

Group Project Final Report

Project Title:

Ride Booking System: Go-Do

Year / Section / Group Name / Number:

1-4 Group 4 – Go-Do

Team Members:

Name	Student ID	Role
Antiquerra, Simeon III B.	2024-05025-MN-0	QA/Test Engineer
Cellona, Dwayne A.	2024-02467-MN-0	Lead Developer
Depatoy, Axel John M	2024-03951-MN-0	UI/UX Designer
Jucutan, Kc Niña P	2024-18673-MN-1	Project Manager
Paligutan, Aaron Jasper D,	2024-04731-MN-0	UI/UX Designer
Villafuerte, Gabriel C.	2024-01338-MN-0	Lead Developer
Vito, Ymund Angelo C.	2024-04222-MN-0	UI/UX Designer

Instructor/Adviser:

Mr. Godofredo Tamalayan Avena

Date Submitted:

July 3, 2025

Table of Contents

1. Introduction
2. Objectives
3. Scope and Limitations
4. Methodology
5. System Design
6. Technologies Used
7. Implementation <User interface>
8. Testing and Evaluation
9. Results and Discussion (Challenges)
10. Conclusion
11. References
12. Appendices

Introduction

Go-Do is a ride-booking and driver management system designed to simplify transportation services for both users and administrators. The system is built with the guidance of Object-Oriented Programming principles, specifically inheritance, polymorphism, and encapsulation, to ensure modularity and scalability. To enhance user experience, the application features a graphical user interface (GUI) created with CustomTkinter. Additionally, file handling techniques will be implemented to manage and store ride and booking data persistently.

Objectives

- Apply object-oriented programming principles (inheritance, polymorphism, classes)
- Master Python file management for data persistence.
- Utilize Tkinter to develop graphical user interfaces.
- Implement core software engineering practices like encapsulation and modularity in a practical context.

Scope and Limitations

This project focuses on the development of a basic ride booking system that allows users to:

- Select from multiple vehicle types (Car, Van, Motorcycle)
- Book rides by providing destination and ride details
- Cancel or view existing bookings
- View available rides in the system
- Have Users and Admin Interface

The application uses Object-Oriented Programming concepts such as inheritance, encapsulation, and polymorphism. It also utilizes Python's Tkinter library to create a graphical user interface and basic file handling to store booking information persistently.

Limitations

- The system operates in a single-user mode and does not support account-based logins or multi-user management.
- Real-time ride availability or integration with GPS/location services is not implemented.
- Data storage is limited to local text or CSV files; no database or cloud storage was used.
- The GUI is designed for basic navigation only and may not be mobile-friendly.

Methodology

The developers followed a structured development approach consisting of five main phases: planning, design, development, testing, and deployment. This methodology ensured the systematic implementation of both functional and technical aspects of the **Go-Do** ride booking system.

The project team was composed of six members with clearly defined roles to streamline the workflow:

- **2 Developers** focused on backend logic, driver assignment, booking processes, and fare calculation.
- **3 UI Designers** created intuitive and visually appealing interfaces for the login, booking, and ride management screens using CustomTkinter and figma.
- **1 Project Manager** coordinated tasks, managed schedules, and oversaw project progress.
- **1 Test Engineer** performed rigorous testing and debugging to ensure reliability and usability.

1. Planning and Requirements Gathering

The team identified the core features of the system based on the outline and guidelines:

- User functionalities: book rides, view booking history, cancel rides, and track ride status
- Admin functionalities: drivers directory, view all bookings, and manage vehicles
- Data storage and retrieval

During this phase, tasks were assigned based on roles: front-end development and UI design for the UI team, backend programming for developers, test planning for the test engineer, and overall coordination by the project manager.

2. System Design

The system architecture was modeled using class diagrams to define the relationships between components such as user, vehicle, ride, and booking manager. Key OOP concepts applied include:

- **Encapsulation:** Keeping data Sensitive user data (like username and password) is protected using private variables and accessed only through getter and setter methods, as seen in the User class.
- **Inheritance:** Creating specific vehicle subclasses (Car, Van, Motorcycle) that inherit common properties and methods from a general Vehicle superclass.
- **Polymorphism:** Methods like `calculate_cost_with_tax()` behave differently depending on the vehicle type, allowing dynamic and reusable code.
- **Abstraction :** Defining an abstract base class Vehicle with abstract methods (e.g., `calculate_cost_with_tax()`) that must be implemented by all subclasses, ensuring a consistent interface while hiding implementation details.

The UI team designed wireframes and mockups for screens related to booking, browsing, and ride management.

3. Implementation

Development was done using Python. The UI team implemented the graphical user interface using CustomTkinter to allow users to interact intuitively with the system. Developers handled backend logic, including file handling for persistent storage of booking and ride data.

4. Testing and Debugging

The test engineer conducted module-wise testing to ensure correctness:

- Functionality tests for booking, canceling, and viewing rides.
- Validation of error-handling for invalid inputs and unavailable rides.

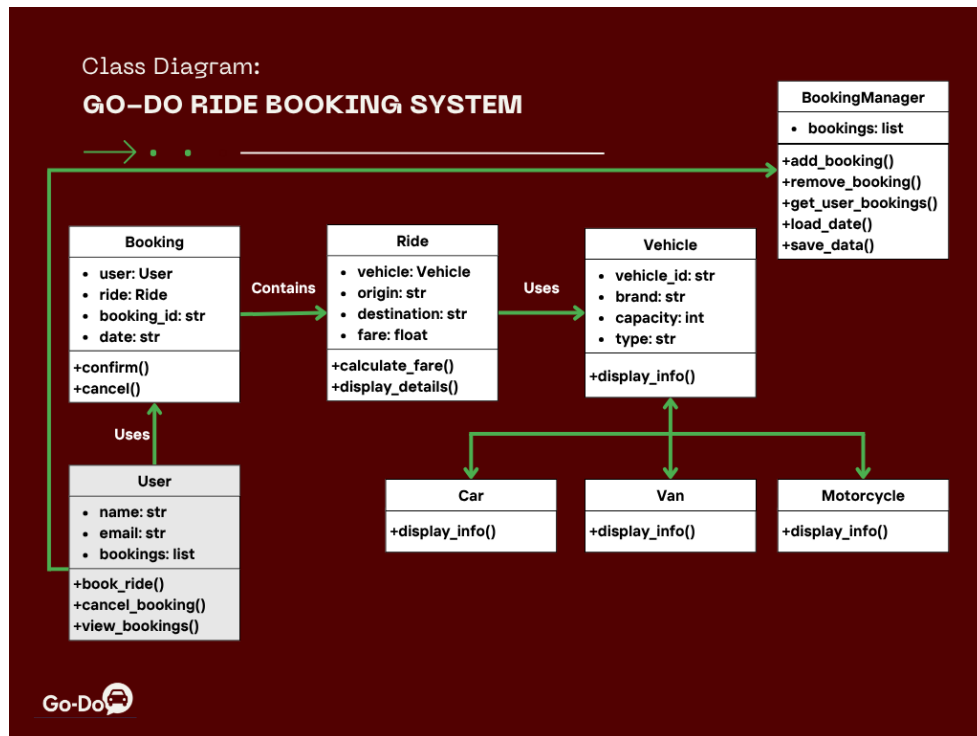
- GUI usability and consistency checks.

Bugs were addressed iteratively through manual testing and peer reviews.

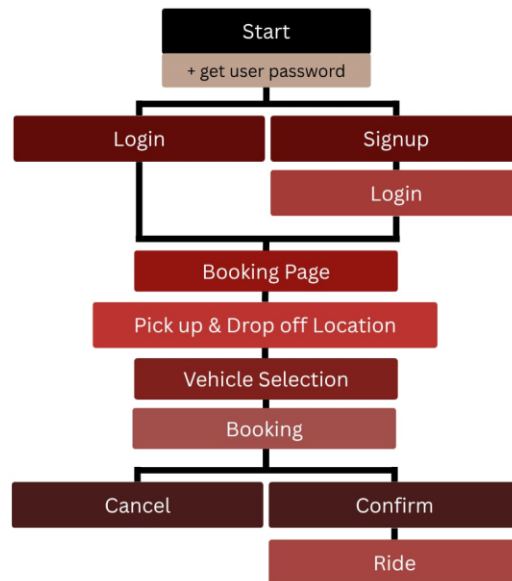
5. Deployment and Documentation

After stabilizing the system, the team packaged the final version and prepared documentation. This includes user guides, class structure explanations, test summaries, GUI screenshots, and sample data files.

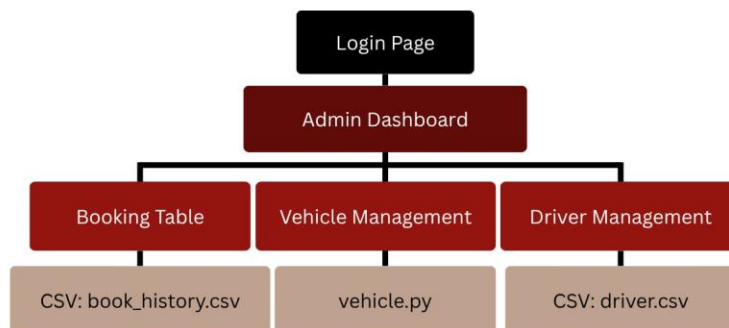
System Design



Flow Diagram (User):



Flow Diagram (Admin):



Technologies Used

This project was developed using a combination of modern Python tools, libraries, and design principles to build a functional and user-friendly ride booking system. The following technologies were used:

Frontend:

- **CutomTkinter** – used to develop a modern, responsive graphical user interface(GUI) with enhanced aesthetics and interactivity compared to the standard Tkinter module.
- **Figma** - utilized for designing and prototyping the UI/UX layouts before development, ensuring clear visual guidelines and user flow.

Backend:

- **Python** – Served as the primary programming language for all backend logic, data processing, and system operations.
- **CSV** – Utilized for lightweight and persistent data storage for bookings, user accounts, and vehicle information.

Design and Architecture:

- **Object-Oriented Programming (OOP)** – Ensured modularity, reusability, and maintainability of code through the use of classes and inheritance.
- **Modular Architecture** – Separated major features (e.g., booking, login, vehicle management) into distinct modules for better organization and scalability.

Tooling and Utilities:

- **pip** – Used for installing third-party libraries and managing dependencies.

Libraries and Modules:

- **Pillow** – Used for image processing and displaying icons or images within the interface.
- **Pathlib** – Handled platform-independent file path operations.
- **Tkintermapview** – Integrated map-based features such as selecting pickup and drop-off locations.
- **Math** – used for performing mathematical calculations required in the system.
- **Time** – utilized for handling time-based operations, such as timestamps and delays.
- **Threading** – employed to enable multi-threading for smoother performance and concurrent tasks.

- **OS** – used for file handling and interacting with the operating system.
- **Pandas** – applied for tabular data processing and manipulation, especially with CSV files.
- **bcrypt** – Implemented for secure password hashing and authentication
- **Pathlib** – used for modern and efficient handling of file system paths.

These technologies combined to create a system that is both visually intuitive and functionally robust.

Implementation

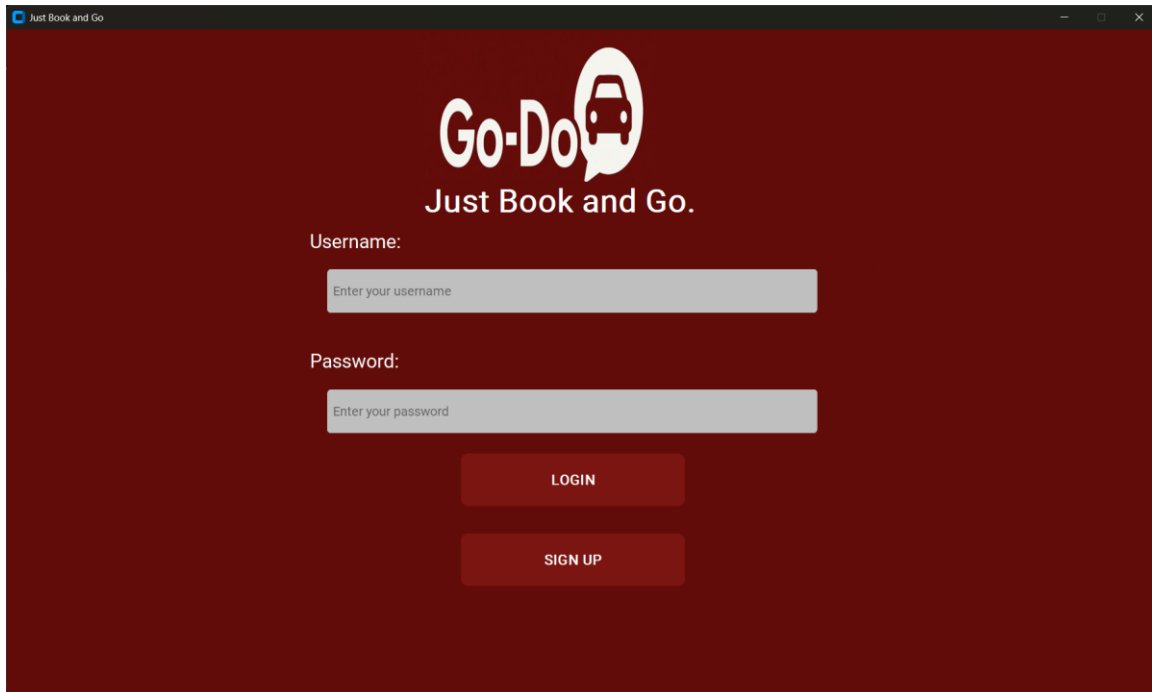
The ride booking system was implemented using Python with a modular, object-oriented architecture. The application is divided into multiple components, each responsible for specific features of the system. Both the frontend and backend work together to provide a seamless user experience. Below is a breakdown of the key modules and functionalities:

1. User Authentication Module

- Handles user login and signup processes using hashed passwords via the bcrypt library.
- Stores and verifies credentials from a secured CSV file.
- Prevents duplicate registrations and invalid login attempts.

Key Features:

- Username-password validation
- Secure storage using hashed passwords
- Redirect to user-specific dashboards based on role (user/admin)



The screenshot shows a web browser window titled "Just Book and Go". The page has a dark red background. At the top center is the "Go-Do" logo, which includes a car icon inside a speech bubble. Below the logo is the tagline "Just Book and Go.". The login form consists of two input fields: "Username:" and "Password:". Each field has a placeholder text "Enter your username" and "Enter your password" respectively. Below the password field are two buttons: "LOGIN" and "SIGN UP".

Just Book and Go

Go-Do

Just Book and Go.

Username:

Enter your username

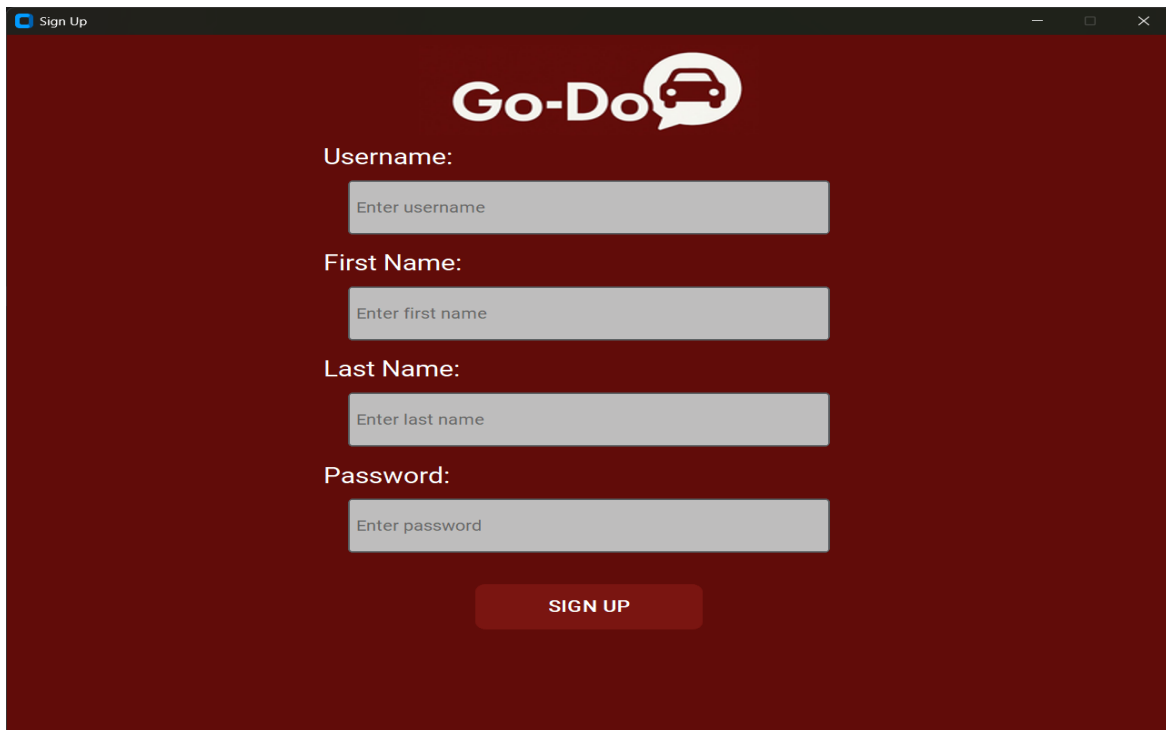
Password:

Enter your password

LOGIN

SIGN UP

Figure 1 Login Page



The screenshot shows a web browser window titled "Sign Up". The page has a dark red background. At the top center is the "Go-Do" logo, which includes a car icon inside a speech bubble. Below the logo are four input fields: "Username:", "First Name:", "Last Name:", and "Password:". Each field has a placeholder text: "Enter username", "Enter first name", "Enter last name", and "Enter password" respectively. Below the password field is a "SIGN UP" button.

Sign Up

Go-Do

Username:

Enter username

First Name:

Enter first name

Last Name:

Enter last name

Password:

Enter password

SIGN UP

Figure 2 SignUp Page

2. Booking Module

- Allows users to select pickup and drop-off locations via an interactive map powered by Tkintermapview.
- Offers multiple vehicle types such as Car, Van, and Motorcycle, implemented using inheritance from a base Vehicle class.
- Calculates total fare based on distance and cost-per-mile for the selected vehicle.

Key Features:

- Dynamic vehicle list display
- Route input using map widget
- Real-time fare estimation based on inputs
- Ride confirmation or cancellation

Go-Do - Modern Ride Booking App

Go-Do

Select Vehicle Type

	Car(4 Seater)	₱ 100.00
	Car(6 Seater)	₱ 120.00
	Mini Van	₱ 200.00
	Van	₱ 250.00
	Motorcycle	₱ 50.00

Driver's Information:

Driver Name:	Vehicle Description:
N/A	N/A
Plate Number:	Contact Number:
N/A	N/A

Price: 0 Pesos [Price Breakdown](#)

Select Mode of Payment:

☒ Cash ☐ Online Payment

[Cancel](#) [Confirm](#)

Figure 1 Booking Page (User)

3. Admin Dashboard Module

1. Provides administrative tools for managing the ride system.

2. Admins can view all current bookings, edit vehicle cost-per-mile settings, and monitor the rider directory.

Key Features:

- View full booking directory
- Vehicle fare adjustment interface
- Rider and vehicle management tools

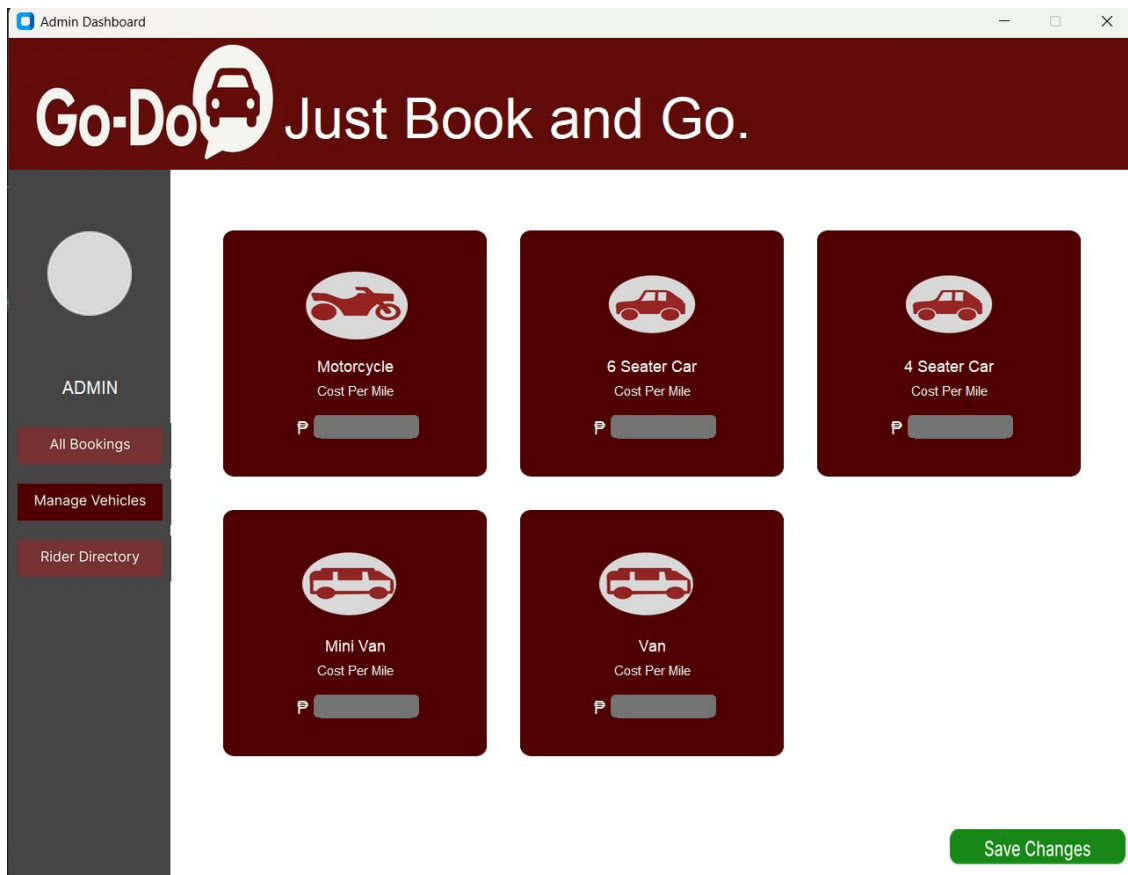


Figure 2 Admin Dashboard - Manage Vehicle

Admin Dashboard

Go-Do Just Book and Go.

Rider Directory

Vehicle_Type	driver_name	vehicle_name	contact_no	plate_no	status
Car(4 Seater)	Juan Dela Cruz	Toyota Corolla	9171234567	3F8B1Z	available
Car(4 Seater)	Maria Santos	Honda Civic	9182345678	X7N9K2	available
Car(4 Seater)	James Johnson	Ford Focus	9183456789	8T5JH3	available
Car(4 Seater)	Ana Reyes	Nissan Sentra	9184567890	K2L9Q8	available
Car(4 Seater)	Michael Lee	Hyundai Elantra	9185678901	J5V7M1	available
Car(6 Seater)	Josefina Cruz	Toyota Innova	9186789012	Q9X4Z7	available
Car(6 Seater)	David Smith	Mitsubishi Outlander	9187890123	A1B2C3	available
Car(6 Seater)	Luz Fernandez	Honda Odyssey	9188901234	D4E5F6	available
Car(6 Seater)	Ryan Brown	Ford Explorer	9189012345	P8R2V9	available
Car(6 Seater)	Sophia Garcia	Kia Carnival	9190123456	Z3X6V7	available
Mini Van	Carlos Mendoza	Toyota Hiace	9191234567	M4N8B2	available
Mini Van	Emily Johnson	Nissan NV350	9192345678	H5J7L3	available
Mini Van	Mark Gonzales	Ford Transit	9193456789	R9T2Y5	available
Mini Van	Linda Nguyen	Mercedes Sprinter	9194567890	F1C8D4	available
Mini Van	Juanita Lopez	Fiat Ducato	9195678901	L0Z7X3	available
Van	Ramon Santos	Ford Econoline	9196789012	V3G1M8	available
Van	Angela Brown	Chevrolet Express	9197890123	S9D2J6	available
Van	Benjamin Reyes	Dodge Ram Van	9198901234	C7K4N1	available
Van	Sarah Thompson	GMC Savana	9199012345	N2P8L9	available
Van	Ricardo Lim	Nissan Urvan	9190123456	W6E3R5	available
Motorcycle	Pedro Martinez	Honda CBR	9191234567	B7F4Z9	available
Motorcycle	Jennifer Lee	Yamaha R3	9192345678	E2M9P8	available
Motorcycle	Anthony Garcia	Suzuki GSX	9193456789	U8K1X7	available
Motorcycle	Ana Santos	Kawasaki Ninja	9194567890	Q5J2H4	available
Motorcycle	David Cruz	Ducati Monster	9195678901	T3V6L0	available

Figure 3 Admin Dashboard - Rider Directory

Admin Dashboard

Go-Do Just Book and Go.

Booking Dashboard

Driver_name	Vehicle	Pickup_point	Destination	Booking_ID	Price	Status
Michael Lee	Hyundai Elantra	Las Piñas	Makati	BOOK011	₱250.00	Successful
David Smith	Mitsubishi Outlander	Pasig	Taguig	BOOK012	₱306.00	Cancelled
Luz Fernandez	Honda Odyssey	Mandaluyong	Quezon City	BOOK013	₱374.00	Successful
Ryan Brown	Ford Explorer	San Juan	Manila	BOOK014	₱408.00	Successful
Sophia Garcia	Kia Carnival	Caloocan	Valenzuela	BOOK015	₱442.00	Cancelled
Mark Gonzales	Ford Transit	Makati	BGC	BOOK016	₱288.00	Successful
Linda Nguyen	Mercedes Sprinter	Pasay	Taguig	BOOK017	₱312.00	Cancelled
Juanita Lopez	Fiat Ducato	Alabang	Muntinlupa	BOOK018	₱273.00	Successful
Benjamin Reyes	Dodge Ram Van	Parañaque	Cubao	BOOK019	₱400.00	Cancelled
Sarah Thompson	GMC Savana	Makati	Ortigas	BOOK020	₱360.00	Successful
Ricardo Lim	Nissan Urvan	Las Piñas	Pasig	BOOK021	₱440.00	Successful
Pedro Martinez	Honda CBR	Malabon	Navotas	BOOK022	₱108.00	Cancelled
Anthony Garcia	Suzuki GSX	Pasig	Ortigas	BOOK023	₱90.00	Successful
Ana Santos	Kawasaki Ninja	Cubao	San Juan	BOOK024	₱144.00	Cancelled
James Johnson	Ford Focus	Quezon City	Makati	BOOK025	₱325.00	Successful
Juan Dela Cruz	Toyota Corolla	BGC	Makati	BOOK026	₱180.00	Successful
Maria Santos	Honda Civic	Taguig	Caloocan	BOOK027	₱375.00	Cancelled
Emily Johnson	Nissan NV350	Makati	Las Piñas	BOOK028	₱312.00	Successful
Ramon Santos	Ford Econoline	BGC	Taguig	BOOK029	₱200.00	Successful
Angela Brown	Chevrolet Express	Cubao	Manila	BOOK030	₱400.00	Cancelled

Figure 6 Admin Dashboard - Rider Directory

4. Data Management

- Booking data, user records, and vehicle information are all stored in .csv files for persistent access across sessions.
- File operations are managed using pathlib and pandas for structured data handling.

```
Book_history.csv > data
1 Driver_name,Vehicle,Pickup_point,Destination,Booking_ID,Price,Status
2 Michael Lee,Hyundai Elantra,Las Piñas,Makati,BOOK011,250.00,Successful
3 David Smith,Mitsubishi Outlander,Pasig,Taguig,BOOK012,306.00,Cancelled
4 Luz Fernandez,Honda Odyssey,Mandaluyong,Quezon City,BOOK013,374.00,Successful
5 Ryan Brown,Ford Explorer,San Juan,Manila,BOOK014,408.00,Successful
6 Sophia Garcia,Kia Carnival,Caloocan,Valenzuela,BOOK015,442.00,Cancelled
7 Mark Gonzales,Ford Transit,Makati,BGC,BOOK016,288.00,Successful
8 Linda Nguyen,Mercedes Sprinter,Pasay,Taguig,BOOK017,312.00,Cancelled
9 Juanita Lopez,Fiat Ducato,Alabang,Muntinlupa,BOOK018,273.00,Successful
10 Benjamin Reyes,Dodge Ram Van,Parañaque,Cubao,BOOK019,400.00,Cancelled
11 Sarah Thompson,GMC Savana,Makati,Ortigas,BOOK020,360.00,Successful
12 Ricardo Lim,Nissan Urvan,Las Piñas,Pasig,BOOK021,440.00,Successful
13 Pedro Martinez,Honda CBR,Malabon,Navotas,BOOK022,108.00,Cancelled
14 Anthony Garcia,Suzuki GSX,Pasig,Ortigas,BOOK023,90.00,Successful
15 Ana Santos,Kawasaki Ninja,Cubao,San Juan,BOOK024,144.00,Cancelled
16 James Johnson,Ford Focus,Quezon City,Makati,BOOK025,325.00,Successful
17 Juan Dela Cruz,Toyota Corolla,BGC,Makati,BOOK026,180.00,Successful
18 Maria Santos,Honda Civic,Taguig,Caloocan,BOOK027,375.00,Cancelled
19 Emily Johnson,Nissan NV350,Makati,Las Piñas,BOOK028,312.00,Successful
20 Ramon Santos,Ford Econoline,BGC,Taguig,BOOK029,200.00,Successful
```

Figure 7 Booking History.csv

```
driver.csv > data
1 Vehicle_Type,driver_name,vehicle_name,contact_no,plate_no,status
2 Car(4 Seater),Juan Dela Cruz,Toyota Corolla,9171234567,3F8B1Z,busy
3 Car(4 Seater),Maria Santos,Honda Civic,9182345678,X7N9K2,busy
4 Car(4 Seater),James Johnson,Ford Focus,9183456789,8T5JH3,busy
5 Car(4 Seater),Ana Reyes,Nissan Sentra,9184567890,K2L9Q8,busy
6 Car(4 Seater),Michael Lee,Hyundai Elantra,9185678901,J5V7M1,busy
7 Car(6 Seater),Josefina Cruz,Toyota Innova,9186789012,Q9X4Z7,busy
8 Car(6 Seater),David Smith,Mitsubishi Outlander,9187890123,A1B2C3,available
9 Car(6 Seater),Luz Fernandez,Honda Odyssey,9188901234,D4E5F6,available
10 Car(6 Seater),Ryan Brown,Ford Explorer,9189012345,P8R2W9,available
11 Car(6 Seater),Sophia Garcia,Kia Carnival,9190123456,Z3X6V7,busy
12 Mini Van,Carlos Mendoza,Toyota Hiace,9191234567,M4N8B2,available
13 Mini Van,Emily Johnson,Nissan NV350,9192345678,H5J7L3,available
14 Mini Van,Mark Gonzales,Ford Transit,9193456789,R9T2Y5,available
15 Mini Van,Linda Nguyen,Mercedes Sprinter,9194567890,F1C8D4,available
16 Mini Van,Juanita Lopez,Fiat Ducato,9195678901,L0Z7X3,available
17 Van,Ramon Santos,Ford Econoline,9196789012,V3G1M8,busy
18 Van,Angela Brown,Chevrolet Express,9197890123,S9D2J6,available
19 Van,Benjamin Reyes,Dodge Ram Van,9198901234,C7K4N1,available
20 Van,Sarah Thompson,GMC Savana,9199012345,N2P8L9,available
21 Van,Ricardo Lim,Nissan Urvan,9190123456,W6E3R5,available
```

Figure 8 Driver.csv

Object Oriented Programming Concept

1. Encapsulation

```
import uuid

class User:
    def __init__(self, username, password, first_name=None, last_name=None, user_id=None):
        if not username or len(username) < 5:
            raise ValueError("Username must be at least 5 characters long.")
        self._user_id = user_id or str(uuid.uuid4())
        self._username = username
        self._password = password
        self._first_name = first_name
        self._last_name = last_name
```

Sensitive user data (like username and password) is protected using private variables and accessed only through getter and setter methods, as seen in the User class

2. Inheritance

```
class Car(Vehicle):
    def __init__(self, vehicle_name, vehicle_type, capacity):
        super().__init__(vehicle_name, vehicle_type, capacity)

class Car4Seater(Car):
    def __init__(self, vehicle_name):
        super().__init__(vehicle_name, "Car4Seater", 4)

class Car6Seater(Car):
    def __init__(self, vehicle_name):
        super().__init__(vehicle_name, "Car6Seater", 6)
```

Specific vehicle types (e.g., Car4Seater, Minivan) inherit shared properties and methods from a general Vehicle superclass, reducing code duplication.

3. Polymorphism

```
45
46     def calculate_cost_with_tax(self, distance):
47         base_cost = self.calculate_cost(distance)
48         tax = self.calculate_tax(distance)
49         base_fare = self.BASE_FARES.get(self.vehicle_type, 0.0)
50         total_base = base_cost + base_fare
51         return total_base, tax
52
```

Methods like calculate_cost_with_tax() behave differently depending on the vehicle type, allowing dynamic and reusable code.

4. Abstraction

```
from abc import ABC

class Vehicle(ABC):
    TAX_RATES = {
        "Car4Seater": 0.05,
        "Car6Seater": 0.05,
        "Minivan": 0.05,
        "Van": 0.06,
        "Motorcycle": 0.04,
    }
```

The Vehicle class serves as an abstract base class, defining a common structure for all vehicle types and preventing direct instantiation.

Testing and Evaluation

The system underwent informal but thorough testing during the development phase to ensure functionality and usability. Each core module—user login, ride booking, and admin management—was tested individually to confirm proper behavior under typical and edge-case conditions.

- **Functionality Testing:** All features, including booking creation, cancellation, fare computation, and CSV-based storage, were manually tested by multiple group members. Inputs such as blank fields, incorrect login credentials, and unsupported vehicle types were used to test validation logic.
- **GUI Evaluation:** The interface was checked for responsiveness and clarity. Testers confirmed that all buttons, labels, and windows performed as expected. Layout consistency and visual flow were considered during interface reviews.
- **Error Handling:** Various forms of invalid input were deliberately entered to test the robustness of error-handling routines. For example, attempting to log in with non-existent accounts triggered appropriate warning messages. Errors related to file paths or data access were also resolved during this phase.
- **Performance Check:** Basic performance testing was conducted by simulating multiple user actions in rapid succession. The system was able to handle multiple bookings and form interactions without crashing or freezing.

While formal automated testing and QA tools were not utilized due to time constraints, the application passed all essential manual tests and was deemed ready for user interaction.

Results and Discussion

The system successfully demonstrated the core functionalities of a ride booking application using Python. Users can:

- Browse and choose different types of vehicles
- Enter ride details and confirm bookings
- View and cancel rides through the GUI

The application correctly utilized object-oriented principles. Each vehicle type behaved according to its subclass properties, showcasing polymorphism. The booking records were saved and retrieved from local files, proving the effectiveness of the file-handling implementation.

Through testing, minor bugs such as missing input validation and file overwrite errors were encountered and resolved. GUI usability was tested through sample use inputs, and the interface was deemed intuitive for basic tasks. The ride fare calculation was also consistent across vehicle types.

Overall, the system met the objectives defined during the planning phase and served as a practical implementation of OOP concepts combined with basic GUI and data handling.

Conclusion

The Go-Do ride booking system project's success showcased the application of Object-Oriented Programming in solving real-world problems. By integrating core principles like inheritance, encapsulation, and polymorphism, the program demonstrated modular and maintainable code structures. The use of CustomTkinter allowed the developers to create a user-friendly interface, while file handling ensured data persistence without requiring a database.

Although the system has limitations in terms of scalability and real-time features, it serves as a strong foundation for future development, proving that even a simple application can stimulate real-world operations when built on solid programming principles.

Future Works

To enhance the functionality and user experience of the ride booking system, the following features are proposed for future development:

1. Waypoint Marker Integration

Incorporating a waypoint system would allow users to add multiple stops between their pickup and drop-off locations. This feature would be especially useful for ride-sharing or deliveries with multiple drop points. Integration with Tkintermapview or a similar mapping API can support this enhancement.

2. Rider Selection Option

An advanced feature for users would be the ability to select or assign preferred riders (drivers) based on availability, rating, or ride history. This would involve developing a rider directory with basic profiles and a system for assigning or requesting specific drivers.

References

Dwaynefg. (2025). Booking [Source code]. GitHub. <https://github.com/Dwaynefg/Booking>

Pillow (Version 10.3.0) [Computer software]. (2025). Retrieved from <https://python-pillow.org/>

pandas (Version 2.2.2) [Computer software]. (2025). Retrieved from <https://pandas.pydata.org/>

bcrypt (Version 4.1.3) [Computer software]. (2025). Retrieved from <https://pypi.org/project/bcrypt/>

tkintermapview (Version 1.27) [Computer software]. (2025). Retrieved from <https://github.com/TomSchimansky/TkinterMapView>

Appendices

I Installation Guide GitHub URL:

GitHub URL:

<https://github.com/Dwaynefg/Booking>

Requirements

- Python 3.x
- pip (Python package manager)

Steps to Install and Run

1. Download the Project Files

- Visit: <https://github.com/Dwaynefg/Booking>
- Click the green “Code” button → choose “Download ZIP”
- Extract the ZIP file to any location (e.g., Downloads, Desktop, etc.)

2. Open your IDE

- Open IDLE, VS Code, PyCharm, or any Python editor you use.
- In the editor, go to File → Open and navigate to the folder you extracted.

- Select the main script: login_page.py

3. Create a .env File

In the same folder where login_page.py is located, create a file named .env with this content (for Admin account):

```
ADMIN_USERNAME=admin
```

```
ADMIN_PASSWORD_HASH=$2b$12$uTu67UluLgwIpGkRghX.Ve2n4BnpaK6R3gvqJX9LeHpsR4yqS2Llm
```

4. Install Dependencies

Open your terminal or command prompt and install the required packages:

```
pip install -r requirements.txt
```

5. Run the Application

From your Python editor or IDE, run login_page.py.

II. User Manual Login

1. Enter your username and password.
2. Click the Login button to proceed.

Sign Up

1. Click the Sign Up button.
2. Fill in the required fields:
 - Username (must be at least 5 characters)
 - First Name
 - Last Name
 - Password
3. After successfully creating your account, return to the login screen.
4. Enter your username and password, then click Login to continue.

Booking a Ride

1. Select your pickup and drop-off locations by clicking on the map.
2. Choose your preferred vehicle by clicking one of the buttons on the right.

3. Click the Confirm button to book your ride.

Canceling a Booking

1. You have 15 seconds after booking to cancel your ride.
2. To cancel, click the Cancel Ride button within the allowed time.