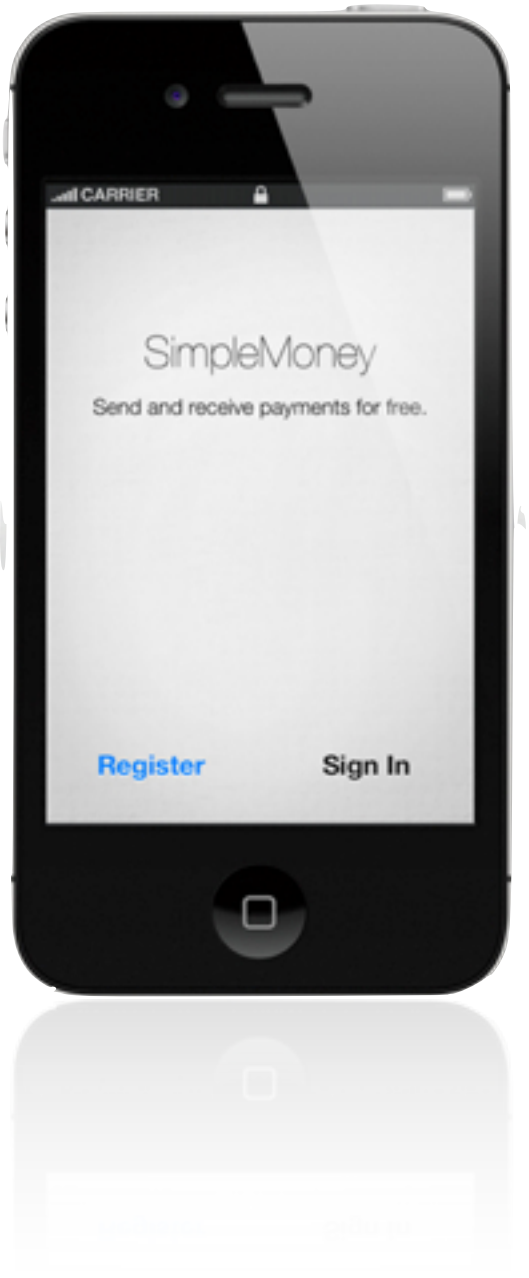
User Experience Optimization for Mobile Commerce Application: Final Report



Arthur Pang
Joshua Conner
Nicholas Pallares

Revision 1.0 - 5/10/12

Introduction

Put simply: it's 2012, and there's still no fast, easy way to transfer money digitally.

While credit and debit cards are an easy way for consumers to pay for things, they present a lot of problems. For one, consumers can only use them to pay merchants for purchases; there's no way to use your debit card to pay a friend back for lunch. Additionally, the card is often the only way for consumers to transfer money; a consumer who loses their wallet on vacation, for example, could be left helpless while they wait for their bank to ship them a new card.

Many merchants, too, have a love-hate relationship with credit cards. While accepting credit cards can increase a merchant's sales, it comes at a cost: in 2011, 44 cents of the average transaction was paid to credit card processors in the form of "swipe fees." Additionally, fraud is a huge issue; according to the Aite Group, credit card fraud costs the U.S. card payments system $8.6 billion per year[1]; to maintain a profit while dealing with those costs, not only do credit card companies charge merchants this percentage fee per transaction, but they often hold businesses accountable to pay back fraudulent credit card payments made to them.

Businesses - especially in the wake of the deepest recession in generations - must pass along these costs, resulting in higher prices for all consumers.

Consumers use smartphones for more and more these days. There are apps for almost everything: turn-by-turn navigation, calendar and to-do list management, communication, and much more. Wouldn't it be great if we could use our smartphones to pay both merchants and our peers?

Solution

Hermes Commerce Inc. (HCI) is a National Science Foundation-funded startup focused on developing a mobile payment system that facilitates peer-to-peer payments and consumer-merchant transactions. Dr. Joshua Cross, the founder and Chief Technical Officer of HCI, is a Cornell-educated physicist and serial entrepreneur; Dr. Cross and two other HCI employees have thus far spearheaded the company's development efforts.

In contrast to the current card payment system, HCI aims to provide a first and foremost secure transaction system to drastically reduce fraud and eliminate these point-of-sale charges; in short, to make the digital transfer of money quick, easy and reliable.

---

[1] http://searchfinancialsecurity.techtarget.com/news/1378913/Payment-card-fraud-costs-86-billion-per-year-Aite-Group-says

Our goal: build "SimpleMoney," the consumer iPhone and Android app for HCI

Momentum - or lack thereof - is the enemy of anyone looking to change how business is done, and it's certainly a concern for HCI. Merchants won't derive any value from adopting a new payments system if nobody uses it, and so HCI seeks to drive consumer adoption of its platform by providing a powerful, intuitive user experience through mobile apps for the iPhone and Android platforms. Our goal was to build those mobile apps.

Additionally, by including value-added services in our application — local coupons, recommendations and customer-relations management — our application will drive adoption among consumers and merchants.

Process Overview

Our capstone team consists of three members:

1. Arthur Pang - Team lead, iPhone lead, back-end developer, client coordinator
2. Joshua Conner - Website lead, iPhone & back-end developer
3. Nicholas Pallares - Android lead, documentation lead

Development Process

Our framework for development mostly consisted of weekly meetings and daily email messaging. In our weekly meetings we:

- showcased our work to our client or mentor
- assisted each other with issues
- planned what to work on for the following week

In between meetings, we used email and the web-based project management service Asana to monitor our progress, ask for help from teammates, and make decisions on small tasks that need to be done between meetings.

An **agile design process** was a natural fit for this project; it allowed us to rapidly prototype and test concept interfaces, and also for our interfaces to grow in sophistication as our skills working with iOS and Android developed. Additionally, the more loosely-structured development plan allowed us to work on development remotely; this was helpful, as it was sometimes hard to find times we are all three available to work together.

Deliverables

As we progressed through our project, we created several major pieces of documentation; this helped us define both the scope of our project and the way in which we were going to execute it.

- **Team Inventory:** introductory memo for our client detailing each team member's experience and our initial ideas for the project.
- **Team Standards:** binding group document defining expectations, process, and conflict resolution procedures.
- **Requirements and Execution Plan:** a document formally defining both functional — what our final project should do — and more qualitative — usability or performance, for example — specifications.
- **Software Design Specification:** defined our project's overall architecture — components and the interactions between components — and our current completion timeline

Timeline

Major Capstone Course Milestones:

- 2/7/12 – Complete Requirements Document
- 2/9/12 – Requirements Presentation
- 2/23/12 – Complete Design Document
- 3/10/12 – First UI Prototype for iPhone application
- 3/10/12 – Complete Registration for UGIRADS conference
- 3/27/12 – Second UI Prototype for iPhone application
- 4/5/12 – Design/Implementation Presentation
- 4/14/12 – Finish capstone poster
- 4/24/21 – Final implementations of iPhone and Android
- 4/27/12 – Capstone Conference Final Project Presentation
- 5/4/12 – Final website updating of all documentation
- 5/7/12 – Complete Final Report

Figure 1: Initial development timeline

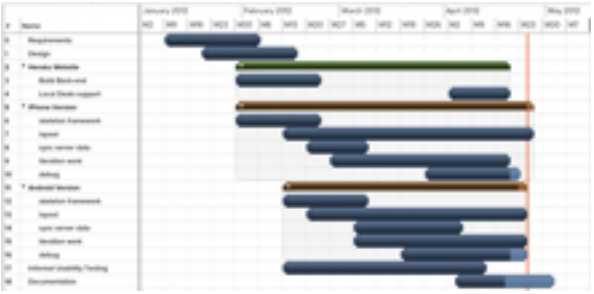


Figure 2: Actual project development timeline

In the timeline defined in our Requirements document (fig. 1), we had set aside time to create two iterations of our iOS prototype, with some user testing in between to help define what we needed to change. Once we felt like we had a good idea of what the final (third) iOS app iteration was going to look like, we were to spend April putting the finishing touches on the iOS app while porting it to Android.

What actually happened was quite different: we developed a single initial iOS prototype and threw it away, and then spent the rest of the semester on a final version, while Android development of a single version occurred concurrently. User testing — which we had set aside significant time for at multiple "checkpoints" throughout the semester — instead occurred in only a limited fashion and only at the end of the project.

What happened? We think four major issues affected this:

1. **Expanding requirements scope.** At our Requirements presentation, however, the feedback we received was that we needed to do more to differentiate ourselves from existing alternatives; basically to show that what we were building could be a viable alternative to Paypal, Google Wallet, or other existing payment platforms. Thus, the scope of our requirements expanded to include making payments to merchants by scanning a QR code and implementing a basic recommendations system. WIth this broader scope, we had to scale back our original development plan.

2. **Poor task tracking and management.** Due to some miscommunication within our team, there was some duplicated effort over spring break when part of our team started work on new version of the iPhone app while work continued on an already existing version.
3. **iOS storyboards can make team work difficult.** We chose to develop using the new Storyboard functionality in iOS 5, which has many advantages over the iOS4 .xib format; most notably the ability to visualize and specify view controller transitions in Interface Builder and see your entire project's flow at once.

   Unfortunately it also has one significant drawback: any changes made to the storyboard not only affect that part of the layout file, but often also update the metadata at the beginning of the file, which basically meant that **auto-merging always failed when two developers had concurrently made changes.** Having to manually merge storyboards is doable, but a pain, and in our opinion held us back as a development team.

   (This doesn't mean Storyboards are a total fail for any projects that involve more than one developer; it's possible to break an application into multiple storyboards and connect them programmatically, giving you most of the advantages of Storyboards while still allowing multiple developers to work on different parts of the interface.)

4. **Little team interaction.** Though we met weekly as a team, we rarely wrote code together, and only when there were immediate issues to solve. Our team leader had experience coding in iOS and two of our team members had experience with REST, but because we all worked separately the vast majority of the team, we weren't able to leverage this experience as well as we could have.

Ultimately, we feel good about what we delivered, but prudent application of some — in hindsight — common-sense software engineering principles, could have made for fewer late nights in the process.

Requirements

We elicited our initial requirements over the course of two meetings with our project sponsor. After drafting our initial requirements, we again consulted with our project sponsor and then presented them for our peers and the department faculty. We received a lot of great feedback from both presentations, and expanded our original requirements scope to better encapsulate what a truly competitive mobile app would look like, including QR payments to merchants and recommendations features.

Goals

Put simply, we were to make a easy-to-use, simple secure mobile payments application for iOS and Android devices. In addition, Dr, Cross had originally requested an application that would interface with a SOAP interface they were developing as the API gateway to their web service. As we developed our vision for the apps and started to

realize the number of changes and additions we'd have to make to the current service, we — with Dr. Cross' permission — built our own REST API that encapsulated the features our app needed to provide.

Solution Statement

The prototype web service is live on the web right now, and our apps can use it to send and request money, pay merchants, locate nearby merchants and get recommendations. New users can sign for the service from the application and begin requesting and sending "money" right away.

Our required functional features include:

1. Login/Sign Up - Any user could sign in or log in using either app.
2. Send Money - A user will be able to send money instantly from their Hermes account to any recipient, whether or not they had a Hermes account or not.
3. Request Money - A user can request money from any peer's email
4. Transaction History - A user can view their history of all their transactions.

Some optional features we included as requirements include:

1. QuickPay - The ability to pay for a purchase from a merchant in a quick and easy way. We ended up using QR codes to facilitate transaction info.
2. Local Deals - The user's location data would be used for advertising nearby merchants with support for Hermes. Would also present the user with merchant coupons to encourage local economic patronage.
3. Recommendations - upon paying a merchant with SimpleMoney, users are presented with another nearby merchant they may be interested in.

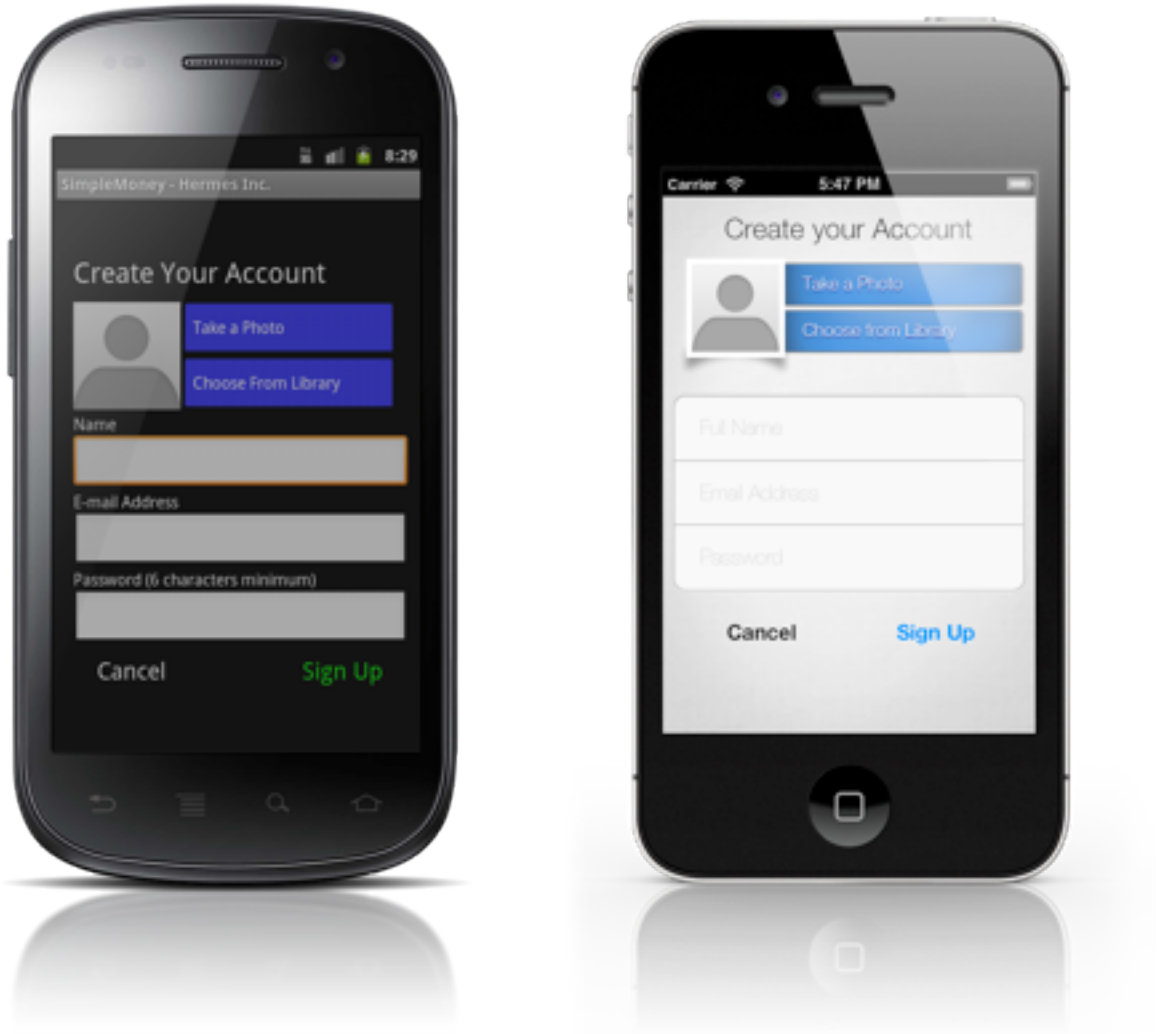
Some performance requirements we met were:

4. Speed - It takes less time to use the Hermes app to pay for transactions than other conventional methods like credit cards.
5. Maintenance - All code is well commented so that if any adjustments need to be made by our client, they can easily understand our work and make desired changes.

Functional Specifications

We have implemented the following functional specifications: User sign up and login, Quick Pay, Sending and Receiving Money, Transaction History, and Local Deals.

Sign Up/Login



If a user wants to use the Hermes app, they could sign up from either the Android or iPhone. Required parameters are a name, email, and password. An optional parameter of an image can also be used for the user's avatar when signing up. The image can be taken directly from the user's camera or pulled from the user's gallery. Only the user's email and password are required for logging in.

QuickPay

When the QuickPay feature is selected from the main menu, the app will start its QR scanner mode. From here the app will scan and recognize QR codes that were designed for Hermes' app. The QR code contains transaction info about a purchase the

user wants to make to someone (usually the merchant who generates the QR code) and will then confirm the user's choice of wanting to pay the bill by displaying the transaction info that was encoded in the QR code.

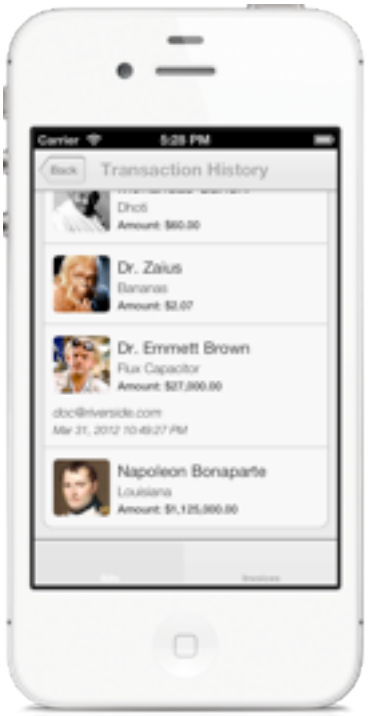
Send Money

When the Send Money feature is selected from the main menu, the app we go to a form that asks the user for info about the invoice. A recipient email and amount are required to send money and an optional description can be entered as well. The user can also select the recipient's email from their phone's contacts list. If approved, the user's balance will be deducted by the amount and a transaction will be sent to the recipient, indicating that they were paid by a this user.

Request Money

When the Request Money feature is selected from the main menu, the app we go to a form that asks the user for info about the invoice. A recipient email and amount are required to request money and an optional description can be entered as well. The user can also select the recipient's email from their phone's contacts list. If approved, a transaction will be sent to the receiver as a bill and the user will be paid for the request if the user approves this transaction.

Transaction History

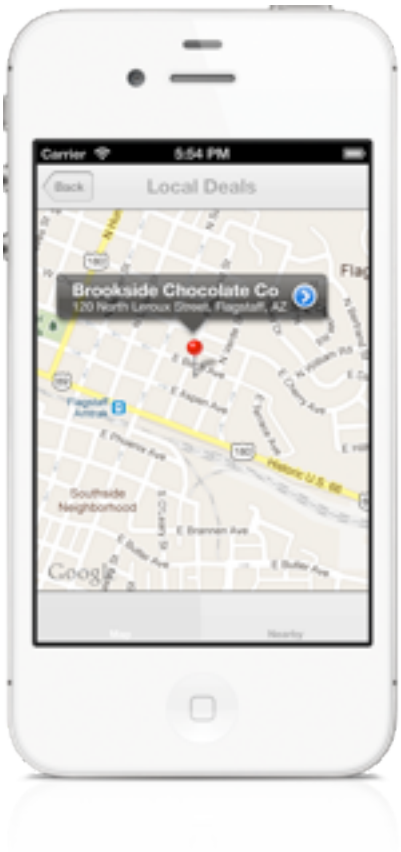


When the Transactions feature is selected from the main menu, the app we go to a page that shows the user all their transactions with Hermes. Each transaction contains details about the recipient's name and email, the amount of the transaction, the date the transaction was created or updated, and the transaction description.

There are two types of transactions, Bills and Invoices, and each transaction is either paid or unpaid.

- Unpaid Bills - shows the user which of there transactions, made by peers requesting money or merchants, need to be paid off.
- Paid Bills - shows the user history of transaction made to pay peers or merchants.
- Unpaid Invoices - shows list of peers who the user requested money from but have yet to receive payment
- Paid Invoices - shows list of peers who paid for money requests made by the user.

Local Deals



Uses the user's location data stored on the phone to show map of nearby merchants with registered support for Hermes. If a store is selected from this view, the user will be presented with info about the merchant and a coupon if the merchant has a promotion.

Architecture Overview

The HCI payment platform can be broken down into three components:

1. Merchant Application
2. HCI Server
3. Customer Application

The Merchant Application is responsible for posting transactions to the server and generating QR codes containing transaction data. The server performs CRUD operations for users and transactions, and also authenticates users. The Customer Application captures images with a camera, processes the images with a QR code reader, checks the QR code for a HCI signature, creates transactions, and fetches transaction data from the server.

Payment Process

The payment process can be summarized in 10 steps from actions by the Merchant to services by the HCI Server to tasks by Customer and back to the HCI Server for final checks.

Merchant

1. Create new Transaction
2. Display QR Code
3. POST Transaction to HCI Server

HCI Server

4. Create new Transaction

Customer

5. Capture Image
6. Process QR Code, check for HCI signature
7. Create new Transaction
8. PUT Transaction to HCI Server

HCI Server

9. Edit Transaction
10. Send response (success/error) to Merchant and Client

Component Descriptions

Each screen or service mentioned has a view controller associated with it and is indicated by the number next to it in parentheses.

The Customer Application begins with the login screen (1) where the user enters their HCI username and password. Using these parameters, the app connects to the HCI server to authenticate the user, and redirects them to the app's home menu (2).

The home menu has one small window at the top displaying basic info about the user. The home menu also has a menu consisting of buttons showing the user access to five different functions. These functions are: send money (3), view pending transactions (4), view transaction history (5), and view local coupons (6), scan a QR code (7). Each service will redirect the user to a different view, associated with the task, where they will be guided through completing their desired action in an orderly and natural process flow.

Component Functionality

1. LoginViewController
    - User enters their username and password into text fields
    - Authenticates user's login parameters with HCI server and redirects them if successful or stays here if denied

2. RootViewController

- Contains one small window located at the top of the screen displaying basic info about the user such as their full name and account balance
- For the iPhone, beneath the window is a single column table with buttons for the different app services.
- For the Android, beneath the window is a 3x2 table with buttons for the different app services. This 3x2 table is really created as a 2x2 table and a 1x1 table directly underneath it to house 5 buttons in a way that looks like a 3x2 minus 1 cell table.

3. SendMoneyViewController

- For the iPhone:
  - Displays contacts from address book
  - Displays modal dialog for a new transaction
  - Create a new transaction and sends a POST request to the server

- For the Android:
  - Displays simple fill-in form
  - Can select contact from
  - Displays contacts from address book
  - Displays modal dialog for a new transaction
  - Create a new transaction and sends a POST request to the server

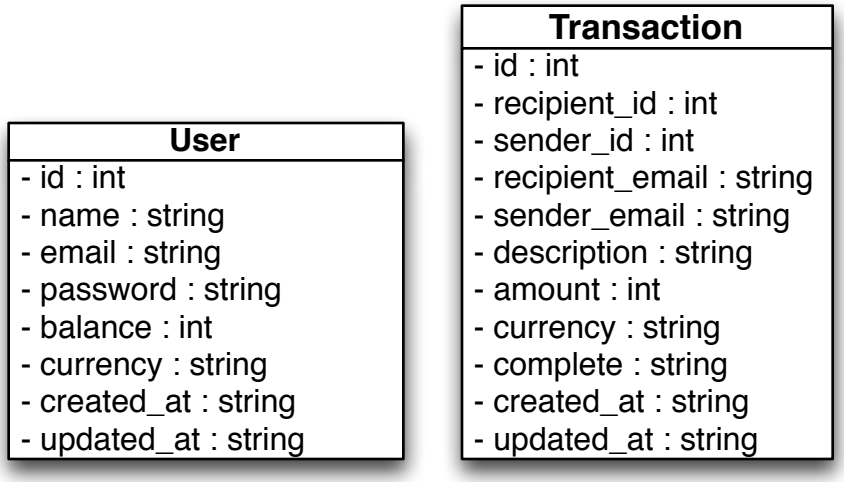
4. TransactionViewController
   - Fetches and displays pending transactions from server
   - Each transaction cell will have info about the sender's name and the balance needed to be paid to them

5. HistoryViewController
   - Fetches and displays paid transactions from server.
   - Each transaction cell will have info about the recipient's name and the amount paid to them

6. CouponViewController
   - Fetches and displays local coupons from server

7. ImageCaptureViewController
   - Captures images from camera
   - Processes QR code
   - Creates a new transaction and sends a PUT request to the server.

Model Descriptions
Merchants and Consumers have many transactions, and each transaction contains the following fields:



Usability Testing

Our main subjects for usability testing were our client and capstone mentor. Using nothing but our notes and light conversation, every week we discussed our project's progress and showcased completed parts of the design. From these meetings we've gathered insight on what parts of our projects were good and what parts needed refinement (for example, local deals was continually discussed and the concept evolved during and after development).

These informal usability tests were satisfying as we knew that user satisfaction would be met once deployment began. We were even able to showcase our apps as live demos during our capstone's poster session, where many onlookers were impressed by the purpose of the company, the look and feel of the apps, and the functionality that we outlined to intrepid questioners.

Future Work

Beyond the obvious immediate future task of integrating our apps with the HCI infrastructure (influenced by our work through the semester, HCI rewrote their backend to use a REST API and integrate some of the features they knew we were building), the most immediate future task is to more completely unify the iOS and Android user experiences. While we were careful to keep a consistent flow in both apps (consumers follow the same sets of actions to perform tasks on both platforms), the look-and-feel of both apps is less consistent than we would like.

Additionally, we think a more fully-integrate and robust user of location-awareness features would continue to differentiate HCI from other competitors. Social features

could be a great addition as well — share something when you purchase it! — and a fantastic vector for commercializing the service; after all, what better metric of consumer engagement is there than when the consumer actually purchases a product?

Conclusion

Our sponsor Joshua Cross, was very pleased with our design and implementation. He stated that he particularly likes the ‘look and feel’ that we have created, and intends to incorporate our design into the first version of the application. Our design met all of the project’s functional requirements, and included two additional features, local deals and Quick Pay. We are very happy to have had to opportunity to define our sponsors application from the ground up.