Automating Form Submission and Email Notification using Django and Selenium

**Assignment Objectives:**

This project demonstrates how to automate the submission of a Google form and send a confirmation email with a screenshot of the submission confirmation page. The process involves using Selenium to fill out the form and Django to send the email. The system is designed to run in a console-based environment where user inputs are taken from the console.

Google Form Link - https://forms.gle/WT68aV5UnPajeoSc8

Email subject will be Python (Selenium) Assignment - {Your Name}

Email to tech@themedius.ai and CC HR@themedius.ai

GIT - Share the GIT Repo with your code

**Tech Stack:**

1. **Python:**

- The primary programming language used for scripting and automation.

2. **Selenium:**

- A browser automation tool used to interact with web elements on the Google form.

- We use Selenium WebDriver to control a web browser (Chrome in this case) to fill out and submit the form.

3. **Django:**

- A high-level Python web framework used to handle the backend logic for sending emails.

- Django provides the email sending functionality through its built-in email library.

4. **WebDriver:**

- ChromeDriver: A standalone server that implements the WebDriver's wire protocol for Chromium.

**Approach:**

1. **XPath Identification:**

- We inspected the Google form to identify the correct XPath selectors for each input field. This allows Selenium to locate and interact with these elements.

- Example XPath for the full name input field: `//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[1]/div/div/div[2]/div/div[1]/div/div[1]/input`.

2. **Form Filling and Submission:**

- Using Selenium, we automate the process of opening the Google form, filling in the user-provided data, and submitting the form.

- We handle potential exceptions to ensure the script runs smoothly and provides meaningful error messages if something goes wrong.

3. **Email Sending:**

- Once the form is submitted, we take a screenshot of the confirmation page.

- Using Django's email framework, we send an email with the screenshot attached to a specified recipient.

Source Code

**- form\_submitter/settings.py:**

INSTALLED\_APPS = [

*# Default Django apps...*

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

*# Your custom apps*

    'email\_sender',

]

*# Other settings...*

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_HOST\_USER = 'your-email@gmail.com'

EMAIL\_HOST\_PASSWORD = 'your-generated-password'

- Contains the Django project settings, including email configuration.

**- email\_sender/submit\_form.py**

*from* selenium *import* webdriver

*from* selenium.webdriver.common.by *import* By

*from* selenium.webdriver.support.ui *import* WebDriverWait

*from* selenium.webdriver.support *import* expected\_conditions *as* EC

*from* selenium.common.exceptions *import* TimeoutException

*import* time

def submit\_form():

*# Taking user input*

    full\_name = input('Enter Full Name: ')

    contact\_number = input("Enter Contact Number (10 digits): ")

    email\_id = input("Enter Email ID: ")

    full\_address = input("Enter Full Address: ")

    pin\_code = input("Enter Pin Code: ")

    date\_of\_birth = input("Enter Date of Birth (DD-MM-YYYY): ")

    gender = input("Enter Gender (Male/Female): ")

    verification\_code = input("Enter Verification code (GNFPYC): ")

*# Initialize the WebDriver*

    driver = webdriver.Chrome()

    driver.get('https://forms.gle/WT68aV5UnPajeoSc8')

*try*:

*# Full Name*

        full\_name\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[1]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        full\_name\_field.send\_keys(full\_name)

*# Contact Number*

        contact\_number\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[2]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        contact\_number\_field.send\_keys(contact\_number)

*# Email ID*

        email\_id\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[3]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        email\_id\_field.send\_keys(email\_id)

*# Full Address*

        full\_address\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[4]/div/div/div[2]/div/div[1]/div[2]/textarea'))

        )

        full\_address\_field.send\_keys(full\_address)

*# Pin Code*

        pin\_code\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[5]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        pin\_code\_field.send\_keys(pin\_code)

*# Date of Birth*

        dob\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[6]/div/div/div[2]/div/div/div[2]/div[1]/div/div[1]/input'))

        )

        dob\_field.send\_keys(date\_of\_birth)

*# Gender*

        gender\_field = WebDriverWait(driver, 10).until(

            EC.element\_to\_be\_clickable((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[7]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        gender\_field.send\_keys(gender)

*# Verification Code*

        verification\_code\_field = WebDriverWait(driver, 10).until(

            EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[2]/div[8]/div/div/div[2]/div/div[1]/div/div[1]/input'))

        )

        verification\_code\_field.send\_keys(verification\_code)

*# Submit the form*

        submit\_button = WebDriverWait(driver, 10).until(

            EC.element\_to\_be\_clickable((By.XPATH, '//\*[@id="mG61Hd"]/div[2]/div/div[3]/div[1]/div[1]/div/span'))

        )

        submit\_button.click()

*# Wait for confirmation page and take screenshot*

        time.sleep(5)

        driver.save\_screenshot('confirmation.png')

*except* Exception *as* e:

        print(f"An unexpected error occurred: {e}")

*finally*:

        driver.quit()

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

    submit\_form()

- **email\_sender/views.py:**

- *# from django.shortcuts import render*

*# Create your views here*

*from* django.core.mail *import* EmailMessage

*from* django.http *import* HttpResponse

*from* django.conf *import* settings

*import* os

def send\_email(*request*):

    subject = 'Python (Selenium) Assignment - Vaibhav Rawat'

    message = (

        'Dear Hiring Team,\n\n'

        'I hope this email finds you well.\n\n'

        'I would like to extend my sincere apologies for the delay in submitting my Python (Selenium) assignment. '

        'Due to unforeseen circumstances, I encountered some difficulties that required additional time to resolve.\n\n'

        'Nevertheless, I am pleased to inform you that I have successfully completed the assignment. '

        'Please find the attached screenshot for confirmation.\n\n'

        'GitHub Repository: https://github.com/Rawat107/From-Auto-Mailer.git\n'

        'Resume: https://drive.google.com/file/d/1Op1voY7rZXEzsuUvSLZA\_glotHJTzKs6/view?usp=drive\_link\n\n'

        'Thank you for your understanding and patience.\n\n'

        'Best regards,\n'

        'Vaibhav Rawat'

    )

    email = EmailMessage(

        subject,

        message,

        settings.EMAIL\_HOST\_USER,

        ['tech@themedius.ai'],

*cc*=['HR@themedius.ai']

    )

    ss\_path = "C:\\Users\\vaibh\\OneDrive\\Desktop\\From-Auto-Mailer\\form\_submitter\\confirmation.png"

*if* os.path.exists(ss\_path):

        email.attach\_file(ss\_path)

*else*:

*return* HttpResponse('Test file does not exist')

    email.send()

*return* HttpResponse('Email sent successfully')

- **email\_sender/urls.py:**

*# email\_sender/urls.py*

*from* django.urls *import* path

*from* .views *import* send\_email

urlpatterns = [

    path('send-email/', send\_email, *name*='send\_email'),

]

- Maps URLs to the views.

**- form\_submitter/urls.py**

*from* django.contrib *import* admin

*from* django.urls *import* path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include('email\_sender.urls')),

]

- Maps URLs

- **submit\_and\_email.py:**

*# email\_sender/management/commands/submit\_and\_email.py*

*from* django.core.management.base *import* BaseCommand

*from* email\_sender.submit\_form *import* submit\_form

*from* email\_sender.views *import* send\_email

*from* django.test *import* RequestFactory

class Command(BaseCommand):

    help = 'Submit the Google form and send an email with the confirmation screenshot'

    def handle(*self*, \**args*, \*\**kwargs*):

        submit\_form()

        factory = RequestFactory()

        request = factory.get('/send-email/')

        send\_email(request)

- Integrates form submission and email sending in one script for simplicity.

**Console-Based User Input:**

The script is designed to run in a console-based environment. It prompts the user to input their data, which is then used to fill out and submit the Google form.

**Conclusion:**

The system integrates Selenium and Django to automate the process of submitting a Google form and sending a confirmation email. Users can configure their email settings in settings.py, run the Django development server, and execute the form submission script. The console-based interface ensures that users can input their details directly, streamlining the automation process.