Group Project Final Report

Project Title: Safe and Dry Ride (SADride) Course: BSCPE 1-4

Object Oriented Programming

Year / Section / Group Name / Number:

Group 8 - SADride

Team Members:

Name	Student ID	Role
Earl Vanesse Lumabi	2024-04985-MN-0	Lead Developer & Project Developer
Delfin Simbulan III	2024-02727-MN-0	Back-end Developer & Documentation Assistant
Robert Aron Rivera	2024-09019-MN-0	UI/UX Design & Front-end Integrator
Jowie Neilson Odrunia	2024-04847-MN-0	UI/UX Design & Front-end Integrator
James Pol Quimen	2024-00264-MN-0	QA/Test Engineer & Documentation Assistant
Isaiah Modesto	2024-16637-MN-0	QA/Test Engineer & Code Reviewer
Prince Jeoff Araja	2024-04758-MN-0	Research Assistant

Instructor/Adviser: Godofredo T. Avena

Date Submitted:

July 04, 2025

Table of Contents

Introduction

SADRide (Safe and Dry Ride) is designed to simulate a basic transportation service platform that allows users to book rides from one destination to another, manage the rides that they have booked, and choose from an array of vehicles that they would like to be transported by – cars, vans and bikes.

This system allows users to select a vehicle type, calculate the cost of a ride, select the start-point and the end-point, and manage reserved rides in one interface. The system offers basic functionalities, such as booking rides, viewing all existing bookings, and letting users cancel bookings at any time. In addition, the program also handles data persistence, meaning that bookings are saved even after the application is closed and can be accessed again after being reopened.

By developing this application, the aim is to emulate the functionality of actual ride-booking programs and software in a simple and easy-to-understand fashion. The system is not just a useful simulation of an actual transportation service, but also a foundation for any future innovations and enhancements that could be done to the ride-booking software model.

Objectives

The aim is to develop a functional Ride-Booking System that makes it easy for the user to travel from one point to another via the safest and fastest possible routes.

- **Develop** a functional ride-booking desktop application.
- **Calculate** trip cost dynamically based on distance, vehicle type, and taxation.
- **Enable** users to book, view, and cancel rides, with history retained for reference.
- **Provide** an intuitive GUI that mimics real-world ride-hailing apps.

Scope and Limitations

The Ride Booking System is designed to provide a basic but functional platform for users to book different types of vehicles (e.g., Motorcycle, Car, Electric Car, Van) within a mapped area. The system offers the following features.

- **User Login:** Single-field username entry.
- **Booking Interface:** Click-to-set pickup & drop-off on an embedded map.
- **Distance & ETA Estimation:** Automated via geopy calculations.
- **Vehicle Selection:** Motorcycle, Car, Electric Car, Van, Taxi—each with unique rate logic.
- **Booking Summary:** Displays ETA & fare before confirmation.
- **Booking Management:** View, cancel, finish rides and all saved to CSV.
- **Modern GUI:** Built with CustomTkinter + TkinterMapView.
-

LIMITATIONS

- **No real-time GPS tracking.**
- **No account/password system; user is identified only by name input.**
- **No driver assignment simulation.**
- **No integrated payment processing.**

Assumptions:

- All distances are straight- lines between points.
- Estimated travel time (ETA) is simulated using fixed average speed, based on straight-line distance.

Methodology

The development followed a structured but collaborative approach:

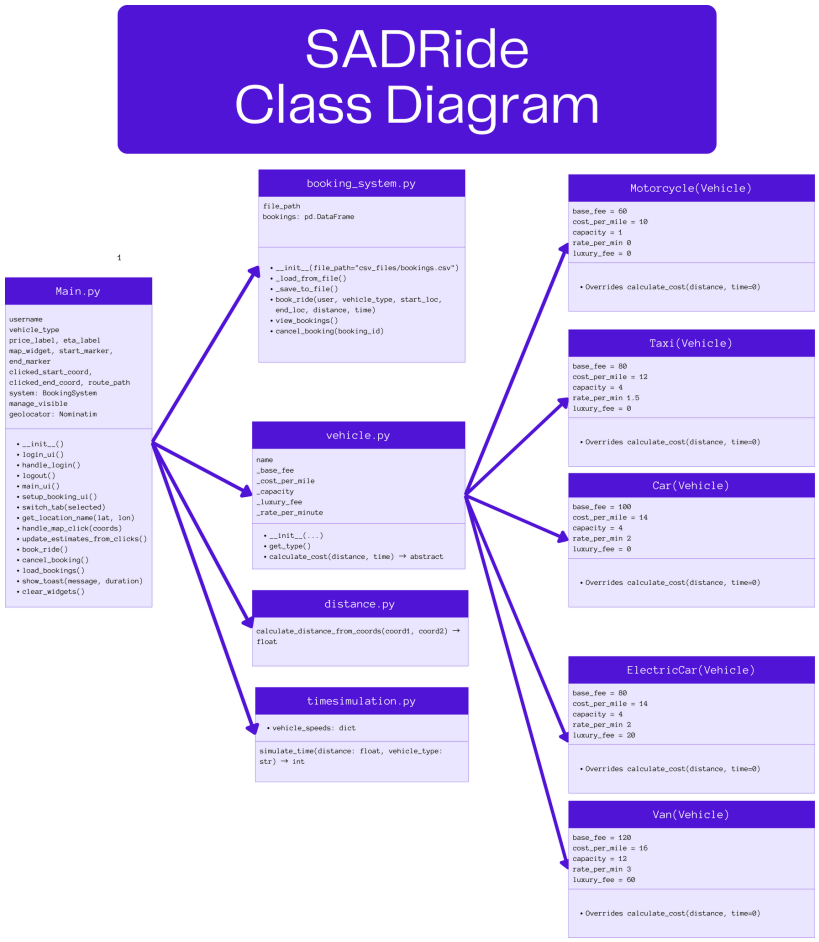
1. **Initial Planning & Task Delegation:** Members chose their roles. Backend developers began implementing the ride and vehicle logic while UI/UX members worked on initial mockups and design flow.
2. **Backend Development:** Booking system, vehicle calculations, and CSV file management were created and tested independently.
3. **Frontend Integration:** Once the backend was functional, it was integrated into the GUI with map interaction and theming.
4. **Testing Phase:** QA members verified all features worked as intended. Bugs were logged and fixed collaboratively.

5. **Finalization:** All modules were cleaned, minor UI issues resolved, and final documentation was written.

Despite hardware issues and creative blocks, the team maintained steady progress through communication and division of labor.

System Design

Class Diagram



How the program works:

- Upon starting, the user is shown a login screen where they are prompted to enter their login details.
- Afterwards, they are shown the main booking screen with two tabs: Book Ride, and Manage Bookings.
- For the Book Ride tab, it shows a map that lets them set their start, and their destination. They can also select their preferred way of transportation, being able to choose between a:
    - Motorcycle

    - Car
    - Van
    - Taxi
    - Electric Car
- The user is then shown the estimated travel time, along with the cost of the ride. The ride is then booked, and is given an ID.
- For the Manage Bookings tab, the user is shown a record of all their previously booked rides, showing them their active bookings, finished and their canceled bookings.
    - In the same tab, there is an option for cancelling and finishing bookings. If the user wants to cancel a booking, they take the ID from the list of previously-booked rides, enter it into the input box, and press the Cancel Booking button. It is the same for the finish booking, if the user wants to finish the booking, they take the ID, enter it into the input box and press Finish Button.
- Even if the user leaves the application and comes back to it, their booking history will still be there.

Technologies Used

The programming language used was Python, version 3.13. For the GUI, Tkinter was used, and the general IDE used for coding was Visual Studio Code. Listed below are the Modules used in the making of the program.

- Tkinter
- CustomTkinter
- Threading
- abc
- Random
- Pandas
- GeoPy
- Datetime

Implementation

ui.py

Purpose: Core GUI using customtkinter and tkintermapview.

Features:

- Login screen
- Tab system for booking and managing rides

- Map interface for selecting start and end points
- Vehicle type selection
- ETA and fare display
- Ride booking and cancellation

booking_system.py

Purpose: Handles logic for saving, retrieving, finishing and canceling bookings.

Features:

- Stores bookings in a CSV file
- Manages unique booking IDs
- Filters bookings by user
- Updates status on finish and cancellation

vehicle.py

Purpose: Object-oriented classes to represent each type of vehicle.

Subclasses:

- Motorcycle
- Taxi
- Car
- ElectricCar
- Van

Functionality: Each class calculates total fare using its own formula.

distance.py

Purpose: Calculates the distance of the start and end point in the map using latitude and longitude

timesimulation.py

Purpose: Simulates the estimated time of arrival based on the distance and vehicle average speed specified

Testing and Evaluation

Testing Methods

- **Manual Testing:** Performed end-to-end walkthroughs on all features (login, booking, cancellation, finishing rides).
- **Functional Tests:** Ensured proper vehicle cost computation, ETA generation, and route display.
- **Bug Simulation:** Invalid Booking IDs, rapid-click map selections, rapid-login/logout simulation and empty login field.
- **Theme Toggle Test:** Verified switching appearance modes mid-session.

Test Case	Expected Result	Status
Booking with Motorcycle	Displays ETA and correct fare	Pass
Entering blank name	Shows error and blocks login	Pass
Cancel ride with wrong ID	Shows "Booking not found"	Pass
Toggle theme during session	GUI switches instantly	Pass
Booking saves to CSV	File contains new entry	Pass

Results and Discussion

The three main objectives of this project are: to provide a program for booking rides, to estimate the total cost of a ride (depending on what vehicle was used, the distance traveled, and any possible taxes), and provide the quality service via the app, ensuring users can book rides, access receipts in cases where they are needed, and finishing/cancelling booked rides at anytime in case the user needs/wants to. And after finalizing the project, the developers proved that the final project meets its objectives.

Some challenges the team encounters during the making of this projects are:

- Several team members experienced creative blocks during UI planning, which were resolved through group brainstorming and discussion.
- Integration of frontend and backend initially caused coordination delays but was resolved through version control and shared testing.

Conclusion

Throughout the SADride project, our group improved in software development, version control, and user-centered design. We strengthened our understanding of Python, object-oriented design, and modular programming.

Most importantly, we learned the value of teamwork and adaptation, recognizing when to support one another, how to resolve technical hurdles, and how to deliver a completed project through clear goals and planning.

Future Work

- **Real-Time Location Tracking**: Provides real time location for both driver and the user for faster tracking and real time updates during the trip.
- **User Authentication**: Ensuring the safety of the drivers to know who are the persons who will ride on his mode of transportation provided.
- **Driver Assignment**: Allowing the system to automatically match a driver to a passenger and shows ride information.
- **Payment Integration**: Enables users/passengers to pay fares, improving transaction security for the future of the app.
- **Dashboard Tab Implementation:** Summarizes data that are essential for the user to be seen on the front page.

References

Schimansky, T. (2020). CustomTkinter.

https://customtkinter.tomschimansky.com

Gillies, S. (2008). Geopy Documentation

https://geopy.readthedocs.io/

McKinney, W. (2009). Pandas Documentation

https://pandas.pydata.org/docs/

TomSchimansky. (n.d.). GitHub - TomSchimansky/TkinterMapView:

A python Tkinter widget to display tile based maps like OpenStreetMap or Google Satellite Images. GitHub. https://github.com/TomSchimansky/TkinterMapView

Threading — Thread-based parallelism. (n.d.). Python Documentation.

https://docs.python.org/3/library/threading.html

SADride Ride Booking System
User Manual

by Group 8

Lumabi, Earl Vanesse
Rivera, Robert Aron
Odrunia, Jowie Neilson
Simbulan, Delfin
Modesto, Isaiah
Quimen, James Pol
Araja, Prince Jeoff

July 04, 2025

TABLE OF CONTENTS

INSTALLATION

Prerequisites:

- Python 3.11 or newer installed
- Any IDE of choice
- Internet connection for installing dependencies

Steps:

1. Install the dependencies on **requirements.txt** in either:
    a. Windows Command Prompt/Powershell
        i. If installing from powershell, you need to navigate the folder first by entering **cd "C:\Path\To\Your\Project"**
    b. IDE Terminal (preferably as it lowers the chance of modules not being detected)

    Type "**pip install -r requirements.txt**".

    NOTE: If pip doesn't work, use this instead "**python -m pip install -r requirements.txt**"
2. Verify installation completes without errors.

LAUNCHING THE APP

After you've done installing the dependencies, you can now open the app using any IDEs(e.g., Visual Studio Code, PyCharm, Thonny).

Simply locate the main.py and run it, the SADride window should appear within a few seconds.

Logging In

The main window shows the login screen, simply enter your username in the field provided.

After that, press **Enter** or click **Log In.** You will be redirected into the main menu.

Booking a Ride

Now, there is a map in the left side of the window, click on the map to select your **Start** location, a start pin will appear. Click again to set your **Destination**, an end pin will appear and a route line will be shown.

Choose your preferred **vehicle type**(e,g. Motorcycle, Car, Van)
The **ETA** and **Total Cost** will show up on the right panel after that.

Click **Book Ride** to place a booking. A confirmation will show up alongside the booking ID.

Managing Bookings

Switch to the **Manage Bookings** tab to:

- View all your current and past bookings.
- **Cancel** a booking by entering its ID and clicking the cancel button.
- Mark a booking as finished by entering the ID and clicking the **Finish** button

Personalization and Display

Theme Toggle

- Use the toggle switch in the right side of the logout button to toggle between **light** and **dark** mode. (Note: The theme preference resets when the app is restarted).