| **Redbus Data Scraping with Selenium & Dynamic Filtering using Streamlit** |
| --- |

1. **Data Scrapin**

## **Overview**

This script uses Selenium to automate web scraping from the Kadamba Transport Corporation's bus booking page on the Redbus website. It extracts various details about bus routes, including bus names, types, timings, durations, prices, seat availability, and star ratings. The data is collected across multiple pages and stored in a Pandas DataFrame.

## **Prerequisites**

* Python 3.x
* Selenium: Web automation tool
* ChromeDriver: WebDriver for Chrome browser
* Pandas: Data manipulation and analysis library

### **Installation**

pip install selenium pandas

## **Script Breakdown**

### **1. Import Libraries**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

### **2. Initialize WebDriver**

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

### **3. Open the Website**

driver.get('<https://www.redbus.in/online-booking/ktcl/?utm_source=rtchometile>')

### **4. Wait for Page Load and Scroll**

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

### **5. Extract URLs of Detail Pages**

all\_ktclroute\_urls = []

page\_number = 1

while True:

try:

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5)

ktcl\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

ktcl\_urls = [element.get\_attribute("href") for element in ktcl\_elements]

all\_ktclroute\_urls.extend(ktcl\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break

### **6. Extract Bus Details from Each URL**

bus\_details = []

for url in all\_ktclroute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5)

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2)

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

button.click()

time.sleep(2)

except Exception as e:

print(f"Failed to click expand button: {e}")

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

bus\_details.append({

'route\_name': route\_name,

'route\_link': url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue

df = pd.DataFrame(bus\_details)

print(df)

### **7. Close WebDriver**

finally: driver.quit()

## **Error Handling**

The script includes error handling to catch and report issues during navigation, URL processing, and data extraction. Exceptions are printed with descriptive messages, allowing for troubleshooting.

## **Notes**

1. **Page Structure Changes**: If the website structure changes, XPath or CSS selectors might need to be updated.
2. **Performance**: The script might need adjustments based on the website’s response time and the amount of data.
3. **Legal Considerations**: Ensure you comply with the website’s terms of service and scraping policies.

here, i paste 10 state code,

# bus 1 #step of Data Extract for Kadamba Transport Corporation

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

try:

# Open the main website

driver.get('https://www.redbus.in/online-booking/ktcl/?utm\_source=rtchometile')

# Wait for the body to load

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

# Scroll down to ensure all elements are loaded (optional)

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

# Initialize an empty list to store the URLs of detail pages

all\_ktclroute\_urls = []

# Adjusted to navigate through 4 pages

page\_number = 1

while True:

try:

# Wait for pagination container to load

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

# Find the next page button for current page\_number

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5) # Wait for the new page to load

# Extract URLs from current page

ktcl\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

ktcl\_urls = [element.get\_attribute("href") for element in ktcl\_elements]

all\_ktclroute\_urls.extend(ktcl\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break # Exit loop if next page button not found or any other error

# Initialize an empty list to store bus details

bus\_details = []

# Iterate through each URL and extract data from detail pages

for url in all\_ktclroute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5) # Adjust if needed

# Scroll down the page until no more content is loaded

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2) # Wait after clicking, adjust as necessary

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

button.click()

time.sleep(2)

except Exception as e:

print(f"Failed to click expand button: {e}")

# Wait for bus items to load

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

# Find and extract bus details

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

# Append bus details to bus\_details list

bus\_details.append({

'route\_name': route\_name,

'route\_link':url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue # Skip to the next bus element if extraction fails

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue # Skip to the next URL if processing fails

# Create a DataFrame from bus\_details list

df = pd.DataFrame(bus\_details)

print(df)

except Exception as e:

print(f"Error occurred: {e}")

finally:

# Close the WebDriver

driver.quit()

**Navigated to page 2**

**Navigated to page 3**

**Navigated to page 4**

**Failed to navigate to page 5: Message: no such element: Unable to locate element: {"method":"xpath","selector":".//div[contains(@class, "DC\_117\_pageTabs") and text()="5"]"}**

**(Session info: chrome=126.0.6478.185); For documentation on this error, please visit:** [**https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#no-such-element-exception**](https://www.selenium.dev/documentation/webdriver/troubleshooting/errors#no-such-element-exception)

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E110BBAC]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E110BA76]**

**(No symbol) [0x00007FF7E113D240]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**...**

**130 [41 Seats available\n19 Window] [4.4]**

**131 [39 Seats available\n19 Window] [4.4]**

**[132 rows x 10 columns]**

# bus-2 #Extract Data of HRTC(Himachal Road Transport Corporation)

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

try:

# Open the main website

driver.get('https://www.redbus.in/online-booking/hrtc/?utm\_source=rtchometile')

# Wait for the body to load

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

# Scroll down to ensure all elements are loaded (optional)

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

# Initialize an empty list to store the URLs of detail pages

all\_hrtcroute\_urls = []

# Adjusted to navigate through 4 pages

page\_number = 1

while True:

try:

# Wait for pagination container to load

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

# Find the next page button for current page\_number

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5) # Wait for the new page to load

# Extract URLs from current page

hrtc\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

hrtc\_urls = [element.get\_attribute("href") for element in hrtc\_elements]

all\_hrtcroute\_urls.extend(hrtc\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break # Exit loop if next page button not found or any other error

# Initialize an empty list to store bus details

bus\_details = []

# Iterate through each URL and extract data from detail pages

for url in all\_hrtcroute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5) # Adjust if needed

# Scroll down the page until no more content is loaded

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2) # Wait after clicking, adjust as necessary

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

# Wait for bus items to load

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

# Find and extract bus details

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

# Append bus details to bus\_details list

bus\_details.append({

'route\_name': route\_name,

'route\_link':url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue # Skip to the next bus element if extraction fails

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue # Skip to the next URL if processing fails

# Create a DataFrame from bus\_details list

hrtc = pd.DataFrame(bus\_details)

print(hrtc)

except Exception as e:

print(f"Error occurred: {e}")

finally:

# Close the WebDriver

driver.quit()

Navigated to page 2

Navigated to page 3

Navigated to page 4

Navigated to page 5

Failed to navigate to page 6: Message:

Stacktrace:

GetHandleVerifier [0x00007FF7E128EEB2+31554]

(No symbol) [0x00007FF7E1207EE9]

(No symbol) [0x00007FF7E10C872A]

(No symbol) [0x00007FF7E1118434]

(No symbol) [0x00007FF7E111853C]

(No symbol) [0x00007FF7E115F6A7]

(No symbol) [0x00007FF7E113D06F]

(No symbol) [0x00007FF7E115C977]

(No symbol) [0x00007FF7E113CDD3]

(No symbol) [0x00007FF7E110A33B]

(No symbol) [0x00007FF7E110AED1]

GetHandleVerifier [0x00007FF7E1598B2D+3217341]

GetHandleVerifier [0x00007FF7E15E5AF3+3532675]

GetHandleVerifier [0x00007FF7E15DB0F0+3489152]

GetHandleVerifier [0x00007FF7E133E786+750614]

(No symbol) [0x00007FF7E121376F]

(No symbol) [0x00007FF7E120EB24]

(No symbol) [0x00007FF7E120ECB2]

(No symbol) [0x00007FF7E11FE17F]

...

541 [10h 55m] [575] [31 Seats available\n11 Window] [3.8]

542 [10h 05m] [1078] [24 Seats available\n10 Window] [4.1]

[543 rows x 10 columns]

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

#bus-3 Extract Data from WBTC West Bengal

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

try:

# Open the main website

driver.get('https://www.redbus.in/online-booking/wbtc-ctc/?utm\_source=rtchometile')

# Wait for the body to load

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

# Scroll down to ensure all elements are loaded (optional)

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

# Initialize an empty list to store the URLs of detail pages

all\_wbtcroute\_urls = []

# Adjusted to navigate through 4 pages

page\_number = 1

while True:

try:

# Wait for pagination container to load

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

# Find the next page button for current page\_number

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5) # Wait for the new page to load

# Extract URLs from current page

wbtc\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

wbtc\_urls = [element.get\_attribute("href") for element in wbtc\_elements]

all\_wbtcroute\_urls.extend(wbtc\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break # Exit loop if next page button not found or any other error

# Initialize an empty list to store bus details

bus\_details = []

# Iterate through each URL and extract data from detail pages

for url in all\_wbtcroute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5) # Adjust if needed

# Scroll down the page until no more content is loaded

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2) # Wait after clicking, adjust as necessary

print("success")

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

# Wait for bus items to load

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

# Find and extract bus details

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

# Append bus details to bus\_details list

bus\_details.append({

'route\_name': route\_name,

'route\_link':url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue # Skip to the next bus element if extraction fails

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue # Skip to the next URL if processing fails

# Create a DataFrame from bus\_details list

wbtc = pd.DataFrame(bus\_details)

print(wbtc)

except Exception as e:

print(f"Error occurred: {e}")

finally:

# Close the WebDriver

driver.quit()

Navigated to page 2

Navigated to page 3

Navigated to page 4

Navigated to page 5

Failed to navigate to page 6: Message:

Stacktrace:

GetHandleVerifier [0x00007FF7E128EEB2+31554]

(No symbol) [0x00007FF7E1207EE9]

(No symbol) [0x00007FF7E10C872A]

(No symbol) [0x00007FF7E1118434]

(No symbol) [0x00007FF7E111853C]

(No symbol) [0x00007FF7E115F6A7]

(No symbol) [0x00007FF7E113D06F]

(No symbol) [0x00007FF7E115C977]

(No symbol) [0x00007FF7E113CDD3]

(No symbol) [0x00007FF7E110A33B]

(No symbol) [0x00007FF7E110AED1]

GetHandleVerifier [0x00007FF7E1598B2D+3217341]

GetHandleVerifier [0x00007FF7E15E5AF3+3532675]

GetHandleVerifier [0x00007FF7E15DB0F0+3489152]

GetHandleVerifier [0x00007FF7E133E786+750614]

(No symbol) [0x00007FF7E121376F]

(No symbol) [0x00007FF7E120EB24]

(No symbol) [0x00007FF7E120ECB2]

(No symbol) [0x00007FF7E11FE17F]

...

103 [04h 55m] [] [24 Seats available\n6 Single] [2.6]

104 [02h 05m] [899] [61 Seats available\n30 Window] [3.2]

[105 rows x 10 columns]

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

#bus-4 Extract Data from CTU RTC Chandigarh Transport Undertaking (CTU)

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

try:

# Open the main website

driver.get('https://www.redbus.in/online-booking/chandigarh-transport-undertaking-ctu')

# Wait for the body to load

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

# Scroll down to ensure all elements are loaded (optional)

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

# Initialize an empty list to store the URLs of detail pages

all\_cturoute\_urls = []

# Adjusted to navigate through 4 pages

page\_number = 1

while True:

try:

# Wait for pagination container to load

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

# Find the next page button for current page\_number

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5) # Wait for the new page to load

# Extract URLs from current page

ctu\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

ctu\_urls = [element.get\_attribute("href") for element in ctu\_elements]

all\_cturoute\_urls.extend(ctu\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break # Exit loop if next page button not found or any other error

# Initialize an empty list to store bus details

bus\_details = []

# Iterate through each URL and extract data from detail pages

for url in all\_cturoute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5) # Adjust if needed

# Scroll down the page until no more content is loaded

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2) # Wait after clicking, adjust as necessary

print("success")

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

# Wait for bus items to load

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

# Find and extract bus details

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

# Append bus details to bus\_details list

bus\_details.append({

'route\_name': route\_name,

'route\_link':url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue # Skip to the next bus element if extraction fails

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue # Skip to the next URL if processing fails

# Create a DataFrame from bus\_details list

ctu = pd.DataFrame(bus\_details)

print(ctu)

except Exception as e:

print(f"Error occurred: {e}")

finally:

# Close the WebDriver

driver.quit()

Navigated to page 2

Navigated to page 3

Navigated to page 4

Navigated to page 5

Failed to navigate to page 6: Message:

Stacktrace:

GetHandleVerifier [0x00007FF7E128EEB2+31554]

(No symbol) [0x00007FF7E1207EE9]

(No symbol) [0x00007FF7E10C872A]

(No symbol) [0x00007FF7E1118434]

(No symbol) [0x00007FF7E111853C]

(No symbol) [0x00007FF7E115F6A7]

(No symbol) [0x00007FF7E113D06F]

(No symbol) [0x00007FF7E115C977]

(No symbol) [0x00007FF7E113CDD3]

(No symbol) [0x00007FF7E110A33B]

(No symbol) [0x00007FF7E110AED1]

GetHandleVerifier [0x00007FF7E1598B2D+3217341]

GetHandleVerifier [0x00007FF7E15E5AF3+3532675]

GetHandleVerifier [0x00007FF7E15DB0F0+3489152]

GetHandleVerifier [0x00007FF7E133E786+750614]

(No symbol) [0x00007FF7E121376F]

(No symbol) [0x00007FF7E120EB24]

(No symbol) [0x00007FF7E120ECB2]

(No symbol) [0x00007FF7E11FE17F]

...

503 [05h 45m] [318] [33 Seats available\n12 Window] [3.9]

504 [05h 20m] [293] [37 Seats available\n13 Window] [3.9]

[505 rows x 10 columns]

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

#bus-5 Extract Data from PEPSU Punjab

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.support.ui import WebDriverWait

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

import time

import pandas as pd

# Initialize the WebDriver

driver = webdriver.Chrome()

wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits

try:

# Open the main website

driver.get('https://www.redbus.in/online-booking/pepsu/?utm\_source=rtchometile')

# Wait for the body to load

wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))

# Scroll down to ensure all elements are loaded (optional)

for \_ in range(3):

driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)

time.sleep(1)

# Initialize an empty list to store the URLs of detail pages

all\_pepsuroute\_urls = []

# Adjusted to navigate through 4 pages

page\_number = 1

while True:

try:

# Wait for pagination container to load

pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))

# Find the next page button for current page\_number

next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')

driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)

time.sleep(2)

next\_page\_button.click()

print(f'Navigated to page {page\_number + 1}')

time.sleep(5) # Wait for the new page to load

# Extract URLs from current page

pepsu\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')

pepsu\_urls = [element.get\_attribute("href") for element in pepsu\_elements]

all\_pepsuroute\_urls.extend(pepsu\_urls)

page\_number += 1

except Exception as e:

print(f"Failed to navigate to page {page\_number + 1}: {e}")

break # Exit loop if next page button not found or any other error

# Initialize an empty list to store bus details

bus\_details = []

# Iterate through each URL and extract data from detail pages

for url in all\_pepsuroute\_urls:

try:

driver.get(url)

driver.maximize\_window()

time.sleep(5) # Adjust if needed

# Scroll down the page until no more content is loaded

old\_page\_height = driver.execute\_script("return document.body.scrollHeight")

route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')

route\_name = route\_name\_element.text.strip()

print(f"Extracted Route Name: {route\_name}")

try:

button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))

#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking

button.click()

driver.execute\_script("arguments[0].scrollIntoView(true);", button)

time.sleep(2) # Wait after clicking, adjust as necessary

print("success")

except Exception as e:

print(f"Failed to click expand button: {e}")

while True:

driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")

time.sleep(2)

new\_page\_height = driver.execute\_script("return document.body.scrollHeight")

if new\_page\_height == old\_page\_height:

break

old\_page\_height = new\_page\_height

# Wait for bus items to load

bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))

for bus in bus\_container:

try:

# Find and extract bus details

bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")

bus\_names = [i.text for i in bus\_name]

bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")

bus\_types = [i.text for i in bus\_type]

starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")

starting\_times = [i.text for i in starting\_time]

ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")

ending\_times = [i.text for i in ending\_time]

bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")

time\_duration = [i.text for i in bus\_duration]

price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")

prices = [i.text for i in price]

seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")

seats = [i.text for i in seat]

star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")

star\_ratings = [i.text for i in star\_rating]

# Append bus details to bus\_details list

bus\_details.append({

'route\_name': route\_name,

'route\_link':url,

'bus\_name': bus\_names,

'bus\_type': bus\_types,

'departing\_time': starting\_times,

'reaching\_time': ending\_times,

'bus\_duration': time\_duration,

'price': prices,

'seat': seats,

'star\_rating': star\_ratings

})

except Exception as e:

print(f"Failed to extract details from bus: {e}")

continue # Skip to the next bus element if extraction fails

except Exception as e:

print(f"Failed to process URL {url}: {e}")

continue # Skip to the next URL if processing fails

# Create a DataFrame from bus\_details list

pepsu= pd.DataFrame(bus\_details)

print(pepsu)

except Exception as e:

print(f"Error occurred: {e}")

finally:

# Close the WebDriver

driver.quit()

**Navigated to page 2**

**Navigated to page 3**

**Failed to navigate to page 4: Message:**

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E115F6A7]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**(No symbol) [0x00007FF7E120ECB2]**

**(No symbol) [0x00007FF7E11FE17F]**

**BaseThreadInitThunk [0x00007FFB7E9F257D+29]**

**RtlUserThreadStart [0x00007FFB7F58AF28+40]**

**...**

**196 [44 Seats available\n22 Window] [2.7]**

**197 [31 Seats available] [4.5]**

**[198 rows x 10 columns]**

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...***

**#bus-6 Extract Data from NBSTC NORTH BENGAL STATE TRANSPORT CORPORATION**

**from selenium import webdriver**

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

**from selenium.webdriver.common.keys import Keys**

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

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

**import time**

**import pandas as pd**

**# Initialize the WebDriver**

**driver = webdriver.Chrome()**

**wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits**

**try:**

**# Open the main website**

**driver.get('https://www.redbus.in/online-booking/north-bengal-state-transport-corporation')**

**# Wait for the body to load**

**wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))**

**# Scroll down to ensure all elements are loaded (optional)**

**for \_ in range(3):**

**driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)**

**time.sleep(1)**

**# Initialize an empty list to store the URLs of detail pages**

**all\_nbsroute\_urls = []**

**# Adjusted to navigate through 4 pages**

**page\_number = 1**

**while True:**

**try:**

**# Wait for pagination container to load**

**pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))**

**# Find the next page button for current page\_number**

**next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')**

**driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)**

**time.sleep(2)**

**next\_page\_button.click()**

**print(f'Navigated to page {page\_number + 1}')**

**time.sleep(5) # Wait for the new page to load**

**# Extract URLs from current page**

**nbs\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')**

**nbs\_urls = [element.get\_attribute("href") for element in nbs\_elements]**

**all\_nbsroute\_urls.extend(nbs\_urls)**

**page\_number += 1**

**except Exception as e:**

**print(f"Failed to navigate to page {page\_number + 1}: {e}")**

**break # Exit loop if next page button not found or any other error**

**# Initialize an empty list to store bus details**

**bus\_details = []**

**# Iterate through each URL and extract data from detail pages**

**for url in all\_nbsroute\_urls:**

**try:**

**driver.get(url)**

**driver.maximize\_window()**

**time.sleep(5) # Adjust if needed**

**# Scroll down the page until no more content is loaded**

**old\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')**

**route\_name = route\_name\_element.text.strip()**

**print(f"Extracted Route Name: {route\_name}")**

**try:**

**button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))**

**#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking**

**button.click()**

**driver.execute\_script("arguments[0].scrollIntoView(true);", button)**

**time.sleep(2) # Wait after clicking, adjust as necessary**

**print("success")**

**except Exception as e:**

**print(f"Failed to click expand button: {e}")**

**while True:**

**driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")**

**time.sleep(2)**

**new\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**if new\_page\_height == old\_page\_height:**

**break**

**old\_page\_height = new\_page\_height**

**# Wait for bus items to load**

**bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))**

**for bus in bus\_container:**

**try:**

**# Find and extract bus details**

**bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")**

**bus\_names = [i.text for i in bus\_name]**

**bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")**

**bus\_types = [i.text for i in bus\_type]**

**starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")**

**starting\_times = [i.text for i in starting\_time]**

**ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")**

**ending\_times = [i.text for i in ending\_time]**

**bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")**

**time\_duration = [i.text for i in bus\_duration]**

**price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")**

**prices = [i.text for i in price]**

**seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")**

**seats = [i.text for i in seat]**

**star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")**

**star\_ratings = [i.text for i in star\_rating]**

**# Append bus details to bus\_details list**

**bus\_details.append({**

**'route\_name': route\_name,**

**'route\_lin':url,**

**'bus\_name': bus\_names,**

**'bus\_type': bus\_types,**

**'departing\_time': starting\_times,**

**'reaching\_time': ending\_times,**

**'bus\_duration': time\_duration,**

**'price': prices,**

**'seat': seats,**

**'star\_rating': star\_ratings**

**})**

**except Exception as e:**

**print(f"Failed to extract details from bus: {e}")**

**continue # Skip to the next bus element if extraction fails**

**except Exception as e:**

**print(f"Failed to process URL {url}: {e}")**

**continue # Skip to the next URL if processing fails**

**# Create a DataFrame from bus\_details list**

**nbs= pd.DataFrame(bus\_details)**

**print(nbs)**

**except Exception as e:**

**print(f"Error occurred: {e}")**

**finally:**

**# Close the WebDriver**

**driver.quit()**

**Navigated to page 2**

**Navigated to page 3**

**Navigated to page 4**

**Navigated to page 5**

**Failed to navigate to page 6: Message:**

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E115F6A7]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**(No symbol) [0x00007FF7E120ECB2]**

**(No symbol) [0x00007FF7E11FE17F]**

**...**

**162 [22 Seats available\n9 Window] [4.9]**

**163 [45 Seats available\n23 Window] [2.8]**

**[164 rows x 10 columns]**

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...***

**#bus-7 Extract Data from BSRTC Bihar state road transport corporation (BSRTC)**

**from selenium import webdriver**

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

**from selenium.webdriver.common.keys import Keys**

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

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

**import time**

**import pandas as pd**

**# Initialize the WebDriver**

**driver = webdriver.Chrome()**

**wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits**

**try:**

**# Open the main website**

**driver.get('https://www.redbus.in/online-booking/north-bengal-state-transport-corporation')**

**# Wait for the body to load**

**wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))**

**# Scroll down to ensure all elements are loaded (optional)**

**for \_ in range(3):**

**driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)**

**time.sleep(1)**

**# Initialize an empty list to store the URLs of detail pages**

**all\_bsrroute\_urls = []**

**# Adjusted to navigate through 4 pages**

**page\_number = 1**

**while True:**

**try:**

**# Wait for pagination container to load**

**pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))**

**# Find the next page button for current page\_number**

**next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')**

**driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)**

**time.sleep(2)**

**next\_page\_button.click()**

**print(f'Navigated to page {page\_number + 1}')**

**time.sleep(5) # Wait for the new page to load**

**# Extract URLs from current page**

**bsr\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')**

**bsr\_urls = [element.get\_attribute("href") for element in bsr\_elements]**

**all\_bsrroute\_urls.extend(bsr\_urls)**

**page\_number += 1**

**except Exception as e:**

**print(f"Failed to navigate to page {page\_number + 1}: {e}")**

**break # Exit loop if next page button not found or any other error**

**# Initialize an empty list to store bus details**

**bus\_details = []**

**# Iterate through each URL and extract data from detail pages**

**for url in all\_bsrroute\_urls:**

**try:**

**driver.get(url)**

**driver.maximize\_window()**

**time.sleep(5) # Adjust if needed**

**# Scroll down the page until no more content is loaded**

**old\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')**

**route\_name = route\_name\_element.text.strip()**

**print(f"Extracted Route Name: {route\_name}")**

**try:**

**button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))**

**#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking**

**button.click()**

**driver.execute\_script("arguments[0].scrollIntoView(true);", button)**

**time.sleep(2) # Wait after clicking, adjust as necessary**

**print("success")**

**except Exception as e:**

**print(f"Failed to click expand button: {e}")**

**while True:**

**driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")**

**time.sleep(2)**

**new\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**if new\_page\_height == old\_page\_height:**

**break**

**old\_page\_height = new\_page\_height**

**# Wait for bus items to load**

**bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))**

**for bus in bus\_container:**

**try:**

**# Find and extract bus details**

**bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")**

**bus\_names = [i.text for i in bus\_name]**

**bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")**

**bus\_types = [i.text for i in bus\_type]**

**starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")**

**starting\_times = [i.text for i in starting\_time]**

**ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")**

**ending\_times = [i.text for i in ending\_time]**

**bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")**

**time\_duration = [i.text for i in bus\_duration]**

**price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")**

**prices = [i.text for i in price]**

**seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")**

**seats = [i.text for i in seat]**

**star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")**

**star\_ratings = [i.text for i in star\_rating]**

**# Append bus details to bus\_details list**

**bus\_details.append({**

**'route\_name': route\_name,**

**'route\_link':url,**

**'bus\_name': bus\_names,**

**'bus\_type': bus\_types,**

**'departing\_time': starting\_times,**

**'reaching\_time': ending\_times,**

**'bus\_duration': time\_duration,**

**'price': prices,**

**'seat': seats,**

**'star\_rating': star\_ratings**

**})**

**except Exception as e:**

**print(f"Failed to extract details from bus: {e}")**

**continue # Skip to the next bus element if extraction fails**

**except Exception as e:**

**print(f"Failed to process URL {url}: {e}")**

**continue # Skip to the next URL if processing fails**

**# Create a DataFrame from bus\_details list**

**bsr= pd.DataFrame(bus\_details)**

**print(bsr)**

**except Exception as e:**

**print(f"Error occurred: {e}")**

**finally:**

**# Close the WebDriver**

**driver.quit()**

**Navigated to page 2**

**Navigated to page 3**

**Navigated to page 4**

**Navigated to page 5**

**Failed to navigate to page 6: Message:**

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E115F6A7]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**(No symbol) [0x00007FF7E120ECB2]**

**(No symbol) [0x00007FF7E11FE17F]**

**...**

**157 [22 Seats available\n9 Window] [4.9]**

**158 [45 Seats available\n23 Window] [2.8]**

**[159 rows x 10 columns]**

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...***

**#bus-8 Extract Data from South Bengal State Transport Corporation (SBSTC)**

**from selenium import webdriver**

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

**from selenium.webdriver.common.keys import Keys**

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

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

**import time**

**import pandas as pd**

**# Initialize the WebDriver**

**driver = webdriver.Chrome()**

**wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits**

**try:**

**# Open the main website**

**driver.get('https://www.redbus.in/online-booking/south-bengal-state-transport-corporation-sbstc/?utm\_source=rtchometile')**

**# Wait for the body to load**

**wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))**

**# Scroll down to ensure all elements are loaded (optional)**

**for \_ in range(3):**

**driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)**

**time.sleep(1)**

**# Initialize an empty list to store the URLs of detail pages**

**all\_sbtcroute\_urls = []**

**# Adjusted to navigate through 4 pages**

**page\_number = 1**

**while True:**

**try:**

**# Wait for pagination container to load**

**pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))**

**# Find the next page button for current page\_number**

**next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')**

**driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)**

**time.sleep(2)**

**next\_page\_button.click()**

**print(f'Navigated to page {page\_number + 1}')**

**time.sleep(5) # Wait for the new page to load**

**# Extract URLs from current page**

**sbtc\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')**

**sbtc\_urls = [element.get\_attribute("href") for element in sbtc\_elements]**

**all\_sbtcroute\_urls.extend(sbtc\_urls)**

**page\_number += 1**

**except Exception as e:**

**print(f"Failed to navigate to page {page\_number + 1}: {e}")**

**break # Exit loop if next page button not found or any other error**

**# Initialize an empty list to store bus details**

**bus\_details = []**

**# Iterate through each URL and extract data from detail pages**

**for url in all\_sbtcroute\_urls:**

**try:**

**driver.get(url)**

**driver.maximize\_window()**

**time.sleep(5) # Adjust if needed**

**# Scroll down the page until no more content is loaded**

**old\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')**

**route\_name = route\_name\_element.text.strip()**

**print(f"Extracted Route Name: {route\_name}")**

**try:**

**button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))**

**#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking**

**button.click()**

**driver.execute\_script("arguments[0].scrollIntoView(true);", button)**

**time.sleep(2) # Wait after clicking, adjust as necessary**

**print("success")**

**except Exception as e:**

**print(f"Failed to click expand button: {e}")**

**while True:**

**driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")**

**time.sleep(2)**

**new\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**if new\_page\_height == old\_page\_height:**

**break**

**old\_page\_height = new\_page\_height**

**# Wait for bus items to load**

**bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))**

**for bus in bus\_container:**

**try:**

**# Find and extract bus details**

**bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")**

**bus\_names = [i.text for i in bus\_name]**

**bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")**

**bus\_types = [i.text for i in bus\_type]**

**starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")**

**starting\_times = [i.text for i in starting\_time]**

**ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")**

**ending\_times = [i.text for i in ending\_time]**

**bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")**

**time\_duration = [i.text for i in bus\_duration]**

**price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")**

**prices = [i.text for i in price]**

**seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")**

**seats = [i.text for i in seat]**

**star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")**

**star\_ratings = [i.text for i in star\_rating]**

**# Append bus details to bus\_details list**

**bus\_details.append({**

**'route\_name': route\_name,**

**'route\_link':url,**

**'bus\_name': bus\_names,**

**'bus\_type': bus\_types,**

**'departing\_time': starting\_times,**

**'reaching\_time': ending\_times,**

**'bus\_duration': time\_duration,**

**'price': prices,**

**'seat': seats,**

**'star\_rating': star\_ratings**

**})**

**except Exception as e:**

**print(f"Failed to extract details from bus: {e}")**

**continue # Skip to the next bus element if extraction fails**

**except Exception as e:**

**print(f"Failed to process URL {url}: {e}")**

**continue # Skip to the next URL if processing fails**

**# Create a DataFrame from bus\_details list**

**sbtc= pd.DataFrame(bus\_details)**

**print(sbtc)**

**except Exception as e:**

**print(f"Error occurred: {e}")**

**finally:**

**# Close the WebDriver**

**driver.quit()**

**Navigated to page 2**

**Navigated to page 3**

**Navigated to page 4**

**Navigated to page 5**

**Failed to navigate to page 6: Message:**

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E115F6A7]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**(No symbol) [0x00007FF7E120ECB2]**

**(No symbol) [0x00007FF7E11FE17F]**

**...**

**220 [05h 00m] [190] [52 Seats available\n20 Window] [3.6]**

**221 [05h 00m] [190] [52 Seats available\n20 Window] [2.8]**

**[222 rows x 10 columns]**

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings…***

***#bus-9 Extract Data from jksrtc jammu kashmir road***

***from selenium import webdriver***

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

***from selenium.webdriver.common.keys import Keys***

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

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

***import time***

***import pandas as pd***

***# Initialize the WebDriver***

***driver = webdriver.Chrome()***

***wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits***

***try:***

***# Open the main website***

***driver.get('https://www.redbus.in/online-booking/jksrtc')***

***# Wait for the body to load***

***wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))***

***# Scroll down to ensure all elements are loaded (optional)***

***for \_ in range(3):***

***driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)***

***time.sleep(1)***

***# Initialize an empty list to store the URLs of detail pages***

***all\_jksrtcroute\_urls = []***

***# Adjusted to navigate through 4 pages***

***page\_number = 1***

***while True:***

***try:***

***# Wait for pagination container to load***

***pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))***

***# Find the next page button for current page\_number***

***next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')***

***driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)***

***time.sleep(2)***

***next\_page\_button.click()***

***print(f'Navigated to page {page\_number + 1}')***

***time.sleep(5) # Wait for the new page to load***

***# Extract URLs from current page***

***jksrtc\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')***

***jksrtc\_urls = [element.get\_attribute("href") for element in jksrtc\_elements]***

***all\_jksrtcroute\_urls.extend(jksrtc\_urls)***

***page\_number += 1***

***except Exception as e:***

***print(f"Failed to navigate to page {page\_number + 1}: {e}")***

***break # Exit loop if next page button not found or any other error***

***# Initialize an empty list to store bus details***

***bus\_details = []***

***# Iterate through each URL and extract data from detail pages***

***for url in all\_jksrtcroute\_urls:***

***try:***

***driver.get(url)***

***driver.maximize\_window()***

***time.sleep(5) # Adjust if needed***

***# Scroll down the page until no more content is loaded***

***old\_page\_height = driver.execute\_script("return document.body.scrollHeight")***

***route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')***

***route\_name = route\_name\_element.text.strip()***

***print(f"Extracted Route Name: {route\_name}")***

***try:***

***button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))***

***#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking***

***button.click()***

***driver.execute\_script("arguments[0].scrollIntoView(true);", button)***

***time.sleep(2) # Wait after clicking, adjust as necessary***

***print("success")***

***except Exception as e:***

***print(f"Failed to click expand button: {e}")***

***while True:***

***driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")***

***time.sleep(2)***

***new\_page\_height = driver.execute\_script("return document.body.scrollHeight")***

***if new\_page\_height == old\_page\_height:***

***break***

***old\_page\_height = new\_page\_height***

***# Wait for bus items to load***

***bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))***

***for bus in bus\_container:***

***try:***

***# Find and extract bus details***

***bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")***

***bus\_names = [i.text for i in bus\_name]***

***bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")***

***bus\_types = [i.text for i in bus\_type]***

***starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")***

***starting\_times = [i.text for i in starting\_time]***

***ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")***

***ending\_times = [i.text for i in ending\_time]***

***bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")***

***time\_duration = [i.text for i in bus\_duration]***

***price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")***

***prices = [i.text for i in price]***

***seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")***

***seats = [i.text for i in seat]***

***star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")***

***star\_ratings = [i.text for i in star\_rating]***

***# Append bus details to bus\_details list***

***bus\_details.append({***

***'route\_name': route\_name,***

***'route\_url':url,***

***'bus\_name': bus\_names,***

***'bus\_type': bus\_types,***

***'departing\_time': starting\_times,***

***'reaching\_time': ending\_times,***

***'bus\_duration': time\_duration,***

***'price': prices,***

***'seat': seats,***

***'star\_rating': star\_ratings***

***})***

***except Exception as e:***

***print(f"Failed to extract details from bus: {e}")***

***continue # Skip to the next bus element if extraction fails***

***except Exception as e:***

***print(f"Failed to process URL {url}: {e}")***

***continue # Skip to the next URL if processing fails***

***# Create a DataFrame from bus\_details list***

***jksrtc= pd.DataFrame(bus\_details)***

***print(jksrtc)***

***except Exception as e:***

***print(f"Error occurred: {e}")***

***finally:***

***# Close the WebDriver***

***driver.quit()***

***Navigated to page 2***

***Failed to navigate to page 3: Message:***

***Stacktrace:***

***GetHandleVerifier [0x00007FF7E128EEB2+31554]***

***(No symbol) [0x00007FF7E1207EE9]***

***(No symbol) [0x00007FF7E10C872A]***

***(No symbol) [0x00007FF7E1118434]***

***(No symbol) [0x00007FF7E111853C]***

***(No symbol) [0x00007FF7E115F6A7]***

***(No symbol) [0x00007FF7E113D06F]***

***(No symbol) [0x00007FF7E115C977]***

***(No symbol) [0x00007FF7E113CDD3]***

***(No symbol) [0x00007FF7E110A33B]***

***(No symbol) [0x00007FF7E110AED1]***

***GetHandleVerifier [0x00007FF7E1598B2D+3217341]***

***GetHandleVerifier [0x00007FF7E15E5AF3+3532675]***

***GetHandleVerifier [0x00007FF7E15DB0F0+3489152]***

***GetHandleVerifier [0x00007FF7E133E786+750614]***

***(No symbol) [0x00007FF7E121376F]***

***(No symbol) [0x00007FF7E120EB24]***

***(No symbol) [0x00007FF7E120ECB2]***

***(No symbol) [0x00007FF7E11FE17F]***

***BaseThreadInitThunk [0x00007FFB7E9F257D+29]***

***RtlUserThreadStart [0x00007FFB7F58AF28+40]***

***...***

***47 [05h 35m] [329] [34 Seats available\n18 Window] [2.8]***

***48 [05h 25m] [899] [24 Seats available\n15 Window] [3.9]***

***49 [04h 48m] [999] [38 Seats available\n5 Single] [2.5]***

***50 [04h 30m] [] [30 Seats available\n20 Window] [2.2]***

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...***

**#bus-10 Extract Data from jAssam State Transport Corporation (ASTC)**

**from selenium import webdriver**

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

**from selenium.webdriver.common.keys import Keys**

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

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

**import time**

**import pandas as pd**

**# Initialize the WebDriver**

**driver = webdriver.Chrome()**

**wait = WebDriverWait(driver, 10) # 10 seconds timeout for explicit waits**

**try:**

**# Open the main website**

**driver.get('https://www.redbus.in/online-booking/astc/?utm\_source=rtchometile')**

**# Wait for the body to load**

**wait.until(EC.presence\_of\_element\_located((By.TAG\_NAME, 'body')))**

**# Scroll down to ensure all elements are loaded (optional)**

**for \_ in range(3):**

**driver.find\_element(By.TAG\_NAME, 'body').send\_keys(Keys.PAGE\_DOWN)**

**time.sleep(1)**

**# Initialize an empty list to store the URLs of detail pages**

**all\_astcroute\_urls = []**

**# Adjusted to navigate through 4 pages**

**page\_number = 1**

**while True:**

**try:**

**# Wait for pagination container to load**

**pagination\_container = wait.until(EC.presence\_of\_element\_located((By.XPATH, '//\*[@id="root"]/div/div[4]/div[12]')))**

**# Find the next page button for current page\_number**

**next\_page\_button = pagination\_container.find\_element(By.XPATH, f'.//div[contains(@class, "DC\_117\_pageTabs") and text()="{page\_number+1}"]')**

**driver.execute\_script("arguments[0].scrollIntoView(true);", next\_page\_button)**

**time.sleep(2)**

**next\_page\_button.click()**

**print(f'Navigated to page {page\_number + 1}')**

**time.sleep(5) # Wait for the new page to load**

**# Extract URLs from current page**

**astc\_elements = driver.find\_elements(By.XPATH, '//a[@class="route"]')**

**astc\_urls = [element.get\_attribute("href") for element in astc\_elements]**

**all\_astcroute\_urls.extend(astc\_urls)**

**page\_number += 1**

**except Exception as e:**

**print(f"Failed to navigate to page {page\_number + 1}: {e}")**

**break # Exit loop if next page button not found or any other error**

**# Initialize an empty list to store bus details**

**bus\_details = []**

**# Iterate through each URL and extract data from detail pages**

**for url in all\_astcroute\_urls:**

**try:**

**driver.get(url)**

**driver.maximize\_window()**

**time.sleep(5) # Adjust if needed**

**# Scroll down the page until no more content is loaded**

**old\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**route\_name\_element = driver.find\_element(By.XPATH, '//h1[@class="D136\_h1"]')**

**route\_name = route\_name\_element.text.strip()**

**print(f"Extracted Route Name: {route\_name}")**

**try:**

**button = wait.until(EC.element\_to\_be\_clickable((By.XPATH, '//div[@class="button" and text() ="View Buses"]')))**

**#driver.execute\_script("arguments[0].scrollIntoView(true);", button) # Scroll into view if necessary # Optional: Wait a bit before clicking**

**button.click()**

**driver.execute\_script("arguments[0].scrollIntoView(true);", button)**

**time.sleep(2) # Wait after clicking, adjust as necessary**

**print("success")**

**except Exception as e:**

**print(f"Failed to click expand button: {e}")**

**while True:**

**driver.execute\_script("window.scrollTo(0, document.body.scrollHeight);")**

**time.sleep(2)**

**new\_page\_height = driver.execute\_script("return document.body.scrollHeight")**

**if new\_page\_height == old\_page\_height:**

**break**

**old\_page\_height = new\_page\_height**

**# Wait for bus items to load**

**bus\_container = wait.until(EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "bus-item")))**

**for bus in bus\_container:**

**try:**

**# Find and extract bus details**

**bus\_name = bus.find\_elements(By.XPATH, ".//div[@class='travels lh-24 f-bold d-color']")**

**bus\_names = [i.text for i in bus\_name]**

**bus\_type = bus.find\_elements(By.XPATH, ".//div[@class='bus-type f-12 m-top-16 l-color evBus']")**

**bus\_types = [i.text for i in bus\_type]**

**starting\_time = bus.find\_elements(By.XPATH, ".//div[@class='dp-time f-19 d-color f-bold']")**

**starting\_times = [i.text for i in starting\_time]**

**ending\_time = bus.find\_elements(By.XPATH, ".//div[@class='bp-time f-19 d-color disp-Inline']")**

**ending\_times = [i.text for i in ending\_time]**

**bus\_duration = bus.find\_elements(By.XPATH, ".//div[@class='dur l-color lh-24']")**

**time\_duration = [i.text for i in bus\_duration]**

**price = bus.find\_elements(By.XPATH, ".//span[@class='f-19 f-bold']")**

**prices = [i.text for i in price]**

**seat = bus.find\_elements(By.XPATH, ".//div[@class='column-eight w-15 fl']")**

**seats = [i.text for i in seat]**

**star\_rating = bus.find\_elements(By.XPATH, ".//div[@class='rating-sec lh-24']")**

**star\_ratings = [i.text for i in star\_rating]**

**# Append bus details to bus\_details list**

**bus\_details.append({**

**'route\_name': route\_name,**

**'route\_url':url,**

**'bus\_name': bus\_names,**

**'bus\_type': bus\_types,**

**'departing\_time': starting\_times,**

**'reaching\_time': ending\_times,**

**'bus\_duration': time\_duration,**

**'price': prices,**

**'seat': seats,**

**'star\_rating': star\_ratings**

**})**

**except Exception as e:**

**print(f"Failed to extract details from bus: {e}")**

**continue # Skip to the next bus element if extraction fails**

**except Exception as e:**

**print(f"Failed to process URL {url}: {e}")**

**continue # Skip to the next URL if processing fails**

**# Create a DataFrame from bus\_details list**

**astc= pd.DataFrame(bus\_details)**

**print(astc)**

**except Exception as e:**

**print(f"Error occurred: {e}")**

**finally:**

**# Close the WebDriver**

**driver.quit()**

**Navigated to page 2**

**Navigated to page 3**

**Navigated to page 4**

**Navigated to page 5**

**Failed to navigate to page 6: Message:**

**Stacktrace:**

**GetHandleVerifier [0x00007FF7E128EEB2+31554]**

**(No symbol) [0x00007FF7E1207EE9]**

**(No symbol) [0x00007FF7E10C872A]**

**(No symbol) [0x00007FF7E1118434]**

**(No symbol) [0x00007FF7E111853C]**

**(No symbol) [0x00007FF7E115F6A7]**

**(No symbol) [0x00007FF7E113D06F]**

**(No symbol) [0x00007FF7E115C977]**

**(No symbol) [0x00007FF7E113CDD3]**

**(No symbol) [0x00007FF7E110A33B]**

**(No symbol) [0x00007FF7E110AED1]**

**GetHandleVerifier [0x00007FF7E1598B2D+3217341]**

**GetHandleVerifier [0x00007FF7E15E5AF3+3532675]**

**GetHandleVerifier [0x00007FF7E15DB0F0+3489152]**

**GetHandleVerifier [0x00007FF7E133E786+750614]**

**(No symbol) [0x00007FF7E121376F]**

**(No symbol) [0x00007FF7E120EB24]**

**(No symbol) [0x00007FF7E120ECB2]**

**(No symbol) [0x00007FF7E11FE17F]**

**...**

**92 [8 Seats available\n2 Single] [4.8]**

**93 [22 Seats available\n11 Window] [2.8]**

**[94 rows x 10 columns]**

***Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...***

**all\_bus = pd.concat([df, hrtc, wbtc, ctu, pepsu, nbs, bsr, sbtc, jksrtc, astc], axis=0, ignore\_index=True)**

The code you provided is used to concatenate multiple Pandas DataFrames into a single DataFrame.

csv\_filename = 'redbus.csv'

all\_bus.to\_csv(csv\_filename, index=False)

The code you provided is used to save a Pandas DataFrame (all\_bus) to a CSV file

all\_bus\_cleaned = all\_bus.where(pd.notnull(all\_bus), None)

The line of code you provided is used to replace missing values (NaNs) in a Pandas DataFrame with None.

**all\_bus**:

* This is the original DataFrame that you want to clean.

**pd.notnull(all\_bus)**:

* pd.notnull() is a Pandas function that returns a boolean DataFrame where each cell is True if the corresponding value in all\_bus is not null (i.e., not NaN), and False if the value is null (i.e., NaN).
* Essentially, it creates a DataFrame of the same shape as all\_bus, but with True where the values are not null and False where the values are null.

**all\_bus.where(condition, value)**:

* where() is a Pandas method that replaces values in a DataFrame based on a condition.
* **condition**: The condition is a DataFrame of booleans that determines which values to keep.
* **value**: The value to use where the condition is False. In this case, it’s None.

**None**:

* This is the value that will replace NaN values in the DataFrame where the condition is False.

2**.Data Storage:**

1. Connects to a MySQL Database: Establishes a connection to a MySQL database using pymysql.
2. Creates a Table: Defines and executes a SQL query to create a table if it doesn't already exist.
3. Inserts Data into the Table: Iterates over a DataFrame, processes each row, and inserts it into the MySQL table.

**1. Connecting to the Database**

**import pymysql**

**connection = pymysql.connect(host='127.0.0.1',**

**user='root',**

**password='Madhu@29',**

**database='redbus',**

**charset='utf8mb4',**

**cursorclass=pymysql.cursors.DictCursor)**

**host='127.0.0.1': The IP address of the MySQL server (localhost in this case).**

**user='root': The username for MySQL authentication.**

**password='Madhu@29': The password for the MySQL user.**

**database='redbus': The name of the database to connect to.**

**charset='utf8mb4': Specifies the character set to use for the connection.**

**cursorclass=pymysql.cursors.DictCursor: Specifies that the cursor will return results as dictionaries, where column names are keys.**

### **2. Creating a Table**

**cursor = connection.cursor()**

**create\_table\_query = """**

**CREATE TABLE IF NOT EXISTS bus\_values5(**

**route\_name TEXT,**

**route\_link TEXT,**

**bus\_names TEXT,**

**bus\_types TEXT,**

**departing\_time TIME,**

**reaching\_time TIME,**

**bus\_duration TEXT,**

**prices DECIMAL(10, 2),**

**seats INT,**

**star\_ratings FLOAT**

**)**

**"""**

**CREATE TABLE IF NOT EXISTS bus\_values5**: Creates a table named bus\_values5 if it doesn’t already exist.

**Columns**:

* route\_name, route\_link, bus\_names, bus\_types, bus\_duration (TEXT): Textual data.
* departing\_time, reaching\_time (TIME): Time data.
* prices (DECIMAL(10, 2)): Decimal data with up to 10 digits and 2 decimal places.
* seats (INT): Integer data.
* star\_ratings (FLOAT): Floating-point data.

try:

cursor.execute(create\_table\_query)

print("Table 'bus\_values5' created successfully.")

except pymysql.Error as e:

print("Error creating table:", e)

**cursor.execute(create\_table\_query)**: Executes the SQL query to create the table.

insert\_query = """

INSERT INTO bus\_values5

(route\_name, route\_link, bus\_names, bus\_types, departing\_time, reaching\_time, bus\_duration, prices, seats, star\_ratings)

VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)

"""

**INSERT INTO bus\_values5 ... VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)**: Defines an SQL query for inserting data into the table. %s placeholders are used for parameterized queries to prevent SQL injection.

bus\_values5 = all\_bus\_cleaned.values.tolist() # Assuming all\_bus\_cleaned is a pandas DataFrame

**all\_bus\_cleaned.values.tolist()**: Converts the DataFrame all\_bus\_cleaned into a list of lists, where each inner list represents a row of data.for row in bus\_values5:

try:

if len(row[7]) > 0: # Check if row[7] has elements

prices = float(row[7][0].strip("[]")) # Convert prices to float

else:

prices = None # Handle case where prices are not available

seats = int(row[8]) # Convert seats to int

if len(row[9]) > 0: # Check if row[9] has elements

star\_ratings = float(row[9][0].strip("[]")) # Convert star\_ratings to float

else:

star\_ratings = None # Handle case where star\_ratings are not available

cursor.execute(insert\_query, (

row[0], # route\_name

row[1], # route\_link

row[2], # bus\_names

row[3], # bus\_types

row[4], # departing\_time

row[5], # reaching\_time

row[6], # bus\_duration

prices, # prices

seats, # seats

star\_ratings # star\_ratings

))

print("Inserted row successfully:", row[0])

except pymysql.Error as e:

print("Error inserting data:", e)

connection.commit()

**Loop Through Data**: Iterates over each row in bus\_values5.

**Data Conversion**:

* Converts prices and star\_ratings from strings to floats, handling cases where they might be missing or formatted differently.
* Converts seats to an integer.

**Execute Insert Query**: Executes the insert\_query with the row data.

**Error Handling**: Prints an error message if the insertion fails.

**connection.commit()**: Commits the transaction, saving all the changes made to the database during this session.

import pymysql

connection = pymysql.connect(host='127.0.0.1',

user='root',

password='Madhu@29',

database='redbus',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

cursor = connection.cursor()

create\_table\_query = """

CREATE TABLE IF NOT EXISTS bus\_values5(

route\_name TEXT,

route\_link TEXT,

bus\_names TEXT,

bus\_types TEXT,

departing\_time TIME,

reaching\_time TIME,

bus\_duration TEXT,

prices DECIMAL(10, 2),

seats INT,

star\_ratings FLOAT

)

"""

try:

cursor.execute(create\_table\_query)

print("Table 'bus\_values5' created successfully.")

except pymysql.Error as e:

print("Error creating table:", e)

insert\_query = """

INSERT INTO bus\_values5

(route\_name, route\_link, bus\_names, bus\_types, departing\_time, reaching\_time, bus\_duration, prices, seats, star\_ratings)

VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)

"""

bus\_values5 = all\_bus\_cleaned.values.tolist() # Assuming all\_bus\_cleaned is a pandas DataFrame

for row in bus\_values5:

try:

if len(row[7]) > 0: # Check if row[7] has elements

prices = float(row[7][0].strip("[]")) # Convert prices to float

else:

prices = None # Handle case where prices are not available

seats = int(row[8]) # Convert seats to int

if len(row[9]) > 0: # Check if row[9] has elements

star\_ratings = float(row[9][0].strip("[]")) # Convert star\_ratings to float

else:

star\_ratings = None # Handle case where star\_ratings are not available

cursor.execute(insert\_query, (

row[0], # route\_name

row[1], # route\_link

row[2], # bus\_names

row[3], # bus\_types

row[4], # departing\_time

row[5], # reaching\_time

row[6], # bus\_duration

prices, # prices

seats, # seats

star\_ratings # star\_ratings

))

print("Inserted row successfully:", row[0])

except pymysql.Error as e:

print("Error inserting data:", e)

connection.commit()

**3. Streamlit Application:**

* + Develop a Streamlit application to display and filter the scraped data.
  + Implement various filters such as bustype, route, price range, star rating, availability.

### **1. Imports and Database Connection**

import mysql.connector

import pandas as pd

import streamlit as st

**mysql.connector**: A library for connecting to MySQL databases.

**pandas**: A data manipulation library, used here for handling and analyzing data.

**streamlit**: A library for building web applications with Python.

# Connect to MySQL

conn = mysql.connector.connect(

user='root',

password='Madhu@29',

host='localhost',

database='redbus'

)

* **Establishes a connection** to the MySQL database with the credentials and database name provided.

### **2. Fetch Data from MySQL Database**

# Query data

def fetch\_data():

query = "SELECT \* FROM bus\_values5;" # Replace 'bus\_values5' with your actual table name

df = pd.read\_sql\_query(query, conn)

return df

# Fetch data from SQL database

df = fetch\_data()

**fetch\_data function**: Executes a SQL query to fetch all data from the bus\_values5 table and loads it into a Pandas DataFrame.

**df**: Stores the DataFrame returned by fetch\_data

### **3. Filter Data Based on User Inputs**

# Function to filter data based on user inputs

def filter\_data(df, bus\_types, route\_name, prices, star\_ratings ):

if not bus\_types:

bus\_types = df['bus\_types'].unique()

if not route\_name:

route\_name = df['route\_name'].unique()

#if not seats:

#seats = df['seats'].unique()

filtered\_df = df[

(df['bus\_types'].isin(bus\_types)) &

(df['route\_name'].isin(route\_name)) &

(df['prices'].between(prices[0], prices[1])) &

(df['star\_ratings'] >= star\_ratings)

#(df['seats'].isin(seats))

]

return filtered\_df

**filter\_data function**: Filters the DataFrame based on the user inputs.

* **bus\_types**: Filters by selected bus types.
* **route\_name**: Filters by selected routes.
* **prices**: Filters by price range (using the between method).
* **star\_ratings**: Filters by minimum star rating.
* **Commented out**: Code to filter by availability (seats) is present but not used.

### **4. Streamlit Application**

# Streamlit application starts here

def main():

st.title('Redbus Data Analysis')

# Sidebar filters

st.sidebar.header('Filters')

# Multi-select for bus types

selected\_bustypes = st.sidebar.multiselect('Select Bus Types', df['bus\_types'].unique())

# Multi-select for routes

selected\_routes = st.sidebar.multiselect('Select Routes', df['route\_name'].unique())

# Price range slider

min\_price, max\_price = float(df['prices'].min()), float(df['prices'].max())

price\_range = st.sidebar.slider('Price Range', min\_price, max\_price, (min\_price, max\_price))

# Star rating filter

min\_star\_rating, max\_star\_rating = float(df['star\_ratings'].min()), float(df['star\_ratings'].max())

star\_rating = st.sidebar.slider('Minimum Star Rating', min\_star\_rating, max\_star\_rating, min\_star\_rating)

# Checkbox for availability

# seats\_options = ['Yes', 'No']

#selected\_availability = st.sidebar.multiselect('Available Buses', seats\_options)

# Print filter values for debugging

#st.write("Bus Types Selected:", selected\_bustypes)

#st.write("Routes Selected:", selected\_routes)

#st.write("Price Range Selected:", price\_range)

#st.write("Star Rating Selected:", star\_rating)

# Apply filters and display results

filtered\_data = filter\_data(df, selected\_bustypes, selected\_routes, price\_range, star\_rating)

st.subheader('Filtered Bus Tickets')

st.dataframe(filtered\_data)

### **5. Close the Database Connection**

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

main()

# Close the connection when done

conn.close()

**if \_\_name\_\_ == '\_\_main\_\_': main()**: Runs the main function if the script is executed directly.

**conn.close()**: Closes the database connection when the script finishes executing

Here, full code.

import mysql.connector

import pandas as pd

import streamlit as st

# Connect to MySQL

conn = mysql.connector.connect(

user='root',

password='Madhu@29',

host='localhost',

database='redbus'

)

# Query data

def fetch\_data():

query = "SELECT \* FROM bus\_values5;" # Replace 'bus\_values5' with your actual table name

df = pd.read\_sql\_query(query, conn)

return df

# Fetch data from SQL database

df = fetch\_data()

# Function to filter data based on user inputs

def filter\_data(df, bus\_types, route\_name, prices, star\_ratings ):

if not bus\_types:

bus\_types = df['bus\_types'].unique()

if not route\_name:

route\_name = df['route\_name'].unique()

#if not seats:

#seats = df['seats'].unique()

filtered\_df = df[

(df['bus\_types'].isin(bus\_types)) &

(df['route\_name'].isin(route\_name)) &

(df['prices'].between(prices[0], prices[1])) &

(df['star\_ratings'] >= star\_ratings)

#(df['seats'].isin(seats))

]

return filtered\_df

# Streamlit application starts here

def main():

st.title('Redbus Data Analysis')

# Sidebar filters

st.sidebar.header('Filters')

# Multi-select for bus types

selected\_bustypes = st.sidebar.multiselect('Select Bus Types', df['bus\_types'].unique())

# Multi-select for routes

selected\_routes = st.sidebar.multiselect('Select Routes', df['route\_name'].unique())

# Price range slider

min\_price, max\_price = float(df['prices'].min()), float(df['prices'].max())

price\_range = st.sidebar.slider('Price Range', min\_price, max\_price, (min\_price, max\_price))

# Star rating filter

min\_star\_rating, max\_star\_rating = float(df['star\_ratings'].min()), float(df['star\_ratings'].max())

star\_rating = st.sidebar.slider('Minimum Star Rating', min\_star\_rating, max\_star\_rating, min\_star\_rating)

# Checkbox for availability

# seats\_options = ['Yes', 'No']

#selected\_availability = st.sidebar.multiselect('Available Buses', seats\_options)

# Print filter values for debugging

#st.write("Bus Types Selected:", selected\_bustypes)

#st.write("Routes Selected:", selected\_routes)

#st.write("Price Range Selected:", price\_range)

#st.write("Star Rating Selected:", star\_rating)

# Apply filters and display results

filtered\_data = filter\_data(df, selected\_bustypes, selected\_routes, price\_range, star\_rating)

st.subheader('Filtered Bus Tickets')

st.dataframe(filtered\_data)

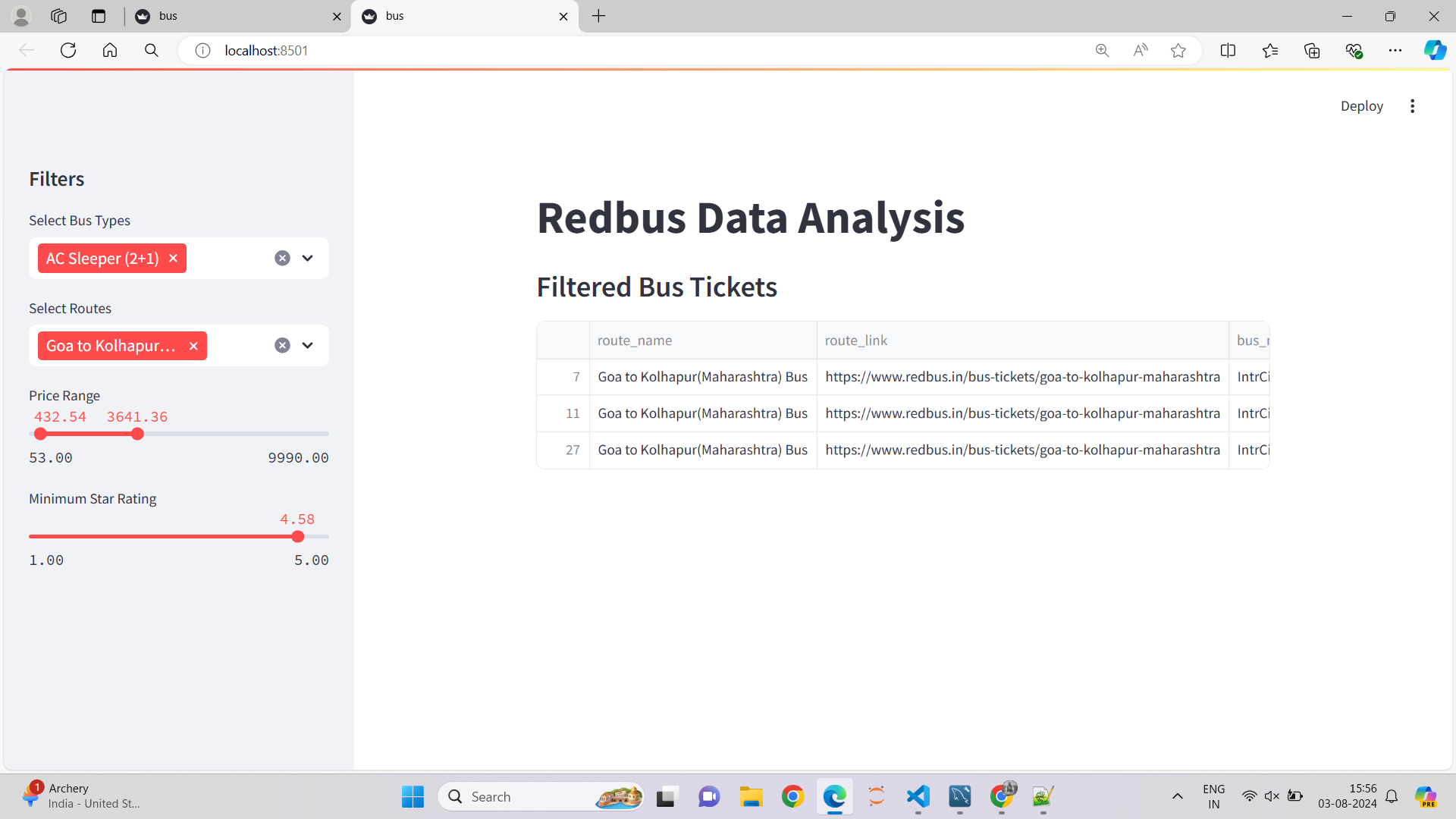
if \_\_name\_\_ == '\_\_main\_\_':

main()

# Close the connection when done

conn.close()

**Output:**

****