Project Report: GUI Car Rental System

PHASE ONE

1. Introduction

Project Description: This project is a "Car Rental Management System" built using Python and the "tkinter" library for the Graphical User Interface (GUI). The program allows users to browse a list of available cars, filter them by color, and calculate the total rental price based on the number of days, with automatic discount application. Why this topic: We chose this topic to apply the concepts of modular programming and GUI development in a real-world scenario. Goal: The goal is to replace manual calculation methods with an automated, user-friendly interface that ensures accuracy in pricing and provides a better user experience.

2. Problem Statement

The Problem: Car rental staff and customers often struggle with manual price calculations, especially when applying complex rules like weekly discounts. Manual searching for available cars by specific attributes (like color) in a paper list is time-consuming and prone to errors. Importance: Therefore, a computerized system with a graphical interface is needed to instantly filter options and calculate costs accurately, saving time for both the business and the customer.

3. System Functionality

The program provides the following functionalities, applying various Python topics:

1. Load Car Data: Reads car details (Name, Price, Color) from an external text file (cars.txt) using file handling .
2. Display Cars: Shows all available cars in a scrollable list (Listbox) using loops .
3. Filter by Color: Allows the user to search for cars by specific color using string manipulation and conditional statements .
4. Calculate Price: Uses a function to calculate the total cost, applying a 20% discount if the rental period is 7 days or more .
5. Input Validation: Ensures the user enters valid numbers for days using try/except blocks to prevent crashes .
6. Graphical User Interface: Provides an interactive window with buttons, labels, and input fields using tkinter.

4. Design (Data Flow Diagrams)

Data Flow Description: [User] → (Select Car & Enter Days) → [System] → (Process Price & Apply Discount) → [Output]

Step-by-Step Process:

1. The system loads car data from the file into a list of dictionaries.
2. The user inputs a color preference; the system filters the list and updates the display.
3. The user selects a car and enters the number of days.
4. The system validates the input (checks if it's a number).
5. The system calculates the total price (applying discount logic) and displays the final amount on the screen.

PHASE TWO

A. How to Run the Program

To execute the "Car Rental System", follow these steps:

1. Ensure **Python 3.x** is installed on your computer.
2. Make sure all project files (main.py, cars.txt, etc.) are in the **same folder**.
3. Open the terminal (or Command Prompt) and navigate to the project folder.
4. Run the following command:

Bash

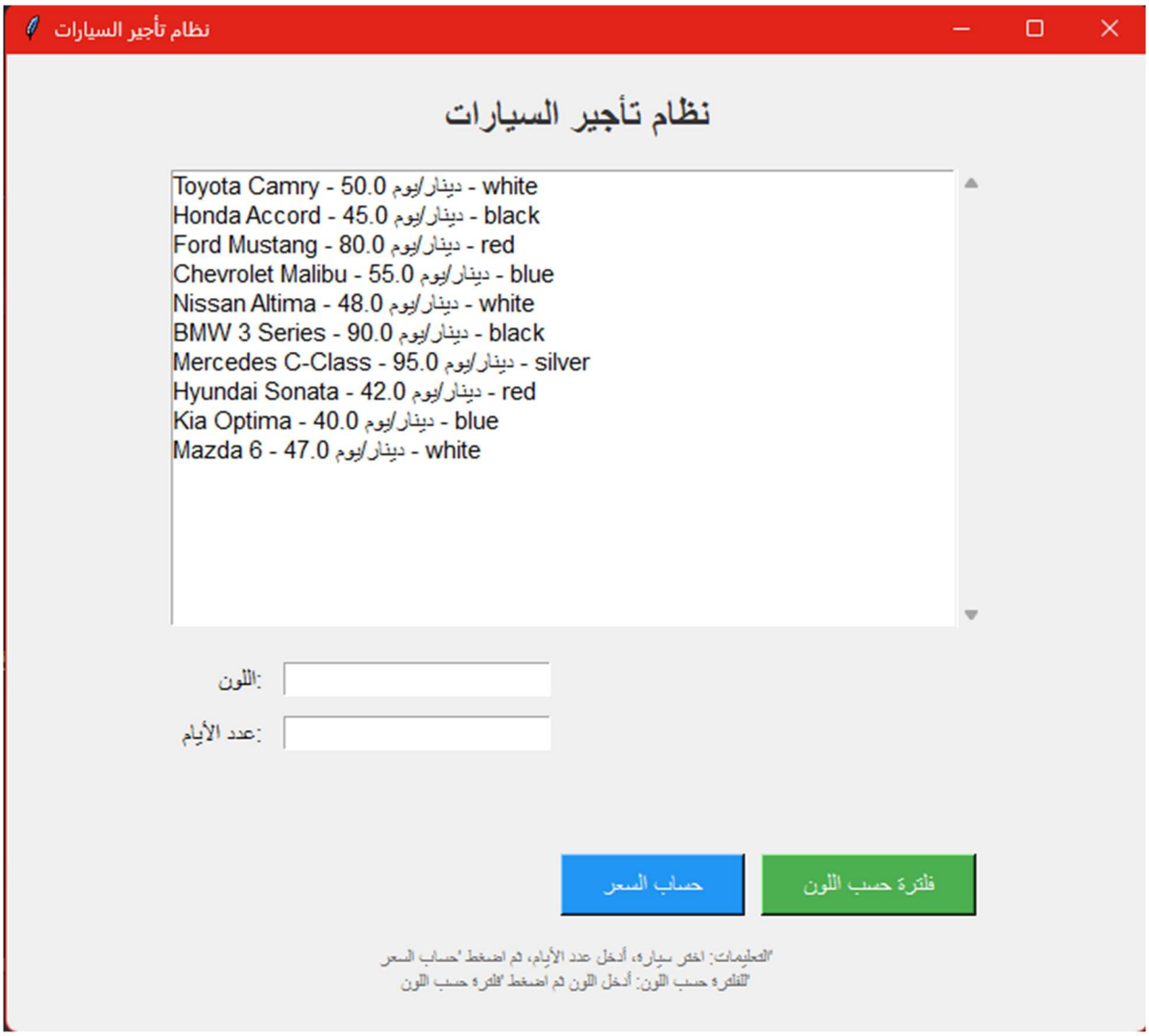
```
python main.py
```

5. The graphical window will appear. You can select a car, enter the number of days, and click "Calculate Price".

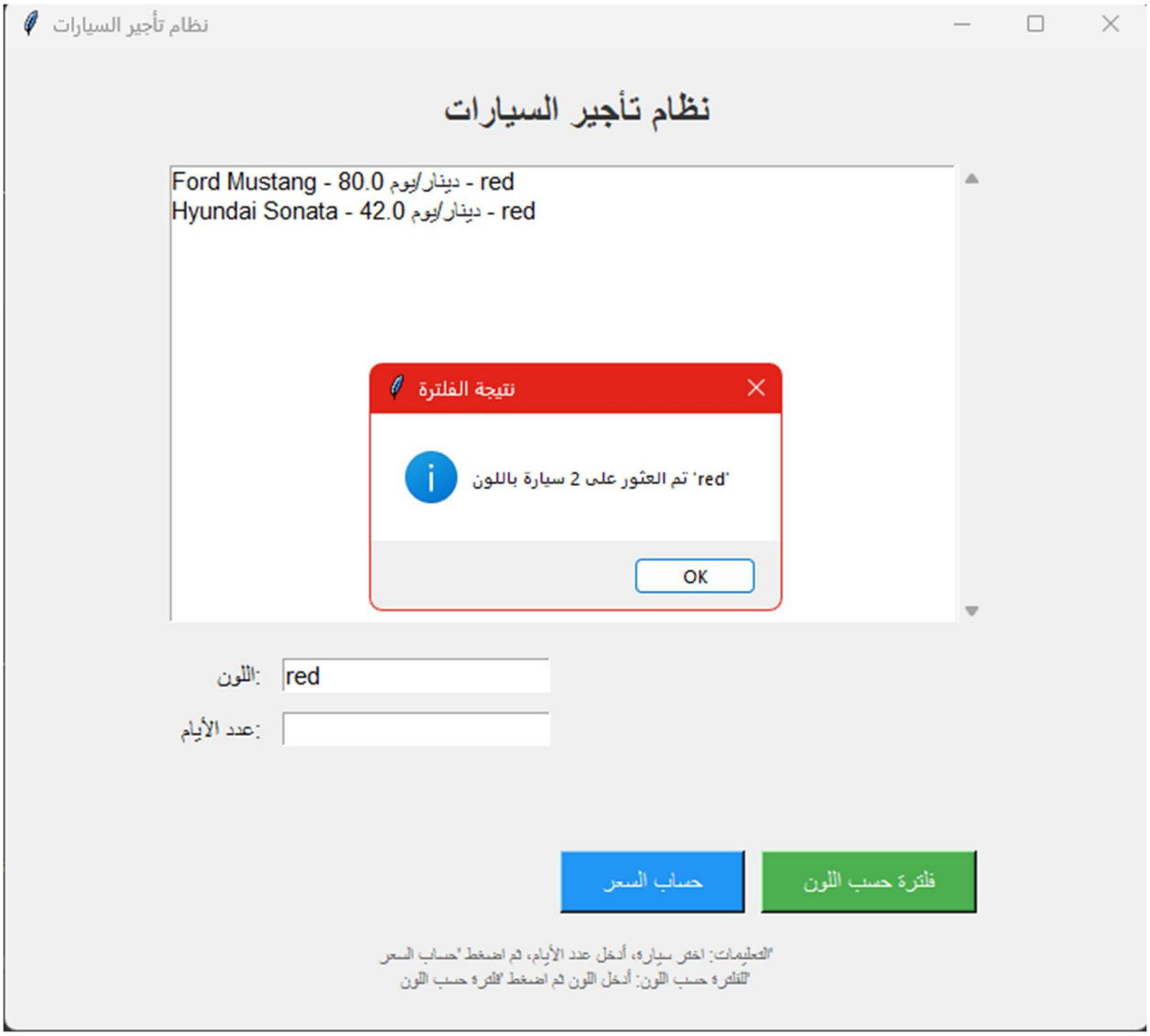
B. Program Output (Screenshots)

Below are screenshots demonstrating the system functionality:

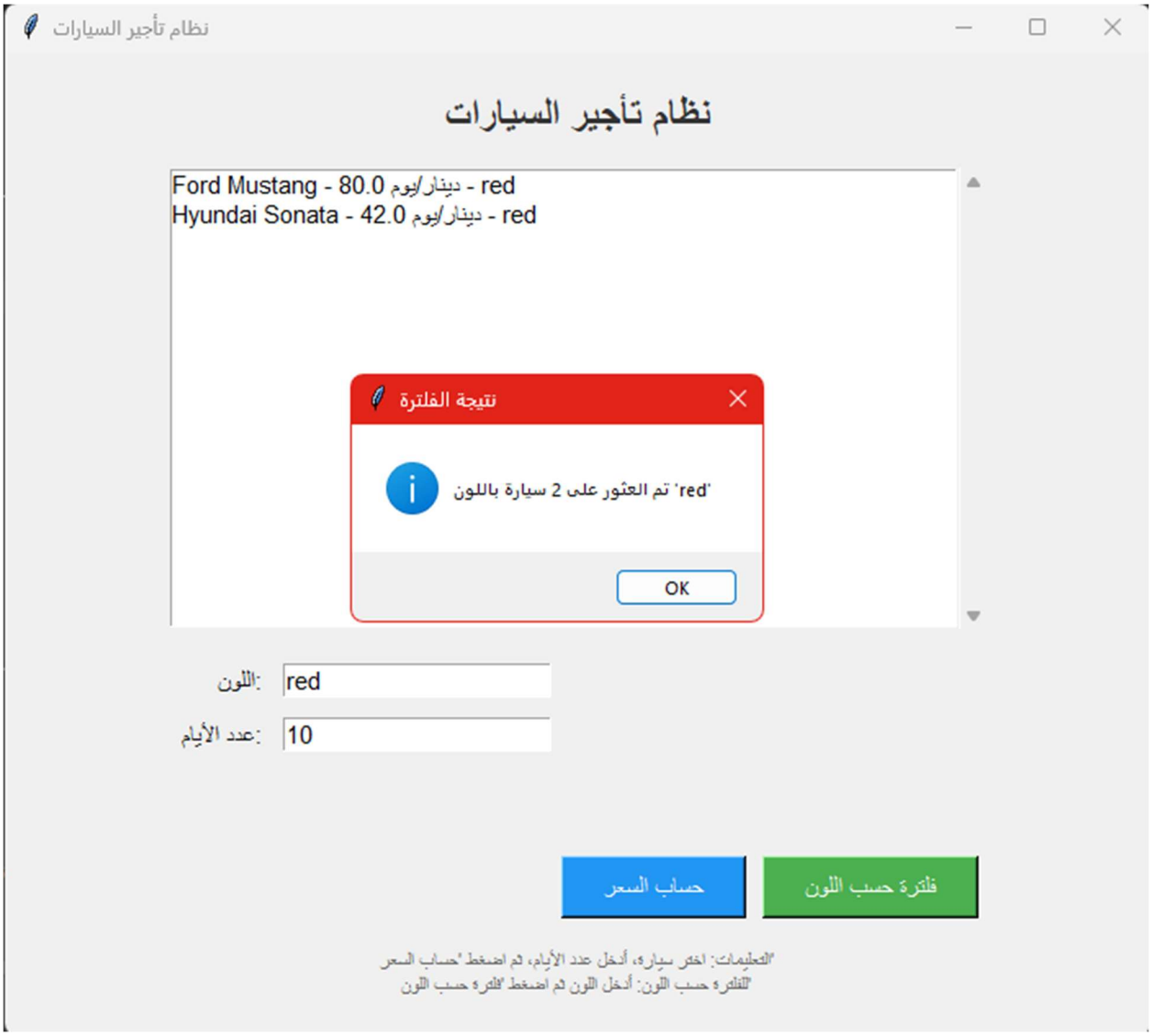
1. **Main Interface:** Shows the list of all cars loaded from the file.



2. **Filtering Feature:** Shows the result when searching for "Red".



3. **Calculation Feature:** Shows the pop-up message with the total price and discount.



C. Summary of Learning

(ملخص ما تم تعلمه)

This project helped us apply various fundamental and advanced Python concepts:

- **Modular Programming:** We learned how to split our code into multiple files (calculator, filtering, database) to make it organized and reusable.
- **File Handling:** We applied Chapter 7 concepts to read real data from cars.txt instead of hardcoding it.

- **GUI Development:** We moved beyond simple command-line programs and learned how to build interactive interfaces using the tkinter library.
- **Logic & Algorithms:** We implemented logic for filtering lists and calculating prices with conditions (if/else).
- **Error Handling:** We used try/except blocks to ensure the program doesn't crash if the file is missing or user input is invalid.