**Red Bus Project**

**Problem Statement:**

The "Redbus Data Scraping and Filtering with Stream lit Application" aims to revolutionize the transportation industry by providing a comprehensive solution for collecting, analyzing, and visualizing bus travel data. By utilizing Selenium for web scraping, this project automates the extraction of detailed information from Redbus, including bus routes, schedules, prices, and seat availability. By streamlining data collection and providing powerful tools for data-driven decision-making, this project can significantly improve operational efficiency and strategic planning in the transportation industry.

**Scope/ Aim:**

Webscrape the Redbus website and store data in SQL DB. To create the Streamlit application to view the filtered.

* Elements used in the project:
* Browser- Chrome
* DB- Mysql
* Webscraping method- Selenium
* APP creation- Streamlit

**.ipynb File:**

This file aims for

1. Selenium automation to extract data from the redbus website for government and private bus details for all the links for the government state transports.
2. Connect to DB to create a database and table. Store the previously fetched data into the created tables for future querying purpose by multiple apps.

|  |  |
| --- | --- |
| Cell | Explanation |
| 1 | Install the selenium module |
| 2 | Instantiate the driver with the chrome browser and open it.  For atleast 10 state transport links iterate the following   * Go to Redbus website. * Once the page is loaded use CSS selector with name “.rtcBack:nth-child("**+**str(x)**+**") .rtcName” is used to search the website for the 10 different links to traverse to it by clicking it. #### x is the loop variable  1. Select it and then extract names and its available links.   Once the page is loaded use find\_elements on the page with the class name ‘route’. Extract the name and links using get\_attribute title and name respectively.  Store these values as dictionary in route\_attributes list   1. go to the next page and repeat step (a)   For pagination, use CSS selector with name “.DC\_117\_pageTabs:nth-child("+str(i)+")” to traverse through the page links by clicking it. #### i is the loop variable  Close the browser. |
| 3 | Initialize redbus\_data to empty list. Cell to iterate and get the bus details for each link got above.   1. Instantiate the browser 2. Get the name and link for each list element , and open it in the browser 3. Check if there are state government transport buses available and unhide the buses.   Use CSS selector with name ".group-data:nth-child("**+**str(x)**+**") .button" to unhide the bus options by clicking it. #### x is the loop variable   1. The page is scrolled down to the end of the page to load all the available bus options(both gov. and private).   window.scrollTo(0, document.body.scrollHeight)script is executed to scroll the page down.  document.body.scrollHeight value is got by executing the script and getting the current page height.  This step is looped until the previous height is same as the current height   1. All the available bus options are available as Unordered list items.These elements are found and stored in the li\_elements list.   find\_elements(By**.**CSS\_SELECTOR,'.bus-items li[id]')is used to search these list items using css\_selector with name bus-items.   1. If the li\_elements is not empty the following steps are performed for each of the items in the list.   Extract find\_element info for CSS\_SELECTOR with names travels,bus-type, dp-time,dur,bp-time,rating-sec,column-seven .f-bold,seat-left for the bus\_name,bus\_type,departing time, duration,reaching\_time,star\_rating,price,seat\_left  Add all these data as dictionary into the redbus\_data list   1. Close the browser and go to set (i) if not done with all the extracted links. |
| 4 | Cells 4-7 are optional, to write the route details and bus details into respective files. |
| 5 | Cell 8 involves installing the mysql-connector-python module |
| 6 | Cell 9 , Import and establish the connection using the respective connection details  (username,password,host) |
| 7 | Cell 10, create a new db called ‘Capstone’ and use cell 10 to check whether such a database is created |
| 8 | Cell 12, Use Capstone database and check whether the appropriate db is used cell 13 |
| 9 | Create a table bus\_routes with the following schema inorder to confine to the given project document  id INT AUTO\_INCREMENT PRIMARY KEY, route\_name VARCHAR(255),route\_link VARCHAR(255),busname VARCHAR(255),bustype VARCHAR(255),departing\_time time,duration VARCHAR(255),reaching\_time . star\_rating DECIMAL(2,1),price DECIMAL(10,2),seats\_available INT  Use cell 15, to view the created table scheme for bus\_routes |
| 10 | *Cell 16,17 are optional. I have used these to create route\_details table (having id INT AUTO\_INCREMENT PRIMARY KEY,*  *route\_name VARCHAR(255),route\_link VARCHAR(255)columns). I have later used this table in page 1 in the streamlit app to display all the link details for all the routes whether buses are available or not.* |
| 11 | Cells 18,20 are used to insert data from the redbus\_data and route\_attributes list into the bus\_routes and route\_details dynamically using for loop.  Once insertion is done, the transactions are committed. |
| 12 | Cells 19,21 are used to view the inserted data in the corresponding tables |

**.py files:**

Intially in the terminal, install the needed modules.

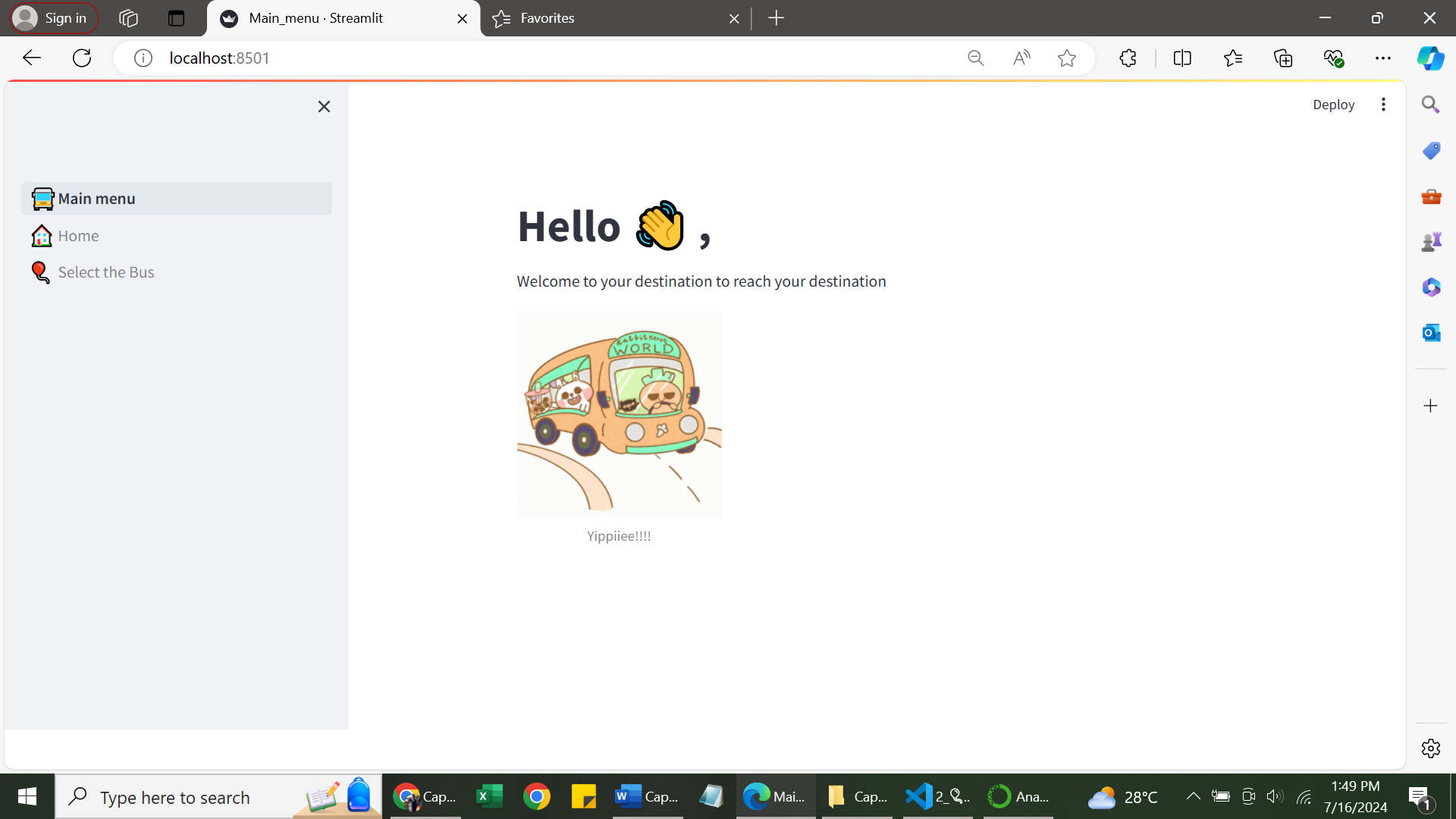
🡪pip install streamlit # This is used to download the streamlit library to create the app

🡪This code uses pymysql to connect to mysql db. Use "pip install PyMySQL" to install it.

[🚍Main\_menu.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/%F0%9F%9A%8DMain_menu.py)

The above file is the main script that has to be run in the terminal to open the streamlit application. (use command ‘streamlit run [🚍Main\_menu.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/%F0%9F%9A%8DMain_menu.py)’)

* Streamlit has been imported to access the streamlit functionalities.
* A simple welcome message and picture (travel.gif- provided in the [Redbus\_Streamlit](https://github.com/Ranjiny/Capstone_Redbus/tree/main/Redbus_Streamlit)) has been added.



Pages folder:

Pages folder contains 2 files [1\_🏠Home.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/pages/1_%F0%9F%8F%A0Home.py) , [2\_🎈Select\_the\_Bus.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/pages/2_%F0%9F%8E%88Select_the_Bus.py) corresponding to the page 1 and page 2 in the streamlit app.

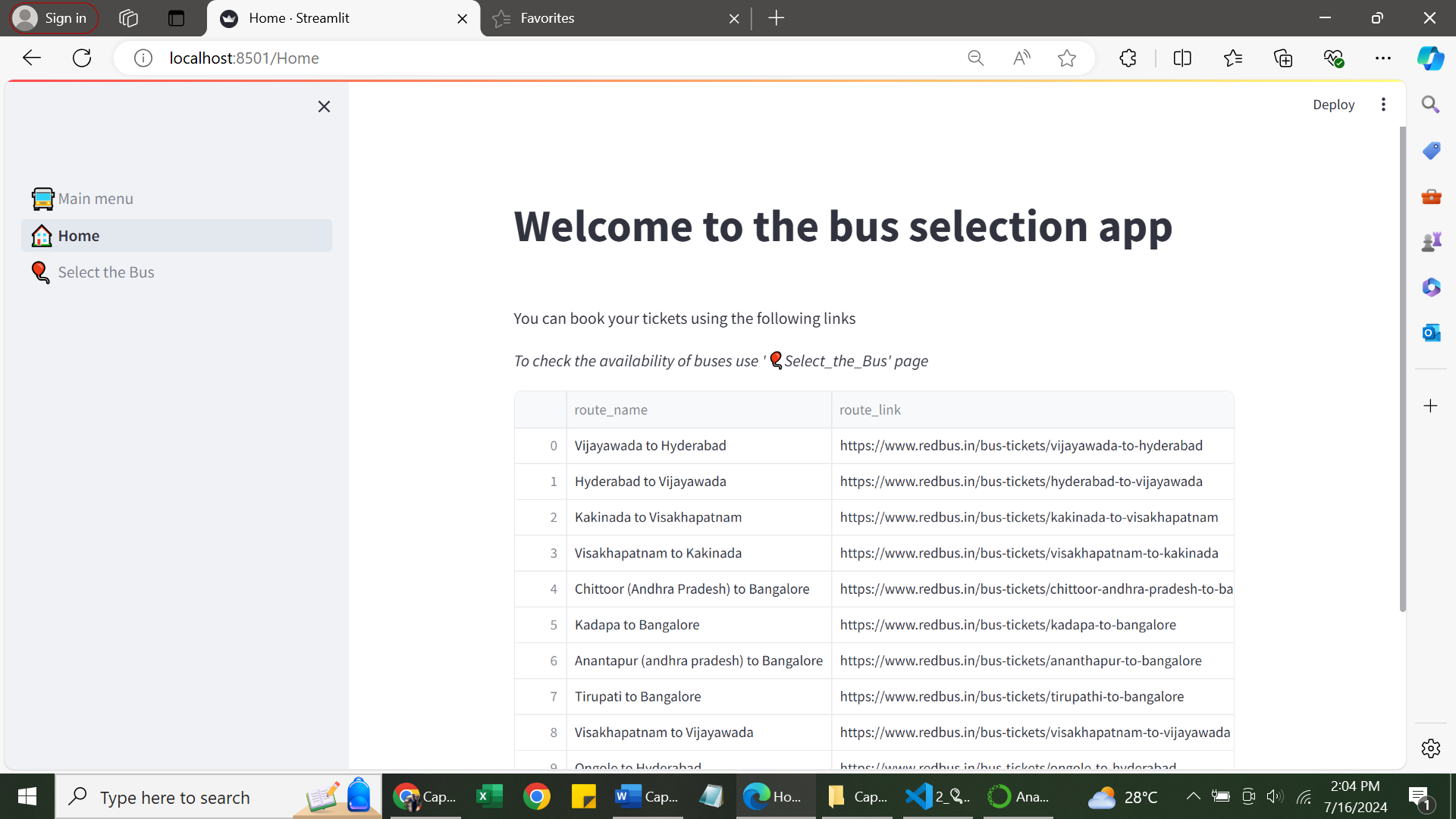
[1\_🏠Home.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/pages/1_%F0%9F%8F%A0Home.py):

This page simply displays an introductory note for the booking links for the various routes.

* Import streamlit and sqlalchemy (streamlit for building app , sqlalchemy for connect to db and populate the table with query results)
* Use st to write a note on the display screen
* Kindly incorporate the appropriate username, db,password , host while creating engine create\_engine(‘dialect+driver://username:password@host:port/database ') to create an engine to execute the sql.
* Metadata and table are assigned with the corresponding values.
* The necessary columns to be extracted are written in the following sqlalchemy format

*query = select([table.c.route\_name.distinct(), table.c.route\_link])*

* Execute the query and fetch all distinct values using the *connection.execute(query).fetchall()* command
* Display the fetched resulted in the dataframe using st.dataframe()



[2\_🎈Select\_the\_Bus.py](https://github.com/Ranjiny/Capstone_Redbus/blob/main/Redbus_Streamlit/pages/2_%F0%9F%8E%88Select_the_Bus.py)

This page simply displays the various buses for the different selection options like route name, bus type, AC type, Ratings and Fare range

* Import streamlit, datetime and sqlalchemy (streamlit for building app , sqlalchemy for connect to db and populate the table with query results)
* Kindly incorporate the appropriate username, db,password , host while creating engine create\_engine(‘dialect+driver://username:password@host:port/database ') to create an engine to execute the sql.
* Metadata and table are assigned with the corresponding values.
* Select Options:
* Routes select options

Fetch data using sqlalchemy to display the different route\_names(table.c.route\_name.distinct()).

Execute the query and fetch the first element as the result returned is a tuple.

* Select options for starting time

Use datetime.time(hour=h).strftime('%H:%M') to create a list comprehension for starting time from and to values.To this append 23:00-00:00 also.

* Select options for Ratings

str(r-1)+' to '+str(r) for r in range(5,0,-1) is used to create the list comprehension for the ratings range (4 to 5 , 3 to 4,…….0 to 1)

* Select options for Fare range

[str(fare)+' to '+str(fare+100) for fare in range(0,1000,100)] is used to create the list comprehension for the ratings range (0-100,100-200….900-1000) and append ‘Others’

* Create a container with 2 rows each having 3 columns

Container 1 with 3 options for where clause

* Option 1 : Pass the Routes select options to the routes select box
* Option 2: ('Seater','Sleeper','Others') pass these options for Seat Type select box
* Option 3: ('AC','NON AC') pass these options for AC Type select box

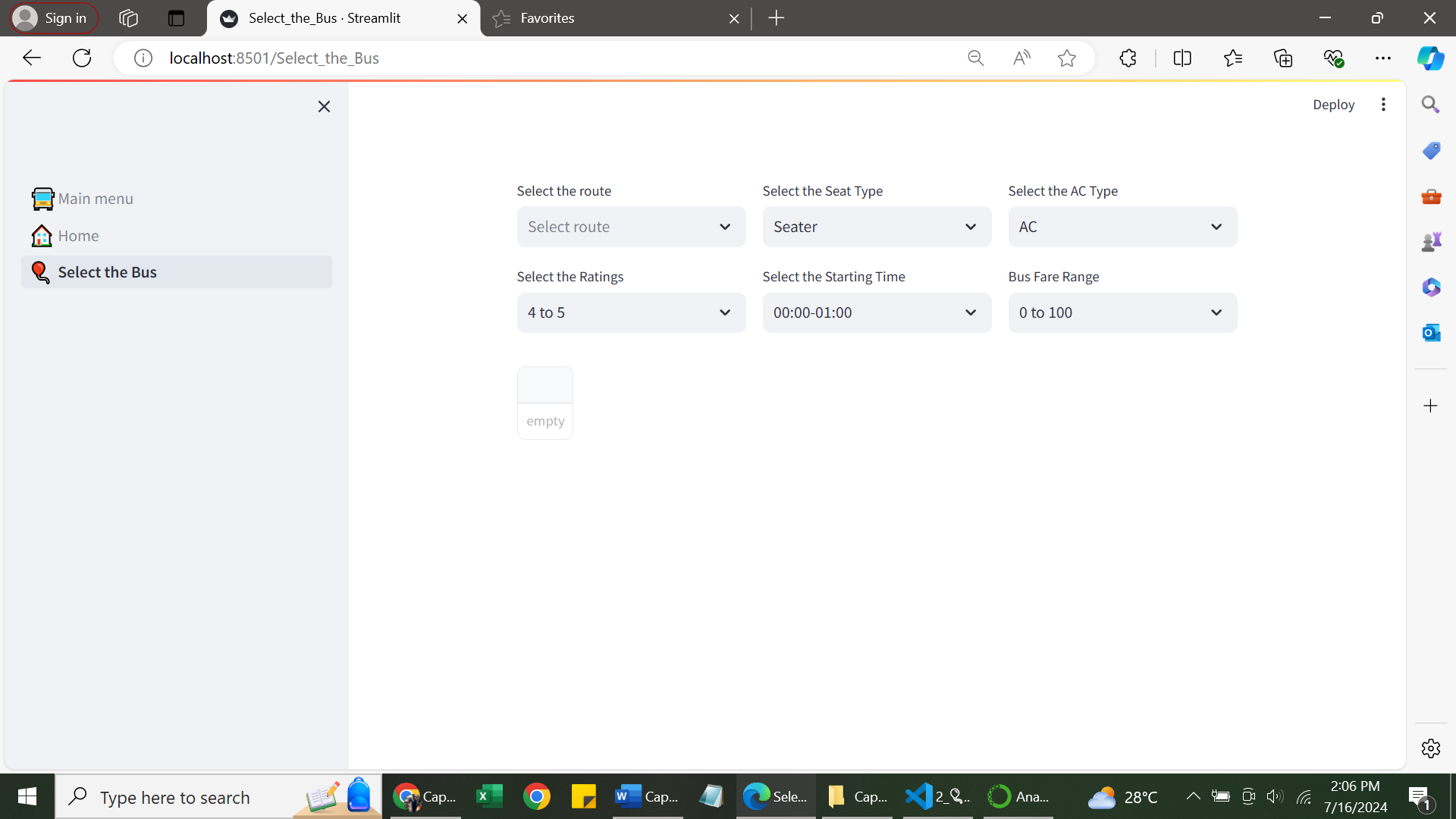
Container 2 with 3 options for where clause

* Option 4: Pass the options for Ratings to the Ratings select box
* Option 5: Pass the options for starting time to the Starting time select box
* Option 6: Pass the options for Fare range to the Fare range select box
* Create the various where clause sentences using the selected options
* Use and\_ and or\_ for this purpose to check like conditions for Seat Type and AC Type
* Use indexing to pick out the where clauses for Ratings and Fare range
* Query the database to extract the data based on the selected conditions.
  + Build the query using the various where clause conditions
  + Select the columns you want to display in the select clause with the alias name as needed

Eg:func.substr(table.c.route\_name, 1, func.instr(table.c.route\_name, 'to' ) - 1).label('Starting\_names')

* + - func.instr returns the position of first occurrence of ‘to’
    - func.substr returns the substring for the specified positions
    - .label is used to display the alias name
* Execute the query and fetch all distinct values using the *connection.execute(query).fetchall()* command
* Display the fetched resulted in the dataframe using st.dataframe()

Page2 initially



Page 2 after making selections for the dropdown boxes

