
**Bronwyn E.** | Manager

**Bernardo V.** | Designer

**Clark P.** | Developer

**Hilary S.** | User Testing/Documentation

---

## Problem & Solution Overview

Joynus is a mobile and location-based way for users to connect through spontaneous events. Users can create, search for, and join events around them. Based on our need-finding, we found that people long to connect with strangers but find it hard to take the first step. Our mission is to provide a service that connects nearby strangers through shared experiences and events.

## Tasks & Final Interface Scenarios

*[Simple] Task #1: Customizing the user profile by adding a category of interest*

Adding a category to the user's interests lets them receive a push notification every time an event nearby is created under that category. We chose this task to provide more user flexibility and customization.

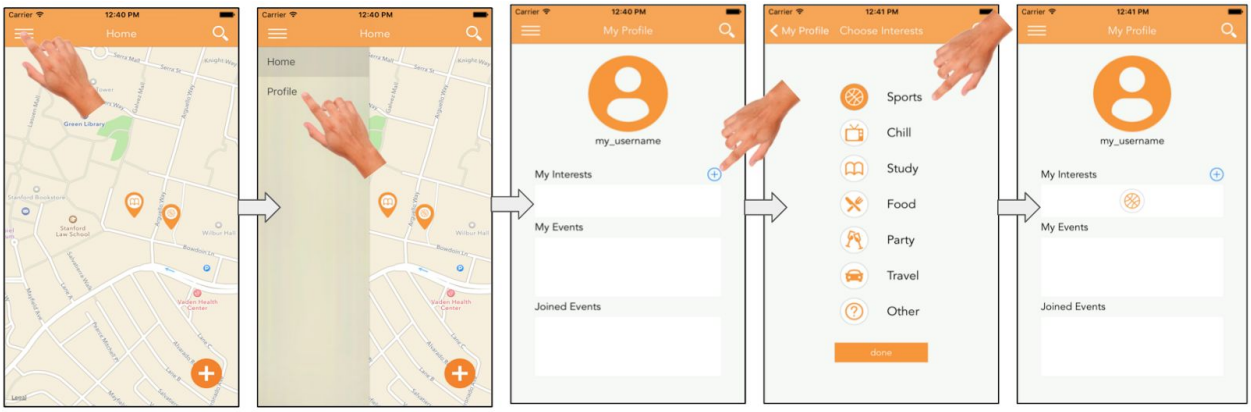


Figure 1: Storyboard of adding an interest to profile.

The task flow is shown in Figure 1. The user starts off at the home screen, opens the side drawer menu, and clicks on “Profile.” Clicking on the “+” next to “My Interests” opens a list of interests that the user can select to receive push notifications for. Clicking “Done” will add the interest icon to the “Interests” box.

*[Medium] Task #2: Finding and joining an event*

Joining an event is one of the key functionalities of our application and allows the user to engage in events created by other users.

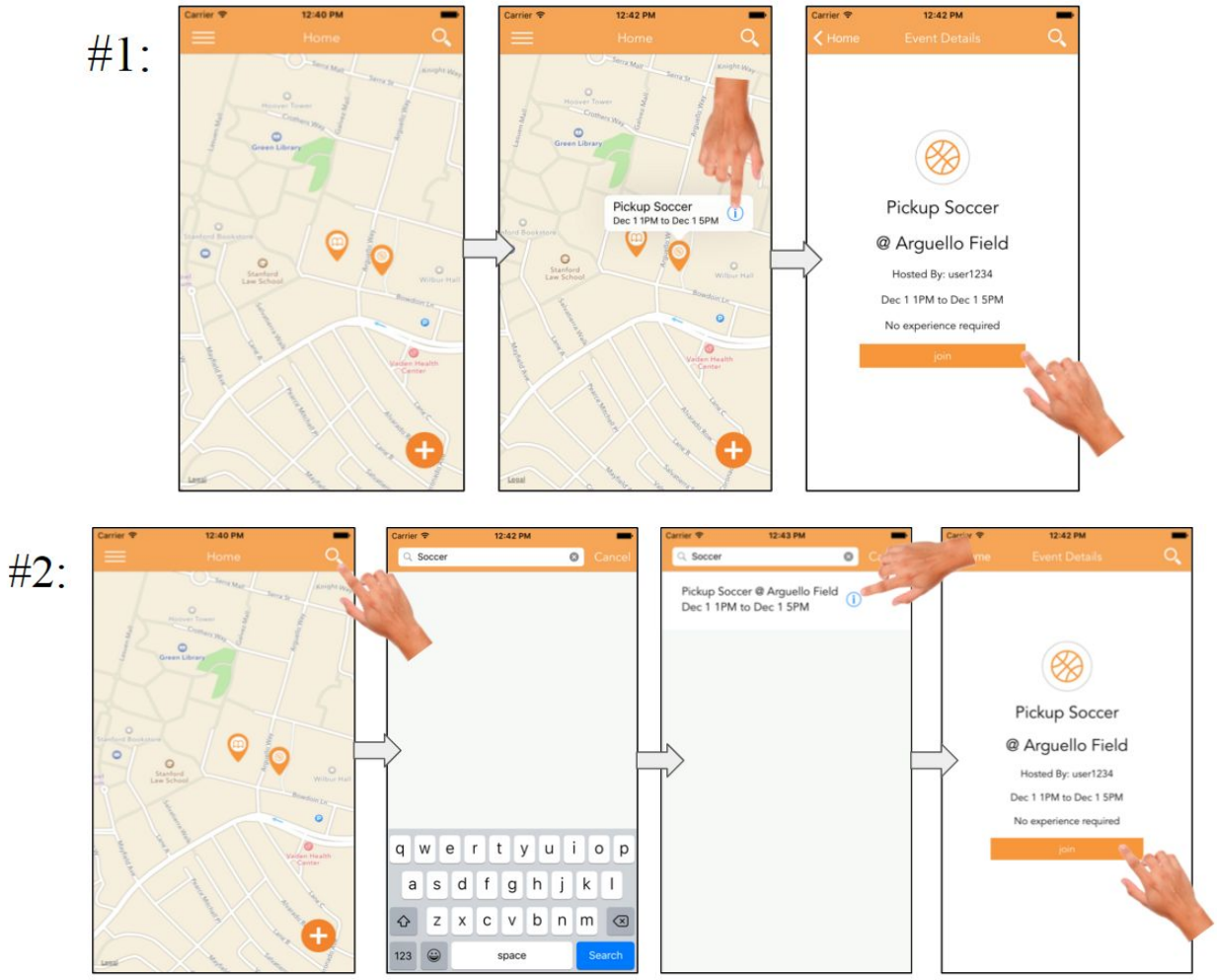


Figure 2: Two storyboards for finding and joining an event.

There are two ways of finding and joining an event. The first way is to start off at the home screen, scroll through the map, find an event of interest, and clicking “Join.” The second way is to click on the search icon on the top bar and typing in a search query, for example “Soccer.” A list of matching events is shown, and the user can click for more information and then click “Join” on the details page.

*[Complex] Task #3: Creating an event*

Creating an event is the other key functionality of our application and allows the users to host his or her own events.

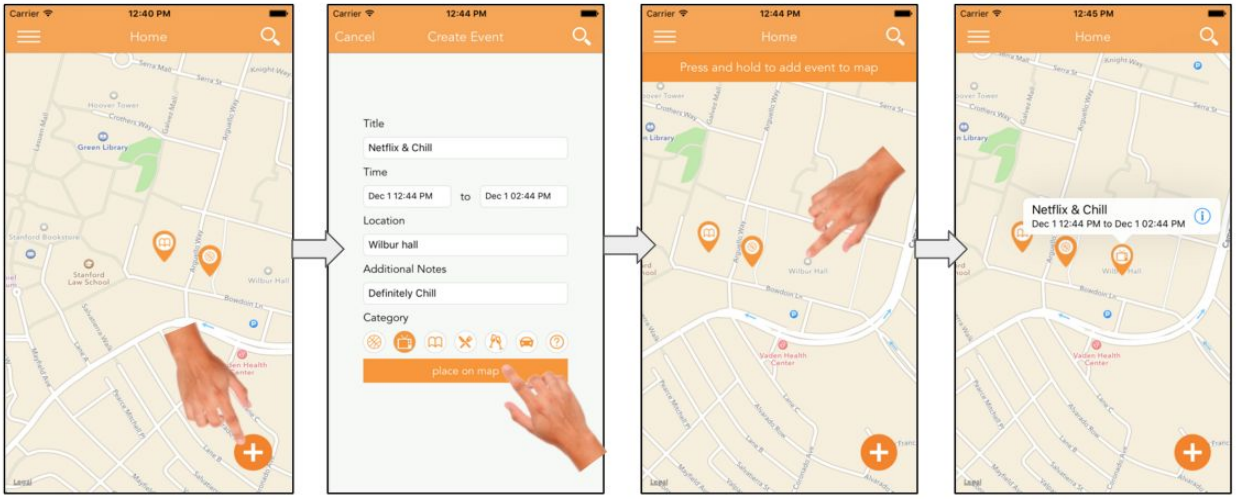


Figure 3: Storyboard of creating an event

The task flow is shown in Figure 3. Click on the "+" button at the bottom right corner of the home screen will bring up a form. Clicking on "place on map" will bring the user to a map where they can press and hold to drop a pin where their new event will be.

## Design Evolution

Our initial sketches and storyboards included different interface options on a variety of mediums, such as a phone or the web. To display events, we brainstormed the current map-based display of events as well as a bulletin-board style of event postings, as shown in the two images in the top left of Figure 4.

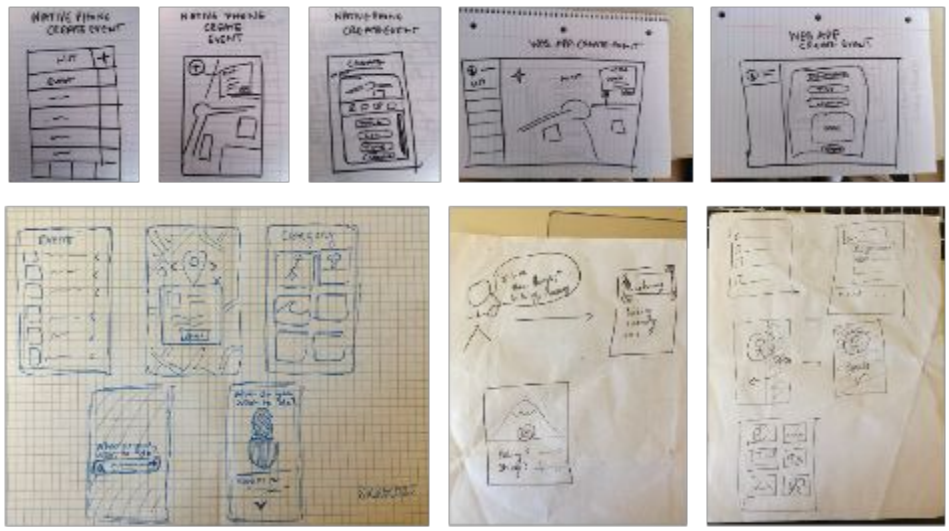

Figure 4: Concept sketches explored a variety of interface options on a variety of mediums, from event display, to event search, to navigation.

We also had different ideas about how to display navigation, including a bottom bar and a side drawer. Ideas for searching for events included basic text input search, a voice-prompted search, and a questionnaire. Eventually, we decided we wanted a visual approach that promoted our location-based ideal and used the map as the home screen, shown in Figure 5. We also kept the list-based way of displaying events because of familiarity as an option. We also decided to use

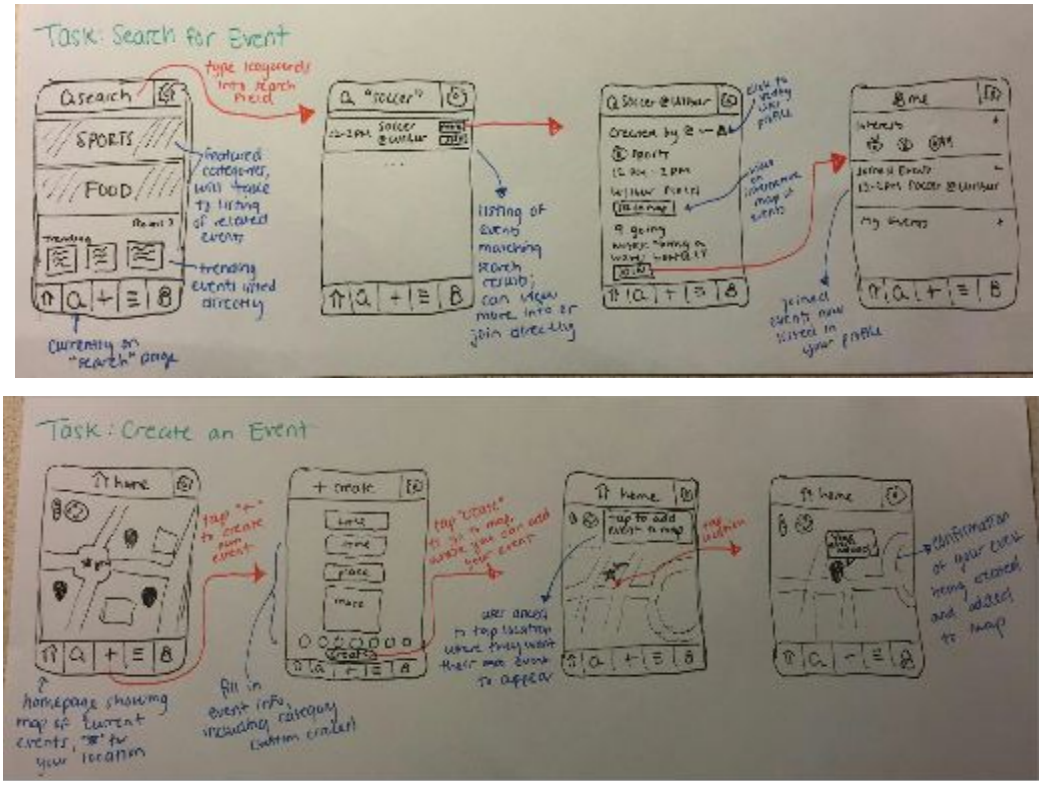


Figure 5: UI storyboards for searching for and joining an event (top) and creating an event (bottom). Shows the map and list implementation of displaying events.

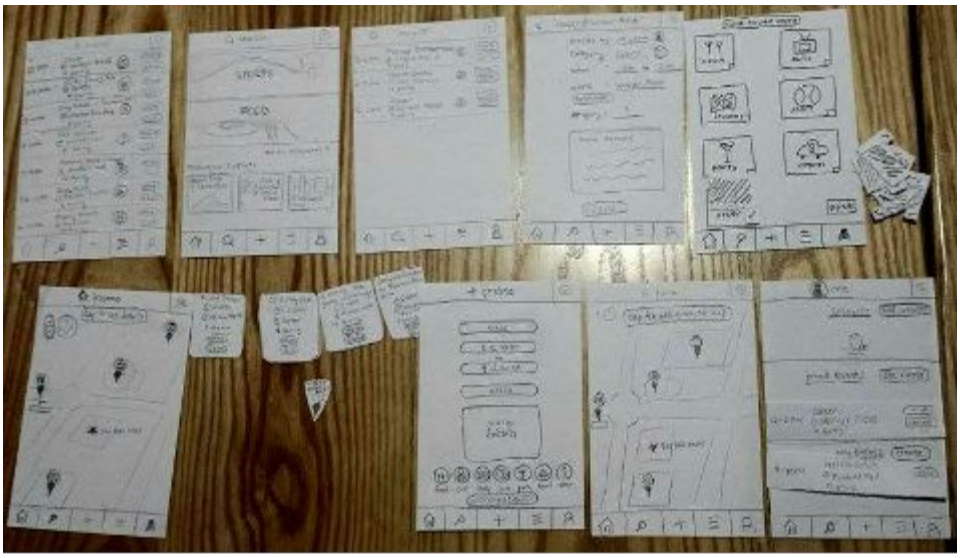

Figure 6: Our low-fidelity prototype, in its entirety.

categories to categorize our events.

Our low-fidelity prototype, shown in Figure 6, was defined by a bottom navigation bar, a map home screen with pins to indicate events, an option to view nearby events

in a list, creating an event by filling in details and then adding it to the map, a browse page with a search bar and featured categories and events, and a user profile with customizable interests. We decided to make it a mobile app for better accessibility. These features made our application more visual and simple to use.

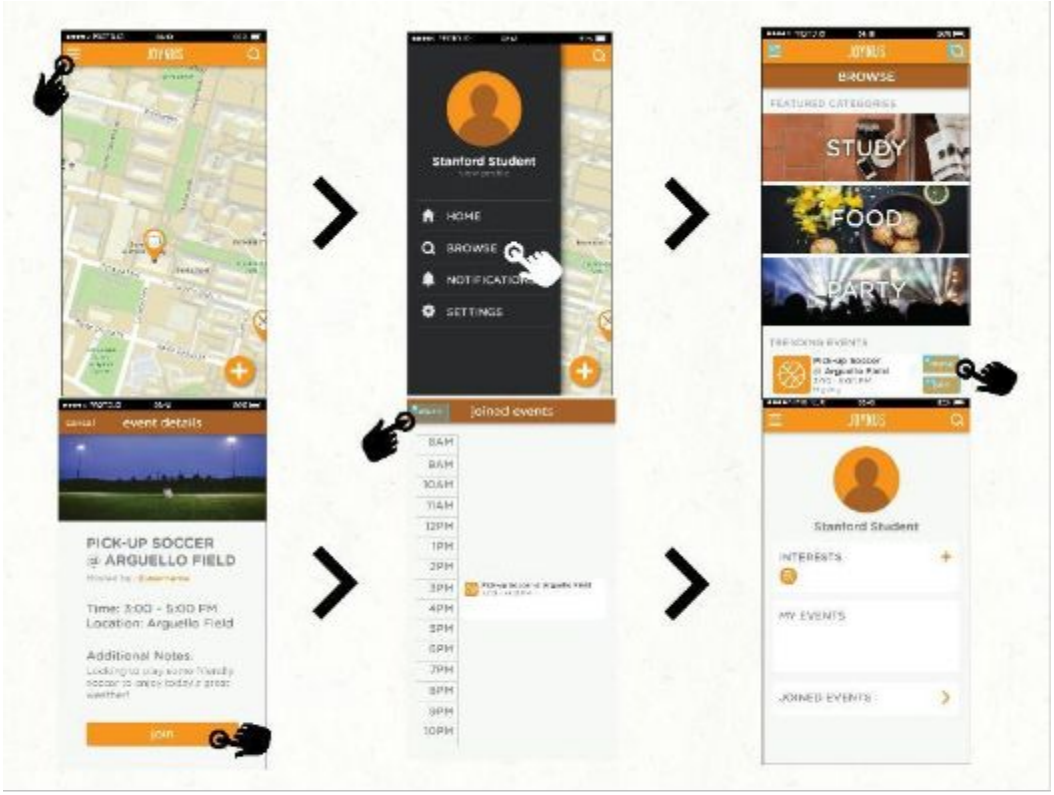


Figure 6: Storyboard of our medium-fi prototype to find and join an event.

After our low-fidelity user testing, we found that all the users were able to complete the tasks successfully and quickly. However, we found that our search icon was small and that the prompt to “tap on the map” to add a pin to the map after filling out the details confused users.

To fix these problems, we decided to change the navigation bar at the bottom to a side drawer with search at the top right and a “+” at the bottom right to add an event. There would be less clutter, keeping only the most frequently used buttons on the screen. We also needed to add more options: “Notifications” and “Settings.” Rather than having a list of joined events and my events on the profile, we decided to use a separate calendar list to help the user to form an intuitive itinerary. To clear up the confusion caused by the create form, we changed the button to say “place on map”. Figure 6 shows the calendar list change in our storyboard of task #2.

## Major Usability Problems Addressed

The following are the severity 3 and 4 violations from our heuristic evaluation.

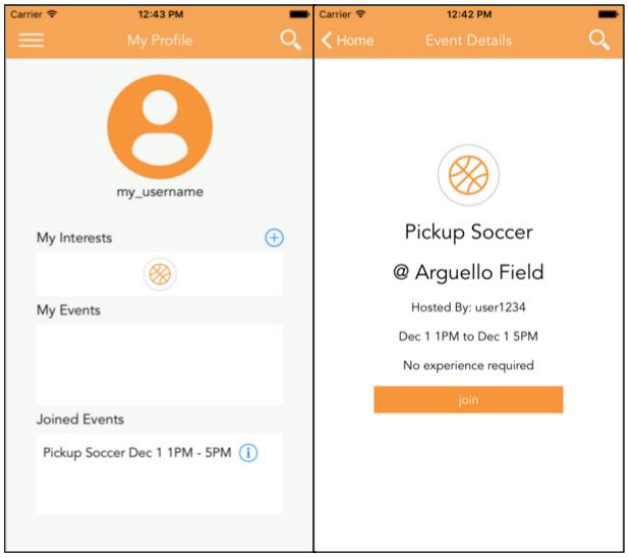


Figure 7: Consistency between event details and event display on the user profile.

*Violation #1: The event time displayed in the user profile under his/her joined events does not match that of the actual event time.*
This violation was a mistake on our medium-fidelity prototype where we accidentally wrote in inconsistent times for the same event. We addressed the problem by ensuring that the times were the same when displayed in the profile, as shown in Figure 7.

*Violation #2: There is no date displayed on any of the events, and there is not an option while creating an event to add a date.*
To stay true to the spontaneous experience users get through Joynus, we created a time frame where events would be restricted to a 48-hour time window from the current time. We changed the settings of the datepicker so that if the user scrolls before the current time, it automatically locks back to the current time, and if they scroll past the 48-hour window it locks back to the last possible time. There are limitations to this locking method; however, we could not figure out to change it so that the only date and times displayed were the relevant ones. The change is shown in Figure 8.

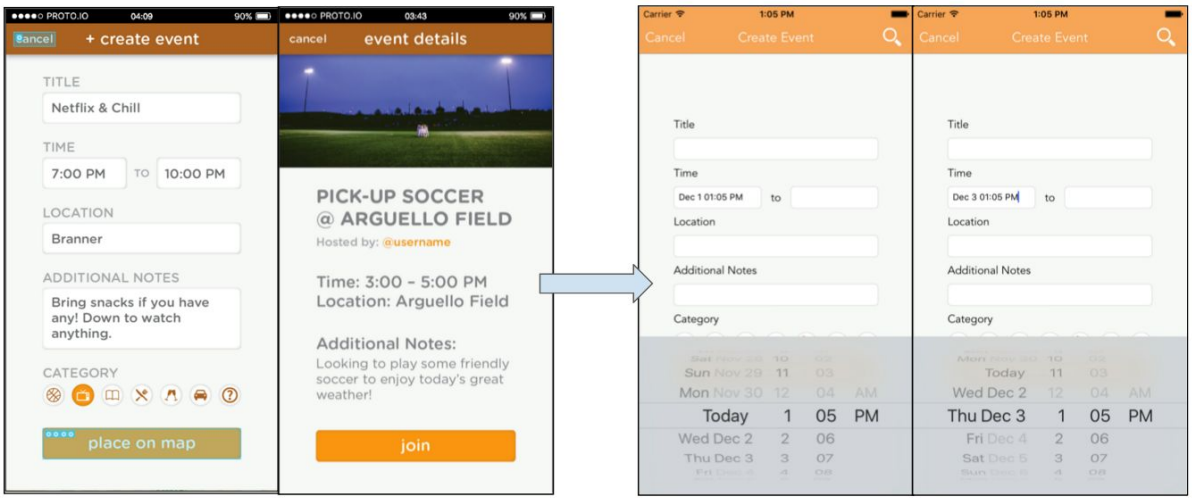


Figure 8: The change from not showing date but not having a time frame where users create events within (on the left) to having a locking limitation when creating events that limits the user to 48 hours from the current time.

*Violation #3: There is no error-checking to see whether an event you're joining overlaps with another event you are already signed up for.*

We chose not to address this violation because we wanted users to be able to sign up for multiple events where they could maybe go to half of one and half of the other. This allows more flexibility for the user. We also switched out of the calendar format back to a list format, shown in the first screenshot of Figure 7 of the user profile, from our medium-fidelity prototype because the events are only within a 48-hour time frame and we felt the extra interface would detract from the simplicity of our app.

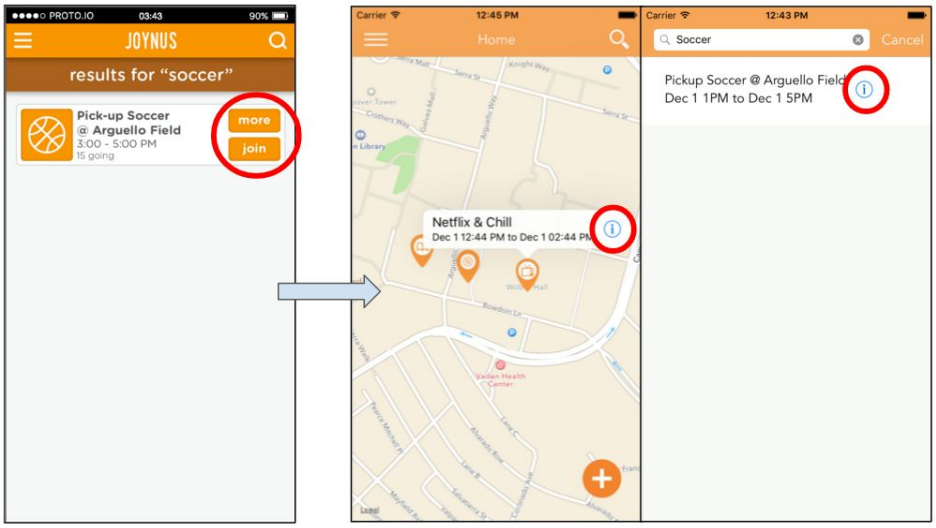


Figure 9: The change from "more" and "join" buttons (left) to blue info icons (right), highlighted by the red circles.

Other than these, we also addressed a few other smaller violations. For example, we addressed consistency of text case by making the stylistic decision to still use both lowercase and regular capitalization, but changed the text in all caps to normal capitalization (i.e. first letter of each word capitalized for titles and first letter of each sentence capitalized for body text). Also seen in Figure 9, we decided to take out the "join" and "more" buttons next to each event and replace it with the recognizable blue information icon to reduce clutter on the screen. Based on the heuristic evaluation as well as our own thoughts, we felt that the "more" and "join" buttons were small for a mobile app and it would be easy to accidentally click the wrong one.

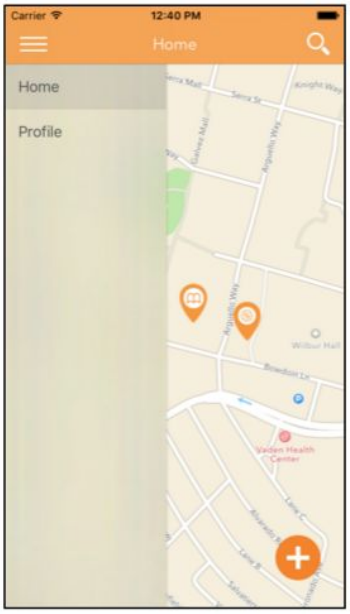


Figure 10: Side drawer for high-fidelity prototype.

The last major change we made is seen in the side drawer in Figure 10. Firstly, we took out the profile picture display in the drawer because the focus of the application isn't on developing a user profile; it's focused on the events. Putting the profile picture and then having a very small "view profile," as shown in the first screenshot of Figure 11 below, distracts the user from simply doing what they wanted to do: go to their profile. Secondly, we chose not

to implement the other two options in the side menu, “Notifications” and “Settings” because of irrelevance to our tasks and time constraints. We also chose to take out “Browse,” shown in Figure 1, because there aren’t too many categories for the user to go through, so having a “Featured Category” didn’t make that much sense and because of time constraints.

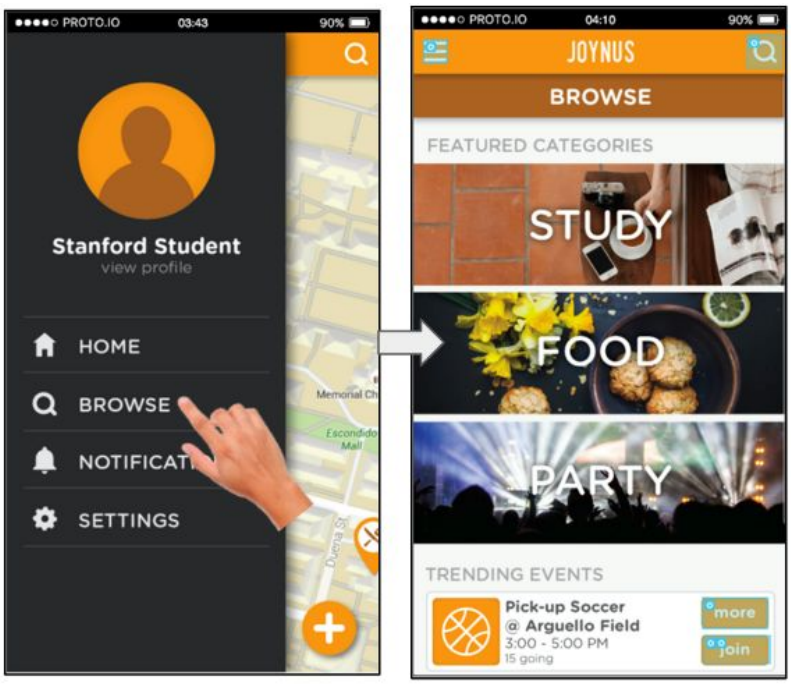


Figure 11: Side drawer for medium-fidelity prototype; clicking on “Browse” would lead the user to a page with featured categories and events.

## Prototype Implementation

For our implementation we created a native iOS application written in Swift within Xcode. Xcode’s Interface Builder (IB) tool, a visual editor to help layout the front end of the application, helped tremendously in the design process. We used IB to plot out relationships between scenes and view controllers within the app, then linked the visual design to the actual code by clicking and dragging to outlets. Although IB was very helpful, at times it was confusing and buggy because carrying out a process in one order would succeed while a different order could lead to disastrous results such as a crash.

Every part of the app was written from scratch except the side drawer navigation, which utilizes an open source library. The open source drawer library helped immensely by providing a simple API to minimize the amount of code written on our part. However, the library had one drawback

in that it didn't support changing the highlighted cell from outside the internal workings. In order to work around this, we had to restrict certain navigation to specific paths.

We used Adobe Illustrator and Flaticon to create the icons for the app, heavily based on the medium-fidelity prototype's aesthetic. These tools in conjunction helped streamline icon design by minimizes the actual amount of illustration done. Most of the work was in coloring and positioning premade vector graphics.

Our app makes heavy use of Wizard of Oz techniques and hard coding. Due to time constraints, we decided to not implement a backend and simply simulated a backend by using local storage and hard coding. When the user does anything in the app that seems like it is calling a server, the app is actually just checking whether or not certain flags in local storage have been set, and updates the interface based on this. For example, when the user creates an event, they can visit their profile and see that that event is now displayed under My Events. The Wizard of Oz technique here is to check if the eventCreated flag has been set, rather than communicating with a server. The two events that appear on the map initially are hard coded, including their detail pages. Functionality such as searching and choosing interests is hard coded in line with the tasks we chose. This means that the search result will always be Pickup Soccer no matter what, and the chosen interest will always be sports.

Obviously the app lacks backend communication. If we were to continue this app in the future, we would add this by communicating with PHP scripts that communicate with a MySQL server. The app also has very primitive to no form validation. Most functionality outside of achieving our three tasks is restricted or impossible because of the hard coded nature of our app. With the MySQL server, we could store more user information, more event information, and category information to be able to implement a "Browse."

## Summary

Joynus is a response to the various concurring responses we received from interviews that they craved new, organic, and spontaneous connections. Because of time limitations, we were not able to fully implement the application, but we believe that my high-fidelity prototype thoroughly displays the experience we want users to have. Throughout our design process, we worked to provide an experience centered around creating and finding/joining spontaneous events, minimizing and simplifying the process to create a hassle-free medium for people to share experiences that matter to them.