SE 3XA3: Software Requirements Specification PasswordProtectionProgram

Team 28, Tuples1
Shabana Dhayananth dhayanas
Suhavi Sandhu sandhs11
Joseph Lu luy89

October 7, 2017

Contents

1 Project Drivers **1**
- 1.1 The Purpose of the Project . . . 1
- 1.2 The Stakeholders . . . 1
- 1.3 Mandated Constraints . . . 1
  - 1.3.1 Solution Constraints . . . 1
  - 1.3.2 Partner or Collaborative Applications . . . 2
  - 1.3.3 Off-the-Shelf Software . . . 2
  - 1.3.4 Budgeting Constraints . . . 2
  - 1.3.5 Schedule Constraints . . . 2
- 1.4 Naming Conventions and Terminology . . . 4
- 1.5 Relevant Facts and Assumptions . . . 5

2 Functional Requirements **5**
- 2.1 The Scope of the Work and the Product . . . 5
  - 2.1.1 The Context of the Work . . . 5
  - 2.1.2 Individual Product Use Cases . . . 6
- 2.2 Functional Requirements . . . 9

3 Non-functional Requirements **10**
- 3.1 Look and Feel Requirements . . . 10
- 3.2 Usability and Humanity Requirements . . . 11
- 3.3 Performance Requirements . . . 12
- 3.4 Operational and Environmental Requirements . . . 13
- 3.5 Maintainability and Support Requirements . . . 13
- 3.6 Security Requirements . . . 13
- 3.7 Cultural Requirements . . . 14
- 3.8 Legal Requirements . . . 14
- 3.9 Health and Safety Requirements . . . 15

4 Project Issues **15**
- 4.1 Open Issues . . . 15
- 4.2 Off-the-Shelf Solutions . . . 16
- 4.3 New Problems . . . 16
- 4.4 Tasks . . . 16
- 4.5 Migration to the New Product . . . 17
- 4.6 Risks . . . 18

4.7 Costs 18
4.8 User Documentation and Training 18
4.9 Waiting Room 18
4.10 Ideas for Solutions 18

List of Tables

1 **Revision History** ii
2 Terminology 4
3 Functional Requirements 9
4 Off-the-Shelf Solutions 16
5 Deliverables 17
6 Implementation Breakdown 17

List of Figures

1 Use Case Diagram 6
2 Business Data Model 7
3 2 Factor Authentication FileCloud (2016) 19

Table 1: **Revision History**

Date	**Version**	**Notes**
Oct 6, 2017	1.0	section 1, 2, 5
Oct 7, 2017	1.1	section 3, 4, 6, tables, figures

This document describes the requirements for PasswordProtectionProgram. The template for the Software Requirements Specification (SRS) is a subset of the Volere template Robertson and Robertson (2012).

1 Project Drivers

1.1 The Purpose of the Project

The purpose of this project is to implement an encrypted password manager, PasswordProtectionProgram (PPP) wherein a person can safely store and access all of the passwords they use with a single master password. Through working on this project the software team also hopes to learn about encryption methods and the development process.

1.2 The Stakeholders

The stakeholders of this project are:

- Users of online services that want a place to store their passwords safely and generate stronger passwords
- Services for which the passwords are created as they have to deal with security threats and users that forget their passwords
- Identity thieves since they are able to breach users personal information due to weak passwords and weak encryption methods
- The development team, as they are the students attempting to solve the problem at hand

1.3 Mandated Constraints

1.3.1 Solution Constraints

Description: The product shall operate on all Operating Systems with python installed. (i.e. Windows, OSX, Linux) through a gui app.
Rationale: The client will use all of these operating systems
Fit criterion: The product shall be approved by testing groups of each operating system.

1.3.2 Partner or Collaborative Applications

The product does not have any direct partner or collaborative applications. This product is fully built on python and MariaDB and will be able to be executed in the desktop.

1.3.3 Off-the-Shelf Software

The off-the-shelf software required for this product to be implemented:

- Python
- MariaDB
- Tkinter (python library)

All the software that is required is open source and can be found on the internet.

1.3.4 Budgeting Constraints

The budget of our project is zero dollars so no further purchases are required.

1.3.5 Schedule Constraints

There is a deadline in December for the project to be completed.

1.4 Naming Conventions and Terminology

Table 2: Terminology

Term	Definition
AES	Abbreviation for the Advanced Encryption Standard, a symmetric cipher that is implemented in the cryptography library that we will be using
Authentication	The process of verifying the identity of the user account holder
Cipher	Another term for cryptographic algorithm
Crypographic Algorithm	A means of altering data from a readable form to a protected form (and vice-versa)
cryptography	Python library that will be used to encrypt and decrypt passwords
Dictionary Attack	A hacking technique that tries to bypass internet security by determining passwords or encryption keys using a very large number of likely possibilities
Encryption	The process of converting information or data into a code, especially to prevent unauthorized access
Hash	A value returned by a cryptographic hash function, a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size decreasing the likelihood of a dictionary attack
Key	A piece of information that determines the output of a cryptographic algorithm
MariaDB	A community-developed fork of the MySQL relational database management system which will be used to locally store usernames and databases
Master Password	The password that the user inputs to access the stored usernames and passwords
Module	A Python object that one can bind and reference by importing, contains definitions and statements
Padlock	The original password management software
pip3	A package management system used to install and manage software packages for Python 3
PW	Abbreviation for password
Python 3	The language that will be used to develop the product
random	Python module that will be used for generating passwords
Salt	A random piece of data used to enhance the one-way function that hashes a password, thereby decreasing the likelihood of dictionary attacks
Symmetric Cipher	The specific cipher that will be used for encryption, the key that it employs for going from readable form to protected form is the same

1.5 Relevant Facts and Assumptions

Some relevant facts regarding this project are that the original program has 29850 lines of code. The program has a GNU General Public Licence v3.0 licence and can be run on Windows, Unix and Linux operating systems as well as iOS and Android mobile devices. The team will be using symmetric encryption from the cryptography library, which implements AES with a 128-bit key for encryption. AES is chosen by the U.S. government to protect classified information throughout the world to encrypt sensitive data. A relevant fact about passwords that has to do with the project is that passwords are easily hacked because humans follow similar patterns. For instance, the numbers 1 and 2 are common and capital letters are often used at the beginning of passwords. Lastly, it should be noted that the original product implemented AES and used PBKDF2, a key derivation function that reduces the chance of brute force attacks.

Some assumptions made the developers are that the cryptography library will be enough for our needs and does not need to be tested for its encryption since that would require executing brute force attacks. Also, it is assumed that there is a need for users to store their usernames and passwords safely. Lastly, it is assumed that Python code can run on Windows, Unix and Linux environments.

2 Functional Requirements

2.1 The Scope of the Work and the Product

We will be re-implementing the original open source software (Padlock) as an offline, desktop application, suitable for Windows, Macintosh or Unix environments. This is due to time constraints set by the project deadline as well as further security by not having the product online. Also, the Python cryptography library will be used for the encryption as the development team does not have experience creating secure encryption methods.

2.1.1 The Context of the Work

Almost all services provided to consumers, including banking, health records and social media, stores sensitive personal information in a password pro-

tected account. However, strong passwords are often easy to forget and therefore, people usually opt for weaker passwords or use the same one for multiple accounts. This makes them more susceptible to security threats.

2.1.2 Individual Product Use Cases

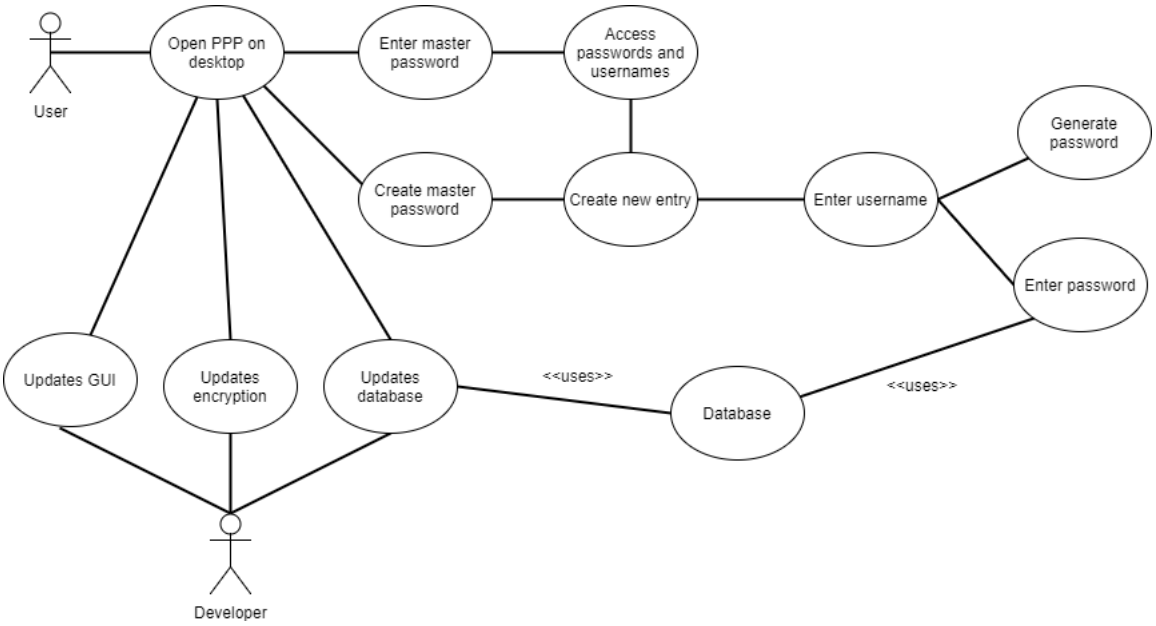

Figure 1: Use Case Diagram

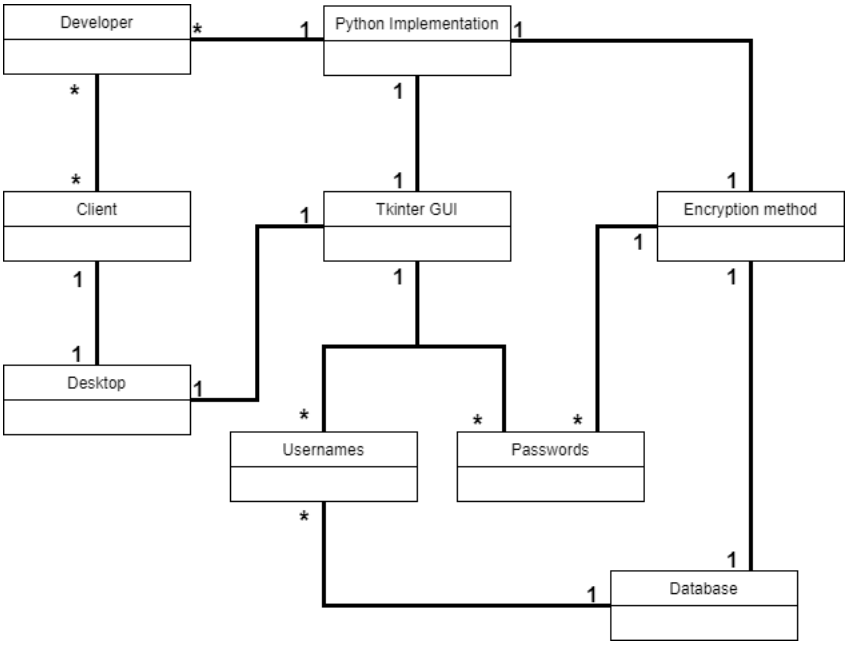

Figure 2: Business Data Model

2.2 Functional Requirements

Table 3: Functional Requirements

Name	Description	Rationale	Fit Criterion
FR1	The executaable Python code will create a user interface window	To allow user to use the application	Run application
FR2	Upon execution, the program will have a connection to a local database	To allow the program to store usernames and encrypted passwords	Create a new entry and check if it shows up on DB
FR3	The user must be able to create a master password	To ensure that only the user can access the data	When opening the app for the first time, create master password and use it
FR4	The user must be able to enter a master password	To access and create passwords and usernames	Enter correct PW and incorrect PW
FR5	The new user must be able to add a new entry	To allow the user to store a new username and password	Create new entry, close app and verify it exists in DB
FR6	The user should be able to generate a random password	To allow for a stronger password	Generate multiple PWs and verify randomness
FR7	The user interface must have a link to the user manual	To aid the user in using the software	Manual link should take user to manual
FR8	After one minute of inactivity, the application should go to the home screen	To protect the users data	Keep app running for 1 minute
FR9	The application should have buttons to directly copy username and password	To allow user to easily input long usernames or passwords	Copy existing password and verify that it pastes onto various text inputters
FR10	The user should be able to change their master password	To allow the user to regularly update PW	Update PW from Settings and verify that new PW works and old one does not

3 Non-functional Requirements

3.1 Look and Feel Requirements

- Requirement Number: NFR1
  The product shall have separate sections to hold the users account authorizations and the creation of new user account usernames and passwords.
  Rationale: The Main functions of the product should be displayed to follow NFR6.
  Fit Criterion: Users are able to access the main functions immediately.
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR2
  The product shall have separate sections to hold the users account authorizations and the creation of new user account usernames and passwords.
  Rationale: The Main functions of the product should be displayed to follow NFR6.
  Fit Criterion: Users are able to access the main functions immediately.
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR3
  The product shall have separate sections to hold the users account authorizations and the creation of new user account usernames and passwords.
  Rationale: The Main functions of the product should be displayed to follow NFR6.
  Fit Criterion: Users are able to access the main functions immediately.
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR4
  The product will have a minimalistic aesthetic that is easy for users to find their way around the application.
  Rationale: User should be able to understand how the product performs.

Fit Criterion: There should be no more than 5 active functions that the user can see and perform at once.
Priority: MED
History: Created October 6, 2017

- Requirement Number: NFR5
  The Google Material Style of the Product will create a modern and functional feel for the product.
  Rationale: The user must want to use this app.
  Fit Criterion: Googles Material Design Constraints will be followed.
  Priority: LOW
  History: Created October 6, 2017

3.2 Usability and Humanity Requirements

- Requirement Number: NFR6
  The product shall be intuitive and easy to understand and use.
  Rationale: Users must be able to use this product.
  Fit Criterion: Majority of Users are able to store a password onto the program
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR7
  The product shall allow for the user to be able to categorize their passwords based on the service that it is used for.
  Rationale: Intended users have multiple accounts that may span many different parts of their lives
  Fit Criterion: Users create sections in the application
  Priority:LOW
  History: Created October 6, 2017

- Requirement Number: NFR8
  The user shall be able to navigate their computer.
  Rationale: Intended users are expected to understand how to copy and paste simple text.
  Fit Criterion: Intended users are able to use the application that meets application requirements.

Priority: LOW
History: Created October 6, 2017

- Requirement Number: NFR9
  The product shall use symbols and words that are naturally understandable by the user community.
  Rationale: The user must easily understand how to use the program as per NFR6.
  Fit Criterion: Majority of users who use the program are able to save their username and password and generate them too.
  Priority: LOW
  History: Created October 6, 2017

- Requirement Number: NFR10
  The product shall hide the details of its construction to the user.
  Rationale: To follow NFR6, the product shall reveal more functions as users execute more processes.
  Fit Criterion: The options for processes that have prerequisites are not shown on the UI until they are fulfilled
  Priority: HIGH
  History: Created October 6, 2017

3.3 Performance Requirements

- Requirement Number: NFR11
  The product shall respond to user input within PROCESSING-TIME when all requirements are fulfilled
  Rationale: Latency disrupts User Experience
  Fit Criterion: The requirements are fulfilled under PROCESSING-TIME for majority of users.
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR12
  The application shall be responsive and when not all requirements for a process are met, it will display a notification or error message within ERROR-TIME.
  Rationale: Latency disrupts the user experience of the application and renders compromises its ease of use.

Fit Criterion: The notification or error message is displayed in under ERROR-TIME for USER-TEST-PERCENTAGE of users.
Priority: HIGH
History: Created October 6, 2017

3.4 Operational and Environmental Requirements

- Requirement Number: NFR13
  The Program shall run on Windows, OSX and Linux OS
  Rationale: Users of our product will have devices on Windows, OSX and Linux.
  Fit Criterion: The product will be approved by the testing groups of each OS and coded in a cross platform language.
  Priority: LOW
  History: Created October 6, 2017

3.5 Maintainability and Support Requirements

- Requirement Number: NFR14
  The product will be available as open source.
  Rationale: As all of the code will be available publicly, it will be easy to update and maintain.
  Fit Criterion: Project will be uploaded on GitLab.
  Priority: LOW
  History: Created October 6, 2017

3.6 Security Requirements

- Requirement Number: NFR15
  The product shall only store usernames and encrypted passwords.
  Rationale: Unencrypted passwords will not be stored, all passwords will be encrypted when added to database.
  Fit Criterion: The encrypted password is displayed to the user
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR16
  The product shall be able to change their master password only if they

are already logged into their accounts.
Rationale: There will not be an email associated with the account to facilitate a reset password option on the log in page.
Fit Criterion: The change password option is only shown when the user is logged in already.
Priority: HIGH
History: Created October 6, 2017

- Requirement Number: NFR17
  The product shall contain security measures in order to prevent the data from being accessed directly from the users machine by people other than the user.
  Rationale: The user may unknowingly leave their computer unattended for a long period of time.
  Fit Criterion: The application logs out after a certain duration of inactivity.
  Priority: HIGH
  History: Created October 6, 2017

- Requirement Number: NFR18
  The product shall not distribute the users personal information to third parties.
  Rationale: The purpose of the software is to secure the clients personal information.
  Fit Criterion: The personal information will not be distributed.
  Priority: HIGH
  History: Created October 6, 2017

3.7 Cultural Requirements

There are none applicable to this project

3.8 Legal Requirements

There are none applicable to this, as all referenced software is open source on github and there are no plans of liscencing the project.

3.9 Health and Safety Requirements

- Requirement Number: NFR18
  This application shall not contain any imagery or animation that can be harming to any human's health
  Rationale: User satisfaction will be greatly decreased if the product may pose a threat to their health in any way.
  Fit Criterion: Application does not contain potential visual health risks.
  Priority: HIGH
  History: Created October 6, 2017

4 Project Issues

4.1 Open Issues

Actions to be taken in the cases in which the user has lost access to their passwords or has lost all of their information due to hardware problems have not yet been explored. Also, the product will be implemented on a Windows operating system as that is the operating system used by all team members. It is intended that the software will work on Linux and Unix operating systems as well so but testing on those environments has not been discussed.

4.2 Off-the-Shelf Solutions

Table 4: Off-the-Shelf Solutions

Existing Product	Key Features
Padlock (template software) MaKleSoft (2017)	Template open source software upon which the product will be based, Chrome and mobile application available
Encryptr SpiderOak (2017)	Cloud-based password manager
KeePass dreichl (2003-2017)	Password database is locked with a master key or key file, encrypted with AES and Twofish
RatticDB tildaslash (2015)	Intended for tracking shared passwords among a group of employees, no application level encryption

4.3 New Problems

If users solely on this product to store all of their passwords, there is potential for them to lose all of their saved data due to a hashing issue, hardware issue or in the theft or loss of their computer. The product requires that the user to enter their current master password in order to change it which could cause issues in the case that they forget it.

4.4 Tasks

Deliverables as outlined by SFWRENG 3XA3 courseware:

Table 5: Deliverables

Deilverable	Deadline
Requirements Document (Revision 0)	October 6, 2017
Proof of Concept Demonstration	October 16, 2017 (week)
Test Plan (Revision 0)	October 27, 2017
Design and Document (Revision 0)	November 10, 2017
Demonstration (Revision 0)	November 13, 2017 (week)
Final Demonstration (Revision 1)	November 27, 2017 (week)
Final Documentation	December 6, 2017

Implementation Breakdown:

Table 6: Implementation Breakdown

Task	Intended Completion Date
System Design	October 12, 2017
User Interface	October 18, 2017
Prototype 1	October 18, 2017
Test 1	October 26, 2017
Test 1 Fixes	October 30, 2017
Encryption	November 7, 2017
Prototype 2	November 13, 2017
Test Plan Revision (Test 2)	November 16, 2017
Test 2 Fixes	November 23, 2017
User Guide	December 1, 2017

The team will be implementing and testing non-functional requirements based on their priority.

4.5 Migration to the New Product

The final product will be run and installed via an executable Python file. As MariaDB will be used to store the usernames and passwords, the database

will have to be converted before the product is released. It also needs to be tested on all intended operating systems.

4.6 Risks

Security is the main concern for this project as the team is not explicitly testing the strength of the encryption method under the assumption that the cryptography library is reliable. The probability of this becoming a problem is unlikely as the product will be offline and will not be a target of many attacks. If the problem does occur, the team will look into a more complex encryption method.

4.7 Costs

Not applicable in the case that all required dependencies are open-source or free.

4.8 User Documentation and Training

There will be a concise and useful manual that provides instructions on how to use the product. To make the product easier to use, the team will be testing for the GUIs user-friendliness and receiving feedback from others.

4.9 Waiting Room

Ideally, there would be a backup data option for the user to use as a precaution in the case of hardware failure and loss of data.
To improve the user interface, it would be nice to provide support for multiple languages (other than only English).
Passwords can be organized by service in the database, so based on that there can be different levels of security provided. For example, bank passwords and library account passwords dont need the same level of security.

4.10 Ideas for Solutions

Instead of storing passwords, another solution is 2 factor authentication (2FA). This eliminates the need to download a password management soft-

ware and instead the user can do a phone verification after typing their password or a fingerprint scanner for mobile devices.

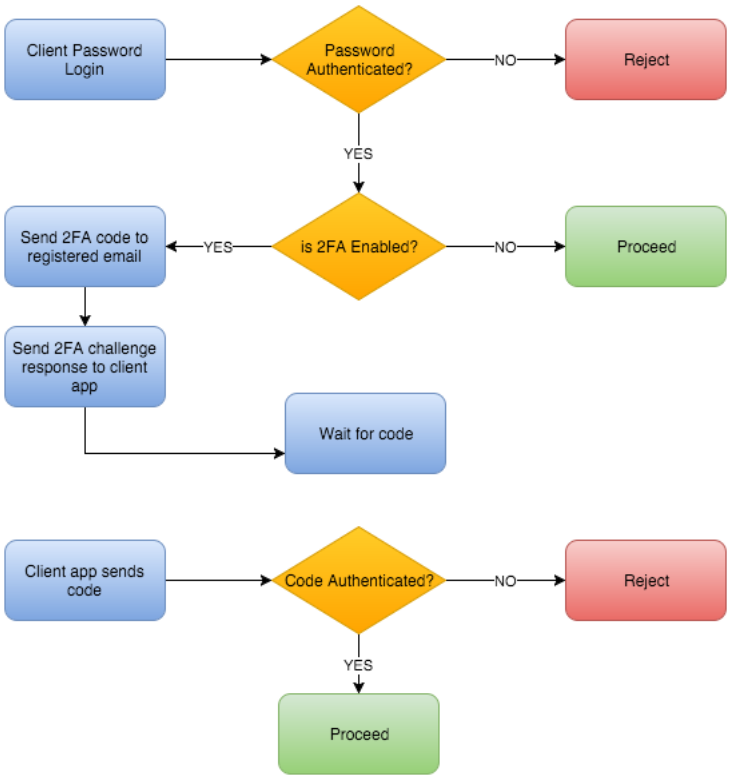

Figure 3: 2 Factor Authentication FileCloud (2016)

References

dreichl. Keepass password safe, 2003-2017. URL https://keepass.info/.

FileCloud. Two factor authentication, 2016. URL https://www.getfilecloud.com/supportdocs/pages/viewpreviousversions.action?pageId=10191111.

MaKleSoft. padlock, 2017. URL https://github.com/MaKleSoft/padlock.

James Robertson and Suzanne Robertson. *Volere Requirements Specification Template*. Atlantic Systems Guild Limited, 16 edition, 2012.

SpiderOak. Encryptr, 2017. URL https://github.com/SpiderOak/Encryptr.

tildaslash. Ratticweb, 2015. URL https://github.com/tildaslash/RatticWeb/wiki/RatticDB-and-Encryption.