Budding Budget

owsenk   Kyle Owsen

lizs3434   Elizabeth Schibig

ischaaf   Isaac Schaaf

jbktsang   Jessica Tsang

hstefan   Stefan Holdener

mjsc   Maxton Scott Coulson

Process Description

Software Toolset

Since we want our app to be available to the largest audience possible, Budding Budget will be developed using HTML, CSS, and javascript with jQuery, and then ported to native apps for Android, and potentially iOS and Windows Phone, using PhoneGap. This will allow us to reach a wide audience without having to create separate versions for different operating systems, and allows us to create the app using primarily Javascript, an easy to learn and very flexible language.

The web backend that syncs data between the app and the cloud database will use MongoDB with Node.js for the external API. This will give us some flexibility, and integrates well with PhoneGap. Since we are all familiar with GitHub, we plan to use that for version control, as well as some bug tracking.

Group Dynamics

The team is roughly split into a network team, an application team, and a UI team. Issac and Maxton make up the network team, Jessica and Elizabeth are the UI team, and Stefan and Kyle are working on the backend within the phone app.

Since none of us have extensive design experience, or a particular specialization in any of these fields, we expect the membership of each of these groups to fluctuate. Additionally, we believe that more fluid team design - potentially swapping out roles week to week - will allow a more even distribution of the workflow, since teams will likely need more people some weeks than others. The only permanently assigned role at this time in the process is that Kyle Owsen (owsenk) is the PM.

As we only have a six person group, we've opted not to designate a team specifically dedicated to testing. Instead, unless otherwise specified, developers are expected to handle tests of both their implementation details, as well as adherence to requirements.

If there's some disagreement about the requirements or implementation of the app, and the two parties cannot come to a resolution, it will be put up to a team discussion and vote at our next meeting. If a consensus cannot be reached, both teams will come up with an appropriate game of skill or chance to decide the outcome.

Schedule/Timeline

Week of 1/18: Discuss and finish software requirements specification
Week of 1/25: Software design specification & Design presentation slides. Research
more technologies and think about integration
Week of 2/01: Practice presentation & Finish coding for zero-feature release
Week of 2/08: Zero-feature due, continue to code for days, flesh out features
Week of 2/15: Finish beta release, prep for beta demo
Week of 2/22: Feature complete release due
Week of 2/29: Release candidate due
Week of 3/07: Final release due, prep for presentations

Risk Summary

Our biggest risk is that the user will not find the app compelling. With most apps, if the user does not use them for days or weeks at a time, they can come back to the app later with little or no issue. If they skip budget tracking for just one day, the information given by the app is no longer valid.

We plan on mitigating this risk by forcing the user to either update their assets, or confirm that the budgeting information is still correct if they miss a day of tracking. We plan to do extensive user testing so the tracking is easy and appealing to the user. If we are unable to overcome this risk, our app will not be useful, which is why it's the most important risk.

Another potential risk lies with the algorithm used to track spending. Though we don't believe it will be too complicated, the number of variables and potentially vague ending date of the calculation could make it more complex than we think.

We can manage this risk by creating the algorithm early in development, so we can adjust the direction our development goes based on any shortcomings in the process we create. If we

cannot make a satisfactory algorithm, we'll have to cut back on some of the more ambitious elements of the app, like an indefinite end date, or considering recurring charges.

The addition of a cloud sync component to the application adds a new risk - there's a chance that if a user has two devices connected to the same account, their information could get out of sync and confuse the database.

Our plan for this is to settle conflicts based on whatever syncs to the database first, and to force the other device to pull data from this sync. Since we see the user tracking a budget on multiple devices at the same time as an edge case, this solution being less than elegant from a user perspective is fine so long as the database remains consistent. If we cannot make this work, we'll need to either only allow one device to be associated with an account at any one time, or cut the online feature entirely, but these options should only be seen as a last resort, doomsday scenario.

Another risk we are encountering while developing the app is that there are a lot of moving pieces, and a lot that can go wrong. Between calculating the dates, syncing data with the network, and communicating through a poorly defined interface to the phone's notification scheduler, there's a lot that can go wrong.

The best way to mitigate this is to test thoroughly, and to have a very clear idea of what we can easily cut without gutting the usefulness of the application if we run out of time.

Design Changes and Rationale

The most major changes to the app are our choice of Javascript framework and server-side database language.

We determined early on in development that Angular was far too heavy for what we were planning on doing. For a relatively small application like ours, it would likely end up complicating the code rather than keeping it clear, and it would require us to all learn a new and complex framework. Instead, we chose to go with a more traditional callback-based structure with DOM manipulation handled by jQuery.

The switch from SQL to MongoDB was inspired by looking at how the data object was structured and passed around. We decided that the flexible schema it offers is more important than any potential speed gains SQL has.