Stay in the Blue – Complete Spec

“Stay in the Blue” is a mobile application for iOS and android platforms. It’s main function is to allow a user to keep track of their BAC over the course of a drinking session, while also keeping track of the drinks the user has had, and allows the user to view their drinking history over the lifetime of the app.

In keeping with the Stay in the Blue brand, the app lets the user know whether they have stayed within Stay in the Blue’s recommended BAC range, represented by a color, ranging from blue to maize to red. This is the focal point of the app, and what separates it from similar, competing apps.

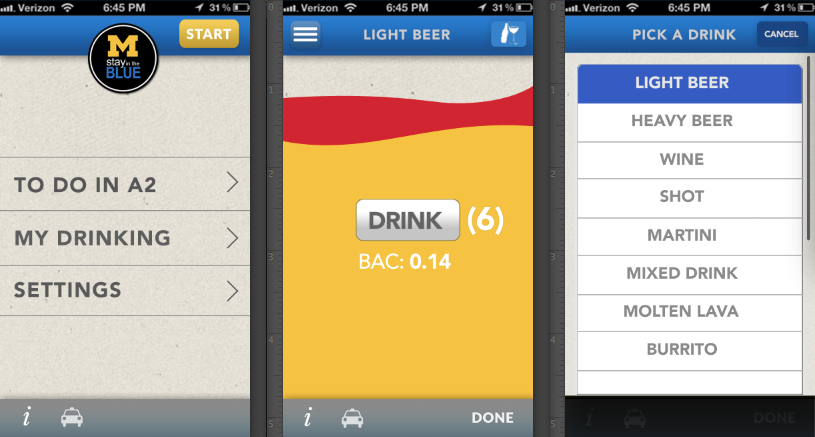
The standard color ranges are as follows:

BAC < .07 == blue

(BAC > .07 && < .19) == maize

BAC > .19 == red

When the user opens the app for the first time, they are directed to the settings screen, where they are prompted to enter their weight and gender, the two attributes associated with the user that are used in the BAC calculation. If they don’t enter any values, a UIAlert appears prompting them to do so. They are then directed to the home screen, where they have the option to start a new drinking session, go back to the settings screen, hit a “To Do in A2” button, (which links to a prebuilt webpage with a database of sober things to do in Ann Arbor), a “my drinking” button, which directs to a screen that shows the user’s overall drinking habits over the lifetime of the app(discussed later). On the bottom nav bar, the user can click a button for more educational information related to safe drinking, or press a button to be directed to a list of cabs, should the user require transportation. The main screen layout is as follows.



Upon pressing the start button, a new instance of a user object is created with the appropriate inputted data, and other values initialized to 0.

**User Class**

User is the only custom class used in Stay in the Blue. The User class has the following attributes.

* Gender (typedef (enum) m or f) sex //comes from Settings screen
* Int weight //comes from settings screen
* Int startTime //in seconds, used to calculate elapsed time
* Int elapsedTime //also in seconds, used to calculate BAC
* Double alcOz //used for BAC calculations. Each drink type has an associated amount of alcohol (stored in a plist file.)
* Double BAC //the current BAC of the user
* BOOL rageInProgress //used to determine if the user is currently drinking
* Double maxBacHolder // holds the highest BAC for the night.
* UIColor userColor //the current color level of the user based on the BAC
* NSString currentDrink //a string which holds drink the user is currently drinking
* NSDictionary overallStats // color level and number of times reached in a pair, used for “my drinking” graph
* NSDictionary overallDrinks //holds drink name and number of times consumed for use in main stats screen

The class also has the following methods:

* (NSString) getFilePath
  + Returns a string with the filepath for saving and loading information
* (void) saveData
  + saves user data to a the device
* (void) loadData
  + loads user data from a plist file
* (void) CalcBac
  + Calls calcTime
  + Calculates the users BAC using the equation:
    - (alcOz\*5.14/weight \*c) – (.015(elapsedtime\*.002)) where c is a constant that differs based on gender (.73 for male, .66 for female.)
  + Ensures that BAC does not dip below 0.
  + Updates maxBAC
  + Saves user data
  + Calls update color
* (void) calcTime
  + sets elapsed time to referencedate – startTime
* (void) updateColor
  + updates userColor based on the BAC:
    - < .07 == blue
    - >.07 && < .19 == maize
    - > .19 == red
* (id) init //constructor
  + sets startTime to referencedate
  + elapsedTime to 0
  + alcOz to 0
  + rageInProgress to NO
  + BAC to 0.0
  + userColor to blue
  + bacHolder to 0.0

**PROGRAM FLOW:**

When the application begins for the first time, the settings screen appears, and creates a user object, and waits for the input from the user. Once the user has inputted their info, he can hit a “close” button and be taken to the main page. From that main page they can always go back to the settings page, though they shouldn’t need to. After the user has opened the app for the first time, they are taken directly to the home page.

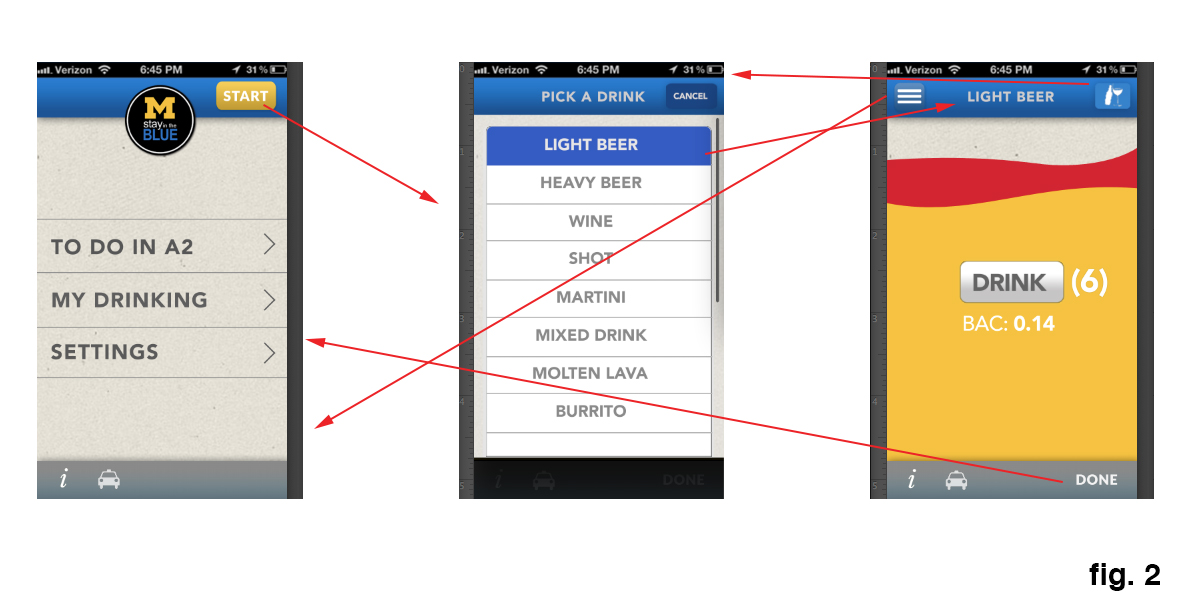
When the user presses the start button at the top of the screen, the user object is passed to the DrinkingScreenViewController object, which also has a User object as an attribute.

**Drinking Screen:**

The weight and gender are passed to the User object, and all night-specific attributes set to their default values (startTime, elapsedTime, alcOz, BAC, userColor, bacHolder). rageInProgress is set to YES, currentDrink is set to nil, and a timer object is created, but its attributes are not set yet.

When the pre-drinking setup is complete, the view controller loads the Drink selection screen, where the user inputs the type of drink they are drinking first. The drink type will correspond to a double value, which represents the amount of alcohol in ounces in that drink. That number is then added to alcOz, calcBac is called on the user object, the labels for BAC, the number of drinks, and the current drink are updated. This is also where the timer object is allocated and started with an interval of 10 seconds, set to call a method which simply updates the BAC and the BAC label.

The philosophy is that the user will already be drinking when they start a session, and thus will want to choose a drink right away instead of hitting a button to choose a drink and then starting the session. The drink the user is currently drinking is displayed on the navigation bar, and directly to the left there is a button which brings back up the drinking screen, allowing the user to change their drink. Again, the thinking is that most people will usually have a few beers, or a mixed drink or two, over the course of the night, as opposed to a variety of different drinks, thus if the user drinks another of the same drink type, they need only hit “drink” again to add that drink to their stats, as opposed to forcing them to select the drink from a list over and over.

In the upper left hand corner there is also a back button, which allows the user to go back to the homepage and view any of the pages on that screen. When rageInProgress == YES, the start button in the upper right hand corner will have changed to “continue,” and the user will be taken back to the drinking screen.

Additionally, they can press the small ‘i’ button at the bottom of the drinking screen or the home screen to be taken to a Stay in the Blue web page which will have information on safe drinking.

The cab icon on the main page and drinking screen will take them to a list of cab companies in ann arbor in a list, where the user can select one and make a call directly from the app.

There are several things that need to happen at key intervals during a drinking session to ensure accurate and sustainable program flow:

* When the user presses the back or the drink button in the drinking screen, all data must be saved in case the app gets terminated or the phone restarted.
* The user must enter a value for their gender and weight on the settings screen, or else should be prompted with a UIAlert to do so, and not allowed to continue to the home page until they have done so.
* When the user goes back to the home page, if rageInProgress == true, the start button must say “continue.”
* The BAC must not dip below 0.
* When the drink button is pressed, the maxBAC value must be updated, and the current drink added to the overallDrinks dictionary. If they have selected a drink different than their current drink, that drink must be assigned to currentDrink.

When the user is done drinking, they press the done button in the lower left hand corner. This will prompt a UIActionSheet to appear, which will ask the user if they want to save the data from the session, or delete it. The thinking behind this is that users will likely play with the app in non-drinking scenarios, and thus don’t want this to skew their overall drinking data.

If the user says no, the data is not added to the overallStats dictionary, and the user is taken back to the home page. If they select yes, the data is saved, the user object passed to the StatsScreenViewController and the user is taken to a screen with his/her stats for the night, which is simply the number of drinks the user had, their max BAC for the night, and the color level they got to. The latter two are derived from the maxBAC value, and the former from the numDrinks attribute. When the user is done viewing, they simply hit done and are taken back to the home screen

**My Drinking:**

When the user clicks the ‘my drinking’ button, they are taken to a page which displays a graph of how many times the user has entered the red, the maize, or stayed in the blue. At the bottom of the screen, the user also will see their top three most commonly drank drinks, which is derived from the overallDrinks dictionary.

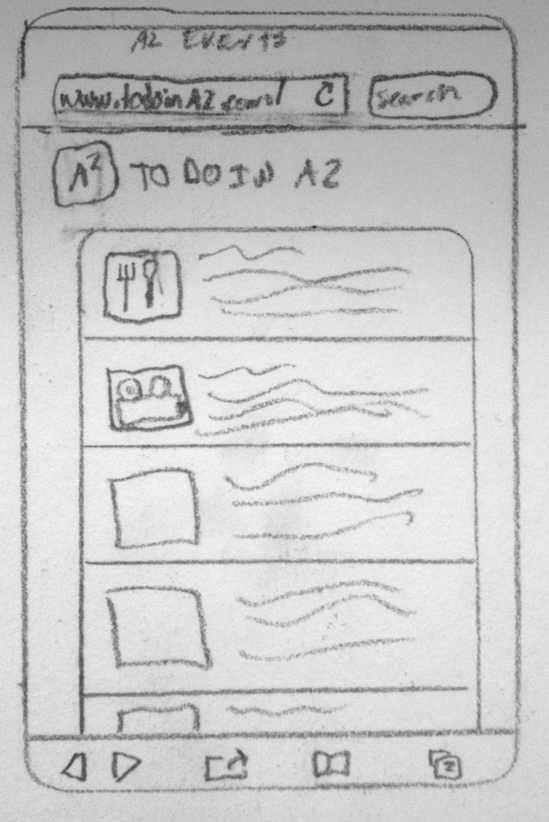
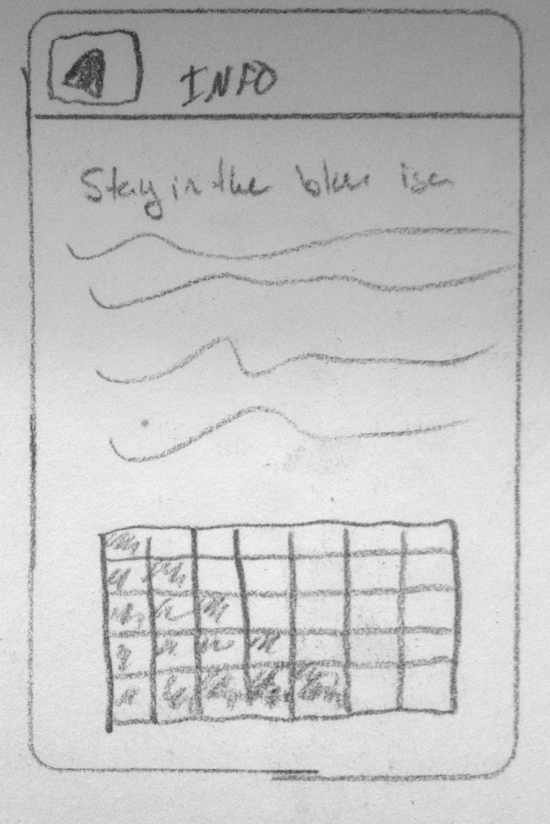
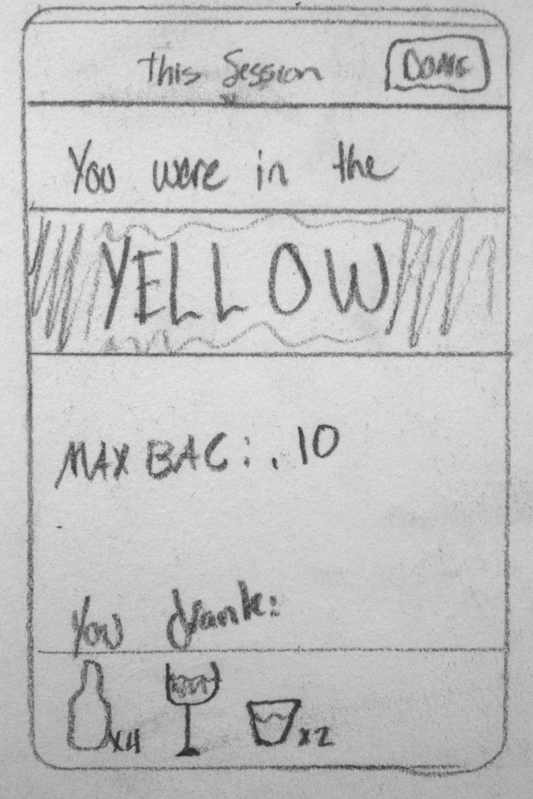
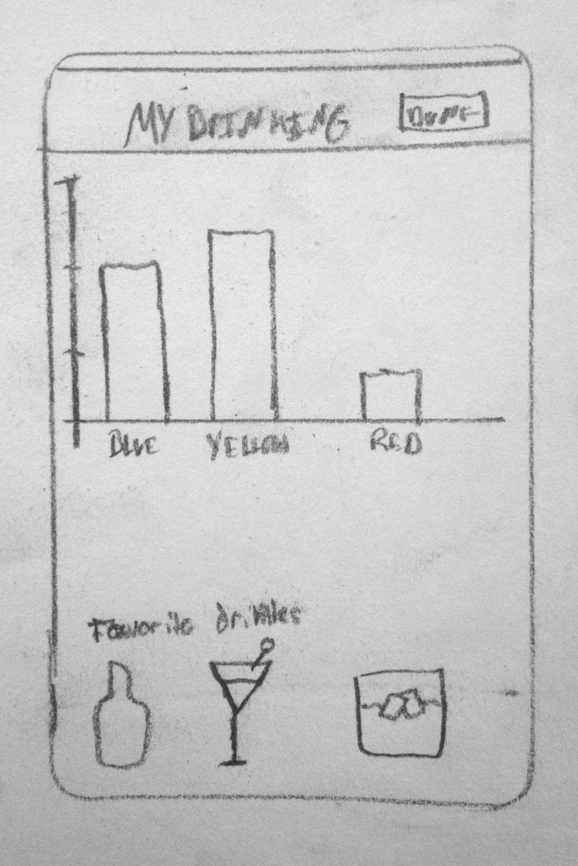
Thus when the view loads, the data in overallStats will give the data for the graph. (E.g {red: 4, maize:10, blue:20})

**Mockup Screens:**

Screens not shown above are denoted here:

Stay in the blue education

Things to do in A2

­­­

My Drinking

StatsScreen