**AVIVO TASK**

TASK 1 : Test Case Design (Manual Testing)

Scenario: A login page with the following fields: Username (Email) Password Login Button "Forgot Password" link

Task: Write test cases covering positive and negative scenarios. Include both functional and UI validation test cases.

**Functional Test case positive scenario**

| Test case id | Description | Test steps | Expected Result |
| --- | --- | --- | --- |
| TC\_FP\_01 | To check whether user can able to the Login with valid email and password | 1. Enter the correct email 2. Enter the correct password 3. Click the login button | * User can able to login to the portal * User can able to view the toaster message like successfully login * User can able to redirect to the dashboard page or home page |
| TC\_FP\_02 | To check whether user can able to click the forgot password button | 1. Click the forgot password button | * User can able to navigate to the reset page |

**Functional Test case Negative scenario**

| Test case id | Description | Test Steps | Expected Result |
| --- | --- | --- | --- |
| TC\_FN\_01 | To check whether user can able to enter invalid email and invalid password | 1. Enter the invalid email example: invalid@ 2. Enter the invalid password “ 123 3. Click the login button | * User can able to view the validation message as “Incorrect email or password” |
| TC\_FN\_02 | To check whether user can able to enter the valid email and invalid password | 1. Enter the valid email example: [bala@gmail.com](mailto:bala@gmail.com) 2. Enter the invalid password example: 123 | * User can able to view the validation as “Incorrect email or password” |
| TC\_FN\_03 | To check whether user can able to enter the invalid email and valid password | 1. Enter the invalid email as example: invlaid@ 2. Enter the valid password as Test@123 | * User can able to view the validation as “Incorrect email or password” |
| TC\_FN\_04 | To check whether user can able to leave the email field as empty and enter the valid password | 1. Leave the email field as blank 2. Enter the valid password 3. Click the login button | * User can able to view the error message as “Email is required” |
| TC\_FN\_05 | To check whether user can able to Enter the valid email and Leave the password field as empty | 1. Enter the valid email 2. Leave the password field as blank 3. Click the login button | * User can able to view the error message as “Password is required” |
| TC\_FN\_06 | To check whether user can able to leave the email and password field as blank | 1. Leave the Email field as blank  2. Leave the password field as blank  3. Click the login button | * User can able to view the error message as “ Email is required & Password field is required” |

**UI Validation Test case**

| Test case id | Description | Test steps | Expected result |
| --- | --- | --- | --- |
| TC\_UI\_01 | To check whether login is display as same as the sample screen | Open the web page | Display as same as sample screen |
| TC\_UI\_02 | To check whether user can able to view the Email field and password | Open the web page | Able to view the email field and password field |
| TC\_UI\_03 | To check whether placeholder text is display for email and password | Navigate to the web page | Email filed : Enter the email  Password field : Enter the Password |
| TC\_UI\_04 | To check whether login button able to display the user | Click the login button | Need to show the validation message enter the email or password |

**Section 2: Web UI Test Automation**

**Scenario:** Use a demo login page (e.g., <https://the-internet.herokuapp.com/login>)

**Task:**

1. Create an automation script using **Selenium / Cypress / Playwright** (any preferred framework) to:
   * Open the login page
   * Input username & password
   * Click login and verify success/error message
2. Add assertions for UI element visibility, error messages, and title/url changes.

Explanation: Below code is written in selenium python, Environment is pycharm, three test case has written

1. Valid user name and password / URL checked & Title checked
2. Invalid username
3. Invalid password

import pytest

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

# Setup for Selenium WebDriver

@pytest.fixture

def setup():

driver = webdriver.Chrome()

driver.maximize\_window()

yield driver

driver.quit()

# Positive Test Case: Login

def test\_check\_Login(setup):

driver = setup

driver.get("https://the-internet.herokuapp.com/login") # Sauce Demo URL

driver.find\_element(By.ID, "username").send\_keys("tomsmith")

driver.find\_element(By.ID,"password").send\_keys("SuperSecretPassword!")

driver.find\_element(By.XPATH,"/html/body/div[2]/div/div/form/button/i").click()

time.sleep(3)

url = driver.current\_url

assert url == "https://the-internet.herokuapp.com/secure" # Expected URL after login

title = driver.title

assert title == "The Internet" #Expected Title after login

# Negative Test Case: Invalid login user name

def test\_check\_Invalidemail(setup):

driver = setup

driver.get("https://the-internet.herokuapp.com/login") # Sauce Demo URL

driver.find\_element(By.ID, "username").send\_keys("tomsmit")

driver.find\_element(By.ID,"password").send\_keys("SuperSecretPassword!")

driver.find\_element(By.XPATH,"/html/body/div[2]/div/div/form/button/i").click()

time.sleep(3)

errormessage = driver.find\_element(By.XPATH,"/html//div[@id='flash']")

print(errormessage.text)

# Negative Test Case: Invalid login password

def test\_check\_Invalidpassword(setup):

driver = setup

driver.get("https://the-internet.herokuapp.com/login") # Sauce Demo URL

driver.find\_element(By.ID, "username").send\_keys("tomsmith")

driver.find\_element(By.ID,"password").send\_keys("SuperSecretPassword")

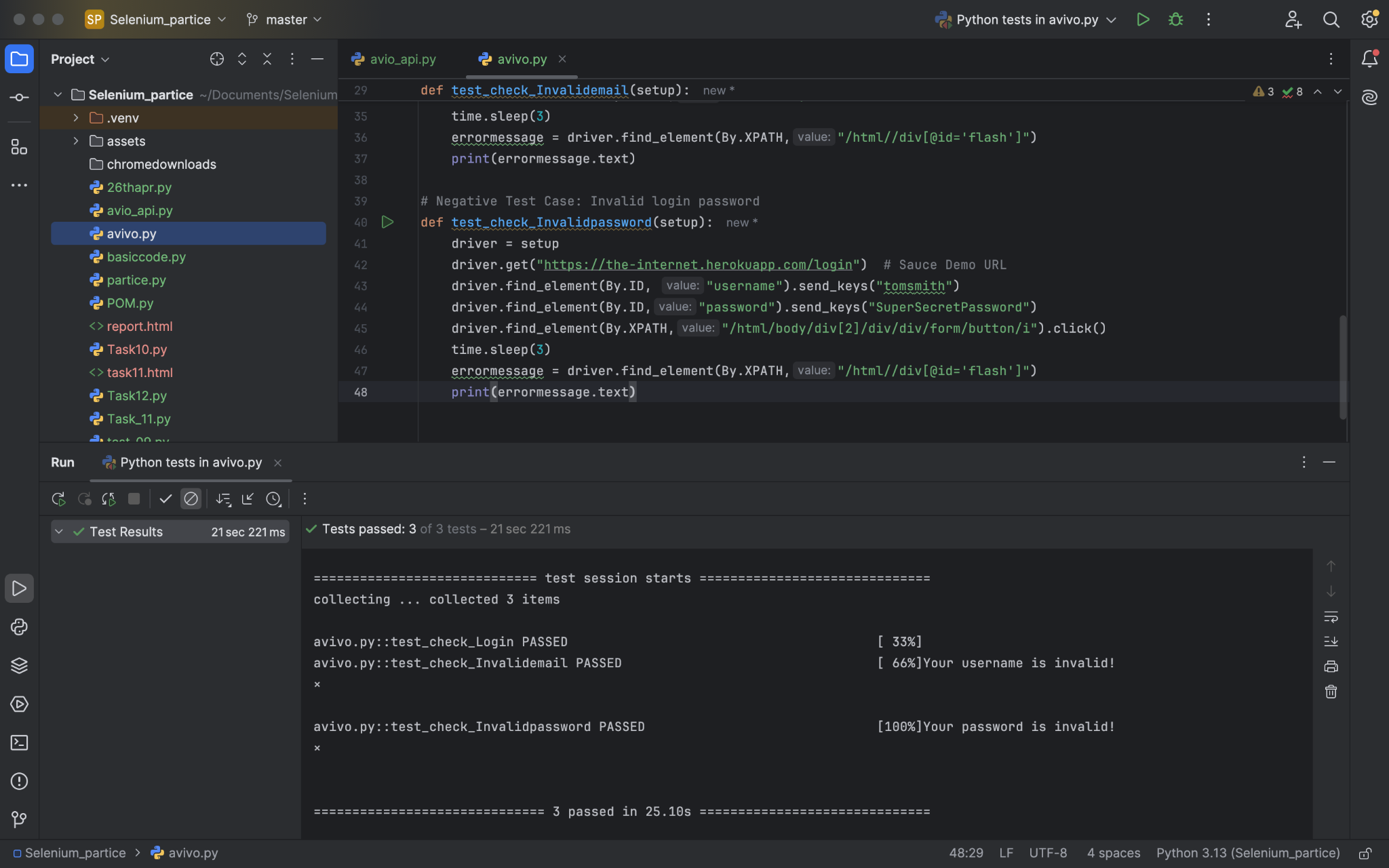
driver.find\_element(By.XPATH,"/html/body/div[2]/div/div/form/button/i").click()

time.sleep(3)

errormessage = driver.find\_element(By.XPATH,"/html//div[@id='flash']")

print(errormessage.text)

Output:



**Section 3: API Testing (Automation)**

**Scenario:** A sample REST API endpoint:

POST /api/login

* Body: { "email": "user@example.com", "password": "123456" }
* Response: 200 OK (on success), 401 Unauthorized (on failure)

**Task:**

1. Write an **automated script** using **Postman (with test scripts)** or a **code-based approach** (e.g., Python + requests / JavaScript + axios / Java + RestAssured) to validate:
   * Successful login
   * Invalid credentials
   * Missing fields
2. Validate response codes and expected keys in response JSON.

Explanation : Below code is written in python, Environment is pycharm, three test case has written

import unittest

import requests

# Updated Base URL

BASE\_URL = "https://reqres.in/api/login"

class TestLoginAPI(unittest.TestCase):

def test\_successful\_login(self):

payload = {

"email": "eve.holt@reqres.in",

"password": "cityslicka"

}

response = requests.post(BASE\_URL, json=payload)

self.assertEqual(response.status\_code, 200)

self.assertIn("token", response.json())

def test\_invalid\_credentials(self):

payload = {

"email": "eve.holt@reqres.in",

"password": "wrongpass"

}

response = requests.post(BASE\_URL, json=payload)

self.assertEqual(response.status\_code, 400)

self.assertIn("error", response.json())

def test\_missing\_email(self):

payload = {

"password": "cityslicka"

}

response = requests.post(BASE\_URL, json=payload)

self.assertEqual(response.status\_code, 400)

self.assertIn("error", response.json())

def test\_missing\_password(self):

payload = {

"email": "eve.holt@reqres.in"

}

response = requests.post(BASE\_URL, json=payload)

self.assertEqual(response.status\_code, 400)

self.assertIn("error", response.json())

def test\_empty\_body(self):

response = requests.post(BASE\_URL, json={})

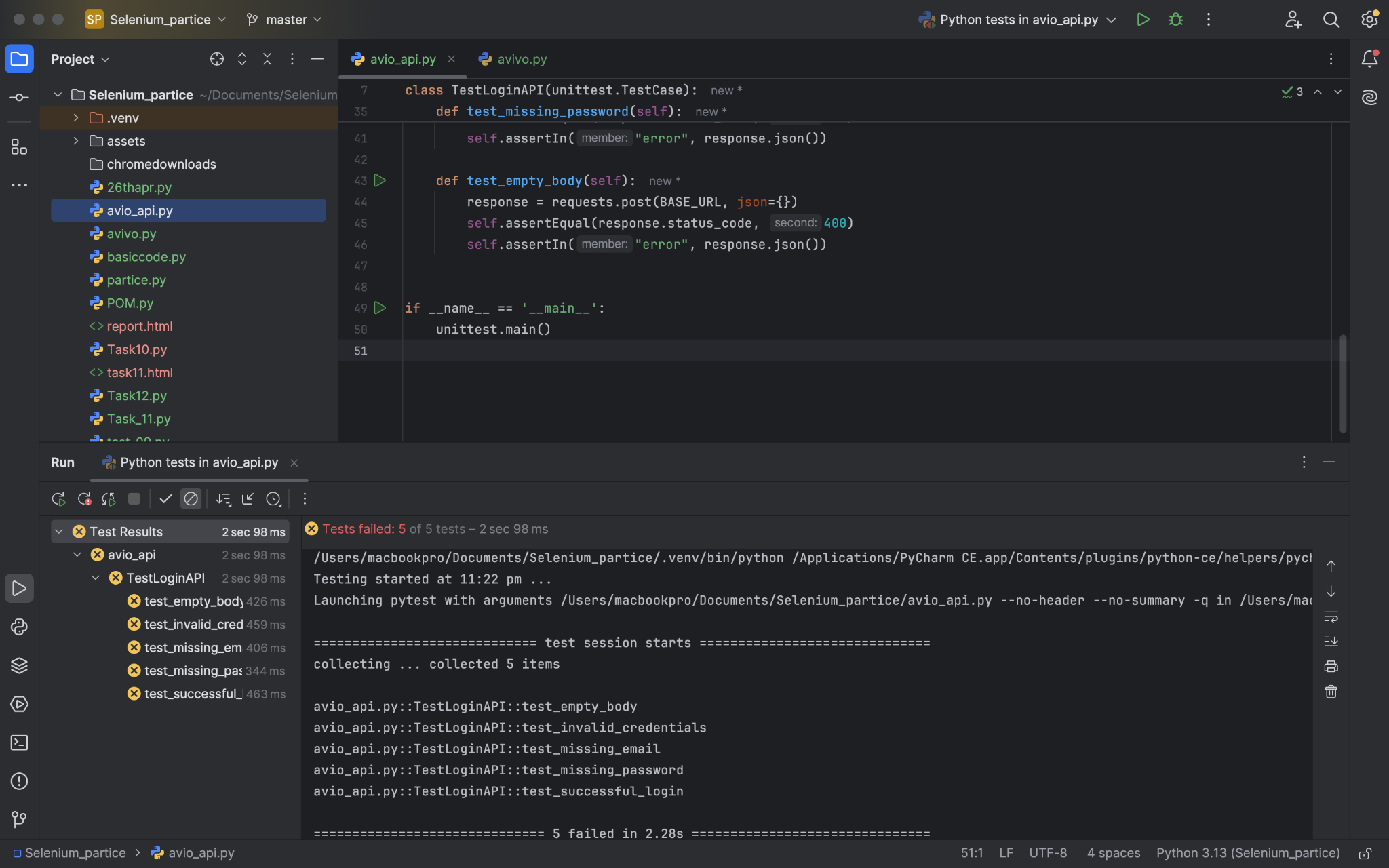
self.assertEqual(response.status\_code, 400)

self.assertIn("error", response.json())

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

Output:



**Section 4: Load Testing**

**Scenario:** Same login API from Section 3.

**Task:**

1. Create a basic **load test script** using **JMeter, k6, or Locust** to simulate **10–50 users** hitting the login API.
2. Measure and report:
   * Average response time
   * Error rate (if any)
   * Peak response time

**Submission Format:**

* Test cases: In Excel or Google Sheets
* Automation scripts: ZIP file or GitHub link
* Load test report: Screenshot or CSV/HTML export
* README: Brief steps to run the scripts

Output:

Test case

| Testcaseid | Description | Endpoint | Input | Method | Expected result | Actual result | status |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC\_01 | To check whether user can able to login with valid credentials | /api/login | eve.holt@reqres.in / cityslicka | Post | 200 with token | 200 ok | pass |
| TC\_02 | To check whether user can able to enter 50 users at the time | /api/login | eve.holt@reqres.in / cityslicka | Post | success | 1 failure | pass |
| TC\_03 | To check whether user can able to login with invalid credentials | /api/login | eve.holt@reqres.in / slicker | Post | 400 | 400 | pass |

Body data

{

"email": "eve.holt@reqres.in",

"password": "cityslicka"

}