**Red bus Data Scraping and Filtering Project Documentation**

**Table of Contents**

1. Introduction
2. Project Structure
3. Data Collection
4. Database Setup
5. Stream lit Application
6. Usage Guide
7. Troubleshooting
8. Future Enhancements

**1. Introduction**

This project aims to scrape bus travel data from Red bus using Selenium, store it in a SQL database, and provide a user-friendly interface for data filtering and analysis using Stream lit. The project covers web scraping, data storage, and interactive data visualization.

**2. Project Structure**

**Red bus\_ project/**

**│**

**├── scraper/**

**│ ├── \_\_init\_\_.py**

**│ ├── redbus\_scraper.py**

**│ └── utils.py**

**│**

**├── database/**

**│ ├── \_\_init\_\_.py**

**│ ├── db\_config.py**

**│ └── db\_operations.py**

**├── stream lit\_ app/**

**│ ├── \_\_init\_\_.py**

**│ ├── app.py**

**│ └── components/**

**│ ├── \_\_init\_\_.py**

**│ ├── filters.py**

**│ └── visualizations.py**

**│**

**├── requirements.txt**

**├── config.py**

**└── main.py**

**3. Data Collection**

**Web Scraping Process**

The data is scraped from Red bus using Selenium. The redbus\_scraper.py script handles the scraping process:

1. Initialize the Selenium WebDriver
2. Navigate to the Red bus website
3. Search for bus routes (minimum 10 Government State Bus Transport routes)
4. Extract required information for each route
5. Store the data in a structured format

**Scraped Data Fields**

* Bus Routes Name
* Bus Routes Link
* Bus Name
* Bus Type (Sleeper/Seater/AC/Non-AC)
* Departing Time
* Duration
* Reaching Time
* Star Rating
* Price
* Seat Availability

**4. Database Setup**

**Database Schema**

The scraped data is stored in a SQL database with the following schema

CREATE TABLE bus routes (

id INT PRIMARY KEY AUTO\_INCREMENT,

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,

seats\_ available INT

);

**Database Operations**

The db\_operations.py script handles all database interactions, including:

* Connecting to the database
* Creating tables
* Inserting scraped data
* Retrieving data for the Stream lit application

**5. Stream lit Application**

The Stream lit application (app.py) provides an interactive interface for data filtering and visualization.

**Features**

* Filter data by bus type, route, price range, star rating, and availability
* Display filtered results in a tabular format
* Visualize data using charts and graphs

**Components**

* filters.py: Contains functions for creating and applying data filters
* visualizations.py: Handles data visualization components

**6. Usage Guide**

1. Install required dependencies:

pip install -r requirements.txt

1. Configure the database connection in config.py
2. Run the scraper to collect data:

python main.py --scrape

1. Launch the Stream lit application:

Stream it run strit\_app/app.py

1. Use the web interface to interact with the data:
   * Apply filters using the sidebar
   * View filtered results in the main area
   * Explore visualizations and analytics

**7. Troubleshooting**

* Ensure all dependencies are installed correctly
* Check the database connection settings in config.py
* Verify that the Red bus website structure hasn't changed, which could affect scraping
* For Selenium issues, ensure the correct WebDriver is installed and configured

**8. Future Enhancements**

* Implement automated daily scraping
* Add more advanced analytics features
* Integrate with other travel websites for comprehensive data
* Implement user authentication for personalized experiences

This documentation provides a comprehensive overview of the Redbus Data Scraping and Filtering project. It covers the project structure, data collection process, database setup, Stream lit application details, and usage instructions. This should serve as a guide for understanding, using, and potentially extending the project.