TEST EXECUTION DOCUMENT

FOR

PAAVNI ORGANIC DAIRY FARM

Version 1.1 approved

Members: Aakriti Arya

Aman Jain

Ashwin Jayadev

Hrithik Puri

Deep Parikh

Organization: NIIT University

INTRODUCTION

**PURPOSE**

The purpose of the entire test case document is to ensure that our product which is basically an ERP System for an Organic Dairy Farming firm named PAAVNI is free of any errors and is responding to all the action in the well anticipated time. This document only includes the test cases in different forms. Testing of the product has been done mainly in two different ways.

* Unit Testing
* Integration Testing

**TESTING PURPOSE**

As mentioned above the testing that has been executed can be widely divided in to two major parts which were mentioned above in the purpose.

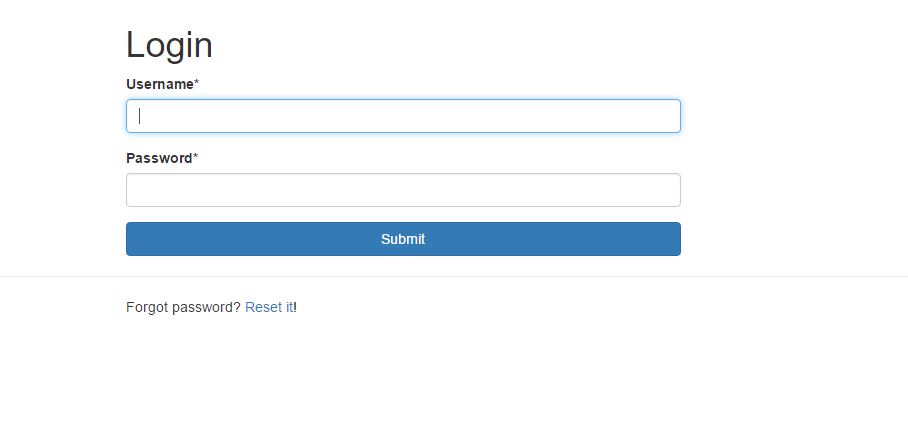
Upon Unit Testing, complete product has been divided in to small units. All the units are tested separately. Each unit test cases can be broadly divided into black box test cases and white box test cases. Black box and white box is a type of deriving test cases for the unit. These entire tests are again described/ explained clearly with appropriate reasons.

Upon Integration Testing, complete product branches have been divided in to integrated parts. All these integrated parts are the addition of units only. So before undergoing integration testing it should have completed unit testing. Each integration test case can be broadly divided into black box test cases and white box test cases. Black box and white box is a type of deriving test cases. These entire tests are again described/ explained clearly with appropriate reasons.

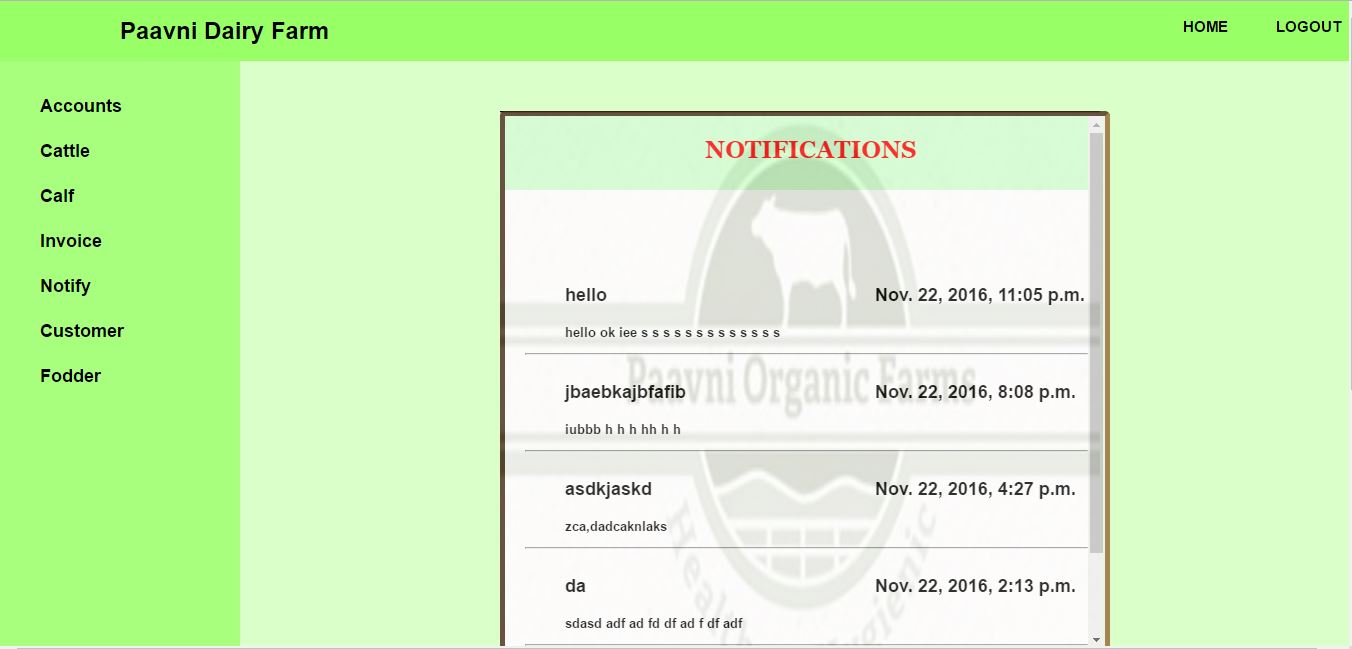
**BLACK BOX TEST CASES**

**TEST CASE 1: LOGIN PAGE TEST CASE**

The login page of our ERP System consists of two entry fields namely Username and Password in which both of them mandatory to be filled in order to log in successfully into the system. Both of the fields are case sensitive.

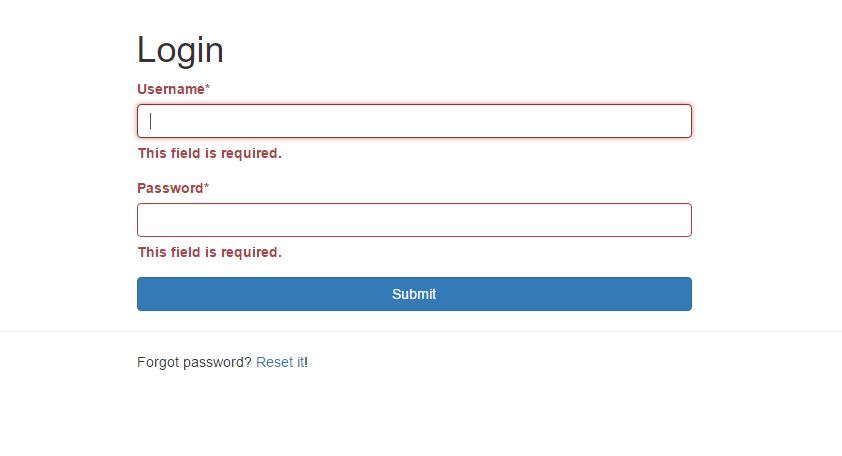


**POISITVE TEST SCENARIO:** To check if our login page is working correctly we need to enter correct username and its respective password and once you do this you will be directed to the home page of the ERP System, which looks something like this

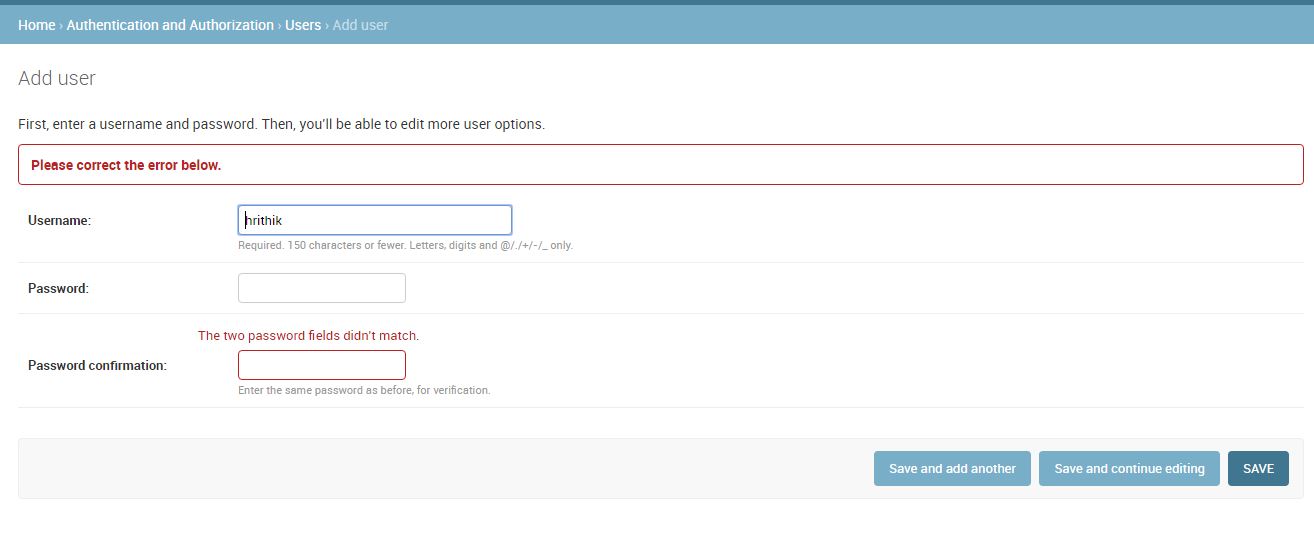


**NEGATIVE TEST SCENARIO:**

1. **EMPTY FIELD SCENARIO:** If the user tries to login into the ERP system without filling in the details for both the fields that is username and password both respectively then one cannot successfully login and access the ERP System.



1. **INVALID ENTRY:** If the use enter invalid username or invalid password for a respective username then one cannot login into the ERP System and it will show an error.

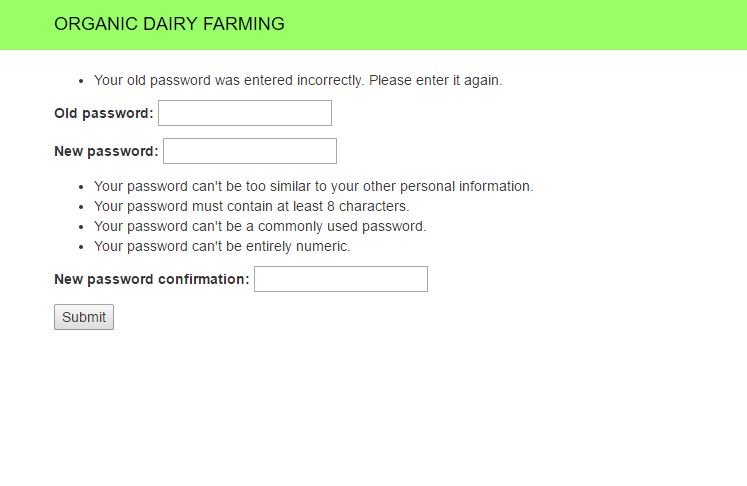


TEST CASE 2: PASSWORD CHANGE

Supposedly one of the existing user of the ERP system wants to change the password for their respective username then there is a certain procedure to be followed, in which the user has to click on this button forgot password and then fill in the old password and choose a new password which should have certain characteristics such as minimum length and mandatory special characters.

POSITIVE TEST SCENARIO:

OLD PASSWORD ENTERED: if the user forgets his or her current password and enters and old password by mistake then he is directed to this very page wherein he get the opportunity to change his password by filling the below shown fields



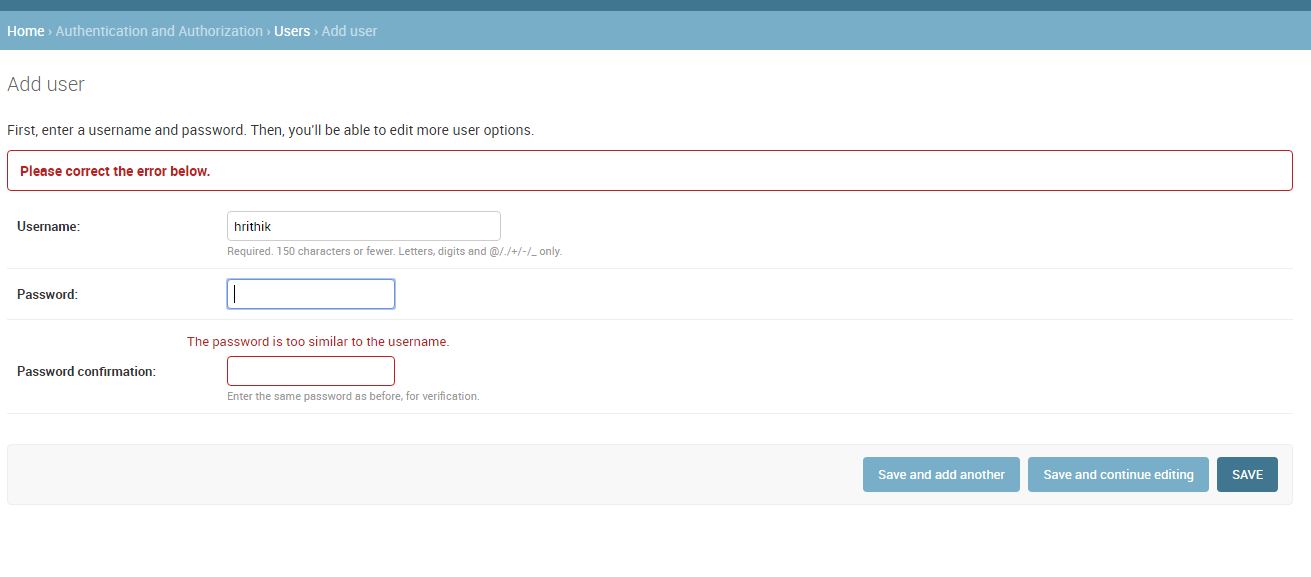
Once the password is successfully changed we directed to a page which shows that the password has been successfully changed



NEGTAIVE TEST CASE SCENARIO:

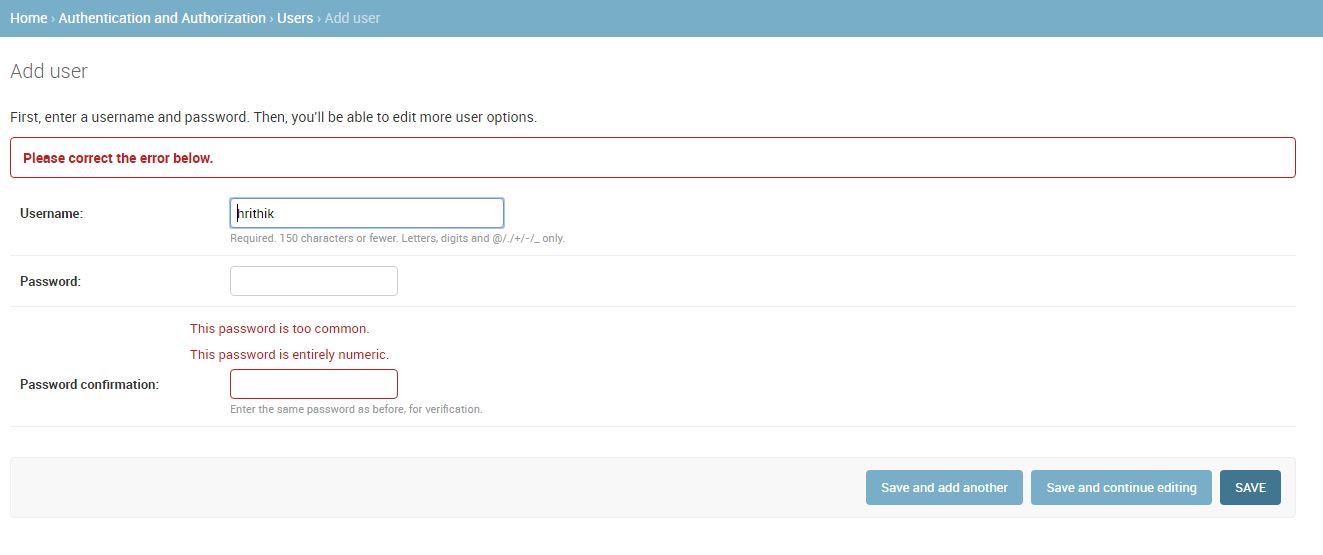
1. USERNAME AND CHANGED PASSWORD MATCHES:

If the password is too similar to the username it is very easy to crack and thus to ensure security if the user tries to fill in a password which is too similar to the username an error is shown.



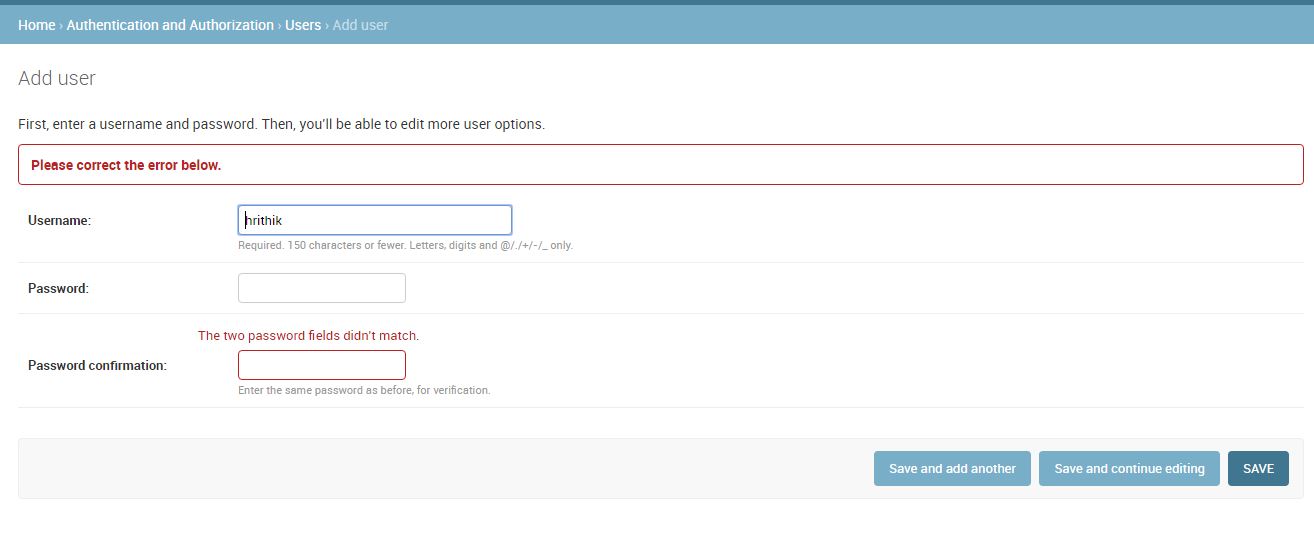
1. PASSWORD IS TOO COMMON:

Just in case the user tries to set up a password which is very common such as 12345 or 00000 then Third party API which is basically used for password verification shows an error that the password is too common and thus can be easily cracked and cant be used as a password.



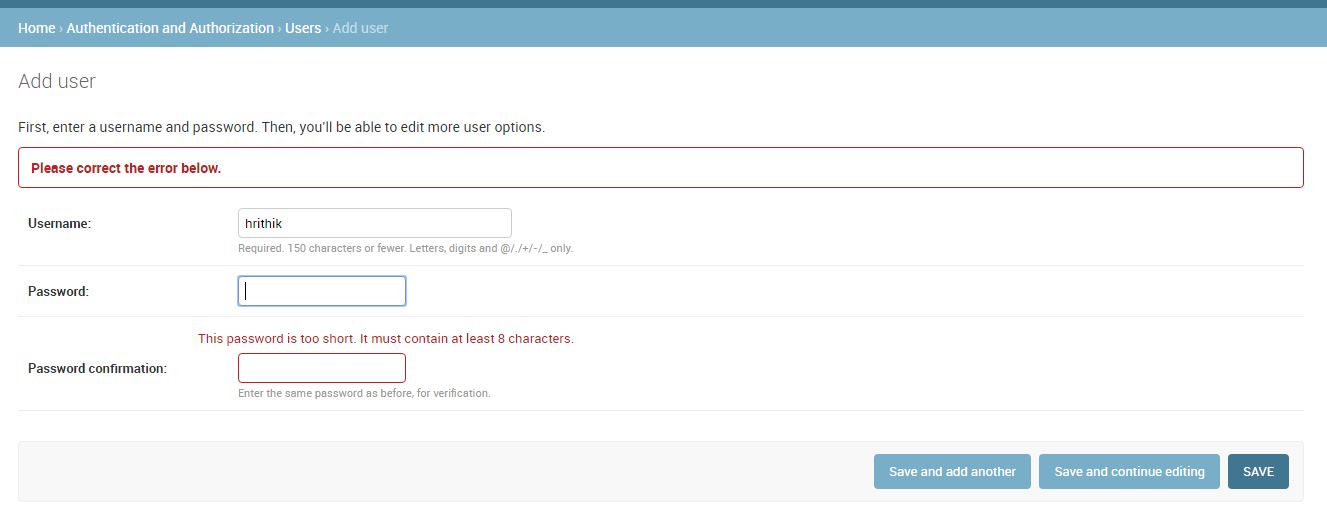
1. PASSWORD AND PASSWORD CONFIRMATION DOES NOT MATCH:

While setting up the new password just for being sure after entering the password the user needs to fill the same password in another field named as password confirmation. If these two fields do not match then an error is shown stating the password field doesn’t match.



1. PASSWORD SHORT ERROR :

If the password entered by the user is too short then an error is generated for security purpose saying that the password is too short and thus easy to guess.



TEST CASE 3: INVOICE GENERATION

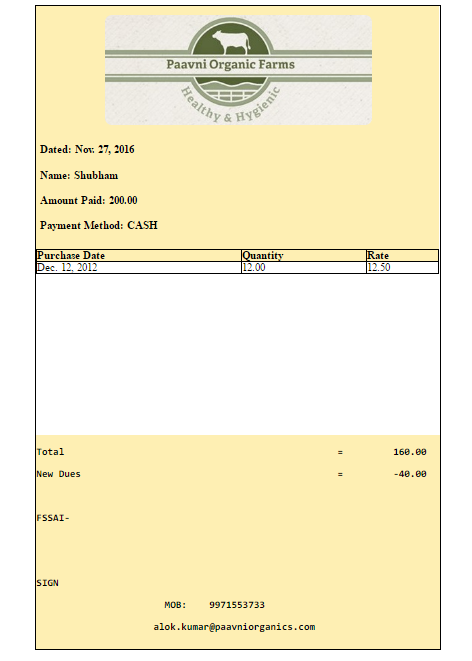
During invoice generation there are some certain specific fields that are required to be filled in order to generate the invoice successfully. If one fails to do so the invoice cannot be generated successfully.

POSITIVE TEST CASE SCENARIO:

Once the user fills in all the necessary details in correct fashion while taking care of all the domains then an invoice can be generated successfully.

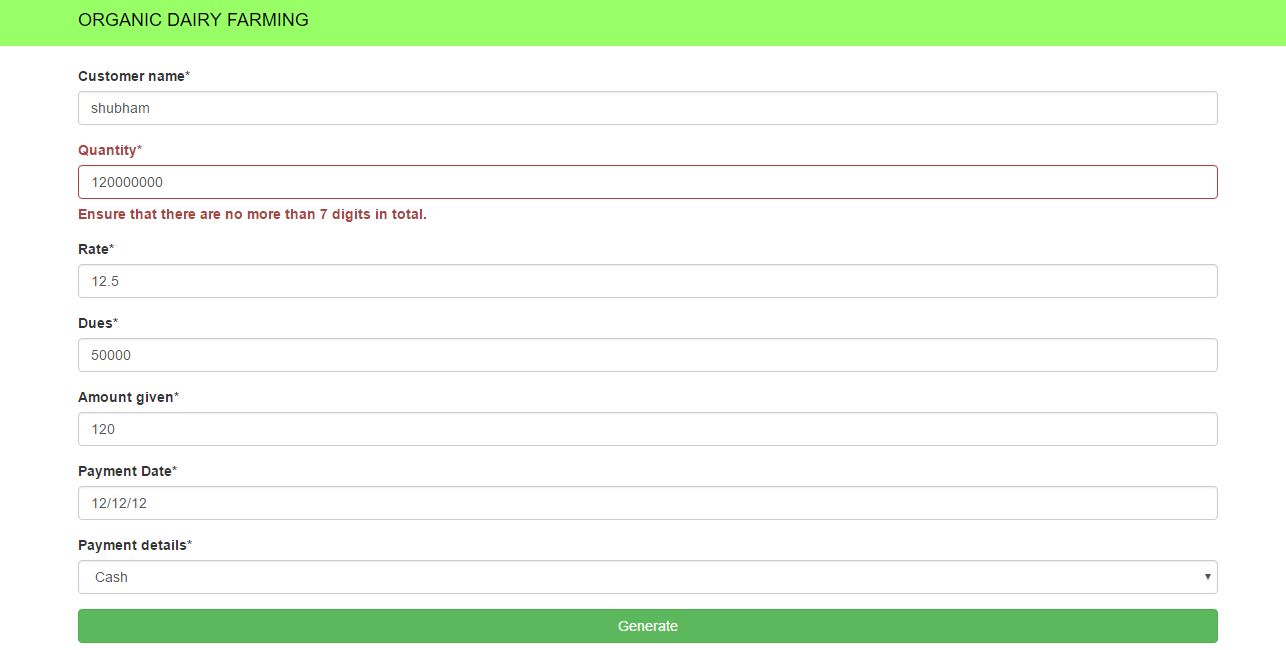


Once the details are filled just like the above then an invoice is printed out



NEGATIVE TEST SCENARIO:

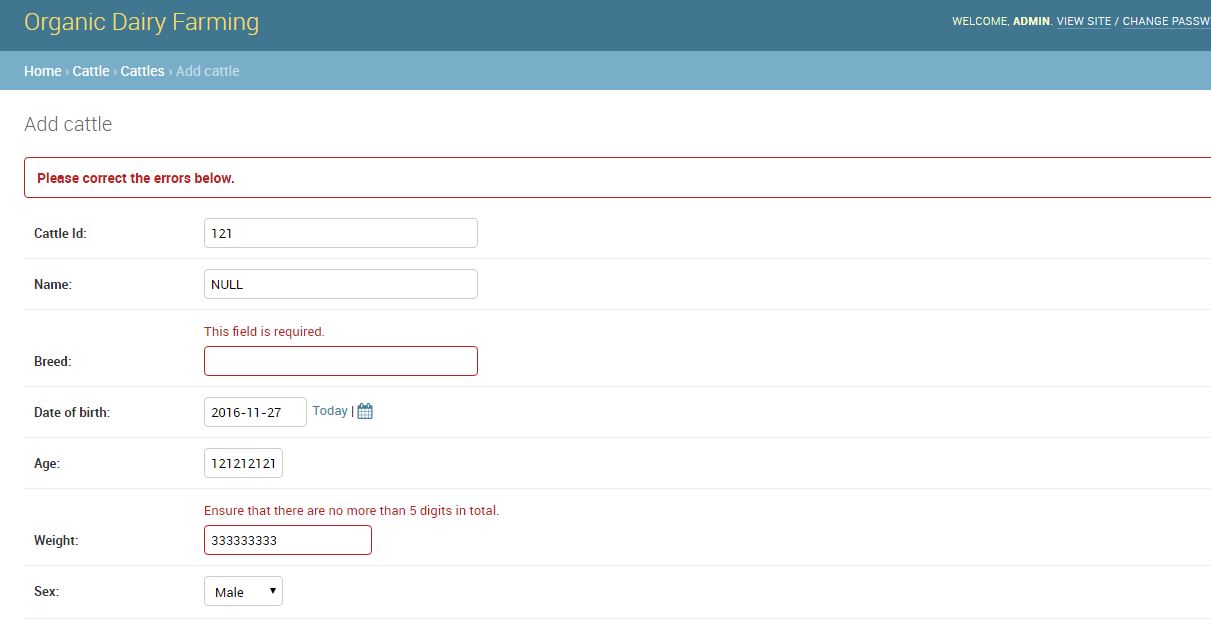
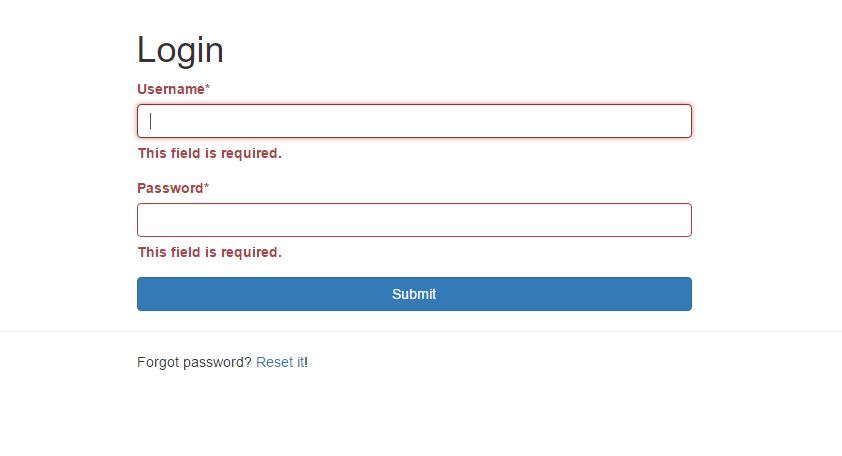
1. LIMIT EXCEEDED:

While filling in the details we need to ensure that the value that one inputs is not out of domain for example while filling the quantity field if you fill the value out of domain then an error is shown.

TEST CASE 4: CATTLE UPDATION

During filling in the information for the cattle the user has to enter correct information and failing in doing so will cause an error in which one cannot update the database.

NEGATIVE CASE SCENARIO:

1. LIMITED EXCEEDED: if we fill in values which are outside the domain then an error is shown and the data cannot be updated properly.
2. LEAVING FIELD EMPTY: If you leave one field empty then also an error is shown and it is not updated properly.

UNIT TESTING

WHITE BOX TESTING

TEST CASE 1: LOGIN CODE

The below given code is written in python for the login in which while logging in for a particular username the code accepts a particular password. Here in this given test case for the username ADMIN the password accepted is password123 if you input any other password an error is generated thus barring the malicious user from accessing the ERP System.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Test2(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_2(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("passw123")

driver.find\_element\_by\_id("submit").click()

try:

self.assertTrue(self.is\_element\_present(By.CSS\_SELECTOR, "section"))

except AssertionError as e:

self.verificationErrors.append(str(e))

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

def tearDown(self):

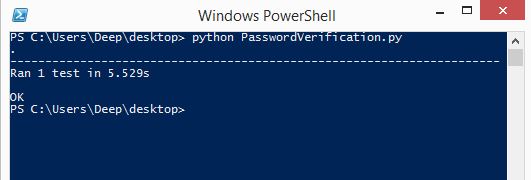
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

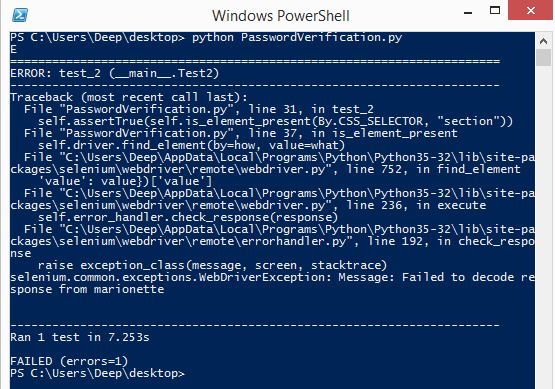
if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGES FOR THE TEST



Since the password entered is correct the code accepts it and runs successfully.



If someone enters the wrong password the code doesn’t accept the password and it shows the following error.

TEST CASE 2: VALID CATTLE

While accessing the ERP System if the admin or the user wants to check the details about a particular cattle then all he has to do is enter the name of the cattle and the data would appear but this function works if and only if there exists such a cattle in the database. If you enter a random name then it wont show the details.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Cattle(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_cattle(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[2]/a/h2").click()

driver.find\_element\_by\_link\_text("3423").click()

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

def tearDown(self):

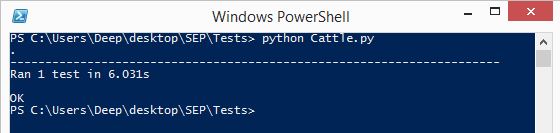
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

TEST CASE IMAGE



TEST CASE 3: INAVLID CATTLE INPUT

If the user enters a random name of cattle which does not exist in the database then this code is run and it outputs that the required cattle does not exist.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Invalidcattle(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://localhost:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_invalidcattle(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[2]/a/h2").click()

driver.find\_element\_by\_link\_text("3423").click()

driver.get(self.base\_url + "/cattlepage/3463/")

def is\_element\_present(self, how, what):

try: self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e: return False

return True

def is\_alert\_present(self):

try: self.driver.switch\_to\_alert()

except NoAlertPresentException as e: return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally: self.accept\_next\_alert = True

def tearDown(self):

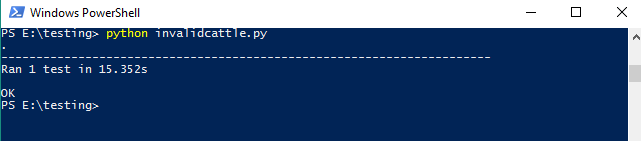
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR TEST CASE



TEST CASE 5: ADD CATTLE

If the user or admin wants to add a new cattle to the database then all he has to do is run this code. While adding a cattle the user needs to enter all the other necessary information such as name, age , breed, vaccination details etc.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class AddCattle(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_add\_cattle(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[2]/a/h2").click()

driver.find\_element\_by\_css\_selector("button").click()

driver.find\_element\_by\_css\_selector("tr.model-cattle > td > a.addlink").click()

driver.find\_element\_by\_id("id\_cattle\_Id").clear()

driver.find\_element\_by\_id("id\_cattle\_Id").send\_keys("1231")

driver.find\_element\_by\_id("id\_Breed").clear()

driver.find\_element\_by\_id("id\_Breed").send\_keys("Vintage")

driver.find\_element\_by\_css\_selector(

"div.form-row.field-date\_of\_expiry\_insurance > div > span.datetimeshortcuts > a").click()

driver.find\_element\_by\_name("\_save").click()

driver.find\_element\_by\_link\_text("View site").click()

driver.find\_element\_by\_xpath("//li[2]/a/h2").click()

driver.find\_element\_by\_link\_text("1231").click()

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

def tearDown(self):

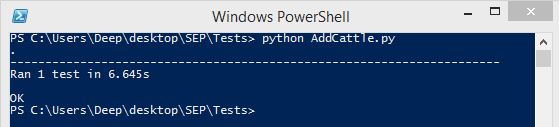
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR THE TEST:



TEST CASE 6: CREATE USER

This particular code is used for making a new user for the ERP system. By using this particular code a new user can be created and thus we need to fill in the basic information for the new user such as name age and get him a new username and password.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class CreateUser(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_create\_user(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_css\_selector("h2").click()

driver.find\_element\_by\_link\_text("Users").click()

driver.find\_element\_by\_link\_text("Add user").click()

driver.find\_element\_by\_id("id\_username").clear()

driver.find\_element\_by\_id("id\_username").send\_keys("deep")

driver.find\_element\_by\_id("id\_password1").clear()

driver.find\_element\_by\_id("id\_password1").send\_keys("passcode")

driver.find\_element\_by\_id("id\_password2").clear()

driver.find\_element\_by\_id("id\_password2").send\_keys("passcode")

driver.find\_element\_by\_name("\_save").click()

driver.find\_element\_by\_id("id\_is\_staff").click()

driver.find\_element\_by\_name("\_save").click()

driver.find\_element\_by\_link\_text("Log out").click()

driver.find\_element\_by\_link\_text("Log in again").click()

driver.find\_element\_by\_id("id\_username").clear()

driver.find\_element\_by\_id("id\_username").send\_keys("deep")

driver.find\_element\_by\_id("id\_password").clear()

driver.find\_element\_by\_id("id\_password").send\_keys("passcode")

driver.find\_element\_by\_css\_selector("input[type=\"submit\"]").click()

driver.find\_element\_by\_link\_text("View site").click()

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

def tearDown(self):

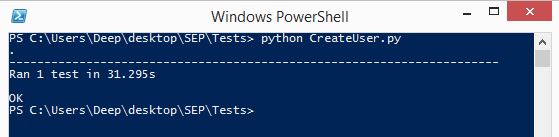
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR THE TEST



CASE TEST 7:NOTIFICATION CODE

The ERP system permits the user to post notification on the website in case of emergency thus this code is responsible for the notification part of the website.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Notify(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_notify(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[5]/a/h2").click()

driver.find\_element\_by\_id("id\_title").clear()

driver.find\_element\_by\_id("id\_title").send\_keys("A\_Notification")

driver.find\_element\_by\_id("id\_body").clear()

driver.find\_element\_by\_id("id\_body").send\_keys("Just a Notification")

driver.find\_element\_by\_xpath("//input[@value='POST']").click()

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

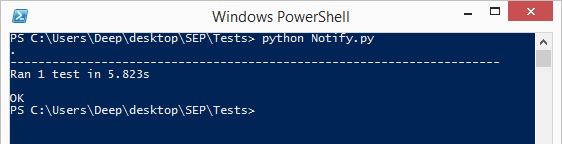
def tearDown(self):

# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()



TEST CASE 8: INVOICE GENERATION CODE

Once the order is successfully played by the customer an invoice is generated thus this code is responsible for this invoice generation and the invoice is generated only when all the necessary inputs are provided in the well defined domain.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Notify(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://127.0.0.1:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_notify(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[5]/a/h2").click()

driver.find\_element\_by\_id("id\_title").clear()

driver.find\_element\_by\_id("id\_title").send\_keys("A\_Notification")

driver.find\_element\_by\_id("id\_body").clear()

driver.find\_element\_by\_id("id\_body").send\_keys("Just a Notification")

driver.find\_element\_by\_xpath("//input[@value='POST']").click()

def is\_element\_present(self, how, what):

try:

self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e:

return False

return True

def is\_alert\_present(self):

try:

self.driver.switch\_to\_alert()

except NoAlertPresentException as e:

return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally:

self.accept\_next\_alert = True

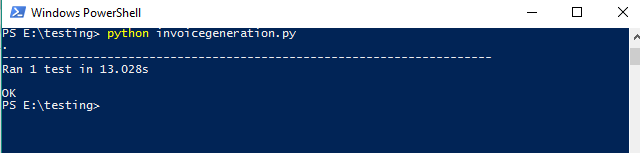
def tearDown(self):

# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR TEST CASE   


TEST CASE 9:CUSTOMER VIEW CODE  
This particular code lets you view the customer and the basic information about the customer.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Custview(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://localhost:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_custview(self):

driver = self.driver

driver.get(self.base\_url + "/accounts/login/")

driver.find\_element\_by\_id("id\_username").clear()

driver.find\_element\_by\_id("id\_username").send\_keys("Admin")

driver.find\_element\_by\_id("id\_password").clear()

driver.find\_element\_by\_id("id\_password").send\_keys("password123")

driver.find\_element\_by\_xpath("//input[@value='Submit']").click()

driver.find\_element\_by\_xpath("//li[6]/a/h2").click()

driver.find\_element\_by\_link\_text("771").click()

def is\_element\_present(self, how, what):

try: self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e: return False

return True

def is\_alert\_present(self):

try: self.driver.switch\_to\_alert()

except NoAlertPresentException as e: return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally: self.accept\_next\_alert = True

def tearDown(self):

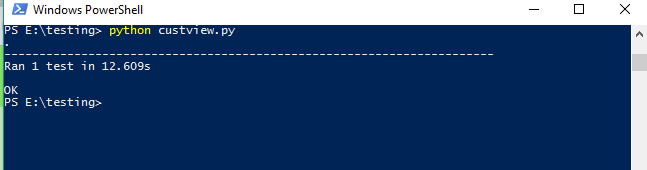
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR TEST CASE



TEST CASE 10:CALF VIEW CODe

Using this code we get to know about the calf and its details which are present in the database iff the calf name is already there in the database.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Calfview(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://localhost:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_calfview(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[3]/a/h2").click()

driver.find\_element\_by\_link\_text("123").click()

def is\_element\_present(self, how, what):

try: self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e: return False

return True

def is\_alert\_present(self):

try: self.driver.switch\_to\_alert()

except NoAlertPresentException as e: return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally: self.accept\_next\_alert = True

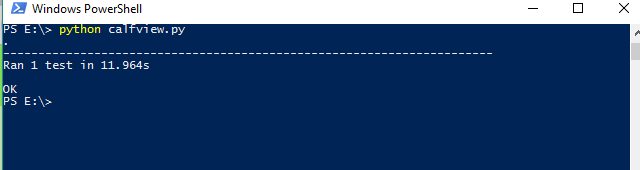
def tearDown(self):

# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()



TEST CASE 11: DELETE CODE

One can delete an entry related to the cattle or calf using this code.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Calfview(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://localhost:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_calfview(self):

driver = self.driver

driver.get(self.base\_url + "/")

driver.find\_element\_by\_id("username").clear()

driver.find\_element\_by\_id("username").send\_keys("Admin")

driver.find\_element\_by\_id("password").clear()

driver.find\_element\_by\_id("password").send\_keys("password123")

driver.find\_element\_by\_id("submit").click()

driver.find\_element\_by\_xpath("//li[3]/a/h2").click()

driver.find\_element\_by\_link\_text("123").click()

def is\_element\_present(self, how, what):

try: self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e: return False

return True

def is\_alert\_present(self):

try: self.driver.switch\_to\_alert()

except NoAlertPresentException as e: return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally: self.accept\_next\_alert = True

def tearDown(self):

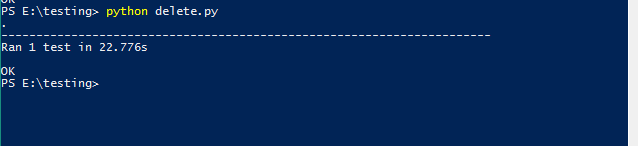
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR THE TEST CASE:



TEST CASE 12: LOGOUT CODE

This code refers to the logout procedure from the ERP System. Once the user is done using the ERP System using this code can logout from the system.

# -\*- coding: utf-8 -\*-

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import Select

from selenium.common.exceptions import NoSuchElementException

from selenium.common.exceptions import NoAlertPresentException

import unittest, time, re

from selenium.webdriver.firefox.firefox\_binary import FirefoxBinary

binary = FirefoxBinary(r'C:\Program Files (x86)\Mozilla Firefox\firefox.exe')

class Logout(unittest.TestCase):

def setUp(self):

self.driver = webdriver.Firefox(firefox\_binary=binary)

self.driver.implicitly\_wait(30)

self.base\_url = "http://localhost:8000"

self.verificationErrors = []

self.accept\_next\_alert = True

def test\_logout(self):

driver = self.driver

driver.get(self.base\_url + "/accounts/login/")

driver.find\_element\_by\_id("id\_username").clear()

driver.find\_element\_by\_id("id\_username").send\_keys("Admin")

driver.find\_element\_by\_id("id\_password").clear()

driver.find\_element\_by\_id("id\_password").send\_keys("password123")

driver.find\_element\_by\_xpath("//input[@value='Submit']").click()

driver.find\_element\_by\_link\_text("LOGOUT").click()

driver.find\_element\_by\_link\_text("Login").click()

def is\_element\_present(self, how, what):

try: self.driver.find\_element(by=how, value=what)

except NoSuchElementException as e: return False

return True

def is\_alert\_present(self):

try: self.driver.switch\_to\_alert()

except NoAlertPresentException as e: return False

return True

def close\_alert\_and\_get\_its\_text(self):

try:

alert = self.driver.switch\_to\_alert()

alert\_text = alert.text

if self.accept\_next\_alert:

alert.accept()

else:

alert.dismiss()

return alert\_text

finally: self.accept\_next\_alert = True

def tearDown(self):

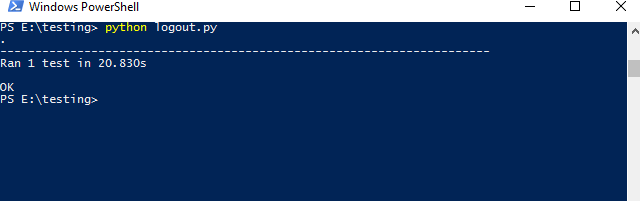
# self.driver.quit()

self.assertEqual([], self.verificationErrors)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

IMAGE FOR THES TEST CASE:



Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Result | Run Time(in sec) | Remarks |
| PasswordVerification\_1 | Pass | 5.529 |  |
| PasswordVerification\_2 | Fail | 7.252 | Invaild credentials |
| AddCattle | Pass | 6.645 |  |
| Cattle\_view | Pass | 6.033 |  |
| Cattle\_update | Pass | 11.23 |  |
| Cattle\_delete | Pass | 22.776 |  |
| addCalf | Pass | 7.321 |  |
| Calf\_view | Pass | 11.964 |  |
| Calf\_update | pass | 12.321 |  |
| Calf\_delete | Pass | 19.542 |  |
| Customer\_add | pass | 22.311 |  |
| Customer\_update | Pass | 14.421 |  |
| Customer\_delete | Pass | 5.532 |  |
| Customer\_view | Pass | 12.609 |  |
| createUser | Pass | 31.259 |  |
| Status\_change | Pass | 6.722 |  |
| Userdata\_updation | Pass | 16.321 |  |
| Delete\_user | pass | 4.252 |  |
| Create\_notification | Pass | 5.823 |  |
| Update\_notification | Pass | 5.134 |  |
| Delete\_notification | Pass | 4.235 |  |
| Invalid\_cattle | Pass | 15.352 |  |
| Invalid\_calf | Pass | 3.241 |  |
| Invalid\_cust | Pass | 3.123 |  |
| Invoice\_generation | pass | 13.028 |  |
| Print\_invoice | Pass | 13.423 |  |
| Password\_change | Pass | 21.432 |  |
| Password\_updation | Fail | 24.233 | Wrong entry |
| Password\_updation | Pass | 21.334 |  |
| Logout | Pass | 20.083 |  |