SE 3XA3: Test Plan
PasswordProtectionProgram

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

October 28, 2017

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
2017-10-27	0.0	Creation

1 General Information

1.1 Purpose

This document is a rigorous guideline that encompasses the testing, validation and verification processes that the software, PasswordProectionProgram, shall undergo.

1.2 Scope

This test plan will be used as a outline for unit testing the encryption of user information, the GUI and the database used to store the user data. This is to facilitate the identification of software bugs in each specific section and to allow the testing to be done by each group member. These components will also be integration tested to verify that the system as a whole functions as intended.

1.3 Acronyms, Abbreviations, and Symbols

Refer to Table 2 and Table 3

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
GUI	Graphical User Interface
OS	Operating System
POC	Proof of Concept, used for naming tests for proof of concept

1.4 Overview of Document

The functional and nonfunctional requirements for which tests are determined in this document refer to a more specific version of those requirements mentioned in the Software Requirements Specification Document (SRS-Rev0). The names for the functional and non-functional test cases are prefixed with FR for functional and NFR for non-functional. The middle abbreviation for

Table 3: **Table of Definitions**

Term	**Definition**
Home Screen	Screen that application first opens up to, user can enter master password here to sign in to password manager
Password Manager Screen	Screen in which user can input and view accounts, usernames and passwords.
Setting Screen	Screen in which master password can be changed
PROCESSING-TIME	Timeframe within which system must visibly respond to user input, 2 seconds.
ERROR-TIME	Timeframe within which system must visibly respond to incomplete/incorrect user input, 2 seconds.

the test cases refer to the category that they are listed under and then they are suffixed with a numerical value.

2 Plan

2.1 Software Description

The PasswordProtectionProgram is a password manager that uses a Python encryption library to keep user passwords safe and holds them in a local database. The implementation for this application is in Python.

2.2 Test Team

The members of the team, Suhavi Sandhu, Shabana Dhayananth and Joseph Lu, are responsible for all the procedures of the validation process including writing and executing tests.

2.3 Testing Tools

The only tool required for testing is PyUnit, which will be used for unit testing and integration testing.

2.4 Testing Schedule

Refer to Table 4

Table 4: **Testing Schedule**

Date	**Task**	**Team Member**
2017-10-30	Master Password	Suhavi Sandhu
2017-10-30	Adding Entries	Shabana Dhayananth
2017-10-30	Navigation	Joseph Lu
2017-10-04	Encryption	Suhavi Sandhu
2017-11-04	Database	Shabana Dhayananth
2017-11-04	Look and Feel	Joseph Lu
2017-11-07	Personalization	Suhavi Sandhu
2017-11-07	Understandibility and Politeness	Shabana Dhayananth
2017-11-07	Performance	Joseph Lu
2017-11-10	Operational and Environment Requirement	Suhavi Sandhu
2017-11-10	Security	Joseph Lu

2.5 Types of Tests

Refer to Table 5

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Input

3.1.1.1 Master Password

1. FR-MP-1

   Type: Functional, Dynamic, Manual

Table 5: **Types of Tests**

Test Type	**Where/when is it applied**
Structural	Will be used on encryption method, database insertion and deletion
Functional	A type of Black box test to test functional requirements
Unit	For each function in the program. Mostly Functional
System	For testing whole system after integrating database, GUI and encryption
Error Checking	Erroneous password input and missing parameter. Will be conducted near the end
Black box	Testing functionality of product, used primarily for testing existing implementation
White box	Tests internal structures, for testing database
Beta	User testing used for testing non-functional requirements

Initial State: Home screen of application, when master password has not been initialized

Input: Password with at least 8 characters, at least one capital and lowercase letter and at least 1 digit.

Output: The account is created and the user is brought to the password management page.

How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. Once the password is submitted and satisfies requirements, the user will be introduced to a new page that is a result of changing user interface frames.

2. FR-MP-2

   Type: Functional, Dynamic, Manual

   Initial State: Home screen of application, when master password has not been initialized

Input: Password that has more than 8 characters, has lower and uppercase but no numbers

Output: The system does not create account and instead prompts with Password must include at least 1 number

How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. If the password does not satisfy requirements, it outputs the appropriate response to what the password is missing.

3. FR-MP-3

   Type: Functional, Dynamic, Manual

   Initial State: Home screen of application, when master password has not been initialized

   Input: Password that is not at least 8 characters and does not have at least one capital and lowercase letter and at least 1 digit.

   Output: The system does not create account and instead prompts with Password must be at least 8 characters

   How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. If the password does not satisfy requirements, it outputs the appropriate response to what the password is missing.

4. FR-MP-4

   Type: Functional, Dynamic, Manual

   Initial State: Home screen of application, when master password has not been initialized

   Input: Password has numbers and at least 8 characters but no uppercase letters

   Output: The system does not create account and instead prompts with Password must have upper and lowercase letters

   How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. If the password does not satisfy requirements, it outputs the appropriate response to what the password is missing.

5. FR-MP-5

   Type: Functional, Dynamic, Manual

   Initial State: Home screen of application, when master password has not been initialized

   Input: Empty string

   Output: The system does not create account and instead prompts with Password must not be empty!

   How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. If the password does not satisfy requirements, it outputs the appropriate response to what the password is missing.

6. FR-MP-6

   Type: Functional, Dynamic, Manual

   Initial State: Master password has been initialized and the user is at the Settings page and wants to change the master password

   Input: User enters a new master password with at least 8 characters, numbers, upper and lowercase letters and submits it

   Output: The system lets the user proceed. The user should be able to access the application with the new updated master password

   How test will be performed: The function that checks that the password satisfies all requirements will be called when the user submits the password. If the password satisfies requirements, a function updates the master password in the database.

7. FR-MP-7

   Type: Functional, Dynamic, Manual

   Initial State: Master password has been initialized and the user opens application

Input: User enters correct password

Output: The system lets the user proceed. The user should be taken to the password management page where one can add new accounts or view saved account information

How test will be performed: A function will be called to compare the users input and the decrypted master password stored in the database. If the password is the same, the window is updated.

8. FR-MP-8

   Type: Functional, Dynamic, Manual

   Initial State: Master password has been initialized and the user opens application

   Input: User enters incorrect password

   Output: The system does not let the user proceed and

   How test will be performed: A function will be called to compare the users input and the decrypted master password stored in the database. If the password is the same, the window is updated.

3.1.1.2 Adding Entries

9. FR-AE-1

   Type: Functional, Dynamic, Manual

   Initial State: User is logged in with master password and wishes to add an entry to the password manager by clicking with Add option

   Input: For Type, enter what the password is for (i.e Facebook account), for Username enter a valid username and for password enter a valid password

   Output: The system adds the entry to the database, upon encrypting it, and the user should be able to see the entry in their list of saved account information.

   How test will be performed: Given that a password is provided, the function that encrypts the username and password will be called, thereby encrypting it and the data will be stored in the database. To make it appear, a new label will be added to the window that shows what the user has just added (with the password in asterisks).

10. FR-AE-2

    Type: Functional, Dynamic, Manual

    Initial State: User is logged in with master password and wishes to add an entry to the password manager by clicking with Add option

    Input: For Type, Username and Password, leave fields empty and click Add

    Output: The system does not let user proceed and prompts for a password

    How test will be performed: The system uses a function to check that the password is not null and is it is, displays a label prompting user to enter a password

11. FR-AE-3

    Type: Functional, Dynamic, Manual

    Initial State: User is logged in with master password and wishes to add an entry to the password manager by clicking with Add option

    Input: Leave Type and User blank, for password enter a valid password

    Output: The system adds the entry to the database, upon encrypting it, and the user should be able to see the entry in their list of saved account information.

    How test will be performed: Given that a password is provided, the function that encrypts the username and password will be called, thereby encrypting it and the data will be stored in the database. To make it appear, a new label will be added to the window that shows what the user has just added (with the password in asterisks).

3.1.2 Navigation

12. FR-N-1

    Type: Functional, Dynamic, Manual

    Initial State: Password Management screen

    Input: User clicks on user manual

    Output: User Manual opens and is displayed on the screen

    How test will be performed: The application will be opened and the function to make the user manual appear will be called

13. FR-N-2

    Type: Functional, Dynamic, Manual

    Initial State: User manual screen

    Input: User clicks clicks back button

    Output: Password management screen opens and is displayed on the screen

    How test will be performed: The application will be opened and the function to make the password management screen appear will be called

14. FR-N-3

    Type: Functional, Dynamic, Manual

    Initial State: User is at password management screen and nothing has been added

    Input: User clicks Add button

    Output: 3 fields are displayed: Type, Username and Password

    How test will be performed: The application will update the interface to show 3 new labels and text fields when the button is clicked

15. FR-N-4

Type: Functional, Dynamic, Manual

Initial State: User is at password management screen and an entry has been added

Input: User clicks Add button

Output: 3 fields are displayed under the existing entry: Type, Username and Password

How test will be performed: The application will update the interface to show 3 new labels and text fields when the button is clicked, keeping the previous entries

16. FR-N-5

    Type: Functional, Dynamic, Manual

    Initial State: User has added an entry and wishes to copy the password

    Input: User clicks on the Copy button beside the password in asterisks

    Output: The password is copied and can be pasted into any text inputter

    How test will be performed: The application decrypts the password and copies it for the user using the copy() function

17. FR-N-6

    Type: Functional, Dynamic, Manual

    Initial State: User is on password management screen and clicks Add

    Input: User clicks on Generate button for password

    Output: A random password is generated

    How test will be performed: The function to generate a random password is called and the password appears in the text field

18. FR-N-7

    Type: Functional, Dynamic, Manual

    Initial State: User is on password management screen and clicks Add

Input: User clicks on Generate button twice

Output: A random password is generated and then replaced with the new

How test will be performed: The function to generate a random password is called and the password appears in the text field. The second time, this text is replaced with the new random password

3.1.3 Encryption

19. FR-E-1

    Type: Unit, Dynamic, Automated

    Initial State: Encryption function is called

    Input: Any string

    Output: String gets encrypted

    How test will be performed: The encrypted string will be printed to check if it is encrypted

20. FR-E-2

    Type: Unit, Dynamic, Automated

    Initial State: Decryption function is called

    Input: An already encrypted string

    Output: String gets decrypted

    How test will be performed: The decrypted string will be printed to see if it matches the original string that was encrypted

3.1.4 Database

21. FR-DB-1

    Type: Functional, Dynamic, Manual

Initial State: Empty database

Input: A master password is created

Output: The master password appears in the table as its encrypted form with all other fields NULL

How test will be performed: The database table that stores records will be checked to see if the first entry is a master password with ID of 1

22. FR-DB-2

    Type: Functional, Dynamic, Manual

    Initial State: Master password has already been created

    Input: User modifies master password

    Output: The updated master password appears in the table as its encrypted form with all other fields NULL

    How test will be performed: The database table that stores records will be updated and then verified to see if the first entry of the database has updated

23. FR-DB-3

    Type: Functional, Dynamic, Manual

    Initial State: User is at password management screen, master password is initialized already

    Input: User adds an entry of type, username and password

    Output: The entry is added to the database with the type in the Type field, username encrypted in the Username field, encrypted password in Password field.

    How test will be performed: The database table that stores records will have a new entry inserted and will be verified to see if a new entry has been added

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel Requirements

1. NFR-LFR-1

   Type: Structural, Dynamic, Manual

   Initial State: Main screen, before logging on

   Input: Click enter or button to access another screen (ex. settings)

   Output: Next screen containing different functions appears

   How test will be performed: Test navigation between screens and make sure that functions on each screen are best placed in that grouping using beta testing.

2. NFR-LFR-2

   Type: Structural, Dynamic, Manual

   Initial State: Password manager screen, after the user has logged in using their master password

   Input: The required fields are account type (for example, email, bank, etc.) and the username and password associated with the account.

   Output: If one field is not filled in by the user (empty) a pop-up notification will appear to inform the user that they are missing a required field

   How test will be performed: Different combinations of each of the three fields will be omitted intentionally to check if the notification appears in each of those cases.

3. NFR-LFR-3

   Type: Structural, Dynamic, Manual, Unit

   Initial State3 Main screen, upon first entry when the user is required to choose a master password for their account

   Input: A strong password that contains at least 8 characters consisting of uppercase, lowercase, and numbers

Output: After the enter button is pressed, messages will appear under the input box denoting the missing password criteria.

How test will be performed: A variety of passwords missing a combination of the strong password criteria will be input in order to check if the messages are displayed correctly.

3.2.2 Personalization Requirements

4. NFR-PSR-1

   Type: Structural, Dynamic, Manual

   Initial State: Password manager screen, after the user has logged in using their master password

   Input: The user specifies the account/service type which they wish to add a username and password for

   Output: After all three fields (account type, username, password) are input, the account type gets stored into the database.

   How test will be performed: The account type will be input using normal user data entry so the success of the test case can be verified by ensuring that the input data has been added to the database.

3.2.3 Understandability and Politeness Requirements

5. NFR-UPR-1

   Type: Manual

   Initial State: Before the application is released

   Input: Online user survey with potential symbols that will be used in the product

   Output: User feedback on survey

   How test will be performed: The user feedback on what the symbols are assumed to represent will allow for the most understandable symbols to be used in the product design.

3.2.4 Performance

6. NFR-PER-1

   Type: System, Dynamic, Automated, Unit

   Initial State: All required fields of input are filled but the enter button is not yet pressed

   Input: User presses enter button

   Output: In test report, time elapsed between button press and output display is given.

   How test will be performed: An in-code time will be used to ensure that the response time to all user inputs is within PROCESSING-TIME (2 seconds).

7. NFR-PER-2

   Type: System, Dynamic, Automated, Error checking

   Initial State: Not all required fields of input are filled but the enter button is not yet pressed

   Input: User presses enter button

   Output: In test report, time elapsed between button press and error message output display is given.

   How test will be performed: An in-code time will be used to ensure that the response time to all user input error is within ERROR-TIME (2 seconds).

3.2.5 Operational and Environmental Requirements

8. NFR-OER-1

   Type: System, Dynamic

   Initial State: Application is downloaded onto OS but not yet executed.

   Input: Execution of system

Output: No unexpected behaviour on OS that is not the OS that the product was developed on.

How test will be performed: All functional test cases will be checked on OSX and Linux OS.

3.2.6 Security Requirements

9. NFR-SR-1

   Type: Structural, Dynamic, Automated, Unit

   Initial State: Password manager screen, after the user has logged in using their master password

   Input: The user specifies the account/service type, a username and a password

   Output: Account type, username and encrypted password are stored in database

   How test will be performed: Database will be accessed directly to view format of data that was added.

10. NFR-SR-2

    Type: Structural, Dynamic, Automated, Unit

    Initial State: Settings screen, after the user is logged in using current master password

    Input: User enters new master password

    Output: Password is checked against strength criteria, and if those are met, new password is saved and old password deleted

    How test will be performed: Change master password option will only be displayed if the user is logged in and on the settings page.

11. NFR-SR-3

    Type: Structural, Dynamic, Automated, Unit

    Initial State: User is using application

    Input: User stops actively using application for a period of time

Output: In test report, time elapsed between idle state and automatic logout is given.

How test will be performed: An in-code/test timer will be used to measure if the application logs the user out after it is idle for over one minute.

3.3 Traceability Table Between Test Cases and Requirements

Refer to Table 6 and Table 7

4 Tests for Proof of Concept

Testing for Proof of Concept will be unit and structural test what will be used for a Proof of Concept demo. These tests are to test the functionality of the program. The areas that the proof of concept testing will focus on are User Interface, Encryption and Database.

4.1 Master Password

1. POC-1

   Type: Functional, Dynamic, Manual

   Initial State: GUI is run for the first time

   Input: A valid password is entered under Master Password

   Output: User is brought to the password management screen

   How test will be performed: For the proof of concept, the master password will not be saved in the database or encrypted as those are independent during the POC. Instead, a function that checks password logic will allow the screen to update to the next screen if password criteria is satisfied

2. POC-2

   Type: Functional, Dynamic, Manual

Table 6: **Traceability table for Functional Requirements**

Req ID	**Description**	**Priority**	**Test Cases**
FR1	The executable Python code will create a user interface window	High	FR-MP-1
FR2	Upon execution, the program will have a connection to a local database	High	FR-DB-1, FR-DB-2, FR-DB-3
FR3	The user must be able create a master password	High	FR-MP-1, FR-MP-2, FR-MP-3, FR-MP-4-, FR-MP-5
FR4	The user must be able to enter a master password	High	FR-MP-7, FR-MP-8
FR5	The user must be able to add a new entry to the manager	High	FR-AE-1, FR-AE2, FR-AE-3, FR-N-3, FR-N-4
FR6	The user should be able to generate a random password	High	FR-N-6, FR-N-7
FR7	The user interface must have a link to the user manual	High	FR-N-1, FR-N-2
FR8	When user wants to copy or view a password, the software should decrypt it	High	FR-N-5, FR-E-2
FR9	The application should have buttons to directly copy username and password	High	FR-N-5
FR10	The user should be able to change their master password	High	FR-MP-6
FR11	When a user adds a new entry, the software should encrypt the input	High	FR-E-1

Table 7: **Traceability table for Non-functional Requirements**

Req ID	**Description**	**Priority**	**Test Cases**
NFR1	Different screens for different functions	High	NFR-LFR-1, beta testing
NFR2	Pop-up notification when requirements are missing	High	NFR-LFR-2
NFR3	Clearly display master password criteria before processing	High	NFR-LFR-3
NFR4	The product will have a minimalistic aesthetic that is easy for users to find their way around the application	Med	Beta testing
NFR5	Create a modern and functional feel for the product. Rationale: The user must want to use this app	Low	Beta testing
NFR6	The product shall be intuitive and easy to understand and use.	High	Beta testing
NFR7	The product shall allow for the user to be able to categorize their passwords based on the service	Low	NFR-PSR-1
NFR8	The user shall be able to navigate their computer.	Low	Beta testing

NFR9	The product shall use symbols and words that are naturally understandable	Low	NFR-UPR-1
NFR10	The product shall hide the details of its construction to the user.	High	Beta testing
NFR11	The product shall respond to user input within PROCESSING-TIME	High	NFR-PER-1
NFR12	The application shall be responsive and when not all requirements for a process are met, error message within ERROR-TIME	High	NFR-PER-2
NFR13	The program shall run on Windows, OSX and Linux OS	Low	NFR-OER-1
NFR14	The product will be available as open source.	Low	N/A
NFR15	The product shall only store account types, usernames and encrypted passwords.	High	NFR-SR-1
NFR16	Master password cannot be modified unless user is already logged in	Hight	NFR-SR-2
NFR17	The product shall contain security measures in order to prevent the data from being accessed directly from the users machine	High	NFR-SR-3
NFR18	The product shall not distribute the users personal information to third parties	High	N/A

Initial State: GUI is run for the second time after master password is initialized

Input: The correct password is entered under Master Password

Output: User is brought to password management screen

How test will be performed: For the proof of concept, the master password from before will be saved and upon execution for the second time and forwards, it will proceed only if the user types in the correct password

3. POC-3

   Type: Functional, Dynamic, Manual

   Initial State: Password management screen

   Input: Valid account information is entered for type, username and password

   Output: Entry appears as label on screen

   How test will be performed: For the proof of concept, the input is checked for criteria satisfaction (is the password not null), if these requirements are satisfied, the entry is displayed on the GUI frame

4. POC-4

   Type: Unit, Dynamic, Automated

   Initial State: Encryption function is called

   Input: Any string

   Output: String gets encrypted

   How test will be performed: The encrypted string will be printed to check if it is encrypted

5. POC-5

   Type: Unit, Dynamic, Automated

   Initial State: Decryption function is called

Input: An already encrypted string

Output: String gets decrypted

How test will be performed: The decrypted string will be printed to see if it matches the original string that was encrypted

6. POC-6

   Type: Functional, Dynamic, Manual

   Initial State: Connection to database has not been established

   Input: Connect to database (Database.Connect())

   Output: Can print contents through command line

   How test will be performed: For the proof of concept, the database is independent of the GUI and encryption method. For this reason, after the connection is established, it can be verified by printing the table contents through the command line

7. POC-7

   Type: Functional, Dynamic, Manual

   Initial State: Connection to database has been established

   Input: Insert into database

   Output: New content shows up when printing through command line

   How test will be performed: For the proof of concept, the database is independent of the GUI and encryption method. For this reason, on insertion of data, it can be checked that the data was indeed inserted by printing the table contents through the command line

5 Comparison to Existing Implementation

The existing implementation, Padlock, which our implementation is based on, is very similar in nature in terms of user interface. Therefore, the main areas of testing that can be compared to the existing product are user input and navigation. These tests will be valuable in improving the product and testing parallely to see what our implementation lacks or could do differently.

Relevant test cases that can be used for comparison are:

Functional Tests

User Input

- FR-MP-1 - Creation of Master Password
- FR-MP-5 - Empty string for master password creation
- FR-MP-6 - Changing master password from settings
- FR-MP-7 - Logging in using master password
- FR-MP-8 - Attempting to log in using incorrect master password
- FR-AE-1 - Add an entry for account with type, username and password

Navigation

- FR-N-5 - Copy password from existing entry
- FR-N-6 - Generate random password for account entry

Nonfunctional Tests

Operational and Environmental Requirements

- NFR-OER-1 - Program compatible to Windows, OSX and Linux OS

Personalization Requirements

- NFR-PSR-1 - User can categorize usernames and passwords based on account type

Performance Requirements

- NFR-PER-1 - Output response to user input shall happen in a 2 second timeframe
- NFR-PER-2 - Error response to user input shall happen in a 2 second timeframe

Security

- NFR-SR-3 - Logs user out after one minute of inactivity for security

6 Unit Testing Plan

6.1 Unit testing of internal functions

Unit testing will be executed using PyUnit and all functions of the application will be tested. In functional requirements there are few unit tests because the tests are heavily focused on user interface. However, to test that internal functions work as expected, the following tests will be executed.

- `checkPassword(password)` - checks if the password satisfies criteria
- `insert(id, type, username, password)` - inserts id of entry, type, username and encrypted password
- `updateMasterPassword()` - updates master password in database if it is modified
- `username(id)` - returns username from database
- `password(id)` - returns password from database, which is later decrypted
- `copy(id)` - gets password at specific id, decrypts it and returns
- `generateRand()` - generates a random password
- `generKey()` - generates key for encryption
- `cryptEncode(key f, String password)` - encrypts input password using Fernet key
- `decrypt(String encryptedPassword)` - decrypts encrypted password using key

6.2 Unit testing of output files

Below is a description of the output files that will be needed for unit testing and how the functions that are being unit tested will use the output files. Not all functions require an output file for unit testing. Coverage metrics are also discussed.

`passwords.txt`

- `passwords.txt` will have a list of passwords that may or may not satisfy password criteria
- At the end of the file there will be a Boolean array that tells you if the passwords satisfy the criteria or not
- Used for checkPassword() to test if passwords are being evaluated correctly. Coverage metric will be comparison of unit test output to the Boolean array at the end of file
- Used for updateMasterPassword() to change master password consecutively.
- Used for generKey(), cryptEncode(key f, String password), decrypt(String encryptedPassword) to encrypt and decrypt passwords

`tuples.csv`

- Will have records to be inserted into the database
- used for insert(id, type, username, password) to insert into database
- used for username(id) and password(id) to have something in database to return
- Used for copy(id) to see if each password entry gets copied

`copies.txt`

- The output file which will be compared to all password entries in tuples.csv
- Used for copy(id) to check if the passwords in tuples.csv were copied correctly to copies.txt

`generated1.txt` and `generated2.txt`

- Used for generateRand() in which both files will be filled with generated passwords and the coverage metric will be to verify how similar the two are. This will be loosely done by comparing each line in generated1.txt to all lines in generated2.txt. Fewest similarities are ideal