Laker for a Lifetime



Design Document

Daniel Newell, Nickolas Workman, Thomas Peterson, Joshua Walker

**Outline**

1. Introduction
2. Design Pattern and Procedures
3. Languages and Development Tools
4. Backend Architecture
5. User Interface
6. Milestone Schedule

**Introduction**

Our team has been commissioned to complete a mobile app for Grand Valley’s philanthropy organization. They are currently running a promotion called “Laker For a Lifetime”, and they would like to promote the efforts of past donors who have made a big difference at Grand Valley.

Most students who attend Grand Valley are unaware of the people who helped to fund and found this University, and so this app would be an attempt to easily educate students on information about various important landmarks and buildings on Grand Valley’s campus.

The main focus of this application is to utilize a phone’s location to show the user information about landmarks that they are near and to highlight the contributions and efforts of the donors who helped to make these landmarks possible.

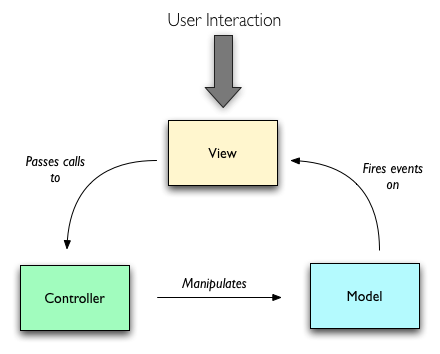
**Design Patterns And Procedures**

We plan on taking an agile approach to our design. After meeting with our client, we have found that the app that we are developing is still more of an ‘idea’ than it is a clear-cut set of requirements. Therefore, adopting an agile approach to the development of this app will allow us to change portions of our app later down the road if we run into changing or updated requirements.

We will also be making use of unit tests, as this sort of testing should go hand in hand with agile programming. Using this type of testing will allow us to make changes to our app far into the development cycle, while remaining confident that our product is still working the way that it is intended. We will also be making use of an iPhone/iPad simulator in xCode to help test our User Interface and ensure that it works as we intend. We would also like to test our app on native devices to further make sure that everything comes out the way we would like.

Our code repository should just have a Master Branch, as the majority of this application should only be in a few files, however if the project extended beyond a simple application, we could choose to split it in order to help keep everything straight.

For our design methodology, as is the default case with iOS development using xCode, we will be developing our application through the lens of the model-view-controller (MVC) architecture pattern. Most of the beginning tutorials on iOS development have mentioned MVC as the desired design pattern for an iOS app, and in addition MVC development is a very common practice for apps (both web and mobile).



*Model View Controller Design Pattern*

**Delegation of Responsibilities**

We would all like to be involved with every portion of the development, however we realize that being deeply involved with every portion of the project would not necessarily be the most efficient approach. So while we plan to all be involved in each portion of this project, we have separated our responsibilities into main focuses.

**Josh Walker** - Location Management, Data Management

**Nickolas Workman -** User Interface, Client Liaison

**Thomas Peterson -** User Interface

**Daniel Newell –** Location Management, Data Management

**Languages and Development Tools**

For the actual coding and development of our application, we will be using **Objective-C** as the primary language of development. We plan on making use of the Macs in the Kennedy building downtown, and I believe a few of us will be also remotely working from home on the project as well.

For our development environment, we are choosing to use xCode. This seems to be the easiest setup and least painful to learn, as it does a lot of the work up front for us.

**APIs/Other Tools**

Since one of the main functionalities of this program requires the information about the phone’s location, we will be using the iOS Core Location Framework to help us find the location of the user phone and compare it to the location of relevant landmarks.

We are also debating on which back-end data manager to use. We have narrowed our options down to SQLite and the iOS Core Data. Both have advantages and disadvantages… **(Update This Once We Have Decided What To Use)**

**Back-End Architecture**

Using the Model-View-Controller paradigm allows us to easily outline the back-end of our application. Both the model and the controller are the main components of our back-end, while the view is the user interface.

First, we will need a class to manage our data. We plan on using (SQLite or Core Data) for this class, and it would act as an interface between core data and our model.

Second, we need a class to manage the location aspect of the phone. This class will reference the Core Location framework, and return the current location of our user.

We then will need a class that will bring these two together, and will call to the data manager to find the relevant locations that are nearby the user’s current location. This class would be the equivalent of a ‘main’ class, that will handle necessary calls to other classes, and gets commands from the controller, and returns data to the view. These three classes should all be a part of the model.

The controller will be used to make calls to the main model class which will allow us to return data that is relevant for the view based upon commands that have been passed from the user interface to the controller.

**User Interface**

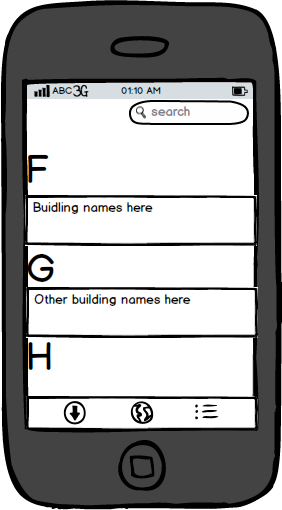
We envision our user interface having three main views. The landing view will list off the locations that are nearby by default.



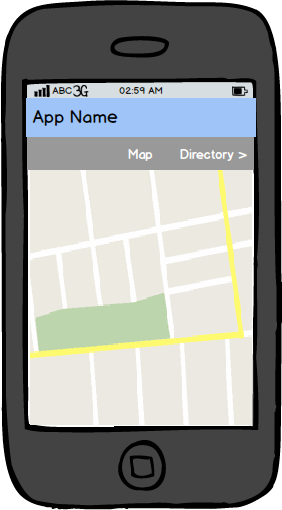
When tapping on one of the landmarks, the interface will transition to a view that shows detailed information about the building. This view will have pictures of the building, the donor(s) who are related to the building, and information about both the building and the donor.



We will also develop a view that will act as a searchable directory, that will show all of the buildings between each campus, and will allow the user to see information on any of these landmarks without actually having to be near them.



Finally, we have a map view that will show each of our relevant locations on a map, similar to the way the google maps works.



Milestone Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Start Date | Duration (Days) | End Date |
| initial Research | 1/14/2013 | 16 | 1/30/2013 |
| Requirements Gathering | 1/17/2013 | 1 | 1/17/2013 |
| Planning/Design | 1/20/2013 | 10 | 1/30/2013 |
| User Interface Prototype | 2/4/2013 | 11 | 2/15/2013 |
| Location Gathering | 2/4/2013 | 11 | 2/15/2013 |
| Data Management | 2/4/2013 | 11 | 2/15/2013 |
| Main Model Class | 2/18/2013 | 20 | 3/10/2013 |
| Controller (transitions) Development | 2/25/2013 | 21 | 3/18/2013 |
| UI Critique and Changes | 3/19/2013 | 14 | 4/2/2013 |
| Testing/Debugging | 3/22/2013 | 7 | 4/2/2013 |
| Final Test/Debugging/Minor Changes | 4/5/2013 | 10 | 4/15/2013 |
| PCEC Project Day | 4/18/2013 | 1 | 4/18/2013 |
| Final Presentations | 4/25/2013 | 1 | 4/18/2013 |

