Overall good job. Comments inside.

G 4
F 6 - simply more detail needed
C 5 - again, more detail if needed
U 4
I 4
S 4
A 4
P 5

36

Michael Jannino

Corey Prak

91.462 GUI Programming II

Professor Jesse Heines

February 10th, 2015

Table of Contents

Project Goal.....3

Major Features/Capabilities.....3

Software Components.....6

User Base.....7

Possible Issues.....7

Preliminary Schedule.....8

Acceptance Criteria.....10

Nice page numbers here, but your pages aren't numbered!

Project Goal

People around the world today are connected in more ways than ever before. We can contact friends on cell phones via talk or text, email, through social media services, and the list goes on. This new facet of social interaction has many benefits, but has also brought with it a long list of problems: plans must be made with clear and direct contact through text or social media with each individual, and word of mouth plans are more scarce than they ever have been.

Squadly is a service that aims to make the most of socializing and to take out the social issues that come with communicating through text or social media. Whether you're planning a gathering or making a casual stop at your local watering hole, Squadly takes the effort out of inviting and organizing a gathering and allows more free-form and welcoming plans to be made.

Major Features/Capabilities

Squadly will start out as a standalone web-based application. Eventually, Squadly will operate as a website and mobile phone application, co-operating and communicating with a shared database infrastructure.

(figure 1)

Squadly's user experience will be compact and directed. The landing page will be sparse and focused on engaging the user to sign in or sign up with an explanation of the product below.

Potentially, we wish to add the functionality that if the user has a session saved and is already logged in they will skip this screen.

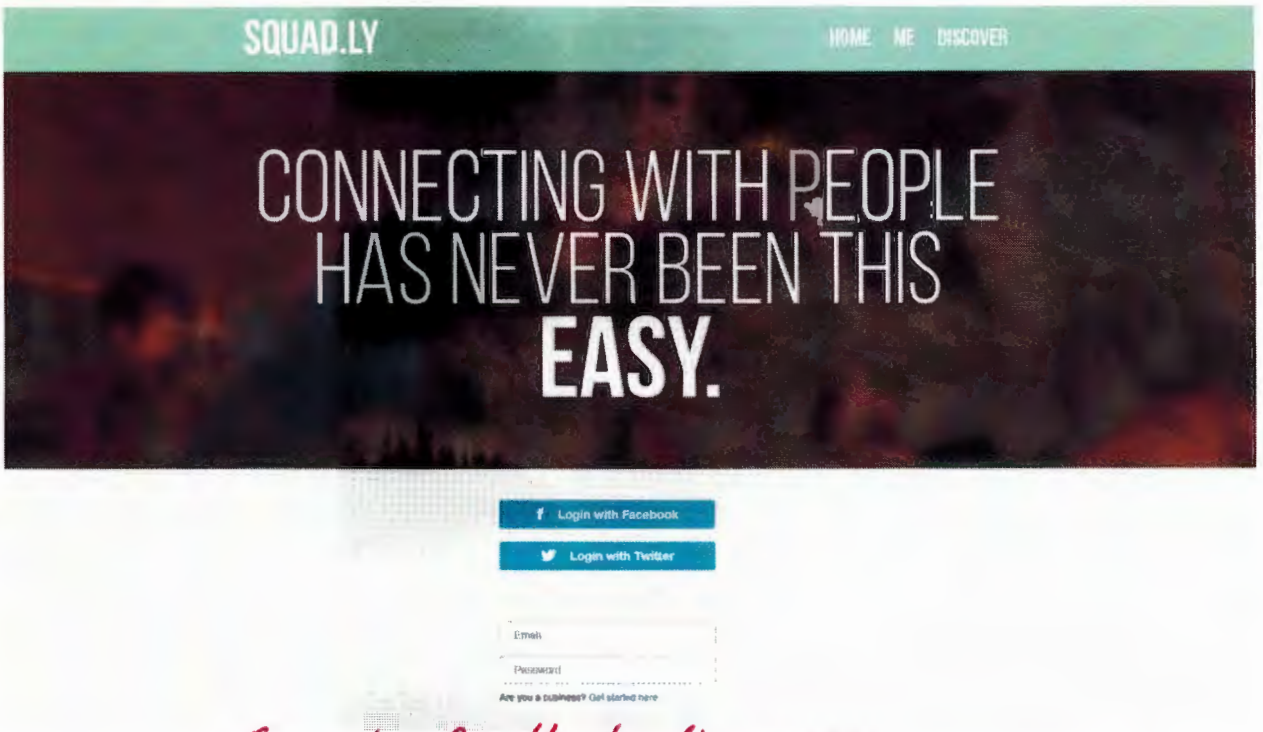

figure 1. Squadly landing page.

Users will then be ushered to a map and list of businesses near them. (see figure 2). All of these places will be "taggable" for event creation. Users will also be able to see a feed of any events they have been invited to. Potentially, there will be a notification center for any interaction with other users.

All figures must be numbered and referred to in the text by their numbers. You can't just drop them in and expect readers to be able to make the connections. Each figure must also be labeled as I have shown.

Figure 2. Map of nearby businesses.

The biggest feature of Squadly that sets it apart from other social network-esque platforms is the broadcasting function, the "Squad Up" function. After deciding on a time and place for plans or events, users can click a location or the map to bring up a Squad Up interface. In this interface, they will choose which "Squad," or circle, they want to broadcast the event to. Options will include suggested circles, user-made circles, or all users associated with the event creator. This mixes the functionality of Google Circles and Facebook Events while providing a real-time environment to communicate any changes in.

Software Components

While Squadly will use a combination of separate technologies for both front and back end, we are considering using a web framework to make development easier. A web framework such as Ruby on Rails will make development between the front and back end almost seamless. Furthermore, the framework will enforce conventions and common practices that members of the team won't have to spend valuable time figuring out.

While there are APIs that will provide Squadly with the capability of interacting with a user's profiles across different social services, there is no central "brain" that exists to compute the interaction between them. In addition to forming the front and back end, this "brain" is a component that we will have to create. By creating this system, a notification system and cross-platform interaction will be possible, using the APIs for strict interaction while localizing computation (distance, etc) seems to be the best approach.

We plan to use Facebook, Twitter, Twilio, Google+, and Google Maps APIs for maximum interaction and functionality. The first four will be used for account creation and exposure, while the Google Maps API will be used to create the environment for users to create events. The backend has yet to be decided, but may be based on PHP, Mongo DB, or Node.js, among others.

parts of these
You need to identify the APIs you're talking about here. They are all very big APIs, with many parts.

User Base

Red flag word! Do not use in user descriptions.

Squadly is a service that is intended for anyone who is interested in a convenient way of organizing a means of communication of a group of people. The main demographic that fits this description will most likely be young adults from 18 up to 30 years old, to meet up for social gatherings, parties, and nightly activities. College aged students will be the specifically targeted demographic, as they will get the most use out of this service.

Possible Issues

To prevent abuse, vendors who provide APIs for their services have limitations on usage. Additionally, we need to find out whether the APIs can provide the functionalities that Squadly needs. Although the basic abilities of the APIs are known, feasibility tests will be done to discover the extent of what can be manipulated.

Security is possibly the biggest issue. Neither member of this team has extensive knowledge of network security or even basic practices of how to prevent exposure of sensitive application data. This single factor makes using Squadly risky, although there are secure connection packages that exist that we can look into.

I believe that there are other issues related to the development, too.

Preliminary Schedule

1. Getting a basic page up and running

We will most likely host Squadly on a Virtual Private Server. A domain name is also under consideration. Since Corey has experience with both, he will be completing this task. Mike will work on the actual design and presentation of the initial page. This will be done before February 18th, 2015.

2. Functional backend

After the VPS is setup and the backend technology is decided upon, we would need to think about its basic architecture and ensure that saving sample data is possible. Once the APIs can be used as desired, the backend will aggregate all related activity. Corey will also be handling this task with Mike, performing most of the technology selection while Mike helps with implementation. This will be done before February 28th, 2015.

3. Functional frontend

Once we have a better idea of design and technology selection, a functional prototype frontend design will be implemented. Mike will handle the brunt of this task with input and assistance with Corey for future-proofing the design to work with the technologies we decide upon. This will be done alongside the functional backend, and will also be done before February 28th, 2015.

4. Exploring APIs and implementing functionality

Separately from Squadly, the APIs need to be explored, learned, and tested for feasibility. Since this step dictates Squadly's overall capabilities, this milestone will most likely take the second most time and effort. Both team members Mike and Corey will work together to complete this task, finishing at least a partial implementation of all technologies before March 19th, 2015.

5. Developing a central feed to aggregate all of the necessary information; polishing.

At this point, both the backend and the necessary APIs are working as desired. The next step is to incorporate the information obtained from the user and APIs into the backend. This step is expected to take the most time and effort. Polishing includes user testing with a beta version of the site that will be completed by March 19th; beta testing will be completed before April 2nd.

6. Final product development

This will include finishing, polishing, and removal of bugs. This will start after April 2nd and finish by May 1st with the final project being completed.

OK. Reasonable list.

Acceptance Criteria

Minimum Functionality

Squadly should be able to save basic user information, such as previous 'sessions' and related content. This would prove that the backend is functional. It should also be able to use the Google Maps API to convey event information. Finally, Squadly should have the functionality of having a central news feed.

More specifically, the user should be able to navigate to the application and start Squadly by specifying the location of an event. If location information is entered, the Google Maps API should display the location of the event. After additional event info is given, the user will use Squadly to access social services that he/she would like to advertise the event to. The APIs that are expected to work are Facebook and Twitter. Once selecting the contacts, the user should select the "Squad Up" function. This will prompt Squadly to advertise the event through the selected social services. Any activity related to the "session," such as a reply, tweet, or message, will be tracked and appended to a universal feed that Squadly is responsible for computing. Once the user finally shuts down or "stops" the session, the event is considered complete. This process proves that Squadly passes the minimum functionality tests.

OK. Much better than original draft.

Break page here

Additional Features

If resources allowed, Squadly would have the ability to convey multiple events for a 'session' at a location rather than a single event. This would increase the Squadly's usefulness to multi-day events and schedules. The effort into realizing this would most likely be design involved since there wouldn't be a need to extend functionality.

While Squadly is intended to be used on multiple platforms, a mobile friendly version would be great for cellular devices and tablets.

"Reach" Features

A few features that may significantly increase the application's usefulness:

- A native mobile application would improve Squadly's interaction with users who are on the go. Contacts could be uploaded to Squadly and notifications could be pushed from the application instead of requiring the user to refresh a web page.

- Ensuring the protection of sensitive user credentials and 'session' information would require extensive knowledge in web security. This is an aspect of Squadly that we would unfortunately not have time to put effort into realizing.

- While this project is for academic purposes, being able to improve security would make Squadly "real world" usable.