# Get Gains
for UTM CSCI 352

Reily Stanford and Enrique Tejeda

Abstract

This application, Get Gains, is mainly to appeal to people who frequently go to the gym at least three times a week. This application will take the user's maximums of their three main lifts and fill out a workout program around that. At the moment, we will implement it to only work with one workout plan and are hoping to add more in the future, if possible. Another goal we have in mind is potentially to move the program to mobile since it would be easier to use the application at the gym.

1. Introduction

This project will allow users easily track their progress in their workout routine week by week. We are hoping to add a variety of workout plans that the user can choose from. Every three weeks, the user will have to enter their new max lifts per the three main workouts. The application will take these three weights and base their workout plan for the next three weeks off of them.

We are aiming to help a people who go to the gym often but don't know what workout plan to follow. We are intending to lessen the effort that comes with making and following a plan. Without the program, user's would have to calculate how much weight they would need to lift per day, which ends up being tedious and overall discouraging.

1.1. Subsection Heading Here

Occasionally you need to break your sections into separate parts, you will likely not need a subsection for every section

1.1.1. Subsubsection Heading Here. Occasionally you will need to break your subsections into separate parts, if you find yourself using this often, you're likely going overboard. Don't try to go any lower down than this. (And make sure you remove this!)

1.2. Background

The reader/user will need to know what the terms reps, sets, AMRAP, and max. A rep is a single complete motion of a workout. A set is a group of a given rep amount. AMRAP is the max number of reps per set you are capable of doing.

Back in our first semester of college, we used to go to the gym on a regular basis. During this time, we followed the workout plan that this program uses. We would often times forget the amount of weight we needed to do based on our week in the regiment. This is why this project was started, we intended to make it much easier for anyone to follow this plan if they so desired.

1.3. Impacts

This program will impact people's health for the better and keep the user motivated with a progress tracker so they notice the changes they are experiencing.

1.4. Challenges

We feel as though, trying to incorporate save files is going to cause problems. Trying to incorporate multiple pages so that when the user presses a button, it would hide everything on the menu they are currently on and show a different menu, will be a challenge.

2. Scope

We will consider our project done when the program is set up to allow the user to enter their max lift weights and have a schedule drawn up for a three week period of time.

One of our stretch goals is to make the program dynamically calculate weight based on what the user is capable of doing at that time instead of being based on a single entry. Another stretch goal is to allow the user to choose between different workout plans.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.1.1. Functional.

- User needs to have a private shopping cart – this cannot be shared between users, and needs to maintain state across subsequent visits to the site
- Users need to have website accounts – this will help track recent purchases, keep shopping cart records, etc.
- You'll need more than 2 of these...

2.1.2. Non-Functional.

- Security – user credentials must be encrypted on disk, users should be able to reset their passwords if forgotten
- you'll typically have fewer non-functional than functional requirements

2.2. Use Cases

This subsection is arguably part of how you define your project scope (why it is in the Scope section...). In a traditional Waterfall approach, as part of your requirements gathering phase (what does the product actually *need* to do?), you will typically sit down with a user to develop use cases.

You should have a table listing all use cases discussed in the document, the ID is just the order it is listed in, the name should be indicative of what should happen, the primary actor is typically most important in an application where you may have different levels of users (think admin vs normal user), complexity is a best-guess on your part as to how hard it should be. A lower number in priority indicates that it needs to happen sooner rather than later. A sample table, or Use Case Index can be seen in Table 1.

Use Case Number:	1
Use Case Name:	Add item to cart
Description:	A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

You will then go on to (minimally) discuss a basic flow for the process:

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

You may need to also add in any alternative flows:

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

You will often also need to include pictures or diagrams. It is quite common to see use-case diagrams in such write-ups. To properly reference an image, you will need to use the `figure` environment and will need to reference it in your text (via the `ref` command) (see Figure 1). NOTE: this is not a use case diagram, but a kitten.

After fully describing a use case, it is time to move on to the next use case:

Use Case Number:	2
Use Case Name:	Checkout
Description:	A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

You will then need to continue to flesh out all use cases you have identified for your project.

Figure 1. First picture, this is a kitten, not a use case diagram

2.3. Interface Mockups

At first, this will largely be completely made up, as you get further along in your project, and closer to a final product, this will typically become simple screenshots of your running application.

In this subsection, you will be showing what the screen should look like as the user moves through various use cases (make sure to tie the interface mockups back to the specific use cases they illustrate).

3. Project Timeline

Go back to your notes and look up a typical project development life cycle for the Waterfall approach. How will you follow this life cycle over the remainder of this semester? This will usually involve a chart showing your proposed timeline, with specific milestones plotted out. Make sure you have deliverable dates from the course schedule listed, with a plan to meet them (NOTE: these are generally optimistic deadlines).

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 2.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

Figure 2. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.