

Software Requirements Specification

CS 320

Calculation Utility

Version <0.1>

Group Name: “Segmentation Fault: 11”

|  |  |  |
| --- | --- | --- |
| Daniel Rehman | 11458334 | daniel.rehman@wsu.edu |
| Jake Palmer | 11634568 | jake.palmer@wsu.edu |
| Andew Oakes | <student #> | <e-mail> |

|  |  |
| --- | --- |
|  |  |
| Date: | 10.05.18 |
|  |  |
|  |  |
|  |  |

Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 1

2 Overall Description 2

2.1 Product Perspective 2

2.2 Product Functionality 3

2.3 Users and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 7

4 Other Non-functional Requirements 8

4.1 Performance Requirements 8

4.2 Safety and Security Requirements 8

4.3 Software Quality Attributes 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft, 0.1 | Daniel Rehman, Jake Palmer, Andrew Oakes | Initial Draft of Document | 10/05/18 |

# 

# Introduction

This project is a Calculation Utility, which allows the user to calculate numbers using a calculator, graph user defined functions, convert existing units to other units, define new units, visualize the unit conversion using graphics, as well as list all current units defined. It is a web appllication written in HTML, CSS, and Javascript.

## Document Purpose

This document serves as a comprehensive overview of version 0.1 of our Calculation Utility project. The project is in its initial stages and ideas are still being formed, so the scope and details of functionality described here are tentative. However, this document will cover the projects details and organization as best as possible.

## Product Scope

As stated in the description, we will be creating a "Calculation Utility" which works similarly to a calculator, but each value used in the calculation will have an associated unit. The user shall be able to define units which have a relationship with basic units, or previously defined units. The output will have a selected unit, and the input value will be computed and converted to the selected output unit. For select unit types there shall be a visual comparison between the input and the output scales (Length for sure). The program shall have a web based interface which will display a list of currently defined units, and give the user the ability to add a new unit in terms of previously defined units. There may be a graphing utility in the future which can graph given functions with scales defined by the units that the function should be returning.

## Intended Audience and Document Overview

The Users of this document are the professor, and the web clients who use the Calculation Utility web application, as well as the developers of this application, which include the members of this project.

## Definitions, Acronyms and Abbreviations

None were used.

## Document Conventions

None were used.

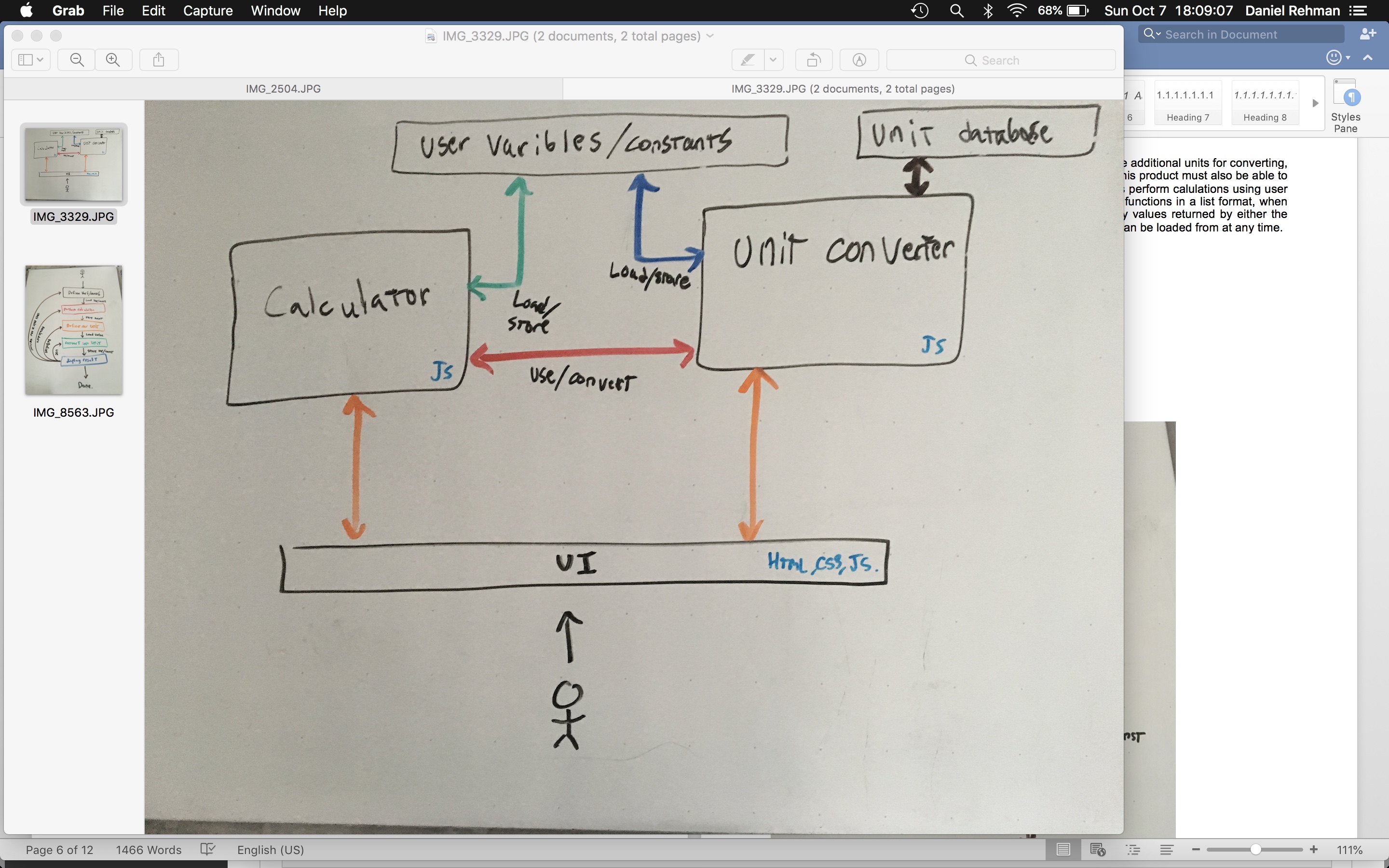
## References and Acknowledgments

None were used.

# Overall Description

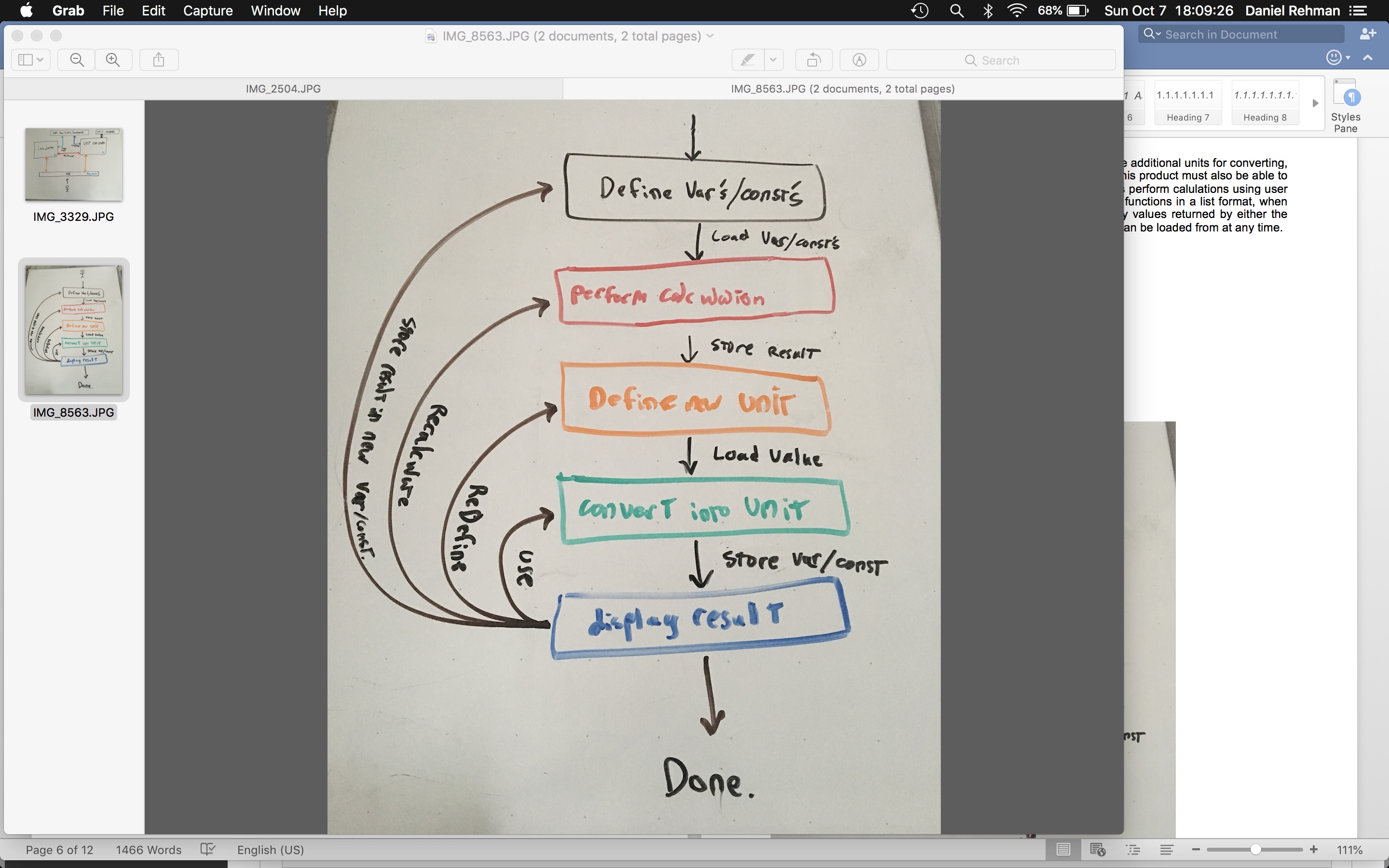
## Product Perspective

The Calulation Utility is a new, standalone web application. It can be used by anyone with access to the internet, but would primarily be of use to engineers, mathematicians, and those in similar fields. The UI allows the user to select the desired units to convert to and from, to define a new unit, or to use the calculator. Results of a unit conversion or calculation are displayed to the user. A separate page displays a list of all available units and their descriptions.



## Product Functionality

The product, Calculation Utility, must be able to convert units, define additional units for converting, and list available units already defined. As it contains a calculator, this product must also be able to perform basic arithematic on numbers given by the user, as well as perform calulations using user defined functions, and make a tumb-nail graph these user defined functions in a list format, when the user views these functions. It should also be able to store any values returned by either the converter or the calculator in a set of variables and constants, that can be loaded from at any time.



## Users and Characteristics

The users that we anticipate are possibly engineers, scientists, and applied mathmaticians, all of which might wish to have a calulating uility which can deal with units. all three of these user bases are important to satisfy, however giving accurate service to the engineers would be possibly the most important, as these users might be in charge of making structures in the real world based on this product.

## Operating Environment

The operating enviorment is a web browser on possibly any machine, and as such it needs to be efficient and responsive.

## Design and Implementation Constraints

The primary constraints on this product are the required programming languages and development schedule. To meet the project requirements, the program must be created using HTML, CSS, and Javascript. Development must also follow a strict schedule, and be completed by the end of the Fall 2018 school semester. These constraints will limit the number of features that can be implemented.

In terms of functionality constraints, the product must be able to deal with numerical user input, and perform computations on that input which might have units. it must also be able to convert between units accurately.

## User Documentation

User documentation will exist solely on the website as part of the UI. The product is fairly simple, so simple directions placed directly in the website should be sufficient to guide the user.

## Assumptions and Dependencies

This product will use the Javascript Underscore Library.

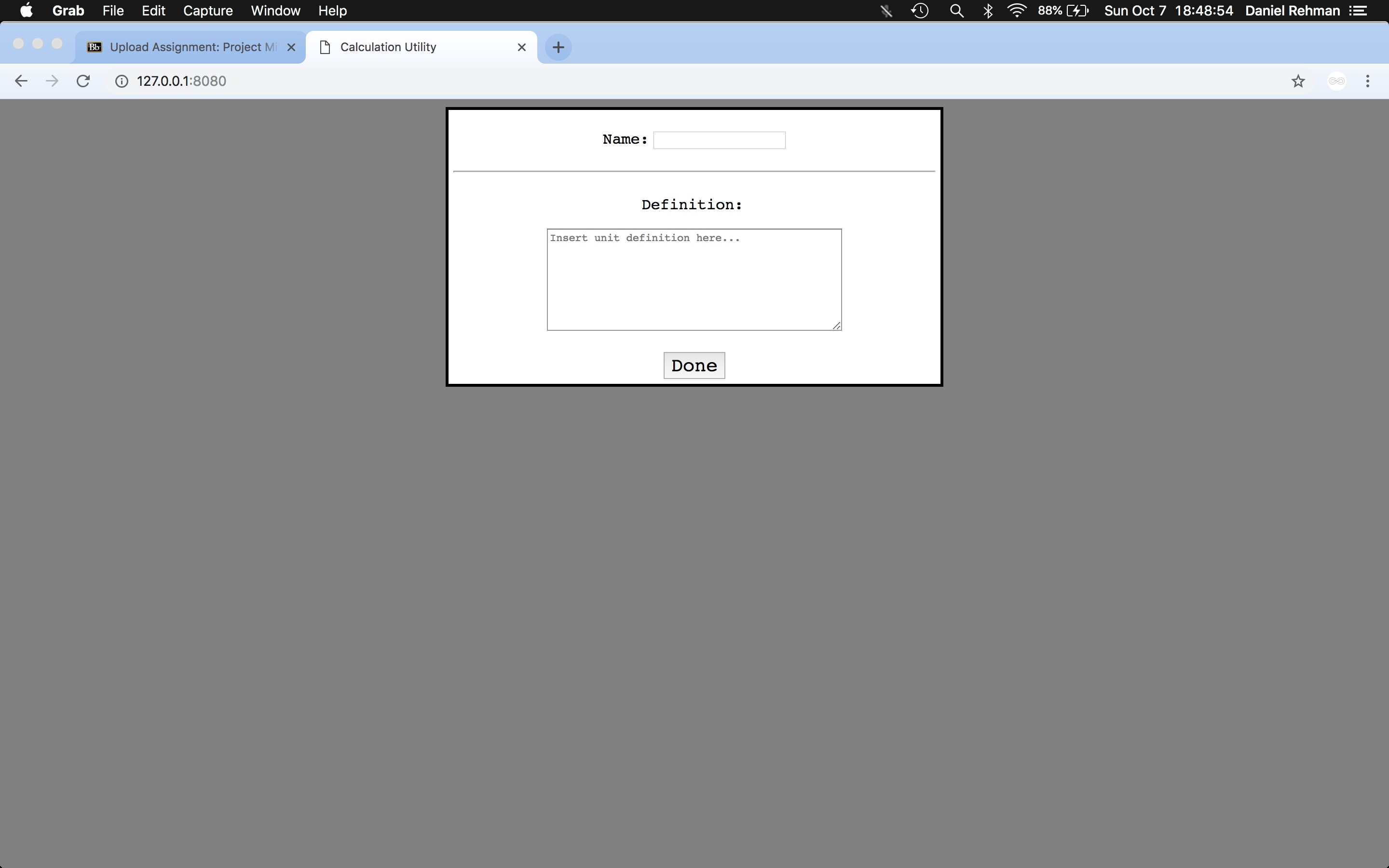
# Specific Requirements

## External Interface Requirements

### User Interfaces

The User Interface will have two main parts, each with multiple views. The major parts are the calaculator and the unit converter. The calulator will have a typical calculator interface, as well a button that goes to a view which displays a list of the current user defined functions, and their thumbnails, and names. There is also a utility to list all the current users defined values, (variables, and cosntants).





In the unit converter, there are three main views: the main view which allows the user to convert between two units, a view to define a new unit based on existing units, as well as a a view to list all the existing units, and their definitions.

### Hardware Interfaces

Because this application is running in a web browser, it will interface with the hardware of the computer running the web browser. If a graphics libary is used, then this hardware will include the Users GPU, if it exists.

### Software Interfaces

Because the software uses the underscore library, we interface with underscore using its API. Furthermore, we are interfacing with HTML CSS, and JS, because this application runs in a web browser. This is the extent of our external software interfacing, as most interfacing is internal to the application.

### Communications Interfaces

Because we are using a web browser, we are using the protocol "HTTP"/"HTTPS", at some point.

## Functional Requirements

## 3.2.1 Convert Units:

The main UI provides a space to enter a value and its units, and a space to select the unit to convert to. When the 'Convert' button or Enter key is pressed, the program calculates and displays the result of the conversion.

## 3.2.2 Define New Units:

The main UI contains a button to 'Define New Unit'. This opens a new UI where the user can enter the name, symbol and description of the new unit, and a formula that defines the unit.

## 3.2.3 List Available Units:

The main UI contains a button to 'List Available Units'. This links to a separate page that lists all default and user defined units. Each entry contains the unit name, symbol, and description.

## 3.2.4 Calculator:

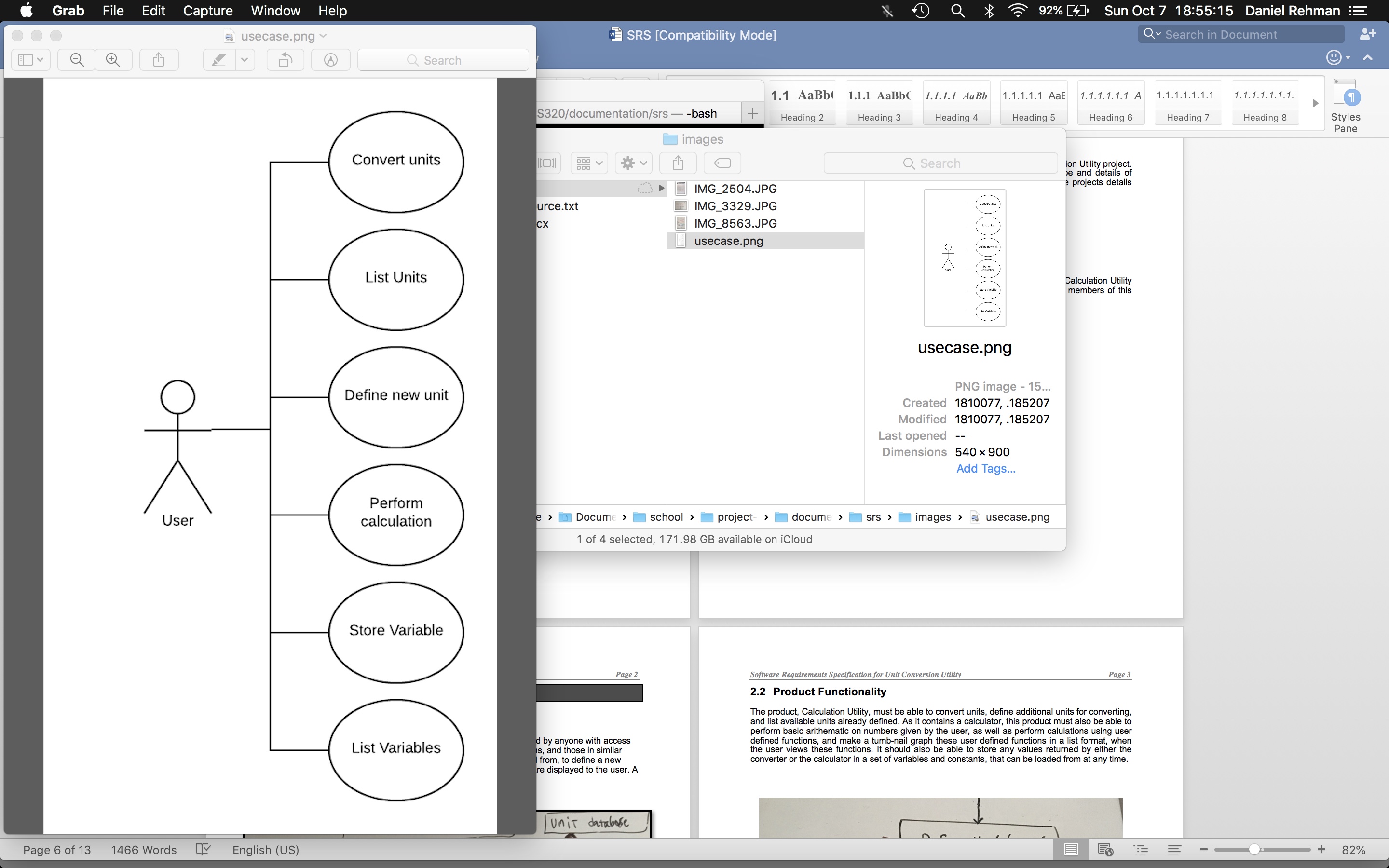
The main UI contains a calculator that can perform basic arithmetic.

## 3.2.5 Variables and Functions:

The main UI contains a way to define new numerical variables (or constants) and user defined functions, and list all current variables (or constants) and functions.

## Behaviour Requirements

### Use Case View



# Other Non-functional Requirements

## Performance Requirements

The software needs to be responsive, as the users (whether engineers or mathmeticians) probably do not want to wait more than a couple seconds (eg. 3 seconds.) for a single simple calculation/conversion. Thus, the user interface, and internal logic needs to be written efficently.

## Safety and Security Requirements

There is no direct safety requirements of the system, as it is running in a web browser. However since engineers might be users of this system, giving back an accurate answer is important, as giving a wrong answer might possibly lead to incorrect systems being built by the users.

Safety Requirements:

1. give correct/accurate answers to calculations.

2. give correct/accurate answers to unit conversions.

## Software Quality Attributes

4.3.1 Readability: The software shall be readable by anyone working on the project. We can achieve this by using a standardized coding style and including comments describing functionality.

4.3.2 Testable: The software shall be testable; every piece of code should have a something that tests that it is functional. We can achieve this by using TDD or a similar development cycle.

4.3.3 Reusability: The code should be written in such a way that it can be reused, and when some piece of the program is similar to something that is already written, what is already written should be used. To achieve that, code should regularly be refactored to reflect this practice.

Appendix A – Data Dictionary

*<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>*

Appendix B - Group Log

September, 2018 – Group formed.

October 5th, 2018 – Met at WSU Library. Decided on our project, set up Git Repo, and began SRS Version 0.1

October 7th, 2018 – Collaborated on Discord server. Completed SRS Version 0.1.