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 the users’ 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.

1. Requirements.
   1. Functional Requirements:
   * The system should allow the user to add new expenses and keep track of those expenses.
   * The system should allow the user to edit past expenses.
   * The system should record the date and time of expenses.
   * The system should save pictures, audios, locations or/and texts for specific expenses.
   * The system should be able to extract data from the memory and analyze it.
   * The system should provide charts.
   * The system should provide multiple accounts management function.
   * The system should provide different UIs for different account types.
   * The system should provide a view where users can scroll down to see their history.
   * The system should support PDF generation.
   * The system should send the user his/her spending analysis through email/ Dropbox.
   * The system should provide budget setting.
   * The system should allow user to specify their income.
   * The system should allow the user to set a goal of their saving.
   * The system should allow the user to set a his/her upcoming(fixed) payment.
   * The system should be able to alert the user.
   * The system should show the total expense of the user in a period of time.
   * The system should provide a calendar date picker to allow the user to add a new expense to a past date.
   * The system should allow user to track their bank upcoming payments.
   * The system should have the average living cost of a day/week/month
   * The system could be password protected.
   * The system should potentially collect data from users and send it back to developers.
   * The system shouldn’t do math on double/ float.
   * The system should present money as $xK while the expense is bigger than $100000?
   * The system should have several arrays where each number in the field is smaller than
   * 8 digits. While performing 1.00+ 0.10, it should do (1\*100+0.10\*100)/100.
   * The system should always take expense inputs as float/double with a format $.$$.
   * The system should compress the picture/ audio before storing.
   1. Nonfunctional requirement:

* The report generation should be fast.
* The photo compress should be fast.
* The Email should be safe.
* Users should be able to recover their passwords if they forget them.
* The report should be in a proper size.
* Add a new expense should be easy.
* Fixed payment should be generated automatically.
* The code should be portable.
* The code should be well commented.
* 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.

A good design pattern could help us. It helps our team to understand each others’ code.

Although MVC doesn’t scale and controllers usually grow too big. But considering it’s our first IOS app. So we’ll stick with MVC.

1. 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.

1. Code:

The code should be readable and well commented.

The code should follow Object-oriented principles.

Segues should be performed programmatically.

The core code should be written in C++.

1. Intuitive design:

There will be multiple versions of designs.

The most intuitive and clean one will be implemented.

All the designs should be saved in local Git as we don’t want to make it public.

1. UML

The UML should be drawn before we start to code.

Quality Assurance