VIA UNIVERSITY COLLEGE

Process Report
Danish Aesthetics' iOS Application

Date: December 9 2014
Mikkel Cortnum Poulsen – 171460
Janis Bruno Ozolins – 166732

Supervisor: Lars Bech Sørensen – LBS

Table of Contents

1. Project Plan

1.1 Aim

The aim of the project is to build an iOS Application and Web Server that allows users to manage and track their gym workouts and deliver content on iPhones from Danish Aesthetics web site.

1.2 Objectives

- The client application will be developed for iPhone
- The client application will be able to communicate with a web server
- The client app will not communicate with remote database directly
- The web server will store client data in MySql database
- The web server will be developed on Sail.js framework
- The client application will be getting content from existing danish-aesthetics.dk website
- The steps of developing the system will be described in a Project report
- The process of the system development will be described in a Process report

2. Time estimates

To realistically determine the time what the project execution requires and to manage the development and documentation of the project we did initial project analysis. The analysis helped to break the “big-picture” down into manageable increments of work called Product Backlog Items. To find out if all of the items can be implemented in the scope of our project period we spent time to prioritize our tasks by assign priorities to product backlog items by business importance, where higher value of the item has a lower priority. For each items we made time estimates and they are documented in the Product Backlog[1]
As the time period for the project was 72 working days from 1st of September till 12th of December, the total time estimates are suppose to fit in 576 working hours for each group member (assuming 8 hour working day). Furthermore the whole project period was split into three sprints. First sprint will start on 1st of September and will continue till 28th of September. Second sprint starts on 29th of September and end on 2nd of November and third sprint starts on 3rd of november and ends on 9th of December.

3. Group Members

The group consists of Mikkel Cortnum and Janis Ozolins.

4. SWOT Analysis

4.1 Mikkel Cortnum

Strengths:	**Weaknesses:**

• Perfectionist • Great programming skills • Great problem solving skills • Keen logical thinking • Quick learner	• Perfectionist • Postponing • CO-Founder of Danish Aesthetics
Opportunities: • Will make a great project with clean code and sound logic. • Will learn the new frameworks, languages and technologies quickly.	**Threats:** • Will spend too much time on one thing, to make it perfect instead of acceptable. • Will postpone tasks and not complete tasks on deadline. • Will spend too much time on other Danish Aesthetics projects. • Too much KK.

4.2 Janis Bruno

Strengths: • Quick at learning new concepts and programming languages • Experienced JavaScript programmer	**Weaknesses:** • Does not handle stress well. • Forgets to finish smaller tasks if something is more important. • Not familiar with Swift programming language.
Opportunities: • Get familiar with iOS development. • Understand in depth Git version control. • Explore the opportunities of Sails.js framework	**Threats:** • Context switching between programming languages. • Too heavy workload from external sources

5. Group Policy

Both group members have worked on several projects together and therefore both of them understand each other's work habits really well. They both do their work in a timely manner and like to finish tasks on time. To outline some other common items what the members agree on we have made the list:

- Meet on agreed time
- Do not commit code that does not compile

- Do not plagiarize
- Write clean code
- Document your work
- Ask each other if stuck with some problems
- It is okay to have fun while working
- Work place is in VBI room 4.02

6. Implementation phase

6.1 Version Control

In order to start working on the implementation a big task is to set up version control system. Version control is used to store and distribute files and source code for project development. Since we have worked with several source control systems and tools we decided to use Git [1].

Our remote private repositories are hosted on bitbucket.org. We have divided our project into two separate repositories:

1. Swift application
2. https://halfred88@bitbucket.org/halfred88/danish-aesthetics-swift.git
3. Node.js Api Server
4. https://halfred88@bitbucket.org/halfred88/danish-aesthetics-node.git

6.2 Time Management

For keeping track of time and the work needed to be done within a given time, the team used some artifacts of the SCRUM framework. The main elements we used from SCRUM was, the backlog with stories and sprints. We did not however, keep a burndown chart to measure velocity. As for SCRUM roles, our product owner is Mikkel Cortnum Poulsen, our SCRUM master is Janis Bruno Ozolins and the development team consists of both group members. We simply used enough elements from SCRUM, so that we knew which tasks needed to be done when, and if some were not completed on time, we would work nights and weekends to make sure they were.

Before each sprint we follow these steps:

- If its not the first sprint we spend one to two hours to do backlog grooming, hold a short sprint retrospective meeting and write a summary of the retrospective in the process report.

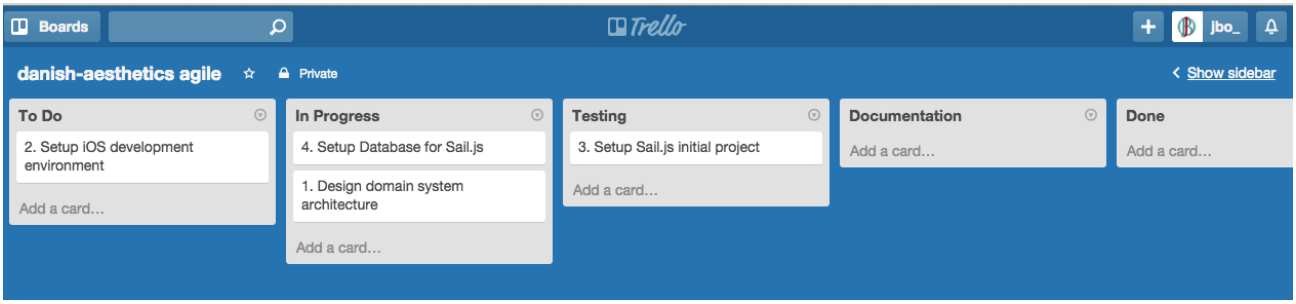
- We take product backlog items with highest priority and split them smaller tasks, and add them to the sprint backlog.

- For each task throughout discussion we estimate how long time the task would take to be successfully completed and add the time units in the sprint backlog along with the task

- We take tasks from sprint backlog and decide which group member will be responsible for the task execution

During the sprint

- We add the tasks to our agile board in trello.com
- According to the task's status, the group member who is responsible for the task, moves them across five columns: To Do, In Progress, Testing, Documentation, Done



Below you can see the entire product backlog for our project.

Product Backlog

Nr.	Title	Description	Priority
1	Design system architecture	Design domain system architecture	1
2	Setup iOS development environment	Set up XCode IDE and create initial project with version control	1
3	Setup Sails.js	Set up Sails.js web server and incorporate version control	1
4	Set up Web servers database	Create database for web server with object relation mapping, populate tables with test data	1
5	Design Graphical User Interface	Design all the screens for the application (login, signup, create workouts etc.)	1
6	Implement Welcome Screens	Implement welcome screens in iOS	1
7	Implement API functionality to support welcome screens	Create Models and Controllers to support user CRUD functionality	1
8	Implement JWT authentication	Create controllers and UserManager class to sustain the JWT authentication process	1
9	Design and Create iOS Models	Design and create the CoreData models in Xcode	1

10	Design and Create API Service	Design the initial data flow and communication with the web server over HTTP	1
11	Create API Models	Create API models and add mapping functionality from API model to CoreData models and vice versa	1
12	Implement workout routine flow	in iOS implement workout routine flow with all CRUD functionality for nested elements - workouts, exercises, sets.	1
13	Implement content pulling from danish-aesthetics.dk	Implement in iOS web view to get the content from the danish-aesthetics.dk website	1
14	Implement Workout Routine CRUD functionality on web server	Create Workout controller and CRUD functionality for Workout Routine model	2
15	Make unit tests for CoreData	Create unit tests to for CoreData on iOS to test if nested relations behave as expected	3
16	Add serialization/deserialization of JSON Objects	Implement for API models JSON serialization and deserialization and test if correct values are retrieved and sent from web server	2
17	Implement tracking feature	Design and implement in iOS tracking feature for workouts	3
18	Implement profile features	Design and implement user profile CRUD functionality	1
19	Implement workout statistics	Design and implement workout statistics in iOS application	3

Sprint 1 (September 1st - September 28th)

For the first sprint the initial project setup must be done for the iOS application and the web server. When the initial setup is done the version control must be established and tested so both developers can keep up to date with the latest changes. The initial setup is required so we can proceed with the feature development for the iOS application and the web server. When the version control is set in place and everything works as expected, the next task is to design and implement the user interface for the welcome screens of the application. Meanwhile the server side of the database must be connected and some basic unit test performed to confirm if the adapter and database works as expected.

Nr.	Title	Description	Responsible	Estimated

				Time
1	Design system architecture	Create initial system architecture diagram	JBO, MCP	12
2	Setup iOS development environment	Set up XCode IDE and create initial project with version control	MCP	4
2.1	Test version control for iOS	Test if initial project is added correctly to repository and it is building on two different Apple laptops.	MCP, JBO	4
3	Setup Sails.js	Download tools what are necessary for Sail.js development and create initial project.	JBO	4
3.1	Test version control for Sails.js	Add the initial Sails.js project to BitBucket repository and test if it is running on two laptops	JBO	4
4	Set up Web servers database	Create database for web server with object relation mapping, populate tables with test data	JBO	4
4.1	Test database	Create first tables and make manual test to populate/delete data in database from web server	JBO	8
5	Design and Create iOS Models	Design and create the CoreData models in Xcode	MCP	48
6	Create API Models	Create API models and add mapping functionality from API model to CoreData models and vice versa	MCP	24
7	Design Graphical User Interface	Design all the screens for the application (login, signup, create workouts etc.)	MCP	96
8	Implement design for welcome screens	Implement welcome screens in iOS	MCP	40
8.1	Implement controller for welcome screens	Add controllers to support the data flow for the welcome screens	JBO, MCP	40
			Total Time	288

Sprint Retrospective

A lot of time during this sprint was spent on getting the knowledge and understanding the iOS development in swift programming language and the XCode IDE. The good part was that we had a clear picture formalized, about what we have to achieve from the initial project description, thus we were able to focus on researching how to achieve the desired results instead of gaining general knowledge. Since we had to allocate quite a lot of time for the research the scheduled time to achieve the tasks were underestimated by about 15% and both developers had to put additional effort to finish their tasks. From that we learned to add an additional 10% - 15% to our time estimates for the next sprints.

Some interesting aspects what we encountered during the process were related to the iOS design. In our first design mockups we included some elements that did not fit in the design guidelines of iOS application, thus some mockups had to be changed before starting the implementation. To carry out the task of implementing the desired design into the iOS project a lot of time was spent to understand the story board feature of XCode. In other development environments with which group members had worked before it is not common to have an overall design “board”. It is more common to have each design element to be separated and the connections between the UI elements are usually achieved with some controller actions, but in XCode thats all in one place and takes more time to fully understand all the functionality what it gives to developers.

On the server side some problems were encountered because of the Waterline database adapter. The waterline adapter is still in constant development process and not all features are well developed and tested. For example our domain data models are working with a lot of relations between models and in if we need to perform CRUD operations for the parent model which contains children in several levels, such a feature in Waterline was not yet implemented. To overcome those issues Janis had to find some workaround and later on he submitted a new feature request for Waterline. If Waterline will implement this feature in further releases, then we will have to update our server side code. Overall the sprint went well and the tasks were successfully implemented, tested and documented.

Sprint 2 (September 29th - November 2nd)

On the second sprint we are focusing on adding additional functionality in the iOS application and web server. The main task is to establish the connection between the iOS app and the web server and make sure that we can send some data across. When the connection is established, the JWT authentication must be implemented because it will impact any further communication with the web server. If the task of implementing the JWT authentication is done within the scheduled time then the CRUD functionality can be implemented and tested.

1	Design and Create API Service	Design the initial data flow and communication with the web server over HTTP	JBO	35
2	Implement JWT authentication	Create controllers and UserManager class to sustain the JWT authentication process	JBO	53
3	Implement API functionality to support welcome screens	Create Models and Controllers to support User CRUD functionality	MCP, JBO	158
4	Implement workout routine flow	in iOS implement workout routine flow with all CRUD functionality for nested elements - workouts, exercises, sets.	MCP	53
5	Implement content pulling from danish-aesthetics.dk	Implement in iOS web view to get the content from the danish-aesthetics.dk website	MCP	9
6	Implement Workout Routine CRUD functionality on web server	Create Workout controller and CRUD functionality for Workout Routine model	JBO	106

Sprint Retrospective

During the second sprint we managed to establish the connection between the web server and iOS application and successfully implemented the JWT authentication. The implementation of the JWT authentication was quite a challenge because we had to familiarize ourselves with the Sails.js framework and from documentation and other online resources find out the best practices of how to handle and achieve the desired functionality. In the process all the parts of the web server had to be understood and how the request flow is handled by the framework, so we can make sure that each request, which comes to server, can be authenticated. In the process we found out a really great tool for web server testing called POSTMAN [2], which can be downloaded as Google Chrome extension. The POSTMAN is a tool designed to test RESTful web servers manually. Once the test were successful we replicated the same requests in the iOS application. To make HTTP request in iOS we had to find out the best library to use, which supports attachments in the body and allows to add custom header, so we can send JSON data and add the authentication token in the request header. With all the pieces in place we were able to fully implement the welcome screen data flow and workout routine CRUD functionality.

Sprint 3 (November 3rd - December 9th)

During the third sprint we have to make sure that the CoreData is working correctly with all the relations what the models have so we can parse the data to the web server as JSON objects. As this sprint is ending 3 days before the final report submission deadline our main focus is to finish the report writing and have everything ready by that time.

1	Make unit tests for CoreData	Create unit tests to for CoreData on iOS to test if nested relations behave as expected	MCP	9
2	Add serialization/deserialization of JSON Objects	Implement for API models JSON serialization and deserialization and test if correct values are retrieved and sent from web server	JBO	53
	Finish final reports	Finish writing the project report and process report. Format the report by the standards and proof read	JBO, MCP	144
3	Implement tracking feature	Design and implement in iOS tracking feature for workouts	MCP	211
4	Implement profile features	Design and implement user profile CRUD functionality	MCP, JBO	158
5	Implement workout statistics	Design and implement workout statistics in iOS application	MCP, JBO	158

Sprint Retrospective

During the sprint we finally achieved the full flow of workout routine data between the iOS and Web API with all the nested relations and that sets a good ground for further development of the project. The data is correctly saved in the iOS application and correctly synchronized with the web server, which in turns gets the data from the web servers database. Unfortunately we were not able to finish all the planned task for the sprint because they were out of the time

scope. However we have developed our project to satisfy the objectives of the project and those tasks can be implemented in near future, before the application is submitted in App Store. What took a lot of time during this sprint that we did not expect, was the writing of the reports and some bugs with both the api and iOS application. We had taken the reports and documentation part of the project a bit too lightly and not spent enough time on writing all the necessary documentation during the first two sprints. This resulted in a lack of time, when it came to implementing the tracking and statistics features.
Regarding the report task it is hard to say how successfully we have completed it, because the judgment day will be on our examination day. We have covered and documented in our project report all the major parts of the project and hopefully examiner will be satisfied with the results.

7. Personal Reflections

The overall outcome of this project has been very satisfactory, and a lot of time and effort went in to make it just right. We did however, encounter some obstacles along the way. The most major obstacle, must be the swift programming language. Even though we initially knew, that the language was new, Apple has promised full support for it, starting with their newest release of iOS 8 in September 2014. Therefore, it seems very odd that they would change the syntax multiple times from August through December 2014, but they did. It was especially annoying and frustrating, when at one point, we had different version of Xcode on our different development machines and the project would compile on one Mac and not the other, then when fixed on the other it would not compile on the first. We spent a lot of valuable time, figuring this one out. Another “obstacle”, was the fact that the team did not know anything about iPhone programming when we started, but the learning curve was really steep and we quickly got ahold of most of the new stuff in iOS. One thing was more challenging than the rest though; the CoreData aspect of iOS. We spend a lot of time, trying to pass these models between different controllers, to no avail, until we finally got a grasp on the technology and implementing the rest was a breeze.

With the web server the major issues were related to unfinished features what the framework offer.

8. Literature

[1] Git visual guide - http://marklodato.github.io/visual-git-guide/index-en.html - 24/09/2014
[2] POST man home page - http://www.getpostman.com/ - 20/09/2014