**BuckleUp Project Report**

**1. Introduction**

BuckleUp is a ride-booking application inspired by real-world ride-hailing platforms such as Uber and InDrive. The core idea is to enable users to register, log in, book rides, and view their history, while allowing drivers to register, log in, and accept rides. The system also provides an admin interface to monitor users, drivers, and rides. This project is implemented in both **C++ (console-based)** and **Java (GUI-based)** to showcase how cross-language implementations can work for the same business logic.

The project simulates key components of a real-time transport system such as:

* **User registration and login**
* **Driver registration and ride assignments**
* **Ride booking with fare estimation**
* **Admin controls and data overview**

**2. C++ Version – Console Based**

**2.1 Structure Overview**

The C++ version is a console application containing the following main components:

* **User Class:** Handles user data, authentication, and storage.
* **Driver Class:** Handles driver data and authentication.
* **Ride Class:** Manages booking logic, fare calculation, and ride details.
* **Admin Class:** Views statistics and lists of users, drivers, and rides.
* **Indrive Class:** Ties everything together, providing the menu and main control flow.

**2.2 Functional Highlights**

* **File-based Data Storage:** Data is persistently saved in users.txt, drivers.txt, and rides.txt.
* **Booking System:** Users can choose pickup and drop-off points, select ride types (Mini, A.C, Courier, Bike), and optionally propose a lower fare or add a tip.
* **Random Driver Assignment:** Drivers are assigned randomly based on the ride type.
* **Driver Panel:** Allows drivers to log in and accept randomly generated rides.
* **Admin Panel:** Allows an admin to log in and view statistics about users, drivers, and rides.

**2.3 Unique Features**

* Custom fare suggestion by users (negotiation model).
* Ride history is saved per user.
* Supports multiple user roles: User, Driver, Admin.

**3. Java Version – GUI Based**

**3.1 Structure Overview**

The Java implementation includes a GUI interface built using **Swing**, and consists of two key files:

* **Main.java:** Launches the application and handles UI switching.
* **IndriveGUI.java:** Contains the GUI layout and logic for login, booking, and ride confirmation.

**3.2 Functional Highlights**

* **Graphical User Interface:** Users interact through a windowed application, making the system user-friendly.
* **Login and Booking Flow:** Users log in, choose pickup/drop-off locations, and confirm bookings with estimated fare.
* **Input Validation:** Ensures that invalid or missing inputs are caught early.
* **Modular GUI Screens:** Multiple panels handle different stages of the ride booking process.

**3.3 Features Compared to C++**

| **Feature** | **C++ Version** | **Java GUI Version** |
| --- | --- | --- |
| User Interface | Console-based | GUI using Swing |
| Data Persistence | File system | Limited in GUI demo |
| User & Driver Registration | Yes | User only |
| Ride History | File-based logs | Not shown |
| Admin Dashboard | Yes | No |
| Tip & Negotiation Support | Yes | No |
| Random Driver Assignment | Yes | No |

**4. Conclusion**

This dual-language implementation of **InDrive** showcases how the same system can be developed across different platforms and interfaces. The **C++ version** focuses on a fully-functional backend system with persistence and admin controls, while the **Java version** demonstrates a clean and responsive **graphical front-end experience**.

This project helps learners and developers understand:

* Object-Oriented Programming principles in both C++ and Java.
* Practical implementation of file handling, class design, and user interaction.
* How to simulate real-world business logic in software projects.

**UML**

