CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

(UIC)

Project Report

ON

Cab Selection and Ride Management System

Program Name: BCA

Subject Name/Code: Object Oriented Programming(24CAH-201)

Submitted by:	**Submitted to:**
Name: Gurjot Singh	**Name:** Ms. Rashmi Saluja
UID: 24BCA10298	**Designation:** Assistant Professor
Section: 24BCA 2	
Group: B	

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project. I am deeply thankful to my instructor for providing guidance, support, and encouragement throughout the development of this work. Their valuable suggestions and insights helped me understand the concepts more clearly and implement them effectively.

I am also grateful to my institution for offering the necessary resources and a supportive learning environment that allowed me to explore and apply programming concepts practically. This project provided me with an opportunity to enhance my understanding of Object-Oriented Programming in C++ and its real-world applications.

Finally, I would like to thank my family and friends for their constant motivation and support during the course of this project. Their encouragement played an important role in helping me stay focused and determined.

This project has been a significant learning experience, and I am thankful to all who guided and supported me throughout the process.

Contents

Introduction

In today's world, technology plays a vital role in enhancing convenience and efficiency in daily activities. One such advancement is the development of ride-hailing and cab booking applications, which allow passengers to connect with drivers quickly and effortlessly. This project is a simple implementation of a cab-matching system using C++ programming, focusing on core Object-Oriented Programming concepts.

The system allows a passenger to enter their basic details and then displays a set of available drivers with different attributes such as distance, rating, and fare. Based on the user's selection of a matching strategy, the system identifies the most suitable driver. The selection may be made on the basis of the nearest driver, the highest-rated driver, or the lowest fare. Once a driver is selected, the ride details are stored in a file for record-keeping, which simulates a basic ride history database.

This project demonstrates the use of classes, data encapsulation, file handling, and decision-making structures in C++. It reflects how simple algorithms can be used to model real-world applications. The implementation emphasizes clarity, modularity, and user interaction, making it an effective example of applying theoretical programming knowledge to practical problem-solving.

Project Summary

This project focuses on the development of a simple and interactive cab matching system using the C++ programming language. With the increasing use of ride-hailing services in modern transportation, there is significant value in understanding how such systems work at a logical and functional level. This project models the basic mechanism of assigning drivers to passengers based on specific selection criteria. It demonstrates how Object-Oriented Programming (OOP) principles can be used to structure a real-world application in a systematic way.

The project consists of two main entities: the **Passenger** and the **Driver**. The passenger provides their personal details, while multiple drivers are generated by the system, each having attributes such as distance from the passenger, service rating, and fare. The user is given the choice to select the method of driver allocation. The system then compares the available drivers based on the user's selected criteria and assigns the most suitable driver. The details of the transaction are then stored in a text file, which represents a simple way of maintaining a ride history.

This implementation achieves clarity and simplicity while demonstrating programming concepts such as encapsulation, classes and objects, control structures, and file handling. Although the system is small in scale compared to commercial applications, it reflects the logic and decision-making strategies used in real-world ride-sharing platforms.

Objectives of the Project

The main objective of this project is to design a basic cab matching system that can assign a suitable driver to a passenger using predefined selection strategies. The additional objectives include:

1. To apply Object-Oriented Programming concepts to construct a modular program structure.
2. To demonstrate the use of user-defined classes with relevant data members and member functions.

3. To incorporate file handling for recording ride transactions in a persistent manner.
4. To simulate the selection logic used in modern transportation applications.
5. To create a user-friendly interface that interacts smoothly with the passenger.

These objectives ensure that the system is both practical and educational, helping the developer gain a deeper understanding of how real-world service-based applications are programmed.

Features of the System

The project contains several useful features:

1. **Passenger Information Input**
   The program accepts the passenger's name and location, ensuring personalized service.
2. **Driver Data Simulation**
   The system generates multiple drivers with randomized values for distance, rating, and fare. This models real-time variations in the availability of drivers.
3. **Multiple Driver Selection Strategies**
   The user can choose whether the driver should be selected based on the nearest distance, highest rating, or lowest fare, reflecting real-world user preferences.
4. **Clear Display of Driver Information**
   The program presents all driver details in a well-formatted and readable layout.
5. **Automatic Matching of Best Driver**
   The system compares all drivers to select the most suitable one according to the chosen criteria.

6. **Ride History Storage**
   The confirmed ride details are saved in a text file for future reference, demonstrating persistent data management.
7. **Option for Multiple Bookings**
   The user may choose to book multiple rides in the same session.

Applications of the System

Although this project is developed at a basic level, it represents the core functionality of real-world cab booking systems. It can be applied in:

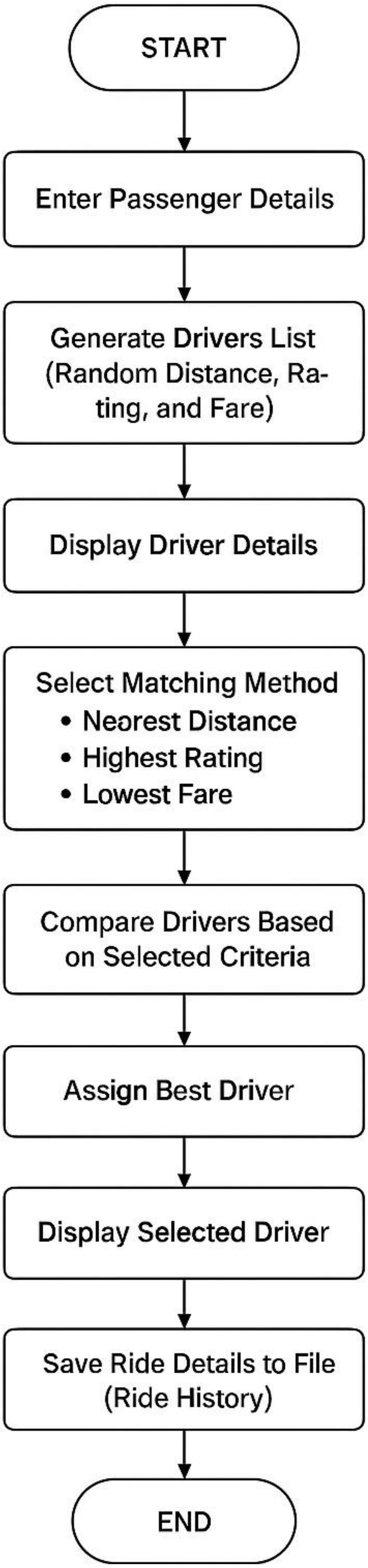
1. **Learning Environments**
   As an educational tool to help students understand OOP and decision-making logic.
2. **Simulation of Ride-Matching Techniques**
   It demonstrates how selection algorithms are used in ride-hailing platforms such as Ola, Uber, and Rapido.
3. **Concept Development**
   The project can serve as a foundation for developing more advanced transportation management systems.

Working of the System

The working of the system begins with collecting the required input from the passenger. Once the user enters their name and location, the system creates a list of five drivers. Each driver is assigned a unique name and random values for distance, rating, and fare.

The system then displays the list of drivers to the user. The passenger is prompted to select the matching strategy they prefer. Depending on the selected option, the system compares the corresponding values among the drivers to determine the most appropriate driver.

After identifying the best match, the system displays the selected driver’s details on the screen and simultaneously records the entire ride information into a text file. Finally, the user is asked whether they want to make another booking. If yes, the process repeats; if not, the program terminates.


START
Enter Passenger Details
Generate Drivers List (Random Distance, Rating, and Fare)
Display Driver Details
Select Matching Method
• Neorest Distance
• Highest Rating
• Lowest Fare
Compare Drivers Based on Selected Criteria
Assign Best Driver
Display Selected Driver
Save Ride Details to File (Ride History)
END

System Requirements

Hardware Requirements

- A computer or laptop
- Minimum 2 GB RAM
- Basic input/output devices (keyboard and monitor)

Software Requirements

- Operating System: Windows / Linux / macOS
- Programming Language: C++
- C++ Compiler (such as GDB, or Visual Studio Code with C++ extension)

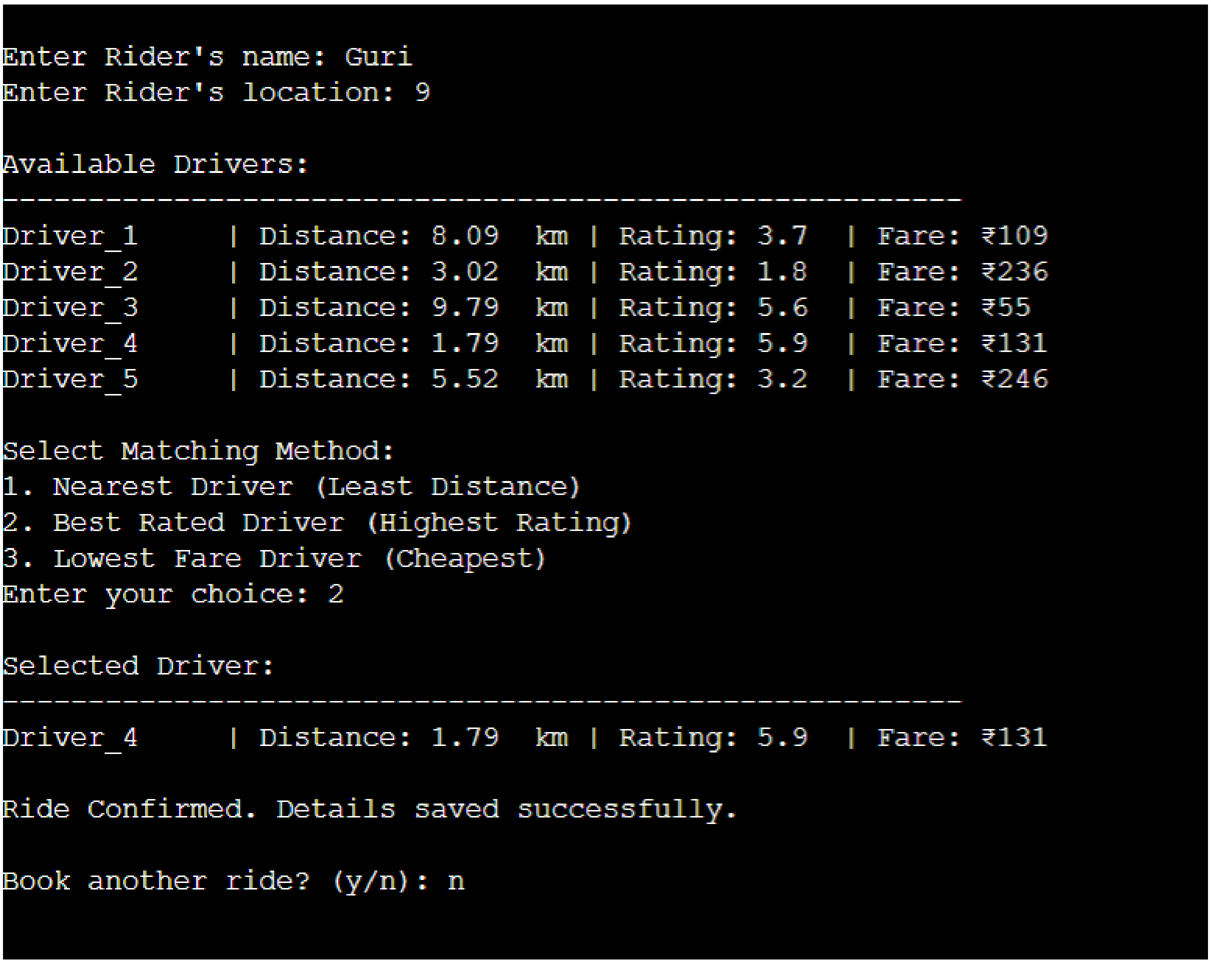
Design and Implementation

The system is designed using Object-Oriented Programming concepts in C++. The program is structured into two main classes: **Passenger** and **Driver**. The Passenger class is responsible for storing basic user details such as name and location. The Driver class holds information related to the drivers including their name, distance from the passenger, rating, and fare. Each class contains appropriate data members and member functions to maintain clarity and reusability.

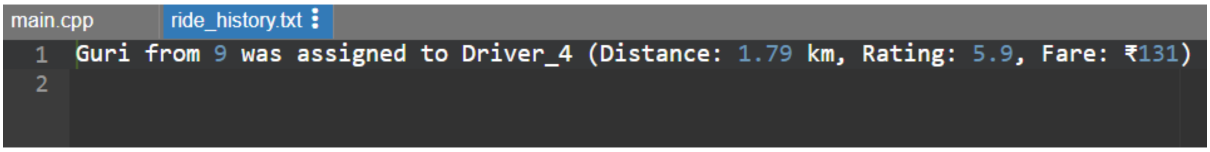
The program workflow begins with taking input from the passenger. The system then generates a list of drivers with randomly assigned values for distance, rating, and fare. These values simulate real-world variations in driver availability. After displaying the list of drivers, the user is asked to select a matching strategy: nearest driver, best-rated driver, or lowest fare. Based on the selection, the system compares all drivers and chooses the one that best fits the selected criteria.

For implementation, the program is written using standard C++ and compiled using a suitable compiler. The <fstream> library is used to handle file operations for storing ride history, making the data persistent for later reference. Conditional statements and looping structures are used to handle comparison logic and repeated ride booking. The formatted output is displayed using <iomanip> to ensure a clean and readable presentation. The final matched driver details are shown to the user, and the booking record is saved before the program exits or moves to the next ride request.

Output



```
Enter Rider's name: Guri
Enter Rider's location: 9

Available Drivers:
--------------------------------------------------------
Driver_1     | Distance: 8.09  km | Rating: 3.7  | Fare: ₹109
Driver_2     | Distance: 3.02  km | Rating: 1.8  | Fare: ₹236
Driver_3     | Distance: 9.79  km | Rating: 5.6  | Fare: ₹55
Driver_4     | Distance: 1.79  km | Rating: 5.9  | Fare: ₹131
Driver_5     | Distance: 5.52  km | Rating: 3.2  | Fare: ₹246

Select Matching Method:
1. Nearest Driver (Least Distance)
2. Best Rated Driver (Highest Rating)
3. Lowest Fare Driver (Cheapest)
Enter your choice: 2

Selected Driver:
--------------------------------------------------------
Driver_4     | Distance: 1.79  km | Rating: 5.9  | Fare: ₹131

Ride Confirmed. Details saved successfully.

Book another ride? (y/n): n
```



```
Guri from 9 was assigned to Driver_4 (Distance: 1.79 km, Rating: 5.9, Fare: ₹131)

```

Conclusion

This project successfully demonstrates the implementation of a simple cab matching system using the concepts of Object-Oriented Programming in C++. The program enables a passenger to enter their details, view available drivers, and select the most suitable driver based on different matching criteria. By using classes, data encapsulation, decision-making statements, and file handling, the system reflects how real-world ride-hailing platforms function at a basic level.

The project also provides practical experience in structuring and organizing code, handling user input, generating dynamic data, and maintaining records using external files. Through this exercise, the understanding of OOP principles becomes clearer, particularly in terms of designing classes and managing interactions between them. Although the system is simple, it can be expanded into a more advanced application by integrating real-time data, graphical interfaces, online databases, and map-based navigation.

Overall, the project fulfils its objectives and serves as a solid foundation for further learning and development in software design and real-world application modelling.

References

1. Balagurusamy, E. *Object Oriented Programming with C++*. Tata McGraw-Hill Education.
2. Schildt, Herbert. *C++: The Complete Reference*. McGraw-Hill.
3. Documentation of C++ Standard Library: https://cplusplus.com/reference/
4. C++ Tutorial Materials and Sample Programs from class notes.
5. Online compilation resources such as https://www.onlinegdb.com/