**“Deliver a written report on what you are going to build, who is going to build it, how you are going to build it, how you are going to ensure quality (in source code and product), how long it will take to build, and why you expect to use this software at least every week. That is, justify what you are going to do. You can already have started building it at this point, but you are not required to do so.”**

Plan for the IOS app “name”

Product description:

1.      App overview.

Our app “name” is a personal financial management (budgeting) mobile app that will run on IOS and be deployed to the apple app store. The user groups we are focusing on are mainly but not limited to students and travelers. The purpose of this app is to help users to record day-to-day expenses in a simple and intuitive way. We will be using c++ as the main coding language, and objective c++ and XCode for the gui interface.

2.      Requirements.

a.       Functional Requirements:

o   The system should allow the user to add, edit, delete new expenses and keep track of those expenses. (in timeline format)

o    The system should record the date and time of expenses and be editable.

o   The system should be able to extract data from the memory and analyze it.

o   The system should provide multiple (bank) accounts management function.

o   The system should provide a view where users can scroll down to see their history.

o   The system should provide a budget setting.

o   The system should allow user to specify/edit/delete their income.

o   The system should allow the user to set/edit/delete a goal of their saving.

o   The system should allow the user to set/edit/delete his/her upcoming(fixed) payments.

o   The system should be able to alert the user(+ notification settings). (eg. upcoming payments, balance low)

o   The system should show the total expense of the user for a period of time.

o   The system should provide a calendar date picker to allow the user to add a new expense to a past date.

o   The system should allow automatically update balance for upcoming payments.

o   The system should potentially collect data from users and send it back to developers.

o   The system shouldn’t do math on double/ float.

o   The system should present money as $xK while the expense is bigger than $100000?

o   The system should have several arrays where each number in the field is smaller than 8 digits.

o   While performing 1.00+ 0.10, it should do (1\*100+0.10\*100)/100.

o   The system should always take expense inputs as float/double with a format $.$$.

o   The system should compress the picture/ audio before storing.

o   The system should save pictures, audios, locations or/and texts for specific expenses.

o   The system should provide different UIs for different account types.

o   The system should support PDF generation.

o   The system should send the user his/her spending analysis through email/ Dropbox.

o   The system should provide charts.

o   The system should have the average living cost of a day/week/month

o   The system could be password protected/encrypted.

The system should have a daily transaction data even if there is no transaction.

b.      Nonfunctional requirement:

o   The report generation should be fast.

o   The photo compress should be fast.

o   The Email should be safe.

o   Users should be able to recover their passwords if they forget them.

o   The report should be in a proper size.

o   Adding a new expense should be easy.

o   Automatic (Fixed) payment should be generated automatically.

o   The code should be portable.

o   The code should be documented well.

o   The UI should be clean and intuitive.

Key points to success:

1.      Design pattern:

We want to make the app scalable, maintainable and easier to implement unit tests.

We’ll implement MVC design pattern in this project.

2.      Version control:

We’ll use version control though the entire development. Every member in the group should learn how to use Git remote, merge, pull, push and commit.

Every commit should be meaningful and should contain proper commit messages.

3.      Code:

The code should be readable and well commented.

Segues should be performed programmatically.

UI could be done in storyboard so that constrains are easier to maintain in our small project.

The core code should be written in C++.

4.      Intuitive design:

There will be multiple versions of designs.

The most intuitive and clean one will be implemented. (distribute surveys among friends and families whose identity is a part of our main user group.)

5.      UML

The UML should be drawn before the coding process.

6.      Testing:

The programmer will perform unit testing.

Others will try to break the app as hard as they can.

Quality control:

1.      Characteristics:

a.      Correctness

                                                             i.       Test cases should be chosen carefully.

                                                            ii.      Use unit testing to reduce the cost of testing.

                                                          iii.      We’ll try to handle as many errors as we can.

b.      Usability

                                                             i.      UI designs will be clear.

                                                            ii.      The prototype will be shown to friends and families and will be promoted over time.

c.       Efficiency

                                                             i.      User’s large inputs will be compressed before storing.

d.      Robustness

                                                             i.      The users’ input should be restricted to a proper type and size.

                                                            ii.      Invalid inputs will be handled.

                                                          iii.      The app will be able to run on iphone5s or higher smoothly.

e.      Maintainability

                                                             i.      Comments of each file would be proof read by every group member. Especially whose function doesn’t include programming. So we can make sure that the code is easy to be picked up by others.

                                                            ii.      Pseudo code and UML will be updated through the development.  (but it could be time consuming and painful.)

The app we will be building, will be a financial/budgeting app. It will provide functionality for keeping track of income and expenses and setting goals. The development team is a group of 4 people who will all be working on the project functions together. We will all take part in planning and designing the app in the planning phase. We will add functions incrementally to the app, during the development phase.  We are going to build the project in c++and build the gui using XCode, using objective c++. We will ensure quality in the source code and product by formal inspection of the code, good test cases, component testing and system testing. We have estimated that the overall project will take X hours to make. We expect to use this software every week because as students, we need to keep track of our income and spending to make sure we have enough money to pay rent etc. We would like to be notified of upcoming payments, calculate funds, and set a goal of an amount to save.