InTouch: Utilizing Technology and Down Time to Better Maintain Relationships

Naicheng Wangyu
Yale University
CPSC 490b
Advisor: Avi Silberschatz

March 25, 2014

Abstract

“InTouch” is an application that helps people organize and maintain communication with their close personal and professional contacts. Oxford anthropologist Robin Dunbar claims that 150 is the "suggested cognitive limit to the number of people with whom one can maintain stable social relationships". This app attempts to avoid what I believe is a pitfall of many of these large social media sites (i.e. LinkedIn, Facebook...etc.): Users have more "friends" or "connections" than they can realistically maintain authentic relationships with, and this overabundance prevents them from clearly identifying their close relationships.

Instead of pulling hundreds, if not thousands of contacts from these platforms, this app allows users to identify and insert their close contacts’ details to the application and set desired contact frequencies. The uniqueness of this app is its ability to create a dynamic and prioritized “stack” of these inserted contacts that can be accessed at one’s convenience. That is, instead of focusing on creating a “notification” or “reminder” based system, this program allows users to maximize random downtimes during their day by staying in touch with the contacts that InTouch recommends.
Motivation

As outlined in my proposal, I wanted to achieve the following goals through my senior project: 1) Learn and build a full program with a UI. 2) Address a social need, utilizing technology to simplify peoples’ lives. 3) Understand and experience the challenges in product design and implementation. In light of these goals, I describe the motivation behind the creating the product that I chose.

One thing that I notice is that it is often very difficult to keep in touch with all my friends from high school and colleagues from my previous summer experiences. It's hard for a number of reasons. There are many different ways to maintain communication with contacts (email, phone, Facebook, LinkedIn...etc.), people prefer different modes of communication, and in many cases people simply forget when they are too busy. However, when I do find some idle time in between classes and meetings, or when waiting around for my delayed flight, I find difficulty remembering those I haven’t spoken to in a long time.

Part of the difficulty in remembering who to speak to comes from the lack of having a centralized contact list. My contacts may be spread out throughout my email, Facebook, and LinkedIn accounts. That can be easily addressed by merging them together using preexisting utilities. However, the other difficulty is in identifying whom I should reach out to when I have time. There may be 1000 people in my contact book, but I don’t necessarily want to talk to all of them.

I want to avoid what I believe is a pitfall of many of these large social media sites (i.e. LinkedIn, Facebook...etc.): You have more "friends" or "connections" than you can realistically maintain. Robin Dunbar, a British anthropologist at Oxford, studied limits on the number of stable social relations that humans are able to maintain. Without going into rigorous analysis of his research, 150 is the "suggested cognitive limit to the number of people with whom one can maintain stable social relationships" [4].

In sum, I believe that the lack of congruency between when one has free time and one’s ability to accessibly manage a list of contacts prioritized based on one’s desired frequency of contact can at times prevent people from using their time well to maintain relationships. If this is true, I want people to have the option of leveraging technology to help them better maintain real, personal relationships.
Assessment of Current Landscape

The current landscape for programs addressing the motivation I mentioned above can be largely broken down into two categories, calendars/to-do lists and contact management reminders.

In the former, the most prominent applications include Wunderlist, Remember The Milk, and Google Calendar. Wunderlist’s key features include ability to implement recurring to-dos, subtasks and reminders, and synchronize across different platforms. In addition, it offers users an ability to collaborate and share to-do lists. Remember the Milk performs the same tasks as Wunderlist does but with two differences. It does not offer collaboration, but does offer synchronization with Evernote. Finally, I include Google Calendar in this list because it essentially allows people to create recurring events on their calendars. Google also offers users the possibility of manually rearranging their to-do lists.

In the latter, the most prominent applications include StayInTouch (Android), Back in Touch (iPhone), and Luper (iPhone & Android). StayInTouch allows users to pick contacts from their phones, set the max number of days between having some sort of communication, and the app generates popup alerts when that number of days has been surpassed. The application searches through a user’s phone call and SMS logs. Back In Touch performs similar functions, except with some additional features. It adds a “gamification” aspect by challenging users to say, keep in touch with 5 friends within 5 days. It also allows users to add a description stating the reason they want to reach out. Finally, Luper sets up reminder popups based on a desired interval, whether that is weekly, monthly, quarterly, or annually.

Shortcomings and Limitations of Existing Programs

While it may seem like the set of programs mentioned above provide a comprehensive set of contract management systems, there are a number shortcomings and limitations that I would like to address.

Security and Privacy

Security and privacy are of utmost importance when handling personal information. In all of the programs mentioned above, the company has authorization to all of the user’s personal contacts and call/SMS logs. Such storage may be convenient for users
when upgrading mobile devices to have the information stored in a cloud server, but is theoretically unsettling. If the companies are exploited or attacked, there is a lot of sensitive information that can be leaked to the wrong people.

**Limitation to Mobile Devices**

The second category of programs offering contact management reminder services is most related to my desired program. However, one large limitation is their exclusive availability on mobile devices. Users often use mobile devices to read emails and provide quick responses. It would be inconvenient to have to constantly switch between an app that sets reminders and email, for example, to contact the people on the reminders. The introduction of a computer program would allow users to be more efficient in generating these emails.

**Reminders Pop Up at “Inconvenient” Times**

While in theory popup reminders are a great idea, they often do not achieve their intended result. The intended result would be a user, whether or a mobile device or a computer stopping and taking the time (5-10min) to write an email or Facebook message to his or her contact. However, in practice the popup notifications rarely popup at “convenient” times. In all seriousness, when is “convenient”? Personally, when I am working, I want to focus on the task at hand. If a popup appears, it is a distraction and a hindrance. In addition, with the proliferation of notifications in nearly all mobile apps and now even in Mac OS X and Windows 8 operating systems, I believe people are becoming desensitized to popups and notifications.

**Inability to manually change reminders**

While some of the applications mentioned above allow users to manually edit recurring events, not all of them incorporate this feature. This is an essential feature because while it may be fitting to set an interval indicating that one wants to email his or her colleague every 3 weeks, it may be the case that the colleague will be traveling for half of the year and has no access to email, or may be generally inconvenient to contact the person during a certain stretch of time.

**Difference in this Program**

This program seeks to address the four main topics described above. In short, I built a web-based application. The contact list would have to be manually managed and
created, but this increases privacy and security because the user is able to add and remove only the contacts that he or she wants to for the application. The program would run on any browser-compatible devise rather than only mobile devices, catering to users who use both mobile and desktop devices as primary forms of communication.

To replace the popup/notification system, I implemented a database system that allows users to insert new contacts into this list whenever they would like, and set variable-length intervals for how frequently they would like to contact their friends and colleagues. Whenever users have free time, as defined by periods of idleness above, they would open the application and it will generate recommendations by calculating the most urgent person to contact at the moment.

Workflow and Design

For the most part, I followed the workflow that I had originally created for this project. Throughout the semester, I kept a detailed log of my progress and hours spent. Below is a chronological breakdown of how my time was spent.

I. Learning and Setup

To begin, I spent ~25 hours learning Ruby on Rails. I followed the Michael Hartl tutorial and found it to be incredibly detailed and helpful. His tutorial focused on creating a Twitter-like rails application and a lot of the features and tools that he explained in his tutorial were highly transferrable to my project.

The next 15 hours I spent simply setting up Rails, associated gems, MYSQL, and ultimately Postgres. I will cover in greater detail why I spent so much time on transitioning from one database client to another in the “Challenges Encountered” section. Compatibility was really a stumbling block that made deployment and testing difficult throughout the project.

II. Backend Round 1

After being 40 hours into the project, I finally had a working platform to write and deploy the application. The following 20 hours of the time I spent were the most substantive changes I made towards the backend of the program. I created a Users table and a Contacts table and mapped a one-to-many relationship between the
I needed to make the database interact, allowing users to fill out forms to add and remove contacts. To do so, I used the SimpleForm gem, which helped me set up form filling capabilities. Next, I used the Devise gem to be able to implement user signups so that different, unique accounts can be created and logged into. I had the bare bones of my program working at this point and was hosting the application locally using a local rails server.

Wanting to see how it would look live online, I signed up for a Heroku account and deployed the app. I was able to get started on their “Hobby-Tier” which allowed me to deploy initial versions of my app on their server for free. While there are certain drawbacks in terms of speed and ultimately size restrictions, this was very helpful for me to visualize and experience the differences between locally and non-locally hosted apps.

### III. Frontend Round 1

The following 30 hours were spent on front-end design. I was pretty particular about style and aesthetics and spent a significant amount of time on this end. First, I installed Bootstrap, which came with many built-in designs, buttons, and formats.

I baked in some features that I thought would make the user experience more engaging and streamlined. For example, I chose to make all “Create New Contact” links modals. I also then worked on ensuring that the top navigation bar would collapse if the screen size got too small.

I realized at one point that while the nature of this application made a lot of sense to me, I needed to be able to communicate the purpose of the app quickly and effectively to people who haven’t heard of my project. What would make someone want to pick up yet another application? I have always found carousels in modern websites to be incredibly informative and clear forms of communication. I spent a number of hours using Photoshop to design what I thought would be a clear way of describing the product.

### IV. Backend Round 2

I then discovered additional vulnerabilities that my application had. For example,
the application did not have a “Forgot my password” feature, “Delete my account” feature, nor did it have a way of certifying that the user was actually human. Though unlikely, I did not want a bot to simply create thousands of accounts on my app and overload both the server and the database.

As a result, I spent the following 10 hours learning how to implement such capabilities. Working on mailer authentications was particularly cumbersome and frustrating, mostly because of compatibility issues between sending mail locally and then realizing how that differed from sending mailer responses once deployed to Heroku.

V. Beta Testing

I decided to release a Beta version of my application to 40 of my friends. Over the course of a week, I collected feedback on a number of parameters. The first half of the survey asked the user to rank how intuitive they thought the application was, how useful they found the graphics, and how likely they’d use the application. The second part of the survey asked more specifics detailing their opinions on how the application can be improved, and any bugs that they might have found.

VI. Frontend Round 2

After analyzing the responses and taking a step back from my project, I made many tweaks to the app. Some of them include the following: When a user signs up at the beginning, s/he is presented a How-To guide on how to get started and utilize the application. I made efforts to increase the number of graphics I used as opposed to text (it has been confirmed that no one likes to read text anymore).

I also did more testing for mobile use. I realized that when the application was open on mobile devises, a lot of the formatting would be distorted. Many people will probably want to use this application when on commutes on their mobile devices so I had to ensure that the user experience on mobile devices would be smooth and enjoyable.

VII. Launch

After additional tweaking, I was ready to launch the application. My app is hosted on Heroku and the link is: in-touch.herokuapp.com. I posted my senior project on
Facebook and LinkedIn, while also emailing friends and family to test the app.

Challenges Encountered and Lessons Learned

1. *As I learned in CS323, you should always write down your plan on paper before you begin.* Think about every scenario that you might be in, and make sure to stress test your application. This project helped me further solidify this idea. While I tried to make it a habit of writing down everything ahead of time, at certain points in the project where I felt like I was on a roll, I would circumvent the cumbersome task of writing down my ideas on paper. Doing so almost always either came back to bite me, or simply dragged out how long it would’ve taken me to accomplish the task.

2. *Compatibility issues are some of the worst that you can encounter. Prepare for them.* As I mentioned above, it took me ~15 hours to simply set up rails properly so that I could begin coding. There are many tutorials online, but often times, depending on when the tutorial was created, certain practices can be outdated and thus incorrect. Incompatibility between different versions of ruby, rails, and bootstrap will not only prevent you from deploying a site, but also generate a lot of unique bugs that will take you a substantial amount of time to debug. I had to change from MYSQL to Postgres because it was until I had everything set up that I realized Heroku doesn’t offer MYSQL support.

3. *Think about product design and talk to more people about it before starting.* I thought I had a great idea at the beginning. And in fact, it was a great idea if I was the only user. However, this project has taught me the importance of communicating effectively and learning to think in other peoples’ shoes. A lot of the feedback that I received after beta testing was surprising. I didn’t realize how difficult some people would find signing up to be simply because of a layout issue. In addition, it was problematic that some people stated that after using the application, they didn’t understand the ultimate purpose of the app.

4. *Stack Overflow is your best friend.* This is a pretty self-explanatory statement. Without this website, I would’ve never understood 90% of all the bugs that encountered or known how to implement different ideas. It was encouraging to see such a helpful community of programmers online who generously helped each other.
Future Work Based on Launch Results

After a period of launch, I learned three additional lessons about user interaction and user experience that will affect my future changes to this project.

1. When using an application, people want instant gratification.
2. People are undisciplined. Removing friction is essential to maintaining usage.
3. This app would be more successful if it was self-learning.

Many apps have trained people to want instant gratification. People want to see “what the app can do for me” as soon as possible. This is especially applicable on the first time that someone uses the application. They want to be recommended the people that they need to contact immediately. However, the first time someone uses the app, they will never be recommended anything. This will cause users to lose interest.

This leads to my second point about users being undisciplined. Many users reported that they weren’t willing to go through the effort of inserting up to 100 contacts into the web app. That makes sense. For me, it was never really an obstacle, but I realized that for something like this to catch on for more people, there has to be some kind of compromise that allows for users to mass-import contacts. Users do not want to go through each individual contact and have to modify their desired contact frequencies, and then have to wait days and weeks before having to log onto the webpage to use it. What makes someone who is bad at keeping in touch with people have the discipline to use this website and even remember to do so?

Finally, this app would be more successful if it collected metadata on user behavior. I would need to add a few more features so that users can have additional behavior outside of simply clicking that they’ve contacted the recommended contact. Some metadata may include the duration of each session and how many times certain contacts are postponed. The metadata could then be used to modify recommendations. For example, the user may indicate that he or she wants to contact Jerry every 30 days. However, after postponing Jerry’s notification multiple times, the application can adjust the contact interval to a higher value in the background.

Conclusion

Creating InTouch was a very challenging and rewarding experience. In completing this project, I was able to achieve the goals that I had set for myself at the beginning of this semester. I am glad that this experience has allowed me to get a taste for what it would be like to work on a start-up idea, and also have the opportunity to experience the
difficulties and joys of working on both the backend and frontend of a full-fledged web application. Now it’s time to actually use the application to get in touch!

References