Fishback Stock and Options Trading Game

Senior Design Project Fall 2017

Team Members:
Caleb Cornett
Chad Stephenson
Melissa Shankle
Mitchell McClure

Customer:
Fishback Management & Research

1. Implementation

1.1 Source Code Listing

Our team is using GitHub to store our source code for the project. It can be found at the following link:

https://github.com/TheSpydog/FishbackOptions.git

1.2 Quality Review (Melissa)

To strive for the best overall code review, we combined two of the example checklists given. The first was from MIT[1] and seemed to cover more of the overarching ideas of the code. It included three sections of what to test: function, form, and economy. The function section focused on the inner working of the code, whether the function was clearly implemented, that the idea could easily be recognized, and that it would be protected from misuse. The form section focused on the coding pattern. It looked at whether the code could be easily traced through comments and repeated style to make it understandable by those unfamiliar with it. The last second on economy checked the cost of the code, where the emphasis was placed on redundancy, storage, and changeability.

The second was from the Toolbox website[2] where it covered specifics of each variable and line of code. This checklist also had overall sections including data reference, data declaration, computation, comparison, control flow, interfaces, input/output, and miscellaneous. Once all sections were covered, all variables would be checked for initialization and declaration, loop indexes would be checked for off-by-one errors, computations would be correct with the appropriate type, parameters would be fitting for their function calls, and inputs/outputs would be checked for consistency and validation.

1.3 User's Manual (Melissa)

1.3.0 Introduction (Mitchell)

Before a user can play this game, they need to learn what an "option" is. Stock Market Brokers have more ways to trade stocks than buying and selling. One such way is the idea of options. Instead of buying or selling individual quantities of stock, an option gives the trader the right to buy or sell stock at an agreed upon price during a period of time before an expiration date. An options contract is the agreement that specifies how many stocks, the strike price (the price of the stock when it will be bought/sold), expiration date, company, and cost to purchase.

1.3.1 Beginning the Game

This app will begin by automatically logging in a user using their Game Center information once opened for the first time. A loading screen will be shown as the app connects and information is received.

The user will start the game with $25,000 of available funds to use for buying options. They are able to continuously buy and sell as they please until they run out of funds and options, in which the game will end.

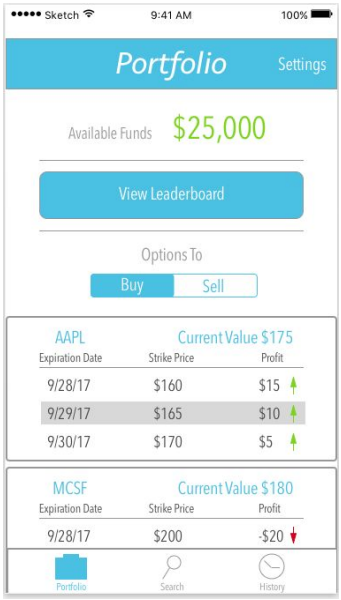
The game saves automatically after each transaction, so no user saving is necessary.

1.3.2 Navigation

This game consists of three main screens: Portfolio, Search, and History. Additionally, the Search screen will take the user to a Company page. The navigational menu will always be available at the bottom on the screen. The icon and words in blue show which screen the user is currently on.

1.3.3 Portfolio

The user will then land on their portfolio page. It will start by showing their available game fund which is updated after each transaction. The amount will be green as long as the user still has money.

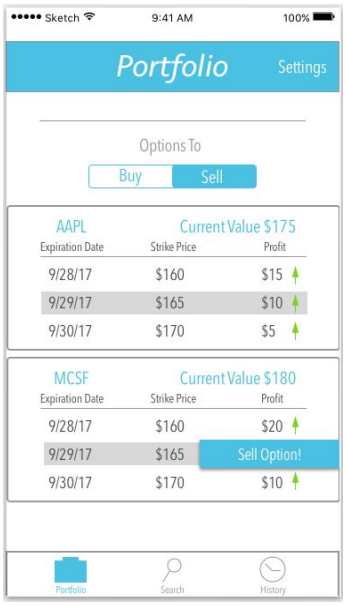


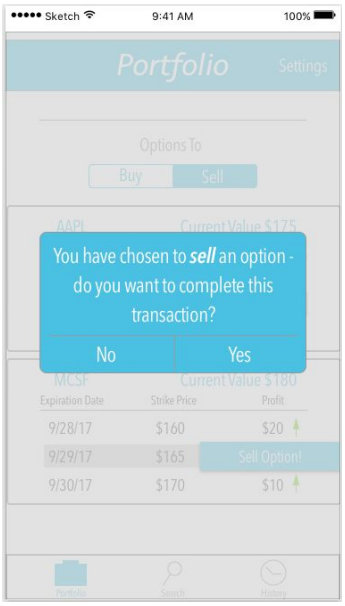
The second item is a button to view the leaderboard. This will be linked to the leaderboard created by Game Center.

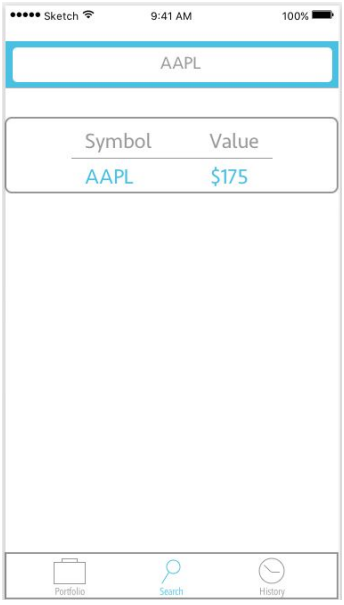
The last section on the screen contains options they have bought. The Options To tab switches between call and put options which helps the user know that one tab will add to their funds and the other will subtract from it based on the type of option. Each company the user buys options from will have their own cards, shown by the gray box outlines. The information for each company includes the symbol, current stock price, and the options the user has obtained. The option information includes the expiration date, strike price, and profit/loss margin to be made with an accompanying arrow reflecting whether it's a profit or loss.

The difference shown in the second image reflects when a user either taps or swipes on an option. A 'Sell Option!' button will appear and become tappable.

The third image shows the pop up message a user will receive after tapping the button. It confirms with the user their action before finishing the transaction.







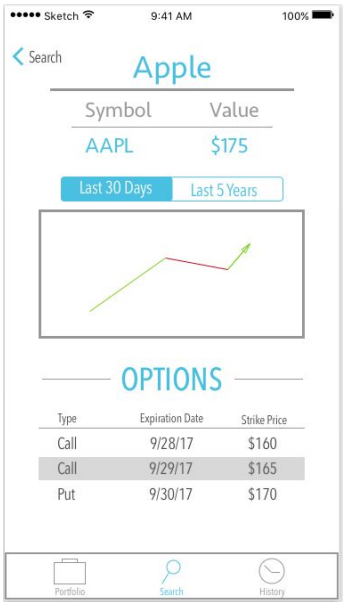
1.3.3 Search

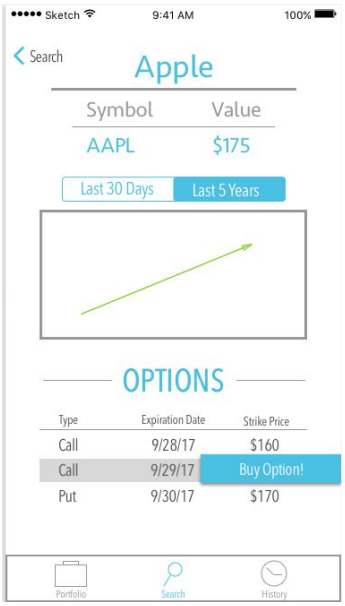
The user is able to use this search to find companies to buy their options or compare them to other companies. A user will land on an empty search page initially. They will just tap on the search bar to enter the company's name or symbol for their search. As they begin typing, results will start automatically appearing and will update with each new letter so the user will not have to type the complete name. Once the search icon or the return key has been tapped, the search results will be shown and the user can tap one to view its company page.

1.3.4 Company Page

The company page contains necessary company information including the name, symbol, current stock value, historical chart, and options list. The historical chart can be switched between last thirty days and last five years using the switch tab above it. The options list contains the type of option, expiration date, and strike price for each option of the company. The third image shows the same behavior as the portfolio page when a user chooses to buy an option.

To return their search, a user can tap the arrow in the upper left corner or tap the search menu icon again.



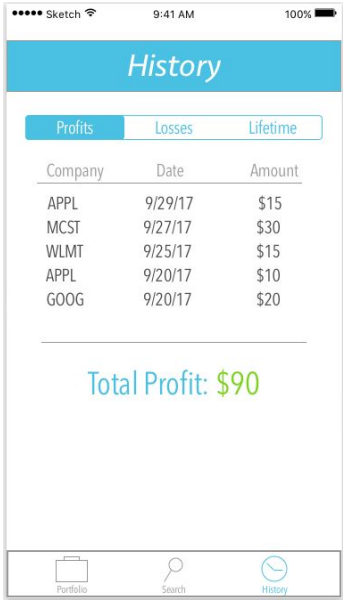


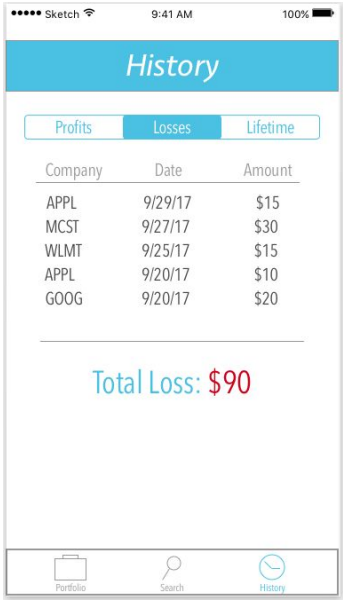
1.3.5 History

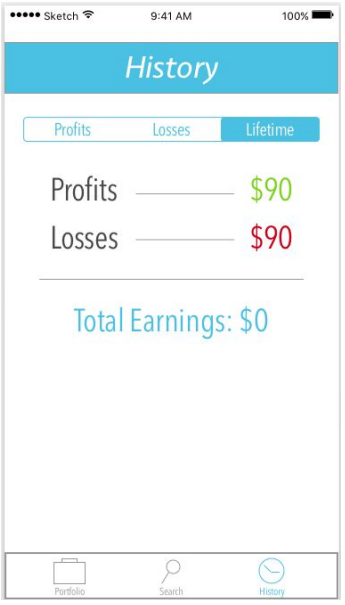
Here a user can see up to 20 previous transactions for profits and losses. The profits tab will show each option that has produced a profit, which includes the company, date of transaction, and profit amount. The second image is the same except for losses. The third image is the lifetime history which simply calculates whether the user has a positive or negative (or is this case neutral) lifetime earnings.

The profits will all have green amounts, the losses will all be red, and in the case of an even profit to loss the amount will be blue.

This page is view only as there are no actions to be done other than switching the tabs.







1.4 Administrator's Manual (Caleb)

Once the app has been released to the App Store, it can be downloaded on any consumer iPhone running iOS 9.0 or above. An internet connection is required for use. For login and authentication purposes, a user must have a GameCenter account connected to their Apple ID.

2. Testing

2.1 Test Plan (Chad)

Our test cases were created to cover both functional and nonfunctional requirements. This would help ensure that we have the app functioning correctly behind the scenes and that a user would have the experience they want and expect.

We have mostly focused on testing the functionality as code was implemented, taking a unit testing approach. This way we could check to see if it worked and didn't interfere with previously implemented code. This relates to our regression testing - as code was added, we verified that it didn't interfere with code written before it.

As features were implemented, we tested the overall app acting as a user would. By doing so, we can get a feel for the flow of the app and hopefully think as a user would to make sure our product also meets the non-functional requirements.

2.2 Test Cases and Results - Acceptance Testing (Chad)

We will use the test cases from Figure 2.2 (below) once we have written our code pertaining to a test case to test the code's functionality. If all of the pass conditions for a test case are satisfied, we will consider the test a pass and the code to be functional. If one or more of the pass conditions for a test are not satisfied, we will consider the test a fail and will rewrite the code. Then we will use the same test case on the rewritten code.

Case No.	Test Case	Pass Conditions	Fail Conditions	Result
1	User creates in-game account by entering their Apple ID and password to sync to GameCenter when prompted	GameCenter prompt is displayed, in-game account is created when user inputs correct Apple ID and password	GameCenter prompt is not displayed, or in-game account is not created when user inputs correct Apple ID and password	Fail: Apple Developer access has not yet been granted, so GameCenter ID authentication is not accessible

2	Stocks and options owned are displayed when on the portfolio page	All stock and options that are owned are displayed in their respective “Buy” or “Sell” tabs when the portfolio page is accessed	None or some of stocks and options that are owned are displayed in their respective “Buy” or “Sell” tabs when the portfolio page is accessed	Pass
3	Leaderboard is displayed to compare user’s score to other users playing the game	Leaderboard is displayed when “Leaderboard” button is pressed, and information in the leaderboard is correct and up to date	Leaderboard is not displayed when “Leaderboard” button is pressed, or the information in the leaderboard is incorrect or outdated	Fail: Refer to Case No. 1
4	User is able to search for company based on company name or symbol	The company that is output from the user search corresponds to the name or symbol that the user searched for, or no company is output if user searched for an invalid company name or symbol	The company that is output from the user search does not correspond to the name or symbol that the user searched for, or a company is output if user searched for an invalid company name or symbol	Pass
5	Current stock price and options that can be bought are displayed on a company’s page	Company’s page is displayed with its correct current stock price and available options to buy	Company’s page is displayed with an incorrect stock price, or with out all available options to buy	Pass
6	Stock options are able to be bought from the list of	After pressing the “Buy” button by an option, the user is prompted to	After pressing the “Buy” button by an option, the user is	Fail: The option that is selected will

	options available on a company's page	follow through with the transaction or cancel. If canceled, option is not bought. If accepted, the cost of the option is deducted from the user's available funds, and the option is stored in the user's portfolio	not prompted, or if prompted and canceled, option is still bought, or if accepted the cost of the option is not deducted from the user's available funds, or the option is not stored in the user's portfolio	be added to the user's portfolio, but the cost of the option is not yet deducted from the user's available funds
7a	Stock controlled by *put* options can be sold from the portfolio	The amount of stocks that are part of the option are sold successfully at the strike price agreed upon, and that money is added to the user's available funds. The transaction will also be added into the history page	The amount of stocks that are part of the option are not sold successfully, or the stocks are not sold at the strike price agreed upon, or that money is not added to the user's available funds. Or the transaction is not stored within the history page	Fail: Functionality not yet implemented
7b	Stock controlled by *call* options can be bought from the portfolio	The amount of stocks that are part of the option are bought successfully at the strike price agreed upon, and that money is deducted from the user's available funds	The amount of stocks that are part of the option are not bought successfully, or the stocks are not bought at the strike price agreed upon, or that money is not deducted from the	Fail: Functionality not yet implemented

			user's available funds	
8a	A list of all past profits can be viewed on the Profits tab on the History page	A list of all stocks sold for a profit is displayed when the Profits tab is pressed, along with the amount of profit each stock made and the total amount of profits made throughout the user's game lifetime	A list of all stocks sold for a profit is not displayed when the Profits tab is pressed, or the amount of profit each stock made is not displayed or is incorrect, or the total amount of profits made throughout the user's game lifetime is not displayed or is incorrect	Fail: Transactions can be added with already implemented code, but the functionality for options actually bought to show on screen on the history tab is not yet implemented
8b	A list of all past losses can be viewed on the Losses tab on the History page	A list of all stocks sold for a losses is displayed when the Losses tab is pressed, along with the amount of money each stock lost and the total amount of losses had throughout the user's game lifetime	A list of all stocks sold for a loss is not displayed when the Losses tab is pressed, or the amount of money each stock lost is not displayed or is incorrect, or the total amount of losses had throughout the user's game lifetime is not displayed or is incorrect	Fail: Functionality not yet implemented
8c	The total earnings of the user can be viewed on the	The total profits, total losses and total earnings are displayed when the Lifetime tab is	The total profits, total losses and total earnings are not displayed	Fail: Functionality not yet implemented

	Lifetime tab on the History page	pressed, and the amounts are all correct	when the Lifetime tab is pressed, or the amounts are incorrect	
9	The user clicks on one of the three buttons at the bottom of the screen that can take you to portfolio, search, or history	The correct page is displayed after pressing	The button does nothing or the wrong page is displayed	Pass

Figure 2.2

2.3 Test Cases and Results - Unit Testing (Chad)

We did unit testing on each of the functions as we created them. These tests help to ensure working code before the developer moved on to a new feature. Figure 2.3 (below) shows some of the test cases and their results.

Case No.	Test Case	Pass Conditions	Fail Conditions	Result
1	func populateTable()	Table on the portfolio page of options is correctly filled.	Table on the portfolio page of options is incorrectly filled or not at all.	Pass
2	func getUserData()	Function gets the correct user data from the database.	Function gets the incorrect user data from the database.	Pass
3	func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)	The table view on the History page is formatted correctly with the correct data.	The table view on the History page is either formatted incorrectly or populated with incorrect data.	Pass
4	func goBackPressed(_ sender: UIButton)	The GoBack button is pressed and the app goes back to the search page.	The GoBack button is pressed and the app goes to an incorrect page or does nothing.	Pass

5	func getFinancialData()	Function gets the options listed for the company the user has searched.	Function gets the wrong options or doesn't return for the listed company.	Pass
6	func searchBarIsEmpty()	Returns true if the text is empty or nil.	Returns false if the text is empty or nil.	Pass
7	func filterContentForSearchText(_ searchText: String, scope: String = "All")	Queries database correctly for searching companies with searchText and updates the cell appropriately.	Queries database incorrectly for searching companies with searchText or does not update cell appropriately.	Pass
8	func isFiltering()	Returns true if a user is searching and typing in the search bar.	Returns false if a user is searching and typing in the search bar.	Pass

Figure 2.3

Note: many of the screens use tables and have very similar code. We tested each individually but only added test cases here for one instance.

2.4 Quality Review (Melissa)

As we stated in 2.1 Test Plan, we chose our test cases by both a user perspective and a developer perspective. By doing so, we were able to cover the app and test all four levels: unit, integration, system, and acceptance. We used our designs for brainstorming about how it would be used to find how the app should react, our story points, and our technical background in databases for our initial test cases.

Since creating the original test cases, we have discovered that the nature of Swift and the third-party libraries in our project provide new testable material. Additionally, some features listed in the test cases have yet to be fully implemented, so we can not test them effectively.

We also were planning to do regression testing during each new sprint, however our schedule didn't allow for it very often. Instead, we relied mostly on unit and system testing to detect bugs developed from previous sprints.

2.4.1 Issues (Caleb)

To ensure the app works as intended, the code has been reviewed and defects have been corrected on an as-needed basis throughout development. Some notable issues have included:

1. Incorrect / unnecessary Swift code style, such as using an extension for a class to implement a behavior instead of implementing it in the class itself. Changing this saved several lines of code and improved readability while retaining the same desired behavior.
2. The Search screen would turn entirely black under the following circumstances: (1) the user taps the search bar and enters text, (2) the user then transitions to another tab, (3) the user returns to the Search screen. This bug was fixed by implementing the UITabViewControllerDelegate protocol and dismissing the search bar controller upon transitioning to another tab.
3. Purchasing an option from a Company page would update the database but the Portfolio screen would not display the newly purchased option until the app was restarted. This was because the app only populated the Portfolio view when it first loaded, not whenever the user returned to the page. This was fixed by overriding a UIViewController function called viewWillAppear() and calling the populating method there.

3. Technical Metric Collection

3.1 Estimated Story Points (Melissa)

To help cover each area of the app when writing user stories, we sorted stories into categories that covered each epic or large feature. Note however that some stories are a blend between two categories. The following user stories each have an assigned value, an epic value (the sum of all user story values for a given epic), and a total project value.

Design - 3 SP
As an elderly user, I want big and easy to read font so that I can easily see what I'm doing in the game. - 1
As an iOS user, I want typical iOS conventions so that I am already familiar with the layout and flow of an app. - 2

Account - 2 SP
As a user, I want an account linked to my Game Center ID so that I can use the leaderboard features. - 2 SP

Game Structure - 10 SP
As a user, I want to be able to restart the game so that I can try again. - 3 SP
As a user, I want to be able to view how much money I have so that I can buy/sell accordingly. - 1 SP
As a user, I want to be asked for permission to enable push notifications after the tutorial so that the timing is convenient and I know what is happening on the app. - 2 SP
As a mid-twenties aged user, I want stock trading to be as realistic as possible so that I can practice and prepare for stock trading in the future. - 3 SP

As a newcomer to the stock market, I want an easily traversable learning curve in the app such as a tutorial so that I can learn quickly and efficiently about stocks and options. - 1 SP

Searching - 8 SP
As a user, I want to be able to search for companies via their name so that I can easily find them without knowing their symbols. - 4 SP
As a user already familiar with the stock market, I want to be able to search for companies via their stock symbol so that I can quickly find them. -2 SP
As a user, I want to be able to type in part of a company's name so that I can still find them without knowing the full name. - 2 SP

Buying Options - 26 SP
As a user, I want to be able to see a chart of historical data so that I can make an informed decision for whether I want to buy the stock or not. - 8 SP
As a user, I want to see a list of options for a company so that I can choose the option I think will return the most profit. - 8 SP
As a user, I want an error message when I try to buy an option I cannot afford so that I do not accidentally believe I have made a purchase. - 2 SP
As a user, I want to be able to buy both Call and Put type options so that I can strategize for buying and selling as the market rises and falls. - 8 SP

Bought Options - 4 SP
As a user, I want be notified that an option is about to expire so that I have the chance to utilize it. - 2 SP
As a user, I want to see recently expired options so that I can be filled with regret. - 2 SP

Selling - 5 SP
As a user, I want to view my stocks' current prices so that I can determine if I want to sell them. - 3 SP
As a user, I want to be able to sell my stock so that I can make a profit. - 2 SP

Portfolio - 11 SP
As a user, I want to see a list of past transactions so I can improve my future game decision making. - 3 SP
As a user, I want to see my earnings so that I will know how much money I have made. - 1 SP
As a user, I want to view my profits and losses so that I can evaluate my progress. - 3 SP
As a user, I want to see how many shares of each stock I own so that I can know how many I have to sell/buy. - 2 SP
As a user, I want to see the price I bought a stock at compared to its current value so that I know whether I will make a profit or loss. - 2 SP

Multiplayer - 5 SP

As a competitive user, I want to be able to see a leaderboard so that I can compete with others. - 5 SP

The total story points estimated is 74.

3.2 Actual Lines of Code (Mitchell)

The total actual number of lines of code is 1006.

PHP Scripts
- Query.php - 97
- SearchCompanies.php - 83
- newUser.php - 69
- insertTransaction.php - 85
- insertOption.php - 85
- updateCurrentFunds.php - 68

Swift Code
- Views
  - PortfolioViewCell.swift - 17
  - CompanyViewCell.swift - 17
  - HistoryViewCell.swift - 17
- Controllers
  - CompanyViewController.swift - 90
  - HistoryViewController.swift - 64
  - SearchViewController.swift - 154
  - PortfolioViewController.swift - 82
  - GCViewController.swift - 78

3.3 Complexity of Each Module (Caleb)

Using the CK Metric of Weighted Methods Per Class:

SearchViewController: **12**
CompanyViewController: **7**
HistoryViewController: **4**
PortfolioViewController: **6**

3.4 Complexity of Overall System (Caleb)

The maximum depth of any inheritance tree in our code is 2.

3.5 Product Size (Chad)

We have planned our game based on three user stories. We have a total of 13 test cases, and 12 classes planned.

3.6 Product Effort

	Hours	Word Count
Caleb Cornett	25+	259
Chad Stephenson	25+	1615
Melissa Shankle	14	2562
Mitchell McClure	25+	92

3.7 Defects (Mitchell)

1. Minor typo (Section 1.2)
2. Reworded for Brevity and Clarity (Section 1.3.0)
3. Switched Issues and Defect Sections because of miscommunication (Sections 2.4.1 and 3.7)
4. Moved Issue Section (Section 1.2 to Section 2.4.1)

4. Web Page and Developer Notebook (Melissa)

Our team chose to use WordPress to keep track of our developer logs. Each team member has their own page in addition to the team page. Member pages are used to record decisions and actions of each member while the team page is for group decisions and meeting notes. Other pages house documents such as the project plan and architecture assignment. Our site also includes our contact information and the project overview.

Our site can be found at https://stockexchangegame.wordpress.com/

Word Count as of November 12th:

- Caleb - 932
- Chad - 291
- Melissa - 2352
- Mitchell - 1343

5. References (Melissa)

[1] Checklist for Code Walkthroughs - http://www.mit.edu/~mbarker/ideas/checkcode.html
[2] Systems Development: Code Walkthrough Checklist - http://it.toolbox.com/blogs/enterprise-solutions/systems-development-code-walkthrough-checklist-49283