E-commerce Security Testing

By Saroj Shahi

Introduction:

This document provides a detailed explanation of Security testing of the e-commerce portal. The tests are written in Python and executed using selenium and Page-object Model Framework. The main objective of this test is to ensure

that the application is protected from all threats, unauthorized users and vulnerabilities of the system and help developers fix security problems through coding.

Overview of the E-Commerce Application:

Goldstar Shoes:

Rooted in a legacy upheld by the esteemed names of Kiran Shoes Manufacturers Pvt. Ltd (KSM) and Modern Slipper Industries Pvt. Ltd (MSI), The brand stands as pioneers in the footwear industry, renowned for its unwavering commitment to craftsmanship and innovation.

The Goldstar brand was initiated by Noor Pratap Rana in mid 1970s as a family business. It is manufactured by Kiran Shoe Manufacturers which produces about 25000 shoes per day. The brand has been marketed in India, Australia and Malaysia, among other countries.

Muncha.com:

Muncha.com is the most prominent name in gifting among Nepalese all over the world. The platform originated as Muncha House in the 1920s, Nepal's first department store, and expanded into online shopping in 2000 to offer a gift gallery for non-resident Nepalese looking to surprise family and friends back home.

Muncha.com values curating thoughtful gifts, delivering with care, and maintaining a 24×7 presence for its customers. The brand was established with the vision of forging life-long relationships with its customers and takes pride in being reliable, ensuring special deliveries are made every day.

Customers rely on Muncha.com to send gifts for various occasions, including birthdays, anniversaries, Nepalese festivals, rituals, weddings, celebrations of achievements, and international holidays. Whether they are missing family and friends or marking special moments, Muncha.com is the go-to platform for connecting across distances.

Salt Nepal:

Salt Nepal is a vibrant fashion brand dedicated to offering the trendiest ladies wear at affordable prices. Founded with the vision of empowering women through fashion, we bring the latest styles and designs to our customers, ensuring they always look and feel their best without breaking the bank. Our collection is carefully curated to reflect the latest global trends, while also being tailored to the unique preferences and culture of our local market.

Execution

Testing Environment

- Programing Language: Python
- **Testing Framework**: PyTest
- **Browser Automation Tool**: Selenium WebDriver
- **Design Pattern**: Page Object Model (POM)
- **IDE**: PyCharm
- **Browser**: Google Chrome
- **Screenshot Tool**: Lightshot
- **Documentation** Tool: Notion

Prerequisites:

- Python 3 installed on your system

  > To set up the python 3, Open PyCharm and on the terminal line, add the following:

  ```
  pip install python
  ```

- Selenium WebDriver

> To set up the selenium, add the following lines of code in the terminal line:

```
pip install selenium
pip install webdriver_manager
```

- Chrome Driver Manager

> To set up the Chrome driver , add the following lines of code in the terminal line:

```
@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER AS driver
    driver = webdriver.Chrome(service=ChromeService(Chro
meDriverManager().install()))
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()
```

Page Object Model (POM) with Implementation and Test Cases

Goldstar Shoes

Registration Page Object

File: `pages/goldstar/registration_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium.webdriver.common.by import By
```

```
import re

#DEFINE CLASS FOR REGISTRATION PAGE OF GOLDSTAR
class RegistrationPageGoldstar:
    def __init__(self,driver):
        self.driver = driver
        self.first_name_field = By.XPATH,"//input[contains(@n
        self.last_name_field = By.XPATH,"//input[@name='LastN
        self.email_field= By.XPATH,"//input[@id='txtRegUserEm
        self.password_field = By.XPATH,"//input[@id='txtRegUs
        self.confirm_password_field = By.XPATH,"//input[@id='
        self.register_button = By.XPATH,"//button[@id='btnReg

    def open_page(self,url):
        self.driver.get(url)

    def enter_first_name(self,firstname):
        self.driver.find_element(*self.first_name_field).send_

    def enter_last_name(self,lastname):
        self.driver.find_element(*self.last_name_field).send_

    def enter_email(self,email):
        self.driver.find_element(*self.email_field).send_keys

    def enter_password(self,password):
        self.driver.find_element(*self.password_field).send_k

    def enter_confirm_password(self, confirm_password):
        self.driver.find_element(*self.confirm_password_field

    def click_register(self):
        self.driver.find_element(*self.register_button).click

    def is_valid_email(self, email):
        # check the format using Regular Expression(re)
        email_regex = r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-z
        return re.match(email_regex, email) is not None
```

Test Case Implementation

File: `run_test1.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Chro
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.goldstar.registration_page import RegistrationPage

@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER
    driver = webdriver.Chrome(service=ChromeService(ChromeDri
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()

#RUNNING TEST FUNCTION ON GOLDSTAR REGISTRATION PAGE
def test_registration_goldstar(driver):
    email = "johndoe01@gmail.com"
    goldstar_registration = RegistrationPageGoldstar(driver)
    goldstar_registration.open_page("https://www.goldstarshoe
    driver.maximize_window()
    time.sleep(1)
    goldstar_registration.enter_first_name("John")
    time.sleep(1)
    goldstar_registration.enter_last_name("Doe")
    time.sleep(1)
    goldstar_registration.enter_email(email)
    time.sleep(1)
    goldstar_registration.enter_password("Nepali@123")
    time.sleep(1)
```

```
    goldstar_registration.enter_confirm_password("Nepali@123"
    time.sleep(1)
    # goldstar_registration.click_register()
    # time.sleep(1)

    # check the validity of the email
    if goldstar_registration.is_valid_email(email):
        print(f"The given email: {email} is valid")
    else:
        print(f"The given email: {email} is invalid")
```

Authorization Test case of GoldStar:

💡 **Test ID**: test_0001

Objective: Verify that counter user can register an account

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Registration Link
4. Move to Registration Page
5. Fill the registration portal with valid user phone number along with the other details
6. Click on Register Button
7. Your user account has been registered

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Register Page
5. Counter user should be redirected to the Dashboard Page

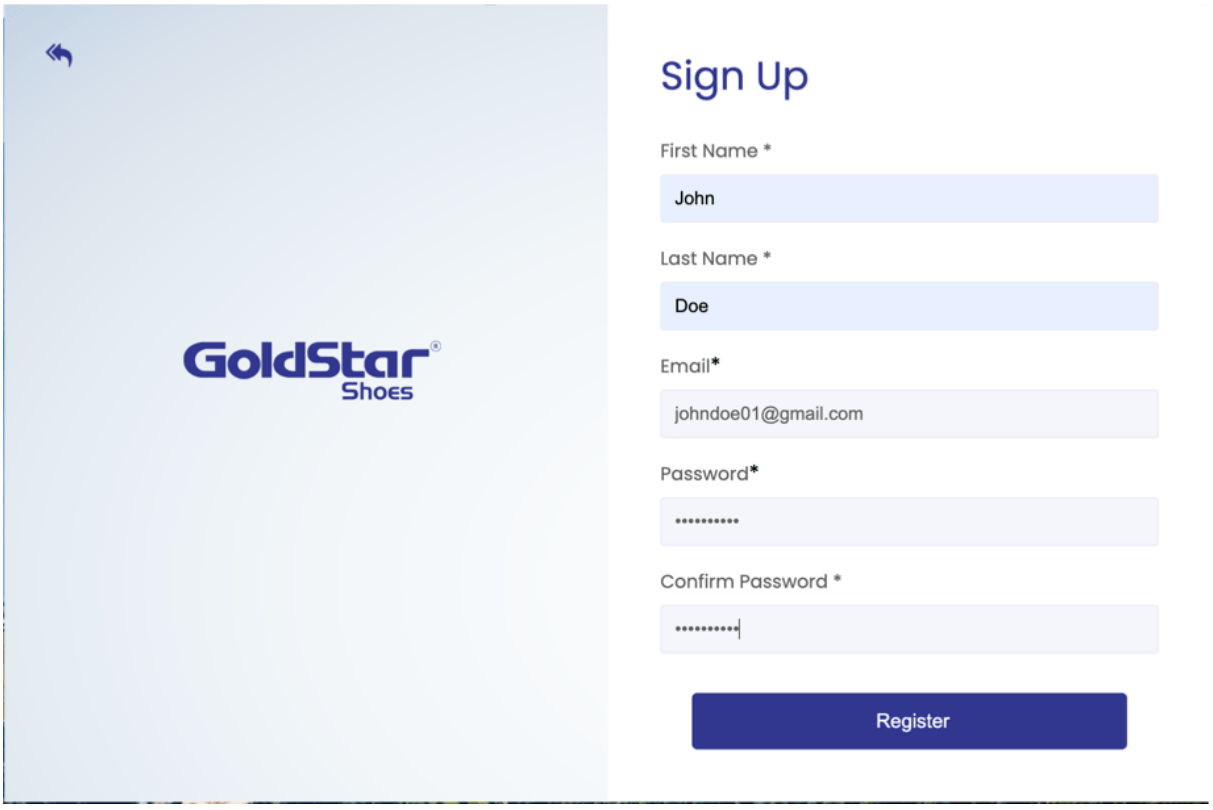
Test Data:

1. URL: https://www.goldstarshoes.com/
2. First Name: John
3. Last Name : Doe
4. Email: johndoe01@gmail.com
5. Password: Nepali@123
6. Confirm-Password: Nepali@123

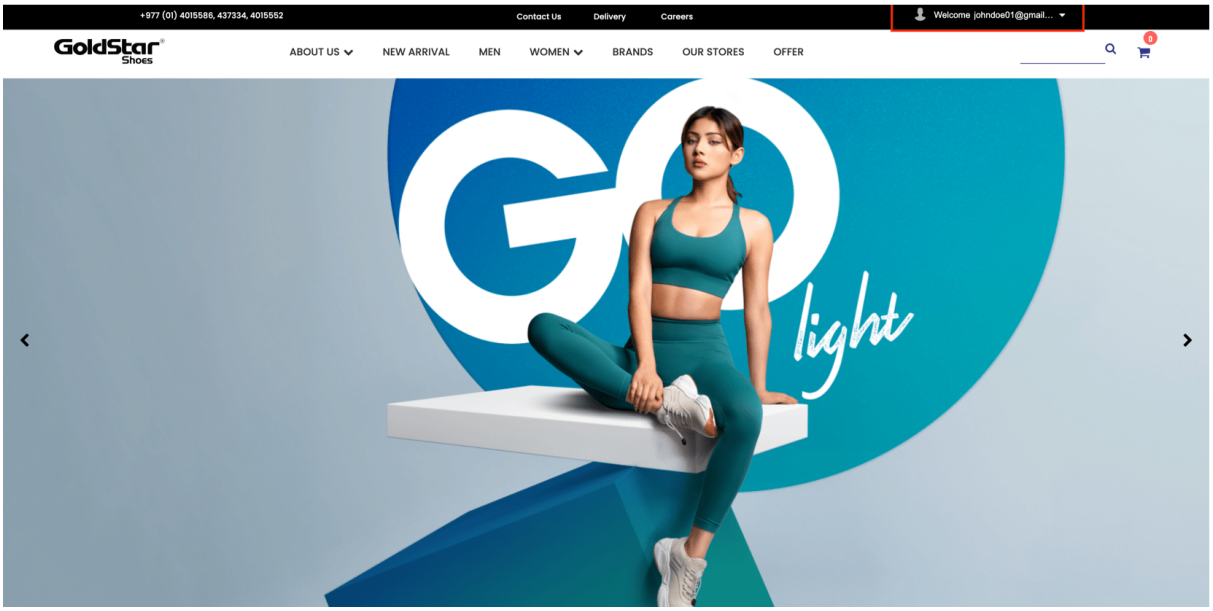
Actual Result: As per expected

Test Status: Pass

Screenshot:



User Input in the Registration Form of Goldstar Shoes



Counter user being redirected to the dashboard page after registration in Goldstar Shoes

Validation Testing of Goldstar:

💡 **Test ID**: test_002

Objective: Verify that email is validated with correct user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Register Link
4. Move to Registration Page
5. Fill the registration portal with valid user email along with the other details
6. Click on Register Button
7. The console will display message "The given email pattern is valid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Register button
5. Valid Email and Phone Message should be displayed in the console

Test Data:

1. URL: https://www.goldstarshoes.com/
2. Email: johndoe01@gmail.com

Actual Result: As per expected

Test Status: Pass

Screenshot:

Checking validation of an email in registration page with correct email pattern

💡 **Test ID**: test_0003

Objective: Verify that Invalid email pattern message is displayed with incorrect user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Register Link
4. Move to Registration Page
5. Fill the registration portal with valid user email along with the other details
6. Click on Register Button
7. The console will display message "The given email pattern is invalid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Create An Account Page
5. Valid Email and Phone Message should be displayed in the console

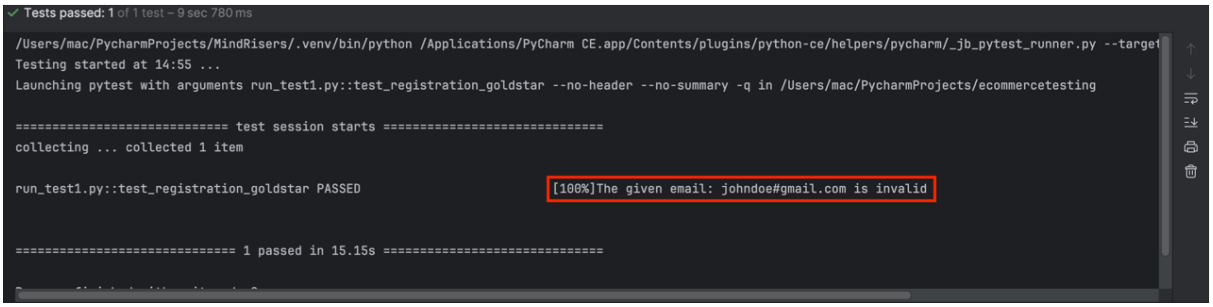
Test Data:

1. URL: https://www.goldstarshoes.com/
2. Email: ram#gmail.com

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking validation of an email in registration page with incorrect email pattern

Login Page Object

File: `pages/goldstar/login_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium.webdriver.common.by import By

#DEFINE CLASS FOR LOGIN PAGE OF GOLDSTAR
class LoginPageGoldstar:
    def __init__(self, driver):
        self.driver = driver
        self.username_field = By.XPATH,"//input[@id='txtUserN
        self.password_field = By.XPATH,"//input[@id='txtUserP
        self.login_button = By.XPATH,"//button[@id='btnSignIn
    def open_page(self,url):
        self.driver.get(url)

    def enter_username(self, username):
        self.driver.find_element(*self.username_field).send_k

    def enter_password(self, pwd):
        self.driver.find_element(*self.password_field).send_k

    def click_login(self):
        self.driver.find_element(*self.login_button).click()
```

Test Case Implementation

File: `run_test1.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Ch
romeService
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.goldstar.login_page import LoginPageGoldstar


#Testing the different login credentials using Parameteriza
tion of GOLDSTAR
@pytest.mark.parametrize("useremail,userpassword",[
    ("johndoe01@gmail.com","Nepali@123"),
    ("johndoe01@gmail.com","nepali@@123"),
    ("johndai@gmail.com","Nepali@123"),
    ("","")
])
def test_login_goldstar(driver, useremail,userpassword):
    goldstar_login = LoginPageGoldstar(driver)
    goldstar_login.open_page("https://www.goldstarshoes.co
m/login")
    driver.maximize_window()
    time.sleep(1)
    goldstar_login.enter_username(useremail)
    time.sleep(1)
    goldstar_login.enter_password(userpassword)
    time.sleep(1)
    goldstar_login.click_login()
    time.sleep(3)
```

Authentication Testing of Goldstar:

💡 **Test ID**: test_0004

Objective: Verify that counter user can login with valid username and password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Login Link
4. Move to Login Page
5. Fill the customer login portal with valid user id and password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should be able to redirected to the dashboard page

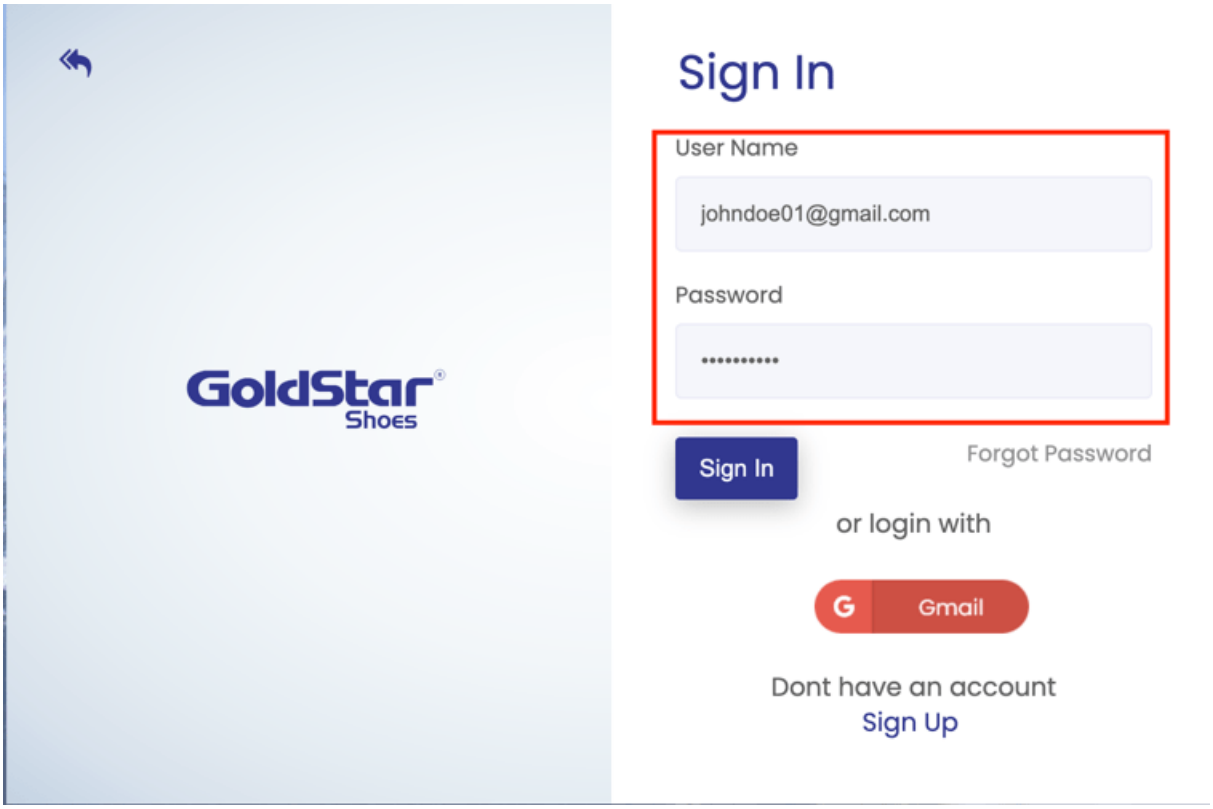
Test Data:

1. URL: https://www.goldstarshoes.com/
2. Username : johndoe01i@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking the login authentication with valid username and password and being redirected to the dashboard

💡 **Test ID**: test_0005

Objective: Verify that counter user cannot login with valid username and invalid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Login Link
4. Move to Login Page
5. Fill the customer login portal with valid user id and password
6. Click on Sign In button
7. The website should display "Credentials Mismatch" message

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should not be able to access the dashboard page

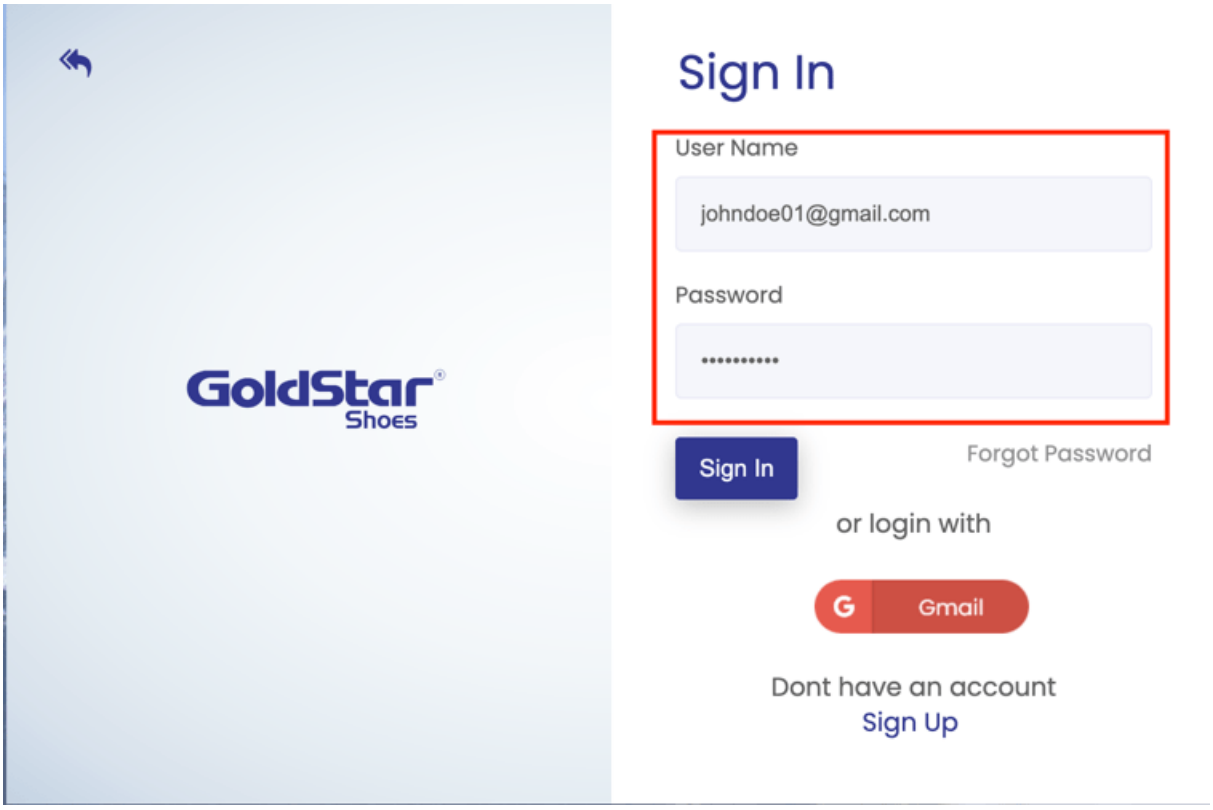
Test Data:

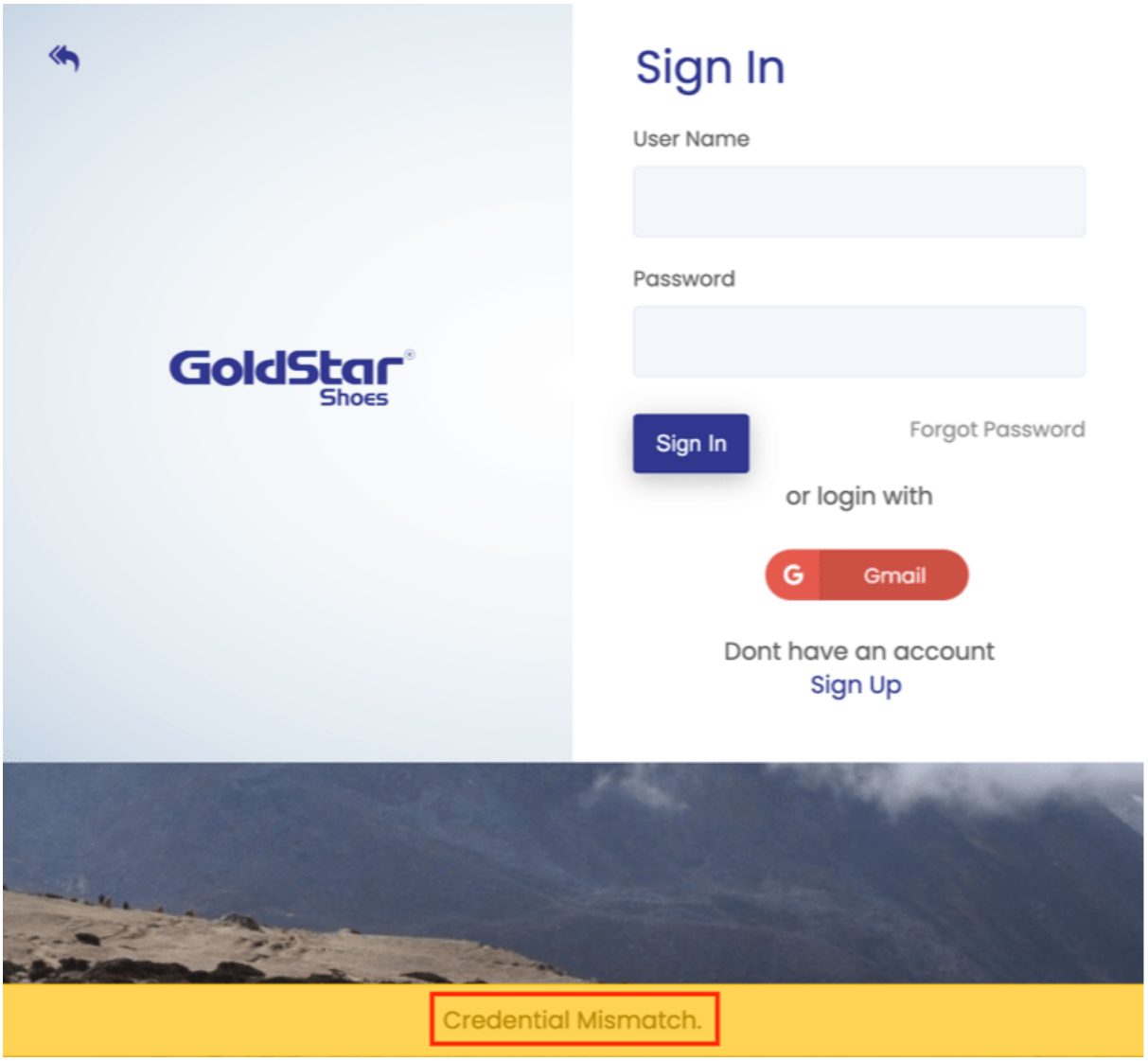
1. URL: https://www.goldstarshoes.com/
2. Username : johndoe01@gmail.com
3. Password : nepali@@123

Actual Result: As per expected

Test Status: Pass

Screenshot:


GoldStar
Shoes
Sign In
User Name
johndoe01@gmail.com
Password
Sign In
Forgot Password
or login with
Gmail
Dont have an account
Sign Up



Checking the login authentication with valid username and invalid password

💡 **Test ID**: test_0006

Objective: Verify that counter user cannot login with invalid username and valid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Login Link
4. Move to Login Page
5. Fill the customer login portal with invalid user id and valid password
6. Click on Sign In button
7. The website should display "Credentials Mismatch" message

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should not be able to access the dashboard page

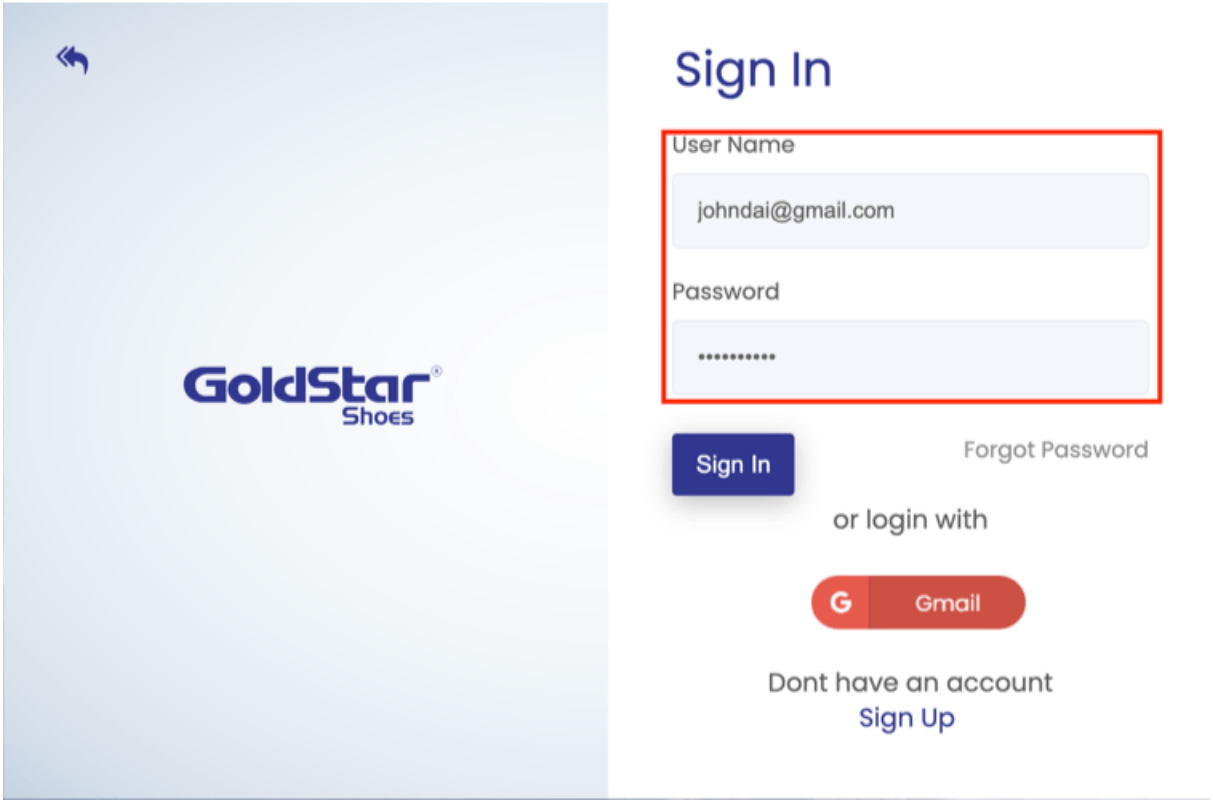
Test Data:

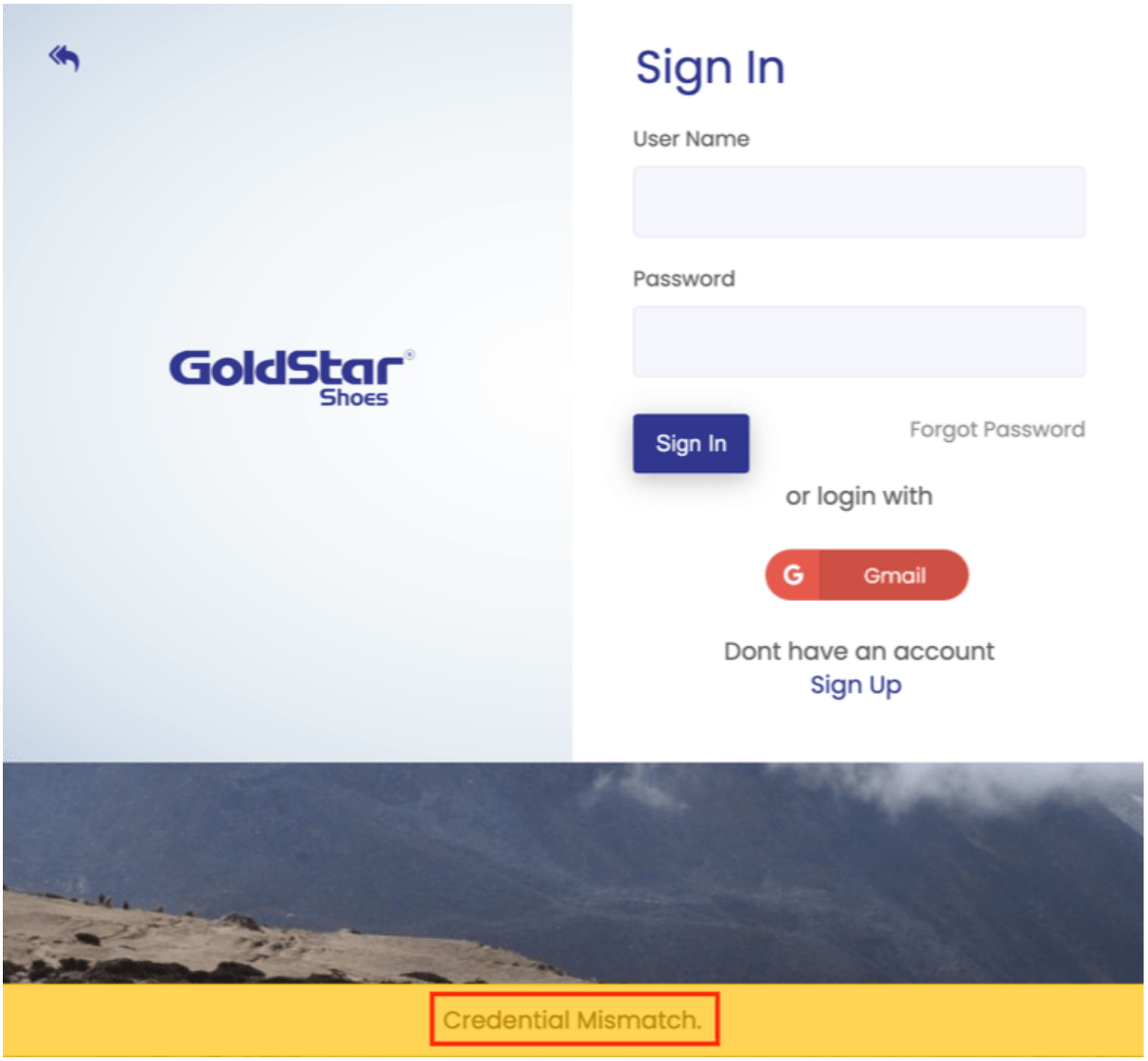
1. URL: https://www.goldstarshoes.com/
2. Username : johndai@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:


GoldStar
Shoes
Sign In
User Name
johndai@gmail.com
Password
Sign In
Forgot Password
or login with
G
Gmail
Dont have an account
Sign Up



Checking the login authentication with invalid username and valid password

💡 **Test ID**: test_0007

Objective: Verify that counter user cannot login with empty credentials

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Set the username and password field empty
6. Click on Sign In button
7. The website should display "Username and password required" message

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to keep the username and password empty in the text box
4. Counter user should be able to press the login button
5. Counter user should not be able to access the dashboard page

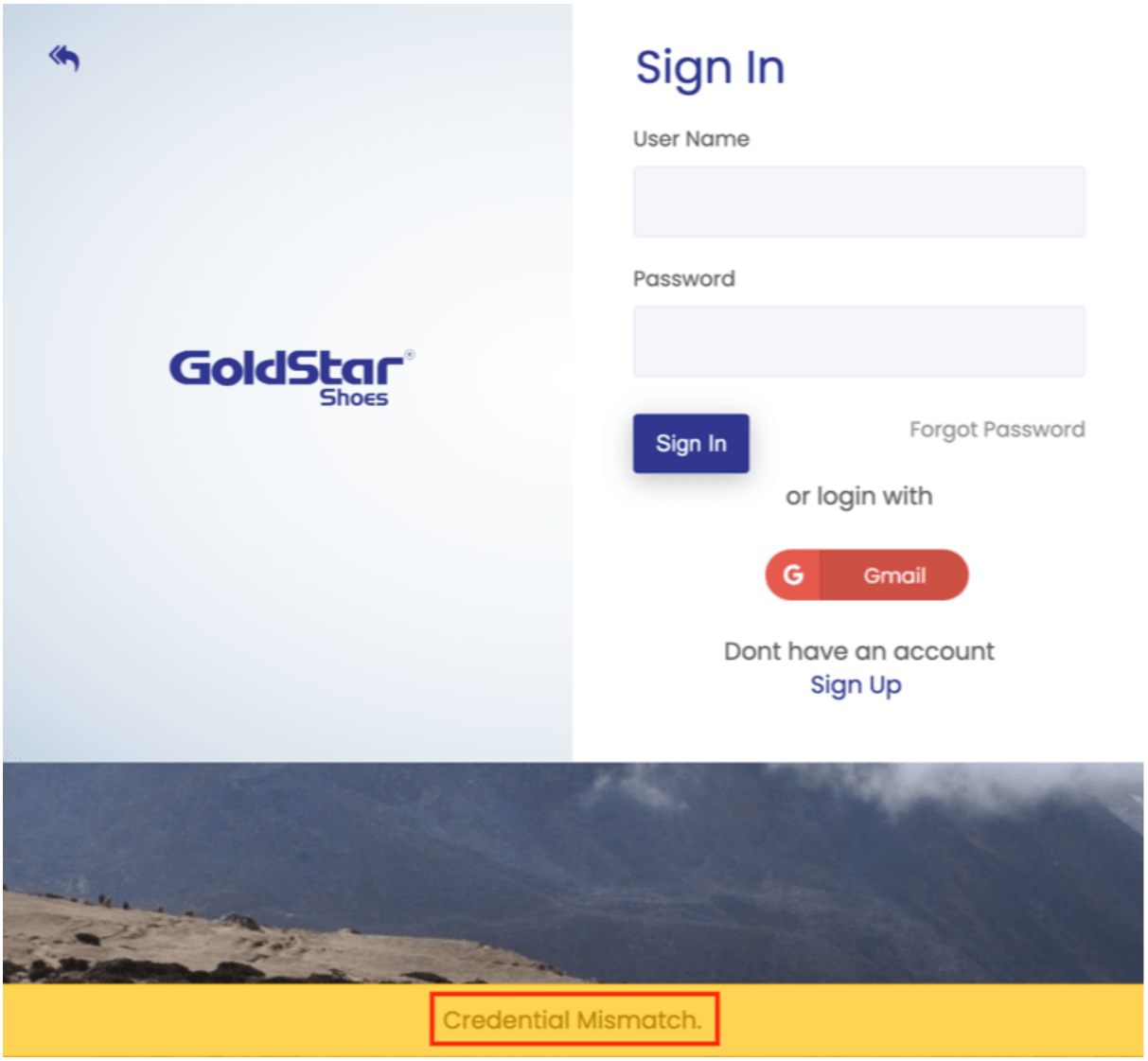
Test Data:

1. URL: https://www.goldstarshoes.com/
2. Username : {empty}
3. Password : {empty}

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking the login authentication with empty username and password

💡 **Test ID**: test_0008

Objective: Verify that Logged-in counter user can log out successfully

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Set the username and password field empty
6. Click on Sign In button
7. The website will not be redirected to the home page
8. Click on the logout button visible on the home page
9. The website should redirect you the Homepage(login page)

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should be able to redirected to the homepage
6. Counter user should be able to view the Logout Button
7. Counter user should be able to click on the logout button
8. Counter user should be redirected to the login page.

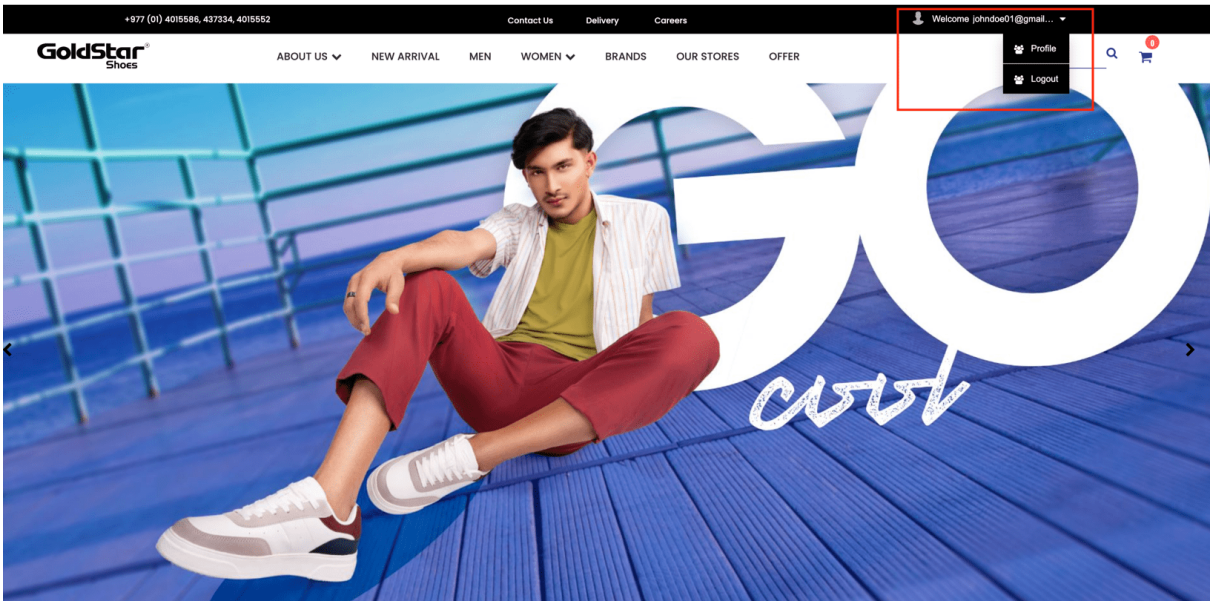
Test Data:

1. URL: https://www.goldstarshoes.com/
2. Username : johndoe01@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:



Verify logout button; The Button is un-clickable, hence failing the test

---

Muncha.com

Registration Page Object

File: `pages/muncha/registration_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium.webdriver.common.by import By
import re
from selenium.webdriver.support.ui import Select

#DEFINE CLASS AS Registration FOR REGISTRATION PAGE OF MUNC
HA

class RegistrationPageMuncha:
    def __init__(self,driver):
        self.driver = driver
```

```
        self.full_name_field = By.XPATH,"//input[@id='Nam
e']"
        self.email_field = By.XPATH,"//input[@id='Email']"
        self.contact_field = By.XPATH,"//input[@id='Contact
No']"
        self.password_field = By.XPATH,"//input[@id='Passwo
rd']"
        self.confirm_password_field = By.XPATH,"//input[@id
='ConfirmPassword']"
        self.hear_about_us_field = (By.ID,'Referer')
        self.newsletter_checklist = By.XPATH,"//input[@id
='accept-email']"
        self.sign_up_button = By.XPATH,"//button[@title='Cr
eate an Account']"


    def open_page(self,url):
        self.driver.get(url)
    def enter_fullname(self,fullname):
        self.driver.find_element(*self.full_name_field).sen
d_keys(fullname)

    def enter_email(self, email):
        self.driver.find_element(*self.email_field).send_ke
ys(email)

    def enter_contact(self, contact):
        self.driver.find_element(*self.contact_field).send_
keys(contact)

    def enter_password(self,password):
        self.driver.find_element(*self.password_field).send
_keys(password)

    def enter_confirm_password(self,confirm_password):
        self.driver.find_element(*self.confirm_password_fie
ld).send_keys(confirm_password)
```

```
    def click_hearabout(self):
        referer_dropdown = self.driver.find_element(*self.h
ear_about_us_field)
        select = Select(referer_dropdown)
        select.select_by_index(1)


    def click_newsletter(self):
        self.driver.find_element(*self.newsletter_checklis
t).click()

    def click_signup(self):
        self.driver.find_element(*self.sign_up_button).clic
k()

    def is_valid_email(self, email):
        # check the format using Regular Expression(re)
        email_regex = r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a
-zA-Z0-9-.]+$"
        return re.match(email_regex, email) is not None
    def is_valid_phone(self, phone):
        return bool(re.match(r'^\d{10}$', phone))
```

Test Case Implementation

File: `run_test1.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Ch
romeService
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.Muncha.registration_page import RegistrationPage
Muncha
```

```
@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER AS driver
    driver = webdriver.Chrome(service=ChromeService(ChromeD
riverManager().install()))
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()

#RUNNING TEST FUNCTION ON MUNCHA REGISTRATION PAGE
def test_registration_muncha(driver):
    email = "johndoe@gmail.com"
    phone = "9810120230"
    muncha_registration = RegistrationPageMuncha(driver)
    muncha_registration.open_page("https://www.muncha.com/A
P/Register")
    driver.maximize_window()
    time.sleep(2)
    muncha_registration.enter_fullname("John Doe")
    time.sleep(1)
    muncha_registration.enter_email(email)
    time.sleep(1)
    muncha_registration.enter_contact(phone)
    time.sleep(1)
    muncha_registration.enter_password("Nepali@123")
    time.sleep(1)
    muncha_registration.enter_confirm_password("Nepali@12
3")
    time.sleep(1)
    muncha_registration.click_hearabout()
    time.sleep(1)
    muncha_registration.click_newsletter()
    time.sleep(1)
    # muncha_registration.click_signup()
    # time.sleep(3)
```

```
    # check the validity of the email
    if muncha_registration.is_valid_email(email):
        print(f"The given email: {email} is valid")
    else:
        print(f"The given email: {email} is invalid")


    # check the validity of the email
    if muncha_registration.is_valid_phone(phone):
        print(f"The given email: {phone} is valid")
    else:
        print(f"The given email: {phone} is invalid")
```

Authorization Testing of Muncha:

💡 **Test ID**: test_0009

Objective: Verify that counter user can register an account

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on User Icon
4. Move to Login Page
5. Click on Sign up Link
6. Fill the registration portal with valid user phone number along with the other details
7. Click on Sign up Button
8. The website should be redirected to the dashboard page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Sign Up Page
5. Counter user should be redirected to the Dashboard Page

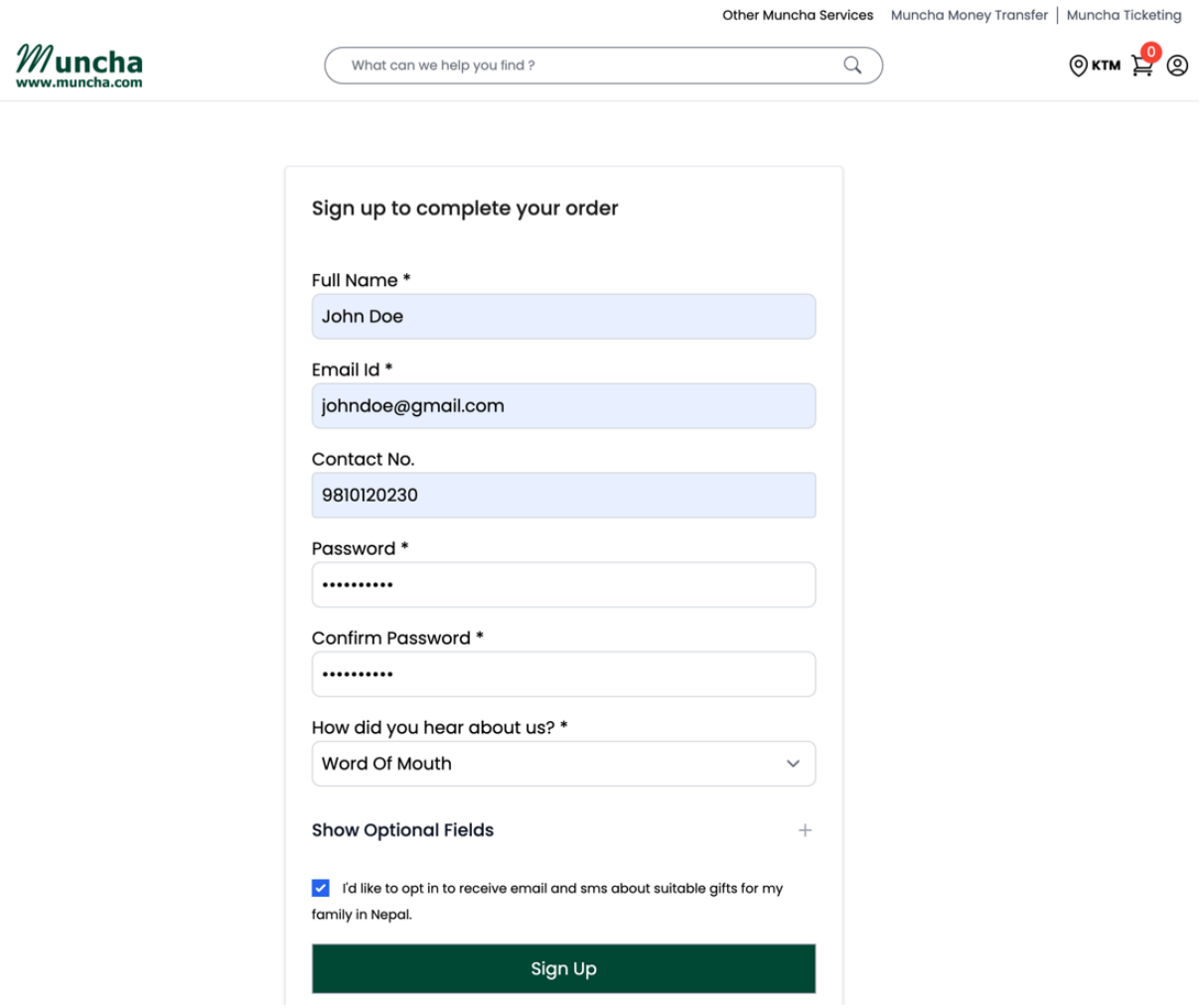
Test Data:

1. URL: https://www.muncha.com/
2. Full Name: John Doe
3. Email: johndoe@gmail.com
4. Contact: 9810120230
5. Password: Nepali@123
6. Confirm Password: Nepali@123
7. How did you head about us: Word of Mouth
8. News letter: accept

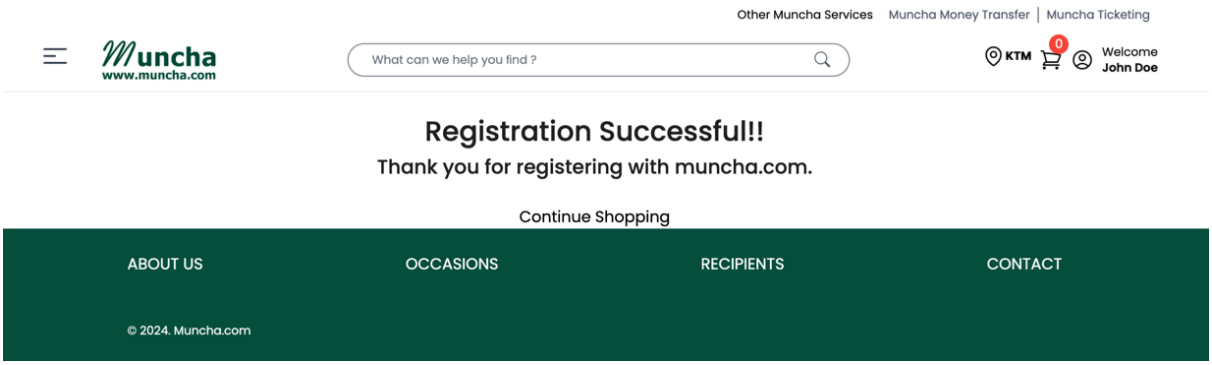
Actual Result: As per expected

Test Status: Pass

Screenshot:



User Input in the Registration Page of Muncha



Counter user being successfully redirected to the dashboard page of Muncha after registration

Validation Testing of Muncha:

💡 **Test ID**: test_0010

Objective: Verify that email and phone is validated correctly with correct user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on User Icon
4. After clicking on the user icon, the Login page is opened.
5. Click on Sign up link
6. Fill the registration portal with valid email and user phone number along with the other details
7. Click on Sign up Button
8. The console will display message "The given input is valid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Sign Up Button
5. Valid Email and Phone Message should be displayed in the console

Test Data:

1. URL: https://www.muncha.com/
2. Email: johndoe@gmail.com
3. Phone: 9810120230

Actual Result: As per expected

Test Status: Pass

Screenshot:

Checking validation of an email and phone in registration page with correct email and phone pattern

💡 **Test ID**: test_0011

Objective: Verify that Invalid email message is displayed with incorrect user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on User Icon
4. After clicking on the user icon, the Login page is opened.
5. Click on Sign up link
6. Fill the registration portal with invalid email and user phone number along with the other details
7. Click on Sign up Button
8. The console will display message "The given input invalid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Sign Up Button
5. Invalid Email and Phone Message should be displayed in the console

Test Data:

1. URL: https://www.muncha.com/
2. Email: johndoe#gmail.com
3. Phone: 9867546

Actual Result: As per expected

Test Status: Pass

Screenshot:

Checking validation of an email and phone in registration page with incorrect email and phone pattern

Login Page Object

File: `pages/muncha/login_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium.webdriver.common.by import By

#DEFINE CLASS AS Login FOR THE LOGIN PAGE OF MUNCHA
class LoginPageMuncha:
    def __init__(self,driver):
        self.driver = driver
        self.email_field = By.XPATH,"//input[@id='Usernam
e']"
        self.user_password_field = By.XPATH,"//input[@id='P
assword']"
        self.signup_button = By.XPATH,"//button[@class='bg-
secondary border border-gray-400 px-4 py-2.5 text-md w-full
text-white']"

    def open_page(self,url):
        self.driver.get(url)

    def enter_userid(self,email_id):
        self.driver.find_element(*self.email_field).send_ke
ys(email_id)

    def enter_user_password(self,password):
        self.driver.find_element(*self.user_password_fiel
d).send_keys(password)
```

```
    def click_signup(self):
        self.driver.find_element(*self.signup_button).click
()
```

Test Case Implementation

File: `run_test1.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Ch
romeService
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.Muncha.login_page import LoginPageMuncha


@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER AS driver
    driver = webdriver.Chrome(service=ChromeService(ChromeD
riverManager().install()))
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()

#RUNNING TEST FUNCTION ON MUNCHA LOGIN PAGE
def test_login_muncha(driver, useremail,userpassword):
    muncha_login = LoginPageMuncha(driver)
    muncha_login.open_page("https://www.muncha.com/ap/logi
n")
    driver.maximize_window()
    time.sleep(2)
    muncha_login.enter_userid(useremail)
```

```
    time.sleep(1)
    muncha_login.enter_user_password(userpassword)
    time.sleep(1)
    muncha_login.click_signup()
    time.sleep(3)
```

Authentication Testing of Muncha

💡 **Test ID**: test_0012

Objective: Verify that counter user can login with valid username and password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Fill the customer login portal with valid user id and password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should be able to redirected to the homepage

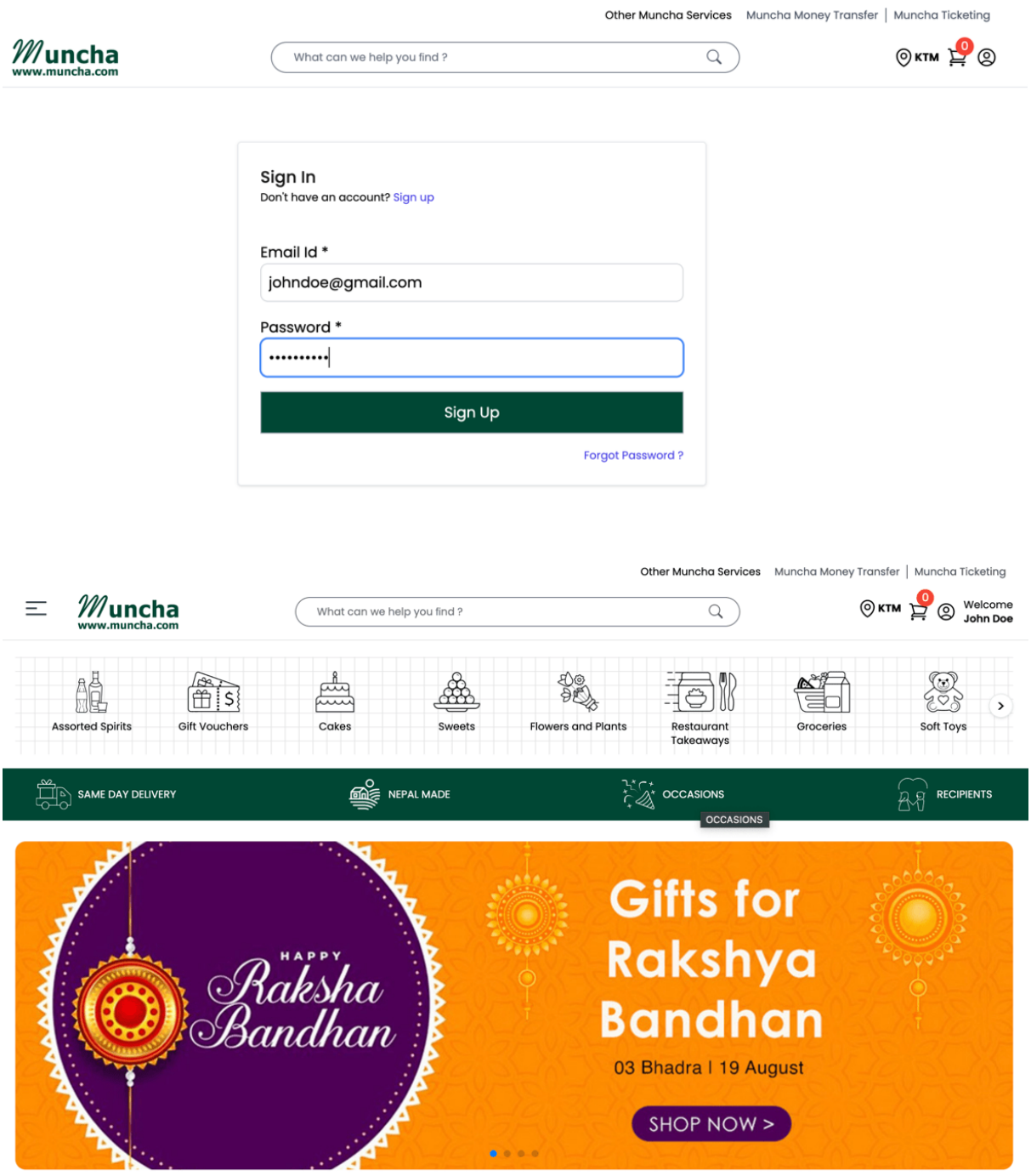
Test Data:

1. URL: https://www.muncha.com/
2. Username : johndoe@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking the login authentication with valid username and password and being redirected to the dashboard

💡 **Test ID**: test_0013

Objective: Verify that counter user cannot login with valid username and invalid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Fill the customer login portal with valid user id and invalid password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the login button
5. The website should display Incorrect username or password message

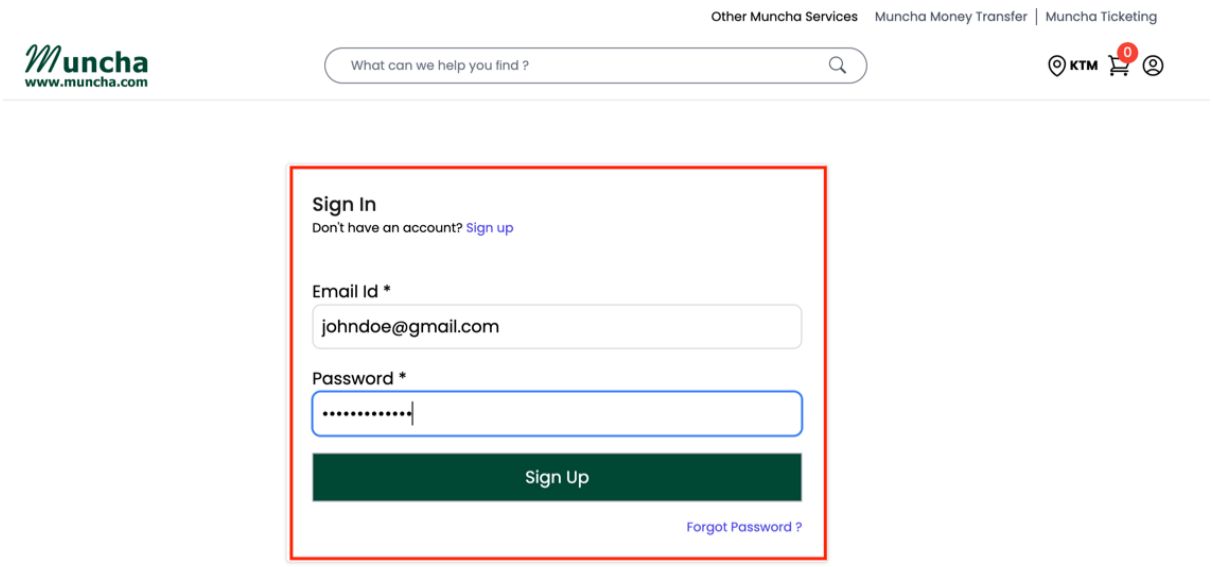
Test Data:

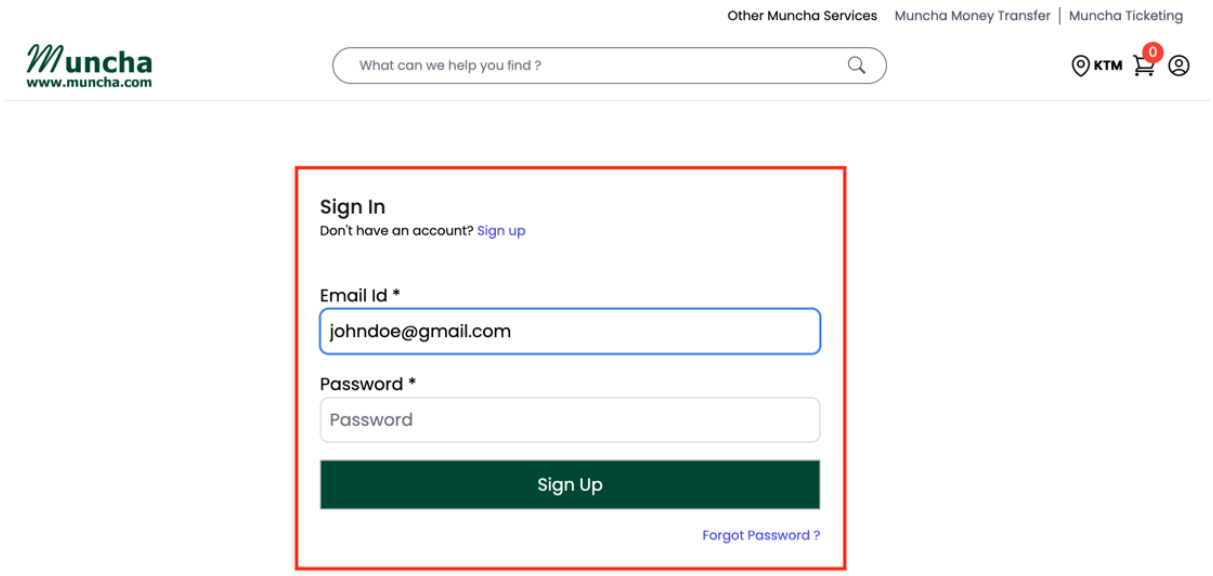
1. URL: https://www.muncha.com/
2. Username : johndoe@gmai.com
3. Password : nepali@@123

Actual Result: As per expected

Test Status: Pass

Screenshot:





Checking the login authentication with valid username and invalid password

💡 **Test ID**: test_0014

Objective: Verify that counter user cannot login with invalid username and valid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Fill the customer login portal with invalid user id and valid password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the login button
5. The website should display Incorrect username or password message

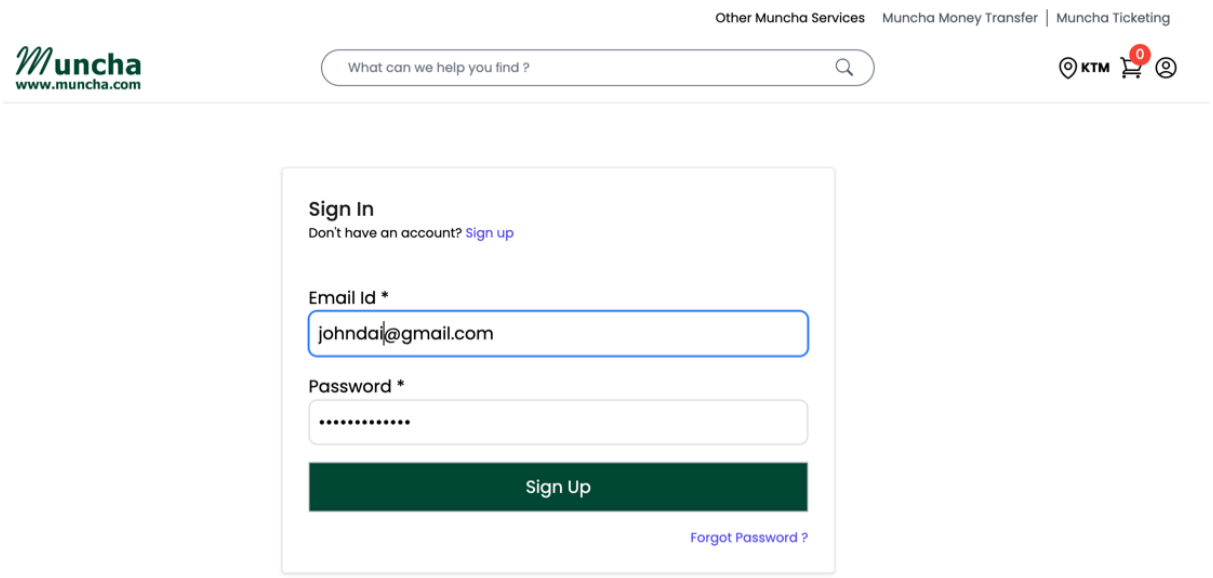
Test Data:

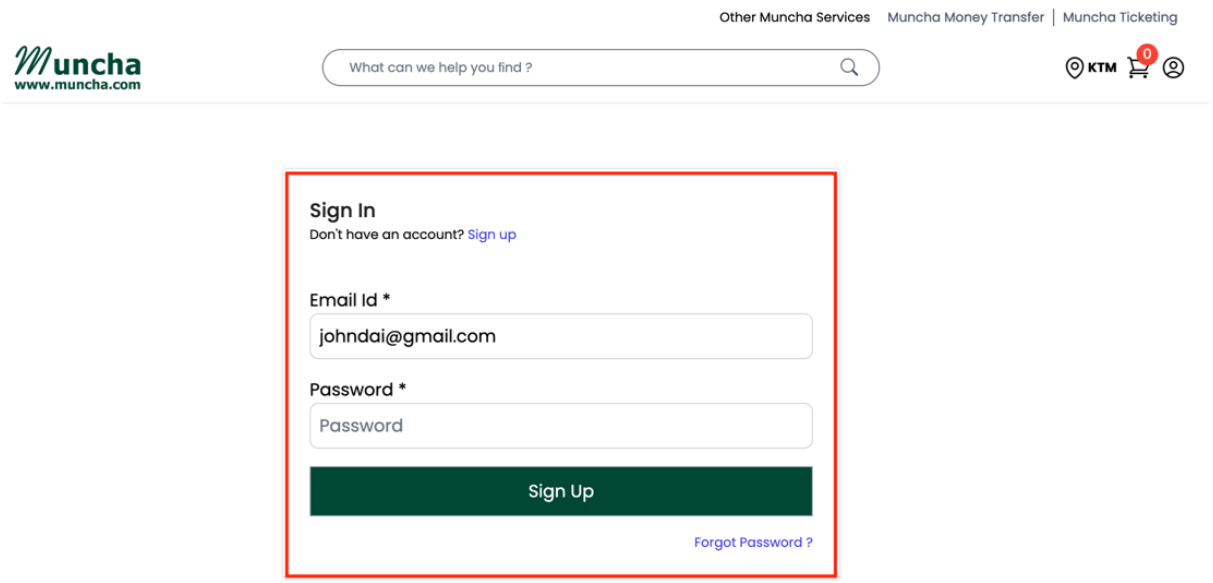
1. URL: https://www.muncha.com/
2. Username : johndai@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:





Checking the login authentication with valid username and invalid password

💡 **Test ID**: test_0015

Objective: Verify that counter user cannot login with empty credentials

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Set the username and password field empty
6. Click on Sign In button
7. The website will not be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the login button
5. The website should display "Username and password required" message

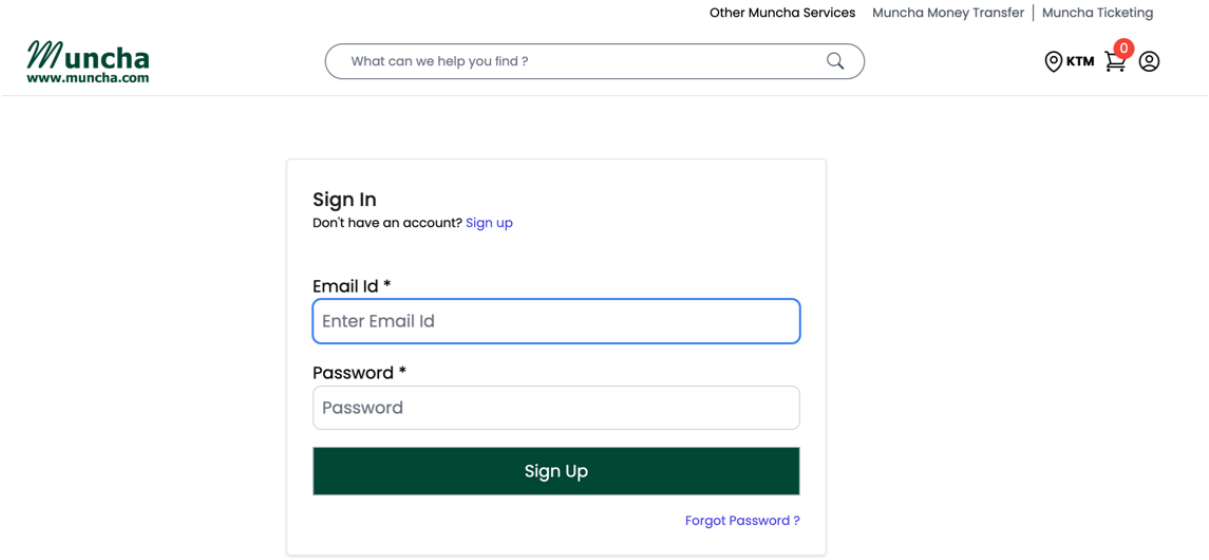
Test Data:

1. URL: https://www.muncha.com/
2. Username : {empty}
3. Password : {empty}

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking the login authentication with empty username and password

💡 **Test ID**: test_00016

Objective: Verify that Logged-in counter user can log out successfully

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Sign In
4. Move to Login Page
5. Set the username and password field empty
6. Click on Sign In button
7. The website will not be redirected to the home page
8. Click on the logout button visible on the homescreen
9. The website should redirect you the Homepage(login page)

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the login button
5. Counter user should be able to redirected to the homepage
6. Counter user should be able to view the Logout Button
7. Counter user should be able to click on the logout button
8. Counter user should be redirected to the login page.

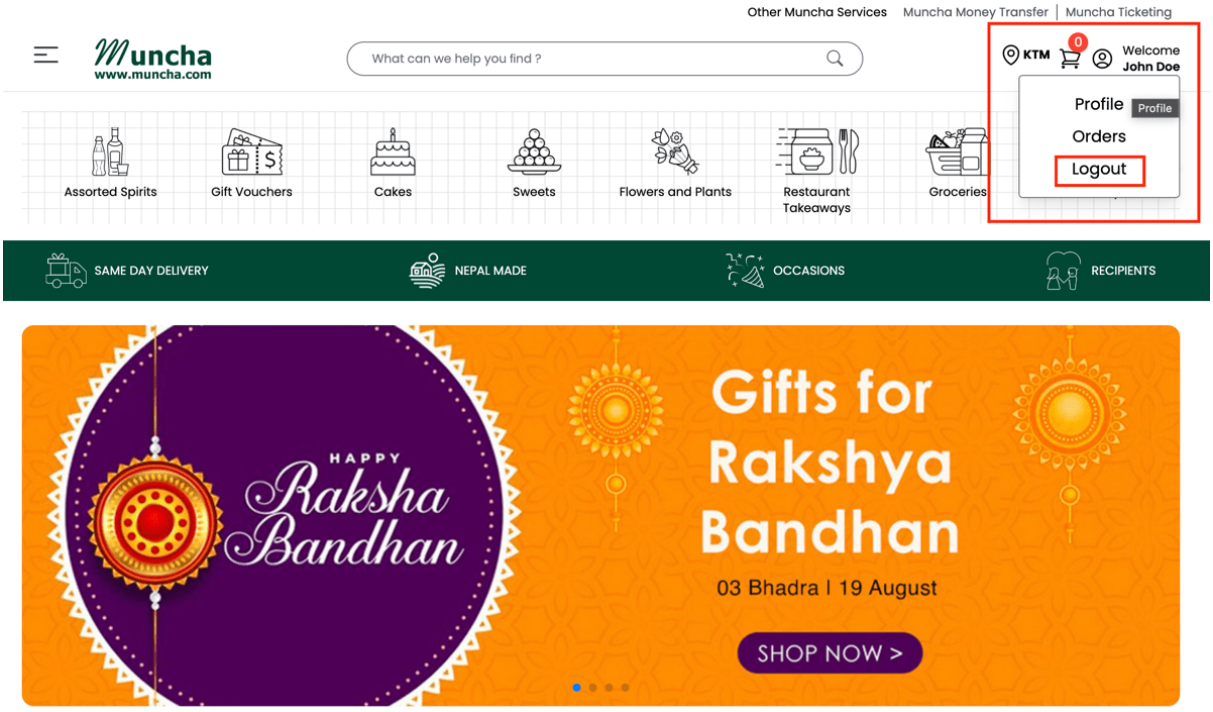
Test Data:

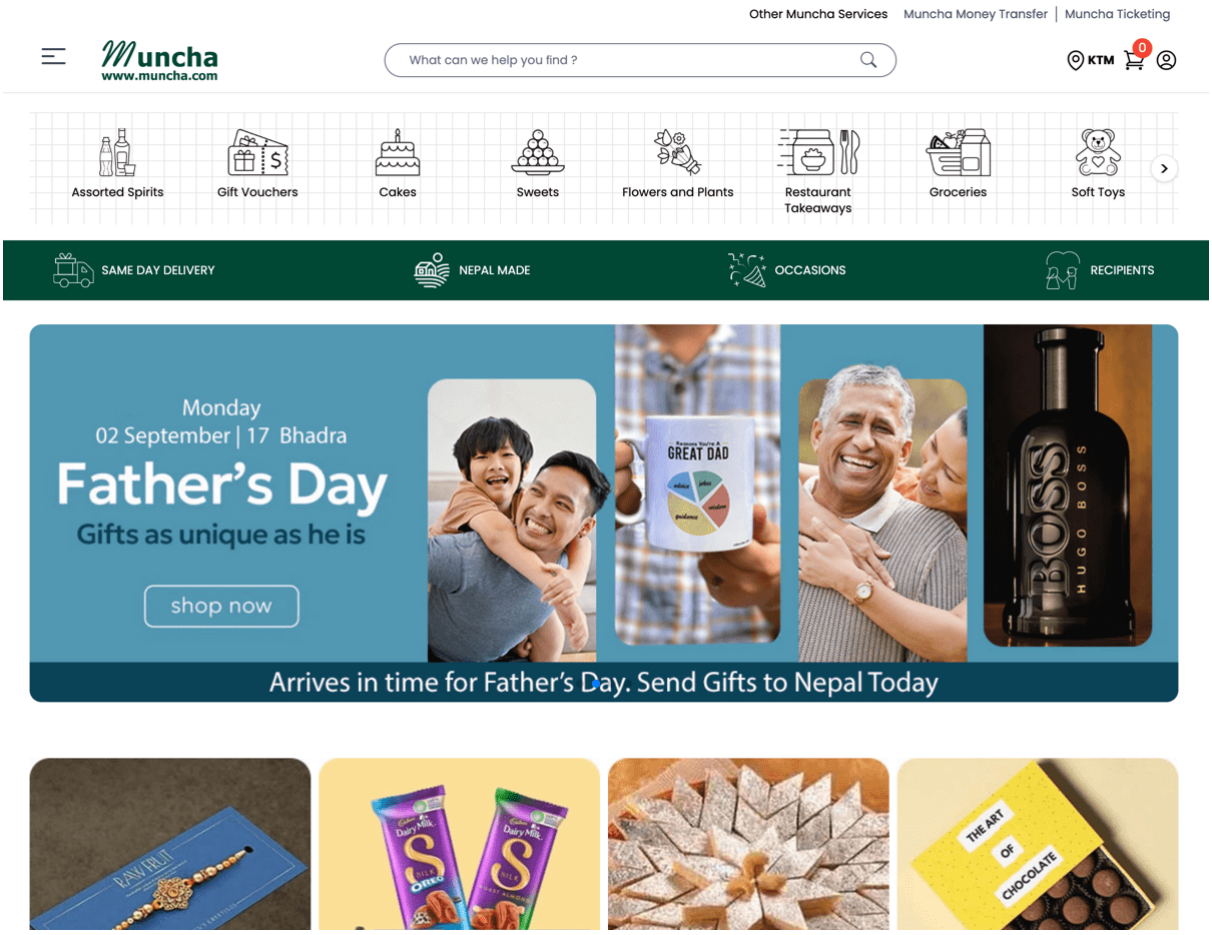
1. URL: https://www.muncha.com/
2. Username : johndoe@gmail.com
3. Password : Nepali@123

Actual Result: The Website fails to logout as the logout is not clickable

Test Status: FAIL

Screenshot:





Verify logout button; After clicking the logout button, The website redirects to the homepage

---

Salt Nepal

Registration Page Object

File: `pages/saltnepal/registration_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium.webdriver.common.by import By
import re

#DEFINE CLASS FOR REGISTRATION PAGE OF SALT NEPAL

class RegistrationPageSalt:
    def __init__(self,driver):
```

```
        self.driver = driver
        self.first_name_field = By.XPATH,"//input[@id='firs
t_name']"
        self.last_name_field = By.XPATH,"//input[@id='last_
name']"
        self.email_field= By.XPATH,"//input[@id='email']"
        self.password_field = By.XPATH,"//input[@id='passwo
rd']"
        self.create_button = By.XPATH,"//input[@value='Crea
te']"

    def open_page(self,url):
        self.driver.get(url)

    def enter_first_name(self,firstname):
        self.driver.find_element(*self.first_name_field).se
nd_keys(firstname)

    def enter_last_name(self,lastname):
        self.driver.find_element(*self.last_name_field).sen
d_keys(lastname)

    def enter_email(self,email):
        self.driver.find_element(*self.email_field).send_ke
ys(email)

    def enter_password(self,password):
        self.driver.find_element(*self.password_field).send
_keys(password)

    def click_create(self):
        self.driver.find_element(*self.create_button).click
()

    def is_valid_email(self, email):
        # check the format using Regular Expression(re)
        email_regex = r"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a
```

```
-zA-Z0-9-.]+$"
        return re.match(email_regex, email) is not None
```

Test Case Implementation

File: `run_test1.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Ch
romeService
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.saltnepal.registration_page import RegistrationP
ageSalt

@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER AS driver
    driver = webdriver.Chrome(service=ChromeService(ChromeD
riverManager().install()))
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()

#RUNNING TEST FUNCTION ON REGISTRATION PAGE OF SALT NEPAL
def test_registration_salt(driver):
    email = "johndoe@gmail.com"
    salt_registration = RegistrationPageSalt(driver)
    salt_registration.open_page("https://saltnp.com/accoun
t/register")
    driver.maximize_window()
    time.sleep(1)
    salt_registration.enter_first_name("John")
```

```
    time.sleep(1)
    salt_registration.enter_last_name("Doe")
    time.sleep(1)
    salt_registration.enter_email(email)
    time.sleep(1)
    salt_registration.enter_password("Nepali@123")
    time.sleep(1)
    salt_registration.click_create()
    time.sleep(1)

    # check the validity of the email
    if salt_registration.is_valid_email(email):
        print(f"The given email: {email} is valid")
    else:
        print(f"The given email: {email} is invalid")
```

Authorization Testing of Salt Nepal:

💡 **Test ID**: test_00017

Objective: Verify that counter user can register an account

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on "Account" Link
4. After navigating to account link, click on CREATE ACCOUNT
5. Move to Registration Page
6. Fill the registration portal with valid email address along with the other details
7. Click on Create Button
8. Your user account has been registered

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Create Page
5. Counter user should be redirected to the Dashboard Page

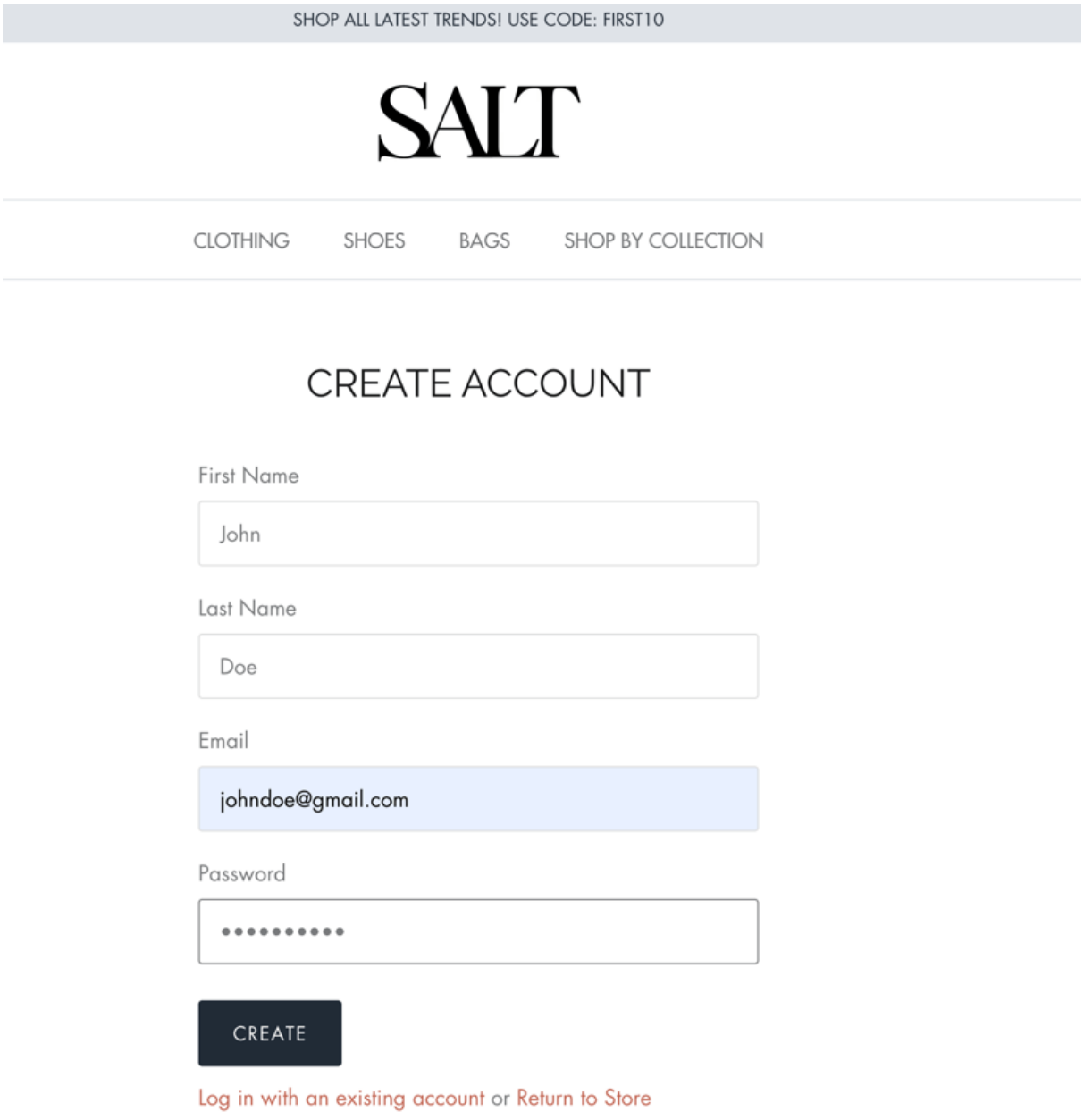
Test Data:

1. URL: https://saltnp.com/
2. First Name: John
3. Last Name: Doe
4. Email: johndoe@gmail.com
5. Password: Nepali@123
6. Confirm Password: Nepali@123
7. Phone: 9765311717

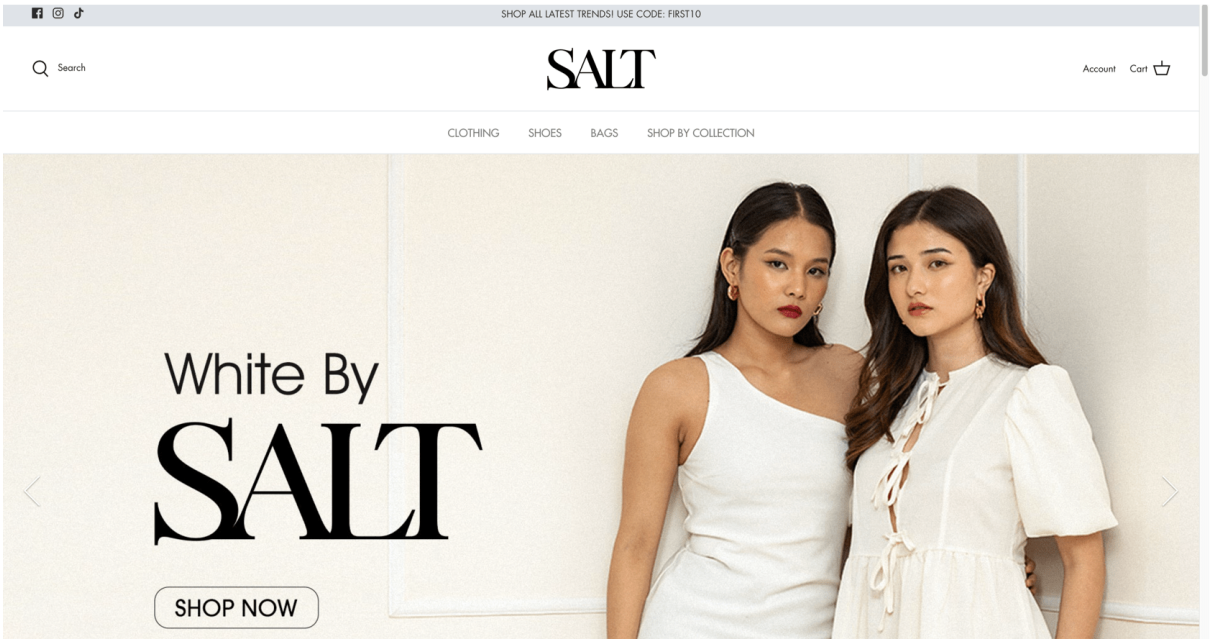
Actual Result: As per expected

Test Status: Pass

Screenshot:



User input for the Registration of Salt Nepal



Counter user being successfully redirected to the dashboard page of Salt Nepal after registration

Validation Testing of Salt Nepal:

💡 **Test ID**: test_00018

Objective: Verify that email and phone is validated correctly with correct user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Create an Account Link
4. Move to Registration Page
5. Fill the registration portal with valid user email pattern along with the other details
6. Click on Register Button
7. The console will display message "The given email is valid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Create An Account Page
5. Valid Email and Phone Message should be displayed in the console

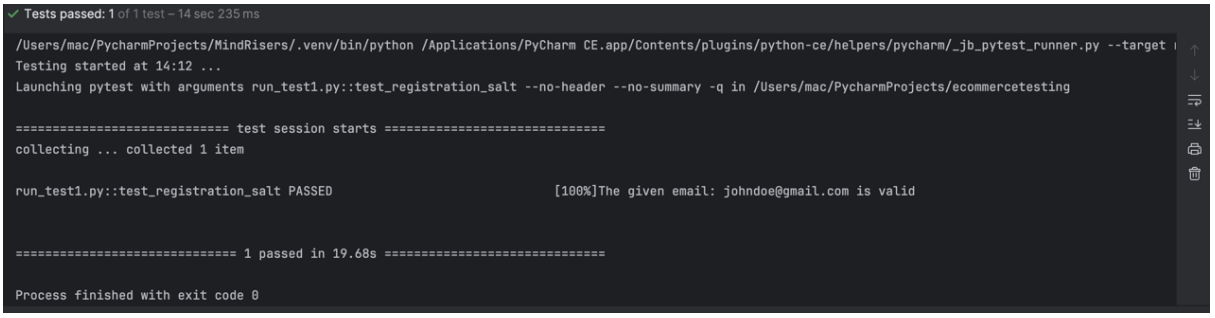
Test Data:

1. URL: https://saltnp.com/
2. Email : johndoe@gmail.com

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking validation of an email in registration page with correct email pattern

💡 **Test ID**: test_0019

Objective: Verify that Invalid email and phone number message is displayed with incorrect user input

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Create an Account Link
4. Move to Registration Page
5. Fill the registration portal with invalid user email pattern along with the other details
6. Click on Register Button
7. The console will display message "The given input is invalid"

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the Registration Page
3. Counter user should be able to fill valid user Input in the text box
4. Counter user should be able to press the Create An Account Page
5. Invalid Email and Phone Message should be displayed in the console

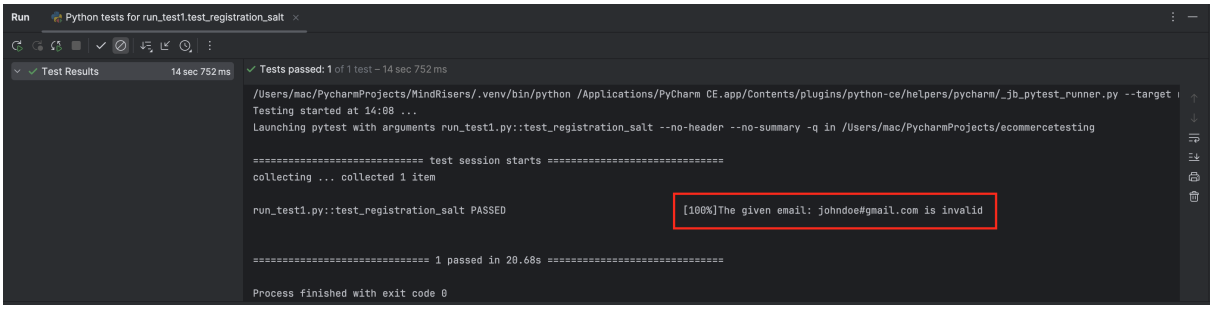
Test Data:

1. URL: https://saltnp.com/
2. Email : johndoe#gmail.com

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking validation of an email in registration page with incorrect email pattern

Login Page Object

File: `pages/saltnepal/login_page.py`

```
#IMPORT ALL THE NECESSARY MODULES
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as Ch
romeService
from webdriver_manager.chrome import ChromeDriverManager
import pytest
import time

#IMPORT THE CLASSES FROM THE PAGES
from pages.saltnepal.login_page import LoginPageSalt

@pytest.fixture()
def driver():
    #SETTING UP THE CHROME DRIVER MANAGER AS driver
    driver = webdriver.Chrome(service=ChromeService(ChromeD
riverManager().install()))
    driver.implicitly_wait(2)
    #YIELD THE DRIVER
    yield driver
    #CLOSE THE DRIVER INSTANCE
    driver.quit()

#Testing the different login credentials using Parameteriza
tion of Salt Nepal
@pytest.mark.parametrize("useremail,userpassword",[
    ("johndoe@gmail.com","Nepali@123"),
    ("johndoe@gmail.com","invalid@123"),
```

```
    ("johndai@gmail.com","Nepali@123"),
    ("","")
])

#RUNNING TEST FUNCTION ON SOCHEKO REGISTRATION PAGE
def test_login_salt(driver,useremail,userpassword):
    salt_login = LoginPageSalt(driver)
    salt_login.open_page("https://saltnp.com/account/logi
n")
    driver.maximize_window()
    time.sleep(1)
    salt_login.enter_useremail(useremail)
    time.sleep(1)
    salt_login.enter_password(userpassword)
    time.sleep(1)
    salt_login.click_signin()
    time.sleep(3)

    # check if the username or password is incorrect
    try:
        alert = driver.switch_to.alert
        alert_text = alert.text
        assert "Invalid username or password" in alert_text
        print(f"Invalid username or password for {useremai
l}")

        # check if the username and password is correct
    except:
        time.sleep(2)
        page_source = driver.page_source
        if "Welcome to the Dashboard" in page_source:
            print(f"Valid Username or Password for user: {u
seremail}")
        else:
            print("Unexpected Error!! Please try again late
r!")
```

Test Case Implementation

Authentication Testing of Salt Nepal:

💡 **Test ID**: test_0020

Objective: Verify that counter user can login with valid username and password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Account Link
4. Move to Login Page
5. Fill the customer login portal with valid user id and valid password
6. Click on Sign In button
7. The website will be redirected to the Account Profile

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the Sign In button
5. Counter user should be able to redirected to the Account Profile

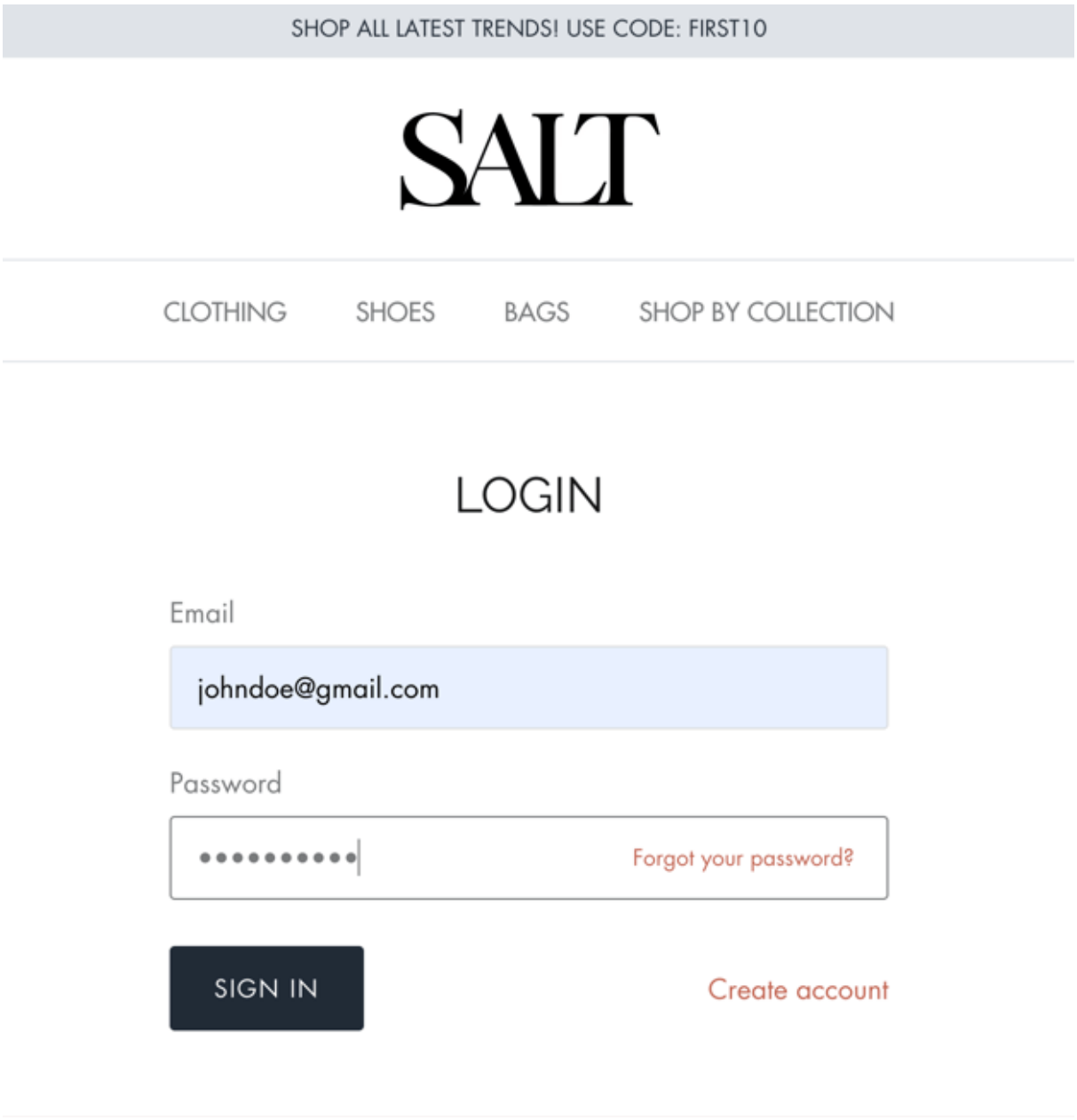
Test Data:

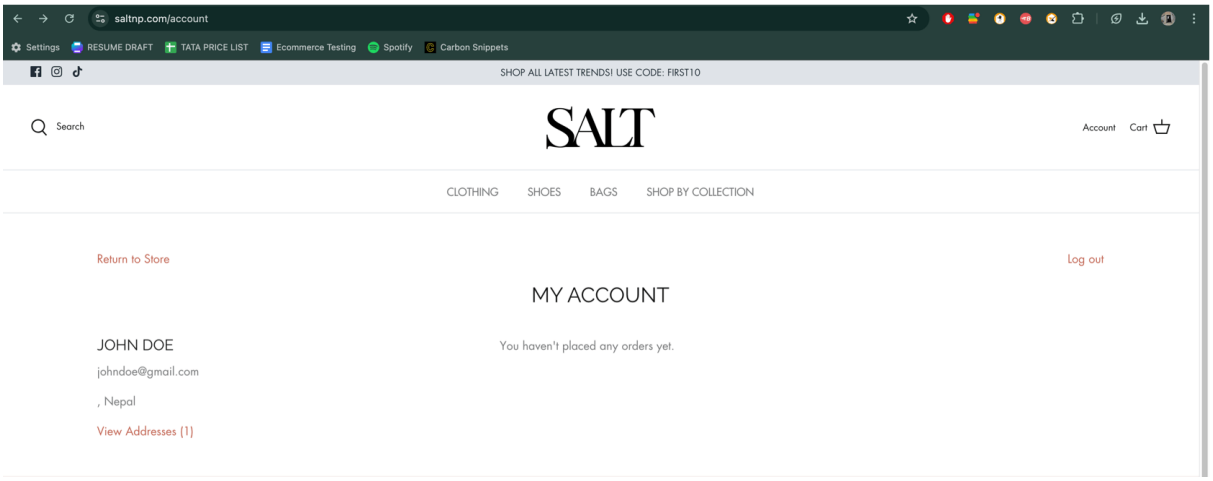
1. URL: https://saltnp.com/
2. Username : johndoe@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:


SHOP ALL LATEST TRENDS! USE CODE: FIRST10
SALT
CLOTHING
SHOES
BAGS
SHOP BY COLLECTION
LOGIN
Email
johndoe@gmail.com
Password
Forgot your password?
SIGN IN
Create account



Checking the login authentication with valid username and password and being redirected to the dashboard

💡 **Test ID**: test_0021

Objective: Verify that counter user cannot login with valid username and valid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Account Link
4. Move to Login Page
5. Fill the customer login portal with valid user id and invalid password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the Sign In button
5. The website should display Incorrect username or password message

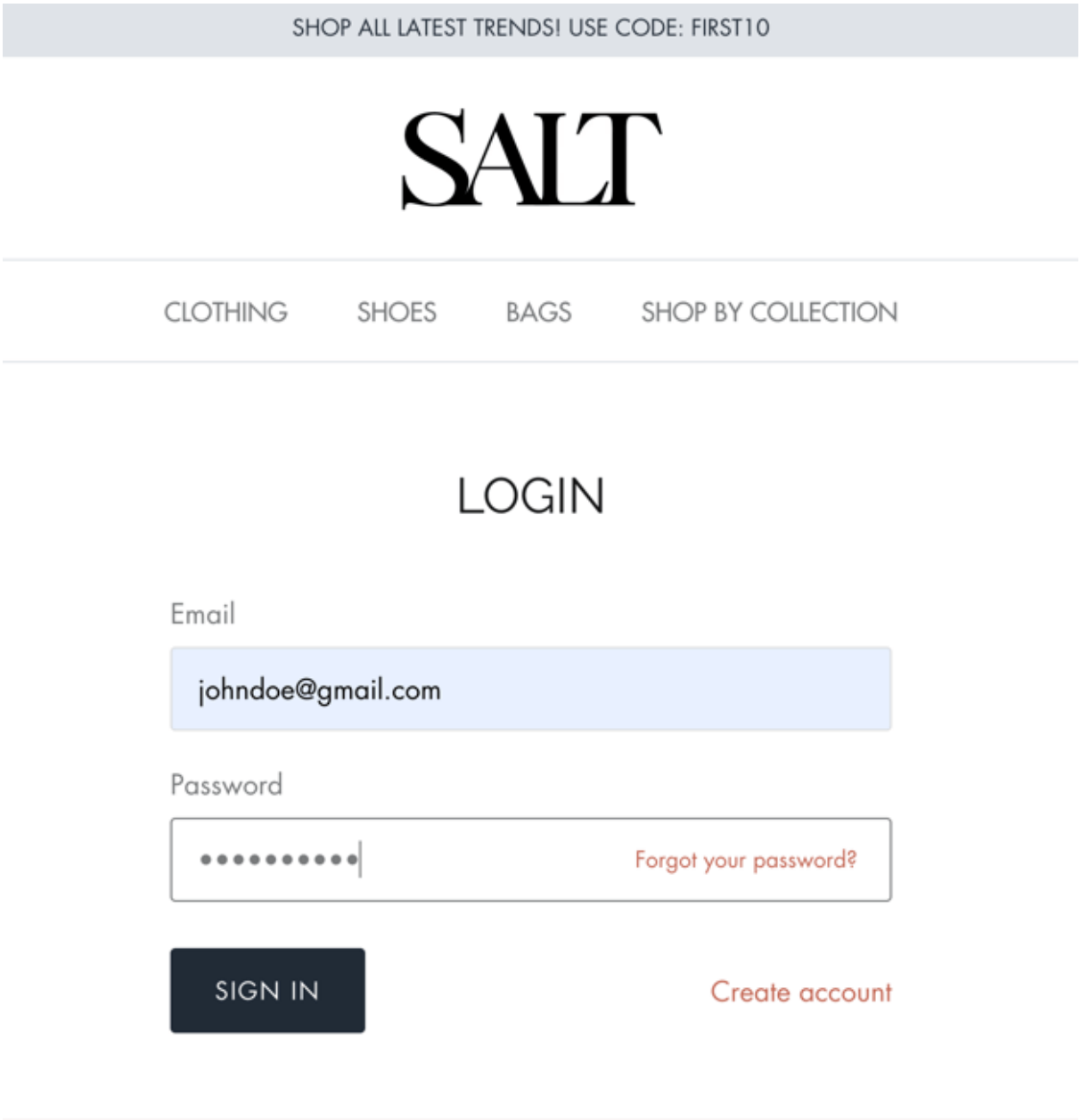
Test Data:

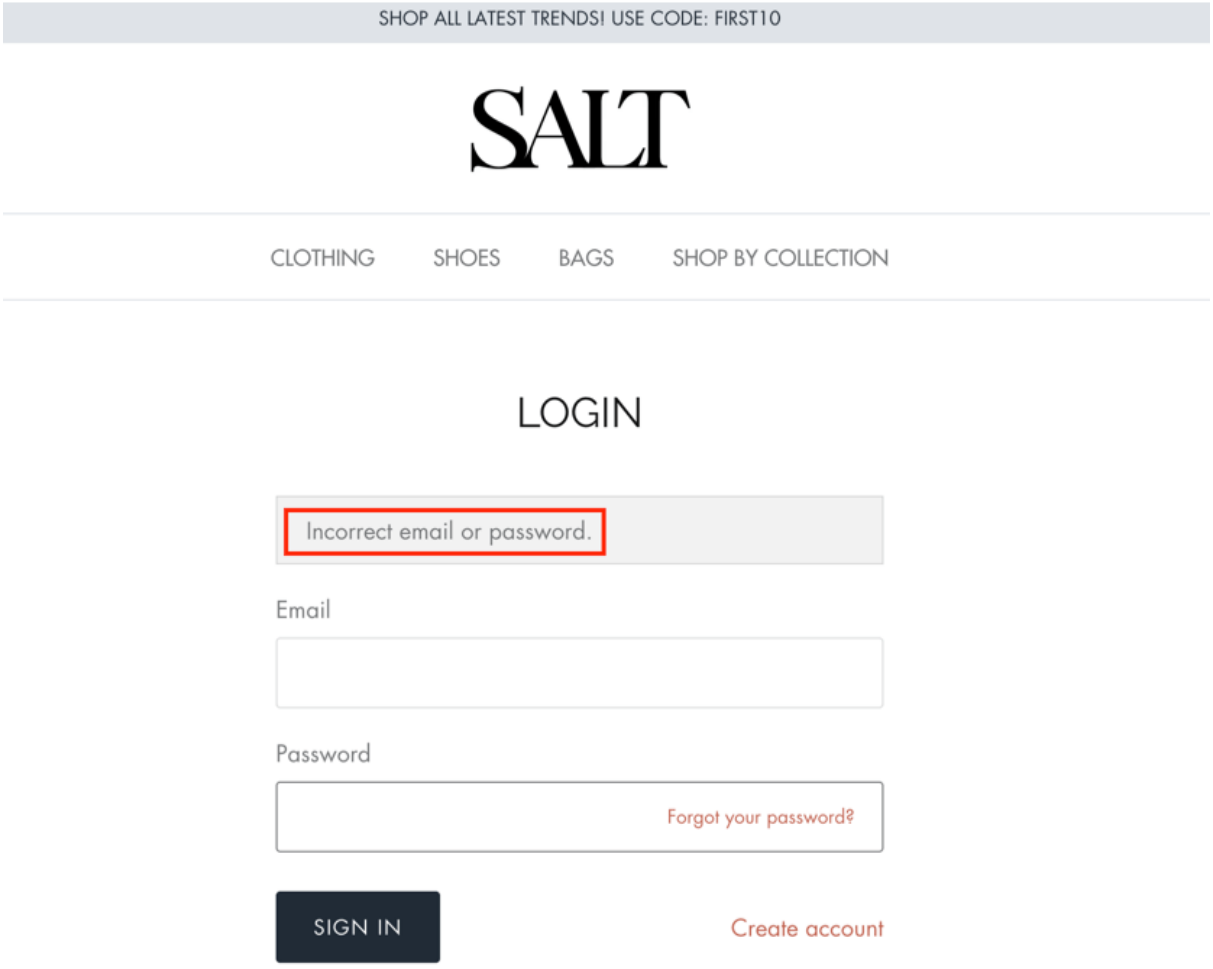
1. URL: https://saltnp.com/
2. Username : johndoe@gmail.com
3. Password : nepali@@123

Actual Result: As per expected

Test Status: Pass

Screenshot:


SHOP ALL LATEST TRENDS! USE CODE: FIRST10
SALT
CLOTHING
SHOES
BAGS
SHOP BY COLLECTION
LOGIN
Email
johndoe@gmail.com
Password
Forgot your password?
SIGN IN
Create account



Checking the login authentication with valid username and invalid password

💡 **Test ID**: test_0022

Objective: Verify that counter user cannot login with invalid username and valid password

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Account Link
4. Move to Login Page
5. Fill the customer login portal with invalid user id and valid password
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill username and password in the text box
4. Counter user should be able to press the Sign In button
5. he website should display Incorrect username or password message

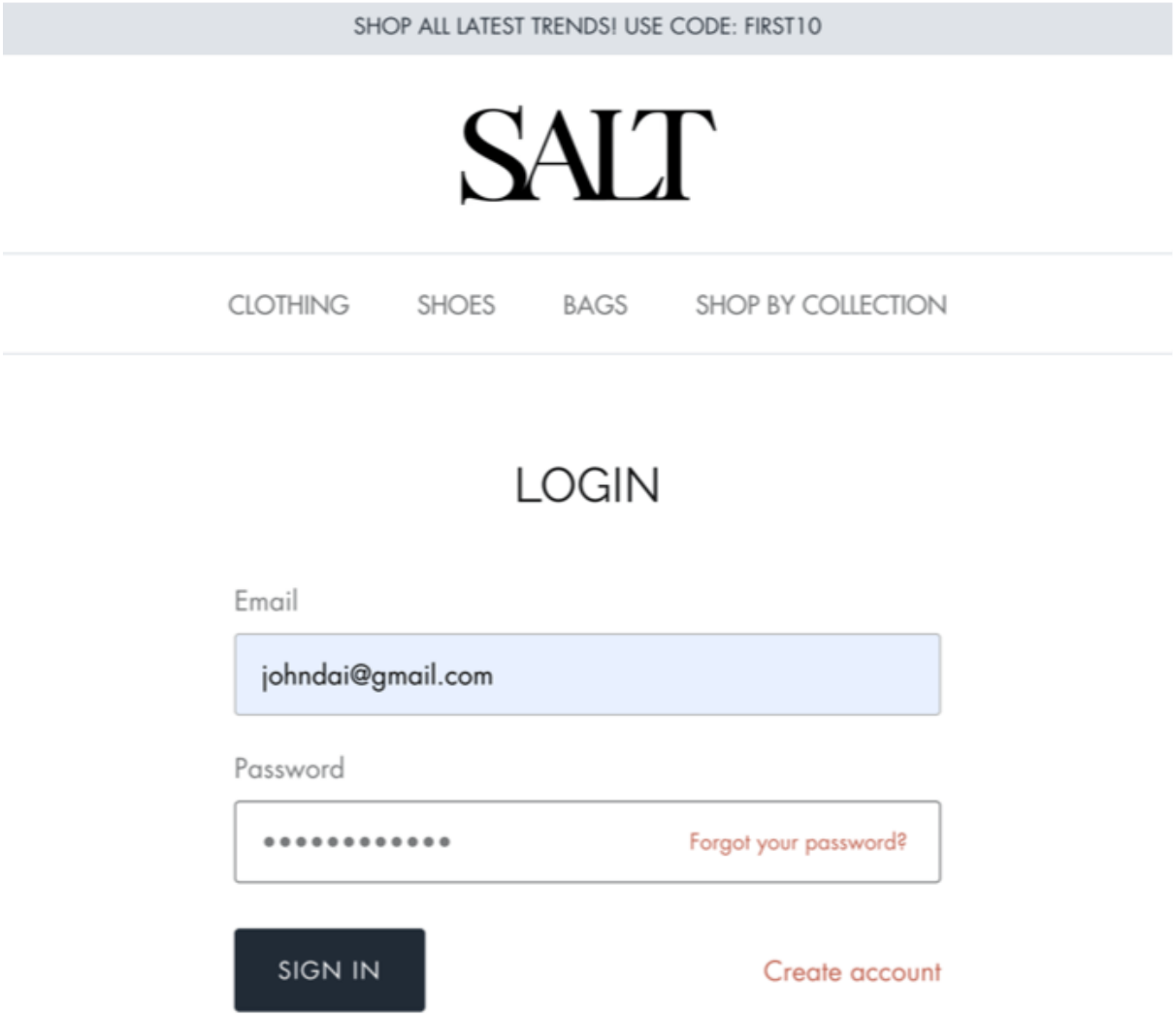
Test Data:

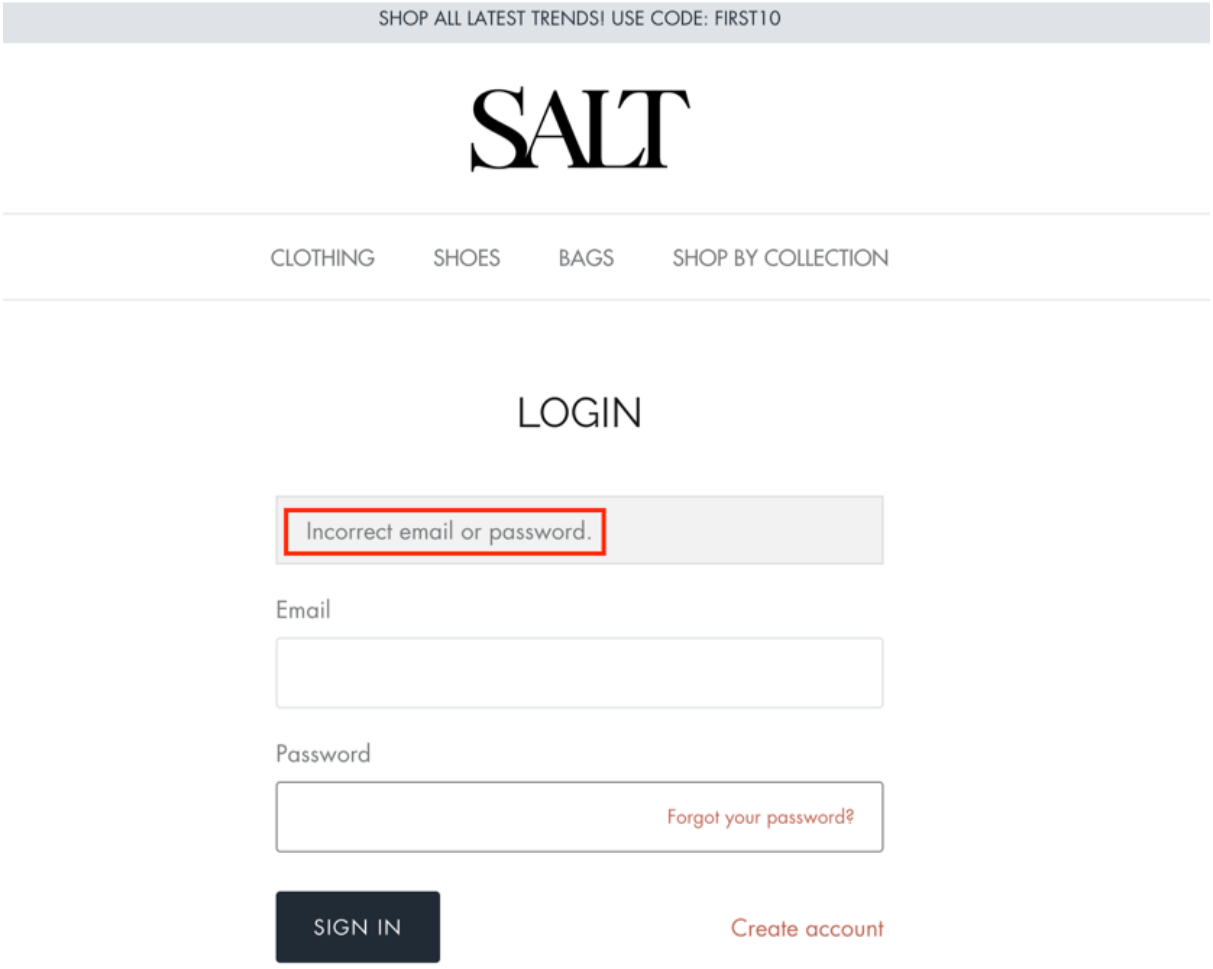
1. URL: https://saltnp.com/
2. Username : johndai@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:


SHOP ALL LATEST TRENDS! USE CODE: FIRST10
SALT
CLOTHING
SHOES
BAGS
SHOP BY COLLECTION
LOGIN
Email
johndai@gmail.com
Password
Forgot your password?
SIGN IN
Create account



Checking the login authentication with invalid username and valid password

💡 **Test ID**: test_0023

Objective: Verify that counter user cannot login with empty credentials

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Account Link
4. Move to Login Page
5. Set the username and password field empty
6. Click on Sign In button
7. The website will be redirected to the home page

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to set empty username and password
4. Counter user should be able to press the Sign In button
5. The website should display "Username and password required" message

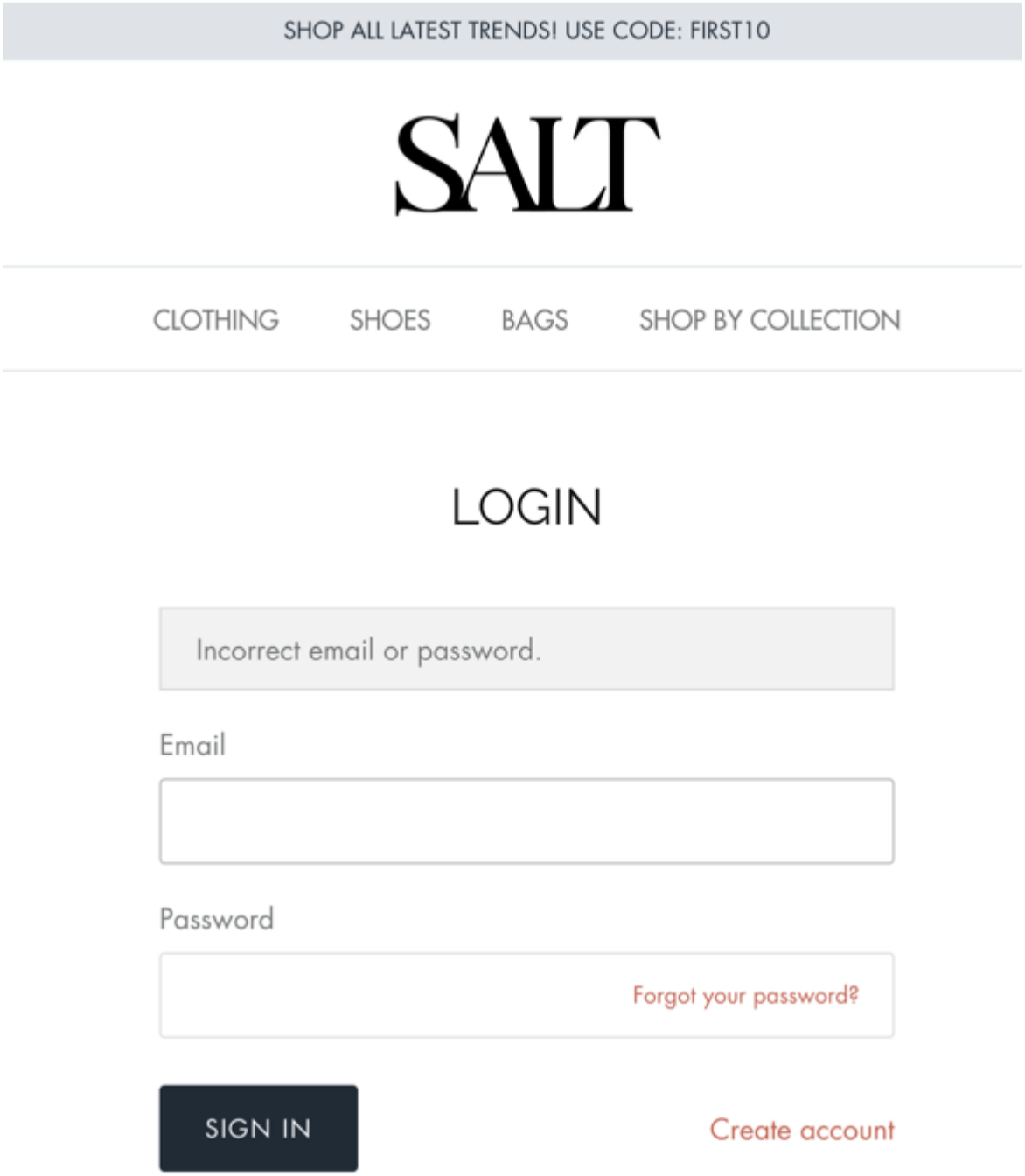
Test Data:

1. URL: https://saltnp.com/
2. Username : {empty}
3. Password : {empty}

Actual Result: As per expected

Test Status: Pass

Screenshot:



Checking the login authentication with empty username and password

💡 **Test ID**: test_0024

Objective: Verify that Logged-in counter user can log out successfully

Test Steps:

1. Launch the website URL
2. Navigate to homepage
3. Click on Account Link
4. Move to Login Page
5. Fill the customer login portal with valid user id and valid password
6. Click on Sign In button
7. The website will be redirected to the Account Profile page
8. Click on the logout button visible on the current page
9. The website should redirect you the Homepage

Expected Result:

1. Counter user should be able to launch the website URL
2. Counter user should be able to navigate to the login page
3. Counter user should be able to fill their username and password in the text box
4. Counter user should be able to press the Sign In button
5. Counter user should be able to redirected to the Account Profile Page
6. Counter user should be able to view the Logout Button
7. Counter user should be able to click on the logout button
8. Counter user should be redirected to the Home page.

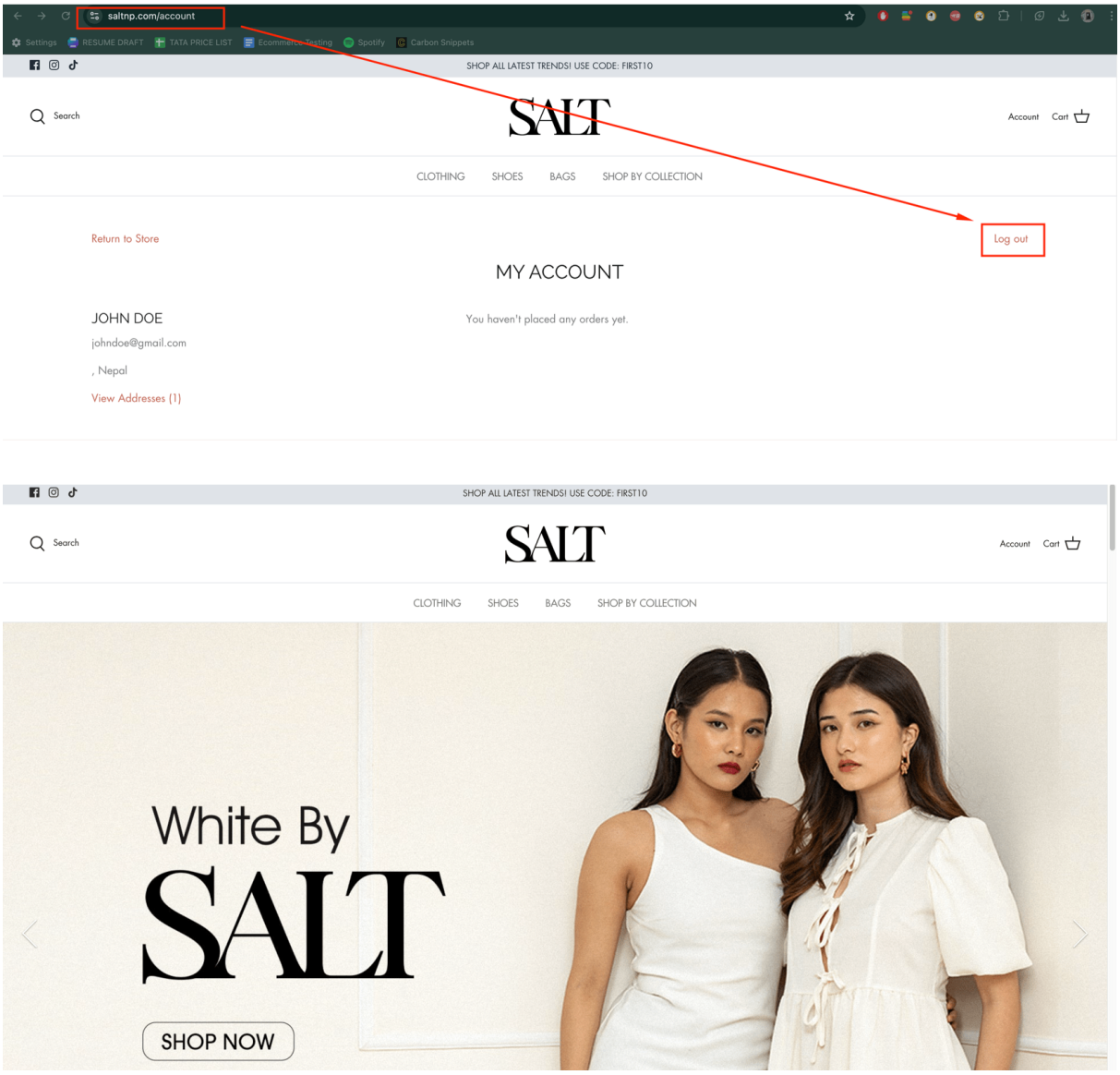
Test Data:

1. URL: https://saltnp.com/
2. Username : johndoe@gmail.com
3. Password : Nepali@123

Actual Result: As per expected

Test Status: Pass

Screenshot:



Verify logout button; After clicking the logout button, The website redirects to the homepage

Conclusion

This document provides a detailed guide for testing the Authorization, Validation, and Authentication testing of an e-commerce web application using Selenium and the Page Object Model (POM) in Python. The implementation includes test cases with respective code snippets and screenshots of the test

executions. By following this approach, you can ensure that your application is secure, functional, and user-friendly.