SE 3XA3: Test Report
Password Protection ProgramG

Team 28, Team Name
Suhavi Sandhu (sandhs11)
Shabana Dhayananth (dhayanas)
Joseph Lu (luy89)

December 7, 2017

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
December 6, 2017	0.0	Creation

This document discusses the results from testing the product, PasswordProtectionProgram, any changes that were implemented as a result of testing and the extensiveness of the tests performed.

1 Functional Requirements Evaluation

1.1 User Input

1.1.1 Master Password

These test cases verify that the functions related to the master password, a critical part of the system, behave as intended.

The following test cases assume that a master password has not yet been initialized and test for the creation of the master password.

- **FR-MP-1**: A valid input of at least characters having upper and lowercase as well as numbers should let user proceed.
  Result: Test pass, user is allowed to proceed.
- **FR-MP-2** to **FR-MP-5**: Invalid input for master password should display appropriate error message.
  Results: All tests pass and display appropriate error message.

The following test cases assume that a master password has already been initialized and test for logging in with the master password.

- **FR-MP-6**: Upon entry of an empty string, the system should display an error message.
  Result: This test originally passed, however, due to a bug found during the testing for update entry, this test was omitted.
- **FR-MP-7**, **FR-MP-8**: Depending on whether input is correct or incorrect, system lets user proceed or displays error message.
  Result: Tests pass

1.1.2 Adding Entries

These tests verify that users are able to add new entries to the database, assuming a connection to the database is already established and user is logged into application.

- **FR-AE-1**, **FR-AE-2**, **FR-AE-3**: Originally, the goal of these test cases was to ensure that when adding an entry, a user cannot add an empty password. Upon testing, it became clear that an empty password should be allowed, in the case that the user has not come up with one. Therefore, the test was changed to verify that a Name, rather than a password, should be required.
  Results: All test cases pass, allowing users to add entries as long as name is provided, else error message displayed.

1.2 Navigation

The underlying goal for the product was to make it more userfriendly than the original. In doing so, great care was given to the tests designated under navigation. The following tests verify the functionality of the user manual.

- **FR-N-1**: Link to user manual works.
  Results: A pdf viewer opens up with the user guide.

- **FR-N-2**: At view entry screen, clicking the question mark should take user to instructions page
  Results: Test passes, proving that instructions are readily accessible.

- **FR-N-5**: User should be able to copy a password or username that is in the system database.
  Results: Copy button works as intended, test passes

- **FR-N-6**, **FR-N-7**: test the functionality of the generate button and verify that a string is outputted and can be called random through visual inspection.
  Results: All tests pass, output is random to the eye.

- **FR-N-8**: Tests the delete button, ensuring that all entries can be deleted, except master password.
  Results: Test passes, deleted entries are immediately removed and the database as well as the interface reflects these changes.

- **FR-N-9**: Tests the update button, ensuring that all entries password and usernames can be changed.
  Results: Test passes.
  Further discussion: This test led to the discovery of a bug that results from updating a master password to be an empty string. It is a legal action, however, when the user attempts to log back in with the updated password, he/she is prohibited since an empty string was not allowed. A temporary workaround for this bug is to allow empty strings but in the future, there will be unique constraints for updating passwords (i.e cannot be empty).

1.3 Encryption

- **FR-E-1**, **FR-E-2**: Check that a string being encrypted can ultimately be decrypted to give the original.
  Results: All tests pass.

1.4 Database

These tests have to do with the functionality of the database and ensure that it works independently to the rest of the system. Therefore, it is vital to ensure basic methods such as insert, delete, and update work as intended - something that can be proven with the test cases mentioned below.

- **FR-DB-1**: Database can be created

- **FR-DB-2**: Database has already been created

- **FR-DB-3**, **FR-DB-4**: When multiple entries exist, database should delete them when user deletes and return them when needed:
  Results: All tests pass.

1.5 Password Generation

- **FR-PWG-1**: This test case ensures that the criteria for what the team defined as random is achieved by the method.
  Results: Test passes. It can be proven that the method generates passwords with exactly 8 characters, including uppercase, lowercase and numbers.

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel Requirements

The following requirements check that the application has a standard interface that is aesthetically pleasing. An example test case below describes generally what was evaluated.

- **NFR-LFR-1**: The user should be able to navigate between screens seamlessly through the use of available buttons.
  Results: This test case, like all others in the Look and Feel category, were performed manually by our peers. To ensure that the cases passed, a rating from 1 to 10 is provided from user.
  For all the tests the average scores were reasonable.

2.2 Personalization

- **NFR-PSR-1**: The user should be able to specify the category for which they are adding an entry for.
  Results: This test was performed manually by peers, who gave feedback in the form of their experience with the personalization aspect as well as a number from 1-10 defining how customizable they felt the application was. In this category, the application received a 7 as the option to add categories is available although there is no sorting by category, something we wish to implement in the future.

2.3 Understandability and Politeness Requirements

- **NFR-UPR-1**: To better understand the needs of users, a survey was conducted in the form of informal interviews with peers that may be

interested in using a password manager.
Results: The outcome of the interview gave insight on the topic of intuitive interfaces. We learned that users are more likely to understand from visual cues rather than text-based as they are more universal. Hence, image symbols were used for as many buttons as possible.

2.4 Performance

- **NFR-PER-1**: To verify the performance of the system, processing time of the reset_timer function, a function of the graphical user interface that keeps track of inactivity period, was determined and compared against a PROCESSING_TIME = 1 second
  Result: The test passed, giving a processing time less than PROCESSING_TIME, verifying that the performance is up to par with standards.

- **NFR-PER-2**: To verify that error messages are displayed within ERROR_TIME, the function for checking an invalid password was timed.
  Result: The test passed, giving an error time less than ERROR_TIME, again, proving that performance is sufficient.

2.5 Operational and Environmnetal Requirements

The following requirement checks that the application is able to run with Linux and OSX.

- **NFR-OER-1**: The user should be able to run the application with the same functionality as if they are on Windows.
  Result: All funcationality tests passes on other operating systesms.

2.6 Security

As the application that we developed being a security applicaiton, testing for security is a high priority of the team.

- **NFR-SR-1**: When the user is creating a new password when starting the application for the first time. They must create the password within the Criteria of set, or the database will not take the password.
  Result: The test Passes. When entering insufficiently secure passwords, the user can not keep going.

- **NFR-SR-3**: When the application is left inactive for the allocated time given, the application logs out.
  Result: The test pass.

3 Comparison to Existing Implementation

Apart from the coding language and narrowed scope, the PasswordProtectionProgram was not too different from the original implementation, Padlock. By executing manual tests on both implementations, it can be verified that the PasswordProtectionProgram successfully fulfils a majority of the main functions that were in the original implementation. Below are the results from those tests.

Functional Tests

User Input

- **FR-MP-1** Creation of master password - can be performed successfully on both with ease. Original implementation, however, does not have criteria for master password (it can be a single character if desired)
- **FR-FR-MP-5** Empty string for master password creation - displays error message in both implementations
- **FR-FR-MP-7** Log in with master password - returning users can successfully login with correct master password
- **FR-FR-MP-8** Login with incorrect master password - both implementations display error message.
- **FR-FR-AE-1** Add an entry for account with name, type, username and password - Same categories are present for this and the original implementation however, duplicates are allowed in the original, something we disallowed to give users a better experience

Navigation

- **FR-N-5** Copy password from existing entry - both implementations give users an easy way to access their passwords if they are too complex, by simply copying to clipboard

- **FR-N-6** Generate random password for account entry - both implementations have a generate button to give users the opportunity to have more complex usernames and passwords, though, both implementations have different criteria for generation (8 characters, numbers, etc)

Nonfunctional Tests

Operational and Environmental Requirements

- **NFR-OER-1** Compatibility with Windows, OSX, Linux OS - both implementations should ideally work on all 3 platforms and they did however, we did not get a chance to test the same for our executable

Personalization

- **NFR-PSR-1** User can categorize usernames and passwords based on account type - this test failed for the PasswordProtectionProgram as there is no categorization by account type as of now. It may be implemented in the future so users can sort account data by category, similar to the original implementation

Security

- **NFR-SR-3** Logs user out after 1 minute of inactivity - currently changed to 30 seconds for testing purposes, both implementations timeout after a minute.

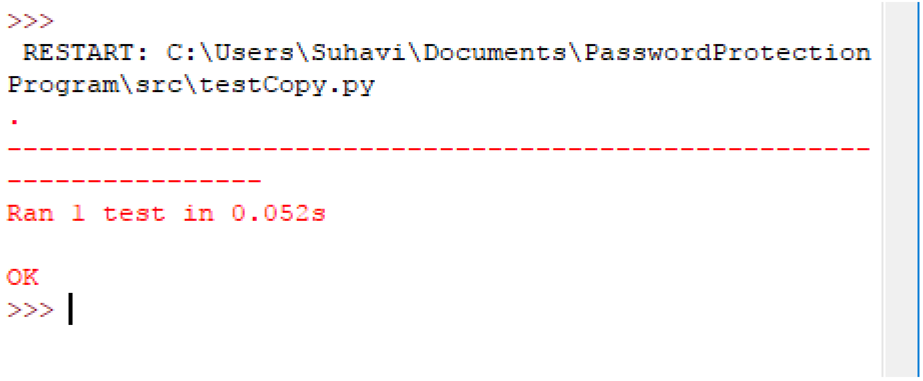
4 Unit Testing and Automated Testing

PyUnit was used to perform unit testing on the system. All of these tests were automated.

Below is description of each test module and a trace of what test cases were tested in each test module.

4.1 testCopy.py

Tests copy.py by checking that the copy function accurately copies what is on the users clipboard



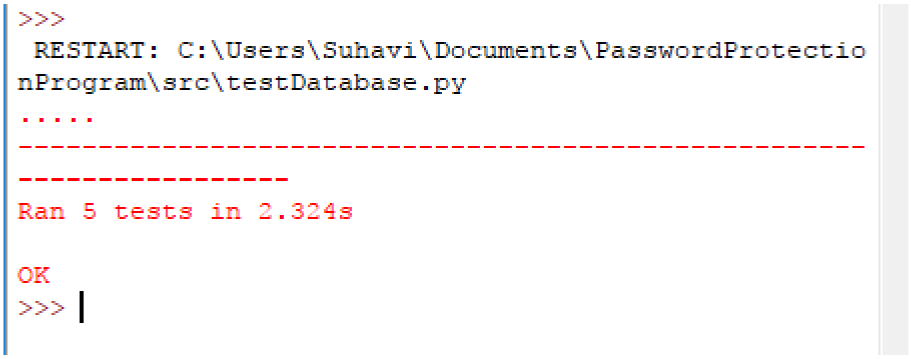
```
>>> 
 RESTART: C:\Users\Suhavi\Documents\PasswordProtection
Program\src\testCopy.py
.
------------------------------------------------------
----------------
Ran 1 test in 0.052s

OK
>>> 
```

4.2 testDatabase.py

Tests database.py by checking that the main functions of the database such as insert, delete and update behave as intended

- Tests **FR-DB-1**, add master password to empty database
- Tests **FR-DB-2**, update entry in populated database
- Tests **FR-DB-3**, delete entry in populated database
- Tests **FR-DB-4**, return all information in populated database



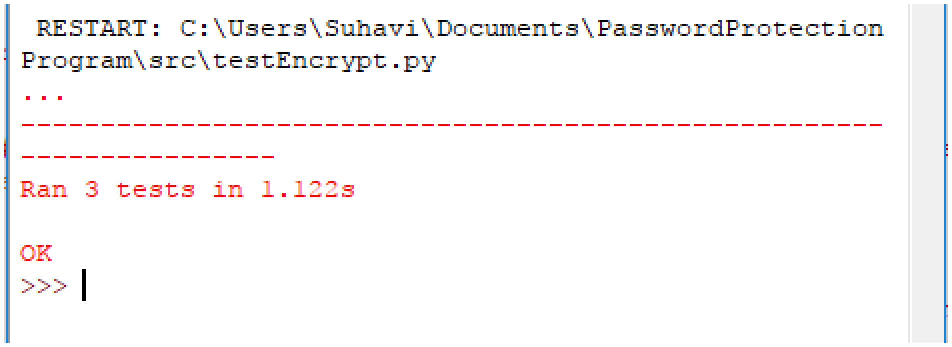
```
>>> 
 RESTART: C:\Users\Suhavi\Documents\PasswordProtectio
nProgram\src\testDatabase.py
.....
-----------------------------------------------------
-----------------
Ran 5 tests in 2.324s

OK
>>> 
```

4.3 testEncrypt.py

Tests Encrypt.py by checking that key generation, encryption and decryption work

- Tests **FR-E-1**, encrypt strings
- Tests **FR-E-2**, decrypt an encrypted string



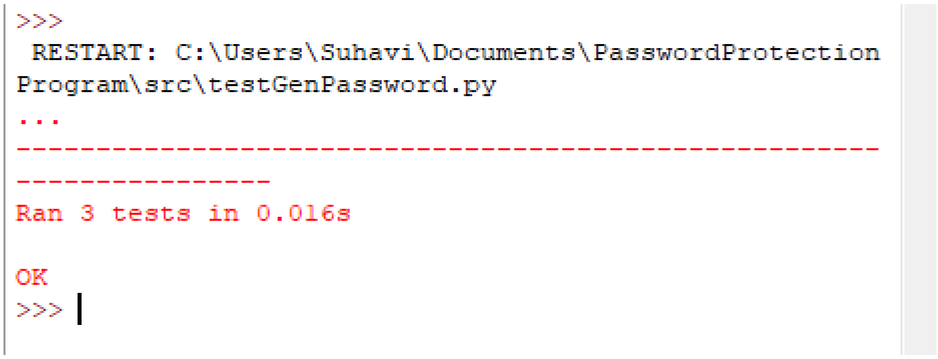
```
 RESTART: C:\Users\Suhavi\Documents\PasswordProtection
Program\src\testEncrypt.py
...
------------------------------------------------------
----------------
Ran 3 tests in 1.122s

OK
>>> 
```

4.4 testGenPassword.py

Tests GenPassword.py by checking that the random strings for username and password meet criteria

- Tests **FR-PWG-1**, criteria for random string generation (8 characters, upper,lowercase, number)



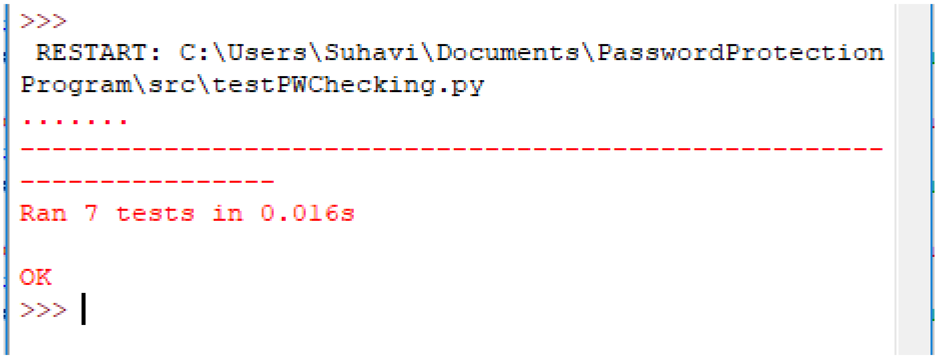
```
>>> 
 RESTART: C:\Users\Suhavi\Documents\PasswordProtection
Program\src\testGenPassword.py
...
------------------------------------------------------
----------------
Ran 3 tests in 0.016s

OK
>>> 
```

4.5 testPWChecking.py

Tests `PWChecking.py` to verify that valid and invalid master password entries are handled correctly

- Tests **FR-MP-1** - create master password, following criteria (allowed)
- Tests **FR-MP-2** - create master password with no numbers (display error message)
- Tests **FR-MP-3** - create master password not at least 8 characters (display error message)
- Tests **FR-MP-4** - create master password with no uppercase (display error message)
- Tests **FR-MP-5** - create master password with empty string (displays error message)
- Tests **FR-MP-6** - later omitted
- Tests **FR-MP-7** - login with correct master password (allowed)
- Tests **FR-MP-8** - login with incorrect master password (displays error message)



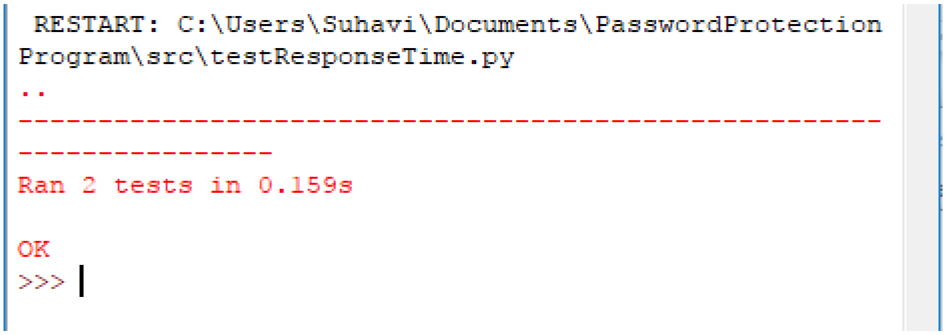
```
>>>
 RESTART: C:\Users\Suhavi\Documents\PasswordProtection
Program\src\testPWChecking.py
.......
------------------------------------------------------
----------------
Ran 7 tests in 0.016s

OK
>>> 
```

4.6 testResponseTime.py

Tests `PPP.py` for its performance by checking PROCESSING_TIME and ERROR_TIME

- Tests **NFR-PER-1** - perform action within PROCESSING_TIME
- Tests **NFR-PER-2** - display error message within ERROR_TIME



```
 RESTART: C:\Users\Suhavi\Documents\PasswordProtection
Program\src\testResponseTime.py
..
------------------------------------------------------
----------------
Ran 2 tests in 0.159s

OK
>>> |
```

5 Changes Due to Testing

There were several changes that were implemented as a result of testing. These changes will be categorized by the main functional requirement that was affected.

1. **FR9** User should be able to change master password:
   While manually testing the update master password function, it was noticed that the master password could be updated to be an empty string, which is contradictory to the creation stage where the master password has to be at least 8 characters. As described earlier in the document, a change was made to the password checking module that accepts an empty string.

2. **FR5** User should be able to add entries to the database
   While testing the adding entry function through integration of the interface and database, when the entries reached beyond the scrollbar, the scrollbar would not update to increase in size. To fix this, there was code added to refresh the application each time an entry is added. This made it less static than what we originally hoped but ensured that all data would be up to date at all times.

Another change that resulted from this test was that duplicate entries should not be allowed. Originally this was not in the plan but upon seeing that multiple entries could be added with the same name, it was evident that duplicates should not be allowed.

3. **FR7** There should be a link to the user manual
   Feedback from those testing the application led to the changes in the instructions page. The font was made bigger and the steps to use the application were numbered to be more user friendly.

4. **NFR-LFR-1** Look and Feel Requirement - User experience
   Some of the feedback that we received regarding user experience was invaluable since the main goal of the application was to increase user-friendliness. Below are the changes that resulted from feedback.
   Making Master Password Creation instructions more clear - explicitly define criteria for a valid master password.
   When user updates an entry, system should show a message like Changes saved! to show that something occurred behind the scenes.

6 Trace to Requirements

7 Trace to Modules

8 Code Coverage Metrics

We believe that the testing that we have conducted on PasswordProtection-Program has covered about 95

As for boundary testing, all inputs into the system have to be string types as they pass through a text field entry. Due to this, all inputs to the database and encryption functions were also strings so there were not many boundary cases to test. The only notable abnormal case was the input of an empty string which we have tested for and handled.

We associate the remaining 5

Table 2: **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, FR-DB-4
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
FR6	The user should be able to generate a random password	High	FR-N-6, FR-N-7, FR-PWG-1
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 3: **Traceability table for Functional Requirements**

Req ID	**Description**	**Priority**	**Test Cases**
FR12	The user will be able to delete any data entries except for the master PW.	High	FR-N-8
FR13	The user will be able to update the username and password	High	FR-N-9

Table 4: **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

NFR9	The product shall use symbols and words that are naturally understandable	Low	NFR-UPR-1
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	High	.
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

Table 5: **Traceability table for Modules**

Mod ID	**Description**	**Test Cases**
M1	Input Module	FR-AE-1, FR-AE-2, FR-AE-3, NFR-LFR-2, NFR-LFR-3, NFR-OER-1, NFR-UPR-1
M2	Password Generation Module	FR-N-6, FR-N-7, FR-PWG-1
M3	Encryption Module	FR-N-5, FR-E-1, FR-E-2, NFR-SR-1, NFR-SR-3
M4	Password Checking Module	FR-MP-1...FR-MP-8
M6	GUI Module	FR-AE-1, FR-AE-2, FR-AE-3, FR-N-1, FR-N-2, FR-N-8, FR-N-9, NFR-LFR-1...NFR-LFR-3, NFR-UPR-1, NFR-PER-1, NFR-PER-2
, M7	Copy Module	FR-N-5
M9	Database Module	FR-DB-1...FR-DB-4, NFR-SR-1
M10	Constants Module	FR-MP-1, NFR-PER-1, NFR-PER-2