BeanPay

Ben Andrews, Charles Dorner

Problem and solution overview

When we began the BeanPay project we wanted to solve the problem of complexity in the personal budget space. We began by asking the question – How can budgeting be simplified such that the barrier of entry increases the likelihood of actively managing ones budget. This is a difficult problem to solve and there are many attempting – Google, Microsoft, Apple, Mint, to name a few.

After some contextual inquiry we realized there is a twofold problem – Firstly, the current payment system is outdated and not integrated well with technology, and secondly there is a significant delay in the feedback loop between making a purchase and seeing the balance sheet. Many existing solutions are setup for long term management of all aspects of personal finance such as loans, incomes retirement etc. But we wanted a solution that focused on the day to day transactions and discretionary spending. So BeanPay set out to solve a twofold problem – On the one hand improve the payment system, and on the other improve the availability of data needed for a user to make sound budget decisions.

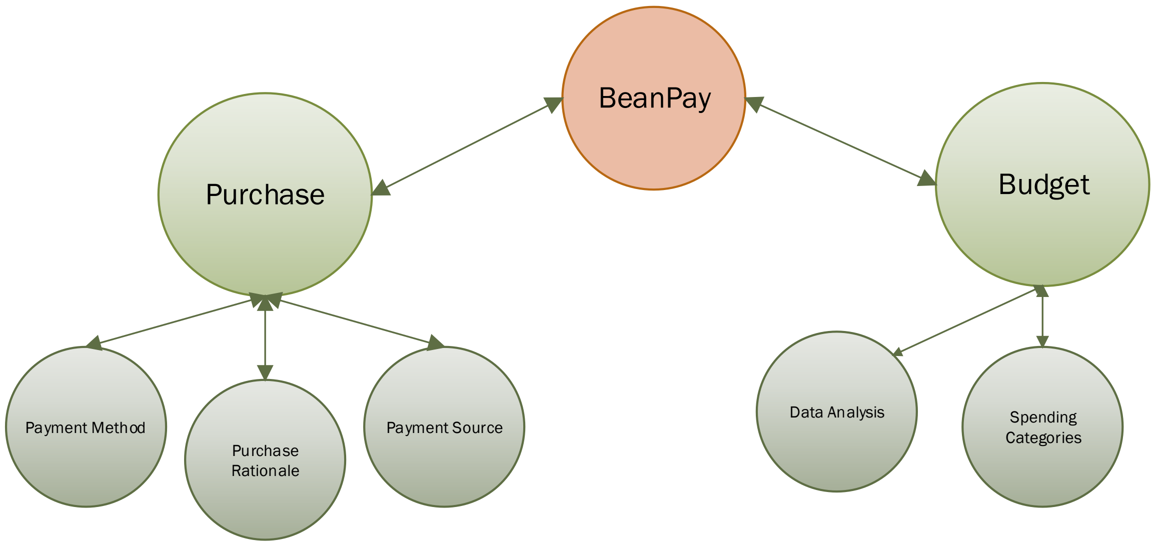


Figure 1

In our contextual inquiry it was evident that there was often a disconnect between the users spending habits and their budget. Figure 1 shows the relationship between purchases, budget and BeanPay. On the purchase side there are three main factors – 1) Payment method e.g. cash, credit, electronic, 2) Payment source e.g. savings, reimbursement, and 3) Purchase Rationale e.g. need, want. Whenever a user sets out to make a purchase they firstly rationalize the purchase based some existing constraints. Once they have decided to make the purchase, payment method and source will quickly fall into line.

On the Budget side there are two primary components - 1) Spending Categories, and 2) Data Analysis. After a user makes a purchase they will eventually consult their balance sheet. While this activity varies between users, those who attempt to maintain a budget will categorize their spending and analyze their spending habits based on a pre-defined budget.

When considering the twofold problem that BeanPay attempts to solve, we must also look at the feedback loop between making a purchase and monitoring a budget. It is here that BeanPay has the most potential to positively impact a user's spending habits.

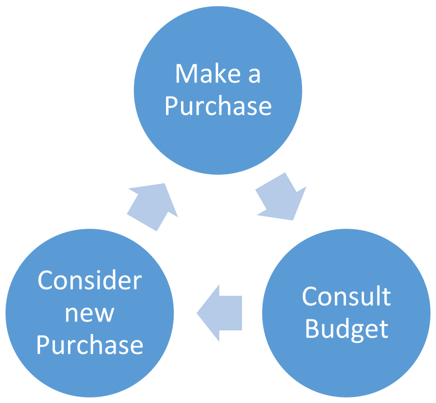


Figure 2

In figure 2 we see the purchasing feedback loop. First a person considers a new purchase which is followed by consulting their budget, followed by considering a new purchase. By providing a single combined solution for both spending and budget monitoring BeanPay reduces this feedback loop which has the potential to positively impact spending.

Ultimately, BeanPay is an interface that incorporates various payment methodologies with simple budget analysis to seamlessly combine the purchase experience with the goal of improving a user's spending and saving habits with real time status and feedback.

Paper prototype description, with overview shot and close-ups

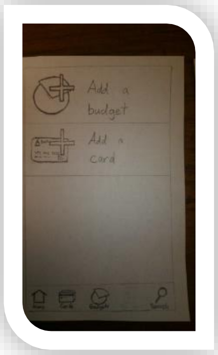
During the contextual inquiry process and design sketching we focused on the twofold problem BeanPay set out to solve – on the one side, solving the problem of user payments – and the other, keeping track of a budget while managing spending categories.

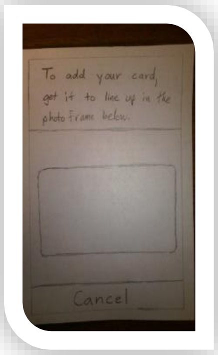
In our first prototype we explored various methodologies for payment tracking as well as budgeting. One component of this system that we identified as being useful was a payment timeline which tracks payment history. From here the user can quickly categorize and analyze a payment.

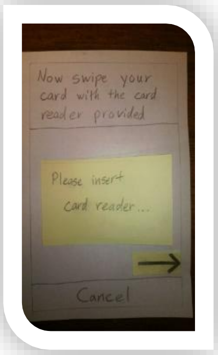
In our second prototype we focused more on the budget side – giving the user control of setting up categories for spending and tracking them against current budgets.

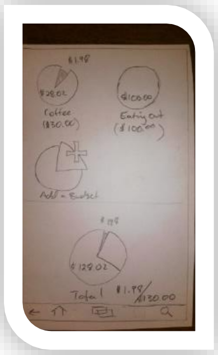
In our 3rd prototype we explored an interface for payments and a credit card replacement. During our inquiry it became clear that while people might use a credit card form payment device for paying, they want a more powerful platform for performing other tasks – for example they expect notifications to happen on their mobile computing device. They want to be able to dive deep into purchase information on their desktop computers. They expect to be able to view, understand, and examine their financial data on any computing platform.

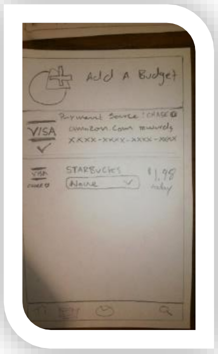
Thus we concluded that given today's technology a multi-device system would be necessary – a single store of information online, with multiple interfaces into that system from a variety of devices. The final BeanPay paper prototype interface allows for standard credit card purchases using a single digital card which we named the "BeanPay Magic Card". It also includes a timeline listing all transactions and their associated category, budgets view, and ability to manage multiple credit cards. In the below figures we have captured screenshots of the prototype we used for testing. Beginning with the home screen, guiding users through the process of adding a card to the system, adding a budget, and making a purchase.

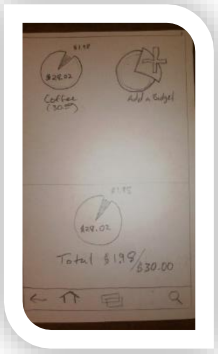


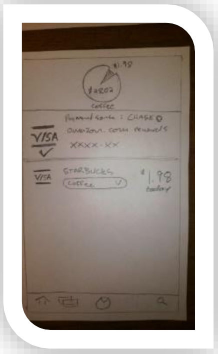

To add your card, get it to line up in the photo frame below.
Cancel

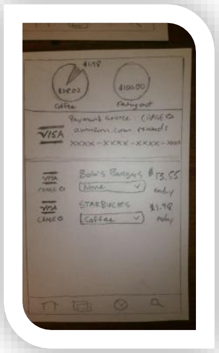

Now swipe your card with the card reader provided
Please insert card reader...
Cancel

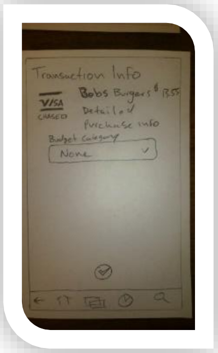












In the large figure above we have a picture of our BeanPay Magic card which electronically stores all credit cards for a given user. Our design assumes that we would be using a programmable interface on the card, but since we used a paper prototype, our card was hard-coded to a single card.

One crucial element to our design was to give the user the most pertinent data needed on the landing screen to impact the spending/budgeting feedback loop. So our final design included a budgets overview at the top of the screen, along with the timeline of the most recent transactions.

Testing Method

Participants

The first participant was a 22 year old woman working for a clothing retailer as an assistant merchandiser. She spends around $800 / month on food and while she has mint.com set up, she hasn't changed or updated a budget since college. She hates looking at mint.com because of "how bad she is doing" on keeping her budgets.

The second participant was a 31 year old male electrician. He is very spending conscious and the spending he does that most worries him is around $250 / month on gas. He does not use any formal budget tracking software but wishes he could understand his spending better.

The third participant is a woman in her early 40s working as a manager for a local coffee shop. She doesn't have time for budgeting but wanted to have more control of how much money she spends. She thinks she spends a lot on baby needs food, diapers, etc. but really has no idea and wouldn't want to cut down on these purchases anyway.

Environment

Our environment was a quiet room with a table with sufficient room for the various testing items such as the paper prototype and note taking materials. We wanted to simulate the new user experience so we packaged the whole system in an envelope (See figure 3). Once the BeanPay system was prepared we began by asking users to add new cards to the BeanPay Magic card and verify them in the BeanPay app.

Tasks

Task #1

You have just received your new BeanPay system and installed the software on your mobile device. It is now time to get started. Using the BeanPay mobile companion app and the included card reader, add your Chase Rewards Visa and Fidelity American Express to the BeanPay system.

Task #2

Since most of your purchases will be completed using your Chase Rewards Visa card, ensure that the magic BeanPay card is setup to pay using your chase rewards visa card.

Task #3

Now that you have added a credit card to the system you decide to head out and buy a coffee. Make a purchase using your BeanPay Magic card at starbucks. Once the transaction is complete open the PeanPay companion mobile device to view the transaction.

Task #4

After making your purchase you realize that you spend a lot of money on coffee each month and you would like to set a budget. Open the BeanPay mobile companion app and create a new budget category for coffee. You want this budget to be set on a monthly basis with a spending limit of $30.

Task #5

You have a single budget for tracking coffee, but you realize you also spend a lot of money eating out. Create a budget of 100.00 for eating out. Again you want this budget to be on a monthly basis.

Task #6

Its lunchtime so you head down the street to Bob's Burgers where you make a $13.55 purchase on a burger and fries. Since Bob's Burgers just opened you're not sure if the BeanPay database will track the purchase accurately. Open the BeanPay mobile companion app and ensure that the purchase is tracked under the Eating Out budget you created.

Task # 7

Now that you have been using BeanPay for several weeks, you are curious to see what establishments you've visited that are tracked under the Eating Out budget. Open the Eating Out budget and view the transaction history.

Procedure

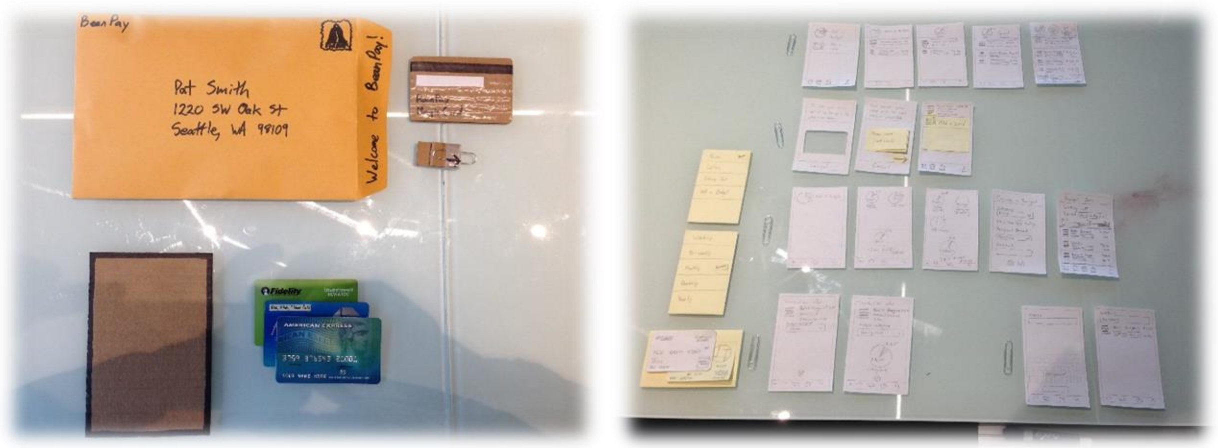


Figure 3

Following the standard methodology of paper prototyping we wanted to ensure that all our scenarios and tasks were up to date and fully implemented in our user interface. We quickly found that several of our tasks no longer made sense and that several features weren't fully accessible through the prototype interface. Once we had validated our scenarios we felt confident using our low fidelity prototype in some user tests.

For each participant, our procedure was firstly to explain the scenarios, give a little background information, and finally begin observing as they executed various tasks using our low fidelity prototype. Throughout the process we took extensive notes, focusing on problems and confusion our users encountered.

Based on the standard practice for conducting user tests, we focused our testing on two types of data: Process data – which includes observations of what the user is doing and thinking, and 2) summary and statistical data – focusing on measuring what happened e.g. time, errors, and thinking.

Test Measures

Most of our testing was focused on process data – e.g. what the user was thinking and doing as they attempted to execute our task list. While we did do some heuristic tests and value sensitive design, it was secondary to our process data collected from real user interaction. Thus, when discussing our testing we will primarily focus on the process data collected during user testing.

Testing results

During our testing there were two areas were we paid particular attention 1) areas where all three participants made similar suggestions and 2) areas where all three had similar problems. By focusing on these two areas we could identify components of the UI that would be the most problematic to users and any improvements in these areas would have the most impact.

The primary area where all three users had suggestions was around the categories section. All three wanted to be able to create their own categories. They felt that the drop down category listings were too limiting (even if there were more options) and wanted to create their own custom titles. Based on this result, we decided that future iterations would not only include auto category detection, but also enable custom category creation.

The major area in which all three users had problems was regarding which card they were supposed to swipe to add to the system – the BeanPay card or one of the credit cards. Connecting the BeanPay card to the mobile device was a missing part of the process for them.

There was also some confusion regarding what the budgets did in relationship to the cards. One participant thought changing the card on the home screen would change the transactions showing. Another thought that when they paid their credit card the transactions from that card would disappear. The third user to test the prototype thought that the transactions would be cleared at the beginning of a new budget period.

Two of the user test participants thought that the budget periods would be the same across budgets. They also had some confusion about the menu options – one with what they meant, and the other with why the options moved around and the back button being in the wrong place.

One participant was confused by the pie chart having the spent amount be a smaller radius that the amount remaining. They were also unsure if the BeanPay card had worked because it did not show a completed transaction on it after it was swiped "shouldn't it tell me it did the purchase?"

Another source of confusion was around the budgets view. One participant was confused that tapping on "Eating Out" on the main screen took him to the "Eating out" budget instead of to see all of the budgets. Another participant was unclear what the pie chart was at the bottom of the budgets screen was for.

Ultimately none of the participants said they would use this system as is. Reasons included: it is not convenient to have to use your phone to change the credit card being shown on the BeanPay card, not wanting to see or use budget information, not wanting to do the work of setting up budgets, wanting more information about budgets before and after transactions, and not seeing enough value in the system. Secondary findings are areas where improvements could be made but may be specific to those user's use cases.

Interface revision sketches

When we began BeanPay we wanted to create a system that made a positive impact on a user's personal budget. Throughout the design process we continuously asked ourserlves: "What is the most relevant information that will positively impact a user's spending habits?" This question was the driving force behind our interface revision.

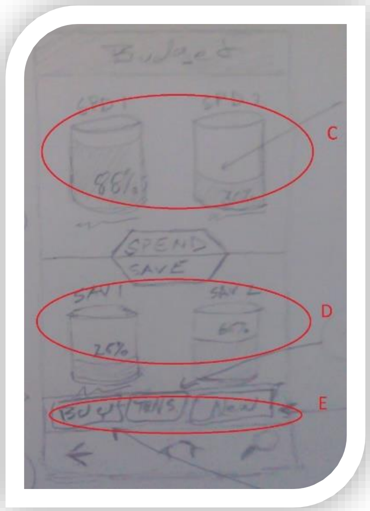


Figure 4

In figure 4 we have one of our original sketches – Here we see two primary pieces of data 1) spending categories, 2) saving categories. But we quickly discovered that this wasn't the primary information people were looking for when the first opened the application. While it is crucial to display a budget view of some sort on the main page, this original sketch took up too much screen real estate without sufficient data to positively impact a user's spending habits.

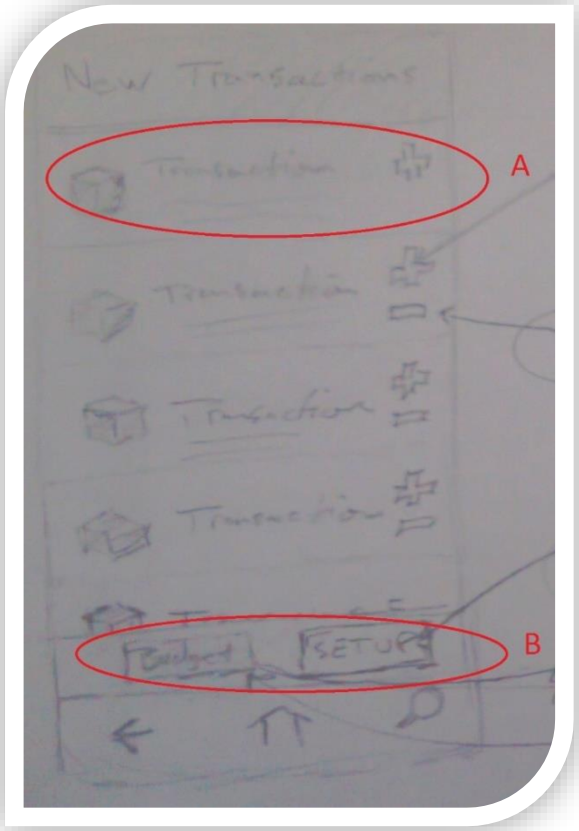


Figure 5

In figure 5 we evolved the design to not include any budget information and instead display a timeline of recent transactions. This approach focused more on the spending side of the application and less on the budgets. We really liked having a timeline as the primary source of data, but we felt that the user would also like to see budget data.

As we iterated on our design we wanted to ensure that not only were we reducing the feedback loop of budget data, we wanted the interaction with the application to require minimal amount of user input. While, the transaction history was very useful we found ourselves having to click away from the main screen to further examine budget information. Thus our next iteration included reduced real estate for the transaction timeline as well as a budgets overview.

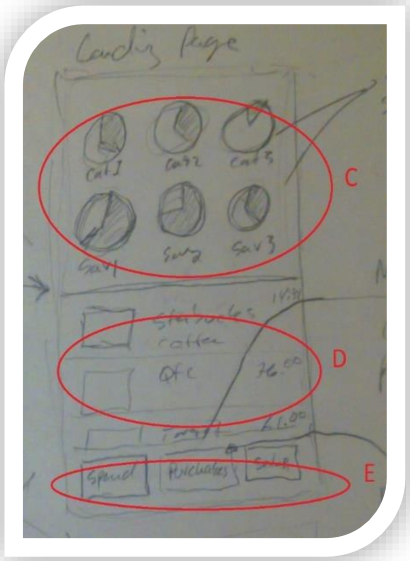


Figure 6

In figure 6 we have our first attempt at combining the transaction timeline view with the budget view by including the top 3 budget categories and the top 3 saving categories. With this design we felt that the primary data required to impact a user's spending was available on the first page. Here the user could see all transaction history along with a summary view of budgets and savings.

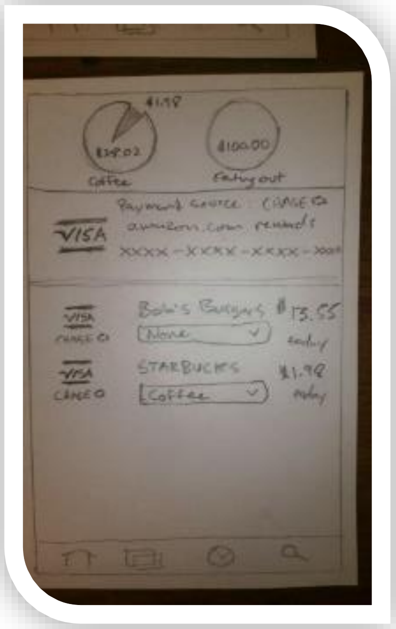


Figure 7

The timeline/budgets combo view was the view we used during our user testing. It not only included a timeline and budgets view, but also listed the current payment source for the BeanPay MagicCard. Also, we felt that it was necessary to be able to adjust categories for new transactions on the main screen, so we included a category drop down next to new purchases (a feature we ultimately ended up removing from the interactive prototype). Based on our testing results we made a few slight modifications to our final interactive prototype.

Interactive prototype overview

After completing several iterations of our paper prototype and gather results from our user testing, we decided to take this feedback and implement an interactive prototype. For our interactive prototype we chose to implement a JavaScript web page with several fixed paths which correlated to our tasks. Our interactive prototype includes a mobile device with the BeanPay app, the BeanPay magic card, and interactive purchase options which enable a user to make fake purchases with the system.

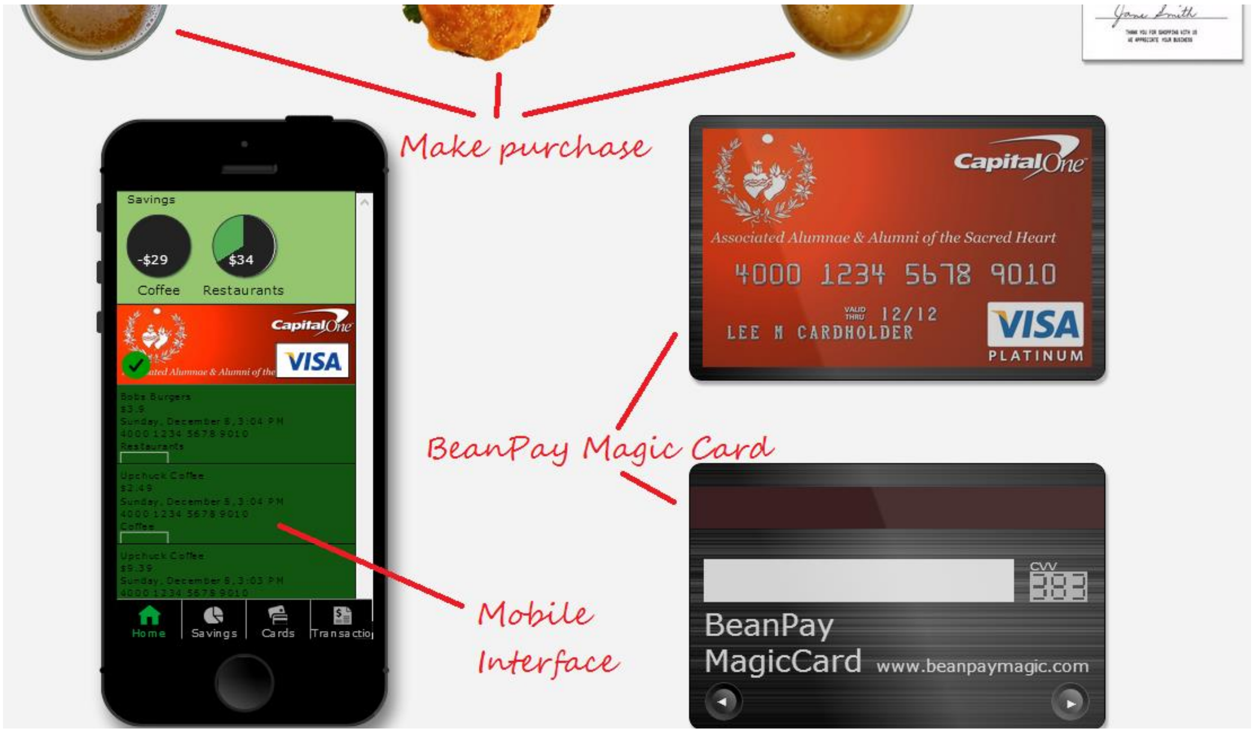


Figure 8

Figure 8 is a screen shot of the interactive prototype with all components required to complete several fixed path scenarios. In the upper section of the interactive demo there are four icons – 1) beer, 2) burger, 3) coffee, 4) receipt. When either the beer, coffee, or burger are clicked, the mobile interface is updated with a purchase from either Joes Tavern, Bobs Burgers, or Upchuck Coffee.

Overview of the implementation

There are three primary activities which we have enabled with our interactive prototype – Payment source management with the BeanPay MagicCard, Purchase tracking for three types of purchases, and budget management for the corresponding budgets.

Below is the front and back of the programmable BeanPay MagicCard. The BeanPay MagicCard in its current form has the ability to scroll through the various cards which have been pre-programed into the device: Visa, AmEx, MasterCard. Through the BeanPay mobile application the user can add a new card or select which card to set as the payment source in the BeanPay MagicCard. The MagicCard itself has limited interactivity – except for the ability to be programed via the companion app and scroll through various payment sources.

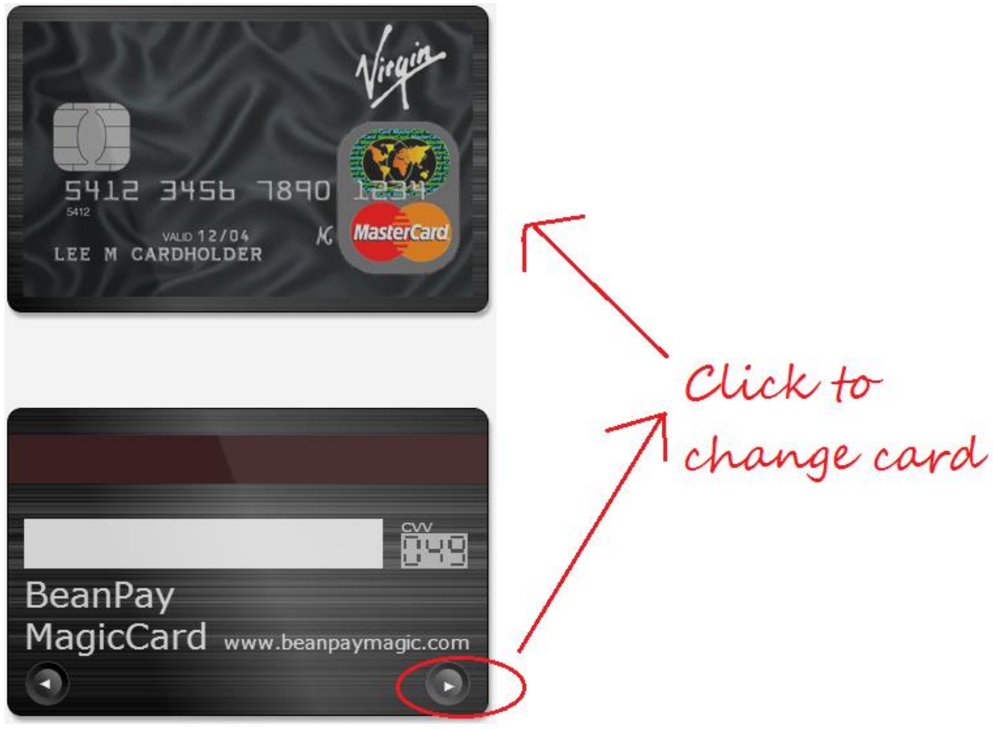
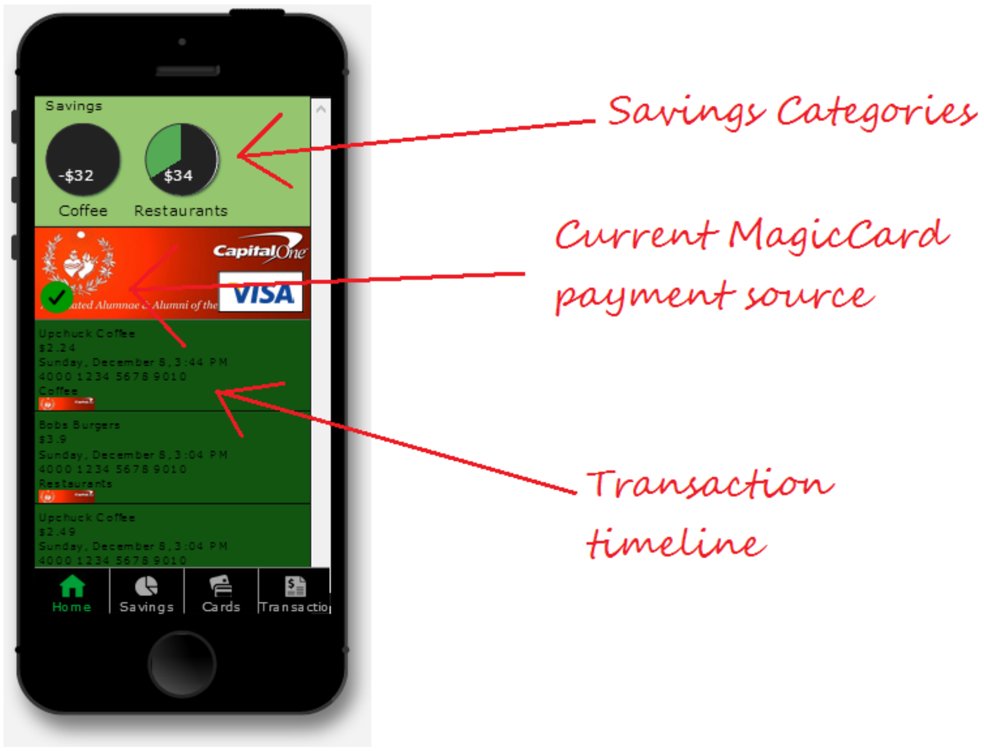


Figure 9

In figure 9 we see that by clicking the arrow button on the bottom right hand corner on the back of the BeanPay MagicCard the user can move from the Capitol One Visa card to the Virgin Master Card. If the user rotates through all his/her cards they will loop back to the first card. For the BeanPay MagicCard there are many features that would need to be implemented in order to make it a viable solution – primarily, there is no consideration of security in our implementation.

After selecting a payment source the user can now make a purchase by clicking on either the burger, coffee, or beer. When the purchase is made using one of these icons it automatically updates the transaction timeline along with the corresponding budget.



Upchuck Coffee
$2.24
Sunday, December 8, 3:44 PM
4000 1234 5678 9010
Coffee

Bobs Burgers
$3.9
Sunday, December 8, 3:04 PM
4000 1234 5678 9010
Restaurants

Upchuck Coffee
$2.49
Sunday, December 8, 3:04 PM
4000 1234 5678 9010

Figure 10

Figure 10 is our final interactive prototype with three primary pieces of information readily available to the user: 1) Transaction timeline with all recent purchases, 2) Current MagicCard payment source, and 3) savings summary for top two spending categories.

In the transaction timeline each transaction has five primary pieces of data: 1) purchase establishment, 2) amount, 3) date and time, 4) category, 5) payment source. Unlike our paper prototype we have decided to implement auto categorization of various purchases. If we were to fully implement the BeanPay system most likely we would want to implement a hybrid model in which the category is automatically selected for the user, but the user could also manually select a custom category.

Scenarios

Adding a card to the system

One of the first activities a user will complete is adding a card to the system. Both in our paper prototype as well as our interactive prototype, this is accomplished by taking a photo of the card. Here we are going slightly outside the boundaries of available technology. While it is possible to extract credit card number using ocr from an image, there is other data stored on the magnetic strip of a card that cannot be extract. But one can envision a future in which a user could download a credit card directly their bank onto the BeanPay mobile app without having to wait for a card in the mail.

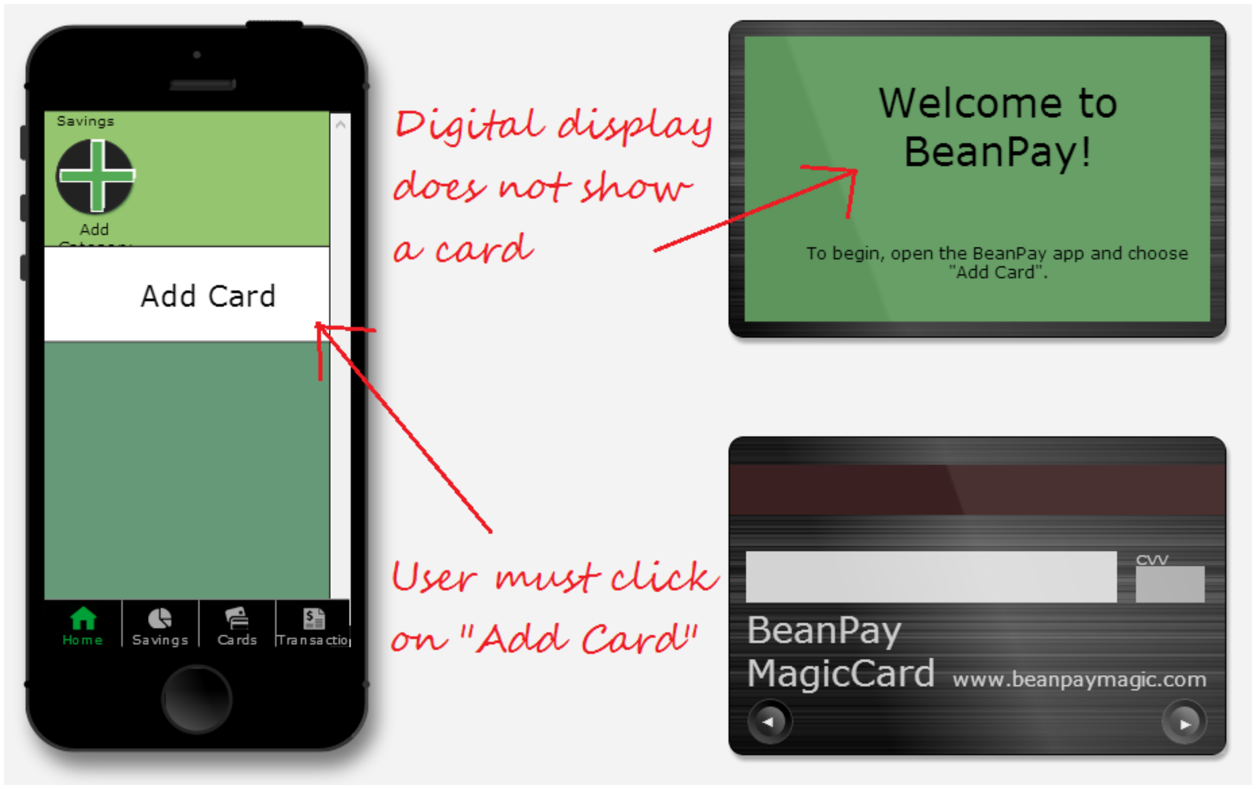

Savings
Add
Add Card
Home
Savings
Cards
Transactio
Digital display does not show a card
User must click on "Add Card"
Welcome to BeanPay!
To begin, open the BeanPay app and choose "Add Card".
CVV
BeanPay MagicCard
www.beanpaymagic.com

Figure 11

When a new user first begins to use the BeanPay system the home screen prompts them to "Add card". After clicking on "Add card" they are taken to a screen which prompts them to take a picture of the card. Figure 12 and 13 highlight this scenario – firstly to select the card, and secondly to position the mobile device over the card such that it can extract necessary data.

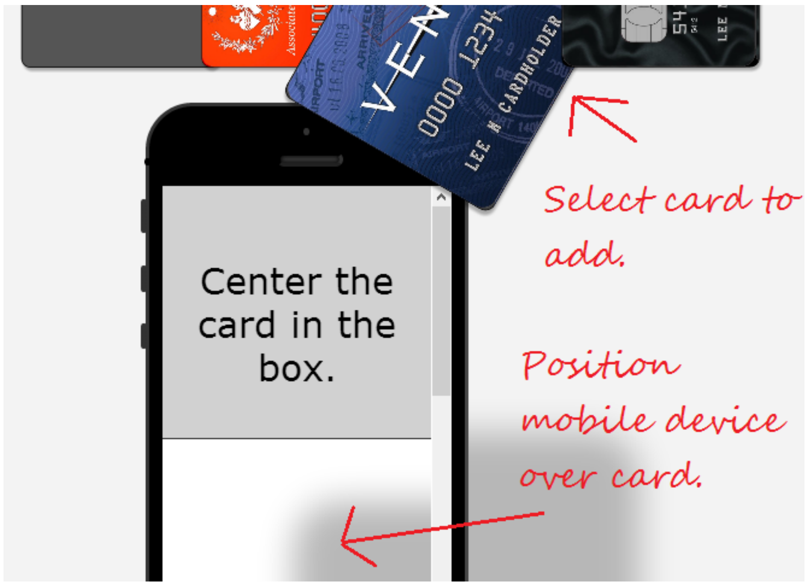

Center the card in the box.
Select card to add.
Position mobile device over card.

Figure 12

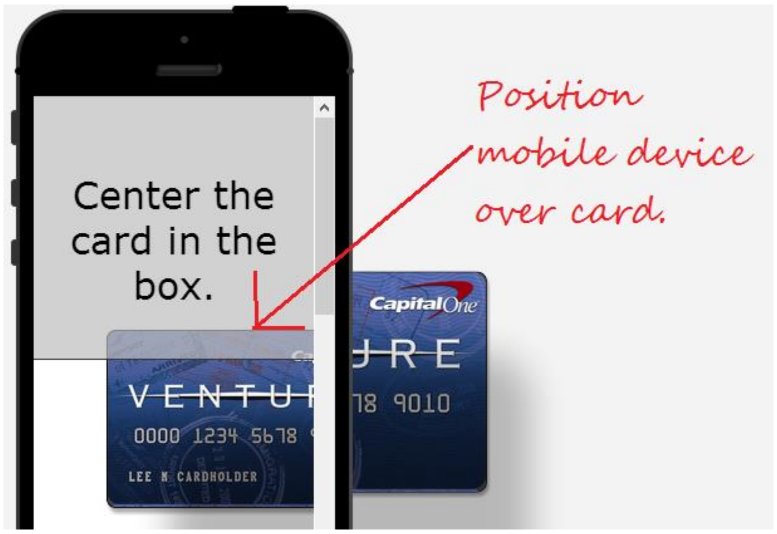

Center the card in the box.
Position mobile device over card.
Capital One
VENTURE
0000 1234 5678 9010
LEE M CARDHOLDER

Figure 13

Once the user has positioned the mobile device over the card, the system automatically detects the presence of a card and adds the card data to the system. Since this is the first card added to the system, it is automatically set as the default cad and the card data is transmitted to the BeanPay MagicCard.

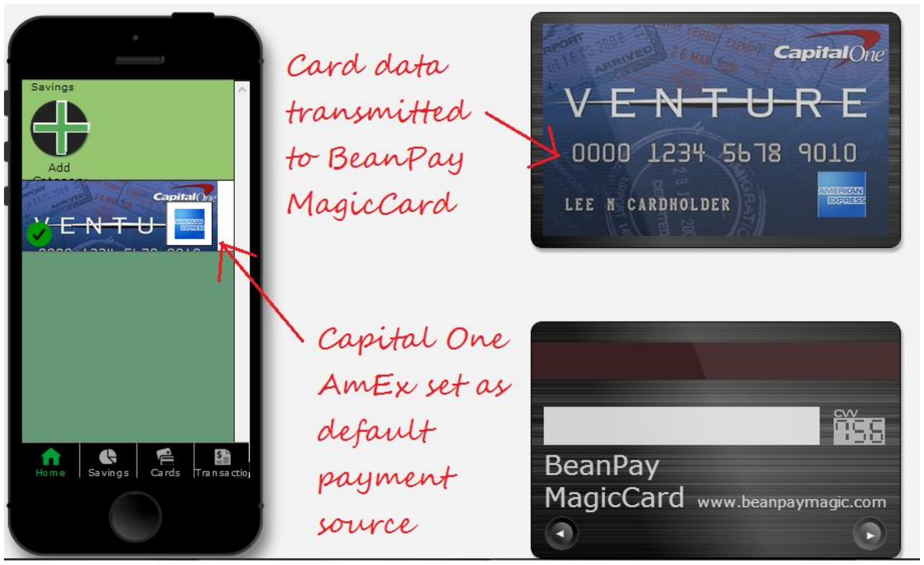


Figure 14

Making a purchase

Once the card has been added to the system the next primary scenario that our interactive prototype supports is making a purchase. Unlike adding a card to the system, making a purchase is a passive action. E.g. the user merely hands a merchant his BeanPay MagicCard and the system does the rest.

In order to simulate a transaction, as mentioned above our system has three built-in merchants: Bob's Burgers, Joes Tavern, and Upchuck Coffee. In order to complete one of these transactions the user first ensures that his BeanPay MagicCard is setup, then clicks on either the beer, burger, or coffee to make a purchase at one of the three preset merchants.

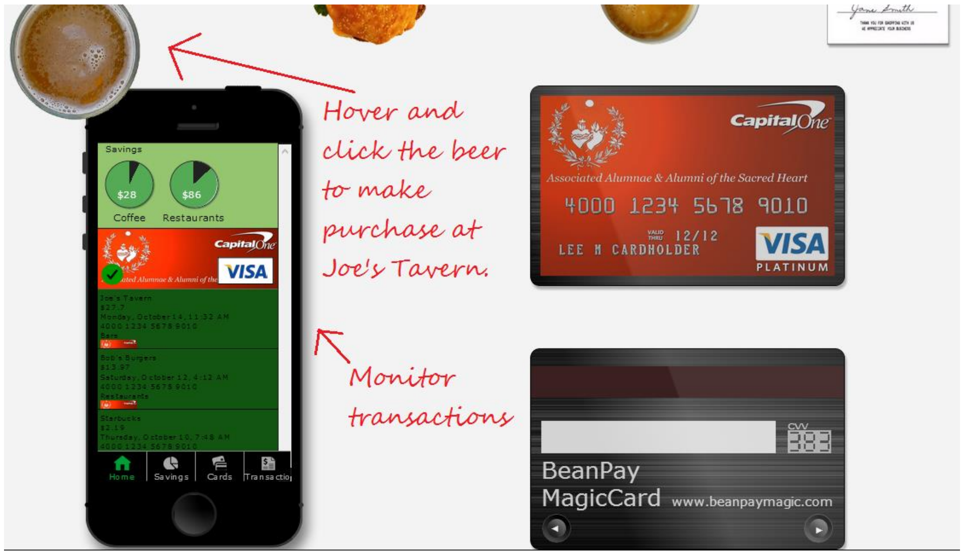


Figure 15

In Figure 15 the user first hovers over the beer which expands from the top of the screen. He/she then selects the beer and observes that a transaction appeared in the transaction log.

Budget/Savings Tracking

The final scenario supported in our interactive prototype is the ability to monitor and track budget categories. To do this, the user must first open the savings menu and add a category. For the purposes of reducing complexity we have reduced this down to three possible categories: Coffee, Restaurants, and Bars.

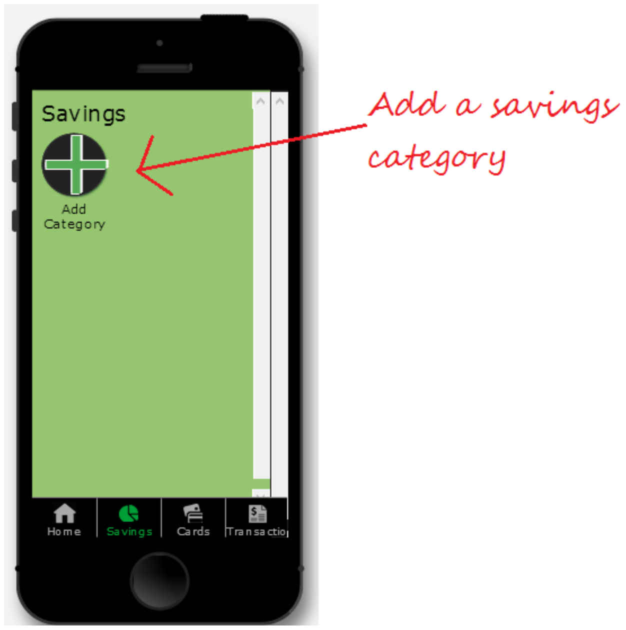


Figure 16

Figure 16 displays the default savings page which is empty for new users. To add a new category click on the "Add Category" icon. For the purposes of reducing complexity of the interactive prototype we have hard-coded the categories to the three listed above.

Once the user has added their categories the user can start spending and BeanPay will track their transactions and automatically categorize their spending.

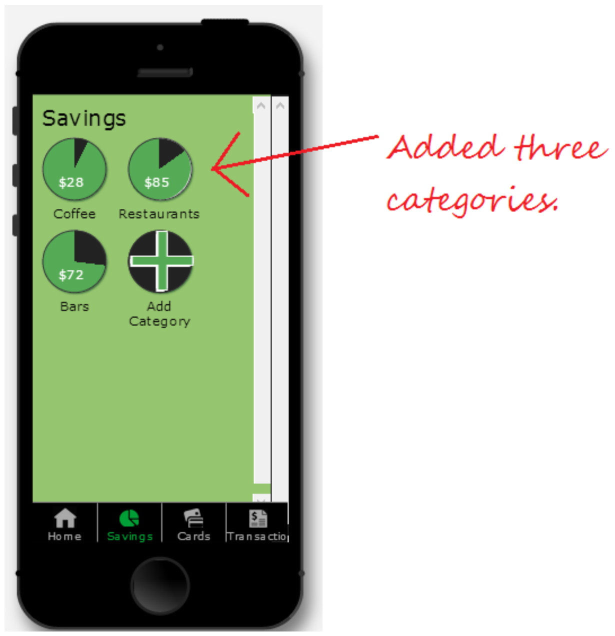
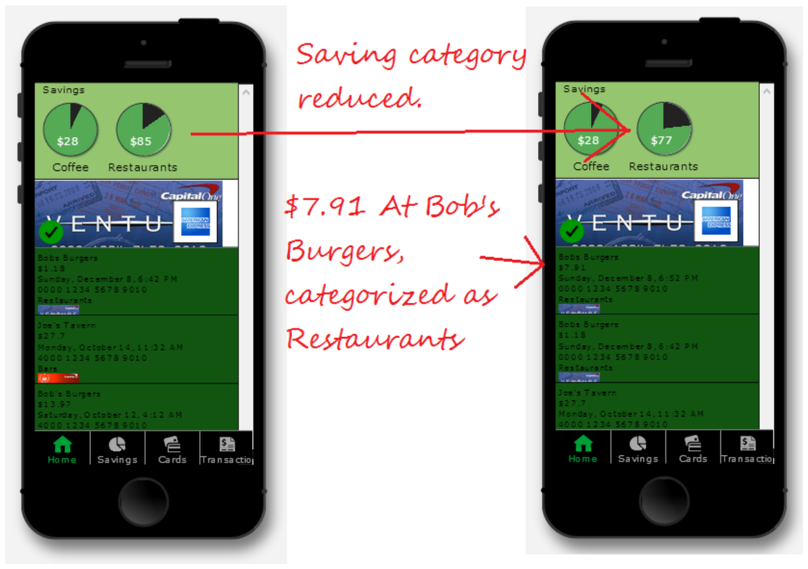


Figure 17

We have labeled our categories "Savings" in order to incentivize a user to save. At the end of each budget period, any money remaining in a category is reported as saved. However, the reporting functionality is not fully implemented in the interactive prototype.



Bobs Burgers
$1.18
Sunday, December 8, 6:42 PM
0000 1234 5678 9010
Restaurants

Joe's Tavern
$27.7
Monday, October 14, 11:32 AM
4000 1234 5678 9010
Bars

Bob's Burgers
$13.97
Saturday, October 12, 4:12 AM
4000 1234 5678 9010

Bobs Burgers
$7.91
Sunday, December 8, 6:52 PM
0000 1234 5678 9010
Restaurants

Bobs Burgers
$1.18
Sunday, December 8, 6:42 PM
0000 1234 5678 9010
Restaurants

Joe's Tavern
$27.7
Monday, October 14, 11:32 AM
4000 1234 5678 9010

Figure 18

As the user continues to make purchases, the category pie charts on the home screen are updated with the most recent purchase. In the above diagram the user starts with 85 dollars remaining in the restaurants category. After the user makes a purchase at Bob's burgers for $7.91, the restaurant category is reduced to $77 (rounded) saved.

Tools used to prototype

Once we completed our paper prototype we felt that an interactive prototype would be most easily implemented using the flexibility of a web page. In order to accomplish this we used several common web development frameworks and tools:

1) GitHub/Git for revision control and shared source amongst team members.
2) Node.js as an http server to serve the web pages.
3) Node.js – npm (package manager) to install various node packages:
    a. Bower – twitter web package manager
    b. Connect – middleware framework for node.js
    c. Less-middleware – more middleware for node.js
4) Backbone.js – Model/View web framework
5) Underscore.js – JavaScript utility library
6) jQuery – JavaScript library for html document traversal
7) Require.js – JavaScript file and module loader
8) Requirejs-text – JavaScript text resource loader

How the tools helped

By leveraging existing JavaScript libraries such as backbone and jQuery we were able rapidly build a prototype which focused on user interaction as opposed building frameworks. Since all of the tools used are available with various forms of open source licensing we were also able to build our prototype with no expensive software requirements.

Another positive factor in using open source web development libraries is that JavaScript is platform independent – We had team members using varied platforms and they were all able to pull the source code from the project github page and build/run locally. This greatly improved collaboration and development speed for the project.

How the tools didn't help

One limiting factor was that the extent of libraries and tools required for this project meant that not all team members had equal level of proficiency. While the team was able to distribute tasks equally, the level of proficiency required for some of the libraries meant that some tasks weren't able to be shared as effectively amongst team members.

What was left out and why

Most of the primary functionality from our paper prototype was implemented in our interactive prototype. The one minor area where the interactive prototype is limited, is in its ability to add custom categories – all the categories in the interactive prototype are preset.

There are a few areas where functionality was not implemented in both the paper prototype as well as the interactive prototype. The primary features we would like to add if given the time would be:

1. Ability to Store receipts
2. More customizable category tracking
3. Credit card reward points
4. AI for figuring out which credit cards to use
5. Notifications to support savings goals
6. Gamification elements

Summary discussion of project and lessons learned

From idea inception through contextual inquiry, paper prototyping, user testing, and ultimately interactive prototype – BeanPay underwent an interface transformation resulting in a tool that attempts to solve the problem of the complexity of personal savings. While our "final" solution still has much work to be done, there are some key learnings that resulted from this exercise.

Beginning with our contextual inquiry, we observed many of the assumptions we had made regarding how people spend and monitor their spending was wrong. Many people don't really consider their day to day spending. It was this discovery that led us to the notion of the spending/budget feedback loop. If we could simplify this process maybe people would be more aware of particular areas that they could improve their savings.

From contextual inquiry we set out to solve a twofold problem – simplify payments, and improve the budget/spending feedback loop. As we considered this twofold problem we constantly asked ourselves what interface elements were required to present the most relevant data. It was here that we came up with the transaction timeline and focusing on only the savings categories that were most relevant to the users discretionary spending.

Once we had a useable paper prototype we began user testing and quickly discovered that tasks we thought were self-evident were not that intuitive to users. As a result we made many changes to our user interface.

Ultimately our interactive prototype included much of the feedback we received from our contextual inquiry and user testing. We learned that solving personal budgeting is a hard problem to solve. We also learned that through the process of applying common HCI principles such as contextual inquiry, user testing, and paper prototyping we were able to converge on a solution that would be applicable to many users.

Appendices

Appendix A - User Testing notes

November 17, 2013

Participants:

The first participant was a 22 year old woman working for a clothing retailer as an assistant merchandiser. She spends around $800 / month on food and while she has mint.com set up, she hasn't changed or updated a budget since college. She hates looking at mint.com because of "how bad she is doing" on keeping her budgets.

The second participant was a 31 year old male electrician. He is very spending conscious and the spending he does that most worries him is around $250 / month on gas. He does not use any formal budget tracking software but wishes he could understand his spending better.

The third participant is a woman in her early 40s working as a manager for a local coffee shop. She doesn't have time for budgeting but wanted to have more control of how much money she spends. She thinks she spends a lot on baby needs food,
diapers, etc. but really has no idea
and wouldn't want to cut down on these purchases anyway.

Results:

Main findings are areas where all three participants made similar suggestions or had similar problems:

● All three participants wanted to be able to create their own categories. They all felt that the drop down category listings were too limiting (even if there were more options) and wanted to create their own custom titles.

● They were all a bit confused about which card they were supposed to swipe to add to the system the
BeanPay card or one of the credit cards. Connecting the BeanPay card to
the mobile device was a missing part of the process for them.

● There was confusion about what the budgets did in relationship to the cards. One participant thought changing the card on the home screen would change the transactions showing. Another thought that when they paid their credit card the transactions from that card would disappear. The third thought that the transactions would be cleared at the beginning of a new budget period.

● None of the participants said they would use this system as is. Reasons included: it is not convenient to have to use your phone to change the credit card being shown on the BeanPay card, not wanting to see or use budget information, not wanting to do the work of setting up budgets, wanting more information about budgets before and after transactions, and not seeing enough value in the system.

Secondary findings are areas where improvements could be made but may be specific to those user's use cases:

● Two of the participants thought that the budget periods would be the same across budgets.

● Two participants had confusion about the menu options one
with what they meant and
the other with why the options moved around and the back button being in the wrong place.

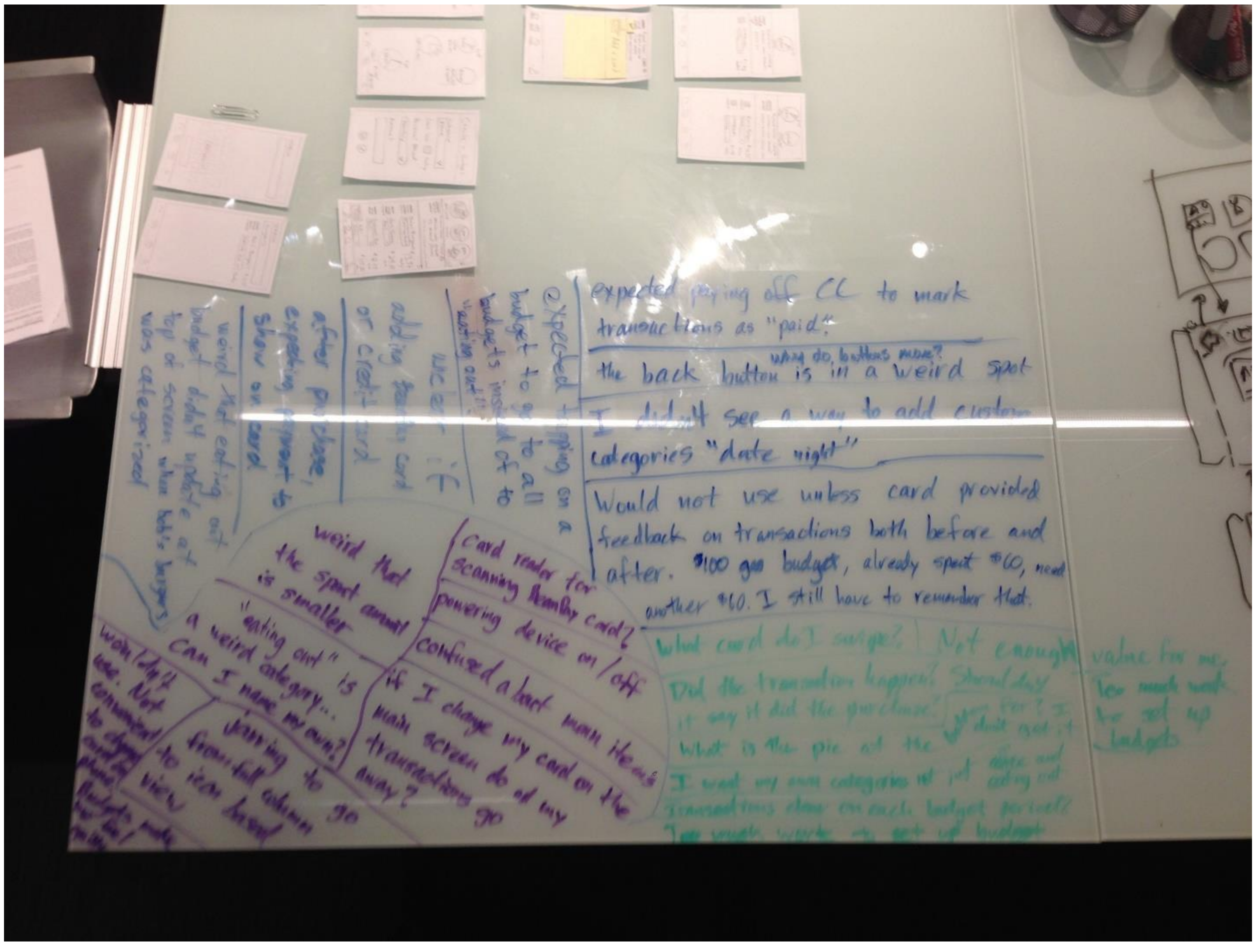
● One participant was confused by the pie chart having the spent amount be a smaller radius that the amount remaining.

● One participant was unsure if the BeanPay card had worked because it did not show a completed transaction on it after it was swiped "
shouldn't it tell me it did the purchase?"

● One participant was confused that tapping on "Eating Out" on the main screen took him to the "Eating out" budget instead of to see all of the budgets.

● One participant was unclear what the pie chart was at the bottom of the budgets screen was for.

Appendix B – User Testing Whiteboard notes



expected paying off CC to mark transactions as "paid".

the back button is in a weird spot
why do buttons move?

I didn't see a way to add custom categories "date night"

Would not use unless card provided feedback on transactions both before and after. $100 gas budget, already spent $60, need another $60. I still have to remember that.

What card do I swipe?

Did the transaction happen? Should [illegible] it say it did the purchase?

What is the pie of the [illegible]

I want my own categories [illegible] date and eating out

[illegible] transactions [illegible] on each budget [illegible]

Too much work to set up budgets

Not enough value for me. Too much work to set up budgets

expected tapping on a budget to go to all budgets instead of to "eating out"

unclear if adding BeanPay card or credit card

after purchase, expecting payment to show on card

weird that eating out budget didn't update at top of screen when bob's burgers was categorized

weird that the spent amount is smaller

Card reader for scanning BeanPay card?

powering device on/off

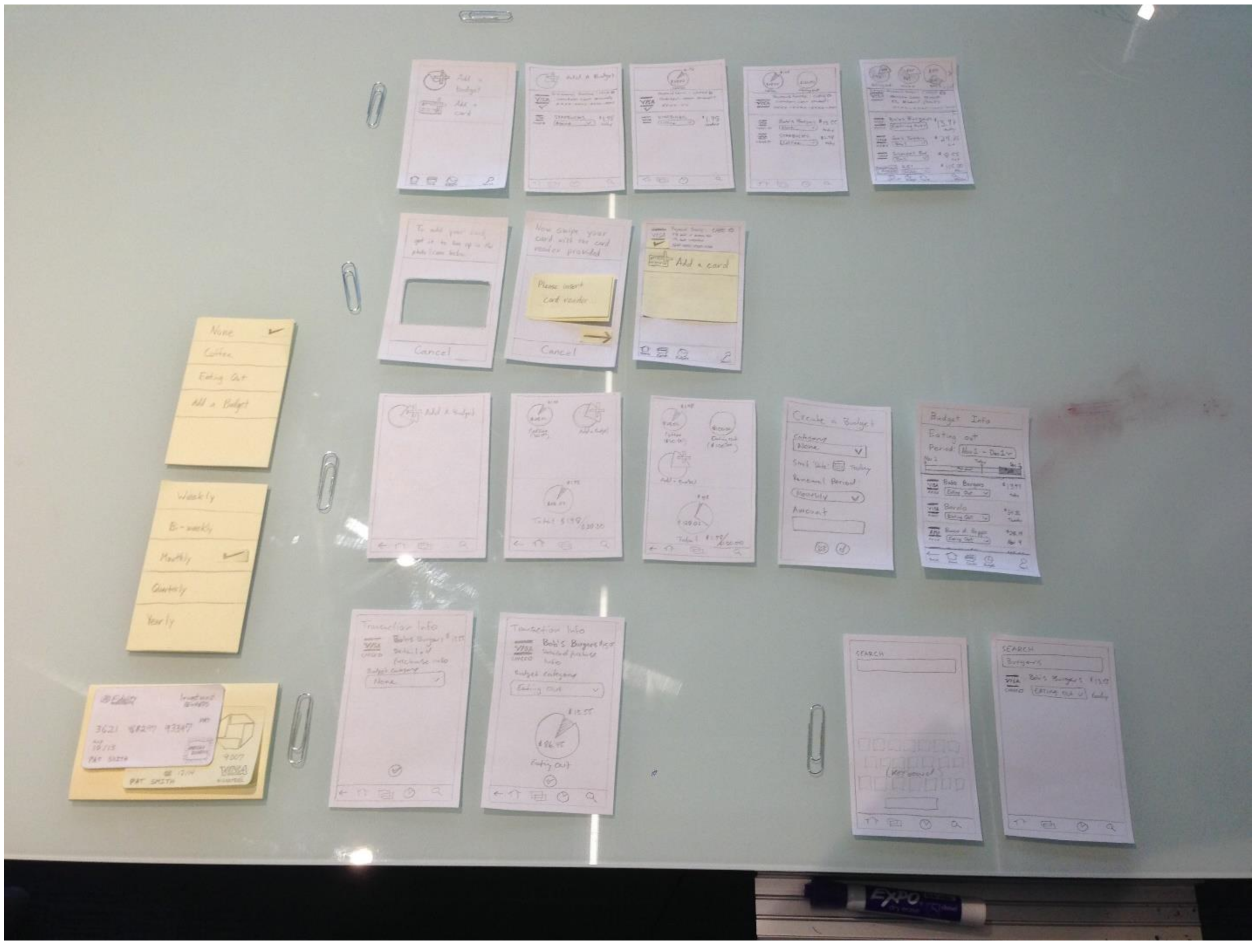
confused a bout main items

"eating out" is a weird category... Can I name my own?

if I change my card on the main screen do all my transactions go away?

jarring to go from full column to icon based view

wouldn't use. Not convenient to change card on phone. Budgets make me [illegible]


None
Coffee
Eating Out
Add a Budget
Weekly
Bi-weekly
Monthly
Quarterly
Yearly
Add a Budget
Add a card
Cancel
Please insert card reader
Create a Budget
Budget Info
Transaction Info
SEARCH

Appendix C – User Testing Supplies