**SYSTEM**

**DESIGN SPECIFICATION**

**FOR**

***Mouseville***

Change History:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Reason for Change |
| 1.0 | *4/18/14* | Ben | Original |
| 2.0 | 4/20/14 | Ben | Update |
|  |  |  |  |

**Document Cross Reference:**

|  |  |  |
| --- | --- | --- |
| No. | Title | Reference |
|  |  |  |

Approvals:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Role** | **Name** | Signature | Date |
|  | Project Sponsor |  |  |  |
|  | Project Manager |  |  |  |
|  | Team Lead |  |  |  |
|  | Operations |  |  |  |
|  | Shared Services |  |  |  |

# System Definition Summary

The goal of the Mouseville iPad application is to create a mobile application that employees of the Wexner Medical Center will use to manage data on research mice at their facilities. Examples of data kept are name, gender, date of birth, and cage position in the rack. Typical tasks are looking up mice by each of these mouse identifiers, editing mice, and exporting them in a CSV file.

# Architecture - Logical View

## Package Details

The program package contains the Xcode project files and all supporting files, including images used to create the application. It also includes all database files used to both create the database and files that were created when the application was used.

All libraries and source files from Xcode are not included in the package. It is assumed that any user opening the project file will have the correct version of Xcode and OSX installed upon trying to launch the project and/or application.

## User Interfaces

**Operating System**

It is a requirement that the application is interacted with on the iPad, an iOS device.

**Development Framework**

There do exist a variety of mobile application development frameworks designed for creating applications that can be ported to multiple mobile operating systems with minimal amount of rework necessary. PhoneGap is one that the Mouseville team was looking at in particular in early development.

However, such frameworks typically offer only a limited subset of features available to native applications. Additionally, an application created from such frameworks will typically exhibit poor performance when compared to an equivalent native application. Because it was desirable for the initial release to perform as smooth as possible, as well as looking exactly as the sponsor desired, it was decided to create the initial release as a native application instead of using a third party framework.

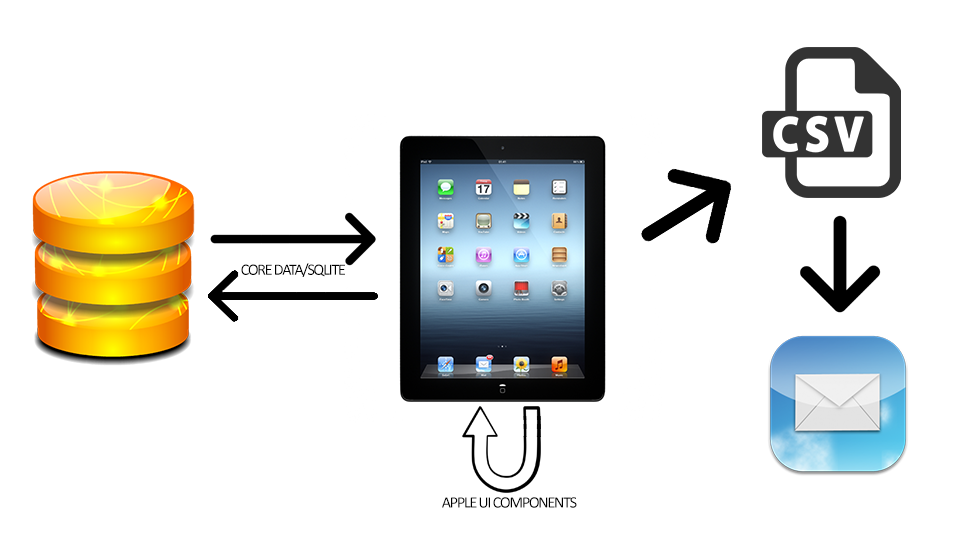
**Target APIs**

Since there have been multiple versions of Apple operating systems, the first question was whether or not it was worth developing compatibility for versions below the current version (version 7). After some research, it appeared that over 80% of the general populace had upgraded to iOS 7. Since the vast majority were on iOS 7, the team decided to only make the app compatible for iOS 7 and above.

Source: <http://david-smith.org/iosversionstats/>

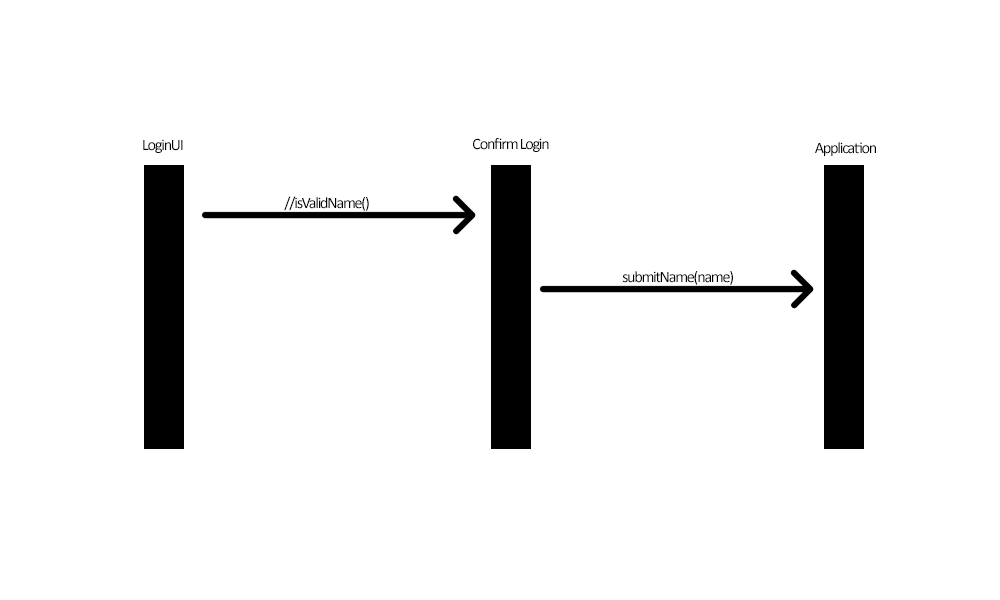
## Component Diagrams - system Interfaces

The Mouseville application contains four main components: the database, application, CSV file export, and email application. Data is read and saved between the application and database through core data and SQLite and calls are made within the application through Apple’s integrated UI components. To export the data, a CSV File is used as an intermediate file between the application and email application to easily transport the database to an external program.

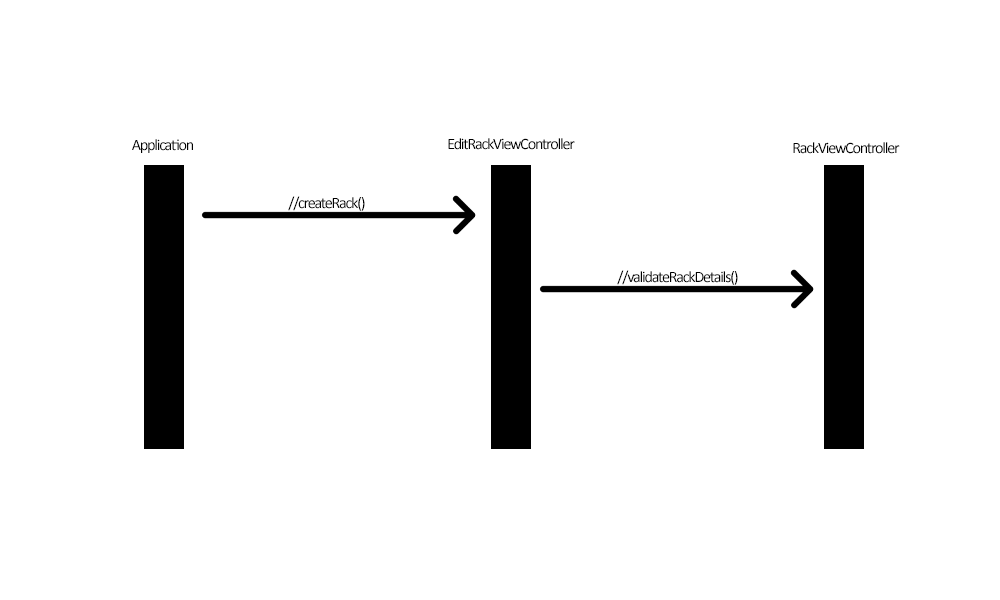


* 1. ***Sequence Diagrams/Object Collaboration Diagrams***

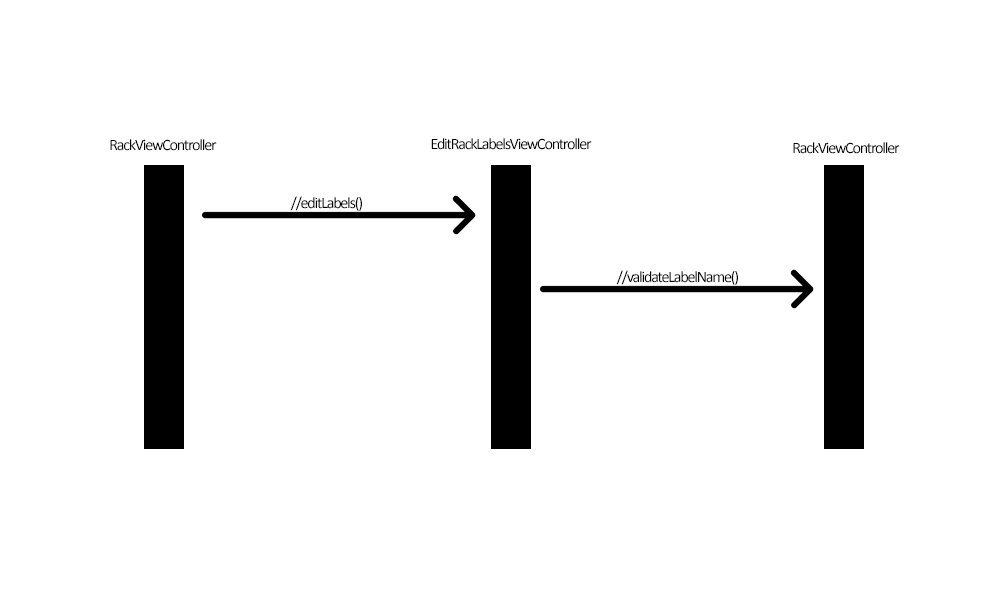
1: Login



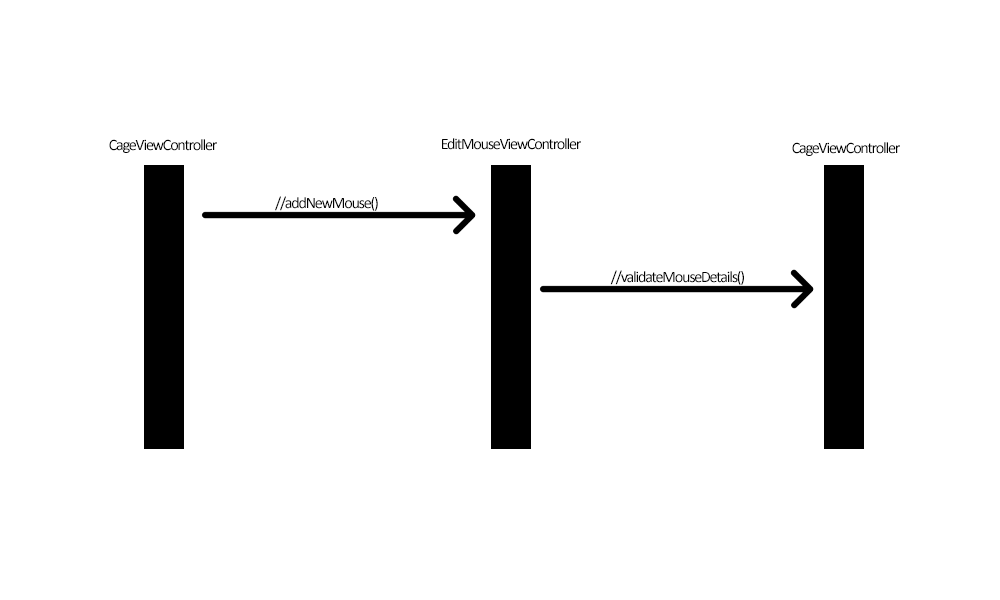
2: New Rack



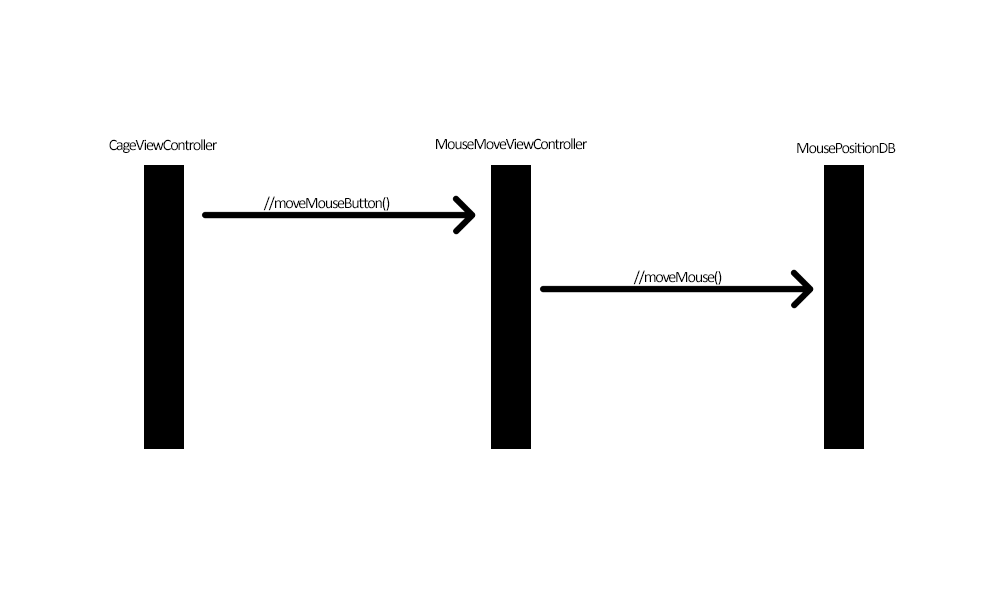
3: Edit Rack Details



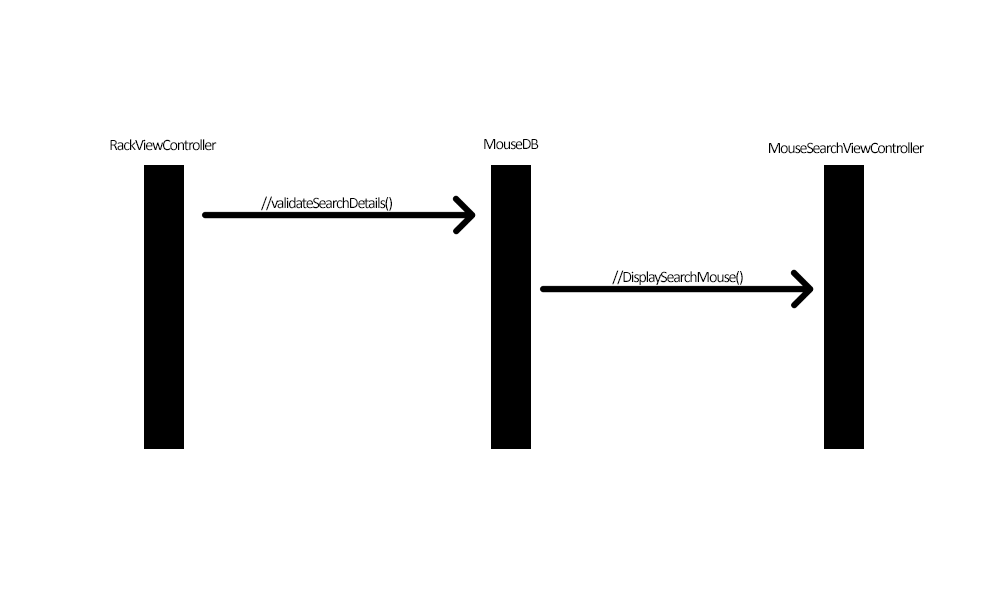
4: New Mouse



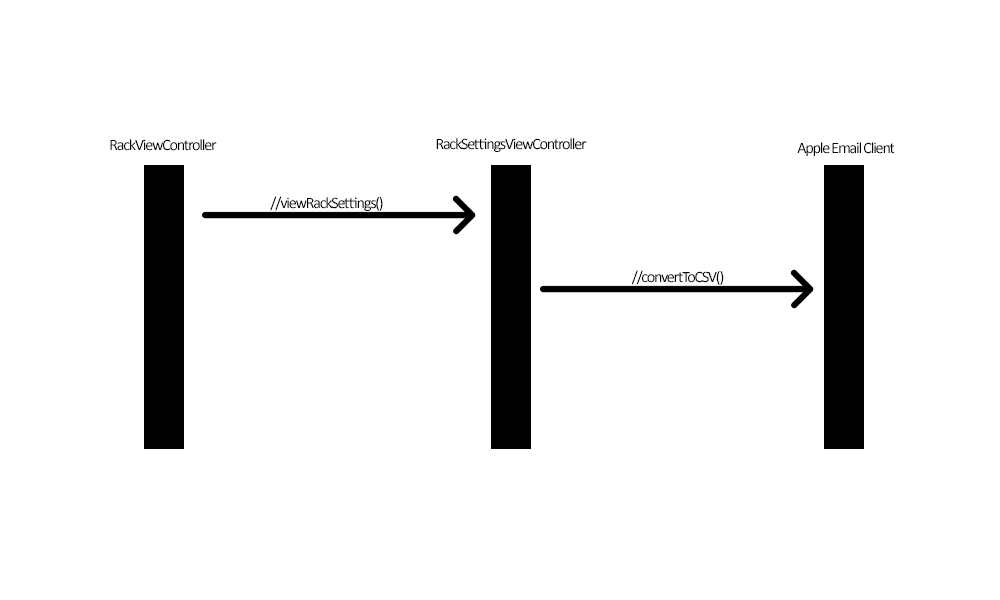
5: Move Mouse



6: Search Mouse

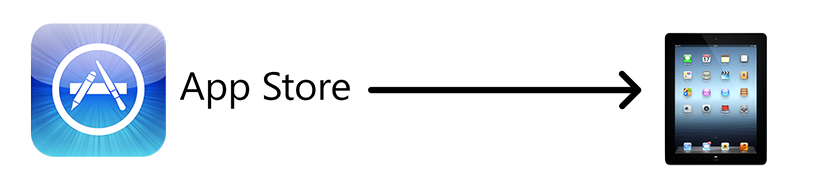


7: Export to CSV



***2.5 Deployment Diagram***

Because Mouseville is a single user application that contains no Internet connectivity. The deployment diagram only requires the user download Mouseville from the Apple App Store.



# Outstanding Issues

The Mouseville team was unable to test the Export to CSV function correctly because the iPad simulator built into Xcode did not have functionality to examine files created by applications run in it. Therefore, the function may not work as intended.