Project Report:
Parking Management Utility

Table of Contents

7. Conclusion

1. Introduction

Purpose of the Utility:

The Parking Management Utility is a command-line-based application developed in C to facilitate efficient management of a parking lot. The utility provides functionalities for vehicle entry, exit, parking availability information, and fee calculation. It aims to streamline the parking management process and ensure a seamless experience for parking lot staff and customers.

Scope and Objectives:

The utility's main objective is to automate the parking management tasks and reduce manual efforts in tracking vehicle entries and exits. The scope includes handling multiple vehicle types, real-time availability status, and generating reports. The utility is designed for a single parking lot and has predefined capacities for each vehicle type.

Project Overview:

The project involves designing, implementing, and testing the Parking Management Utility. The utility is developed using the C programming language, adhering to best practices in software development. The code is modularized, error handling is implemented gracefully, and the utility is documented effectively.

2. Utility Design and Implementation

Functionality and Specifications:

The utility offers the following main functionalities:

- Vehicle Entry: Allows vehicles to enter the parking lot, categorizes them as motorcycle, light vehicle, or heavy vehicle, and records the entry time.

- Vehicle Exit: Allows vehicles to exit the parking lot, calculates the parking fee based on the vehicle type and duration of parking, and records the exit time and fee information.

- Parking Availability: Provides real-time information about available parking spaces for each vehicle type.

- Parking Fee Change: Enables the administrator to change parking fees for different vehicle types.

- Saving Exiting Information: The utility saves information about exited vehicles to a file named `record.txt`.

Input/Output Requirements:

The utility interacts with the user through a command-line interface. It prompts the user for input when entering or exiting a vehicle and displays relevant information, such as vehicle details and parking fees. The utility saves exiting vehicle information to a file named `record.txt`.

Command-Line Options:

The utility does not have any command-line options; it is an interactive application with a menu-based interface.

Expected Behavior:

The utility is expected to handle user inputs gracefully and display appropriate messages for successful actions or error scenarios. It should accurately calculate parking fees and save vehicle information to the file. The application should not crash or exhibit unexpected behavior during normal usage.

Modularization and Code Organization:

The code is structured into functions for each functionality, promoting modularity and maintainability. Common operations, such as converting vehicle numbers to uppercase and calculating parking fees, are placed in separate functions for reuse.

Error Handling and Documentation:

The utility includes error handling mechanisms to handle invalid inputs and display user-friendly error messages. The code is documented effectively, providing clear explanations of each function, its parameters, and return values. Additionally, the README file provides usage instructions and information about the utility's purpose and dependencies.

3. Version Control and Repository Setup

Version Control System Selection:

The version control system chosen for this project is Git, a widely used and robust distributed version control system.

Repository Initialization:

The project repository is initialized on a code hosting platform, such as GitHub, where the utility's source code and documentation are stored.

Code Management and Collaboration:

The repository enables collaboration among team members. Issues and discussion boards can be used to manage tasks and track progress. Reviewing and merging code changes follow standard Git workflows.

4. Dependencies and System Requirements

Dependencies:

The Parking Management Utility relies on standard C libraries for input/output operations, time handling, and string manipulation. There are no external dependencies or additional libraries required.

System Requirements:

To compile and run the utility, a C compiler (e.g., GCC) must be installed on the system. The utility is designed to be platform-independent and should work on any system supporting a C compiler.

5. Usage Instructions

Compilation and Execution:

To use the utility, follow these steps:

1. Clone the source code repository from GitHub.
2. Navigate to the `parking management system` directory.
3. Compile the utility using a C compiler:

```
gcc -o parking_management_util parking_management.c
```

4. Run the utility using the command:

```
./parking_management_util
```

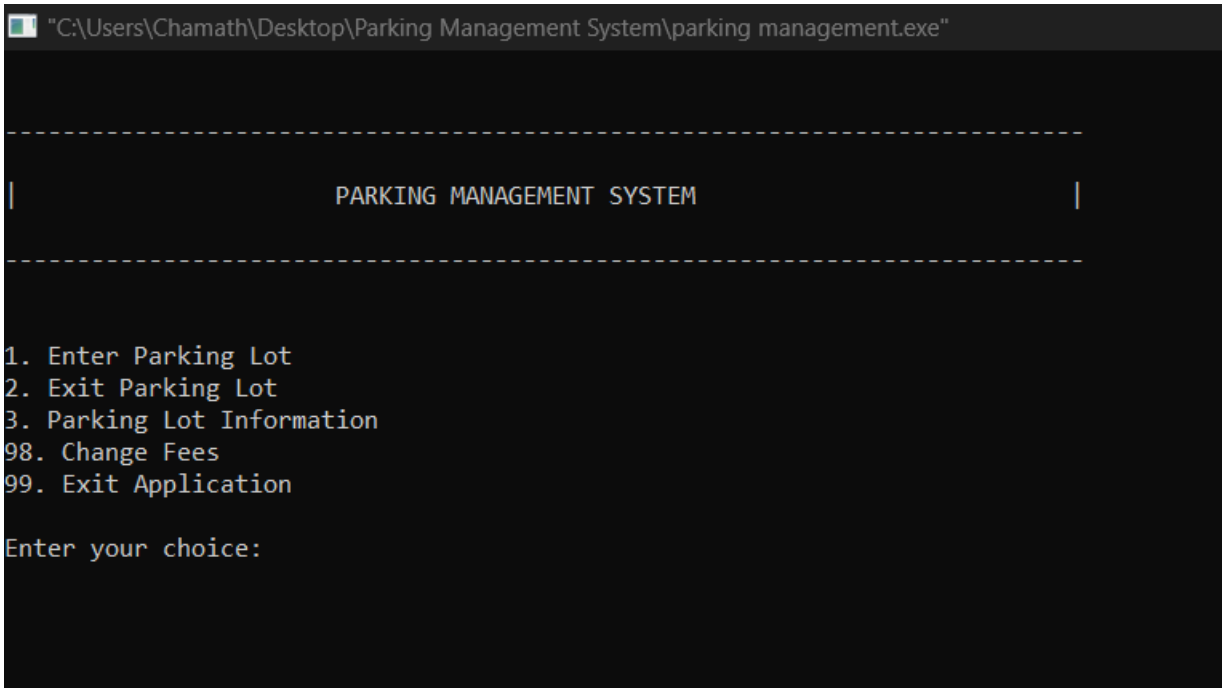
Utility Features and Menu Options:

Upon running the utility, a menu with options will be displayed. Users can perform the following actions:

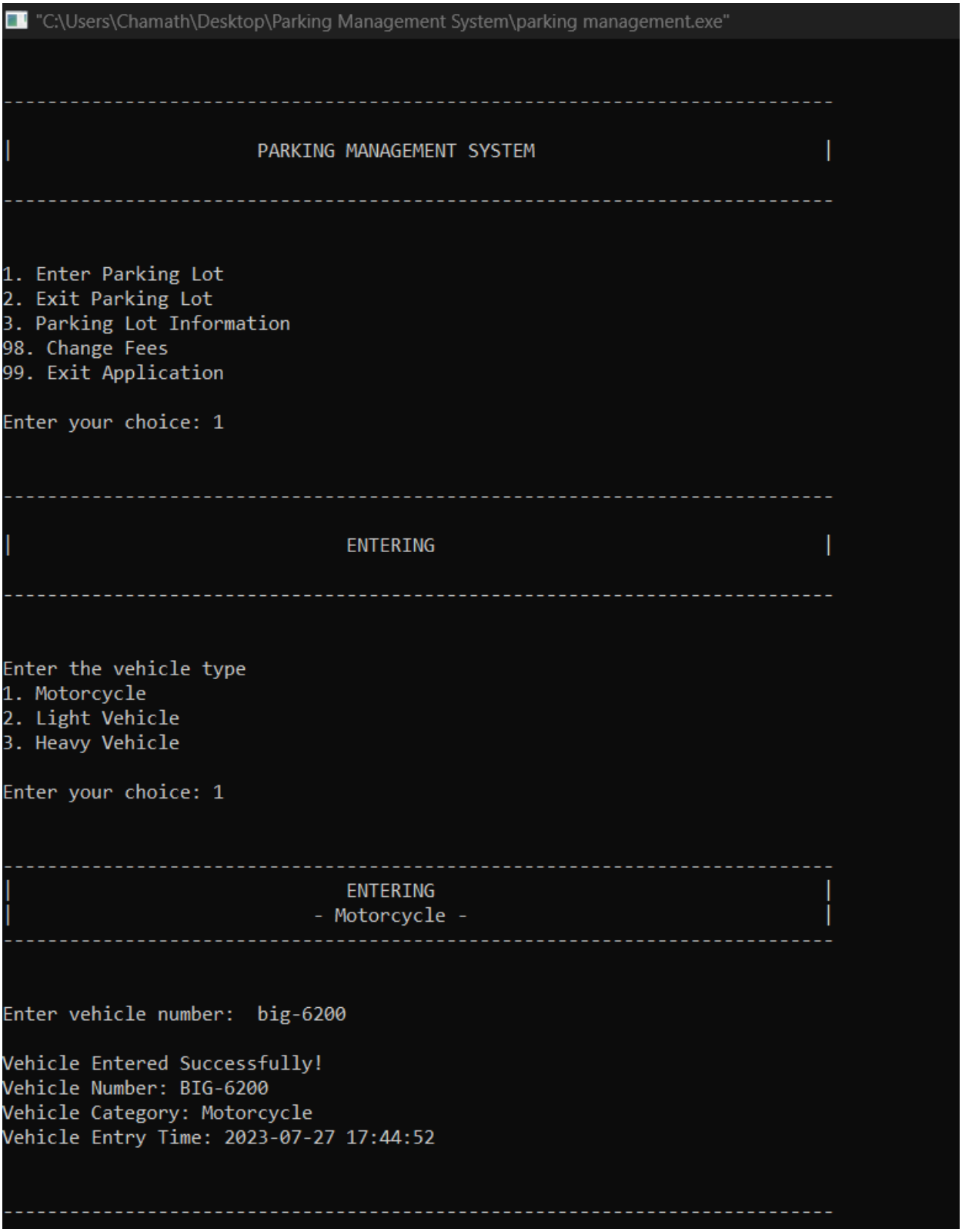
- Enter Parking Lot
- Exit Parking Lot
- Parking Lot Information
- Change Parking Fees
- Exit Application

Functionality:

1. Main Menu



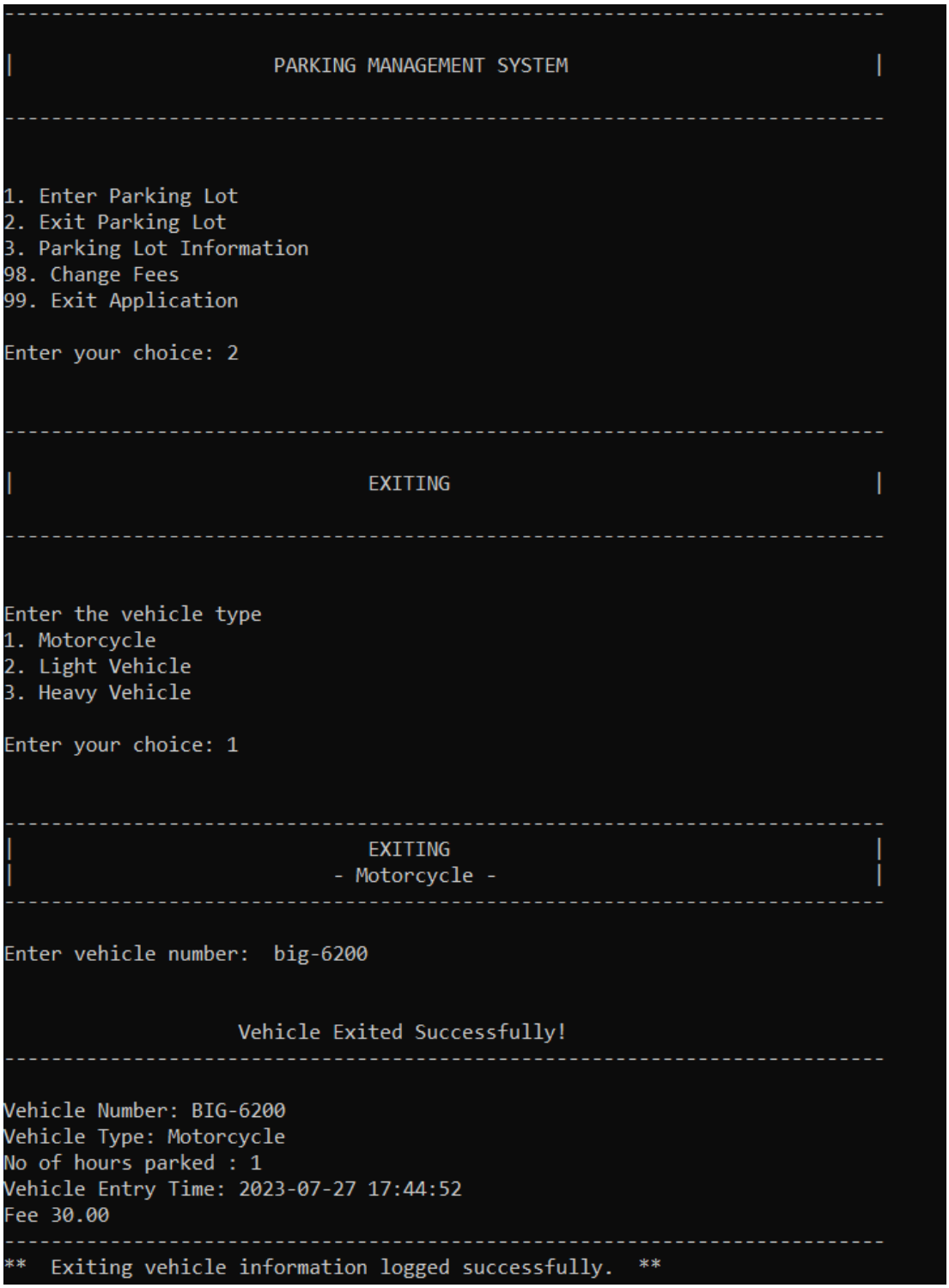
2. Enter Parking Lot:
   - The user selects the vehicle type and enters the vehicle number.
   - The utility records the entry time and displays vehicle details.


"C:\Users\Chamath\Desktop\Parking Management System\parking management.exe"
PARKING MANAGEMENT SYSTEM
1. Enter Parking Lot
2. Exit Parking Lot
3. Parking Lot Information
98. Change Fees
99. Exit Application
Enter your choice: 1
ENTERING
Enter the vehicle type
1. Motorcycle
2. Light Vehicle
3. Heavy Vehicle
Enter your choice: 1
ENTERING
- Motorcycle -
Enter vehicle number: big-6200
Vehicle Entered Successfully!
Vehicle Number: BIG-6200
Vehicle Category: Motorcycle
Vehicle Entry Time: 2023-07-27 17:44:52

3. Exit Parking Lot:

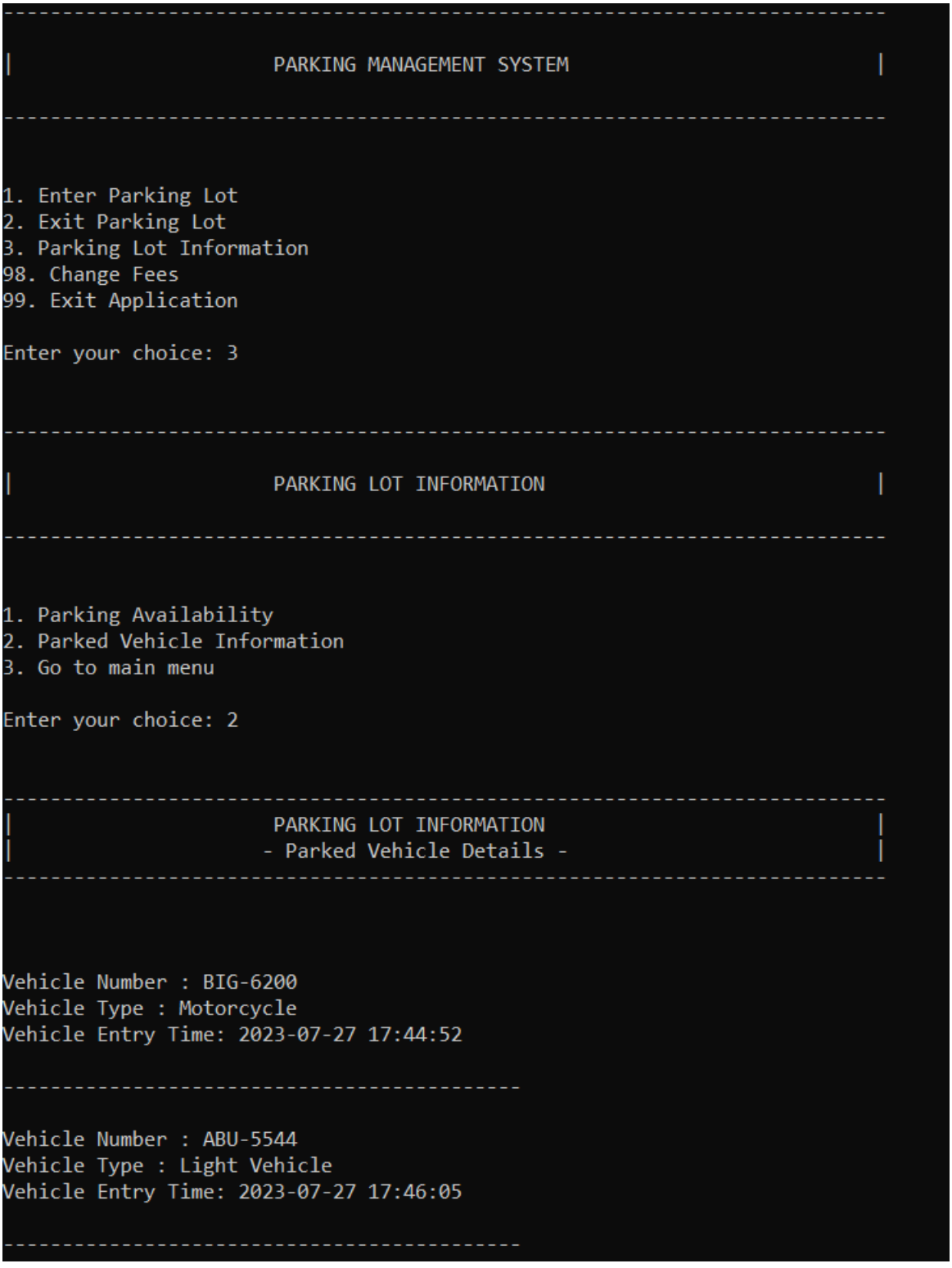
- The user enters the vehicle number for exit.

- The utility calculates the parking fee based on the duration of parking and the vehicle type.



4. Parking Lot Information:

- The user can view real-time parking availability for different vehicle types.
- The utility displays the list of vehicles parked along with their entry times.

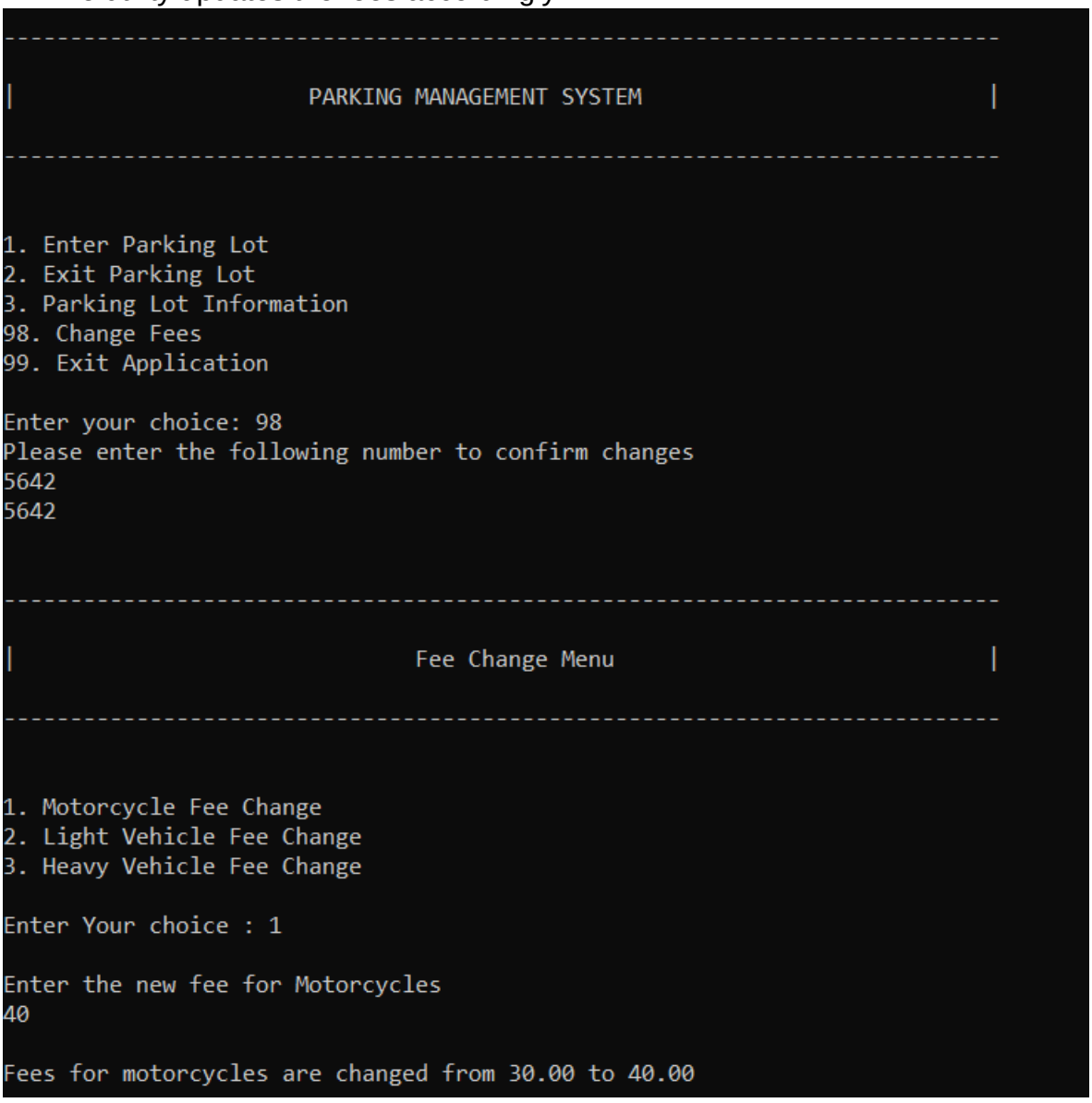


```
--------------------------------------------------------------------------

|                      PARKING MANAGEMENT SYSTEM                         |

--------------------------------------------------------------------------


1. Enter Parking Lot
2. Exit Parking Lot
3. Parking Lot Information
98. Change Fees
99. Exit Application

Enter your choice: 3


--------------------------------------------------------------------------

|                      PARKING LOT INFORMATION                           |

--------------------------------------------------------------------------


1. Parking Availability
2. Parked Vehicle Information
3. Go to main menu

Enter your choice: 1


--------------------------------------------------------------------------
|                      PARKING LOT INFORMATION                           |
|                   - Parking Space Availability -                       |
--------------------------------------------------------------------------


Number of Motorcycle spaces left: 49
Number of Light Vehicle spaces left: 24
Number of Heavy Vehicle spaces left: 9


_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
```

5. Change Parking Fees:

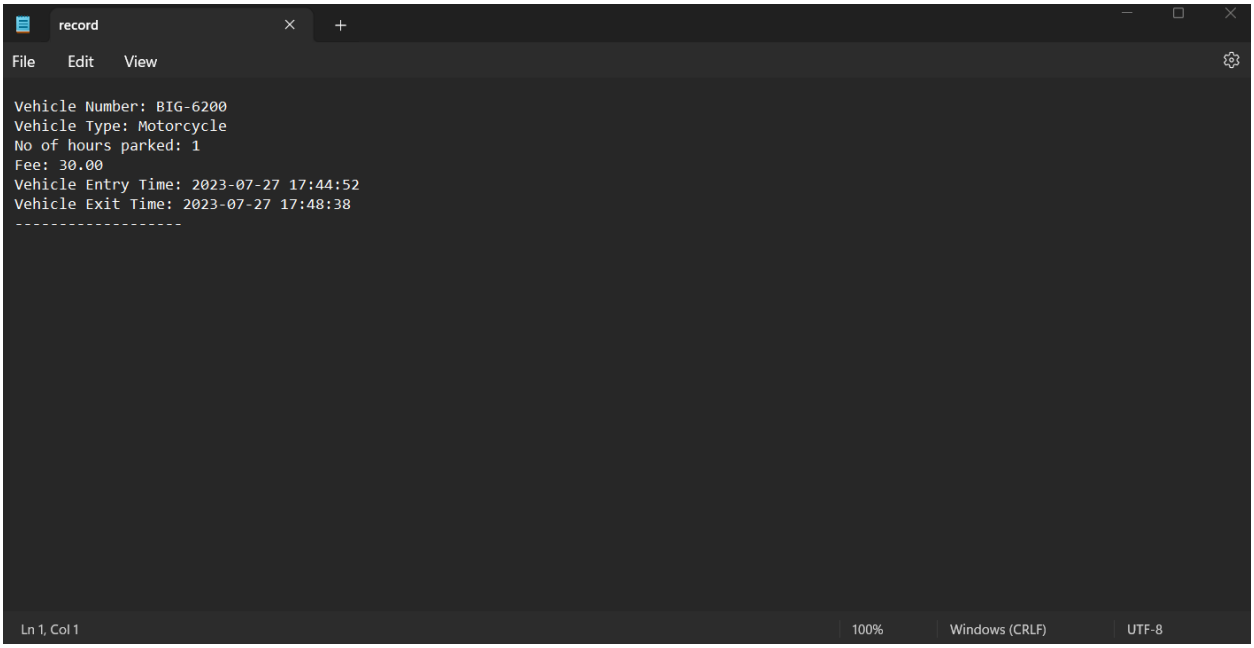
- The administrator can change parking fees for different vehicle types after verification.

- The utility updates the fees accordingly.



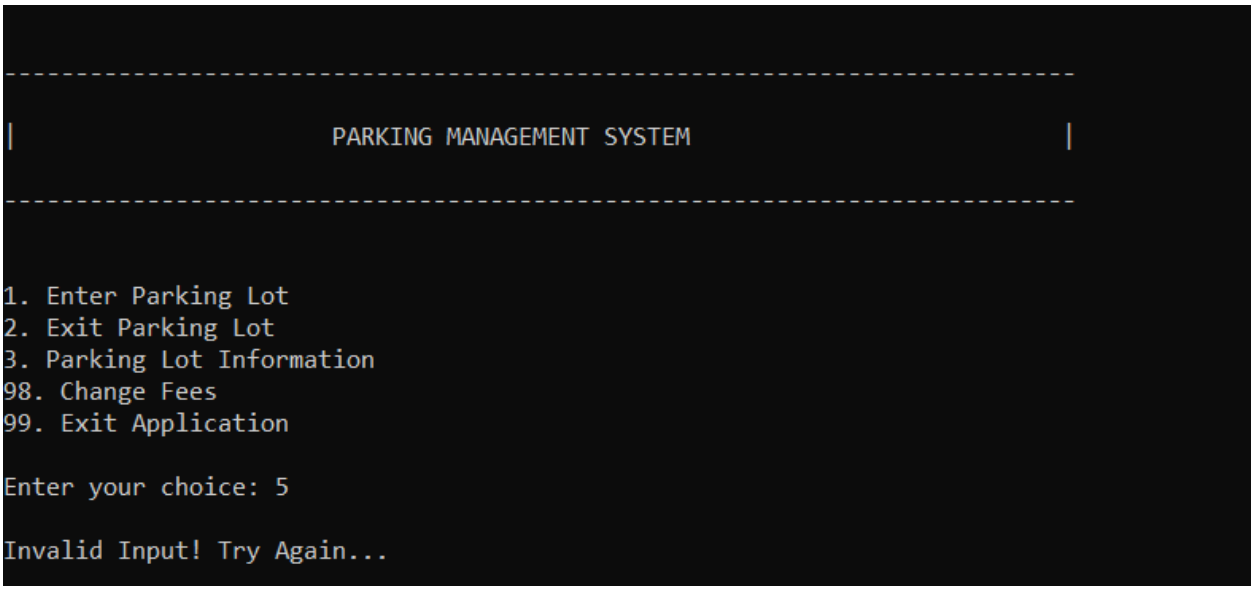
6. Log the parking ticket information in a text file:

- Application log all the ticket details which issued when a vehicle is exiting in a text file



7. Invalid input handling

- Application displays necessary error messages to user when user input a invalid input



6. License

The Parking Management Utility is distributed under the MIT License, allowing users to use, modify, and distribute the code freely.

7. Conclusion

The Parking Management Utility offers an efficient solution for managing a parking lot. Its interactive interface and straightforward functionalities make it user-friendly and effective. The utility's modular and well-documented code ensures maintainability and extensibility.

Contributors:

Chamath Kaushalya

CS/2020/012