Project Part 6: *Testing*

Title: Fireproof

Vision: One Password to Rule Them All

Who: Annie Kelly, Tori Augustine, Kara James, Tyler Tafoya

Automated Tests: For our automated testing we utilized the Python unittest framework. We split up our testing into two categories: testing account creation/credentials, and testing the insertion/retrieval of information from our MySQL database.

Testing the creation of MasterAccount objects

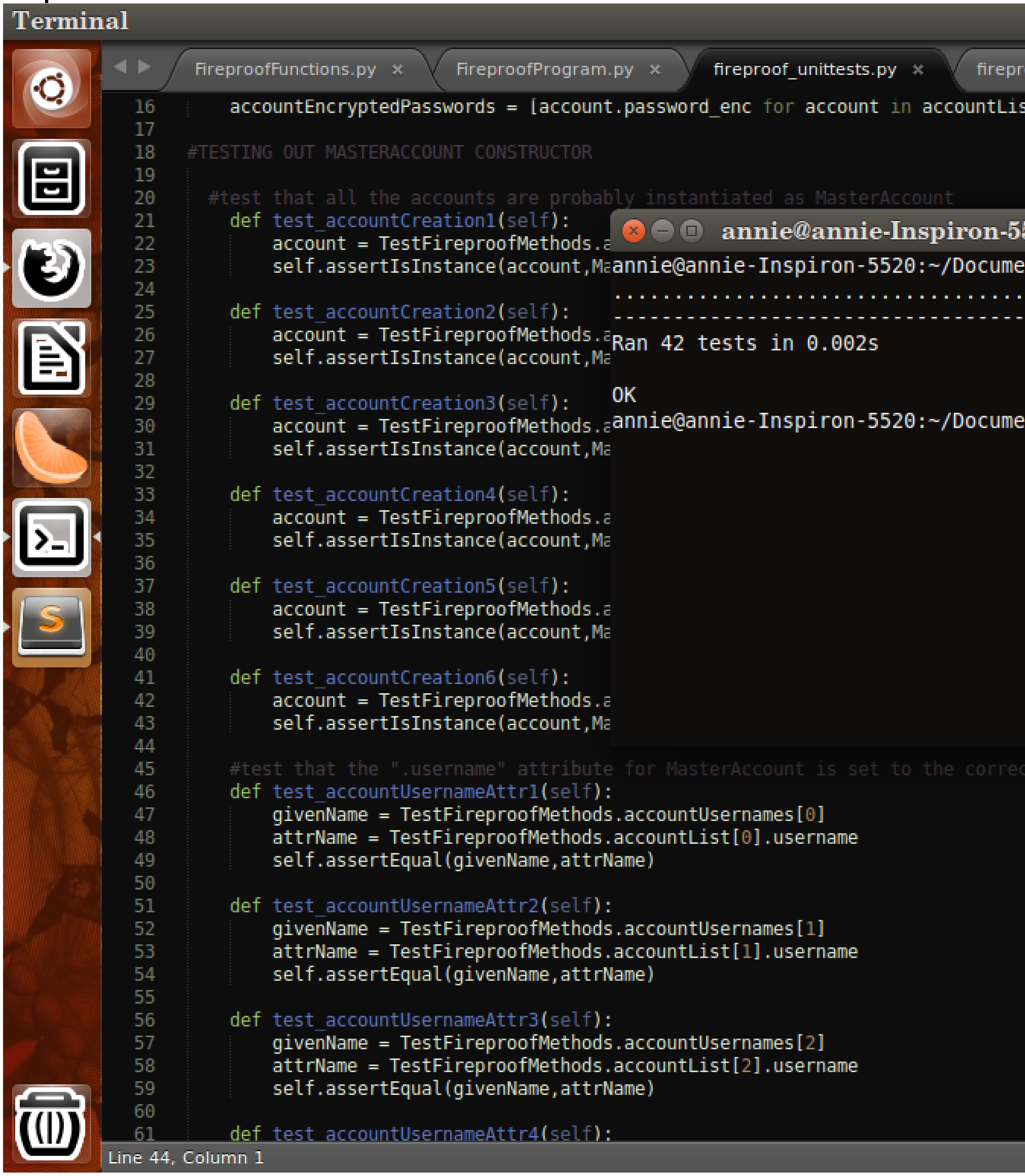
Our tests included:

- ensuring that the MasterAccount() constructor intialized correctly by:
  - asserting that the object was an instance of MasterAccount
  - asserting that the username given to the constructor was initalized correctly and accessible using the .username attribute
  - asserting that the password given to the constructor was initialized correctly and accessible using the .password attribute
- ensuring that our encryption suite worked properly by:
  - asserting that the username given to the constructor was not equal to the username after encryption
  - asserting that the password given to the constructor was not equal to the password after encryption
  - asserting that when we decrypt the encrypted username using the same symmetric key and initialization vector, that it is equal to the username that was originally given to the constructor
  - asserting that when we decrypt the encrypted password using the same symmetric key and initialization vector, that it is equal to the password that was originally given to the constructor

Instructions to run:

- Clone entire git repository
- Must have mysql-server, python-mysqldb, and python-tk packages installed
- run “python -m unittest fireproof_unittests”

Output:



```
    accountEncryptedPasswords = [account.password_enc for account in accountLis

#TESTING OUT MASTERACCOUNT CONSTRUCTOR

  #test that all the accounts are probably instantiated as MasterAccount
    def test_accountCreation1(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    def test_accountCreation2(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    def test_accountCreation3(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    def test_accountCreation4(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    def test_accountCreation5(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    def test_accountCreation6(self):
        account = TestFireproofMethods.a
        self.assertIsInstance(account,Ma

    #test that the ".username" attribute for MasterAccount is set to the correc
    def test_accountUsernameAttr1(self):
        givenName = TestFireproofMethods.accountUsernames[0]
        attrName = TestFireproofMethods.accountList[0].username
        self.assertEqual(givenName,attrName)

    def test_accountUsernameAttr2(self):
        givenName = TestFireproofMethods.accountUsernames[1]
        attrName = TestFireproofMethods.accountList[1].username
        self.assertEqual(givenName,attrName)

    def test_accountUsernameAttr3(self):
        givenName = TestFireproofMethods.accountUsernames[2]
        attrName = TestFireproofMethods.accountList[2].username
        self.assertEqual(givenName,attrName)

    def test_accountUsernameAttr4(self):
```

Line 44, Column 1

annie@annie-Inspiron-55

```
annie@annie-Inspiron-5520:~/Docume
..................................
----------------------------------
Ran 42 tests in 0.002s

OK
annie@annie-Inspiron-5520:~/Docume
```

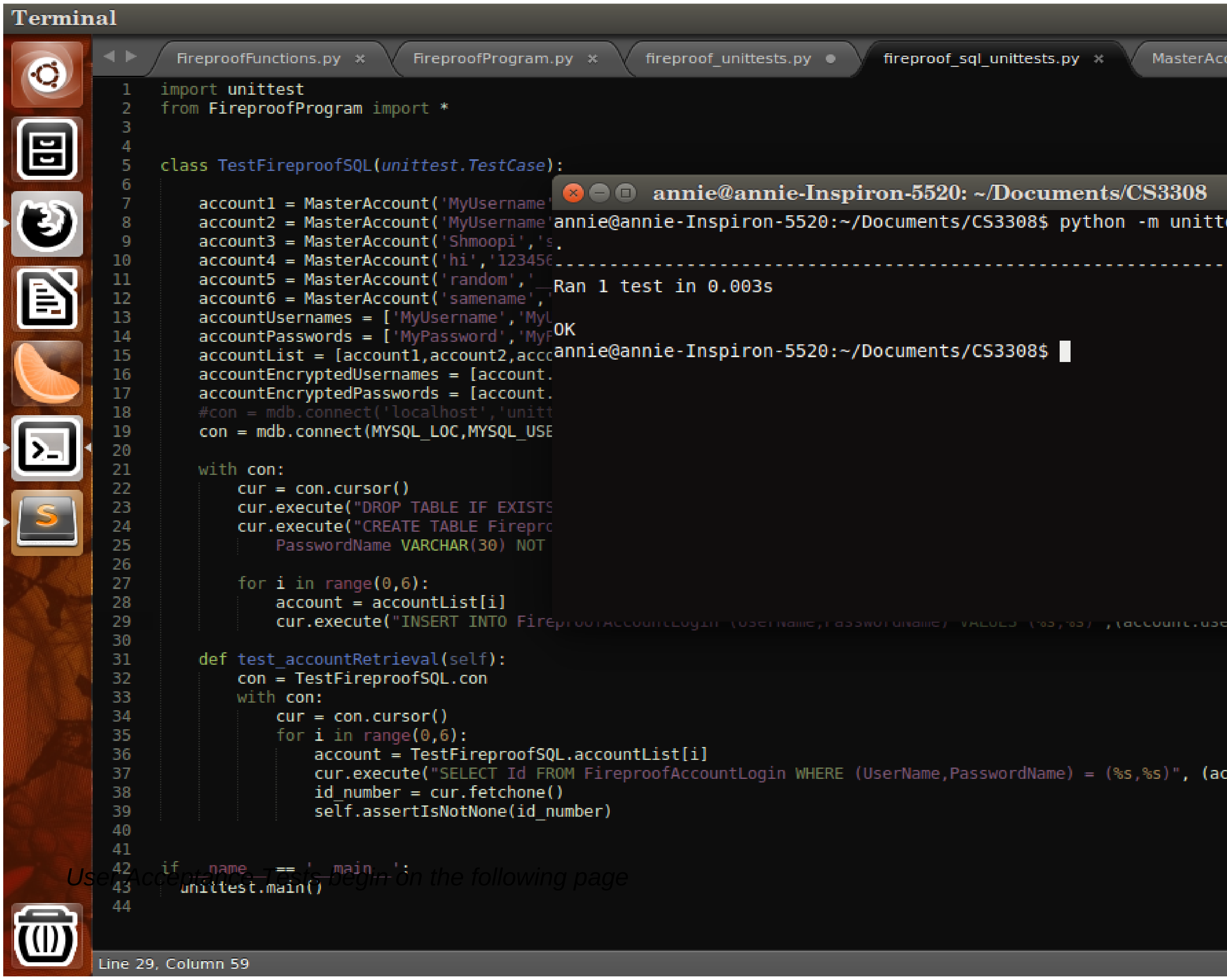
Testing the insertion and selection of accounts in our database
Our tests included:

- inserting 6 different MasterAccounts into our MySQL database and then asserting that when we retrieve an account from the database that it does not return None

Instructions to run:

- Clone entire git repository
- Must have mysql-server, python-mysqldb, and python-tk packages installed
- Running this testfile requires some extra steps:
    - launch MySQL by running "mysql -u root -p"
    - log in with root password
    - In the MySQL prompt run “CREATE DATABASE Fireproof;”
    - in the MySQL prompt run “CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'testAnnie';”
    - next run “GRANT ALL PRIVILEGES ON `%`.* TO 'testuser'@'localhost' IDENTIFIED BY 'testAnnie' WITH GRANT OPTION;”
- run “python -m unittest fireproof_sql_unittests”

Output:


Terminal

FireproofFunctions.py | FireproofProgram.py | fireproof_unittests.py | fireproof_sql_unittests.py | MasterAc

```
import unittest
from FireproofProgram import *


class TestFireproofSQL(unittest.TestCase):

    account1 = MasterAccount('MyUsername'
    account2 = MasterAccount('MyUsername'
    account3 = MasterAccount('Shmoopi','s
    account4 = MasterAccount('hi','123456
    account5 = MasterAccount('random','
    account6 = MasterAccount('samename','
    accountUsernames = ['MyUsername','MyU
    accountPasswords = ['MyPassword','MyP
    accountList = [account1,account2,acco
    accountEncryptedUsernames = [account.
    accountEncryptedPasswords = [account.
    #con = mdb.connect('localhost','unitt
    con = mdb.connect(MYSQL_LOC,MYSQL_USE

    with con:
        cur = con.cursor()
        cur.execute("DROP TABLE IF EXISTS
        cur.execute("CREATE TABLE Fireprc
            PasswordName VARCHAR(30) NOT

        for i in range(0,6):
            account = accountList[i]
            cur.execute("INSERT INTO FireproofAccountLogin (UserName,PasswordName) VALUES (%s,%s)",(account.use

    def test_accountRetrieval(self):
        con = TestFireproofSQL.con
        with con:
            cur = con.cursor()
            for i in range(0,6):
                account = TestFireproofSQL.accountList[i]
                cur.execute("SELECT Id FROM FireproofAccountLogin WHERE (UserName,PasswordName) = (%s,%s)", (ac
                id_number = cur.fetchone()
                self.assertIsNotNone(id_number)


if __name__ == '__main__':
  unittest.main()
```

Line 29, Column 59

annie@annie-Inspiron-5520: ~/Documents/CS3308

```
annie@annie-Inspiron-5520:~/Documents/CS3308$ python -m unitt
.
----------------------------------------------------------------
Ran 1 test in 0.003s

OK
annie@annie-Inspiron-5520:~/Documents/CS3308$
```

User Acceptance Tests begin on the following page

User Acceptance Tests:

Test Case 1:

Test Case ID: FUNC-002-01
Designed by: Kara & Annie
Designed date: 3/31/2015
Test Priority (Low/Medium/High): High
Module Name: Create a new account
Description: Create a new account by providing username and password on the sign up page

Preconditions: Username and passwords fields must be non-Null, password must be 8+ characters, password and confirm password fields must match
Dependencies: MySQL database, MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Click “Sign Up” on start screen		Create Account screen appears		Pass
2	Provide valid username	username = BillyBob	Username can be seen in the field as plaintext		Pass
3	Provide valid password	password = Password	Password is seen as *’s		Pass
4	Confirm valid password	confirmpassword = Password	Confirmed password is seen as *’s		Pass
5	Click “Create Account” button		Username and password get encrypted using the master password and get inserted in the database		Pass

Postconditions: user’s information is added to the database and user is redirected to account login page

Test Case 2:

Test Case ID: FUNC-001-01, FUNC-004
Designed by: Kara & Annie
Designed date: 3/31/2015
Test Priority (Low/Medium/High): High
Module Name: Logging in to account
Description: Login into account by verifying username and password on the login page
Preconditions: Must have previously created an account
Dependencies: MySQL database, MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Navigate to login screen				Pass
2	Provide valid username	username = BillyBob	Username can be seen in the field as plaintext		Pass
3	Provide valid password	password = Password	Password is seen as *'s		Pass
4	Click "Login" button		User is navigated to their account's main screen		Pass

Postconditions: user's information in database is unencrypted and compared with the user's input; if they match, the user is redirected to their account's main screen

Test Case 3:

Test Case ID: NONF-001
Designed by: Kara & Annie
Designed date: 3/31/2015
Test Priority (Low/Medium/High): Medium
Module Name: Go Back Button

Description: Clicking “Go Back” on the Create Account page in order to get back to the Login page
Preconditions: None
Dependencies: Python Tkinter

Step	Test Step	Test Data	Expected Result	Actual Result	Status
1	Navigate to login screen				Pass
2	Click “Sign Up” button on the login screen		Create Account screen appears		Pass
3	Click on the “Go Back” button		User is navigated back to the Login screen		Pass

Postconditions: User is navigated back to the Login screen for Fireproof

Test Case 4:

Test Case ID: FUNC-001-02
Designed by: Kara & Annie
Designed date: 3/31/2015
Test Priority (Low/Medium/High): Medium
Module Name: Account not found
Description: User tried to log in to account that does not exist
Preconditions: Account must not exist
Dependencies: MySQL database, MasterAccount.py methods, Python Tkinter

Step	Test Step	Test Data	Expected Result	Actual Result	Status
1	Navigate to login screen				Pass
2	Provide invalid username	Non existent username	Username can be seen in the field as plaintext		Pass

3	Provide invalid password	Non existent password	Password is seen as *'s		Pass
4	Click "Login" button		A window pops up that says "Username and password not found. Please click create an account" and a button says "OK"		Pass
5	Click "OK" button		Popup window closes		Pass

Postconditions: program checks the database, does not find the login information; user is redirected to the login screen

Test Case 5:

Test Case ID: FUNC-002-02
Designed by: Kara & Annie
Designed date: 3/31/2015
Test Priority (Low/Medium/High): Medium
Module Name: Passwords do not match
Description: While a user is creating an account the entries for the "password" and "confirm password" fields do not match
Preconditions: Password fields must not match
Dependencies: MasterAccount.py methods, Python Tkinter

Step	Test Step	Test Data	Expected Result	Actual Result	Status
1	Click "Sign Up" on the start screen		Create Account screen appears		Pass
2	Provide valid username	username = BillyBob	Username can be seen in the field as plaintext		Pass
3	Provide valid password	password = Password	Password is seen as *'s		Pass
4	Provide valid	password =	Confirm password is seen		Pass

	password for confirm password	Passwordd	as *'s		
5	Click "Create Account" button		A window pops up that says "Passwords do not match" with a button that says "OK"		Pass
6	Click "OK" button		Popup window closes		Pass

Postconditions: user ends up at the create account screen; no account is created

Test Case 6:

Test Case ID: FUNC-005

Designed by: Kara & Annie

Designed date: 3/31/2015

Test Priority (Low/Medium/High): Medium

Module Name: Password too short

Description: While a user is creating an account their password is less than 8 characters

Preconditions: Password must be < 8 characters

Dependencies: MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Click "Sign Up" on the start screen		Create Account screen appears		Pass
2	Provide valid username	username = BillyBob	Username can be seen in the field as plaintext		Pass
3	Provide invalid password	password = Passwor	Password is seen as *'s		Pass
4	Click "Create Account" button		A window pops up that says "Password must be at least 8 characters." with a button that says "OK"		Pass
5	Click "OK"		Popup window closes		Pass

	button				

Postconditions: user ends up at the “Create Account” screen; no account is created

*****Beyond this point, the test cases represent future tests to perform once code and functionality is implemented.**

Test Case 7:

Test Case ID: FUNC-002-03
Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): High
Module Name: Existing username
Description: While a user is creating an account their username matches an existing username in the database.
Preconditions: Username must match existing username
Dependencies: MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Click “Sign Up” on the start screen		Create Account screen appears		Not tested
2	Provide invalid username	username = Existing	Username can be seen in the field as plaintext		Not tested
3	Provide valid password	password = Password	Password is seen as *’s		Not tested
4	Click “Create Account” button		A window pops up that says “Username already exists!” with a button that says “OK”		Not tested
5	Click “OK” button		Popup window closes		Not tested

Postconditions: user ends up at the “Create Account” screen; no account is created

Test Case 8:

Test Case ID: FUNC-001-03
Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): High
Module Name: Incorrect password
Description: While a user is logging into their account, they enter the correct username but wrong password.
Preconditions: Password must not match what is expected for the given username
Dependencies: MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Navigate to login screen		Login screen appears		Not tested
2	Provide valid username	username = fireproof	Username can be seen in the field as plaintext		Not tested
3	Provide invalid password	password = thisaintright	Password is seen as *’s		Not tested
4	Click “Login” button		A window pops up that says “Username or password is incorrect” with a button that says “OK”		Not tested
5	Click “OK” button		Popup window closes		Not tested

Postconditions: user ends up at the “Login” screen; no account is created

Test Case 9:

Test Case ID: FUNC-006

Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): High
Module Name: Add new service
Description: Once user has logged in, they should be able to add a new service with corresponding credentials
Preconditions: User must have logged in successfully
Dependencies: MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**
1	Log in to Fireproof		Home screen appears		Not tested
2	Click “Add Service” button		Screen changes to “Create new service” page		Not tested
3	Provide valid Service name	service = Facebook	Service can be seen in the field as plaintext		Not tested
4	Provide valid username and password		Username can be seen in the field as plaintext, while password is seen as *’s		Not tested
5	Click “Add” button		User is returned to home screen		Not tested

Postconditions: user ends up at the “Home” screen with new service listed.

Test Case 10:

Test Case ID: FUNC-006-01
Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): High
Module Name: View existing service
Description: Once user has logged in, they should be able to view an existing service

with corresponding credentials
Preconditions: User must have logged in successfully and have already created a service with the buttons “Edit” and “Delete” next to it.
Dependencies: MasterAccount.py methods, Python Tkinter

Step	Test Step	Test Data	Expected Result	Actual Result	Status
1	Log in to Fireproof		Home screen appears		Not tested
2	Click on an existing service		Screen changes to “Service” page		Not tested
3	Verify that user can view Service name, username, and password		Page should show service name, username, password, “Edit”, “Delete”, and “Go Home” buttons		Not tested
4	Click “Go Home” button		User is returned to home screen		Not tested

Postconditions: user ends up at the “Home” screen

Test Case 11:

Test Case ID: FUNC-006-02
Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): Medium
Module Name: Edit existing service
Description: Once user has logged in, they should be able to edit an existing service with corresponding credentials
Preconditions: User must have logged in successfully and have already created a service with the buttons “Edit” and “Delete” next to it.
Dependencies: MasterAccount.py methods, Python Tkinter

Step	Test Step	Test Data	Expected Result	Actual Result	Status

1	Log in to Fireproof		Home screen appears		Not tested
2	Click “Edit” button on an existing service		Screen changes to “Edit service” page		Not tested
3	Change service name	Old Service = Facebook New Service = Bookface	Service can be seen in the field as plaintext		Not tested
4	Click “Done editing” button		User is returned to home screen		Not tested
5	Verify that the edit is reflected on the homescreen.		Service should reflect edits on home screen.		Not tested
6	Repeat steps 2-5 changing username and password		Service should reflect edits on home screen		Not tested

Postconditions: user ends up at the “Home” screen with updated and edited service credentials.

Test Case 12:

Test Case ID: FUNC-006-03
Designed by: Tyler
Designed date: 4/1/2015
Test Priority (Low/Medium/High): Medium
Module Name: Delete existing service
Description: Once user has logged in, they should be able to delete an existing service with corresponding credentials
Preconditions: User must have logged in successfully and have already created a service with the buttons “Edit” and “Delete” next to it.
Dependencies: MasterAccount.py methods, Python Tkinter

Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status**

1	Log in to Fireproof		Home screen appears		Not tested
2	Click “Delete” button on an existing service		Popup window appears with message “Are you sure you want to delete <Service name>” with “Yes” and “No” buttons		Not tested
3	Click “No” button		User is returned to home screen		Not tested
4	Click “Delete” button		Popup window appears with message “Are you sure you want to delete <Service name>” with “Yes” and “No” buttons		Not tested
5	Click “Yes”		User is returned to home screen		Not tested
6	Verify that service has been deleted		Service should no longer appear on home screen		Not tested

Postconditions: user ends up at the “Home” screen with the deleted service no longer appearing

VCS: https://github.com/bshadowfax6894/CS3308