### **FindBus Web Scraper and Streamlit App**

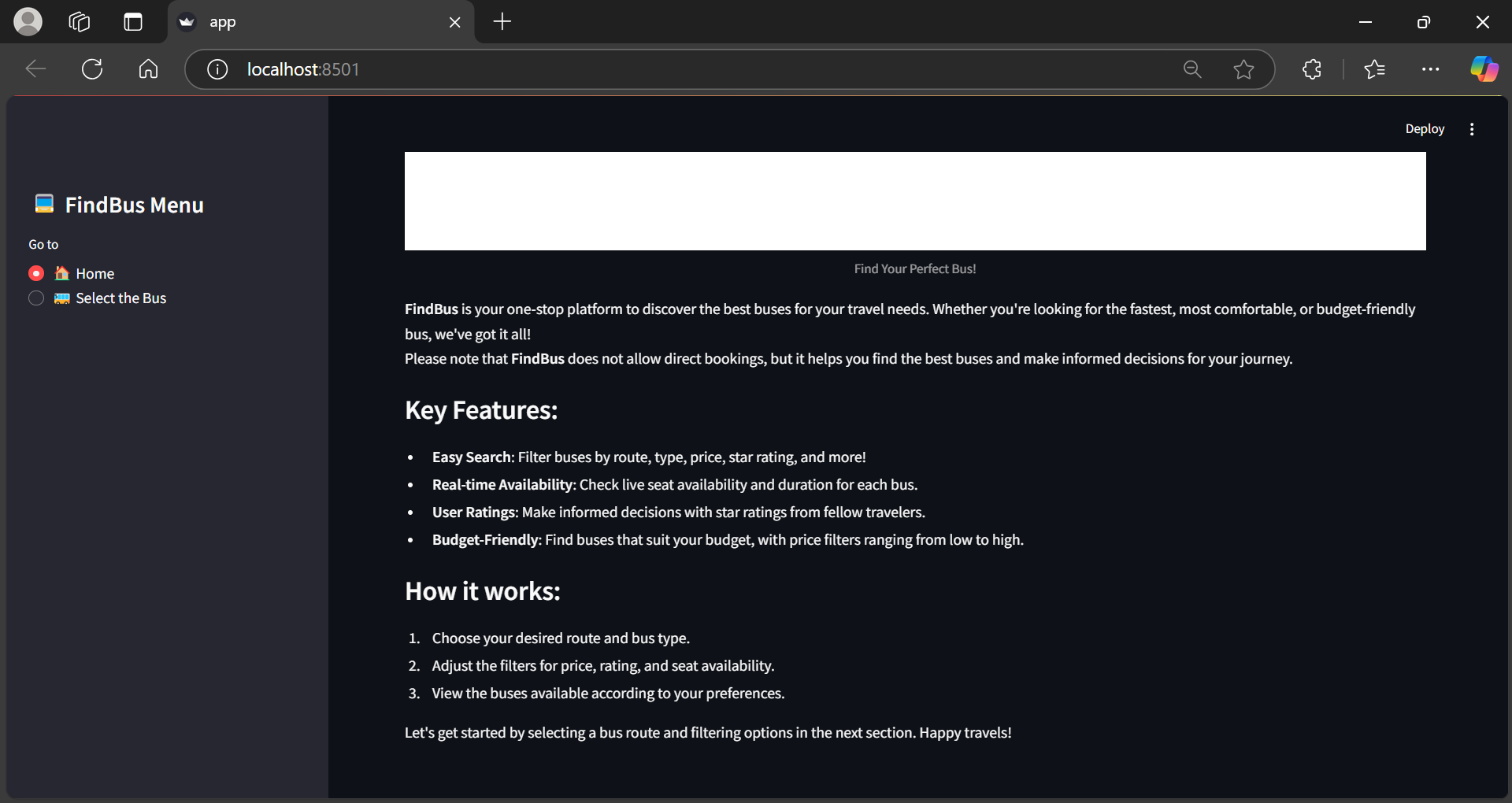
#### **Project Overview**

The **FindBus** project is a web scraping tool and Streamlit app that enables users to search for buses by route, type, price, star rating, seat availability, and more. It scrapes bus information from various state-run bus operators (e.g., Himachal Pradesh, Rajasthan, Punjab) from the RedBus platform and presents the data through a user-friendly interface.

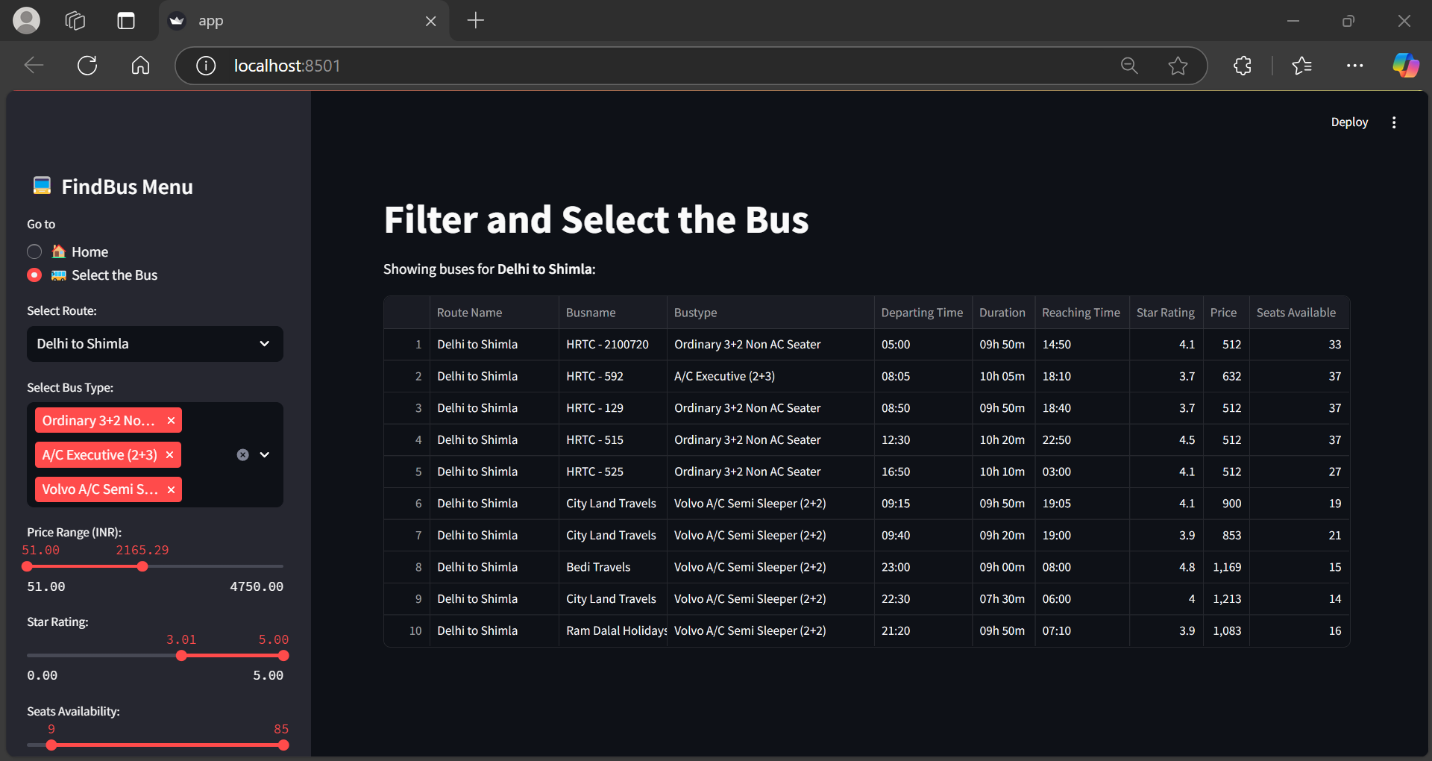
**App Screens**:

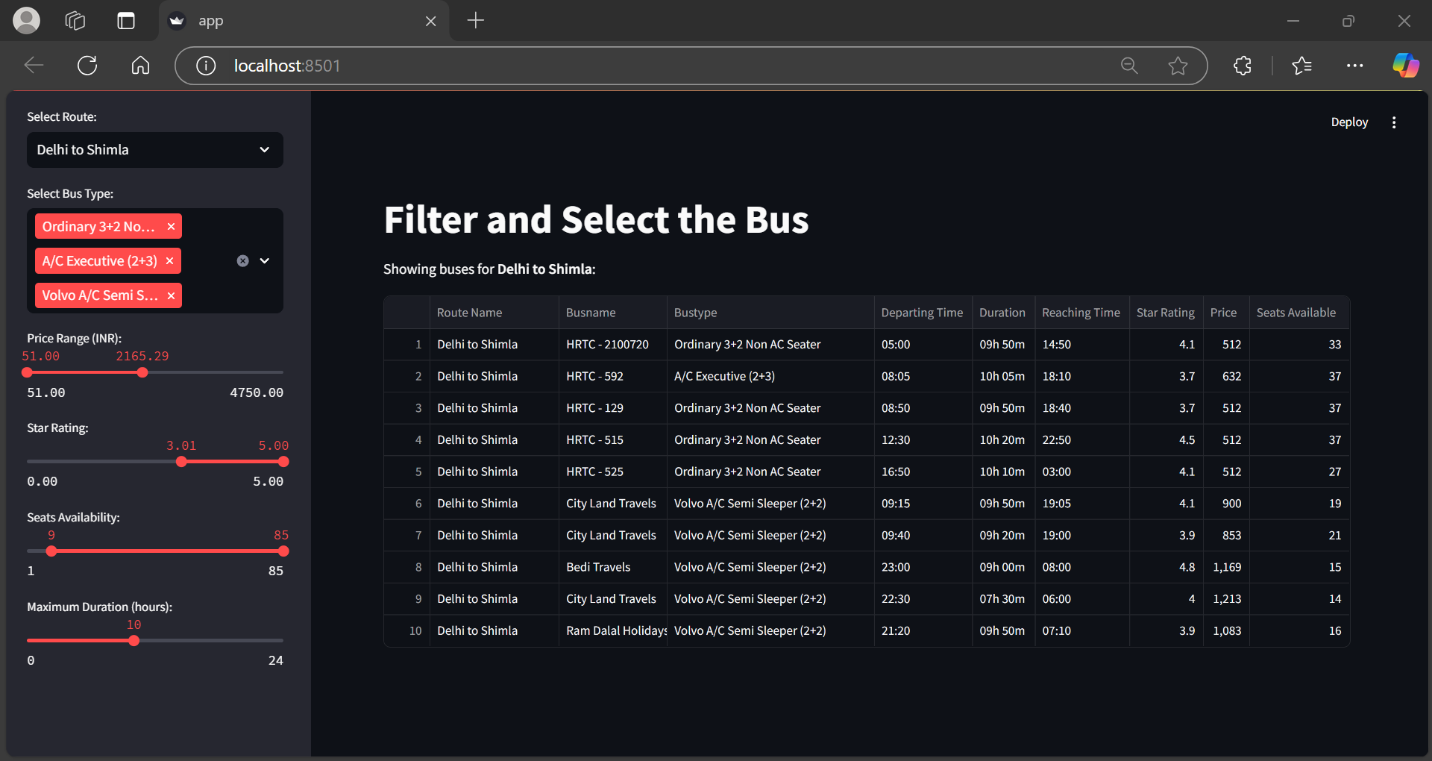
1. Home page:





1. Select the Bus page:





### **Key Components**

1. **Web Scraper (scraper.ipynb)**:  
   * Scrapes bus data from multiple state-run transport websites using Selenium.
   * Extracts details like route name, bus name, bus type, price, seat availability, star ratings, and more.
   * Data is stored and sanitized before being inserted into a MySQL database for further analysis.
2. **Backend Database**:  
   * A MySQL database stores the bus data scraped by the scraper.
   * The data is inserted into a table (busdetails) after sanitizing it, ensuring accurate types and formats.
3. **Streamlit App (app.py)**:  
   * Provides an interactive dashboard where users can filter and search bus details.
   * Users can filter buses by various attributes, including route, bus type, price, star rating, and seat availability.
   * Displays real-time data fetched from the MySQL database based on user-selected filters.

### **Step-by-Step Breakdown**

#### **1. Web Scraper - scraper.ipynb**

##### **Dependencies:**

* selenium: Used for automating browser interaction and scraping dynamic web pages.
* pandas: For handling and storing the data in a structured format (DataFrame).
* mysql.connector: Used to insert the scraped data into a MySQL database.
* concurrent.futures: To parallelize the scraping process for multiple states.

##### **Functions:**

1. **create\_driver()**:  
   * Initializes a Selenium WebDriver (Chrome) for web scraping.
   * Maximizes the browser window for better visibility.
2. **set\_zoom\_level()**:  
   * Adjusts the browser zoom level to ensure the entire content is visible without excessive scrolling.
3. **fetch\_routes()**:  
   * Visits a specific URL and extracts bus route names and links.
   * Handles pagination to scrape data across multiple pages.
4. **scroll\_to\_load\_all\_buses()**:  
   * Scrolls through the webpage to load all buses dynamically, ensuring all data is captured.
5. **fetch\_bus\_details()**:  
   * Scrapes detailed bus information (e.g., bus name, type, price, ratings, etc.) for each route.
   * Clicks on the "View Buses" button if available, to load all buses for a route.
   * Gathers data on each bus and stores it in a list.
6. **process\_state()**:  
   * Scrapes data for each state (calls fetch\_routes() and fetch\_bus\_details()).
   * Returns the collected bus data as a list.
7. **scrape\_all\_states()**:  
   * Iterates over a list of states and scrapes bus data for each one in parallel using a thread pool.
   * Consolidates the data into a Pandas DataFrame for further processing.
8. **Data Sanitization**:  
   * Cleans the data by converting columns like price, seats\_available, and star\_rating into numeric types.
   * Strips any extra spaces and handles missing values for all relevant columns.
9. **MySQL Database Insertion**:  
   * After sanitizing the data, it is inserted into a MySQL database (redbus).
   * Each record is inserted into the busdetails table, containing columns such as route\_name, busname, price, star\_rating, etc.

#### **2. Database - MySQL**

##### **Schema:**

* **Table Name**: busdetails
  + route\_name: Name of the bus route.
  + route\_link: Link to the route on RedBus.
  + busname: Name of the bus.
  + bustype: Type of the bus (e.g., sleeper, AC, etc.).
  + departing\_time: Time the bus departs.
  + duration: Duration of the bus journey.
  + reaching\_time: Time the bus reaches the destination.
  + star\_rating: Star rating of the bus.
  + price: Price of the bus ticket.
  + seats\_available: Available seats on the bus.

##### **Connection:**

* The scraper connects to MySQL using the mysql.connector library.
* The data is inserted using an INSERT INTO SQL statement.

#### **3. Streamlit App - app.py**

##### **Dependencies:**

* streamlit: For building the web application.
* pandas: For querying and displaying data.
* sqlalchemy: To establish the connection to the MySQL database.
* PIL: For handling images (e.g., bus image).

##### **Functions:**

1. **connect\_to\_database()**:  
   * Establishes a connection to the MySQL database using sqlalchemy.
   * Cached with @st.cache\_resource for better performance.
2. **get\_unique\_values()**:  
   * Fetches distinct values from a specified column (e.g., route\_name, bustype) for populating dropdown menus in the app.
3. **get\_min\_max\_values()**:  
   * Fetches the minimum and maximum values for a specified column (e.g., price, star rating) to use in filters.
4. **Main Application (main())**:  
   * Displays a sidebar menu for navigation.
   * On the home page, displays a brief description of the app and an image of a bus.
   * On the "Select the Bus" page:
     + Fetches and displays filtering options like routes, bus types, price, star rating, etc.
     + Builds a dynamic SQL query based on the selected filters.
     + Displays the filtered bus data in a table.

##### **Filters:**

* **Route**: Dropdown for selecting a route.
* **Bus Type**: Multi-select for selecting one or more bus types.
* **Price Range**: Slider for selecting the price range.
* **Star Rating**: Slider for selecting the star rating range.
* **Seats Available**: Slider for selecting the seat availability range.
* **Duration**: Slider for selecting the maximum bus duration.

##### **Display:**

* The filtered bus data is displayed as a table, where each bus’s details (route, bus name, type, etc.) are shown.
* If no buses match the selected filters, a message is displayed: "No buses found for the selected filters."

### **How to Use**

1. **Scraping Data**:  
   * Run the scraper.ipynb file to scrape the bus data for all the defined states.
   * The data is stored in the MySQL database after being sanitized.
2. **Streamlit App**:  
   * Run app.py to launch the Streamlit app.
   * Use the sidebar to filter buses by route, bus type, price, rating, seat availability, and duration.
   * View the filtered results in the table, with details about each bus.

### **Improvements & Future Enhancements**

1. **Pagination in Streamlit**: Implement pagination to manage large datasets in the Streamlit table.
2. **Real-time Data**: Introduce periodic scraping to keep the bus data up-to-date in the database.
3. **Enhanced UI**: Add search suggestions, tooltips, and reset buttons to enhance the user experience.
4. **Advanced Filters**: Add additional filters like bus amenities or operator names to narrow down results.
5. **Deployment**: Deploy the Streamlit app on platforms like Heroku or Streamlit Cloud for public access.

### **Conclusion**

The **FindBus** project efficiently integrates web scraping with a database backend and a user-friendly front-end interface. It allows users to easily search and filter buses, making it an excellent tool for travelers looking to find the best buses for their journey.