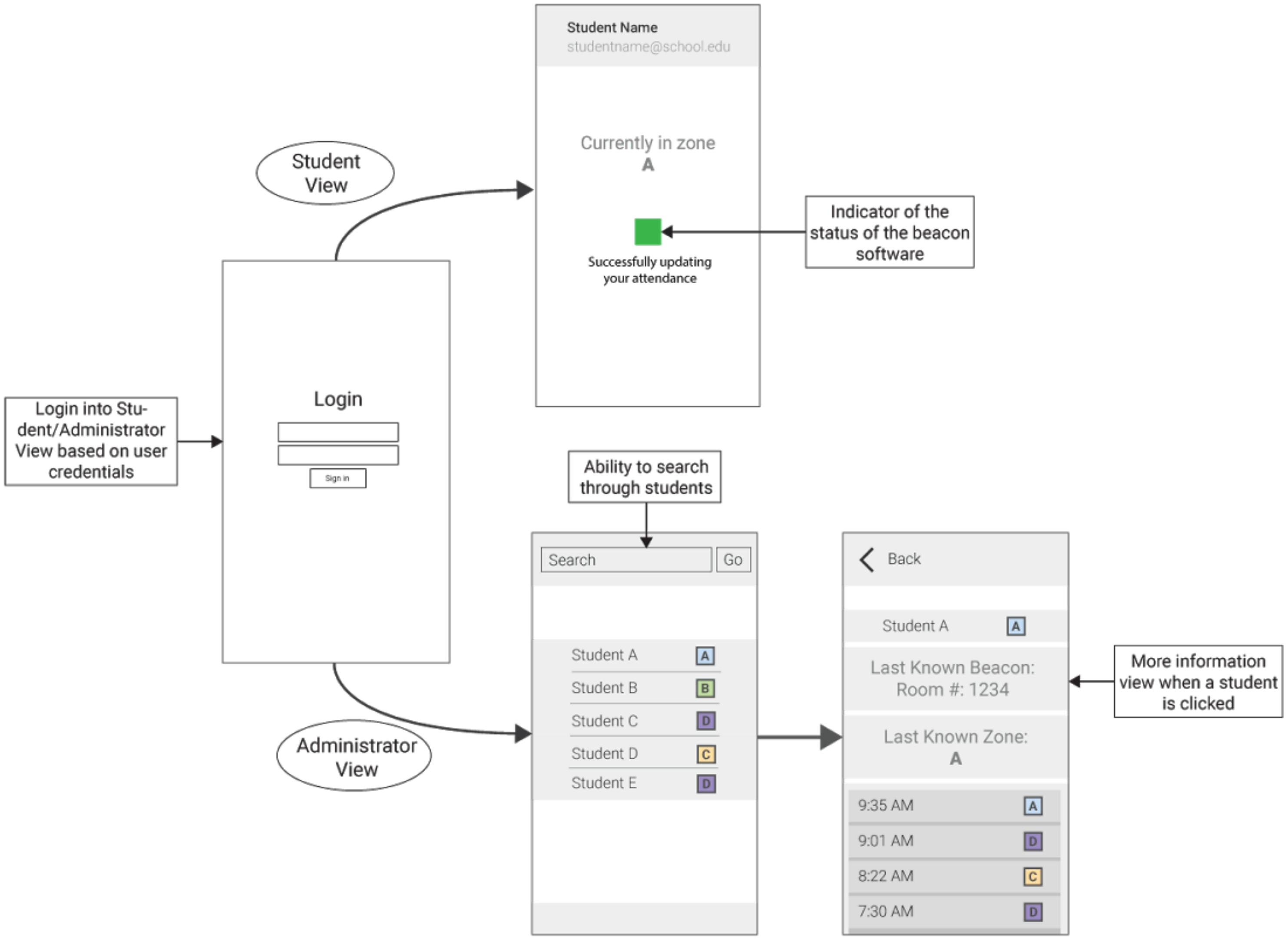
Figure 1: Prototype 1


Student Name
studentname@school.edu
Currently in zone
A
Successfully updating your attendance
Indicator of the status of the beacon software
Student View
Login
Sign in
Login into Student/Administrator View based on user credentials
Ability to search through students
Search
Go
Student A
Student B
Student C
Student D
Student E
Administrator View
Back
Last Known Beacon: Room #: 1234
Last Known Zone: A
9:35 AM
9:01 AM
8:22 AM
7:30 AM
More information view when a student is clicked

The first prototype was designed without the client's recommendation of a map view (that pinpointed the student's location) because it did not seem feasible to either the client or I. I found a way, however, to add a map view that would highlight the zone rather than the exact point making it more feasible, which resulted in the updated prototype in "Crit_B_Design" which has the map view.

Figure 2: Extensibility Considerations

Extensibility Factor	*Elaboration*
Ability to add students and administrators	Instead of limiting the number of students and administrators to the initial user base, I allowed for the addition of both students and administrators. Students will be creatable from the login page, and admins will be addable (by other admins) from a special form within the admin view.
Ability to add, edit, and delete beacons	To leave room to expand the system, instead of hardcoding beacons, I ensured that administrators would be able to add, edit and delete beacons. This allows administrators to change the position, add additional beacons, or remove unnecessary beacons.
Configuration file for database connections	The IP address of the database, as well as the login details for it, have the potential to change in the future. To remedy possible problems raised by this, I will create a custom configuration file written in JSON so that, in the case of database changes or migrations, hard coded variables must not be changed.
Postgresql Database	All persistent data, besides that in the configuration file, will be stored in the postgresql database. This means that the data can be used by other software in the future because it uses a standardized system.

Good Programming Style	*Elaboration*
Conventions for each language used.	Because I am using two different programming languages, it is important that I stick to the language specific conventions. For instance in Python, it is a convention to name methods with underscores: “foo_bar”. In Swift, camelcase is the convention: “fooBar”.
Encapsulation	To avoid coupling, all variables for which it is necessary will be encapsulated. This is most pertinent to Swift, which is based primarily on the object-oriented approach (as opposed to Python, which follows its own approach for the most part).
Commenting	Comments will detail the functionality of methods and lines of code across the middleware, the IOS application, and the MacOS application. This will be done in part by Swift specific commenting guidelines, and Python accepted commenting guidelines.