****

*Budget Manager iOS Application*

*Reflection Document*

1. **Other choices of the tools:**
   1. **Cocoa pods**

CocoaPods is a dependency management tool for iOS and OS X development that has been gaining a lot of traction in the Cocoa community. When using CocoaPods, you define your project’s dependencies in a file, which CocoaPods uses to determine and resolve dependencies between libraries before downloading the needed source code and linking it in an Xcode workspace.

Advantages: By using Cocoa pods we can directly link the third party libraries which are helpful for drawing the bar graphs and pie charts in reports tab.

Disadvantage: Only limited support for configurations that is it uses only libraries in Debug config.

* 1. **PaintCode 2**

PaintCode is a drawing app that turns your drawings into Objective-C, Swift or C# drawing code in real time. The application window is divided into sections that provide you with quick & easy access to everything you'll need.

Advantages: In our application we have used many symbols and date pickers, by using this application it would have been a great help as it automatically creates swift code for symbols and pickers.

Initially we have developed static bar graphs and pie charts that is these reports were not linked to the database. In that case as well PaintCode 2 would helped us for the automatic code generation for the pie chart and bar graph.

* 1. **Fluid UI**

Fluid UI is browser based prototyping tool by which we can develop the prototypes for the application. We have used an online tool called moqups for designing the prototypes, but Fluid UI have provided many advantages over moqups they are

Advantages:

* The user interface for Fluid UI is more better than that of moqups
* There is no limitation for the number of objects to be used in a project in Fluid UI where as in moqups we can use only 300 objects

But there were few disadvantages for Fluid UI over moqups

Disadvantages:

* We can create prototypes only for 9 screens without upgrading that is as a free user but coming to moqups there is no limitation for the screens
* We can only create one project free without upgrading the account but coming to moqups we can create 2 free projects

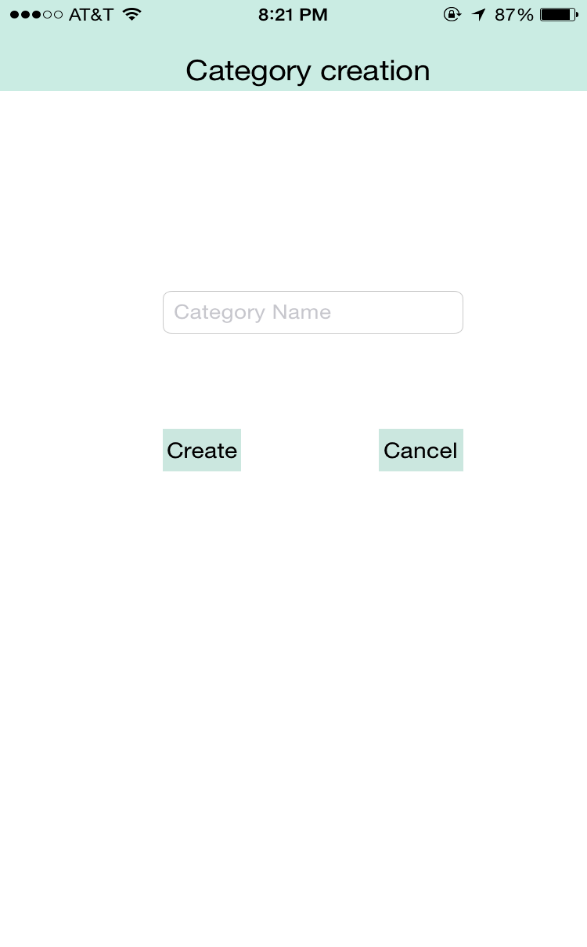
**d) Appium:**

Appium is an opensource test automation framework for mobile applications. Appium is one of the best tool available for automation testing. Appium aims to automate any mobile app from any language and any test framework, with full access to back-end APIs and DBs from test code.

In Appium we need to record all the navigations using the record option in the panel. Then all the navigations are recorded and a test script is generated. This test script is to be added into a java class. Now when we run the application this java code will also get executed and test will be done.

1. **Incomplete Functionalities:** We have completed all the high priority functional requirements and also completed the medium priority functional requirements but completed only few low priority functional requirements. The other few low priority functional requirements that were not completed are:
   1. **Choosing an icon for categories:** In Budget manager application categories is an important field. The user need to have categories created for creating a budget for a particular month. As developers we will be providing few default categories to the users and also will give an option to add the new categories. As of now we have provided icons for the default categories and a same icon for all the other categories that are being created by the user.

So now the functionality that is incomplete is choosing an icon for a category at the time of creation instead of having a default image. For that we need to add a button in add categories screen shown below saying “Choose” and whenever it is clicked it should be redirected to the gallery of the mobile for selecting the image.

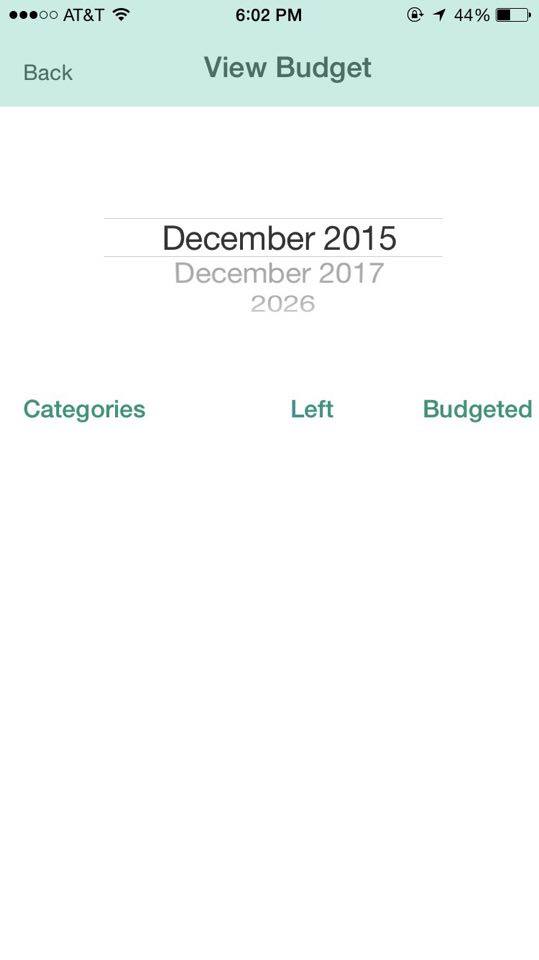
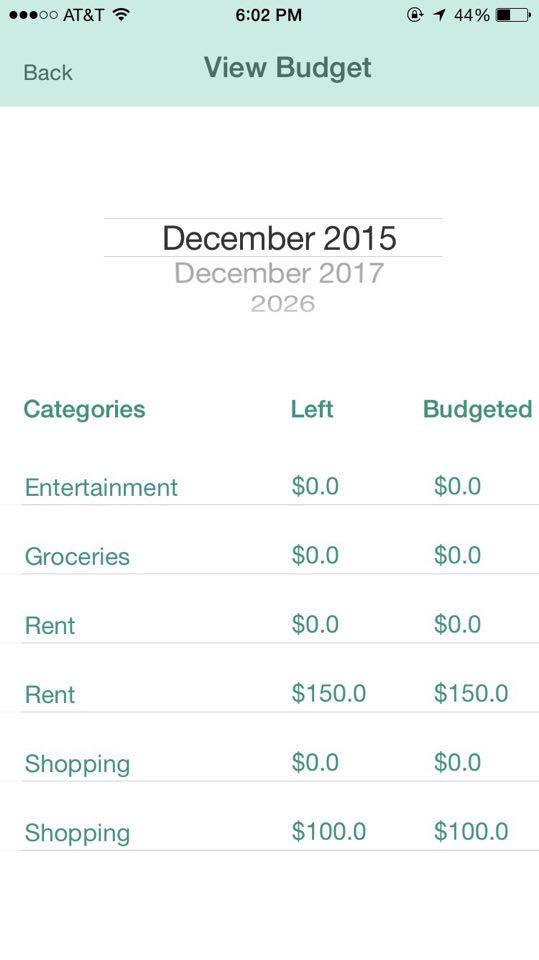


* 1. **To add previous months transactions:** In Budget Manager iOS application the first screen that appears after logging in and selecting the type of the account is the record expenses screen which list the record of that particular month. Here we can add transactions for the present month only. This was the earlier requirement of the client and hence we have designed for entering transactions only for that month.

But in the later stages the client has asked to add transaction for the previous month as well. We as a team has think about it and told the client about the difficulty that we are going to face as we need to completely change the design of the application. Client has asked to do it at the last if the time permits and made it as a low priority, and we were unable to do it as we need to change the entire design of the record expenses tab.

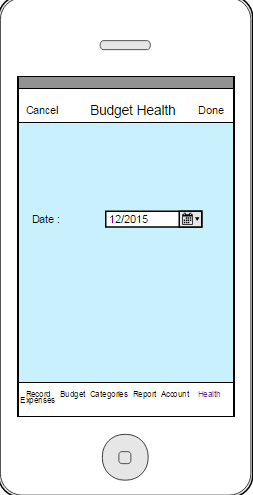
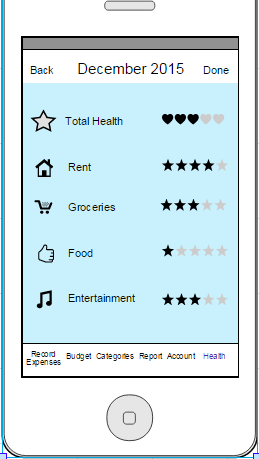
* 1. **View the budget for the current month directly without selection:** The users create a budget in budget tab and can view the created budget by clicking the view budget in the budget tab. Not only the particular month’s budget but also users can view the transactions under each category in the view budget.

To view the budget the user need to click on the view budget button the he/she is navigated to the view budget screen where he can view a month picker at first and an empty table below the picker at first. The user need to select the particular month to view that particular month’s budget, then a table is populated below the screen as shown below

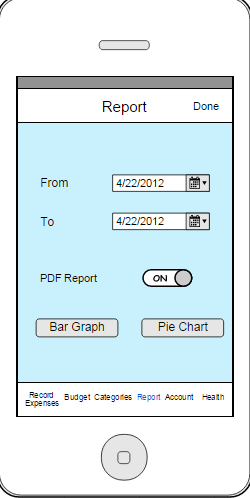
 

But the low priority requirement of the client was to populate the table directly without selecting the month and year. By doing this there will not be any design issues and the implementation is also not difficult but this was found in the last demo and we did not have type to implement this functionality.

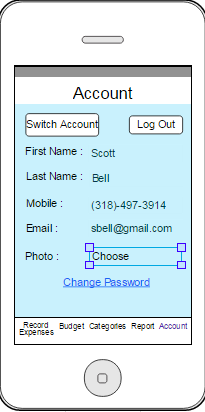
1. **Functionalities for Budget Manager 2.0:**
   1. **Budget Health:** We are proposing a new tab called budget health. The major functionality of this tab is to provide the health of the budget for a particular month that is how well a user is using his budget and how accurately the user has created his budget for all the categories. The total health of the budget which includes all the amounts budgeted and left for all the categories in that particular budget is represented in percentage and health of each category is also represented in percentage. If the health exceeds 100% then the user has spent more than budgeted. In this way budget health will allow user to track his entire budget on a whole.

** **

* 1. **Monthly expenses report:** As of now we are providing monthly reports in the form of a bar graph and a pie chart. In the next version of the app in reports tab we will give an option for the user for a monthly pdf report to be mailed to his mail id. In this pdf report we would provide the total amount budgeted and the amount budgeted for each category, savings for that month if any or losses for that month, List of expenses for each category, List of categories in which the money is left, List of categories in which the money is over spent.

****

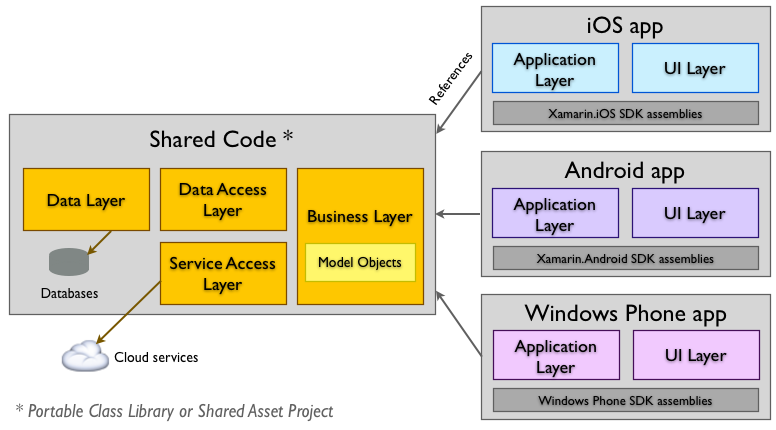
* 1. **Choose profile picture:** Budget Manager is single user application that is each user has his own categories and budgets. There is no option of selecting a unique profile picture for the user. So in the next version of the app we can provide an option for selecting a profile picture. For this we need to add a button in the accounts screen and also provide place to display the image. Coming to the database side we need to add a new attribute named “propic” to the accounts table.

****

* 1. **Reduce dependency:** In the first version of Budget Manager there are few dependencies. That is the app is dependent on the Budget Manager Web application for adding a new user, forgot password. So in the next version we can provide options for creating a new user and forgot password at login screen and change password option in the accounts tab.

* 1. **Bug Fixes:** The bugs which were found in the first version of the Budget Manager will be resolved in the Budget Manager Version 2.0. The main bugs that are going to be resolved is in view budget that is viewing the budget for the current month. And also all the low priority requirements which are not done are needed to be completed in version 2.0.

1. **Design/code impact while implementing an interface:** Our application follows MVC architecture by combining view and model together and hence, it is multi layered architecture.
2. Presentation Layer (View controllers in Swift iOS)
3. Business Layer and Data Access Layer (Business logic is handled in PHP scripts)
4. Data Layer (MySQL database).



Since, we have followed MVC architecture most of the our application code is reusable, flexible and extendable, i.e. when client asks to implement application with other interface for virtual and augmented reality such as voice command, google glass, etc. Since we have single layer, all the interfaces vary in the way user constraints are initialized. If we place the user constraints in the presentation layer and add the code.

Whenever, the user gives constrains with the interface. Code will set the parameters with specific constraints and embed it in the URL. URL will invoke the PHP script and query the database and return the JSON object to presentation.

Design won’t be impacted due to new interface. We have to add the library which handle the interface and save the user inputs in the parameters and hit the database. The major code changes will be in presentation layer which handles the interface to get the user inputs. Hence, our code accommodates implementation of new interface by adding the code to setting the constraints with the user inputs with less effort.

1. **Data stores alternatives:** The amount of data is growing rapidly, and the nature of data is changing as well, as developers find new data types most of which are unstructured or semi-structured and people want to incorporate this data into their applications. All the networking sites go with unstructured data storage rather than RDBMS. Research firm IDC estimates that in 2013 the combined size of the world’s digital data was 4.4 zettabytes — i.e. 4.4 trillion gigabytes and that by 2020 it will grow ten times to 44 zettabytes.

NoSQL technology was pioneered by leading internet companies to overcome the limitations of 40-year-old relational database technology for use with modern web applications. Today, enterprises are adopting NoSQL for a growing number of use cases, a choice that is driven by four interrelated megatrends: Big Users, Big Data, the Internet of Things, and Cloud Computing.

Our application is compatible with the usage of NoSQL as we store and retrieve the data on web services using JSON object. Since NoSQL is a document type storage which has BSON data our application accommodate data store changes by changing the logic in PHP code and change in structure of the data layer. There will not be any changes in the presentation layer as the data it sends and receives data in form of JSON object (NSArrays/ NSDictonary).

1. **Performance during heavy Usage:** The applications takes the request from the user and hits the database. Later, it will return result set as JSON object. When user requests increase, database hit ratio will increase, which can impact the performance. In order to reduce the burden on the database, we are loading all budgets, categories of the user into a variable which is in AppDelegate when the user login is successful. Since, budget information is stored in the AppDelegate.  Heavy usage is handled by reducing the user requests and maintain short user sessions.

When the user wants to view the information, application will not hit the database rather it will check in the temporary variable in the AppDelegate. Hence is protected from multiple requests from users. Request will only hit the database, when an entry need to insert/update by the user. Since the burden on the database is reduced the performance is maintained even in the peak time of usage.

We are implementing PHP at the business layer. In PHP code, when every a user request hits database user session is created and will be closed after the request is processed. Since, user session is will be closed at the end of the PHP page, session will be temporary and will persist only for faction of milliseconds. Since the user requests will persist for few milliseconds, it will have the ability to serve more request comfortably.

MySQL can handle a maximum of 4294967295 which is specified in MySQL reference manual. We would probably run out of memory, file handles, and network sockets, on your server long before you got anywhere close to that limit. Practically, if we have around 10,000 users totally not all the users will be hitting the database simultaneously. If we have too many users trying to hit the database as part of future enhancement we can limit the number of active connection in the application layer just by putting some threads to sleep there by reducing the active users and increasing the performance.

1. **Secured Data:** The user requests are sent as the parameters using a URL. All the parameters are embedded in URL. This will invoke the database and wrap the result in the form of JSON object and responds to the user request. The URL will not be visible which assures that the data sent and received is secured.

If an attacker tracks the requests on the server, there is a chance of attacking the server. In order to make the data more secure we can implement POST instead of GET request. When the POST is implemented the parameter will be encrypted and even if the attacker traces the requests the actual data is not visible and security of the data storage is maintained.

We can implement POST request with minor changes to our code in presentation layer and business layer. While, using the POST request method, all the parameters are embedded into an NSDictionary variable and is sent to the business layer (PHP code) and just by decoding the JSON object we can get the parameters and get the result set.

1. **Bad inputs:** Our application works well with bad inputs also. We have provided validations for almost all of the fields.

User can login to our application if he enters valid e-mail id and password. If he tries to enter wrong credentials, he cannot use the application as there are certain validations where e-mail and password match is checked dynamically.

We have provided validations for all the user input fields. If the user forgets to input data for a particular field, an alert message is shown on the screen which says that the user need to fill all the fields.

We have also provided validations for the numeric inputs such as the user will not be allowed to enter any special character or alphabets where a numeric number is to be entered, so there is no chance for any SQL injection attack for the hackers instead of numeric inputs.

Our application stands against SQL injection attack as we have used parameterized queries with PHP data objects.

1. **Automating the testing:** Now a days, it has become a challenge for any company to maintain quality and efficiency of software projects. Due to many time and cost constraints, testing is neglected and as a result the quality of the product is also reduced. In order to improve the quality by taking time into consideration, automated testing being performed.

In our application, we have performed testing as per the test plan. We were able to test the application manually by performing various kinds of testing like unit testing, usability testing, integration testing.

In the unit testing, we tested each and every unit of the application and tried to fix bugs immediately as it occurred. We tested the application in multiple runs and tried to find out more bugs and solved the issues.

In the usability testing, we asked two different users to test our application and give the report like where we need to make changes as to make the application uses-friendly.

Unfortunately, we could not perform automated testing on our application. We thought of using Appium tool in order to perform automation testing. We tried to perform testing by using the Appium tool, due to time constraints, we could not complete do it.

We need to perform testing after the development of each and every module and try to fix the bugs immediately so that there will not be any problem in the next phases. We need to perform rigorous testing by giving various kinds of inputs and try to fix the issues.

The lesson we learned from this is that it is very difficult to do regressive manual testing as we need to enter all the inputs and also list the bugs found. So testing of all the applications need to be done automatically by using some tool instead of doing manual testing.

1. **Redo Project:** Firstly, we would have distributed the work we need to do equally in order to reduce the workload on one person and also save time by not spending lot of time on a single part.

We would make sure that the requirements of the client are accurate so that we need not change the application and design very frequently.

If we could redo our project, we could have started with designing of UI screens and try to develop them. For example, in the reports tab, we have two different screens for pie chart and bar graph. There is a segmented control for pie chart where the user can see budgeted and spent screens separately. We could have placed them on the same screen.

We would have implemented choosing an image from the gallery for a category. In our application, in the categories tab, we do not have an option of choosing an image for every category.