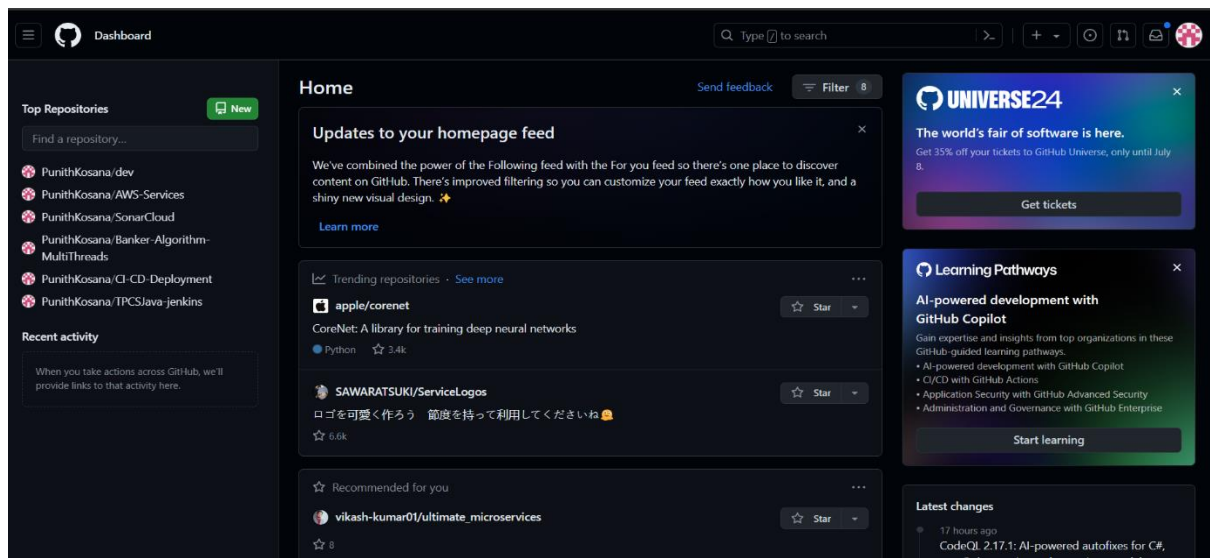
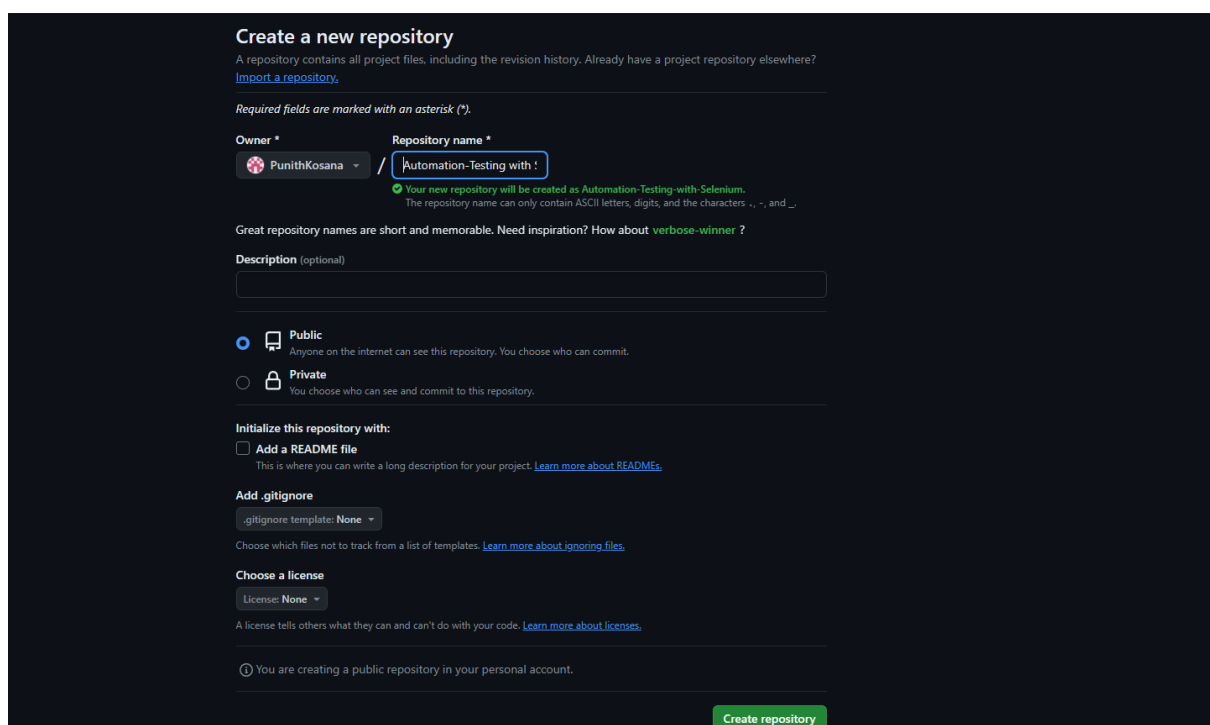


Creating Github Actions Pipeline involving usage of Selenium-Python Automation Testing

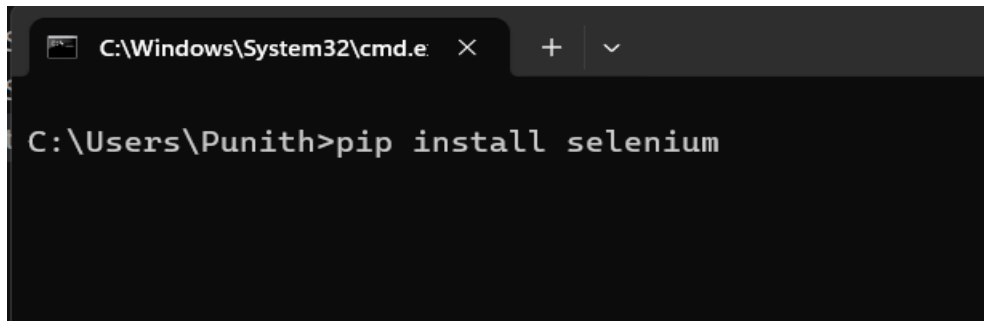
Open github.com and login into it. Choose New to create new repository.



Name the repository and make it public. Choose Create Repository.




Install selenium.



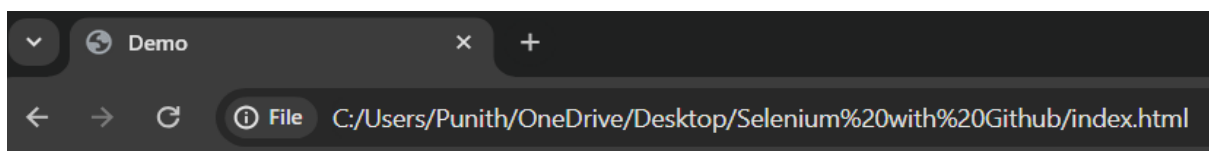
```
C:\Windows\System32\cmd.e X + v
C:\Users\Punith>pip install selenium
```

This is the website we are going to deploy with github actions.



```
index.html X
index.html > html > body > p
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Demo</title>
7 </head>
8 <body>
9   <h1>Welcome to my website</h1>
10  <p>Automation Testing with selenium</p>
11 </body>
12 </html>
13
14
```

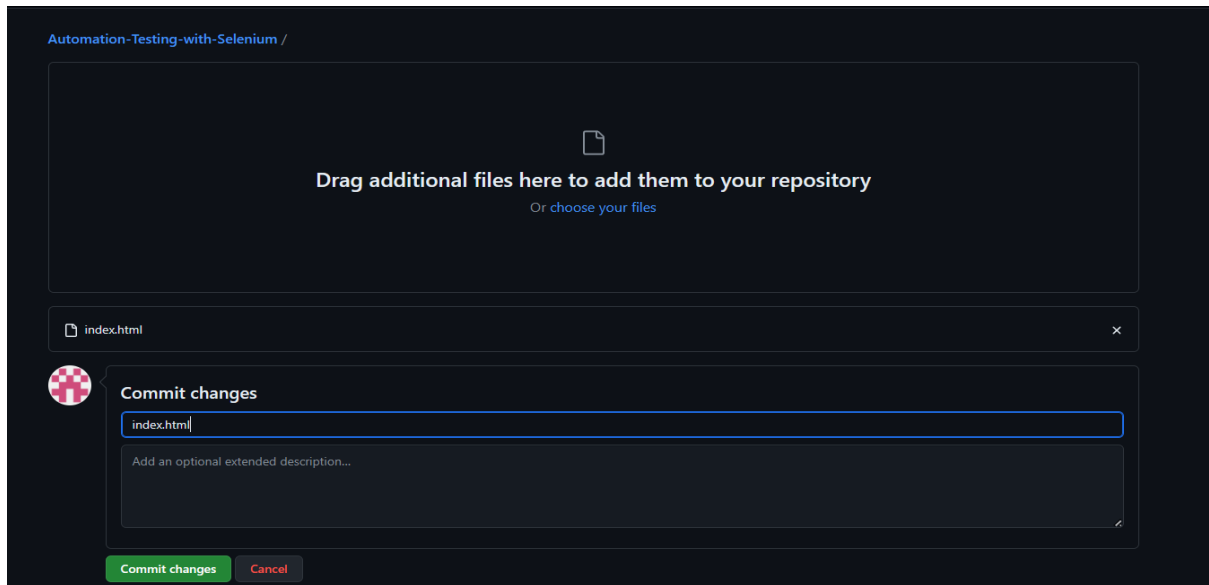
This is how website looks like.



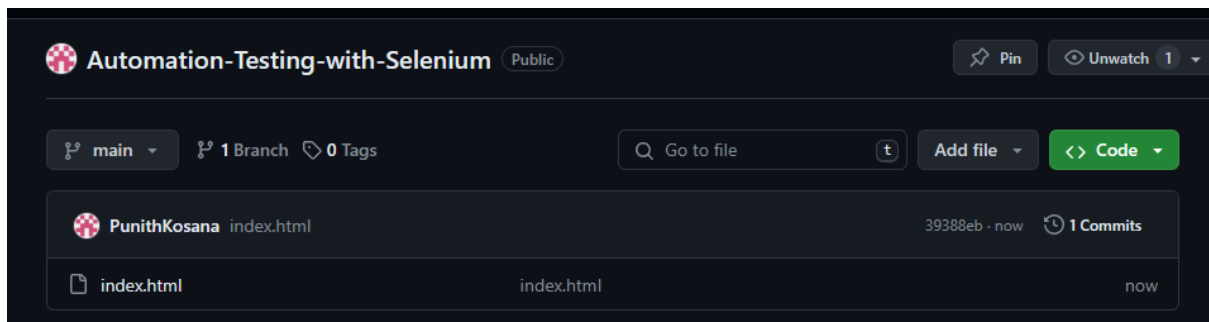
Welcome to my website

Automation Testing with selenium

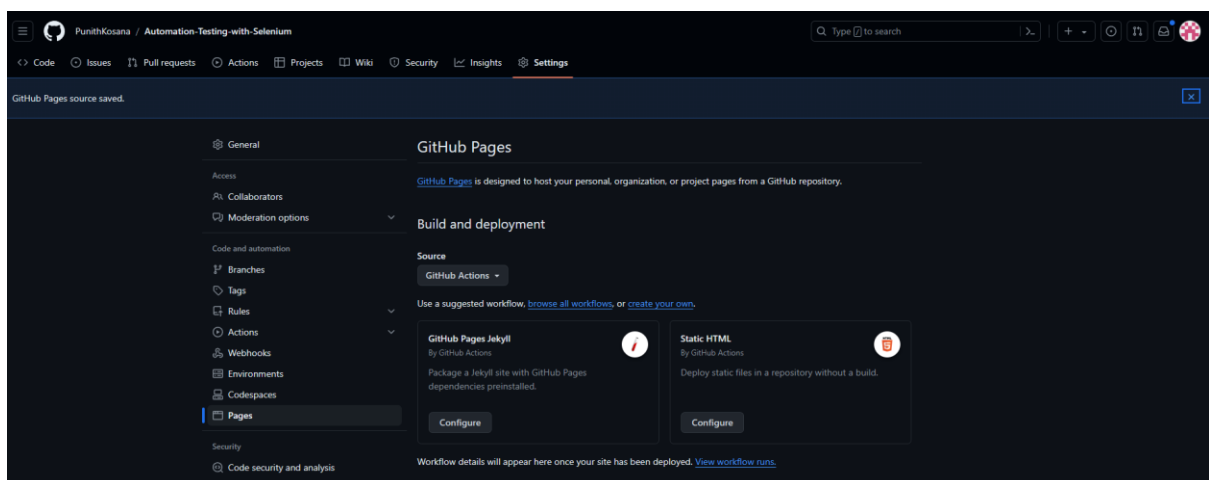
Add the file into repo and commit changes.



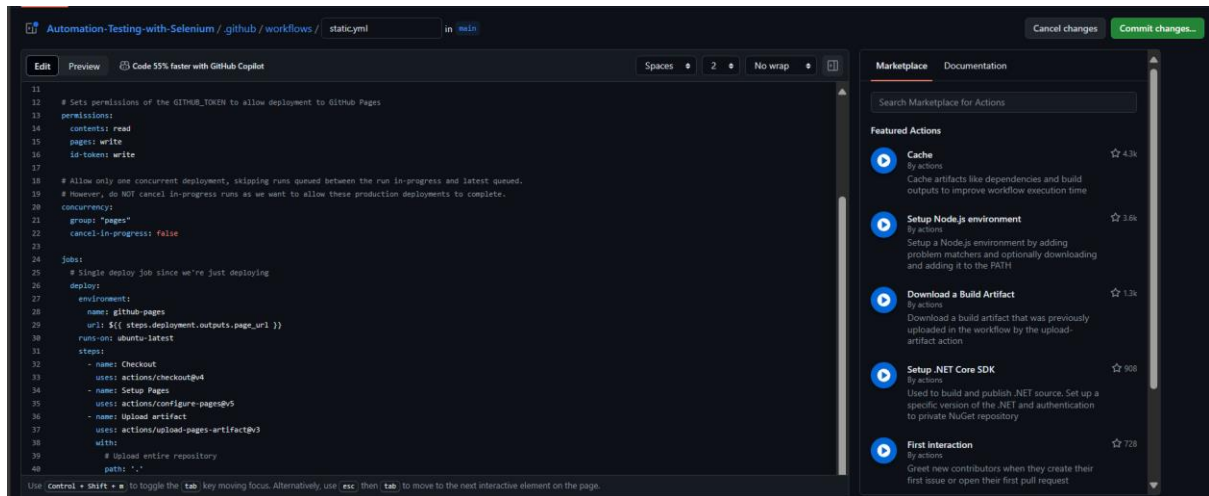
File has been successfully pushed into the repository.



Go to Settings>Github Pages. Configure Static HTML. Make sure to choose the source as Github Actions.

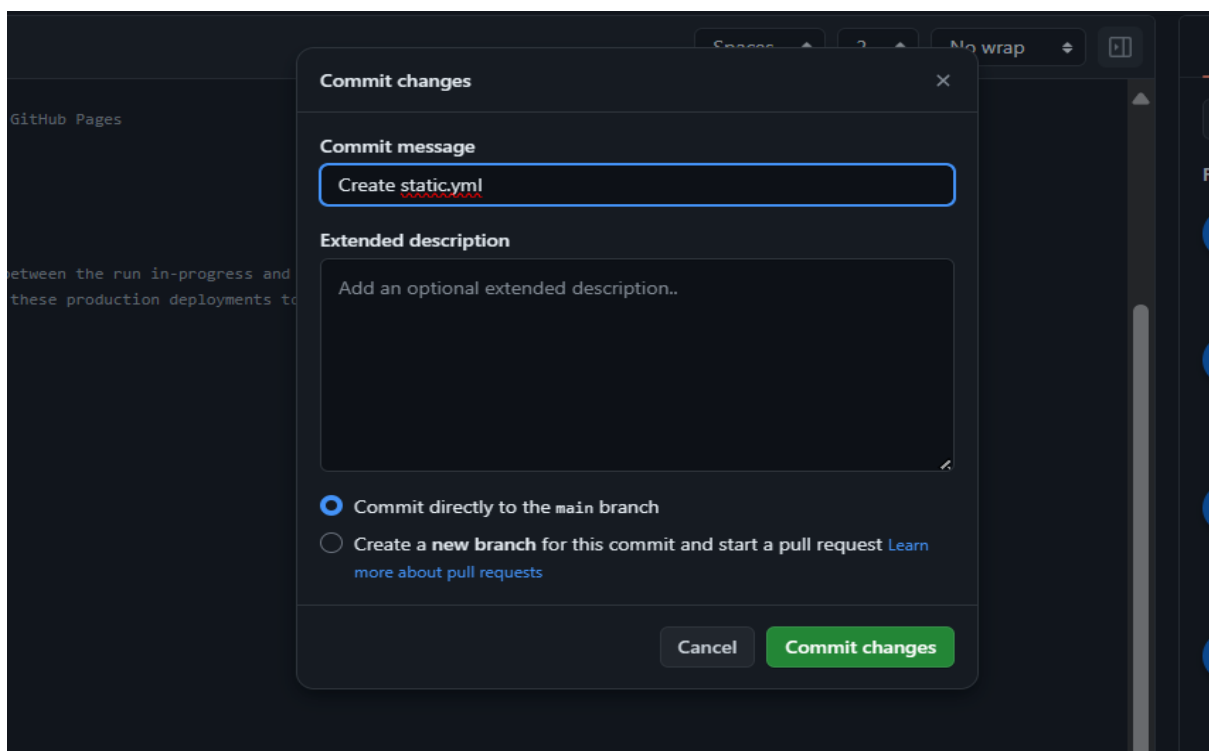


This is the code used to deploy onto the github pages.



```
11
12 # Sets permissions of the GITHUB_TOKEN to allow deployment to GitHub Pages
13 permissions:
14   contents: read
15   pages: write
16   id-token: write
17
18 # Allow only one concurrent deployment, skipping runs queued between the run in-progress and latest queued.
19 # However, do NOT cancel in-progress runs as we want to allow these production deployments to complete.
20 concurrency:
21   group: "pages"
22   cancel-in-progress: false
23
24 jobs:
25   # Single deploy job since we're just deploying
26   deploy:
27     environment:
28       name: github-pages
29       url: ${{ steps.deployment.outputs.page_url }}
30     runs-on: ubuntu-latest
31     steps:
32       - name: Checkout
33         uses: actions/checkout@v4
34       - name: Setup Pages
35         uses: actions/configure-pages@v5
36       - name: Upload artifact
37         uses: actions/upload-pages-artifact@v3
38       with:
39         # Upload entire repository
40         path: "."
```

Commit the changes.



Commit changes

Commit message

Create static.yml

Extended description

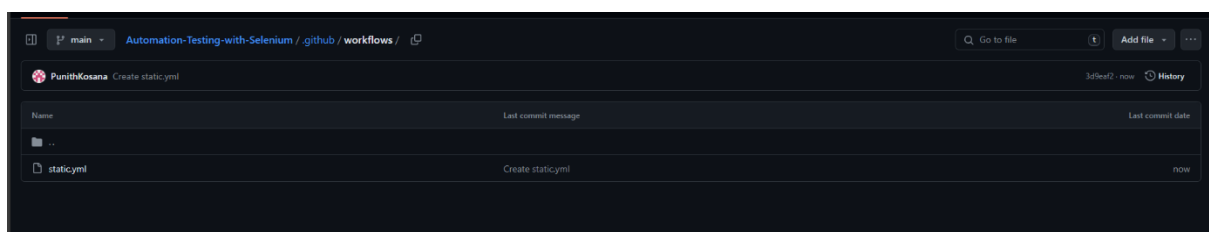
Add an optional extended description..

☒ Commit directly to the main branch

☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

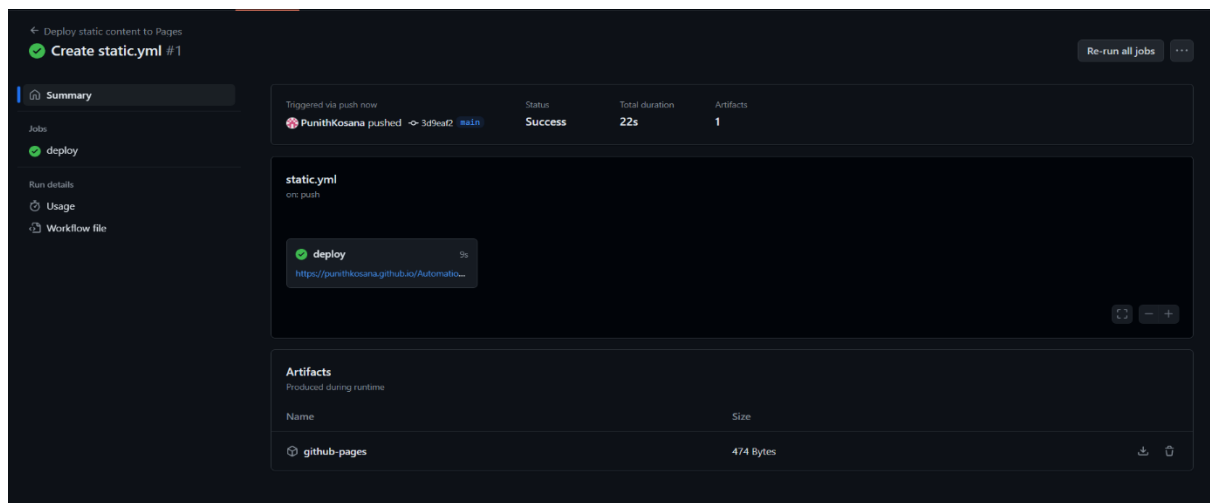
Cancel Commit changes

The github workflow has been created successfully.

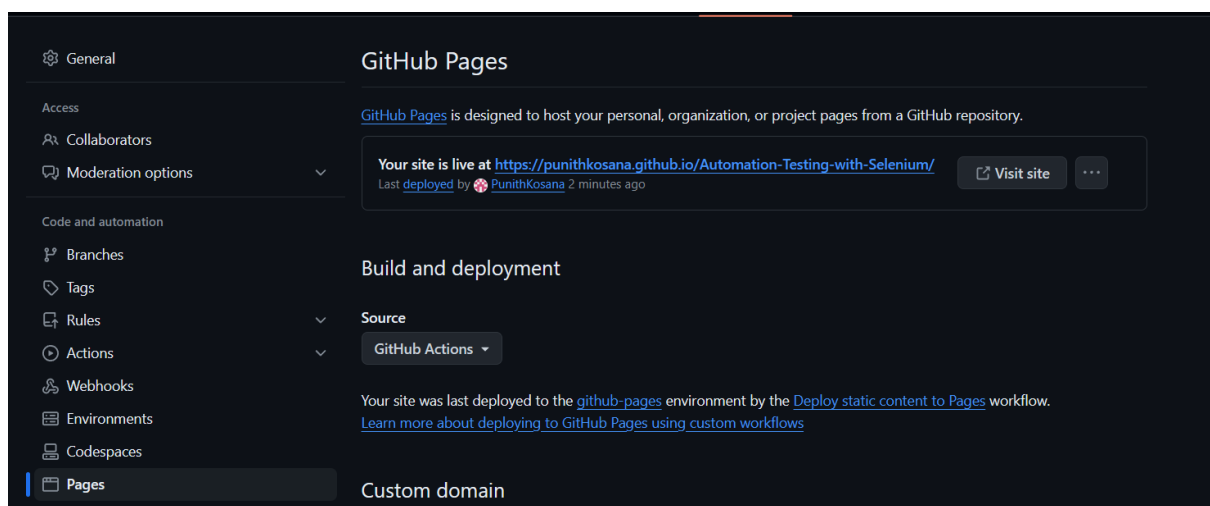


Name	Last commit message	Last commit date
..		
static.yml	Create static.yml	now

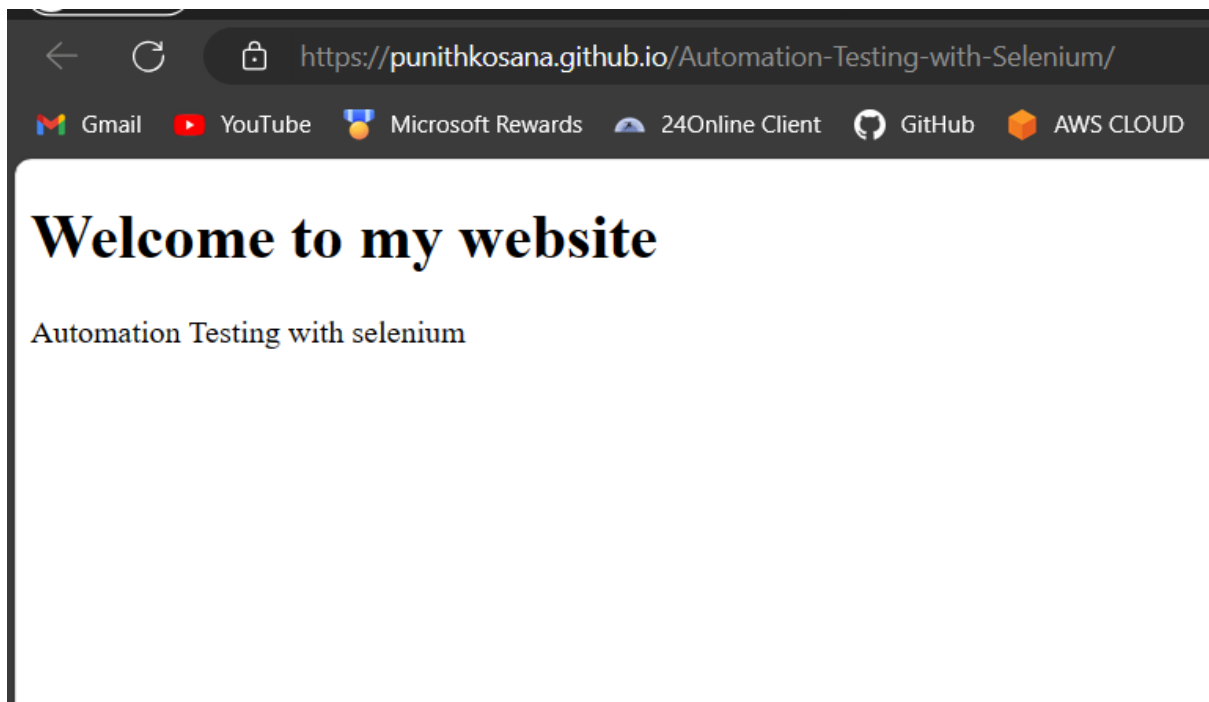
Choose static.yml and click view runs. The deployment has been successful.



Open Github Pages and visit the site.



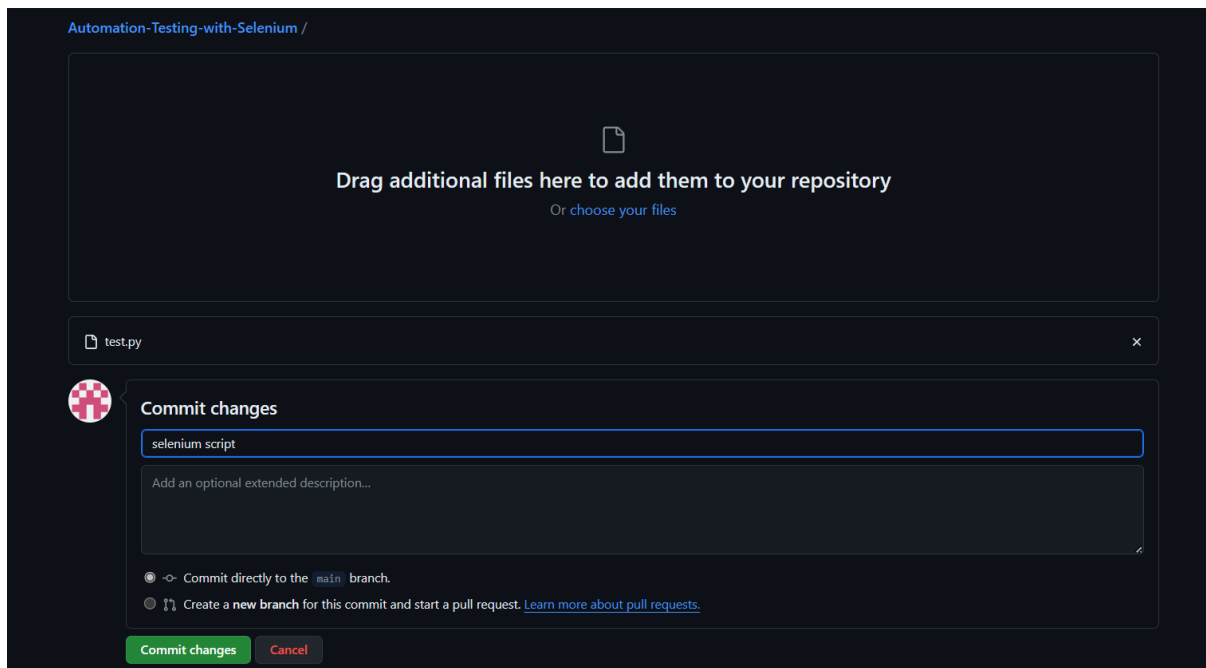
The website have been deployed.



This is the python-selenium script used to automate the process. Whenever the changes made in the repo, the scripts executes and the screenshot of the website will be pushed into the repository automatically.

```
test.py
test.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.chrome.service import Service
3  import time
4
5  # Start the WebDriver and open the HTML page
6  service = Service(executable_path='/usr/local/bin/chromedriver')
7  options = webdriver.ChromeOptions()
8  options.add_argument("--headless")
9  options.add_argument('--no-sandbox')
10 driver = webdriver.Chrome(service=service, options=options)
11
12 driver.get("https://punithkosana.github.io/Automation-Testing-with-Selenium/")
13
14 time.sleep(5) # Adding a delay to see the result
15
16 # Assert some condition to verify the result
17 assert "" in driver.title
18
19 # Take a screenshot
20 timestamp = time.strftime("%Y%m%d-%H%M%S")
21 screenshot_file = f"screenshot_{timestamp}.png"
22 driver.save_screenshot(screenshot_file)
23
24 # Close the WebDriver
25 driver.close()
```

Upload the python file and commit changes.



The screenshot shows a GitHub repository page for "Automation-Testing-with-Selenium". A large dark box with a file icon and the text "Drag additional files here to add them to your repository" is at the top, with a link "Or choose your files" below it. Below this, a file named "test.py" is shown in a light blue bar. Underneath, a "Commit changes" dialog box is open, featuring a red and white checkered profile icon. The dialog has a text input field containing "selenium script" and a larger text area for an optional description. At the bottom of the dialog, there are two radio button options: "Commit directly to the main branch." (which is selected) and "Create a new branch for this commit and start a pull request." with a link to "Learn more about pull requests." At the very bottom of the dialog are two buttons: "Commit changes" in green and "Cancel" in grey.

Add these steps that has to perform and test at the time of deployment in the static.yml file.

```
# Steps to execute within the job
steps:
  # Step to checkout the repository code
  - name: Checkout Repository
    uses: actions/checkout@v2

  # Step to set up Python environment
  - name: Set up Python
    uses: actions/setup-python@v2
    with:
      python-version: "3.12" # Version of Python to set up

  # Step to install Chrome WebDriver for Selenium
  - name: Install Chrome WebDriver
    run: |
      LATEST=$(wget -q -O - https://googlechromelabs.github.io/chrome-for-testing/LATEST_RELEASE_STABLE)
      wget https://storage.googleapis.com/chrome-for-testing-public/$LATEST/linux64/chromedriver-linux64.zip
      unzip chromedriver-linux64.zip
      cd chromedriver-linux64
      sudo mv chromedriver /usr/local/bin/
      sudo chmod +x /usr/local/bin/chromedriver

  # Step to install Selenium library
  - name: Install Selenium
    run: pip install selenium

  # Step to execute Selenium tests
  - name: Run Selenium tests
    run: python test.py

  - name: Save screenshot to repository
```

```

64 # Step to install Selenium library
65 - name: Install Selenium
66   run: pip install selenium
67
68 # Step to execute Selenium tests
69 - name: Run Selenium tests
70   run: python test.py
71
72
73 - name: Save screenshot to repository
74   if: ${{ success() }} # Condition to run only if tests succeed
75   run: |
76     git config --global user.email "actions@github.com"
77     git config --global user.name "GitHub Actions"
78     git add screenshot_*.png
79     git commit -m "Screenshot taken successfully"
80     git push
81
82 deploy:
83   needs: test
84   environment:
85     name: github-pages
86     url: ${{ steps.deployment.outputs.page_url }}
87   runs-on: ubuntu-latest
88   steps:
89     - name: Checkout
90       uses: actions/checkout@v4
91     - name: Setup Pages
92       uses: actions/configure-pages@v5
93     - name: Upload artifact
94       uses: actions/upload-pages-artifact@v3
95     with:
96       # Upload entire repository
97       path: '.'
98     - name: Deploy to GitHub Pages
99       id: deployment
100      uses: actions/deploy-pages@v4

```

Commit the changes.

Commit changes

Commit message

Update static.yml

Extended description

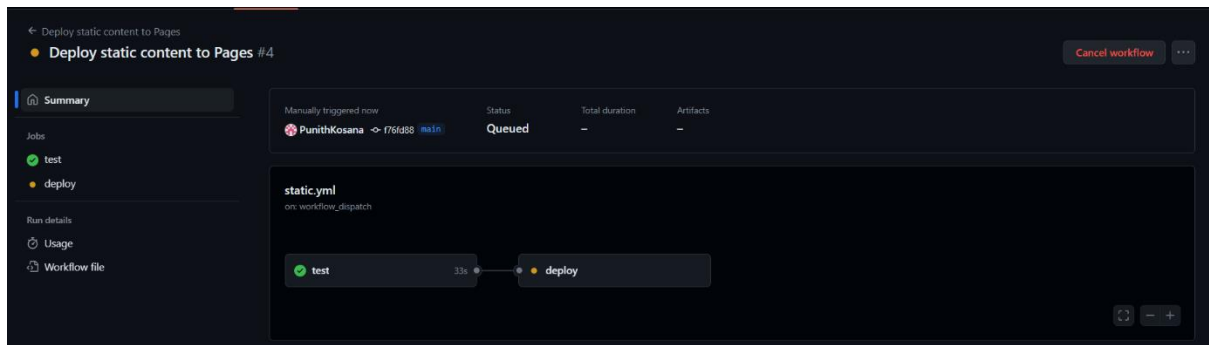
Add an optional extended description..

☒ Commit directly to the main branch

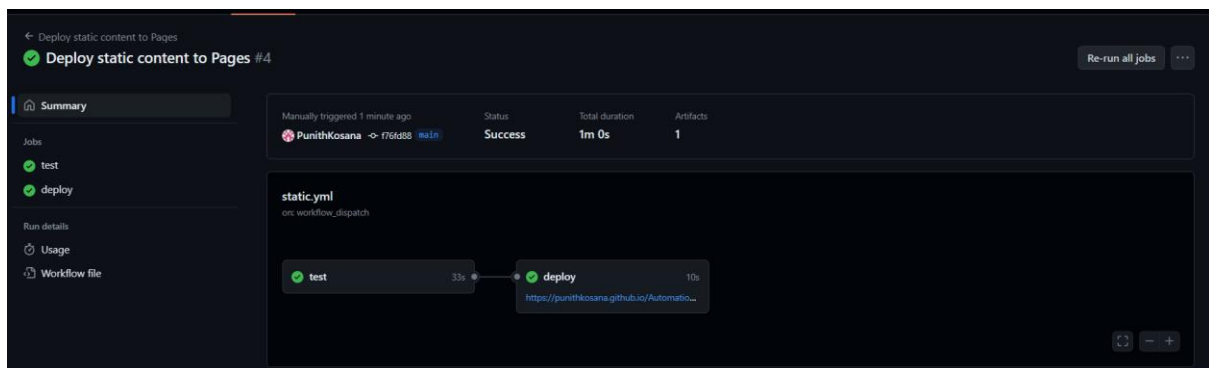
☐ Create a new branch for this commit and start a pull request [Learn more about pull requests](#)

Cancel Commit changes

The test has been executed successfully.



The deployment also done successfully.



The screenshot has been taken and pushed into the repository.

