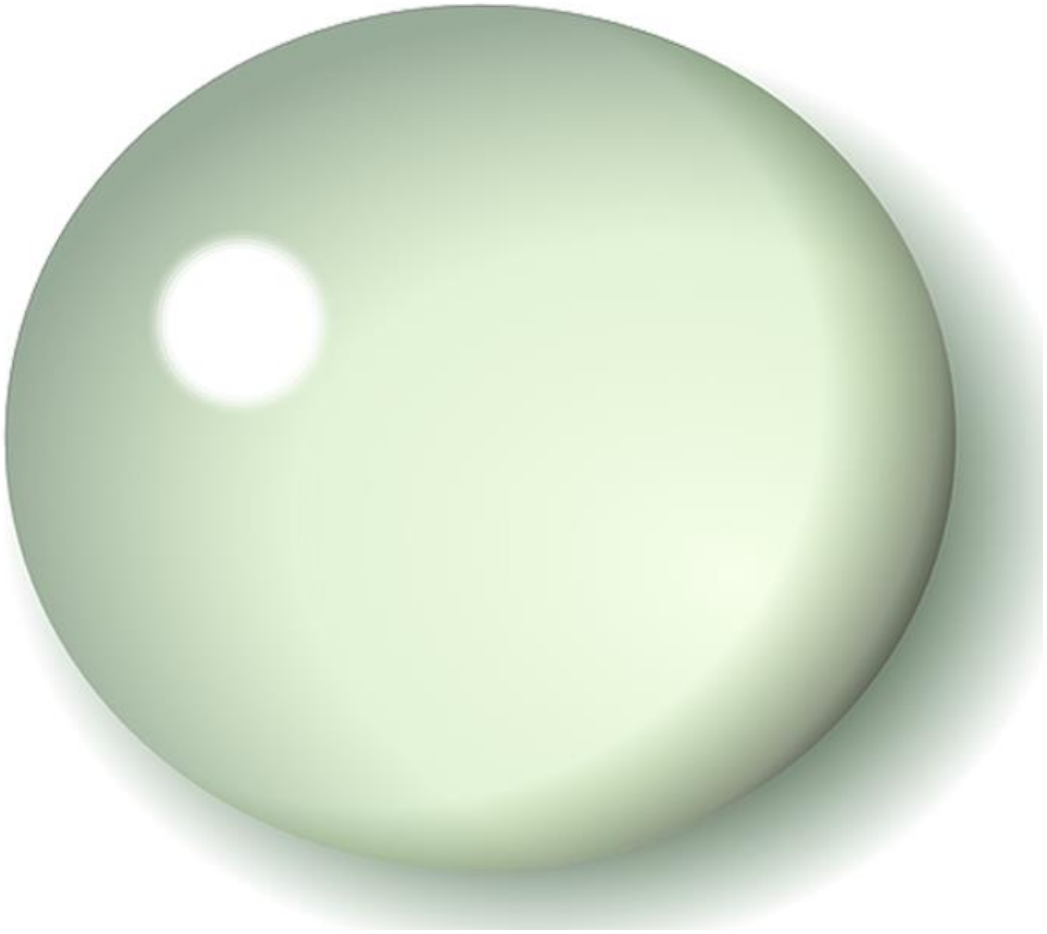
TapWater

Software Project Plan and Management Document



Compiled by:

Kon-Kon Chao

David Fontana

Jonathan Hooper

Cody Rogers

Vision

The purpose of the TapWater Application is to help people who need to monitor their water consumption.

It can help athletes and people with medical conditions that demand they drink a certain amount of water every day.

One of the major gripes about dieting or other apps that track water consumption is how difficult it is to enter data.

Navigating to the screen to enter water is often difficult

After the user reaches the screen, they are usually asked to enter an amount in ounces.

Sometimes it can be hard to guess how many ounces of water you've had to drink.

TapWater seeks to solve this problem by making it easy and straightforward to enter water consumption.

TapWater gives the uses set amounts of water they can log.

The amounts relate to real world objects, so it is easy for the user to gauge how much water they wish to log.

The interface is simple, clean, and easy to understand.

Configuration Management System

The TapWater application makes use of Git/Github as its configuration management system.

Git is a very popular and dependable version control system that allows for frequent revisions to be made to our files on Github's server while also providing peace-of-mind that if any change is damaging, it is always possible to revert to previous versions.

Github is the server used to store files and configurations related to TapWater. With Github, there is no cost to host repositories for TapWater's purposes, and group collaboration is very simple.

Process Model

We chose the Extreme Programming (XP) process model. We chose the XP model because we have a small team working on a project with a simple design. The XP's focus on simplicity would help us focus on keeping he design simple. Since the team all attend the same university having regular scrums was very feasible.

Deliverables

1. System Requirement Specification Document
   This documentation deliverable provides a detailed description of the software system, as well as a detailed list of features (both critical and non-critical), planned interfaces, and other requirements necessary for project success. It is deliverable one to two weeks after project startup.

2. System Design Document
   This document provides a more in-depth look at the systems architectures of the TapWater project. It includes the rationale for the architectural choice of each system of the software, including platforms and subsystems. This document is deliverable roughly three to four weeks into the project.

3. Internal & External Software Documentations
   These are two separate technical documentation deliverables. The Internal Software Documentation deliverable consists of a detail list of all classes (objects) and methods (functions) of the software system. The External Software Documentation deliverable provides a less technical, more functional look at the software system's function (method) capabilities and the overall structure of the code. These deliverables are available roughly six to eight weeks after the project start date.

4. Testing Documentation
   The Testing Documentation deliverable provides proof of iterative use-case testing, following the IEEE 829 format. It includes testing of all features and cases covered in the System Requirement Specification document. This deliverable will be available after roughly ten weeks.

Potential Risks

In any developmental undertaking there is a certain amount of risk. Risks associated with this software development project can be broken down into several categories including time, resources, and liability.

The TapWater software development project has very low overall risk associated with it. The core application features themselves are simple, and provide a foundation upon which further features could be added iteratively without much time constraint. Even the time spent developing the base application is, in this instance, well-spent because the skills, experience, and knowledge gained will undoubtedly be useful in other projects for the developers.

Further, because the TapWater project is an independent undertaking, there is very little other resource or financial risk. In this case, the primary resource risk is time, which as shown above can be mitigated. The risk for delay in the original schedule are also mitigated in that, being an independent project, the communication-loop is relatively swift.

Additional risks for the project (though very small) include the risk of requirement inflation and some liability issues. The first of these – requirement inflation – can be combated by adhering to strict guidelines on what is and is not a “necessary” or “core” feature for the initial development cycle. The second – liability – comes from the fact that TapWater is, essentially, a health app. Therefore, the project team must be very careful and realistic about any promises or advertisements made either within the app or during presentations. A legal disclaimer is also a good idea.

Team Members

Our team style was a slightly modified Chief programmer style. Jonathan did most of the Ruby on Rails server setup and iOS application development while Cody did most of the Android development. The rest of team was primarily focused on project documentation and creation of key documents for the course.

Shih (Kon-Kon) Chao - Documentation and planning

David Fontana - Documentation and planning

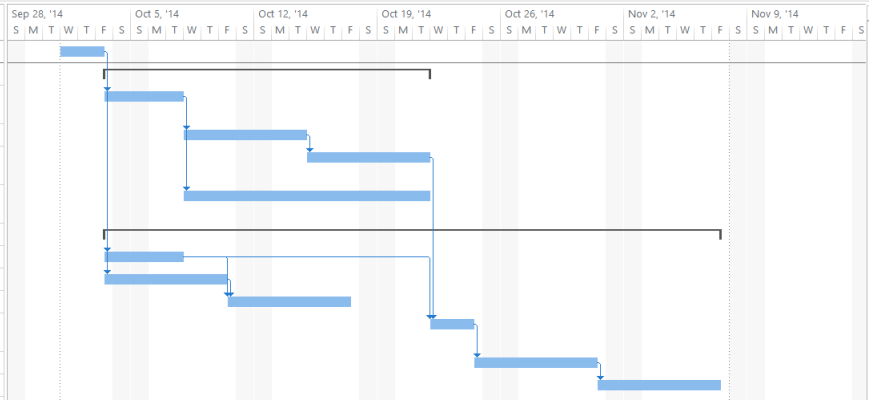
Jonathan Hooper - Ruby on Rails and iOS application development

Cody Rogers - Android application development

Project Schedule (Gantt chart)

GANTT CHART

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Project Initialization	0.5 wks	Wed 10/1/14	Fri 10/3/14	
2		**Server**	**12.5 days**	**Fri 10/3/14**	**Tue 10/21/14**	
3		Database/Model Layer	0.5 wks	Fri 10/3/14	Tue 10/7/14	1
4		Authentication	1 wk	Wed 10/8/14	Tue 10/14/14	3
5		Mobile API Implementation	1 wk	Wed 10/15/14	Tue 10/21/14	4
6		Views/View Controllers	2 wks	Wed 10/8/14	Tue 10/21/14	3
7		**Mobile Apps**	**25 days**	**Fri 10/3/14**	**Fri 11/7/14**	
8		Core Data Setup	0.5 wks	Fri 10/3/14	Tue 10/7/14	1
9		Views/Controllers	1 wk	Fri 10/3/14	Fri 10/10/14	1
10		Drink Notifications	1 wk	Fri 10/10/14	Fri 10/17/14	8,9
11		Integration With Server	0.5 wks	Wed 10/22/14	Fri 10/24/14	5,8
12		User Authentication	1 wk	Fri 10/24/14	Fri 10/31/14	11
13		Synchronization	1 wk	Fri 10/31/14	Fri 11/7/14	12



Meeting Summaries

Meeting 1

Time: 28 September 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao

Objective: Group merging, system architecture, roles

Outcome: Assigned roles, discussed merge, decided on Ruby on Rails

Meeting 2

Time: 3rd October 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao, David Fontana

Objective: Development check-in, SRS document planning, plan for next meeting

Outcome: Finished SRS document

Meeting 3

Time: 2nd November 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao, David Fontana

Objective: Create charts and SSD document

Outcome: Finished SSD document

Meeting 4

Time: 16th November 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao, David Fontana

Objective: Create internal and external documentations

Outcome: Finished internal and external documentations

Meetings Summaries (cont.)

Meeting 5

Time: 22nd November 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao

Objective: Software test document creation and presentation preparation

Outcome: Software test document not finished but presentation slides finished.

Meeting 6

Time 23rd November 2014

Members: Jonathan Hooper, Cody Rogers, Shih Chao, David Fontana

Objective: Finalize presentation and practice runs

Outcome: Presentation finished, roles in presentation determined.