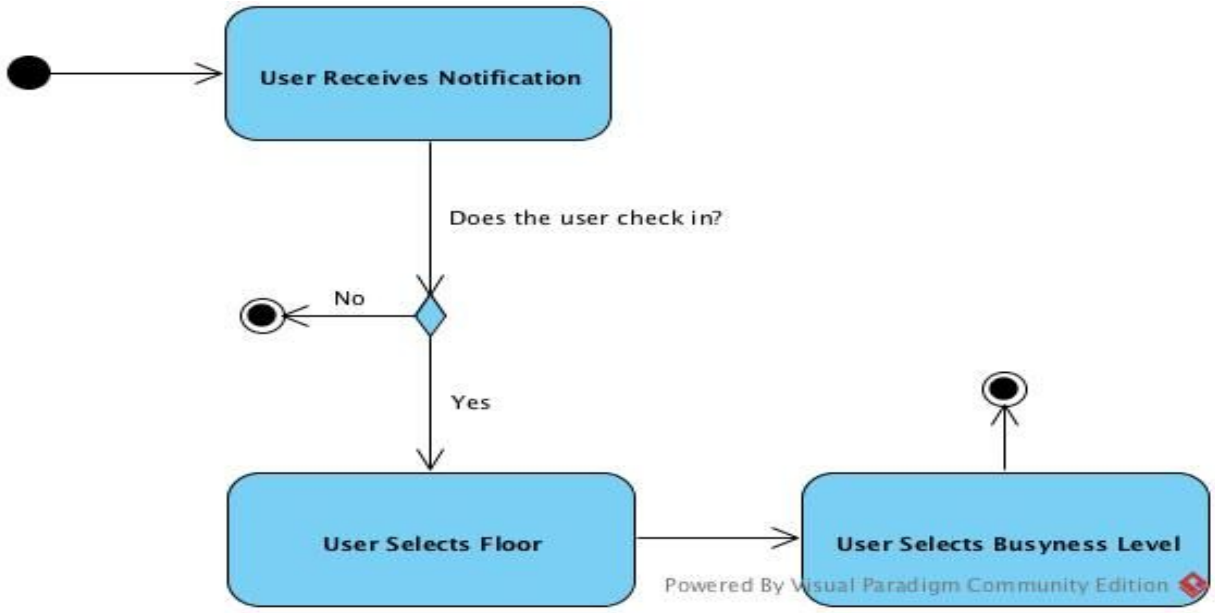
and stores information such as busyness level, advanced search queries, as well as the list of libraries available.

- 4.2 Activity Diagrams

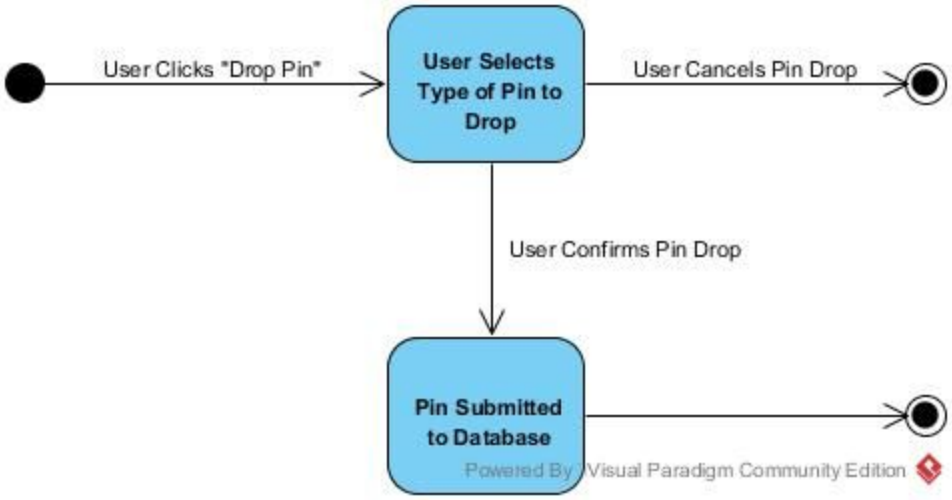


Our first activity diagram describes the screen selection activity. As shown in the user has two options, he may select the Study Area List or the Interactive Map. If the User selects the Study Area List he then has the option to perform an advanced search on all the different

libraries. If the User selects the Map he then has the option to select a pin and get directions to a specific library.

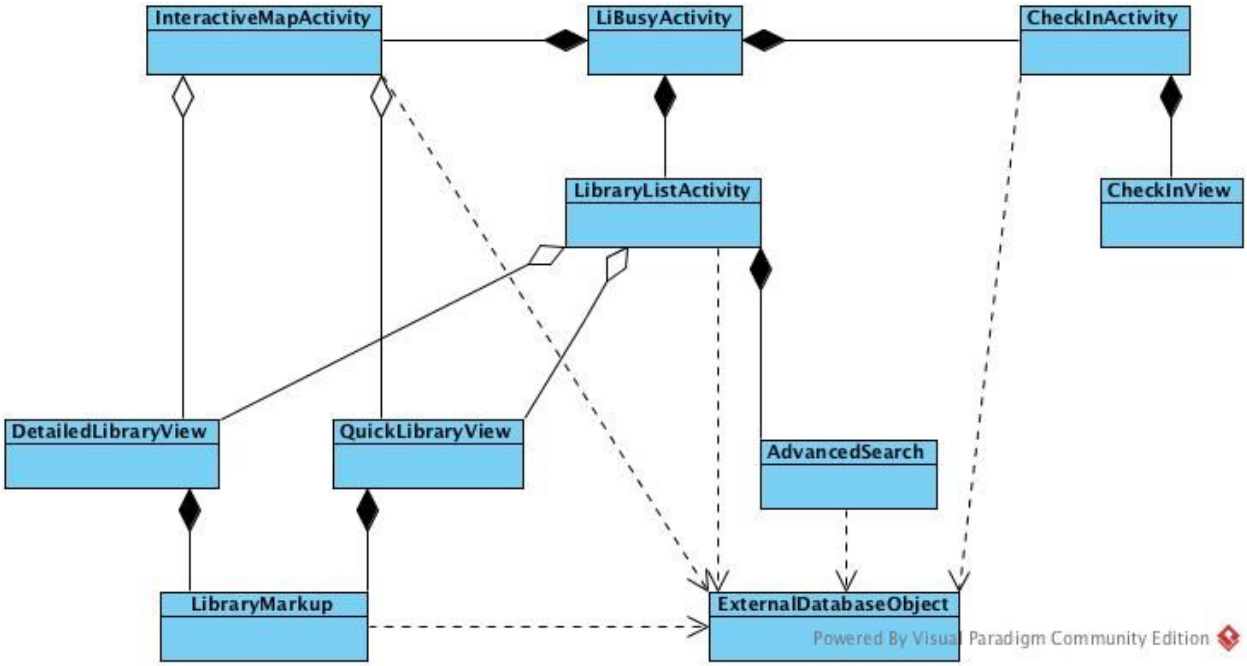


Our second activity diagram describes the notification process. Upon entering a library the User is prompted with a notification that gives him the option to check in to that particular library. If the User does decide to check in he will then select the floor he is currently on followed by the busyness level of that floor.



This diagram describes how a user drops a pin on the map to specify their location. The user simply selects the option to drop a pin, taps the location where he wants to drop it, and it is submitted into the database.

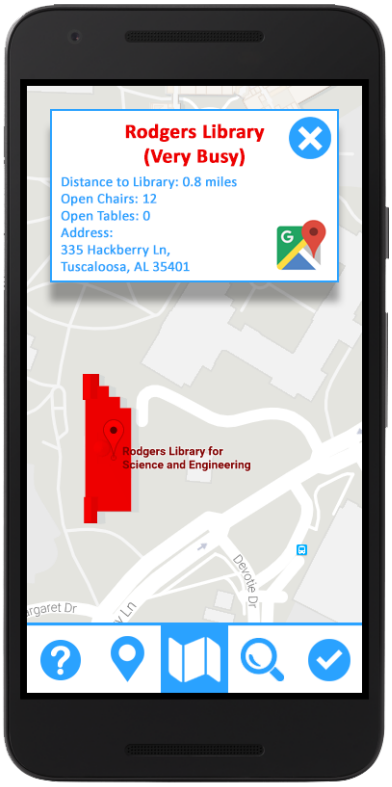
- 4.3 Class Diagram



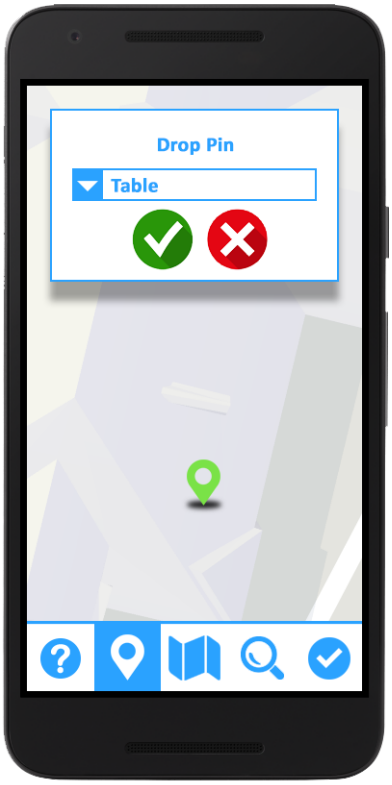
- The LiBusy activity represents the entire application. This is further composed into three activities: the InteractiveMapActivity, the LibraryListActivity, and the CheckInActivity. These are described in the list below.
    - **InteractiveMapActivity**
        - A campus map with pins on the location of study areas, tapping the pin engages the QuickLibraryView, double tapping engages the DetailedLibraryView.
        - Has two aggregate classes, the DetailedLibraryView, and the QuickLibraryView, which contain detailed, and minimal information about the library respectively.

        - Dependent on the database for it's information, denoted by the dependency arrow to the ExternalDatabaseObject.
        - The DetailedLibraryView and QuickLibraryView are composed of the LibraryMarkup class, which contains information about the library stored in the database. This dependency on the database is denoted by the dependency arrow to the ExternalDatabaseObject.
    - **LibraryListActivity**
        - Provides a filterable list of libraries with information on the state of each.
        - Uses DetailedLibraryView and QuickLibraryView as aggregate classes.
        - Composed of AdvancedSearch, which allows filtering and ordering of results based on things such as proximity and busyness level.
        - Dependent on the database for information on libraries, denoted by the dependency arrow.
    - **CheckInActivity**
        - Used to report the busyness level and number of empty seats in a library.
        - User provided information is stored in the database, shown by the dependency arrow to the ExternalDatabaseObject.
        - Composed of the CheckInView, which provides an interface for the user to provide input.

5. Mockups



This is a mockup of what the “Map” feature will look like on a Nexus phone. The bottom of the mockup features a toolbar with five buttons: “Library Information”, “Drop a Pin”, “Map”, “Search for a Library”, and “Check In” (left to right). After pressing the “Map” button in the middle, the rest of the screen space will become a map that the user can use multi-touch to zoom in on and manipulate to find libraries. If a library is clicked (Rodgers Library in the mockup) a new layer will appear over the map detailing the library’s information including our metadata. The user can then send the phone to the native maps application with the address data for easy directions, or close out the layer and continue to traverse the map.



This is a mockup of the interface for dropping a pin and supplying metadata to the application. If the user presses the aforementioned “Drop a Pin” tab in the toolbar, a new layer will appear with a dropdown box with the possible types of pins to drop (Table, Open Seat, etc.) and two buttons which submit the pin or close the layer. The user may then submit the pin by pressing the check, or cancel their submission by pressing the “X” button. After making their decision, the user will be taken back to the tab on the toolbar they had open before pressing the “Drop a Pin” button.

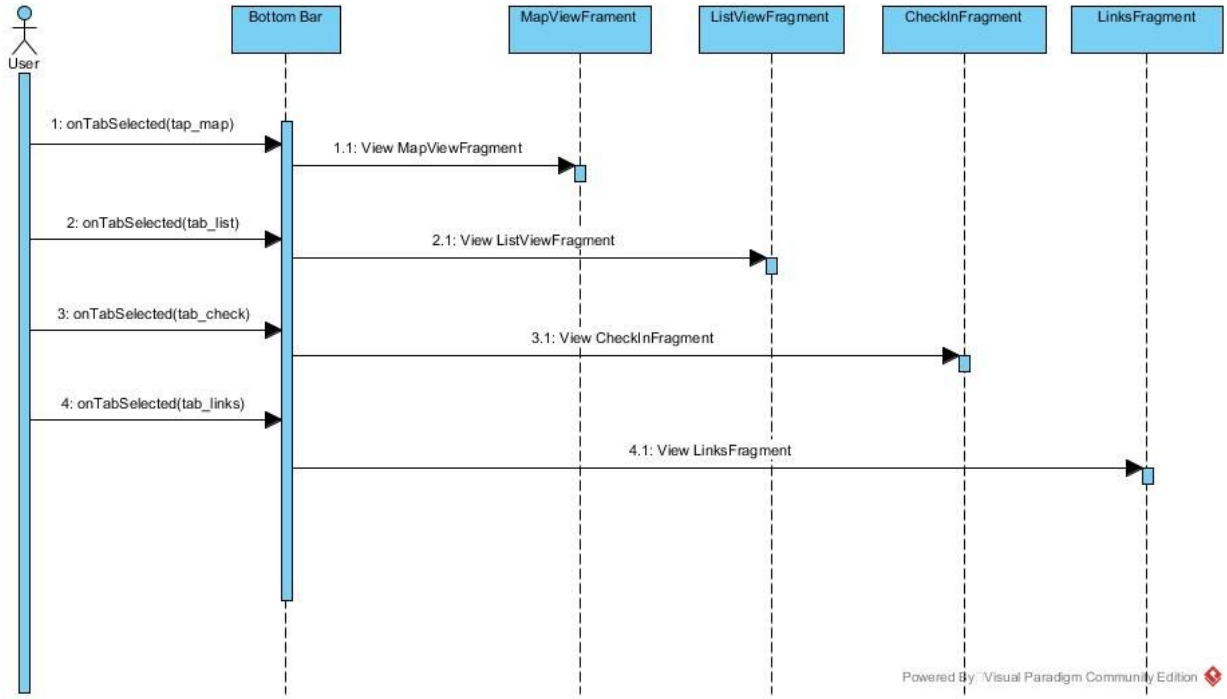
6. Design Changes

1. Using minimal Heroku DB
    a. Subject to change depending on downtime constraints and data limitations
2. Possible changes to individual seat markings
    a. Level of depth will be time sensitive
    b. Must investigate gps granularity using android device
    c. Still hoping to at least have a count of individual seats

7. Design Diagrams

- 7.1 Sequence Diagrams

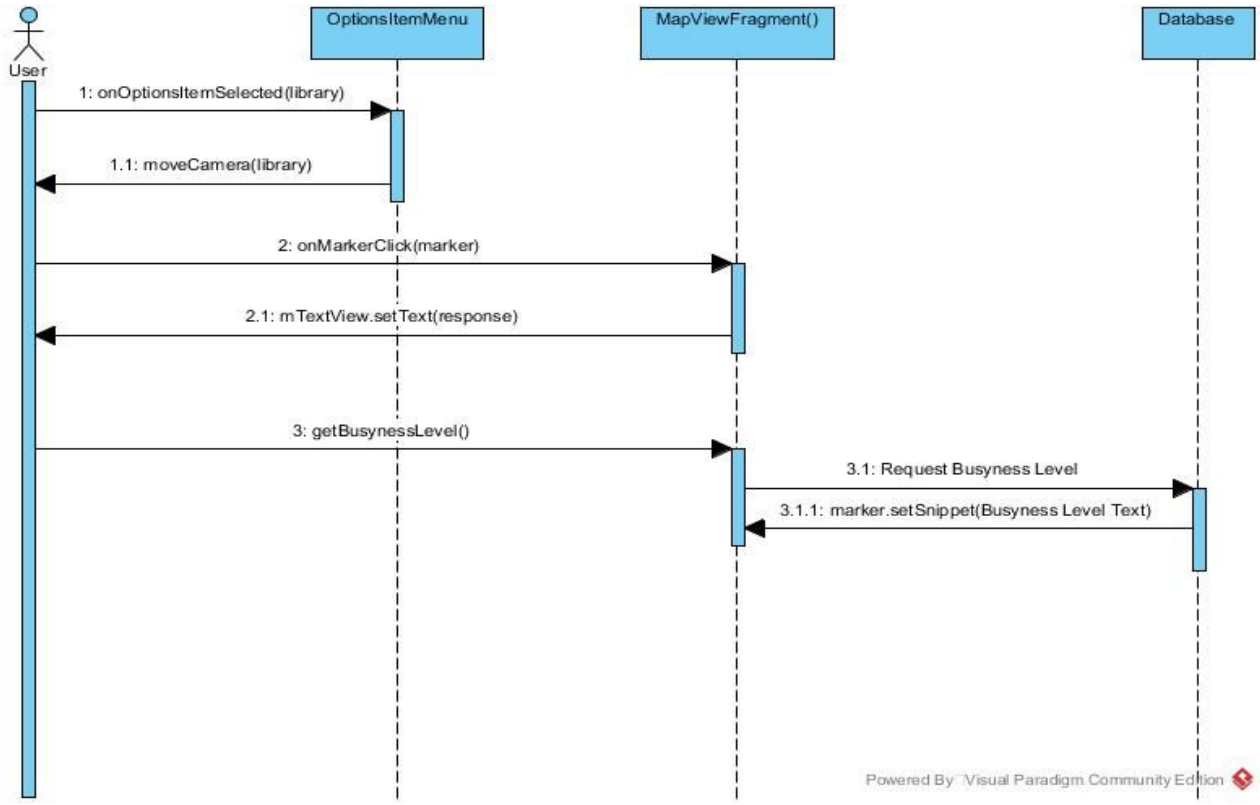
Main Screen Selection:



The main screen sequence diagram consists of selecting different available options from our bottom tab bar. The user can decide to view the map containing all of the different

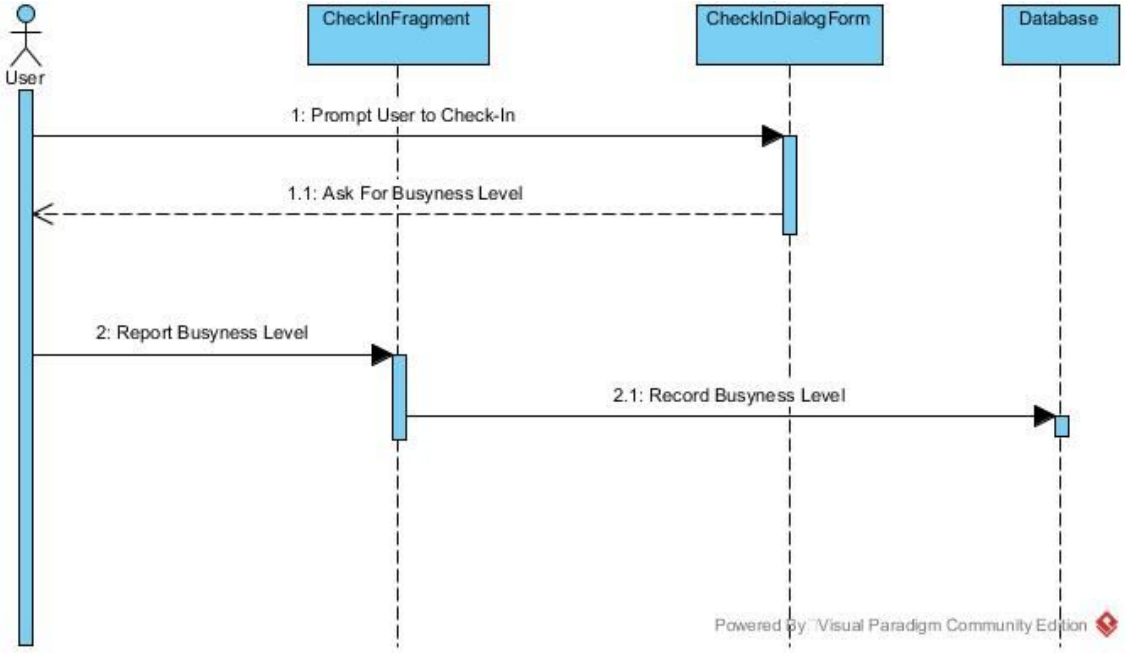
library location, the ListViewFragment, the CheckInFragment, or the HelpfulLinksFragment. Each of these options presents the user with a different interactive screen consisting of various features that our app provides.

Map View Fragment:



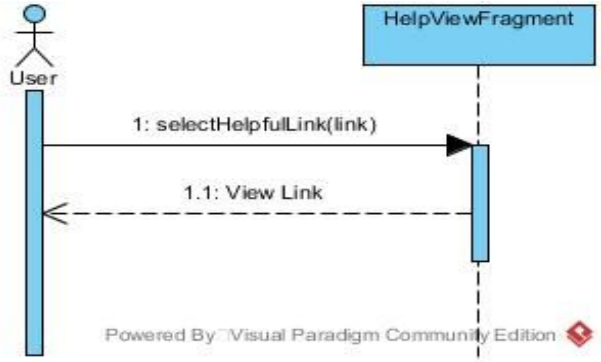
In the Map View fragment, users are presented with a OptionsItemMenu where they can select from a list of libraries, rather than having to search for them on the map. From here they can select a specific library which will move the map camera to that particular library. From here the user can select the marker for the given library, or can navigate to any other library marker on the map. Once a particular marker is selected for a library, the user will be supplied with the text view of the library with associated data, such as busyness level, seats available, etc. Then finally the user will be prompted with a pop up message to provide their opinion on the busyness level.

Check-in Fragment:



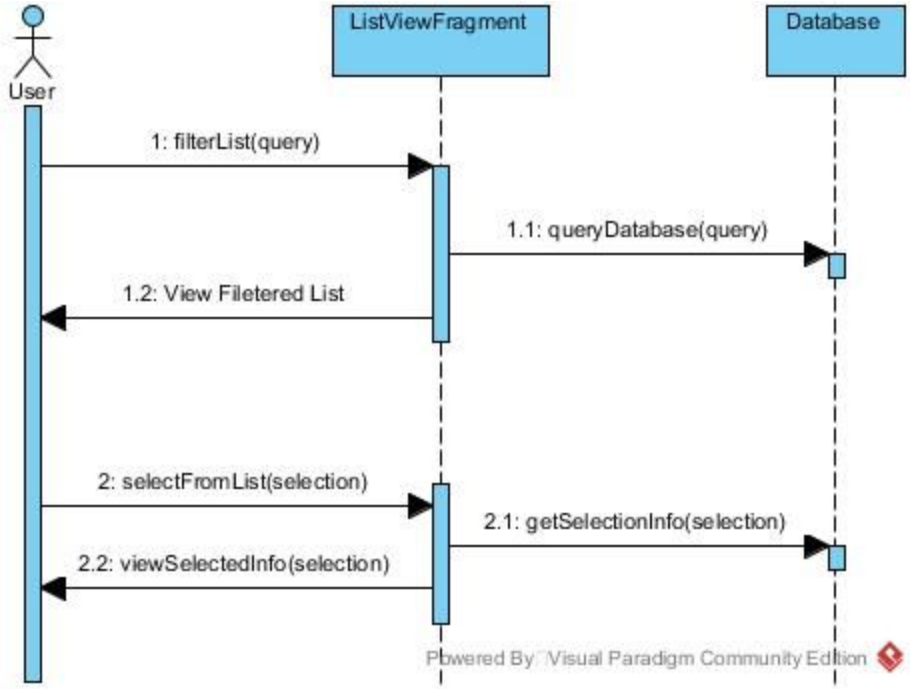
The Check-in activity first prompts the user with two options, whether they wish to check-in to the library that are currently at or not. After they decide to check in the user is prompted with the option to report their opinion of the busyness level of that library. That information is then stored in our database and used to accumulate a good estimate of the busyness level of that library so other users to gain access to that information.

Links Fragment:



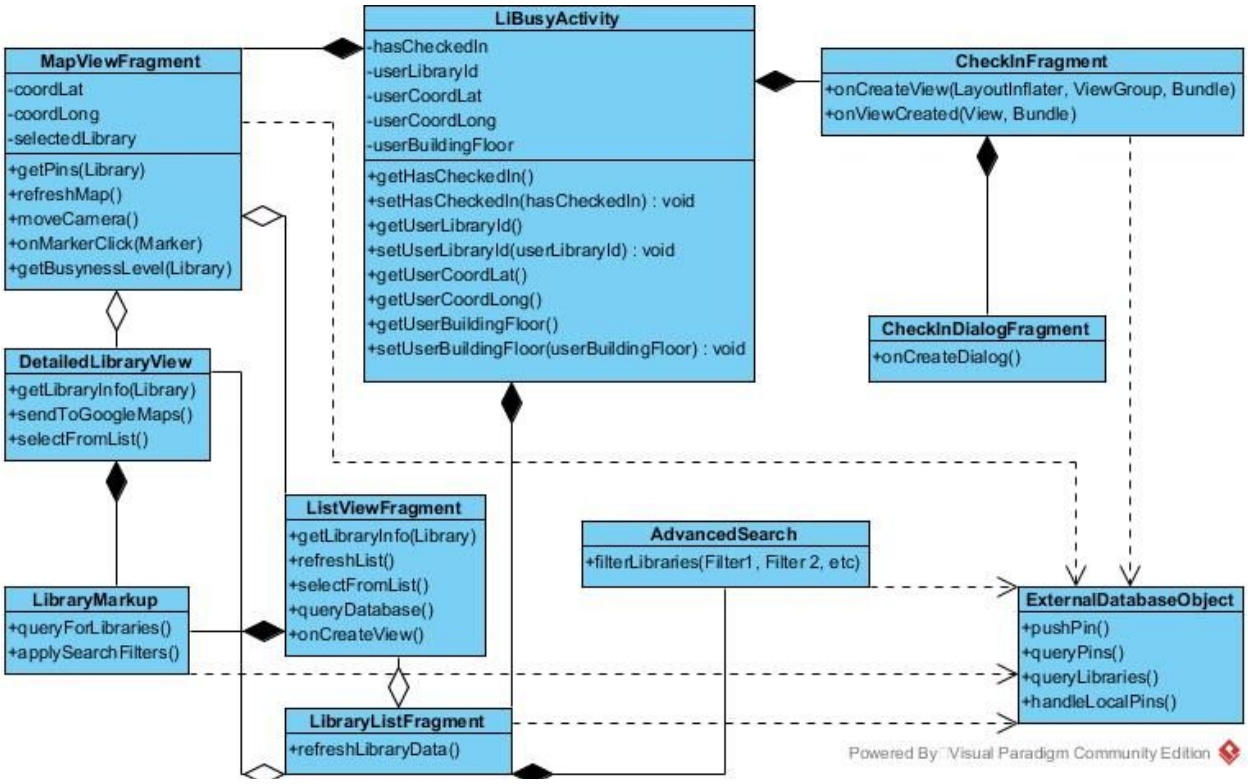
The HelpfulLinksFragment allows the user to view and select various helpful links about the libraries on campus. These links and resources will be provided by the university and displayed as an extra tool on our app.

List View Fragment:



The ListView fragment filters the list of libraries based on given user input (such as busyness level, distance, noise level, etc.). Based on this criteria, the app then queries the database to find the best available library for the given user-specifications. The user will then be presented with a view of the filtered list, from which they can make their selection and proceed to view the selected info.

- 7.2 Updated Class Diagram

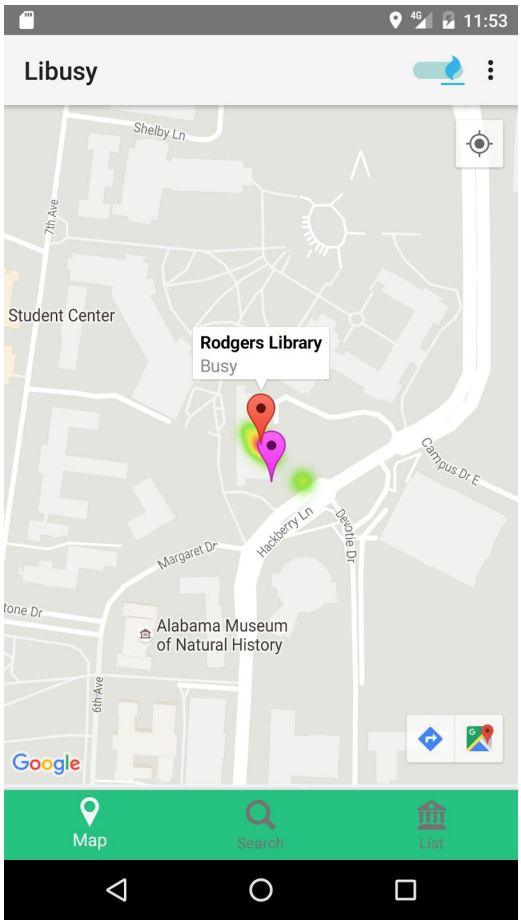


In our current iteration the LiBusyActivity acts as a hub through which much of the local data is stored. Information such as the user's current latitude and longitude is stored here, as well as their current check-in status. These values can be updated through the class's getters and setters. Each page in the application has it's own fragment which communicated with LiBusyActivity. The CheckInFragment is fairly barebones and acts only to initialize changes in the user's check-in status. This fragment interacts with the CheckInDialogFragment which contains the buttons to check in. The MapViewFragment is the fragment which uses Google's Maps API to create a visual representation of the library information, including pins and heatmaps generated from user submitted pin data. This class has coordinates which are used to place the map where the user intends to search and a field for storing which library (if any) is currently selected by the user. Methods to refresh the map, move the map to a specific coordinate set, and fetch library and pin data are available. If the user decides to select a library for further viewing, the DetailedLibraryView class is invoked, which has functions to fetch library data from the external database, send address data to the device's Google Maps App, and select a new library from a list. Library information is retrieved

using the LibraryMarkup class, which handles queries of the database for library data. If the user chooses to select a library through the list feature instead of the map, the ListViewFragment is invoked. This class has methods to fetch library data (again through LibraryMarkup), and refresh the data. Refreshing is done by accessing the LibraryListFragment class. The LibraryList Fragment can also invoke the AdvancedSearch class to return a *filtered* list of libraries based on some criteria that the user chooses. Each class that involves querying the database for pin or library data accesses the ExternalDatabaseObject, which has methods to query for pins or libraries, push new pins to the database based on user input, or perform functions on the pins local to the user such as creating a heatmap from them.

8. App Screenshots

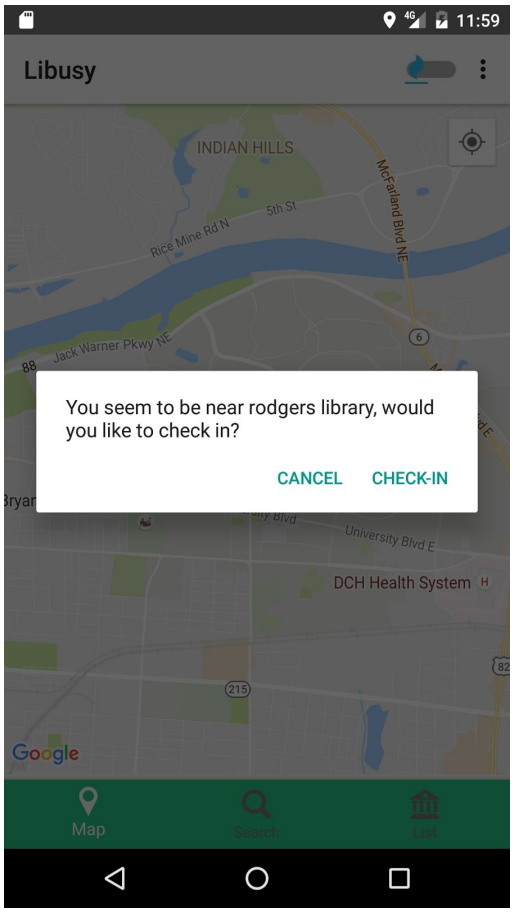
- 8.1 Heatmap



- This screenshot shows the interactive map view, with the optional heatmap overlay.
- Notice the switch in the action bar that is activated. This signals that the heatmap is turned on. Tapping it again will turn the heatmap off.
- The heatmap is generated through retrieving a set of latitude/longitude points from the database representing current user location, and using the Google Maps Heat Map Utility to generate the heatmap
- Also notice the magenta marker, which represents the current user location, and the red Rodgers Library marker with the busyness rating.

- The busyness rating is computing by taking the average of all user ratings within a certain period of time.
- After tapping a library marker, the user can tap the “get directions” button to be routed to the library using Google Maps.

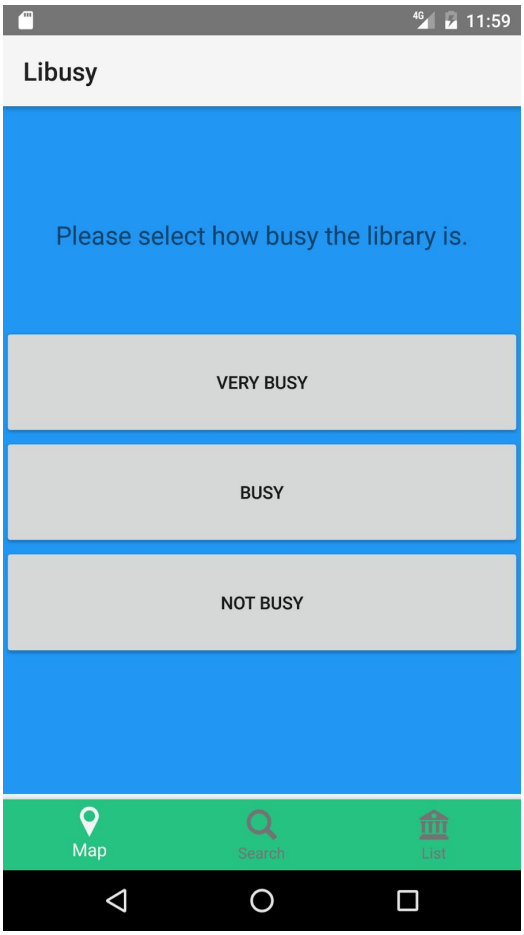
8.2 Check-In Dialog



- This is the check in screen that pops up when the user opens the app within 50 meters of a lat long coordinate representing the center of a library.
- To make the dialog location aware, the app determines the user’s closest library, and checks if he/she is within 50 meters of it. If the user is not within 50 meters of any library, no check-in dialog is shown.

- Inside the dialog, the user can choose to report the library's busyness, or hit cancel to simply view the map.

- 8.3 Check-In



- If the user chooses to check in, this screen pops up giving him the options for reporting busyness.
- When the user makes his selection, his response is sent to the database to be used in computing the library's overall busyness.