University of Cape Town

Twiddle Lock

EEE3096S_MINI_PROJECT_A

Muzanenhamo Tawanda
Wanna McMoses
10-22-2018

Contents

INTRODUCTION 2
REQUIREMENTS 3
SPECIFICATION 4
State Chart 4
Class Diagram 4
Flow Diagram 5
IMPLEMENTATION 6
Password Storage 6
Unsecure Validation 6
VALIDATION AND PERFORMANCE 7
Scenario 1: 8
Scenario 2: 8
Scenario 3: 8
Scenario 4: 8
CONCLUSION 8
REFERENCES 10

INTRODUCTION

The Twiddle lock system is an electronic model of a dial combination safe.

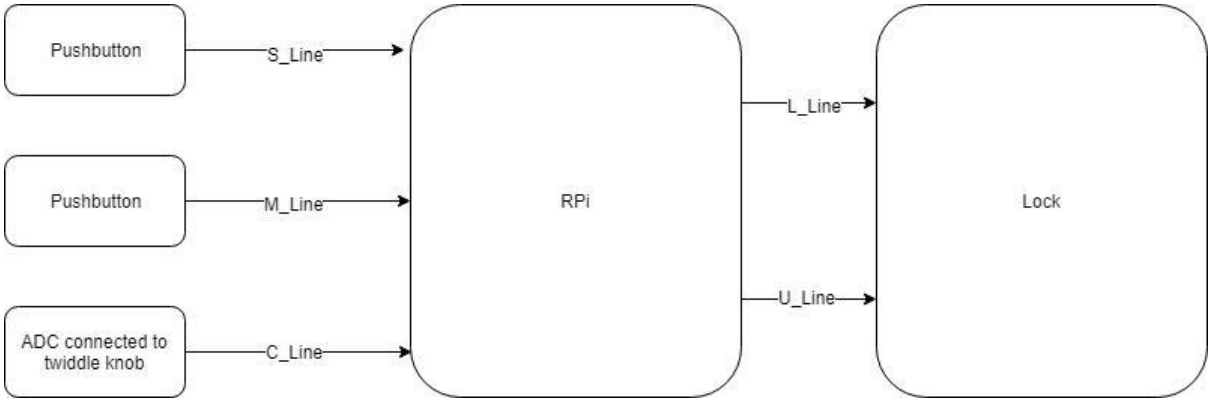


Figure 1:System input and output

The diagram above gives an overview of the system's input and output protocol. Hence we have 3 input lines; S_Line, M_Line and C_line. S_Line is a pushbutton which instructs the lock to get ready to take in a dialled combination. The lock has 2 modes unsecure and secure mode and to toggle between these modes there is a pushbutton that is connected to the M_Line. Our dialled combination is modelled by the varying values in a potentiometer. A single character in a dialled password is the combination of time taken to move the potentiometer in a desired direction and the particular direction you were moving the potentiometer in. All of the above mentioned input is taken in by the C_Line. After the user inputs a password the rpi validates that it is correct and follows 1 of two paths; L_Line or U_Line. When output follows the L_Line it means that the password is incorrect and a red LED is tuned on for 2 seconds. However, when the password is correct a green LED is turned on for 2 seconds meaning that the U_Line is HIGH.

Presentation of this report is based on a mixture of diagrams and theoretical explanation of the different sections covered. The reader will first be given a diagram which is aimed to give an overarching explanation of the theory to come. And then more in depth theory will be given to iron out all details which cannot be described from a diagram.

REQUIREMENTS

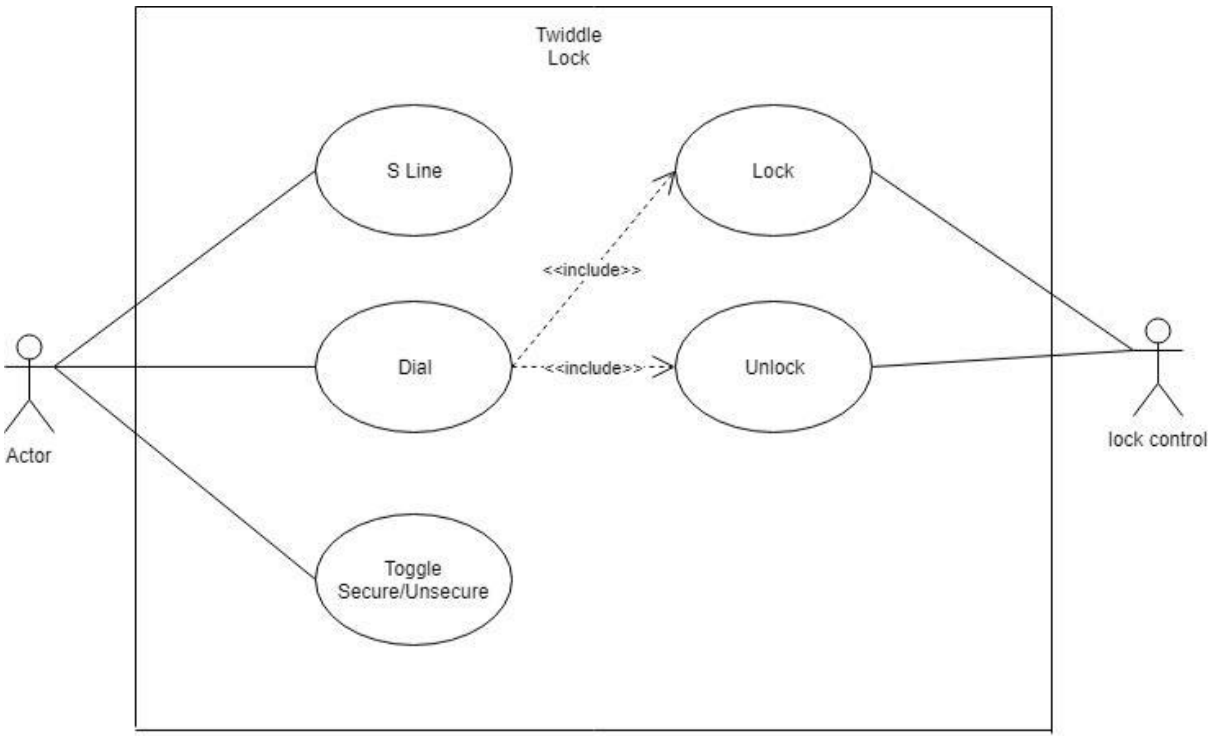


Figure 2: Use Case Diagram

Twiddle lock has the following system requirements which will be outlined based on these headings; input, process and output. From there all additions and deviations from the original specifications will be discussed in detail.

Input:

S_Line: The service line that sets the twiddle lock to take in a combination

M_Line: A function that toggles between secure and unsecure mode

C_Line: The dial from which a combination is entered

Process:

Once the S_Line is pressed the following may take place. The rpi will allow the user to continuously dial the knob until he/she does not make any more entries for 2 seconds. Then it will move onto password checking.

Password is correct

- The U_Line is held HIGH for 2 seconds
- A celebratory sound is played

Password is incorrect

- The L_Line is held HIGH for 2 seconds
- A crestfallen sound is played

Regardless of a correct or incorrect password the rpi will go back to idle mode and wait for the user to give input.

To aide in mode selection a M_Line was added to allow the user to change between different security modes.

Once the M_Line is pressed the following takes place.

The mode toggles between unsecure and secure mode
The rpi returns to idle mode

SPECIFICATION

State Chart

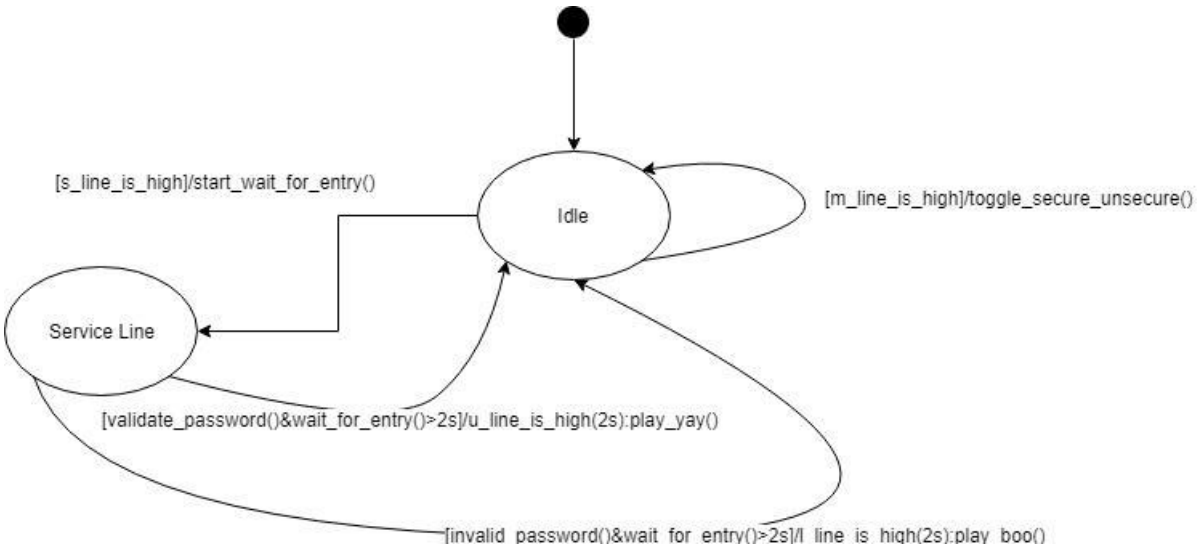


Figure 3: State Chart Diagram

Class Diagram

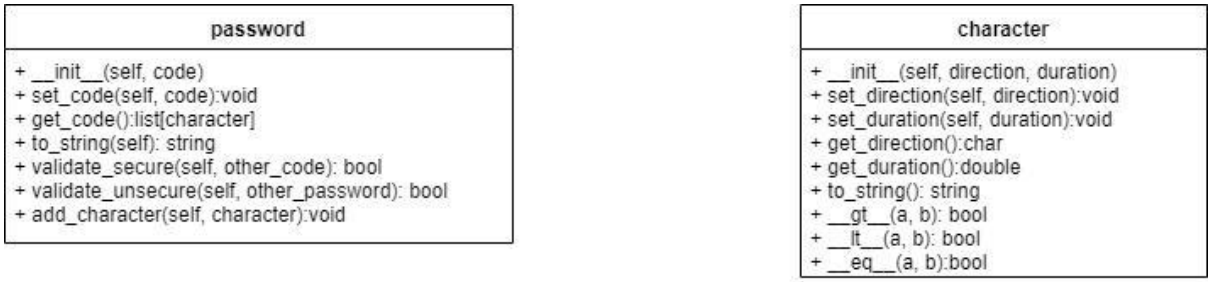
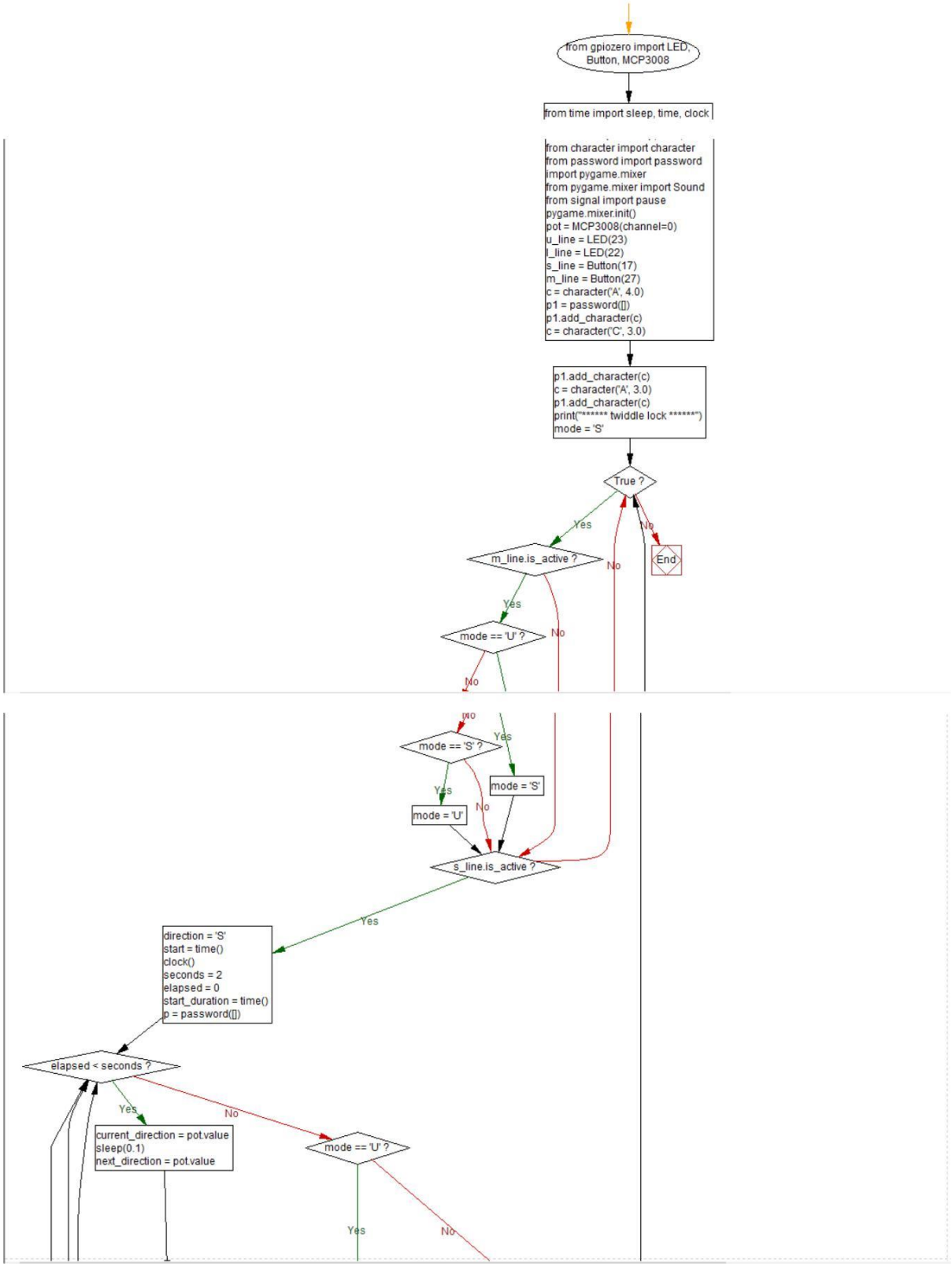


Figure 4: Class Diagram

Flow Diagram



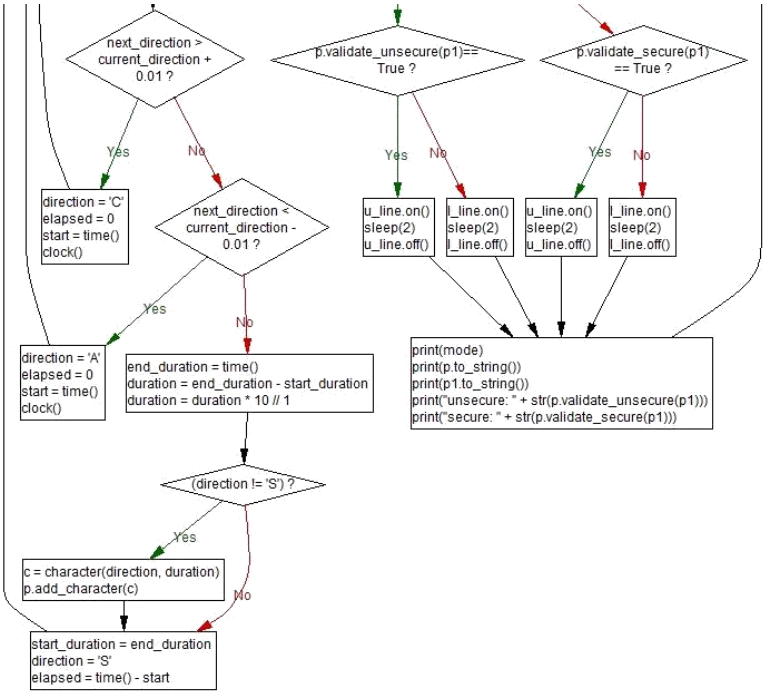


Figure 5: Flow Chart

IMPLEMENTATION

Password Storage

```
""" add a character to code """
def add_character(self, character):
    self.code.append(character)
```

Figure 6: Add Character to Password

Character.py is a class that models a single entry of duration and direction. Duration being the time taken to move the knob in a single direction and either come to a stop or change direction. And direction being the direction that the dial is moved until you come to a stop or change direction.

The password that the person enters would be saved as a list of characters which is stored in password.py.

Unsecure Validation

```
""" define greater than operator """
def __gt__(a, b):
    return a.get_duration() > b.get_duration()

""" define less than operator """
def __lt__(a, b):
    return a.get_duration() < b.get_duration()

""" define equals to operator """
def __eq__(a, b):
    return a.get_duration() == b.get_duration()
```

Figure 7: Override Default Operators

Unsecure validation does not take into consideration the direction or sequence of moves made when making an entry. Validation in this mode required a way to ignore the above attributes of a password and only focus on ensuring that the characters entered contain the same durations.

To know that two password objects have the same durations python allows comparison of lists of characters by first sorting them and then converting them to strings and equating the two strings.

However python does not have a native sort function for character object. Hence I had to override __gt__, __lt__ and __eq__(greater than, less than and equals to) operators. These methods were then configured to look at the duration attribute of each character in the password to sort the password list.

```
    def validate_unsecure(self, other_password):
        """ validate code in unsecure mode """
        return sorted(self.code)==sorted(other_password.code)
```

Figure 8: Validate in Unsecure Mode

From there password objects can be sorted, converted to string and then equated to find out if a password is correct in unsecure mode.

VALIDATION AND PERFORMANCE

```
****** twiddle lock ******
S

A 3.0 C 3.0 A 4.0
unsecure: False
secure: False
S
A 3.0 C 3.0 A 4.0
A 3.0 C 3.0 A 4.0
unsecure: True
secure: True
S
A 3.0 C 3.0 A 3.0
A 3.0 C 3.0 A 4.0
unsecure: False
secure: False
S
C 4.0 A 4.0 C 4.0
A 3.0 C 3.0 A 4.0
unsecure: False
secure: False
U
C 3.0 A 4.0 C 3.0
A 3.0 C 3.0 A 4.0
unsecure: True
secure: False
```

Figure 9: Test Cases

Tests carried out above show a variety of scenarios which a user will encounter when running the application.

Debug messages are configured as follows:

The mode the user is currently in (S=secure, U=unsecure)
The dialled combination from the user (Dir_1 Dur_1, Dir_2 Dur_2, Dir_3 Dur_3, …, Dir_N Dur_N) where Dir = direction and Dur = duration. Dir is split into 2 values Clockwise ('C') and Anticlockwise ('A'). And Duration is a number in milliseconds.
The combocode that is already stored in the program (A 3.0 C 3.0 A 4.0)
Validation in unsecure mode (True or False)
Validation in secure mode (True or False)

Scenario 1:

No combination code entered in and 2 seconds elapse

Scenario 2:

Secure mode and the correct sequence is entered

Scenario 3:

Secure mode and an incorrect sequence is entered

Scenario 4:

Unsecure mode and only the correct durations are dialled in.

CONCLUSION

Twiddle lock in the end achieved all system requirements. The following functionalities were outlined

Table 1: List of Requirements

Functionality	Achieved
1. Compare dialled code with code hardcoded into the system	Yes
2. Twiddle lock must take in code if S_Line is HIGH	Yes
3. Code is compared only if S_Line is HIGH and user does not move the dial for more than 2 seconds	Yes
4. Allow user to toggle between secure and unsecure mode	Yes
5. Set U_Line HIGH for 2 seconds if code is correct	Yes
6. Set L_Line HIGH for 2 seconds if code is incorrect	Yes
7. Play a celebratory sound if code is correct	Yes
8. Play a crestfallen sound if code is incorrect	Yes

Since every functionality was successfully implemented the project is conclusively a success as well.

Determining product success

Table 2: Future Product Success Finder

1. *Does it have unique features?* You can't roll out the "same-old, same-old." Your product has got to have a cool new look that'll make the consumer sit up and take notice.	*1. No*
2. *Does it have mass appeal?* In other words, is it something that will sell to the stay-at-home mother of four as well as the seasoned fisherman?	*2. No*
3. *Does it solve a problem?* Think of something around the house that's troublesome and invent a solution. If your product doesn't solve a problem, you've got a potential problem – consumers aren't as likely to buy it.	*3. Slightly*
4. *Is there a powerful offer with a supportive cost of goods?* The time-tested pitch– But wait, there's more! – is a proven winner. The key is great value at the right price. In today's world, people immediately check the Internet for the same product at a cheaper price.	*4. No*
5. *Can you easily explain how it works?* There has to be an easy-to-understand explanation of how and why your product works. Get your elevator pitch ready. If it takes a college degree to understand the pitch, it's too complicated. You only grab people for a couple of seconds – so you have to tease, please and seize the consumer.	*5. Yes*
6. *Is there a magical transformation or demo?* Before-and-after spots – showing easily noticeable differences – are powerful marketing tools.	*6. No*
7. *Is it multifunctional?* Think like your competitor. If you come out with a product that has just one function, your competitor can steal your thunder – and your sales – with a similar product that offers more functions.	*7. No*
8. *Is it credible; are there testimonials?* An "actual customer" promo is ten times better than any "actor portrayal." Real people offer real results. But you should also seek out professional testimonials from industry associations, doctors and other "experts" in your industry to further build your product's credibility.	*8. No*

9. *Are there proven results?* Be prepared to back up your claims with unshakeable success stories or scientific studies, including third-party clinical studies or reviews from product-testing labs that support your claims.	*9. No*
10. *Can you answer the questions the viewer is thinking?* You must be prepared for any and all questions that could arise over your product. Put yourself in the shoes of consumers, and think of all the questions they could ask.	*10. No*

The above table gives measures that can predict product success from this grading it is clear that the lock cannot work in the real world. As it is not a new idea, static in functionality and does not have any proof of credibility.

REFERENCES

Aivosto.com. (2018). Visutin: See Visustin Flow chart generator. [online] Available at: http://www.aivosto.com/visutin.html [Accessed 22 Oct. 2018].

Harrington, K. (2018). 10 Qualities Of A Successful Product. [online] Forbes. Available at: https://www.forbes.com/sites/kevinharrington/2013/10/08/10-qualities-of-a-successful-product/#26c8f64c126b [Accessed 22 Oct. 2018].

Table 1: List of Requirements 8
Table 2: Future Product Success Finder 9

Figure 1:System input and output 2
Figure 2: Use Case Diagram 3
Figure 3: State Chart Diagram 4
Figure 4: Class Diagram 4
Figure 5: Flow Chart 6
Figure 6: Add Character to Password 6
Figure 7: Override Default Operators 6
Figure 8: Validate in Unsecure Mode 7
Figure 9: Test Cases 7