A

# Project Report

on

# Mini-App : Open_EMR

Submitted as a partial fulfillment

of the module

## Object-Oriented Programming using Python

L&T Technology Services

**Submitted To:** **Jaishree Kantilal Tailor**
Module Faculty (GENESIS Q3 FY 23)

GLOBAL ENGINEERING ACADEMY by LTTS

Submitted By:

| Mini-App Project Team Details | | | |
|---|---|---|---|
| **PS.No.** | **Full Name** | **Email** | **BU** |
| 40032395 | Masani Sai Bhargav Reddy | Masani.saibhargav@ltts.com | EMD(TET) |
| | | | |
| | | | |
| | | | |
| | | | |

## App Requirement Definition

OpenEMR is a free and open-source electronic medical record (EMR) and medical practice management software. It is designed to help healthcare providers manage patient records, appointment scheduling, billing, and other administrative tasks.

In this OpenEMR I have executed

1. Login
2. Entered the patient details
3. Created the Account for the patient
4. Validated the patient name weather it is same or not
5. Exit portal

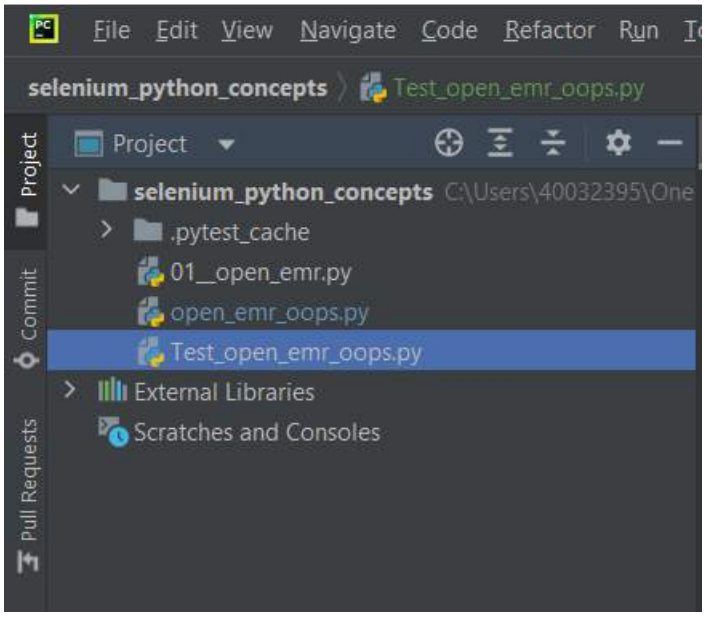
OpenEMR is a web-based application that can be accessed from any device with internet access. It is also customizable and extensible, allowing healthcare providers to tailor the software to their specific needs.

Overall, OpenEMR provides a comprehensive and cost-effective solution for healthcare providers looking to digitize their practices and improve patient care.

# I. App Implementation Source File Structure and Individual Contribution

This section must have 2 items:

**a. Screenshot of the PyCharm Project with its File Structure. It must have the File Explorer View for all the source files, modules created.**




**b. File Summary**

| Items | Items | Description | Responsible Member PSNO. |
|---|---|---|---|
| Modules | **Webdriver** | Webdriver is a tool for automating web application testing | 40032395 |
| | **Selenium.webdriver.common.by** | It provides set of classes that represents the various strategies to locate elements with in a webpage | 40032395 |
| | | | |

| | | | |
|---|---|---|---|
| **Packages** | **selenium** | Selenium is a popular open source suite of tools for automating webbrowesr testing | 40032395 |
| | | | |
| **Source files** | **open_emr_oops.py** | This file performs the successful login and add the details of the patient | 40032395 |
| | **Test_open_emr_oops.py** | This file test the login operations and patient details by using pytest | 40032395 |

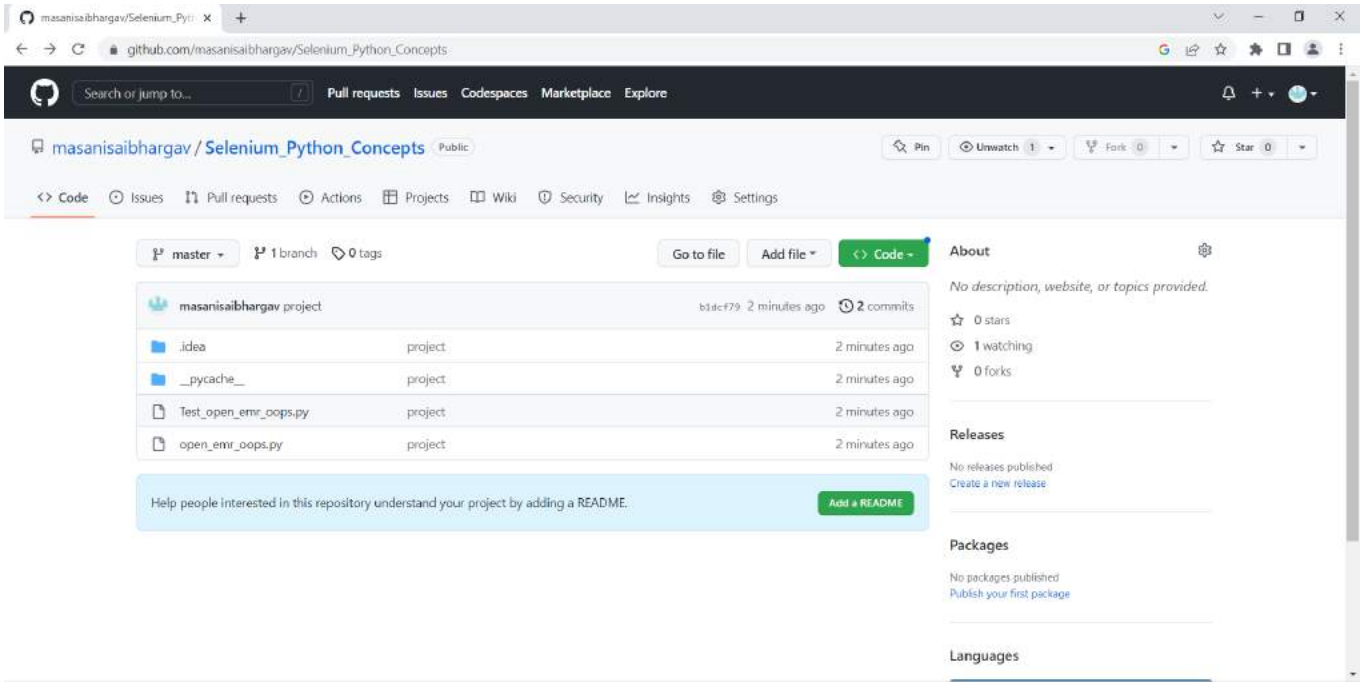
## II. github Dashboard

This section must have 2 items:

### a. github Link

| PSNO | github Link |
|---|---|
| 40032395 | https://github.com/masanisaibhargav/Selenium_Python_Concepts.git |
| | |

### b. github Dashboard view



## III. OOP Concept Utilization Summary

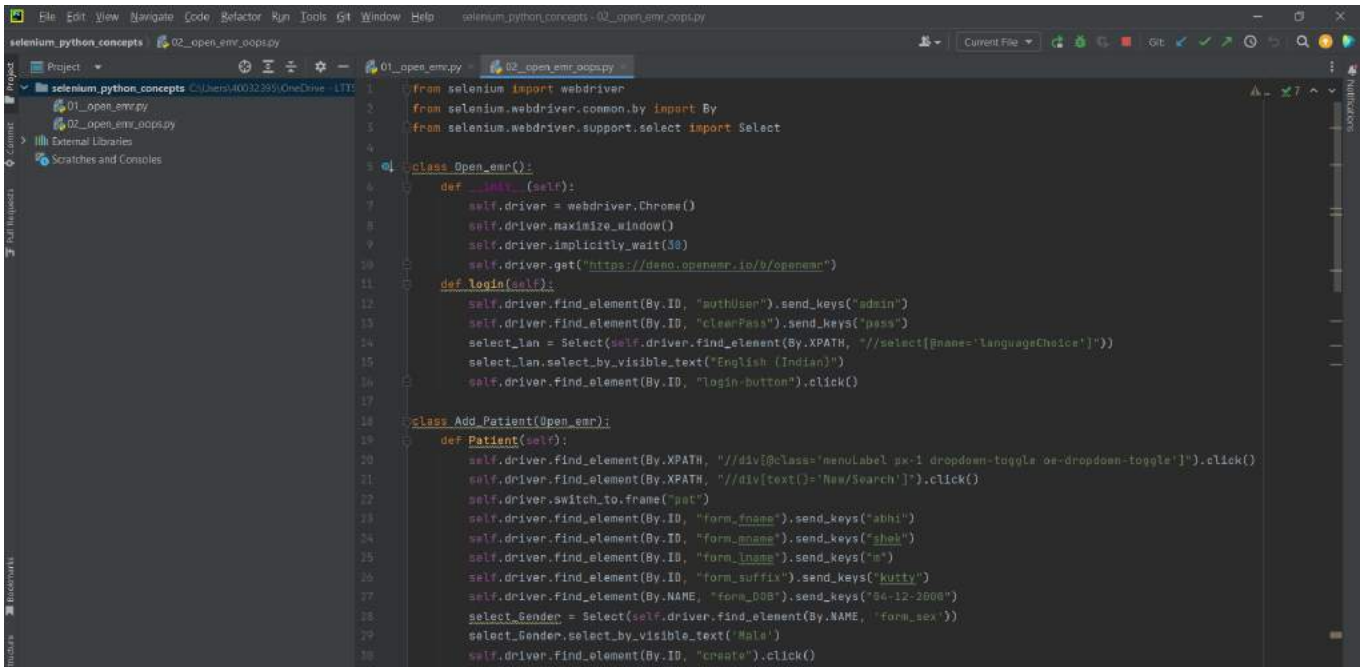
This section must have following items:

| Concept | Description |
|---|---|
| **Class** | I have used two classes<br>• Open_emr:<br>• Add_Patient: |
| **Objects** | obj: An instance of the Add_Patient class, used to interact with the OpenEMR site |
| **Static Methods** | N/A |
| **Class Methods** | N/A |
| **Instance Methods** | • login: A method of the Open_emr class that logs into the OpenEMR site.<br>• Patient: A method of the Add_Patient class that adds a patient to the database.<br>• ele: A method of the Add_Patient class that tests for an exception. |
| **Overloading** | N/A |
| **Overriding** | `def Patient(self):`<br>`  return 0` |
| **Inheritance** | Type of Inheritance: Single inheritance.<br>Base Classes: Open_emr.<br>Child Classes: Add_Patient |
| **Exception Handling** | List Exception Types: Not specified<br>Customized Exception Class:Not used |

# IV. App Implementation Console Layouts

This section must have 2 items:

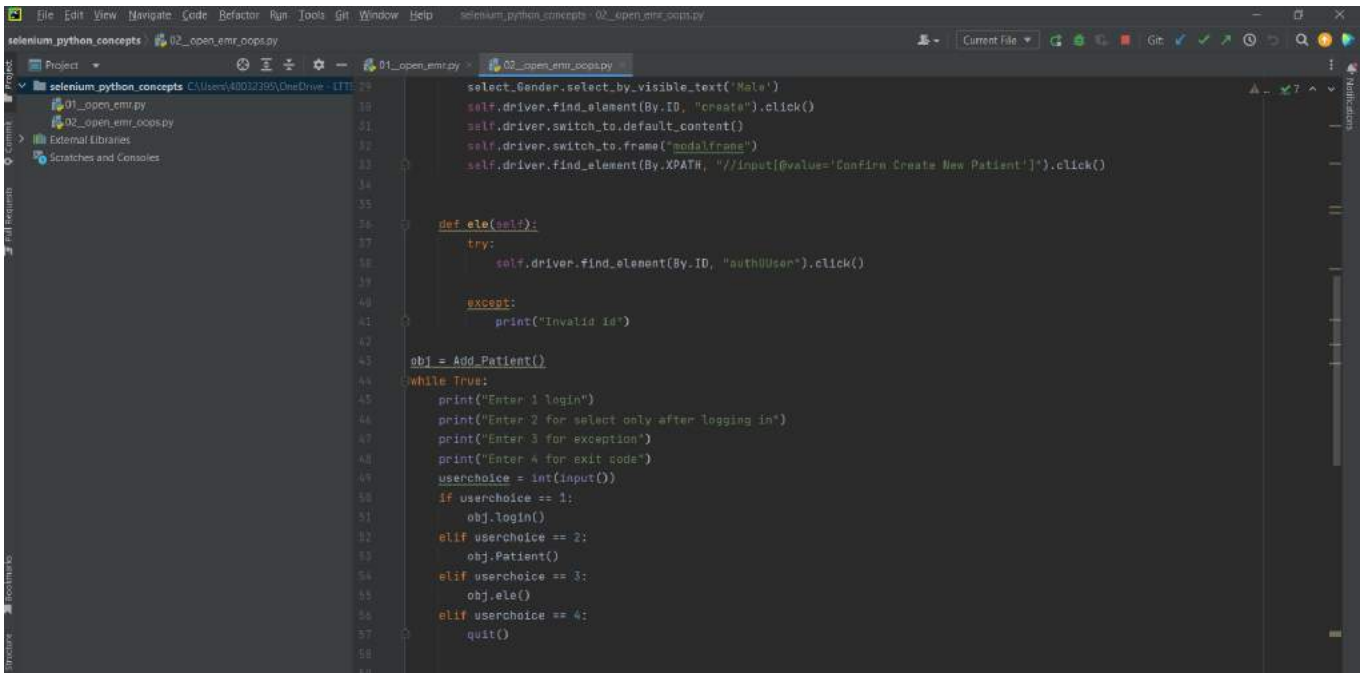
**a. Screenshot of each menu option implemented based on the requirement along with the output.**



```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.select import Select

class Open_emr():
    def __init__(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        self.driver.implicitly_wait(30)
        self.driver.get("https://demo.openemr.io/b/openemr")
    def login(self):
        self.driver.find_element(By.ID, "authUser").send_keys("admin")
        self.driver.find_element(By.ID, "clearPass").send_keys("pass")
        select_lan = Select(self.driver.find_element(By.XPATH, "//select[@name='languageChoice']"))
        select_lan.select_by_visible_text("English (Indian)")
        self.driver.find_element(By.ID, "login-button").click()

class Add_Patient(Open_emr):
    def Patient(self):
        self.driver.find_element(By.XPATH, "//div[@class='menuLabel px-1 dropdown-toggle oe-dropdown-toggle']").click()
        self.driver.find_element(By.XPATH, "//div[text()='New/Search']").click()
        self.driver.switch_to.frame("pat")
        self.driver.find_element(By.ID, "form_fname").send_keys("abhi")
        self.driver.find_element(By.ID, "form_mname").send_keys("shek")
        self.driver.find_element(By.ID, "form_lname").send_keys("m")
        self.driver.find_element(By.ID, "form_suffix").send_keys("kutty")
        self.driver.find_element(By.NAME, "form_DOB").send_keys("04-12-2000")
        select_Gender = Select(self.driver.find_element(By.NAME, 'form_sex'))
        select_Gender.select_by_visible_text('Male')
        self.driver.find_element(By.ID, "create").click()
```



```
        select_Gender.select_by_visible_text('Male')
        self.driver.find_element(By.ID, "create").click()
        self.driver.switch_to.default_content()
        self.driver.switch_to.frame("modalframe")
        self.driver.find_element(By.XPATH, "//input[@value='Confirm Create New Patient']").click()


    def ele(self):
        try:
            self.driver.find_element(By.ID, "authUUser").click()

        except:
            print("Invalid Id")

obj = Add_Patient()
while True:
    print("Enter 1 login")
    print("Enter 2 for select only after logging in")
    print("Enter 3 for exception")
    print("Enter 4 for exit code")
    userchoice = int(input())
    if userchoice == 1:
        obj.login()
    elif userchoice == 2:
        obj.Patient()
    elif userchoice == 3:
        obj.ele()
    elif userchoice == 4:
        quit()
```

**Figure 1 Screenshot of <<menu option>>**

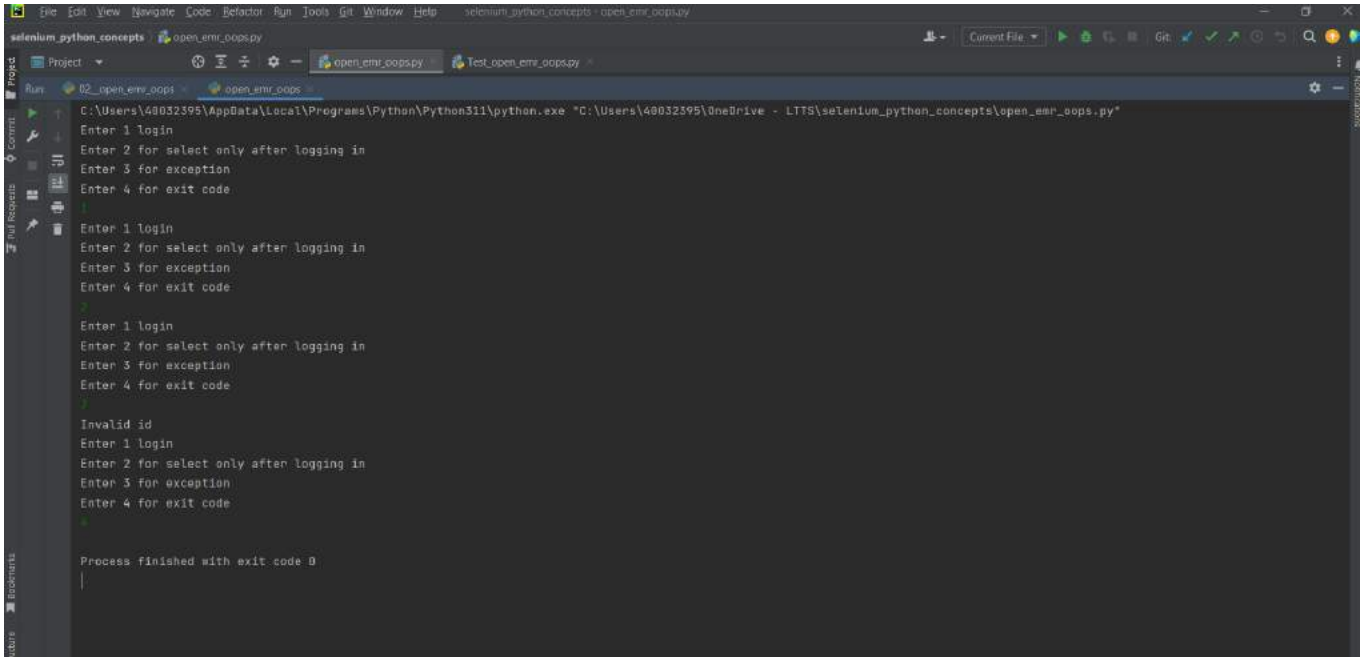



**Figure 2 Screenshot of <<menu option>>**

### b. Brief description of the code, it's input and output

- The code is a Python script using the Selenium WebDriver library to automate actions in a web browser. The script consists of two classes: "Open_emr" and "Add_Patient".
- The Open_emr class is responsible for launching a Chrome browser, navigating to a URL, and logging into the website using admin credentials.
- The Add_Patient class is a subclass of Open_emr and is responsible for creating a new patient in the website by entering patient information and confirming the creation.
- The script also contains a loop that allows the user to interactively select actions to perform such as logging in, adding a new patient, checking for exceptions, or exiting the code.
- The output of the code is either the successful creation of a new patient in the website or error messages in case of exceptions.

## V. Standard Coding Practice Summary

This section PyLint report for any one source file per member. Fix the code with standard practice and re-run for PyLint report

| PSNO | Source File Name | Quality Index and Suggestions before PyLint Recommendation | Quality Index and Suggestions after PyLint Recommendation |
|---|---|---|---|
| 40032395 | open_emr_oops.py | 7.06 | 7.06 |
| 40032395 | Test_open_emr_oops.py | 4.04 | 4.04 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19044.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\40032395\OneDrive - LTTS\selenium_python_concepts>pylint open_emr_oops.py
************* Module open_emr_oops
open_emr_oops.py:20:0: C0301: Line too long (119/100) (line-too-long)
open_emr_oops.py:86:0: C0305: Trailing newlines (trailing-newlines)
open_emr_oops.py:1:0: C0114: Missing module docstring (missing-module-docstring)
open_emr_oops.py:5:0: C0115: Missing class docstring (missing-class-docstring)
open_emr_oops.py:5:0: C0103: Class name "Open_emr" doesn't conform to PascalCase naming style (invalid-name)
open_emr_oops.py:11:4: C0116: Missing function or method docstring (missing-function-docstring)
open_emr_oops.py:5:0: R0903: Too few public methods (1/2) (too-few-public-methods)
open_emr_oops.py:18:0: C0115: Missing class docstring (missing-class-docstring)
open_emr_oops.py:18:0: C0103: Class name "Add_Patient" doesn't conform to PascalCase naming style (invalid-name)
open_emr_oops.py:19:4: C0116: Missing function or method docstring (missing-function-docstring)
open_emr_oops.py:19:4: C0103: Method name "Patient" doesn't conform to snake_case naming style (invalid-name)
open_emr_oops.py:28:8: C0103: Variable name "select_Gender" doesn't conform to snake_case naming style (invalid-name)
open_emr_oops.py:36:4: C0116: Missing function or method docstring (missing-function-docstring)
open_emr_oops.py:40:8: W0702: No exception type(s) specified (bare-except)
open_emr_oops.py:57:8: R1722: Consider using 'sys.exit' instead (consider-using-sys-exit)

-----------------------------------
Your code has been rated at 7.06/10


C:\Users\40032395\OneDrive - LTTS\selenium_python_concepts>pilint 01__open_emr.py
'pilint' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\40032395\OneDrive - LTTS\selenium_python_concepts>pylint 01__open_emr.py
************* Module 01__open_emr
01__open_emr.py:23:0: C0304: Final newline missing (missing-final-newline)
01__open_emr.py:1:0: C0114: Missing module docstring (missing-module-docstring)
01__open_emr.py:1:0: C0103: Module name "01__open_emr" doesn't conform to snake_case naming style (invalid-name)

-----------------------------------
Your code has been rated at 8.70/10
```

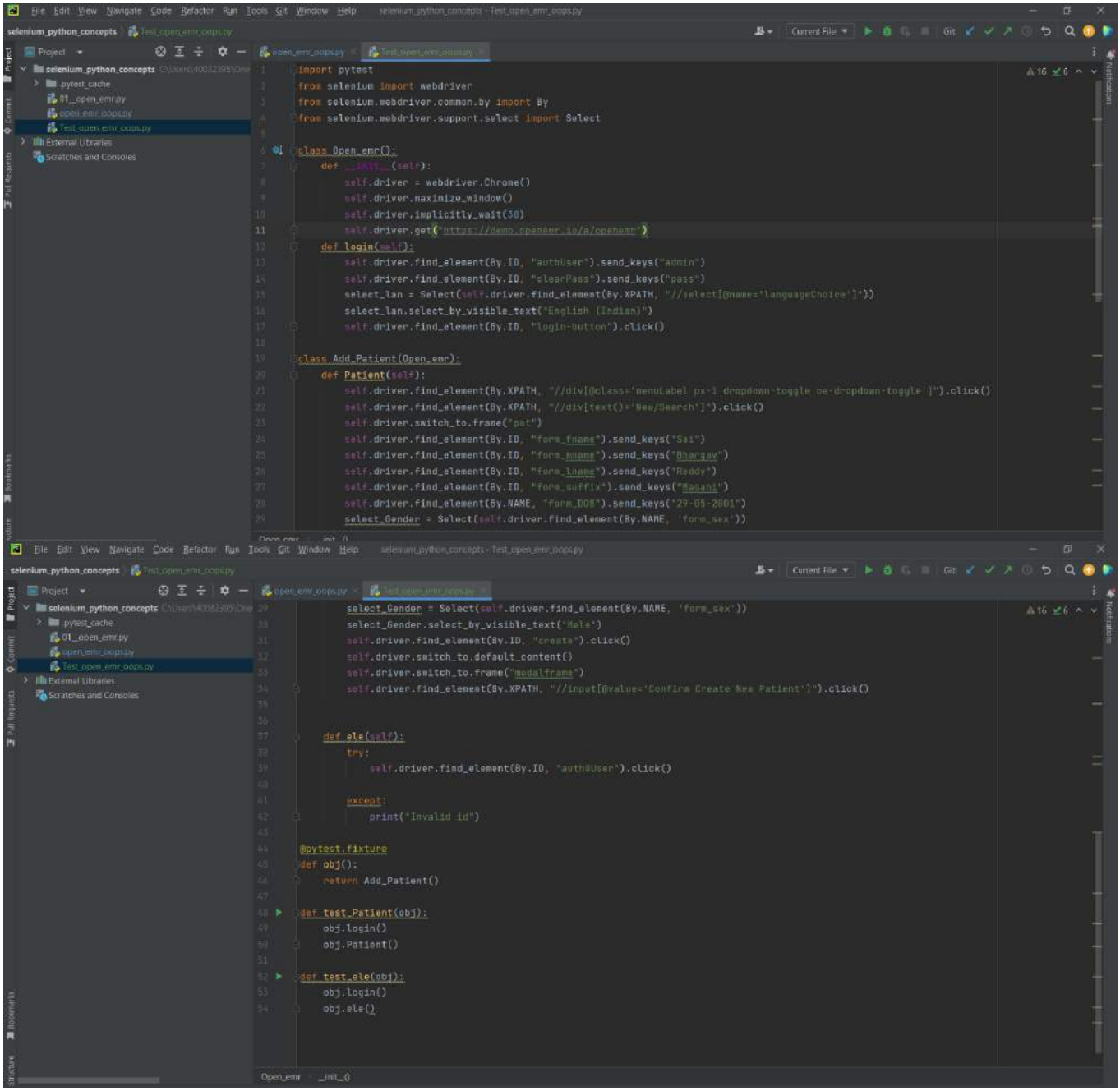
```
C:\Users\40032395\OneDrive - LTTS\selenium_python_concepts>pylint Test_open_emr_oops.py
************* Module Test_open_emr_oops
Test_open_emr_oops.py:24:0: C0301: Line too long (119/100) (line-too-long)
Test_open_emr_oops.py:57:0: C0304: Final newline missing (missing-final-newline)
Test_open_emr_oops.py:1:0: C0114: Missing module docstring (missing-module-docstring)
Test_open_emr_oops.py:1:0: C0103: Module name "Test_open_emr_oops" doesn't conform to snake_case naming style (invalid-name)
Test_open_emr_oops.py:6:0: C0115: Missing class docstring (missing-class-docstring)
Test_open_emr_oops.py:6:0: C0103: Class name "Open_emr" doesn't conform to PascalCase naming style (invalid-name)
Test_open_emr_oops.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:6:0: R0903: Too few public methods (1/2) (too-few-public-methods)
Test_open_emr_oops.py:19:0: C0115: Missing class docstring (missing-class-docstring)
Test_open_emr_oops.py:19:0: C0103: Class name "Add_Patient" doesn't conform to PascalCase naming style (invalid-name)
Test_open_emr_oops.py:20:4: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:20:4: C0103: Method name "Patient" doesn't conform to snake_case naming style (invalid-name)
Test_open_emr_oops.py:23:4: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:23:4: C0103: Method name "Patient" doesn't conform to snake_case naming style (invalid-name)
Test_open_emr_oops.py:23:4: E0102: method already defined line 20 (function-redefined)
Test_open_emr_oops.py:32:8: C0103: Variable name "select_Gender" doesn't conform to snake_case naming style (invalid-name)
Test_open_emr_oops.py:40:4: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:44:8: W0702: No exception type(s) specified (bare-except)
Test_open_emr_oops.py:48:0: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:51:0: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:51:0: C0103: Function name "test_Patient" doesn't conform to snake_case naming style (invalid-name)
Test_open_emr_oops.py:51:17: W0621: Redefining name 'obj' from outer scope (line 48) (redefined-outer-name)
Test_open_emr_oops.py:55:0: C0116: Missing function or method docstring (missing-function-docstring)
Test_open_emr_oops.py:55:13: W0621: Redefining name 'obj' from outer scope (line 48) (redefined-outer-name)

-----------------------------------
Your code has been rated at 4.04/10
```

# VI. Testing

This section must have 2 items:

**a. Screenshot test files along with test methods demonstrating fixtures, parametrization, marking, xfail, xskip**



```
import pytest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.select import Select

class Open_emr():
    def __init__(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        self.driver.implicitly_wait(30)
        self.driver.get("https://demo.openemr.io/a/openemr")
    def login(self):
        self.driver.find_element(By.ID, "authUser").send_keys("admin")
        self.driver.find_element(By.ID, "clearPass").send_keys("pass")
        select_lan = Select(self.driver.find_element(By.XPATH, "//select[@name='languageChoice']"))
        select_lan.select_by_visible_text("English (Indian)")
        self.driver.find_element(By.ID, "login-button").click()

class Add_Patient(Open_emr):
    def Patient(self):
        self.driver.find_element(By.XPATH, "//div[@class='menuLabel px-1 dropdown-toggle oe-dropdown-toggle']").click()
        self.driver.find_element(By.XPATH, "//div[text()='New/Search']").click()
        self.driver.switch_to.frame("pat")
        self.driver.find_element(By.ID, "form_fname").send_keys("Sai")
        self.driver.find_element(By.ID, "form_mname").send_keys("Bhargav")
        self.driver.find_element(By.ID, "form_lname").send_keys("Reddy")
        self.driver.find_element(By.ID, "form_suffix").send_keys("Nagani")
        self.driver.find_element(By.NAME, "form_DOB").send_keys("29-05-2001")
        select_Gender = Select(self.driver.find_element(By.NAME, 'form_sex'))
        select_Gender.select_by_visible_text('Male')
        self.driver.find_element(By.ID, "create").click()
        self.driver.switch_to.default_content()
        self.driver.switch_to.frame("modalframe")
        self.driver.find_element(By.XPATH, "//input[@value='Confirm Create New Patient']").click()


    def ele(self):
        try:
            self.driver.find_element(By.ID, "authUser").click()

        except:
            print("Invalid id")

@pytest.fixture
def obj():
    return Add_Patient()

def test_Patient(obj):
    obj.login()
    obj.Patient()

def test_ele(obj):
    obj.login()
    obj.ele()
```

**Figure 3 Screenshot of <<menu option>>**

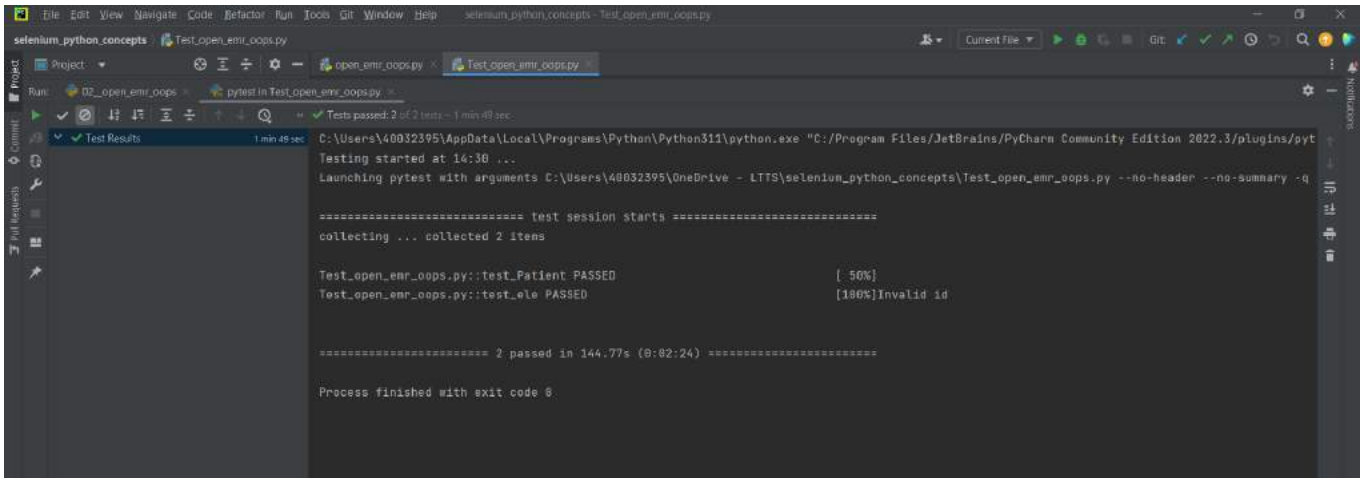



**Figure 4 Screenshot of <<menu option>>**

### b. Brief description of the code

- This code is a Python script using the PyTest testing framework and Selenium library. It defines two classes, "Open_emr" and "Add_Patient," to automate the process of logging into the OpenEMR demo website and adding a new patient.

- The "Add_Patient" class extends the "Open_emr" class and adds functionality to create a new patient.

- The code includes two test cases defined with PyTest, "test_Patient" and "test_ele," which test the functionality of adding a patient and finding an element on the page respectively.

- Additionally, the code has a fixture, "obj," which returns an instance of the "Add_Patient" class. This fixture can be used in multiple test cases to reuse the instance of the class.

## VII. Learnings from the Mini-App Project Journey

- Understanding the requirements: It is important to understand the requirements of the project in detail before starting to work on it. This helps in reducing the scope creep and ensures that the final product meets the expectations of the stakeholders.

- Familiarizing with the tools and technologies: Before starting to work on the project, it is important to familiarize yourself with the tools and technologies being used in the project. This helps in reducing the learning curve and increases the efficiency of the project.

- Planning and execution: Effective planning and execution of the project is critical to its success. This involves breaking down the project into smaller tasks and assigning them to the team members. Regular monitoring of the progress and tracking of the timelines helps in ensuring that the project stays on track.

- Documentation and communication: Documentation of the project requirements, design, and implementation is crucial for the success of the project. Clear and regular communication between the team members and stakeholders helps in ensuring that everyone is on the same page.

- Testing and Quality assurance: Testing and quality assurance play a crucial role in ensuring that the project meets the desired quality standards. Regular testing and fixing of the bugs helps in reducing the number of defects in the final product.

- Teamwork and collaboration: Teamwork and collaboration between the team members is critical to the success of the project. Each team member should have a clear understanding of their role and responsibilities, and they should work together towards achieving the project goals

- Continuous improvement: The project should not be considered complete once it is delivered. Continuous improvement and maintenance of the project are important to ensure that it continues to meet the evolving requirements of the stakeholders.

## VIII. References

List the references in IEEE/APA format for web-resources, books etc. referred for the mini-app implementation

- For selenium to python : https://github.com/balaji-githubstore/python_selenium_concepts_tata_tech.git
- For pylint: pypi.org/project/pytest-pylint/