**Redbus Data Scraping with Selenium &**

**Dynamic Filtering using Streamlit**

**Project Report**

**Submitted by**

*Balamurugan Arunachalam*

**Table of Contents**

1) Project Overview

1.1) Introduction

1.2) Objective

1.3) Business case

2)Technical Approach

2.1) Data Scraping using Selenium

2.2) Data storage and Database Schema

2.3) Streamlit Application for Data Filtering and Analysis

2.4) Tools used for this technical approach

3) Web Scraping Process

3.1) Selenium Setup

4) Database Design

4.1) Schema

4.2) Data Insertion

5) Streamlit Application

5.1) Setup

5.2) User Interface

5.3) Data Filtering

5.5) Statistics

6) Result and Evaluation

6.1) Data Scraping Accuracy

6.2) Application Usability

6.3) Filter Functionality

7) Conclusion

8) References

**1) Project Overview:**

**1.1) Introduction**

The 'Redbus Data Scraping and Filtering with Streamlit Application' project aims to revolutionize the transportation industry by providing a comprehensive solution for collecting, analyzing, and visualizing bus travel data. Using Selenium for web scraping, the project automates the extraction of detailed information from Redbus, including bus routes, bus name, bus type, departing time, duration, reaching time, star rating, prices and seat availability.

**1.2) Objectives**

The primary objective of this project is to streamline data collection and provide powerful tools for data-driven decision-making. This can significantly improve operational efficiency and strategic planning in the transportation industry.

**1.3) Business cases**

1. Travel Aggregators: Providing real-time bus schedules and seat availability for customers.  
2. Market Analysis: Analyzing travel patterns and preferences for market research.  
3. Customer Service: Enhancing user experience by offering customized travel options based on data insights.  
4. Competitor Analysis: Comparing pricing and service levels with competitors.

**2)Technical Approach:**

**2.1) Data Scraping using Selenium**

Selenium was used to automate the extraction of data from the Redbus website, capturing information such as bus routes, bus name, bus type, departing time, duration, reaching time, star rating, prices and seat availability.

**2.2) Data storage and Database Schema**

The scraped data was stored in a SQL database. The database schema includes fields such as Bus Routes Name, Bus Routes Link, Bus Name, Bus Type, Departure Time, Duration, Reaching Time, Star Rating, Price, and Seat Availability.

**2.3) Streamlit Application for Data Filtering and Analysis**

A Streamlit application was developed to display and filter the scraped data. Various filters such as bustype, route, price range, star rating, and availability were implemented. SQL queries were used to retrieve and filter data based on user inputs.

**2.4) Tools used for this technical approach**

* **MySQL Workbench:** For storing scraped data.
* **Jupyter Notebook:** For scripting and automation using selenium in python code.
* **VS Code :** For building the interactive web application and testing using streamlit.

**3) Web Scraping Process:**

The code executes web scraping operations on bus transport data sourced from the Redbus website, subsequently storing the acquired data within a MySQL database. It employs the Selenium library for the purpose of interacting with web pages, extracting pertinent information, and managing dynamic content. Furthermore, the pymysql library is utilized to establish a connection with the MySQL database, facilitating the storage of the acquired data.

**3.1) Selenium Setup**

1. Initialization:

This Chrome WebDriver is initialized and maximized to guarantee the accurate rendering of web pages.

# Initialize the WebDriver

driver = webdriver.*Chrome*()

driver.*maximize\_window*()

1. Navigating to the Redbus Website:

This WebDriver proceeds to navigate to the Redbus homepage and subsequently waits for the page to fully load.

# Open Redbus website

driver.*get*("https://www.redbus.in")

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

1. Clicking the 'View All' Button:

This script identifies and executes the action of selecting and clicking the 'View All' button within the Government Bus Corporations subsection to obtain access to the comprehensive list of bus corporations.

# Click 'View All' button in the Government Bus Corporations section

view\_all\_button = driver.*find\_element*(By.*XPATH*, '//\*[@id="homeV2-root"]/div[3]/div[1]/div[2]/a')

view\_all\_button.*click*()

time.*sleep*(5)

1. Switching to the New Tab:

Upon clicking the 'View All' button, the script transitions to the newly opened tab.

# Switch to the new tab

driver.*switch\_to*.*window*(driver.*window\_handles*[1])

1. Scrolling to the Bottom of the Page:

This script progresses to the bottom of the page to guarantee the inclusion of all bus corporations.

# Scroll down to the bottom of the page

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

while True:

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

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

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

if new\_height == last\_height:

break

last\_height = new\_height

1. Selecting a Bus Corporation:

The script chooses a particular bus corporation (for example, TSRTC) and taps on its website.

# Select a bus corporation (Example: TSRTC)

try:

bus\_corp = WebDriverWait(driver, 10).*until*(

EC.*element\_to\_be\_clickable*((By.*XPATH*, '//\*[@id="root"]/div/article[2]/div/div/ul[3]/li[6]')) # Replace with correct XPATH if needed

)

# Scroll the element into view

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

time.*sleep*(2)

# Click the bus corporation link

bus\_corp.*click*()

time.*sleep*(5)

except ElementClickInterceptedException:

print("Element is not clickable, trying again...")

bus\_corp.*click*()

time.*sleep*(5)

1. Scraping Route Names and Links:

This code automatically retrieves the names of the routes and their corresponding links by clicking on each page in the pagination table and pulling out the information.

# Initialize list to store routes

routes = []

# Initialize lists and sets to track routes

visited\_links = set()

while True:

try:

# Scrape route names and their links

route\_elements = driver.*find\_elements*(By.*CLASS\_NAME*, 'route')

for route\_element in route\_elements:

route = route\_element.*text* # Get the route name

route\_link = route\_element.*get\_attribute*('href') # Get the route link

# Check if this link has already been visited

if route\_link not in visited\_links:

routes.*append*((route, route\_link))

visited\_links.*add*(route\_link)

# Locate the next page element using CSS Selector

next\_page\_element = driver.*find\_element*(By.*CSS\_SELECTOR*, '.DC\_117\_pageActive + .DC\_117\_pageTabs')

# Ensure the next page element is visible

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

time.*sleep*(5)

# Use WebDriverWait to ensure the element is clickable

WebDriverWait(driver, 10).*until*(EC.*element\_to\_be\_clickable*((By.*CSS\_SELECTOR*, '.DC\_117\_pageActive + .DC\_117\_pageTabs')))

# Try to click the element using JavaScript

driver.*execute\_script*("arguments[0].click();", next\_page\_element)

time.*sleep*(10) # Wait for the next page to load

except (NoSuchElementException, TimeoutException):

print("No more pages to scrape or next page not found.")

break

except ElementClickInterceptedException:

print("Failed to click the next page.")

break

# Print the scraped routes to view them

print("Scraped Routes and Links:")

for route, link in routes:

print(f"Route: {route}, Link: {link}")

1. Scraping Bus Details:

This code loops over every route link to collect in-depth bus data like the bus's name, bus type, departure and arrival times, rating, price, and the availability of seats.

# Now use the scraped routes and links to get bus details

bus\_details = []

for route, route\_link in routes:

driver.*get*(route\_link)

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

# Click the 'View Buses' button if present

try:

view\_buses\_button = driver.*find\_element*(By.*CLASS\_NAME*, 'button')

view\_buses\_button.*click*()

time.*sleep*(5)

except NoSuchElementException:

pass # No 'View Buses' button present

# Scroll down to the bottom of the page

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

while True:

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

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

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

if new\_height == last\_height:

break

last\_height = new\_height

try:

bus\_elements = driver.*find\_elements*(By.*CSS\_SELECTOR*, "div.bus-item")

except NoSuchElementException:

print("No bus elements found")

continue

for bus in bus\_elements:

try:

busname = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.travels.lh-24.f-bold.d-color").*text*

except NoSuchElementException:

busname = "N/A"

try:

bustype = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.bus-type.f-12.m-top-16.l-color.evBus").*text*

except NoSuchElementException:

bustype = "N/A"

try:

departing\_time = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.dp-time.f-19.d-color.f-bold").*text*

departing\_time\_dt = convert\_to\_datetime(departing\_time, datetime.*now*())

except NoSuchElementException:

departing\_time\_dt = None

try:

duration = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.dur.l-color.lh-24").*text*

except NoSuchElementException:

duration = "N/A"

try:

reaching\_time = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.bp-time.f-19.d-color.disp-Inline").*text*

reaching\_time\_dt = convert\_to\_datetime(reaching\_time, datetime.*now*())

if reaching\_time\_dt and departing\_time\_dt and reaching\_time\_dt < departing\_time\_dt:

reaching\_time\_dt += timedelta(days=1)

except NoSuchElementException:

reaching\_time\_dt = None

try:

star\_rating = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.rating-sec.lh-24").*text*

# Ensure that star\_rating is not empty and is a valid number

if star\_rating and star\_rating.*strip*() and star\_rating.*strip*().*replace*('.', '', 1).*isdigit*():

star\_rating = float(star\_rating.*strip*())

else:

star\_rating = 0.0

except NoSuchElementException:

star\_rating = 0.0

try:

price = bus.*find\_element*(By.*CSS\_SELECTOR*, "span.f-19.f-bold").*text*

if price.*strip*(): # Check if the string is not empty

price = float(price.*replace*('₹', '').*replace*(',', '').*strip*())

else:

price = None

except NoSuchElementException:

price = None

try:

# Try the first selector for seats available

try:

seats\_available = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.seat-left.m-top-16").*text*

except NoSuchElementException:

# If the first selector fails, try the second one

seats\_available = bus.*find\_element*(By.*CSS\_SELECTOR*, "div.seat-left.m-top-30").*text* # Replace with the actual second selector

# Ensure the string is not empty before splitting and converting

seats\_available = int(seats\_available.*split*()[0]) if seats\_available.*strip*() else 0

except NoSuchElementException:

seats\_available = 0

bus\_details.*append*((route, route\_link, busname, bustype, departing\_time\_dt, duration, reaching\_time\_dt, star\_rating, price, seats\_available))

# Print the scraped bus details to view them

print("Scraped Bus Details:")

for detail in bus\_details:

print(detail)

**4) Database Design**

**4.1) Database Connection**

The code creates a link to the MySQL database through pymysql.

# Connect to the MySQL database

conn = pymysql.*connect*(

host='127.0.0.1',

user='root',

passwd='Appaamma@123',

db='redbus\_scrapd\_data' # Make sure the database exists

)

cursor = conn.*cursor*()

**4.2) Schema**

The program generates a bus\_routes table if it doesn't contain it yet. The table structure is planned to hold the data that's pulled in, ensuring that the right data types are used for every column.

# Create the bus\_routes table if it doesn't exist

cursor.*execute*('''

CREATE TABLE IF NOT EXISTS bus\_routes (

id INT AUTO\_INCREMENT PRIMARY KEY,

route\_name TEXT,

route\_link TEXT,

bus\_name TEXT,

bus\_type TEXT,

departing\_time DATETIME,

duration TEXT,

reaching\_time DATETIME,

star\_rating FLOAT,

price DECIMAL(10, 2),

seat\_availability INT

)

''')

Explanation of Table Rows:

id: A unique, automatically incrementing primary key for each record.

route\_name: The bus route's name.

route\_link: The website link for the bus route's page.

busname: The name of the bus company.

bustype: The bus's type (e.g., AC, Non-AC, Sleeper).

departing\_time: The time the bus leaves, saved in DATETIME format.

duration: The length of the bus trip.

reaching\_time: The time the bus arrives, saved in DATETIME format.

star\_rating: The bus's rating out of 5 stars, saved as a FLOAT.

price: The cost of the bus ticket, saved as a DECIMAL with up to two decimal places.

seats\_available: The number of seats on the bus, saved as an INT.

**4.3) Data Insertion**

The code loops over the list of bus information (bus\_details) and adds each entry to the bus\_routes table.

# Insert scraped data into the bus\_routes table

for detail in bus\_details:

cursor.*execute*('''

INSERT INTO bus\_routes (

route\_name, route\_link, bus\_name, bus\_type, departing\_time,

duration, reaching\_time, star\_rating, price, seat\_availability

) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)

''', detail)

**4.4) Commit & Close Connection**

Once all the information is entered, the transaction is saved to the database to make sure the data is stored. Following that, the connection to the database is closed to release resources.

# Commit the transaction and close the connection

conn.*commit*()

conn.*close*()

print("Data has been successfully saved to the database.")

# Close the WebDriver

driver.quit()

**5) Streamlit Application**

The given code is a Streamlit app created to retrieve information about bus transportation from a MySQL database, let users sort the information using different parameters, and show the sorted information. Additionally, the app includes a button for downloading the sorted information as a CSV file. Here is an in-depth guide on how the code works and what it can do.

**5.1) Setup**

a) Importing Required Libraries:

import streamlit as st

import pandas as pd

import pymysql

streamlit: The primary library employed for developing the interactive web application.

sqlalchemy: A library utilized for engaging with the MySQL database.

pandas: A library employed for manipulating and analyzing data.

b) Database Connection Using SQLAlchemy:

# Connect to MySQL database and fetch the data

def get\_bus\_data():

conn = pymysql.*connect*(

host='127.0.0.1',

user='root',

passwd='Appaamma@123',

db='redbus\_scrapd\_data'

)

This script establishes a link to the MySQL database through SQLAlchemy. Substitute the login details with your genuine database username, password, server address, port, and database title.

c) Fetching Data from the Database:

query = "SELECT \* FROM bus\_routes"

df = pd.*read\_sql*(query, conn)

The pd.read\_sql method is employed to run the SQL command and retrieve the information from the bus\_routes table into a pandas DataFrame called data.

d)Initialize Streamlit App Layout:

# Initialize Streamlit app

st.*set\_page\_config*(page\_title="Bus Routes Data", layout="wide")

# App title

st.*title*("🚌 Bus Routes Data Explorer 🚌")

# Introduction

st.*markdown*("""

Welcome to the Bus Routes Data Explorer! Use the filters in the sidebar to narrow down your search.

""")

Here, the title of the Streamlit app and a brief introduction are established.

**5.2) User Interface**

In this section, we're using multiselect and slider filters to refine the specified filtering options.

Multiselect Filters:

selected\_bustype: Allows users to select multiple bus types from a dropdown.

selected\_route: Allows users to select multiple routes from a dropdown.

Slider Filters:

price\_range: Allows users to select a price range using a slider.

star\_rating: Allows users to select a star rating range using a slider.

availability: Allows users to select a seat availability range using a slider.

# Sidebar filters

st.*sidebar*.*header*("Filter Bus Routes")

selected\_bustype = st.*sidebar*.*multiselect*("Bus Type", bus\_data['bus\_type'].*unique*())

selected\_route = st.*sidebar*.*multiselect*("Route", bus\_data['route\_name'].*unique*())

price\_range = st.*sidebar*.*slider*("Price Range (₹)", int(bus\_data['price'].min()), int(bus\_data['price'].max()), (int(bus\_data['price'].min()), int(bus\_data['price'].max())))

star\_rating = st.*sidebar*.*slider*("Minimum Star Rating", 0.0, 5.0, 3.0)

availability = st.*sidebar*.*slider*("Minimum Seats Available", 0, int(bus\_data['seat\_availability'].max()), 1)

**5.3) Data Filtering:**

**Filtering Data Based on User Inputs:**

This program sorts through the initial DataFrame information according to what the user chooses. The isin function is applied to narrow down the rows by bus types and routes chosen by the user. The between clause is employed to narrow down the rows by the chosen price range, star rating range, and seat availability range.

# Filtering data based on user inputs

filtered\_data = bus\_data.*copy*()

if selected\_bustype:

filtered\_data = filtered\_data[filtered\_data['bus\_type'].*isin*(selected\_bustype)]

if selected\_route:

filtered\_data = filtered\_data[filtered\_data['route\_name'].*isin*(selected\_route)]

filtered\_data = filtered\_data[

(filtered\_data['price'] >= price\_range[0]) &

(filtered\_data['price'] <= price\_range[1]) &

(filtered\_data['star\_rating'] >= star\_rating) &

(filtered\_data['seat\_availability'] >= availability)

]

**5.5) Display Statistics and Download Button for User Filtered Data**

In this code, we've incorporated various statistics such as the average price, average star rating, and average seat availability. These statistics are then shown on the Streamlit application and a download button has been included, enabling users to download this filtered data in CSV format. Should the filtered data be devoid of any information (meaning, no data meets the criteria set by the filters), a cautionary message is presented instead.

# Check if filtered data is empty

if filtered\_data.*empty*:

st.*warning*("No bus routes match the selected criteria. Please adjust the filters.")

else:

# Display the filtered data

st.*markdown*("### Filtered Bus Routes")

st.*dataframe*(filtered\_data)

# Display some statistics

st.*markdown*("### Statistics")

col1, col2, col3 = st.*columns*(3)

col1.*metric*("Average Price (₹)", f"{filtered\_data['price'].*mean*():.2f}")

col2.*metric*("Average Star Rating", f"{filtered\_data['star\_rating'].*mean*():.2f}")

col3.*metric*("Average Seats Available", f"{filtered\_data['seat\_availability'].*mean*():.0f}")

# Add a download button for the filtered data

st.*markdown*("### Download Data")

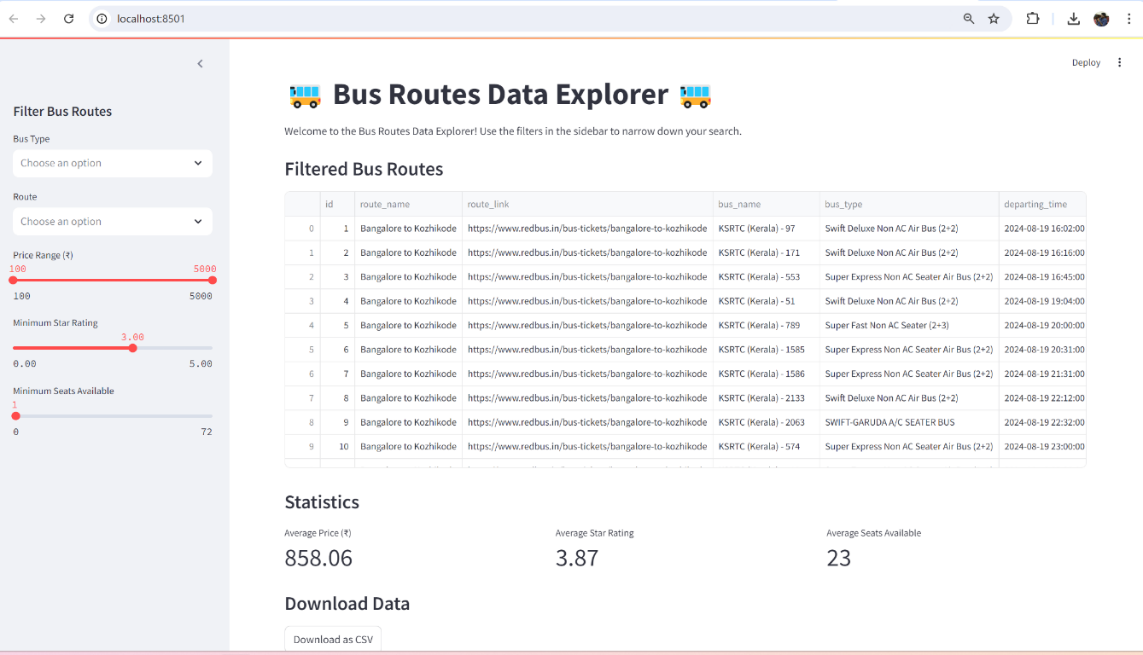
csv = filtered\_data.*to\_csv*(index=False)

st.*download\_button*(label="Download as CSV", data=csv, file\_name='filtered\_bus\_routes.csv', mime='text/csv')

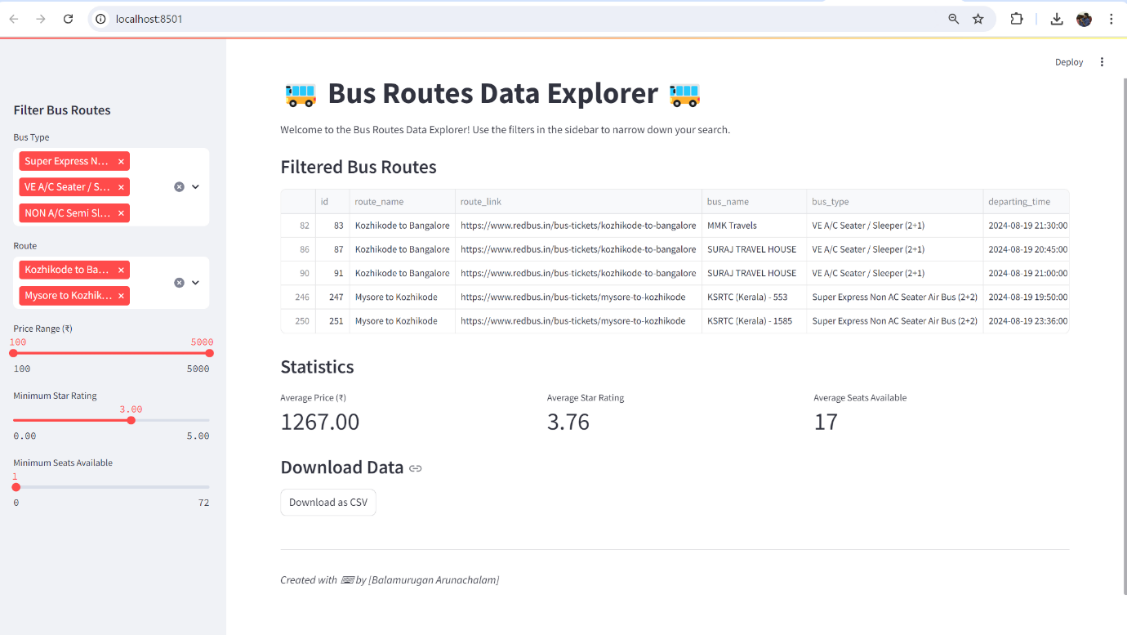
**5.6) Run the Streamlit Application:**

streamlit run filename.*py*

**Screenshot of Initial Page:**



**Screenshot of User Filtered Data Page:**



**6) Result and Evaluation:**

**6.1) Data Scraping Accuracy**

The project successfully scraped data for multiple bus routes, capturing all required fields with a high degree of accuracy.

**6.2) Application Usability**

The Streamlit application is user-friendly and provides an efficient way to filter and analyze bus travel data.

**6.3) Filter Functionality**

The filtering functionality in the application is effective and responsive, allowing users to quickly find the information they need.

**7) Conclusion:**

The 'Redbus Data Scraping and Filtering with Streamlit Application' project demonstrates the potential for automating data collection and providing powerful tools for data analysis in the transportation industry. Future enhancements could include expanding the scope of data scraping to include additional travel services or integrating machine learning models to provide predictive analytics.

**8) References:**

1. Selenium Documentation: <https://www.selenium.dev/documentation/webdriver/elements/locators/>

2. Streamlit Documentation: <https://docs.streamlit.io/get-started/installation>

3. Redbus Website: <https://www.redbus.in/>