



Seahawk Fitness Requirements Engineering Report

02.01.2016

Sean Zeringue
Project Manager
saz5432@uncw.edu

James Gray
Requirements Engineer
jsg6998@uncw.edu

Weston Jones
Designer
wj8170@uncw.edu

Chase Galloway
Quality Assurance Engineer
ecg4965@uncw.edu





Table of Contents



Introduction

Problem Description

Deciding which classes to attend at the gym that do not conflict with your schedule is no trifling matter to handle at the moment. Currently in order to even see the current group exercise schedule at the Student Recreation Center one would need to log onto the UNCW website, search for the Rec. Center's page and then navigate through the menus to find a group exercise schedule. Those steps alone can take several minutes out of the day. Nevermind the time it takes to look at your own schedule and to compare the two. The goal of this project is to create a simple, easy to navigate, streamlined approach to accessing many of the Rec. Center's currently cluttered functions.

This report will delve into the many requirements that need to be fulfilled for the project to be called a success. First we will begin with a walkthrough of the software project plan, which will breakdown into resources needed as well as many work estimations for how much the software will cost. The software project plan will also breakdown which team members will be doing which tasks as well as identifying any risks that the team may run into. This report will also contain all requirement analysis models that have been developed.


Scope and Objectives

The main objective for this project is to create software that is useful to daily life when it comes to dealing with the Student Recreation Center. A major concern is to develop a tool that is very accessible and easy to use. Giving students and staff something that will allow all operations to run smoothly. Another objective would be to insure that the software UI is easily navigated as well as pleasing to the eye.

The software will be responsible for a wide variety of functions, such as scheduling classes and events, allowing for rental equipment to be checked out, as well as signing up for personal training. All data about classes and equipment must be easily manageable from the UI so that the weekly rotations of available classes may be changed with ease.

Success Criteria

This project will be considered a success if the software is reliable, easy to use, and above all useful. It will be a success if all deadlines were able to be met within acceptable parameters, as well as adequately executed.



Resource Requirements

Type	Name	Description	Availability	Price
Hardware	Mac Computers	To write/test/implement code	High	\$1000
Personnel	Developers	People who actually do the writing/testing/implementation	Medium	\$2000/per week
	Manager	Overseer who wrangles the developers	Low	\$3000/per week
Software	Visual Paradigm	Software which allows for easy creation of diagrams and charts	High	\$15/per month
	SQLite	Software that allows for local database implementation	High	\$0
	Swift with X-Code	iOS IDE and storyboard software	High	\$0

Table 1: List of required resources

Table 1 is a simple list of all required resources that the team will need in order for the completion of the project. The IDE Swift only works well with mac and other apple products development will have to proceed with solely with a mac environment. All personnel needed to complete the project have been listed above. The software will have to include SQLite or another SQL database product in order to provide a database implementation that will allow for easy data management of equipment and classes. Visual Paradigm will allow digitizing of handmade models and design blueprints.

Generalized Work Breakdown

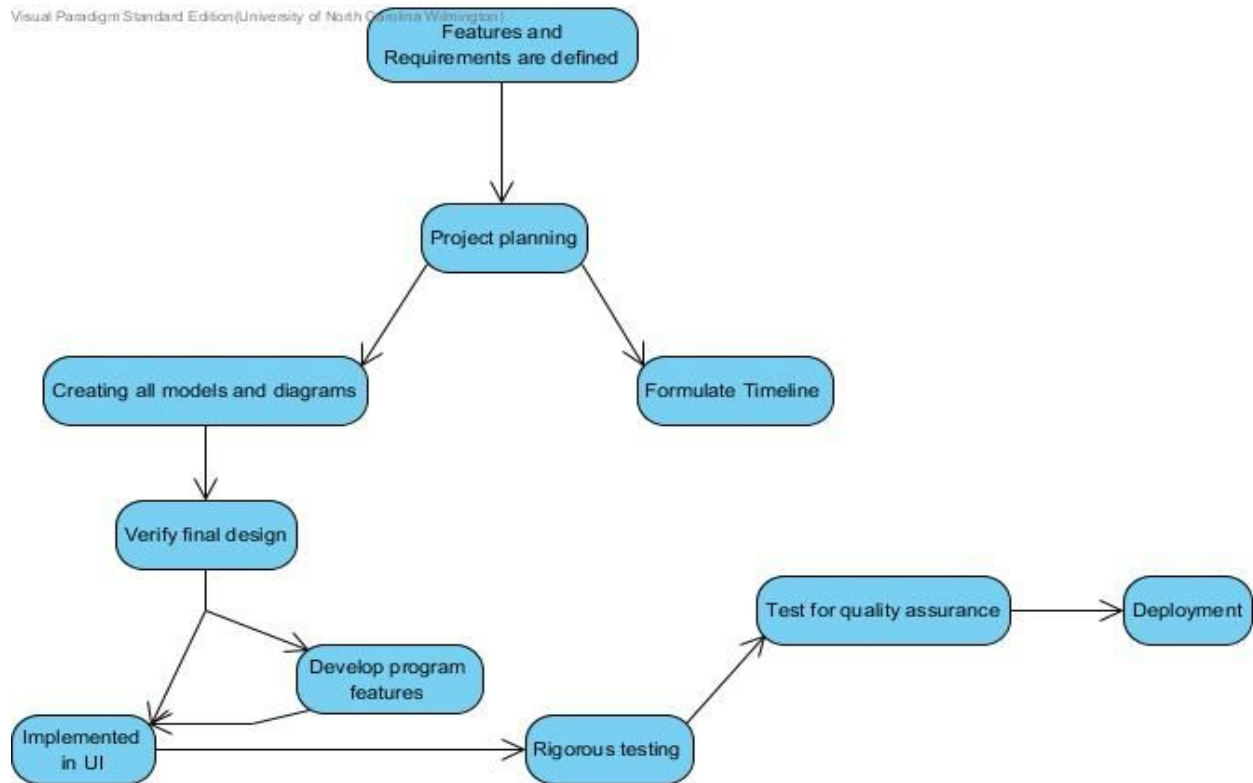


Figure 1: Generalized work breakdown

Figure 1 is a simple work breakdown of tasks that need to be accomplished in order to insure that the project will be a success. It will be crucial for these tasks to be executed cleanly and swiftly.

Cost Estimation

The actual resource requirements are fairly small. Most members of the team already own macbooks and have some familiarity with Swift. UI implementation will be very simple thanks to the functionality that is built into Swift with X-Code.

Function Point Estimation

Domain Value	Count * Weighing Factor	Total
External Inputs	3 * 3	9
External Outputs	2 * 4	8
External Queries	3 * 3	9
Internal Logical Files	5 * 10	50
External Interface Files	2 * 5	10

Count Total = 86

Does the system require reliable backup and recovery?	2
Are data communications required?	1
Are there distributed processing functions?	3
Is performance critical?	4
Will the system run in a existing, heavily utilized operational environment?	0
Does the system require on-line data entry?	3
Does the on-line data entry require the input transaction to be built over multiple screens or operations?	1
Are the master files updated on-line?	0
Are the inputs, outputs, files or inquiries complex?	2
Is the internal processing complex?	1

Is the code designed to be reusable?	1
Are conversion and installation included in the design?	2
Is the system designed for multiple installations in different organizations?	0
F14. Is the application designed to facilitate change and ease of use by the user?	5

Table 2: Function Points

Function Points = $86(.65 + (.01 * 25)) = 77.4$ Function Points

Assuming that the average rate of function point completion is 6.5 function points per month, this project would take a team of four 3 person months. Each function point has a rough cost of \$1,230 meaning that this project would cost \$94,710.

Lines Of Code

Function	Optimistic	Average	Pessimistic	Total
Graphical User Interface	200	500	800	500
BackEnd Database and Queries	150	375	600	375
Rental Service	300	800	1300	800
Calendar	200	550	900	550

Table 3: Lines of Code

Total Lines of code = 2225 LOC

With the Average productivity for systems being 620/LOC per month this project should take the team one person months to implement a project of this scope. Assuming one month is roughly four weeks this project would cost \$32,000 in personnel costs. Which rounds down to being \$14.38 per line of code.

COCOMO II Model

Objects	Quantity	Weight	Total
Screens	10	2	20
Reports	2	4	8
Components	4	10	40

Table 4: COCOMO object points

Total Object Points = 68

$(1 + 0\% \text{ reuse}) * 68 = 68 \text{ NOP}$

PROD = 13

Estimated Effort = $68 / 13 = 5.23$ person months

$5.23 * \$8,000$ (Average monthly salary) = \$41,840

According to the COCOMO model it would take roughly 5 person months to complete a project of this scope. A team of four should be able to complete this project in a little over a month.

Project Schedule

In order to insure timely work a schedule has been created with specific tasks and their required completion dates.

Task	Start Date	Time	End Date
Project Selection	12-Jan	7	18-Jan
Create Models and Use Cases	25-Jan	38	3-Mar
Requirements and Engineering Report	18-Jan	29	16-Feb
Software Design Report	29-Feb	22	22-Mar
Create Database	15-Mar	7	22-Mar
Protoyping Features	22-Mar	19	10-Apr
Implement Rentals	25-Mar	16	10-Apr
Testing	10-Apr	16	26-Apr
Software Implementation Report	1-Apr	25	26-Apr

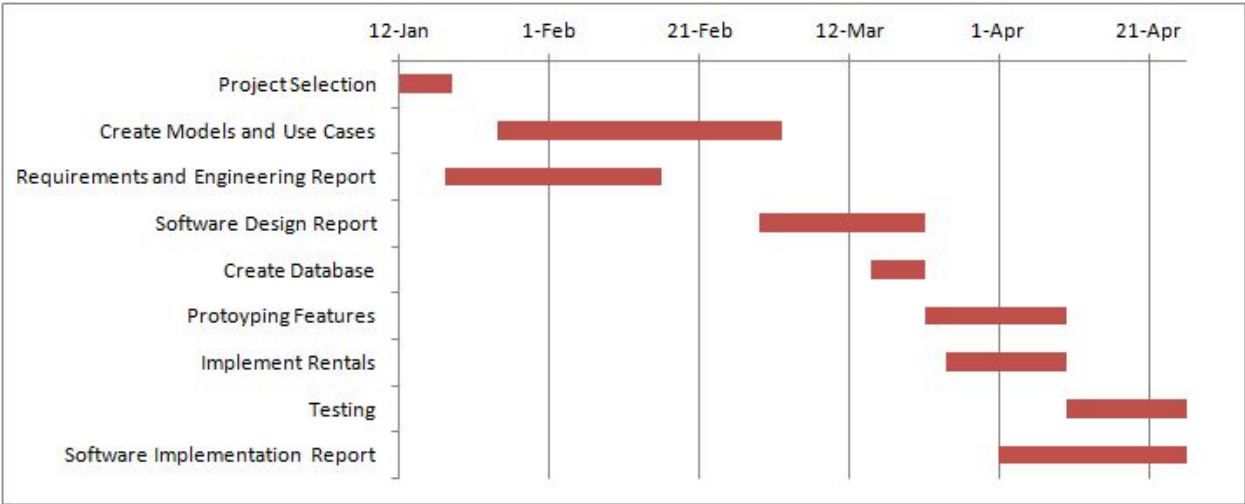


Table 5: Generalized Work Gantt Chart