BUDGiT

Technical Manual

Michael Pabon: 93385198
Jeffrey McLemore: 98839997
William Bennett: 98691651

Table of Contents

Summary of Original Proposal

The general purpose of this application is to help users track their budgeting habits. The app consists of three major views, some with an additional view related:

1) Accounts Overview

The purpose of this feature is to have users be able to see their financial state at a glance. The accounts balance page will contain each of the user's accounts and allow them to investigate the debits and credits associated with each. Before tapping on an account to see its credits and debits, the user can initially see the account name, number, and balance. When an account is tap, the user will see all the purchases and/or refunds related to an account. The user can set recurring pay dates.

2) Budget List

The purpose of this feature is to give users the opportunity to set limits on the amount of money they can spend on different shopping categories. This entails the creation of budget profiles. Every budget profile has default categories associated with it and the user can create new ones. When creating a budget the user needs to set dollar caps for each category. This specifies the maximum desired amount to spend in a particular field. The user can tap a budget in the list to have it expand and view its caps.

3) Analysis Page

The purpose of this feature is to provide the user with a detailed breakdown of a particular budget profile. The user can select a budget profile and view visual aids to help them interpret where they stand with a particular budget. Past budgets may also be viewed. Alerts will be utilized to notify the user if they are approaching a cap.

Project Updates

Project Update 1

Summary:

The purpose of the application is to help users keep track of their finances. The user will have the option to create a budget based on default or customized categories. The application will have three distinct views which share data and interact with each other. The views are accounts overview, budget creator, and the analysis page. Accounts overview will display a general summary of all the accounts the user has. Budget creator will allow users to create a budget and implement it throughout the application. The analysis page will give detailed information on the user's spending and may include visual representation.

Progress:

By this time, our team proposed that we would determine a method for storage of data. We decided that the method taught in class, Parse, would be the best way for persistent storage. Each group member has become comfortable with using Parse within the Xcode environment including how to input, store, and retrieve data.

Updated Timeline:

By 10/20: The group members will have successfully designed the layout of the application. This will serve as a wireframe for the application which each member can build upon. The team will demonstrate test cases using parse to exemplify their familiarity with the website.

By 11/3: The group members will have successfully completed each views functionality. The team will have gathered together to store the information needed for the application on Parse.

By 11/20: The group members will have successfully thoroughly tested the functionality of their respective view and tested the functionality of each other group members view to ensure compatibility. The group will begin to implement the changes that the team agrees upon for their respective views.

By 12/1: The group members will have successfully integrated each responsibility into a fully functional iOS application. The team members will each run through test cases of the application to detect bugs.

Specific Tasks:

Michael: Develop the wireframe for the budget creator. Create test cases to demonstrate how the different budget profiles will be stored in Parse.

Jeff: Develop the wireframe for the Analysis page. Research methods for charts and graphs in swift. Create test cases for possible analysis page content.

William: Develop the wireframe for the Accounts Overview page. Research iOS calendar integration for recurring pay dates. Create test cases to demonstrate how the accounts overview information will be stored in Parse.

Project Update 2

Summary:

The purpose of the application is to help users keep track of their finances. The user will have the option to create a budget based on default or customized categories. The application will have three distinct views which share data and interact with each other. The views are accounts overview, budget creator, and the analysis page. Accounts overview will display a general summary of all the accounts the user has. Budget creator will allow users to create a budget and implement it throughout the application. The analysis page will give detailed information on the user's spending and may include visual representation.

Progress:

The proposed progress is at this point in time each member of the group has developed the wireframe for their portion of the application. The group has decided on a layout for the project. We have all met since then and demonstrated each of our test cases in Parse to showcase how each view will function. Each user is familiar with Parse and cloud storage. The group is on time with the proposed progress.

Challenges:

The group is on time and faces no challenges at the moment.

Updated Timeline:

By 11/3: The group members will have successfully completed each views functionality. The team will have gathered together to store the information needed for the application on Parse. Test data will be present in the application. Begin testing of functionality with test data.

By 11/20: The group members will have successfully and thoroughly tested the functionality of their respective view and tested the functionality of each other group members view to ensure compatibility. The group members will take note of suggestions and bugs and provide them to the other group members. The group will then implement and complete the changes that the team agrees upon for their respective views.

By 12/1: The group members will have successfully integrated each responsibility into a fully functional iOS application. The team members will each run through test cases of the application to detect bugs. The group member will upload the application and run tests cases on actual iPhones.

Updated Specific Tasks:

Michael:

- Develop the wireframe for the budget creator.
- Create test cases to demonstrate how the different budget profiles will be stored in Parse.
- Examine current budget apps on app store to gain more insight
- Determine a way to keep track of current user budget profile across all views on app

Jeff:

- Develop the wireframe for the Analysis page.
- Research methods for charts and graphs in swift.
- Create test cases for possible analysis page content.
- Research cutting edge financial metrics for implementation in app

William:

- Research further Parse documentation and cloud storage
- Develop the wireframe for the Accounts Overview page.
- Research iOS calendar integration for recurring pay dates.
- Create test cases to demonstrate how the accounts overview information will be stored in Parse.

Project Update 3

Summary:

The purpose of the application is to help users keep track of their finances. The user will have the option to create a budget based on default or customized categories. The application will have three distinct views which share data and interact with each other. The views are accounts overview, budget creator, and the analysis page. Accounts overview will display a general summary of all the accounts the user has. Budget creator will allow users to create a budget and implement it throughout the application. The analysis page will give detailed information on the user's spending and may include visual representation.

Proposed Versus Actual Progress:

The proposed progress is that the group members will have successfully completed each views functionality. Additionally the team will have gathered together to store the information needed for the application on Parse. Test data will be also present in the application, and the group members will begin testing of functionality with test data.

The actual progress is that the group members are facing difficulty with creating their own individual views. William Bennett is facing difficulty with iOS calendar integration and needs to work on understanding Parse and cloud storage a little better. Michael Pabon is facing difficulty with managing budget profiles, such that when the user has multiple budgets, his view has trouble recognizing which budget the user has selected. Jeff McLemore is facing difficulty with learning how to integrate features such as diagrams and graphs to depict data on the financial metrics view.

The Challenges That Prevented Us From Meeting Our Goal:

All 3 group members had 3 other tests and a 3 other projects all due within a span of 2 weeks. This has pushed back progress on this app. Because of the lack of time it was difficult to research and analyze new documentation for Swift thoroughly to create a fully functional and integrated app.

How We Intend To Get Back on Track: The group members will re-evaluate the timeline and adjust according using SMART goals. The timeline will remain largely the same due to lack of progress from last update. The group member will cross-examine test dates and home-works and projects from other classes and create a timeline that synchronizes well. The group will make goals that are specific, measurable, attainable, realistic, and timely.

Updated Timeline:

By 11/18: The group members will have successfully completed each views functionality. The team will have gathered together to store the information needed for the application on Parse. Test data will be present in the application. Begin testing of functionality with test data. The works well with each of the group members classes and falls in some down-time, which allows each group member the time to analyze more Parse documentation.

By 11/25: The group members will have successfully and thoroughly tested the functionality of their respective view and tested the functionality of each other group members view to ensure compatibility. The group members will take note of suggestions and bugs and provide them to the other group

members. The group will then implement and complete the changes that the team agrees upon for their respective views.

By 12/2: The group members will have successfully integrated each responsibility into a fully functional iOS application. The team members will each run through test cases of the application to detect bugs. The group member will upload the application and run tests cases on actual iPhones. Furthermore, the group will ask friends and family and peers for feedback on the application and implement and possibly implement changes if time allows.

Updated Specific Tasks Before 11/18:

Michael:

1.) Determine a way to keep track of current user budget profile across all views on app. This is the largest setback and task. 2.) Create test cases to demonstrate how the different budget profiles will be stored in Parse. 3.) Examine current budget apps on app store to gain more insight

Jeff:

1.) Research methods for charts and graphs in Swift documentation and other libraries. This is the largest setback and task. 2.) Create test cases for possible analysis page content 3.) Further research cutting edge financial metrics for implementation in app

William:

1.) Research further Parse documentation and cloud storage, in addition to researching iOS calendar integration for recurring pay-dates. This is the largest setback and task. 2.) Create test cases to demonstrate how the accounts overview information will be stored in Parse

Project Update 4

Proposed Versus Actual Progress:

The proposed progress is that the group members will have successfully completed each views functionality. The team will have gathered together to store the information needed for the application on Parse. Test data will be implemented and the group members will have successfully and thoroughly tested the functionality of their respective view and tested the functionality of each other group members view to ensure compatibility. The group members will take note of suggestions and bugs and provide them to the other group members. The group will have completed any recommended changes.

The actual progress is that the views are not yet fully functional, so the proposed progress is not completed. The team is working together collectively to complete the views at the current time as quickly and effectively as possible.

Updated Specific Tasks Before 11/30:

Michael:

1. Determine a way to delete the activated budget
2. Determine a way to make a distinction between editing the active budget and creating a new one

Jeff:

1. Research methods for charts and graphs in Swift documentation and other libraries.
2. Create test cases for possible analysis page content
3. Further research cutting edge financial metrics for implementation in app

William:

1. Continue researching Parse documentation and Swift language documentation
2. Research human-centered software development for aesthetic and design purposes

The phrase “specific tasks” is used loosely. The team is working collaboratively at this point on each view. The plan for completion is to work on this during the week of Thanksgiving break as the group members will be in a nearby location over the holidays.

Contingency Plan:

Regarding the accounts overview page:
The group has decided that the purpose of the app is strictly for budgeting purposes. Originally a hybrid banking-budget app was proposed. However, complications involving learning how to sync with bank accounts have caused the group to disregard the banking features, such as setting recurring pay-dates.

The accounts overview page will be table-based and will be the central location where the user can debit or credit toward their respective active budget.

Regarding the budget creator page:
The budget creator page is largely working as intended. Instead of clicking on a budget profile to expand it, a “Current Budget” button will be available to segue to a profile page in which user can set details and caps for the budget. The largest setback and overhaul is the user cannot dynamically add their own categories, they must use the built-in categories. The budget creator page is table-based. Selecting a budget will add a “check-mark” to it, indicating that this is the active budget.

Regarding the analysis page:
The analysis page is currently the largest setback. Because the group members have worked on the app collaboratively, the analysis page is yet to be created and is still in the process of being designed. The group is collectively working together to research methods for implementing libraries in Swift that allow for financial metric analysis, such as bar graphs and pie graphs. This deliverable may be less than expected and feature a minimum number of graphical analysis when the project is completed. Past profiles will not be cached and visible for use as originally intended at the moment.

Proposal vs. Submission

The most noticeable change in the development of this application was the shift in the general purpose of the application. The team originally agreed upon a multi-purpose budgeting application which included some aspects of banking such as viewing your checking account. After much thought the team decided that combining both a budgeting application with a banking program would diminish the quality of both individually. This led us to the decision to focus our full attention on the budgeting portion of the application, leaving behind the banking aspects (this led to the title BUDGiT!). This meant dropping all banking related features from every view that included them.

Features Removed

1) Accounts Overview (Renamed "Overview")

- ***View accounts*** – This removal was a result of the decision to remove banking features
- ***Tap to examine credits and debits*** – since the cells do not consist of accounts they were replaced by the debits and credits themselves, leaving the overview screen to be a table of purchases/refunds
- ***Set recurring pay dates*** – This removal was the result of the decision to remove banking features. It was no longer necessary to know if someone received a paycheck, what the app strives to keep track of is how they use it.

2) Budget Creator (Renamed "Budget List")

- ***Time period associated with budget*** – While on paper this seems like an important feature, the app proves quite functional without it. Its removal came about because of its overall difficulty. Keeping persistent data storage through Parse was a difficult enough task for the major components of the app so the date feature was put on hold. It never made it to the final version.
- ***Add, edit, and delete categories*** – Similar to the explanation of above, the team decided that keeping track of dynamic, user-created, categories would not be feasible given the time limit and other responsibilities.
- ***Weight based caps*** – The team decided that dollar caps were more intuitive to work with from the user's perspective than relative weights.
- ***Tap to view overview of the budget caps*** - The team felt that this took too much away from the analysis page and decided to include a similar feature (target lines) in there instead.

3) Analysis Page

- ***View past budgets for comparison –*** This feature was removed as a result of the removal of time periods associated with budgets.
- ***Alerts if user is approaching a budget cap –*** This feature was removed and replaced by the target line feature. The user will not be notified if they are approaching a cap, however they can see it before it happens considering purchase input is manual

Features Added

1) Accounts Overview (Renamed "Overview")

- ***Delete a credit or debit –*** This was included in case the user makes a mistake and needs to delete an update they made

- ***Total spent vs total budget summary –*** This was included at the top of the view in order to give a very fast summary of how the user is doing with their budget. The reason for including this is because since this is the opening screen of the application (and probably the most used one), we provided the user with a quick way to see their progress.
- ***Name of active budget as title –*** This was included to inform the user of the budget that is currently active. It proves helpful in the case that the user has more than one budget created on their account
- ***Logout -*** We allow the user to logout in any view

2) Budget Creator (Renamed "Budget List")

- ***Check to select active budget –*** We felt this was an intuitive way to select a budget to work with so it was included in the final cut.
- ***Logout -*** We allow the user to logout in any view

3) Analysis Page

- ***Target lines on bar graph –*** If you tap on a bar in the bar graph it allows you to see the cap for the particular category the bar is associated with. We included this as an easy way for the user to see how they are doing
- ***Total spent in a category ÷ total spent in all categories-*** If you tap on the pie chart, on a particular category, you can see a percentage amount that describes how much you've spent relative to the total amount you've spent. We included this so that the user can see where most of their money is going.
- ***Logout -*** We allow the user to logout in any view

Feature List

Login Screen: Collective Effort

1. **Signup/login**
   Frameworks Utilized***:*** Parse
   Data Structures Utilized***:*** Parse Tables
   This is the only feature on this page. We felt that it was important to create users with the application because we wanted the application to have cross platform potential in terms of iOS. This means local data storage would not do the trick. If the user has multiple iOS devices, we felt they should be able to access their data from any the app is compatible with. Additionally, the use of a user helps narrow down Parse queries in terms of a particular user.

Overview by William Bennett

1. **Add Debit or Credit**
   Frameworks Utilized***:*** Parse
   Data Structures Utilized***:*** Parse Tables, Arrays
   When the plus sign is pressed the user is segued to another view for this feature. The user has the option to add a purchase to the current active budget. If they choose to do so the purchase is added to the "Purchases" table in parse to include, the category chosen, if it was a debit or credit, the amount, the reason, and the current budget that is selected for the user.
2. **Total Spent vs. Total Budget**
   Frameworks Utilized***:*** Parse
   Data Structures Utilized***:*** Parse Tables, Arrays
   This feature was included to give the user a very quick look at their progress with their budget. It was included under the rationale that since this is the first view in the app (and probably the most used) it might be helpful to provide the user with a short budget summary. To retrieve total spent the purchases class is queried to find the sum of all purchases. To find the total of all the budget caps, the budget class is queried for the active budget of the current user and the sum of all the caps is returned. This information has its own column in the "Budget" table.
3. **Delete Debit or Credit**
   Frameworks Utilized***:*** Parse
   Data Structures Utilized***:*** Parse Tables, Arrays
   This feature was included because just as a user is allowed to add a purchase, the user should be able to delete one. This feature uses a swipe button to reveal a delete button for the cell and pressing delete causes a number of things. The purchase is deleted from the Parse "Purchases" table using its object id. The total spent is recalculated and the label is updated to reflect the total amount spent so far.
4. **Active Budget Listed at Top of Page**
   Frameworks Utilized***:*** Parse
   Data Structures Utilized***:*** Parse Tables, Arrays
   This feature was included to notify the user which budget they are adding or subtracting purchases to. The "User" table in Parse is queried because there is a column for every user that states their current active budget. Once retrieve it is displayed in the navigation bar.
5. **Display Debits/Credits**
   Frameworks Utilized***:*** Parse

Data Structures Utilized*:* Parse Tables, Arrays

This feature was included to allow the user to browse through the list of their purchases. It is a given that the user should be able to do this and this is the primary purpose of this view. The "Purchases" table in Parse is queried and formatted into a String to display the information for the purchase. These are sorted by most recent purchase at the top of the list using a Parse method after establishing the "wherekey" of the query.

Budget List by Michael Pabon

1. **Add Budget**

   Frameworks Utilized*:* Parse

   Data Structures Utilized*:* Parse Tables, Arrays

   This feature was added so that the user could create a budget to work with throughout the application. It also allows for the creation of multiple budgets with different names. The "Budget" table is queried in parse so that the caps, and names of the budget can be stored in the table respectively. A "created by" column in this table also stores the object id of the current user so that, when used, a query on the Budget class can be narrowed. The total of all the caps is also stored in this table.

2. **Edit Budget**

   Frameworks Utilized*:* Parse

   Data Structures Utilized*:* Parse Tables, Arrays

   This feature was included because we wanted to allow the user the freedom to change their budget because we feel like editing of a budget is a plausible real life event. This feature works similar to the add budget feature except the text fields are preloaded by querying the "Budget" table in Parse to find all the caps of the current active budget.

3. **Delete Budget**

   Frameworks Utilized*:* Parse

   Data Structures Utilized*:* Parse Tables, Arrays

   This feature works similar to delete a purchase. Swipe left on the cell to reveal a delete button and press it to delete a budget. Several cases must be considered. However, in every case, when a budget is deleted the "Purchases" Table is queried to find that budget and delete all of its associated purchases. If the budget deleted is the user's active budget, then in the "User" table in Parse, their active budget is set to a Parse nil value to denote that there is no active budget. Otherwise, the budget is deleted from the "Budget" table normally.

4. **Select Budget**

   Frameworks Utilized*:* Parse

   Data Structures Utilized*:* Parse Tables, Arrays

   The user must select a budget in order to work with the rest of the application so it is included because it is the most necessary feature. A checkmark is added to a cell that is tapped and when this happens this budget is stored in parse and in the program as the active budget. This is done by updating appropriate variables and querying the "User" table in parse to modify the budget selected field.

5. **Show List of Budgets**

   Frameworks Utilized*:* Parse

   Data Structures Utilized*:* Parse Tables, Arrays

This is an important feature because the user needs to see all the budgets they've created in order to select which one they want to work with. The list is generated by querying the "Budget" table in Parse and displaying all the budgets associated with the current user by object id.

Analysis Page by Jeffrey McLemore

1. **Bar Graph**
   Frameworks Utilized***:*** Parse, iOS Charts API
   Data Structures Utilized***:*** Parse Tables, Arrays
   This is the first financial visual included in order to inform the user of how much they have spent in each category towards their current active budget. It is included because it is a visual that helps the user easily understand where they are at with their purchases. To display this graph the "Purchases" table in Parse is queried for the sum of all purchases in each category relating to the current active budget. This information is stored in an array and passed to a method to set the bar chart provided by the IOS charts library. Colors are assigned while setting the chart.
2. **Target Lines**
   Frameworks Utilized***:*** Parse, iOS Charts API
   Data Structures Utilized***:*** Parse Tables, Arrays
   When you tap a bar on the bar graph a line appears displaying the cap on the bar graph. This feature is important to include because the purpose of the Analysis page is to help the user analyze where they are at in their budget with respect to their caps. The lines are generated using iOS Charts methods and the data is retrieved from "Budget" table in Parse.
3. **Pie Chart**
   Frameworks Utilized***:*** Parse, iOS Charts API
   Data Structures Utilized***:*** Parse Tables, Arrays
   The inclusion of this view was added because it is a good way for the user to view the distribution of their spending. It works exactly like the bar chart function except the parameters are passed to a different function that sets a pie chart rather than a bar chart.
4. **Total Spent in a Category ÷ Total Spent**
   Frameworks Utilized***:*** Parse, iOS Charts API
   Data Structures Utilized***:*** Parse Tables
   When the user taps on a section of the pie chart a percentage displays in the middle. The number represents how much a category attributes to the total amount of money spent. This was included because it is a user-friendly way to view your distribution of money spent. The values required for this equation are calculated from a query on the "Purchases" Parse table and the result is displayed in the middle of the pie chart using an iOS Charts method.