Iterative Design Process

Solution one: The first solution we considered was one that used very little networking whatsoever. It would require the user to create an account on our website, which would then provide all the financial management services we offer, as well as a native program that the user could download. This native application would have all the functionality of the website. The financial report and budgeting details of the user would be exported to a single file that the user could then download onto their computer. This file would NOT go to our servers. This would eliminate the need for external servers that stored sensitive financial data of customers. The customer would then be able to open their financial management file with either the website or the native application. This would be useful in case either the customer did not have internet access or if our own servers went offline. However, there were a few issues with this idea.

The main issue was its complexity for the user. The user would be required to download a file that contained their financial information, as well as another application. While this may seem trivial to someone with an understanding of computers, to a layman with little technical knowledge, it may be considered too complex to be worth using. They would need to keep track of a file (or multiple files if they have multiple reports generated) that would contain all their information, with no way of recovering them should they be lost or corrupted. If their hard drive dies, their computer gets a virus and becomes unusable, or if the files are accidentally deleted, all of their data will disappear. Additionally, it would only allow the user to see their data from the device(s) they have their files stored on, which may be an inconvenience if they want to check it on at the store while on their phone. While this solution does provide improved security due to sensitive financial information never leaving their computer, the process is not streamlined enough for the average user and has too many ways of the customer losing access to their information.

Another issue is the lack of communication. We would be unable to send live reports to the user about the status of their budget, as we would not have this information. We would be unable to notify the customer if they were close to reaching their budget for the month, or had exceeded it. This would diminish the quality of our service.

There is another issue with this design. It would make the user experience rather clunky, constantly having to upload and download new files and reports, rather than being able to see a complete record of their budget immediately upon login. The lack of streamlining would go against the idea of making this a *simple, easy to use* budgeting and financial management tool. It would be much more convenient for the user to login and immediately be able to access everything they want to see, no matter where they are or what device they are using. This leads us into the second proposed solution.

Solution two: The second solution we considered was a solution that included a web-based application, as well as support for native mobile and PC applications that did not require the website to access. They would both still require constant internet access, and customer reports would be kept on our servers. It would allow a user to create an account that could be used on both the website and the native application. This would have functioned similar to a variety of google products. For example, you can use gmail on your phone through the gmail app, but also on the gmail website. This would provide specialized user interfaces for both viewing formats, leading to a cleaner look and greater convenience, as well as a way to still use the application if the web-servers temporarily went offline. This eliminates the problems from the first solution by allowing real-time updates to the customer (likely via email) as well as ensuring their information cannot be lost due to user error. However, we still decided against this idea for a few reasons.

The main reason for this idea not being viable is the cost. The cost of developing a web-based application, as well as native applications for mobile devices and PCs would be too great for only a minimal increase in customer satisfaction. It would require development in multiple languages for both iOS and Android operating systems, but would also require continual maintenance to accommodate frequent OS updates, whereas the functionality of websites has been very stable throughout the years, regardless of operating system updates. Also, the PC application would be redundant, as users with an internet connection would likely be able to connect to the website anyways. Although it would be a greater convenience (especially if our seb-servers went offline), the increased development cost cannot be justified. It would be better to devote those resources to server maintenance to ensure that does not happen.

Another solution would need to be considered. One which reduces costs while still maintaining the functional components of our program and providing a simple, streamlined user experience.