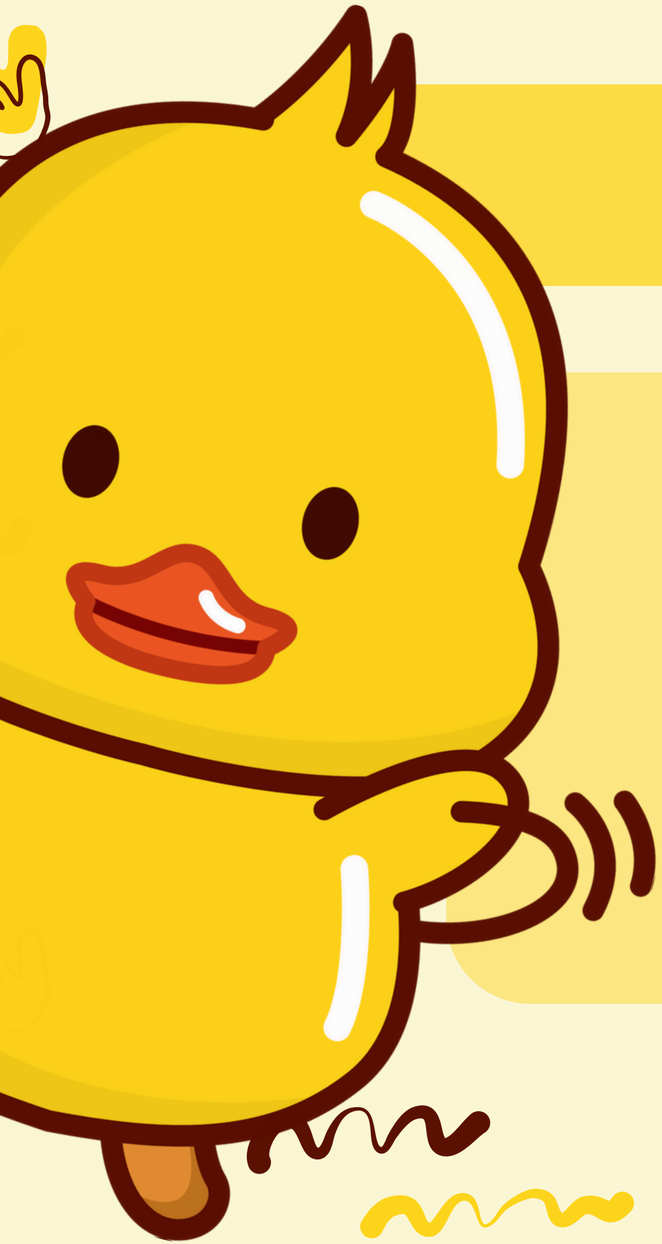
BSCPE 1-7

DUCKDASH

A RIDE BOOKING SYSTEM

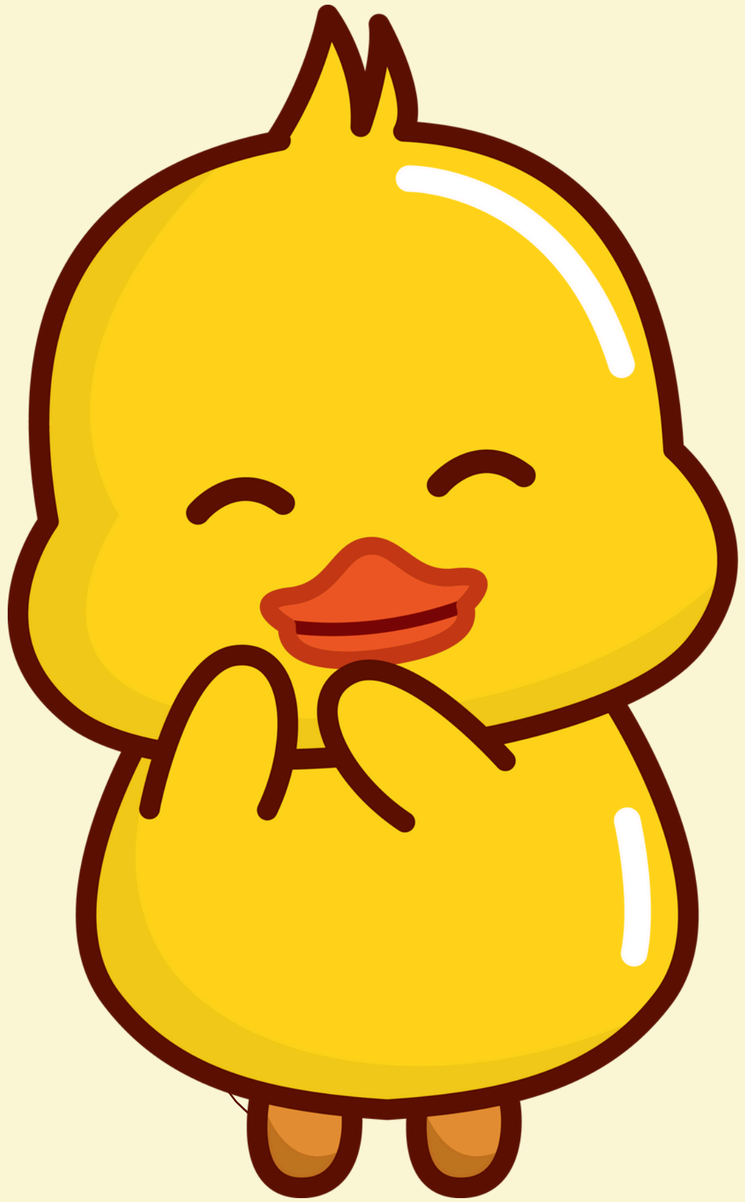
BY: GROUP 6



INTRODUCTION

Our Ride Booking Program was made with the thought of being an indie competitor to industry staples like Grab, Angkas, and MoveIt. Our systems are meant to be an all-in-one hub for Ride Sharing to reduce the need of moving between apps and maximize convenience for the end user.

OBJECTIVES



OBJECTIVES

General Objective:

- To develop an easy to understand, functional, and fun ride-booking application that connects passengers to drivers.

OBJECTIVES

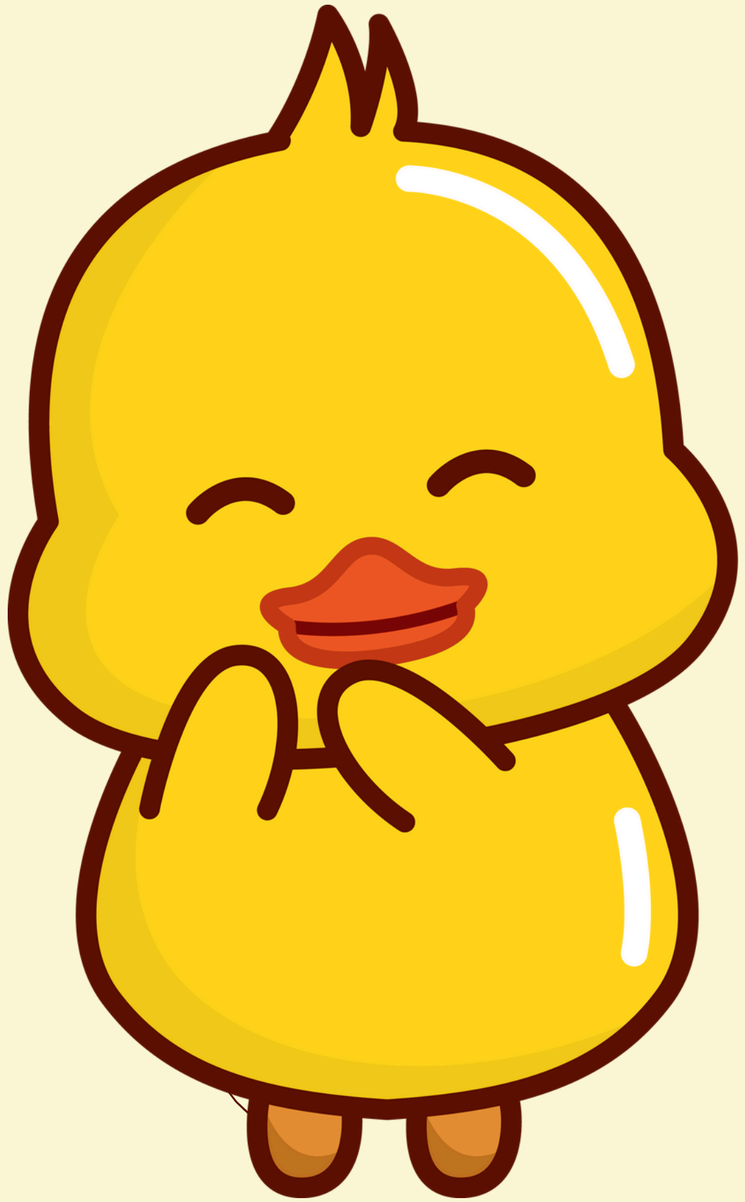
Specific Objectives

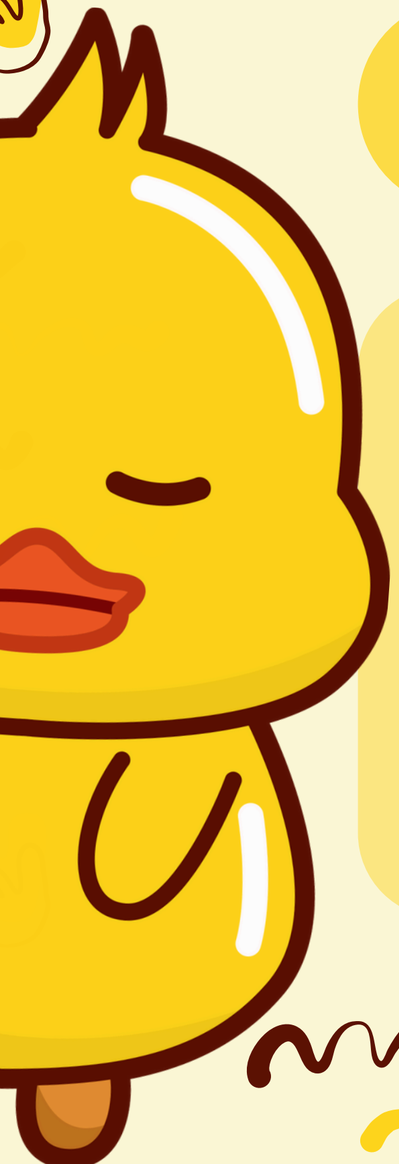
1. User Registration and Profile Management:
   - To allow riders and drivers to create and manage personal profiles, including contact information, vehicle details (for drivers), and payment preferences.
   -
2. Location-Based Ride Booking:
   - To implement a geolocation feature that enables riders to find nearby drivers based on their request.

OBJECTIVES

3. Vehicle Type Selection:
    - To provide users with the option to choose from multiple vehicle types (car, motorcycle, van) based on their preferences and available options, offering flexibility to riders.

SCOPE & LIMITATIONS



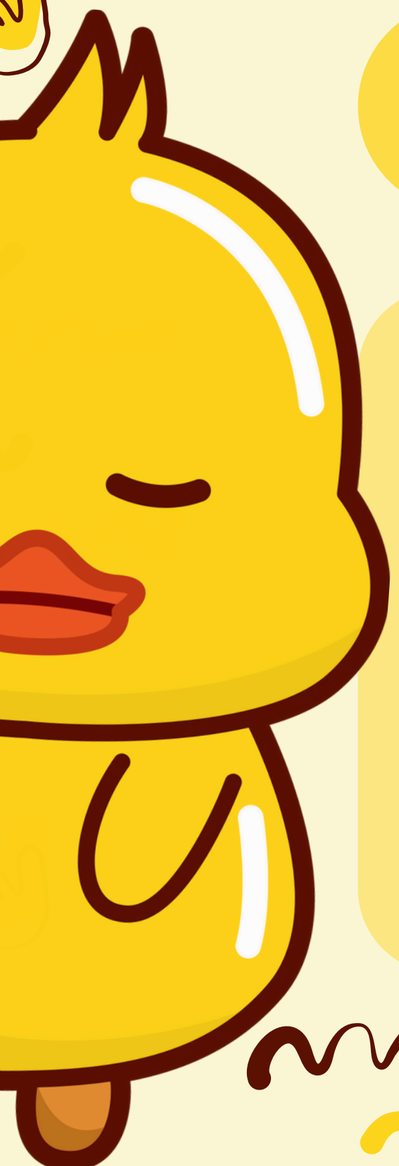


SCOPES AND LIMITATIONS

SCOPE:

- User Registration and Profile Creation
- Location-Based Services
- Ride Booking and Vehicle Selection
- Payment Integration
- Ride Tracking
- Ratings and Reviews
- Admin Dashboard

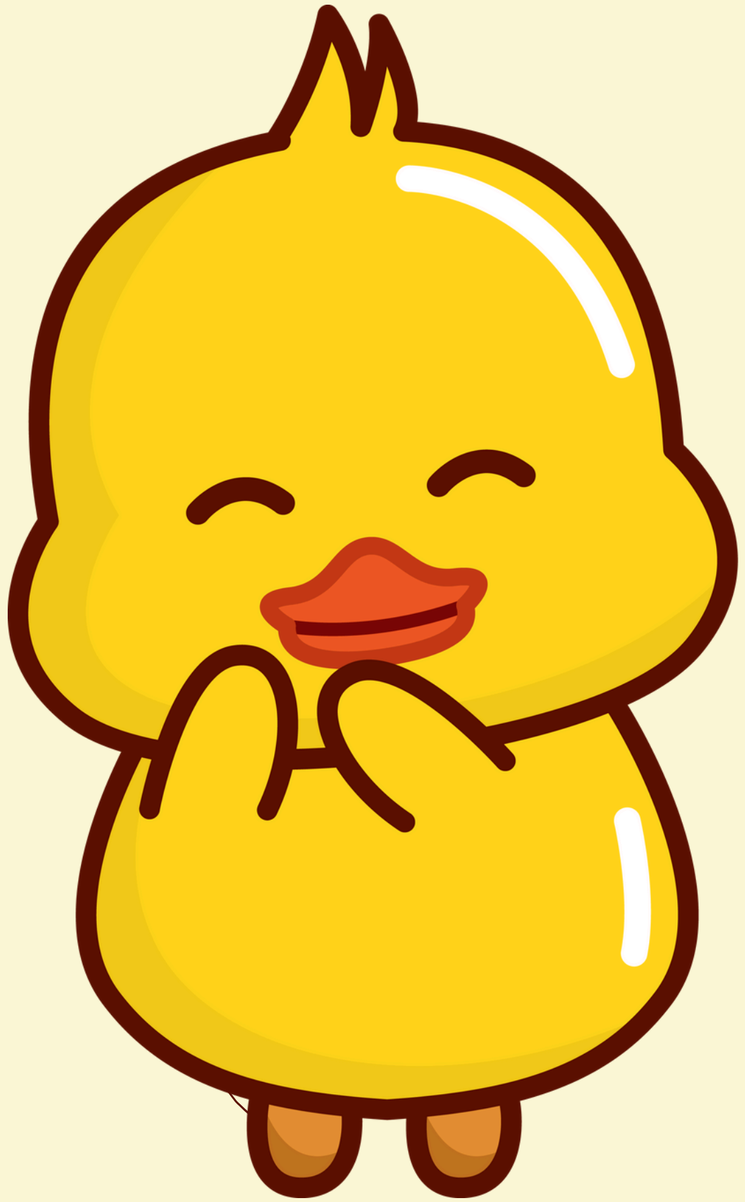
SCOPES AND LIMITATIONS



Limitations:

- Platform Limitation
- Limited Payment Methods
- Lack of Advanced AI Features
- Geolocation Accuracy
- Limited Testing Coverage
- Security Constraints
- No Multilingual Support
- Limited User Load

METHODOLOGY



METHODOLOGY

The development of the Ride Booking Application followed a rushed development, though planning was conceived early, the implementation of such plans was delayed until recently because of exams.

METHODOLOGY

1. Planning Phase

During the planning phase, we decided to divide the work into parts that each member is capable of and experienced but still left room for members to approach any aspect of the program if they so wished or needed to. The main concepts and ideas of the program were brainstormed by the entire group, adding features that would be great to add to the program. In this phase, we also committed to using Github Packages like Custom Tkinter (For GUI) and Tkinter MapView (For Google Maps integration) for an easier and more pleasing look to our program. We also committed to utilizing GitHub as our main platform to share our code and track progress.

METHODOLOGY

2. Design Phase

The design phase was divided into two key tasks:

- User Interface (UI) Design: One team member, specializing in design, is responsible for creating a mockup of the program with no functionality but gives the idea of what the program could look and flow together.
- Database Design: Another team member with experience in database management focused on designing the database system. The database utilized Python and saved those profiles into CSV files.

METHODOLOGY

3. Development Phase

The development phase was divided into several key tasks, each assigned to specific group members based on their skills and strengths:

- **Backend Development:** The set of members set to develop each aspect of the backend, such as the vehicle management, ride booking, and file handling, worked closely to integrate well together

METHODOLOGY

- **Frontend Development:** The members set on creating and building the GUI are set to translate and implement the code from the backend and apply it seamlessly to buttons and other interactive functions.
- **Geolocation Integration:** Members researching how to integrate our TkinterMapview package and Geopy into the program so we could calculate distance and the fare for booked rides.
- **Quality Assurance (QA) and Testing**: Every member had access to the GitHub repository, so everyone was able to check and debug any possible deficiencies in the program. These deficiencies are communicated through and given a solution as soon as they are found.

METHODOLOGY

4. Implementation and Deployment

During the implementation phase, the group collaborated closely to merge the individual components and ensure they worked together effectively. We utilized GitHub for version control, ensuring that all team members could collaborate without overwriting each other's work. Code reviews were conducted regularly to maintain high-quality standards.

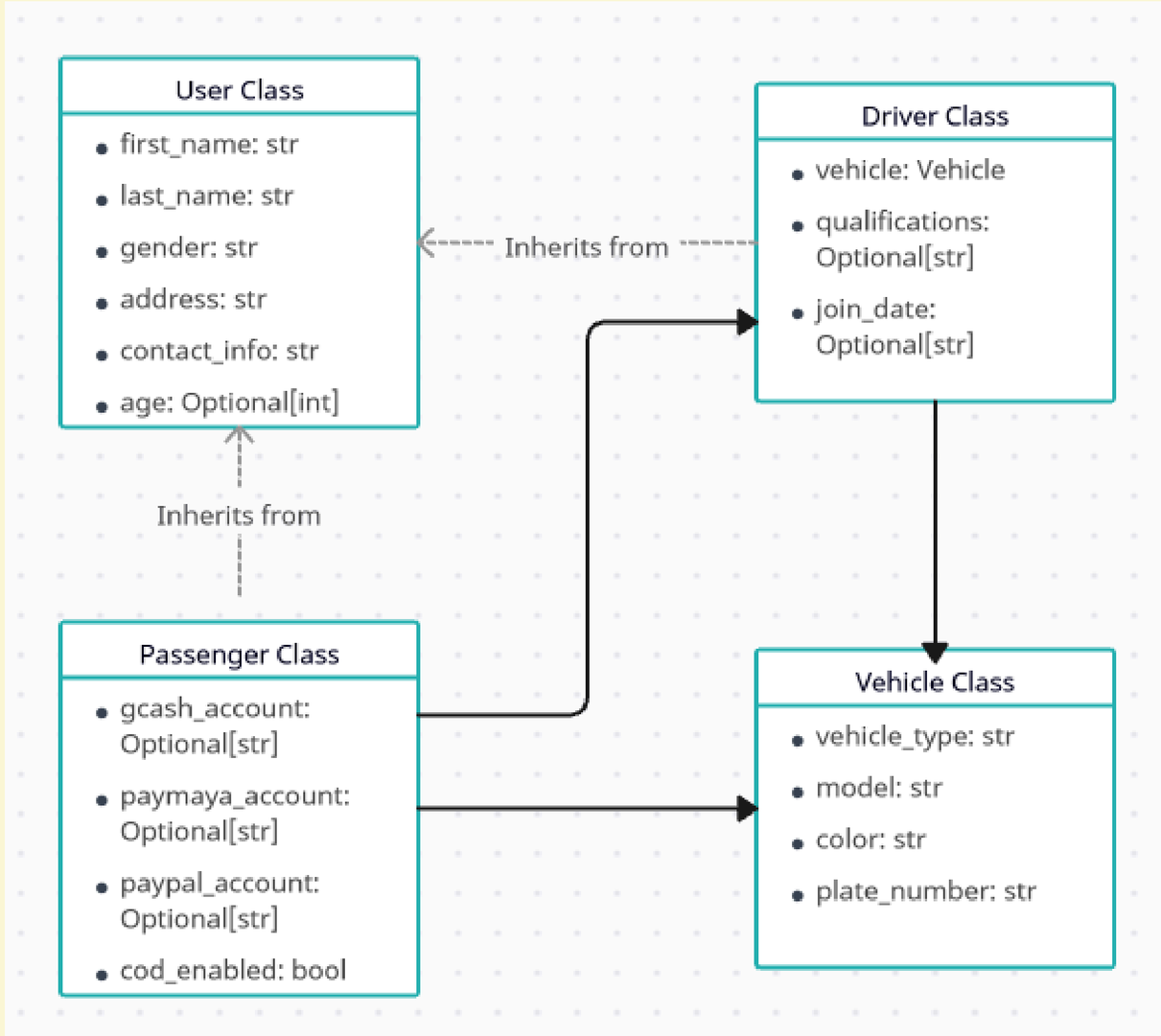
Once the core features were integrated, the team deployed the application for internal testing. The group worked together to fix any bugs and make adjustments based on feedback from the testers.

METHODOLOGY

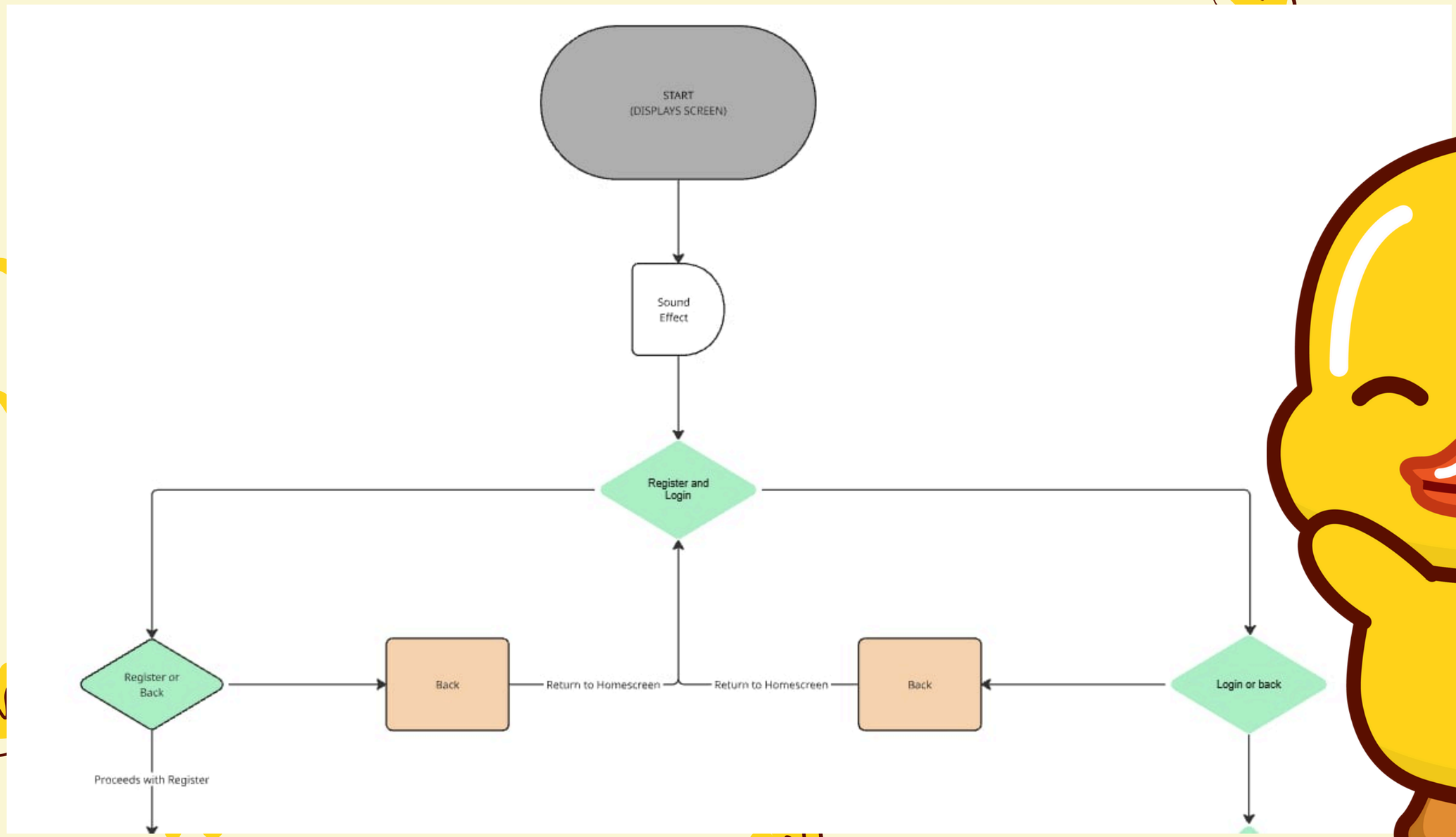
5. Final Review and Documentation

At the final stage, the team came together to review the completed application, ensure all features were functioning as expected, and prepare project documentation. Each member contributed to writing the user manual, system design documentation, and technical reports, which detailed the architecture, database, and codebase.

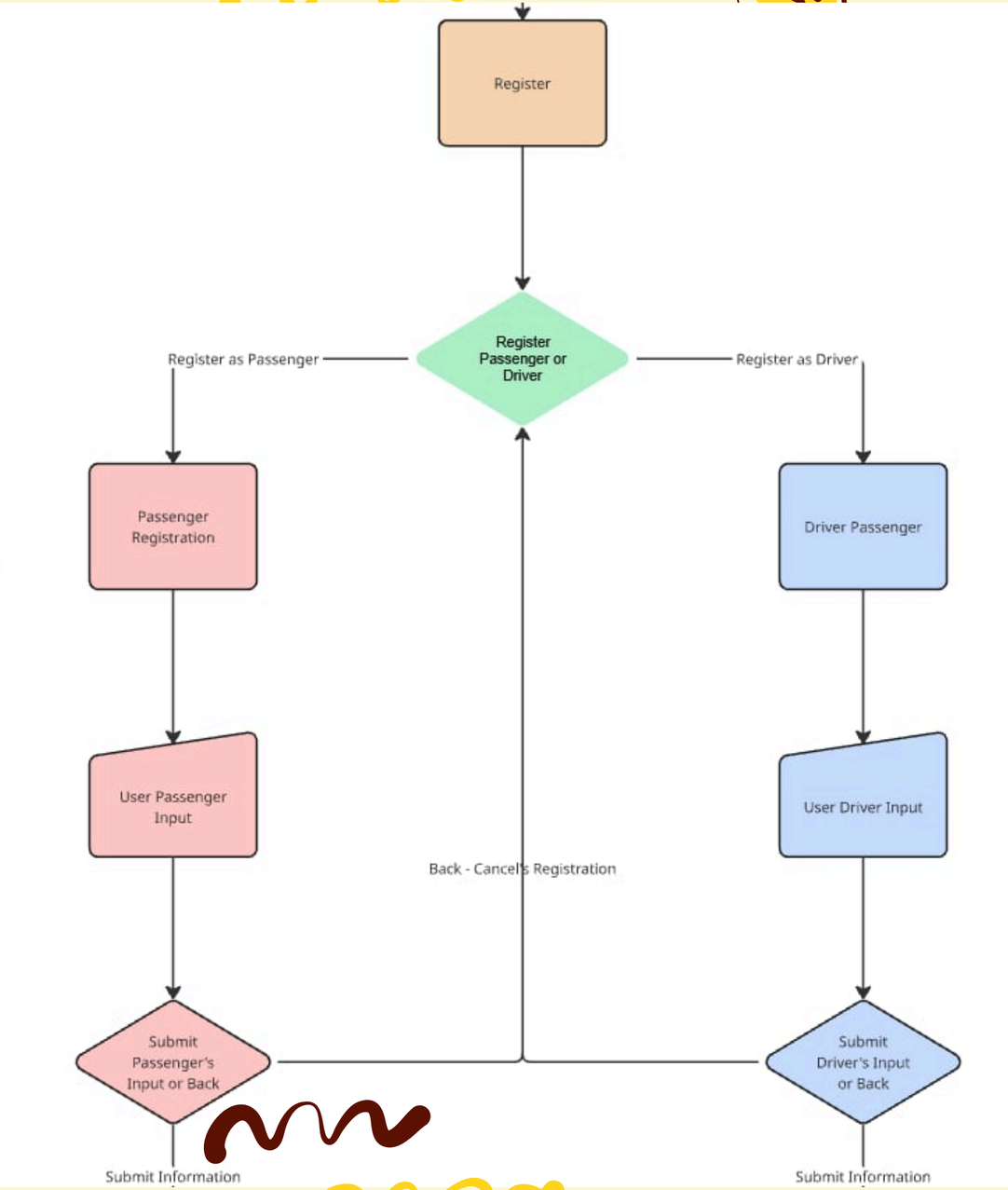
SYSTEM DESIGN: CLASS DIAGRAM

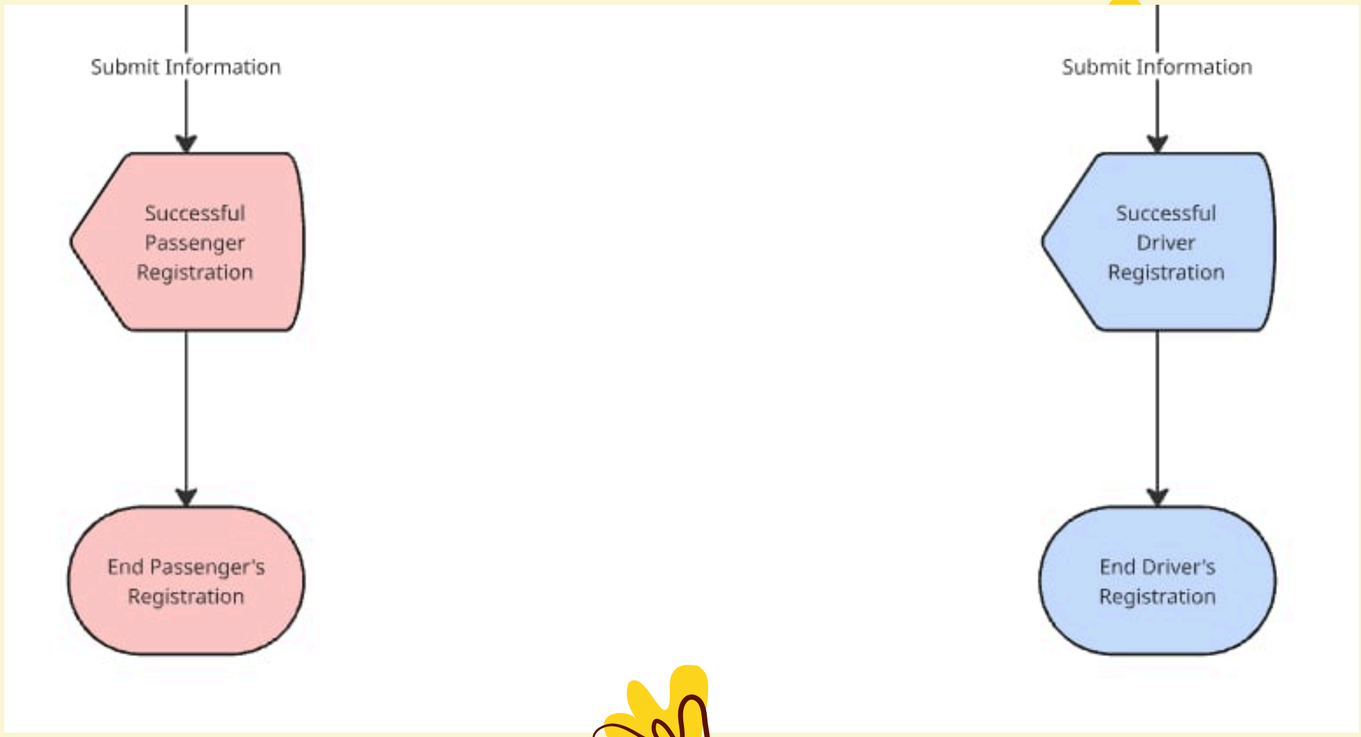


FLOW CHART

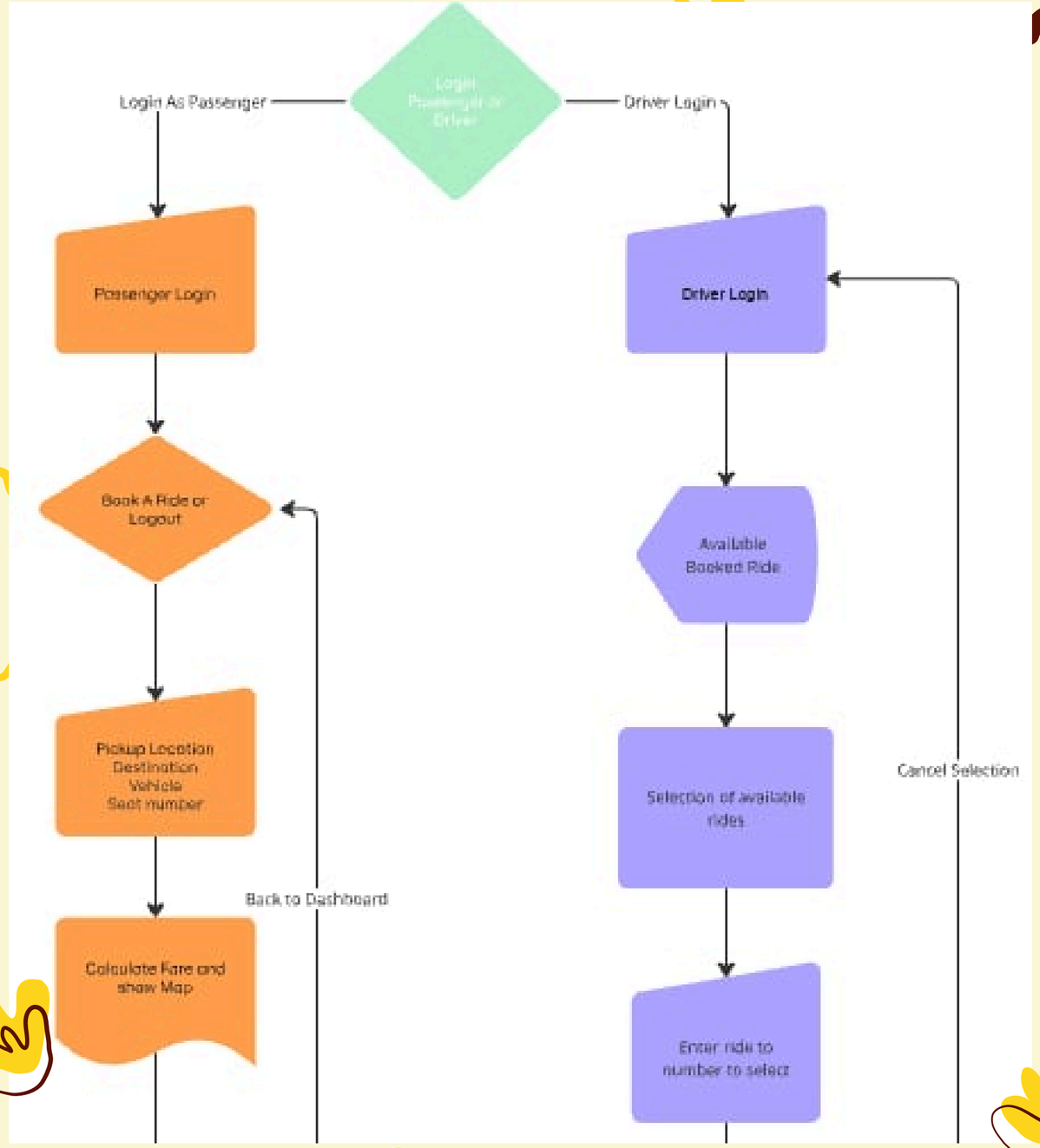

START
(DISPLAYS SCREEN)
Sound
Effect
Register and
Login
Register or
Back
Back
Return to Homescreen
Return to Homescreen
Back
Login or back
Proceeds with Register

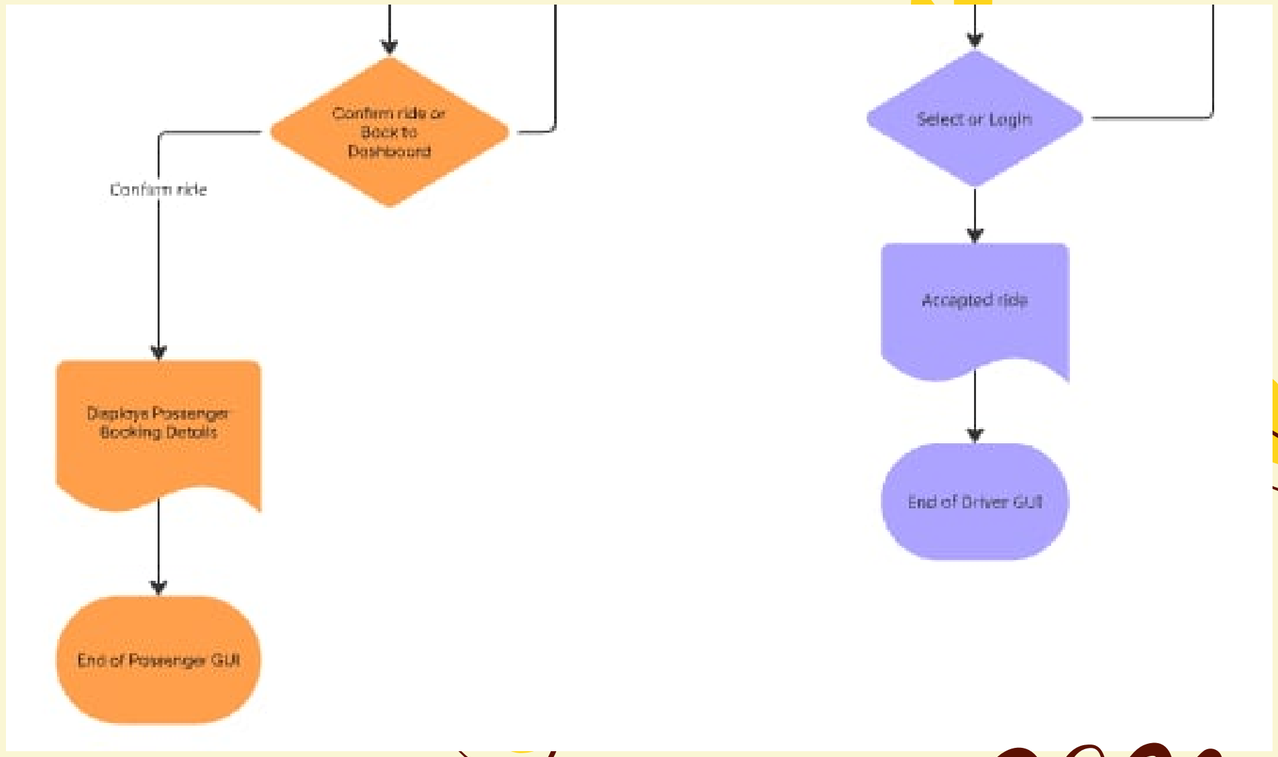
FLOW CHART


Register
Register Passenger or Driver
Register as Passenger
Register as Driver
Passenger Registration
Driver Passenger
User Passenger Input
User Driver Input
Back - Cancels Registration
Submit Passenger's Input or Back
Submit Driver's Input or Back
Submit Information
Submit Information

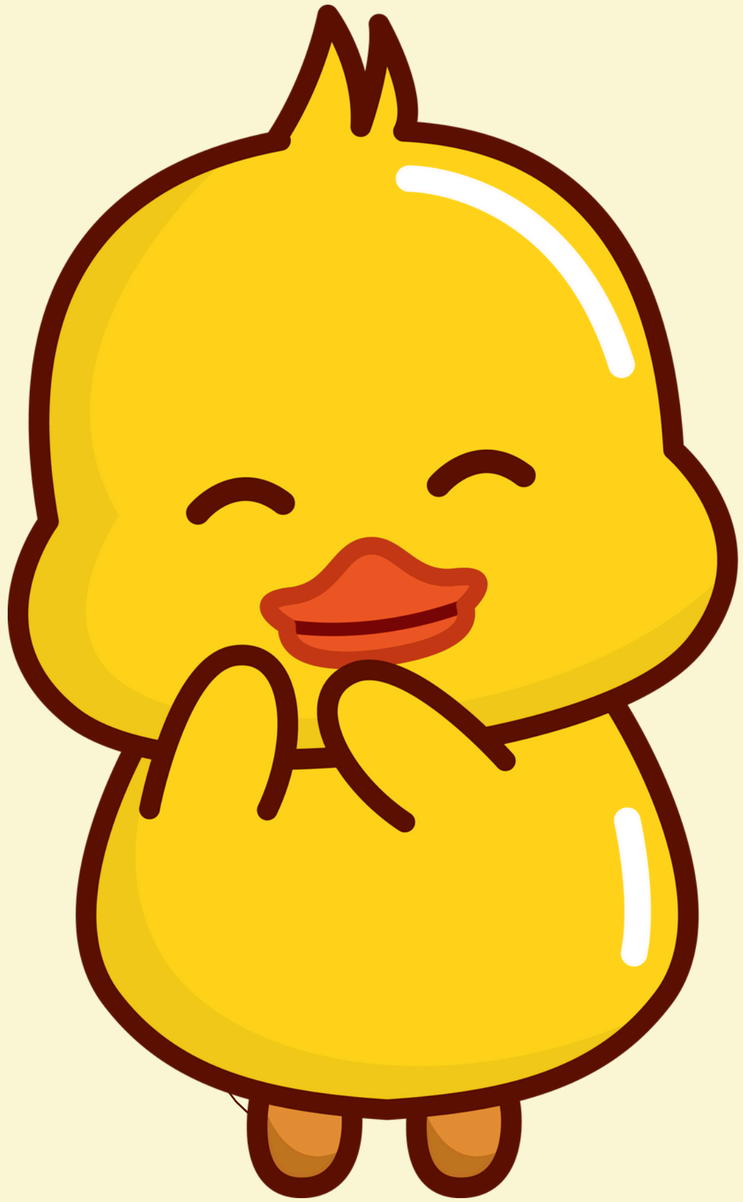

Submit Information
Successful Passenger Registration
End Passenger's Registration
Submit Information
Successful Driver Registration
End Driver's Registration

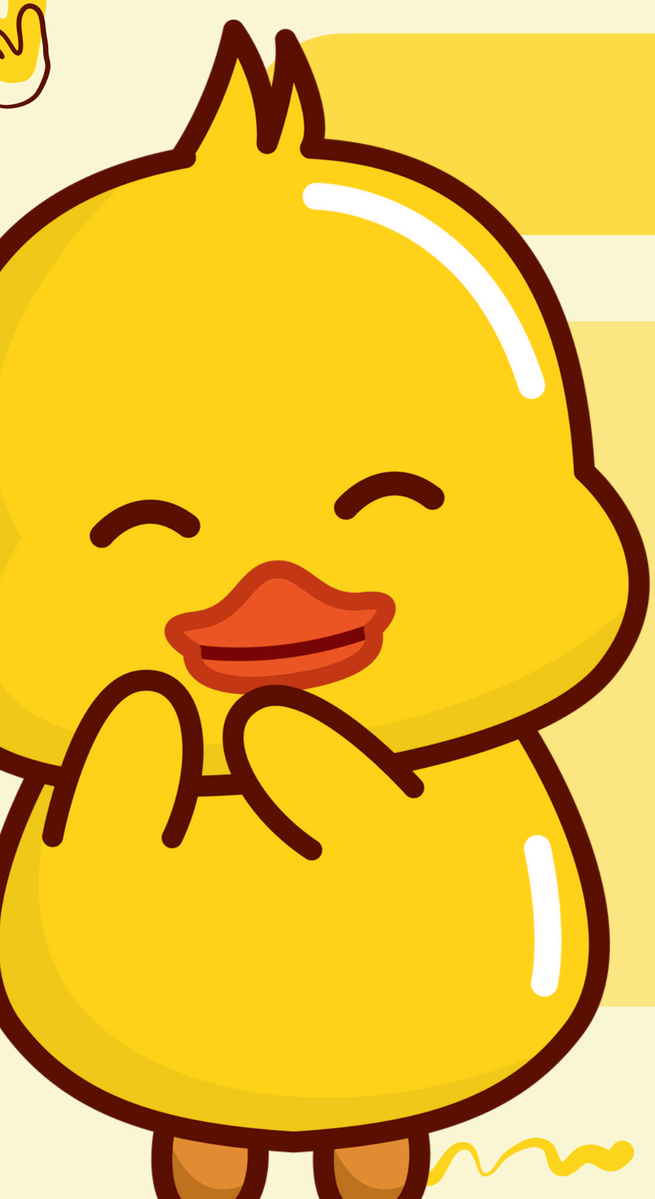
FLOW CHART


Login As Passenger
Driver Login
Passenger Login
Book A Ride or Logout
Pickup Location
Destination
Vehicle
Seat number
Back to Dashboard
Calculate Kare and show Map
Driver Login
Available Booked Ride
Selection of available rides
Cancel Selection
Enter ride to number to select


Confirm ride or Back to Dashboard
Confirm ride
Displays Passenger Booking Details
End of Passenger GUI
Select or Login
Accepted ride
End of Driver GUI

TECHNOLOGY USE





TECHNOLOGIES USE

Duck Dash was developed using Python 3.12.6 as the core language, with Visual Studio Code and GitHub serving as our primary development and collaboration tools. For the user interface and functionality, we used several third-party libraries alongside modules from the Python Standard Library. These technologies allow the app's GUI, map visualization, geolocation, sound playback, and data storage capabilities.

TECHNOLOGIES USE

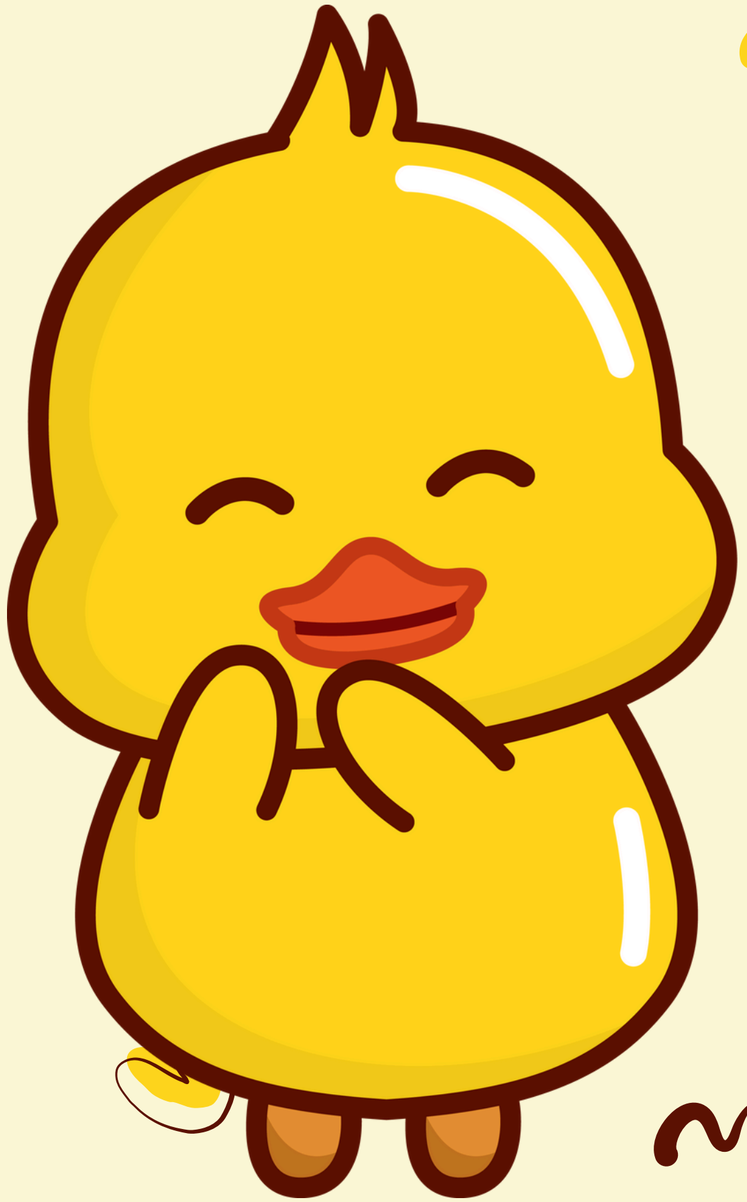
Libraries used that are not in the Standard Python Library:

- geopy
- tkintermapview
- customtkinter
- PIL
- pygame

Libraries used that are in the Standard Python Library:

- tkinter
- datetime
- csv
- os

IMPLEMENTATION

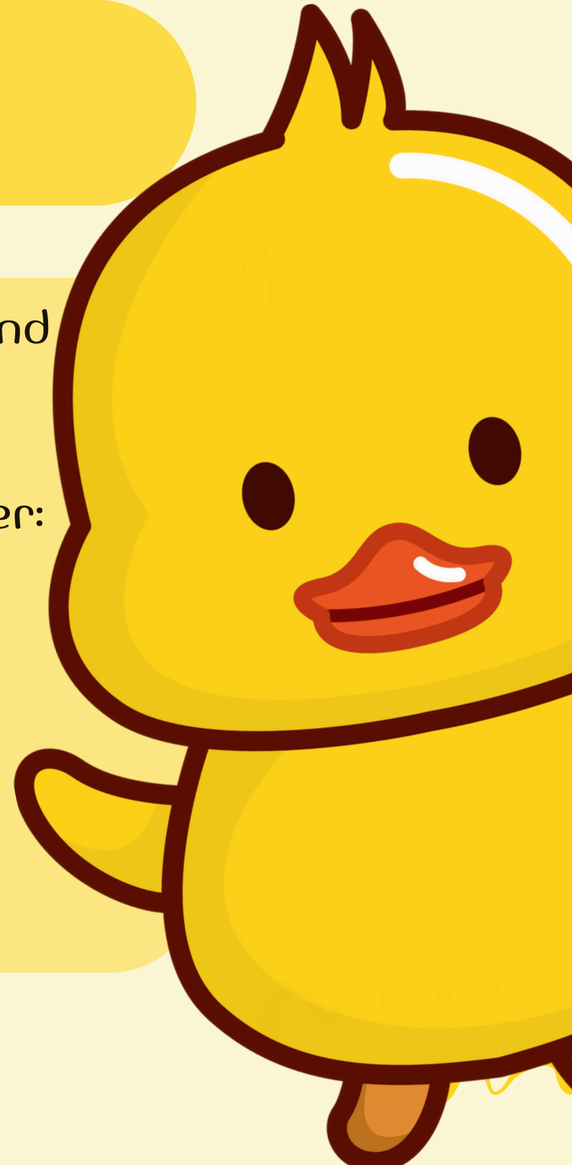


IMPLEMENTATION

Provide details on the modules and features developed. Screenshots and code snippets can be included.

class User

- Used to collect parameters for registering either a passenger or a driver:
  - First Name
  - Last Name
  - Gender
  - Address
  - Contact Information
  - Age



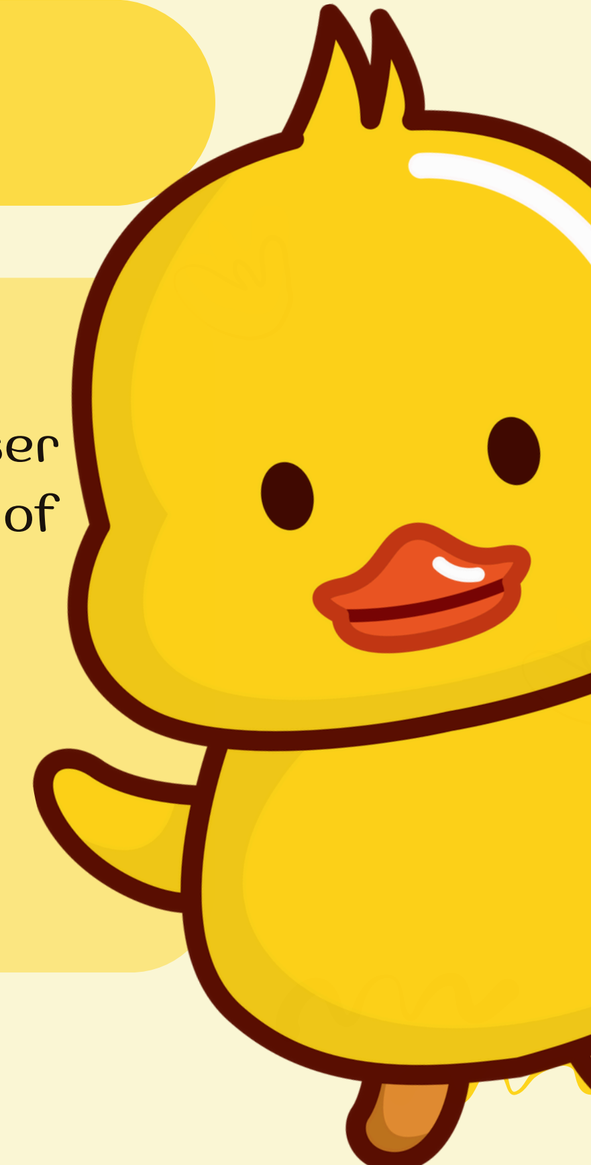
CODE SNIPPET FOR CLASS USER

```
class User:

    def __init__(self, first_name, last_name, gender, address,
contact_info, age=None):

        self.first_name = first_name

        self.last_name = last_name

        self.gender = gender

        self.address = address

        self.contact_info = contact_info

        self.age = age
```

IMPLEMENTATION

class Passenger(User)

- If the user registered as a passenger, it inherits the class User parameters and adds the information about their method of payment
  - Gcash
  - Paymaya
  - Paypal
  - Cash on Delivery



CODE SNIPPET FOR CLASS PASSENGER (USER)

```
class Passenger(User):

    def __init__(self, first_name, last_name, gender, address,
contact_info,

                 gcash_account=None, paymaya_account=None,
paypal_account=None, cod_enabled=True):

        super().__init__(first_name, last_name, gender, address
contact_info)

        self.gcash_account = gcash_account

        self.paymaya_account = paymaya_account

        self.paypal_account = paypal_account

        self.cod_enabled = cod_enabled
```

IMPLEMENTATION

class Driver(Used)

- If the user registered as a driver, it inherits the class User parameters and adds the information as a driver, which is then saved in the drivers.csv file
  - Vehicle
  - Qualifications (Driver's License)
  - Join Date

CODE SNIPPET FOR CLASS DRIVER(USER)

```
class Driver(User):

    def __init__(self, first_name, last_name, gender, address,
contact_info,

                 vehicle, qualifications=None, join_date=None):

        super().__init__(first_name, last_name, gender, address,
contact_info)

        self.vehicle = vehicle

        self.qualifications = qualifications

        self.join_date = join_date or datetime.now().isoformat()
```

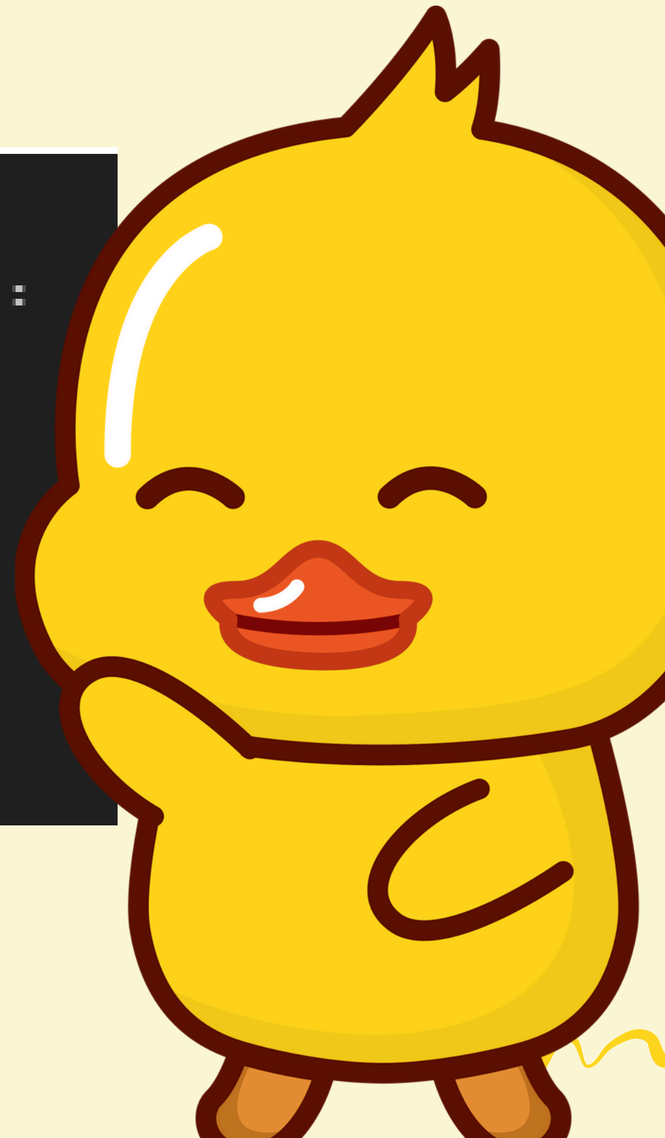
IMPLEMENTATION

class Vehicle

- Part of registration a driver and saved in the same drivers.csv file
  - Vehicle Type
  - Model (Vehicle model)
  - Color (Vehicle color)
  - Plate Number

CODE SNIPPET FOR CLASS VEHICLE

```
class Vehicle:

    def __init__(self, vehicle_type, model, color, plate_number):

        self.vehicle_type = vehicle_type

        self.model = model

        self.color = color

        self.plate_number = plate_number
```



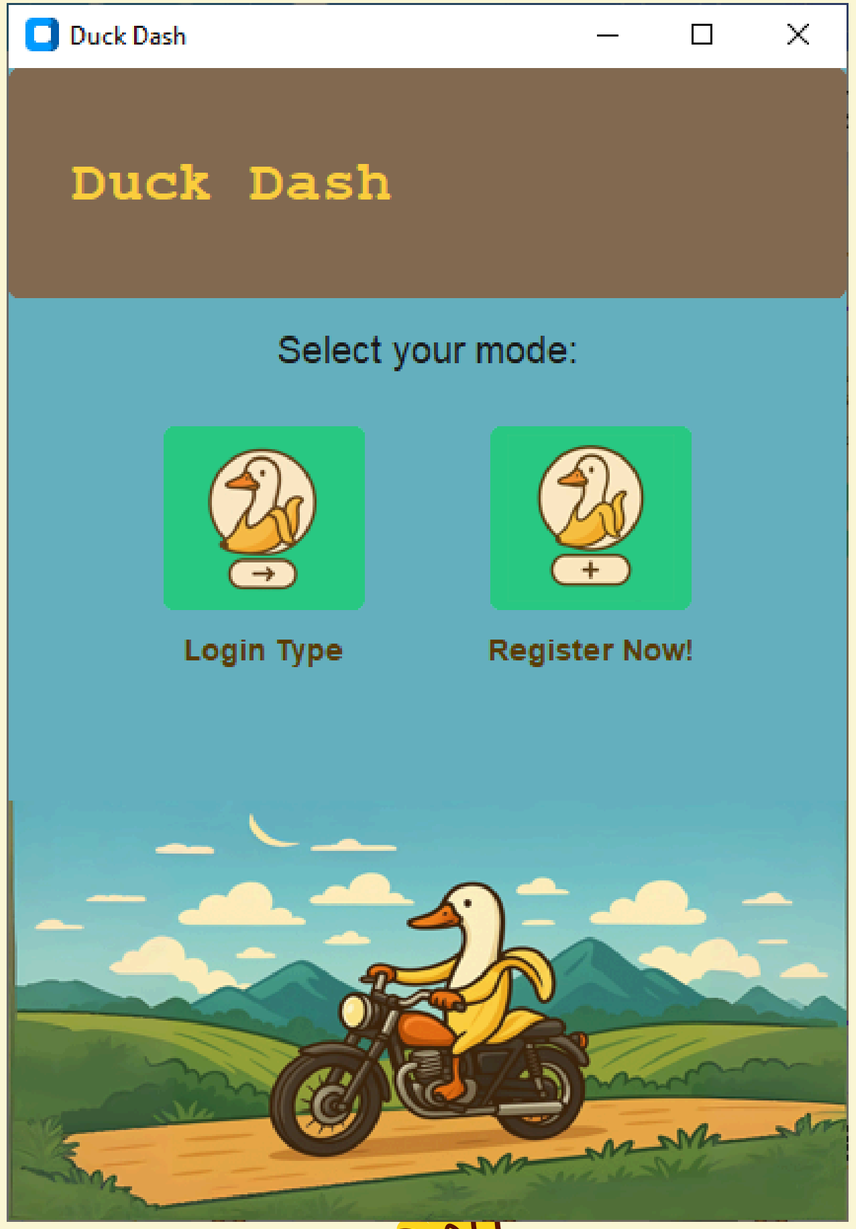
IMPLEMENTATION

class DuckDashApp(ctk.CTk)

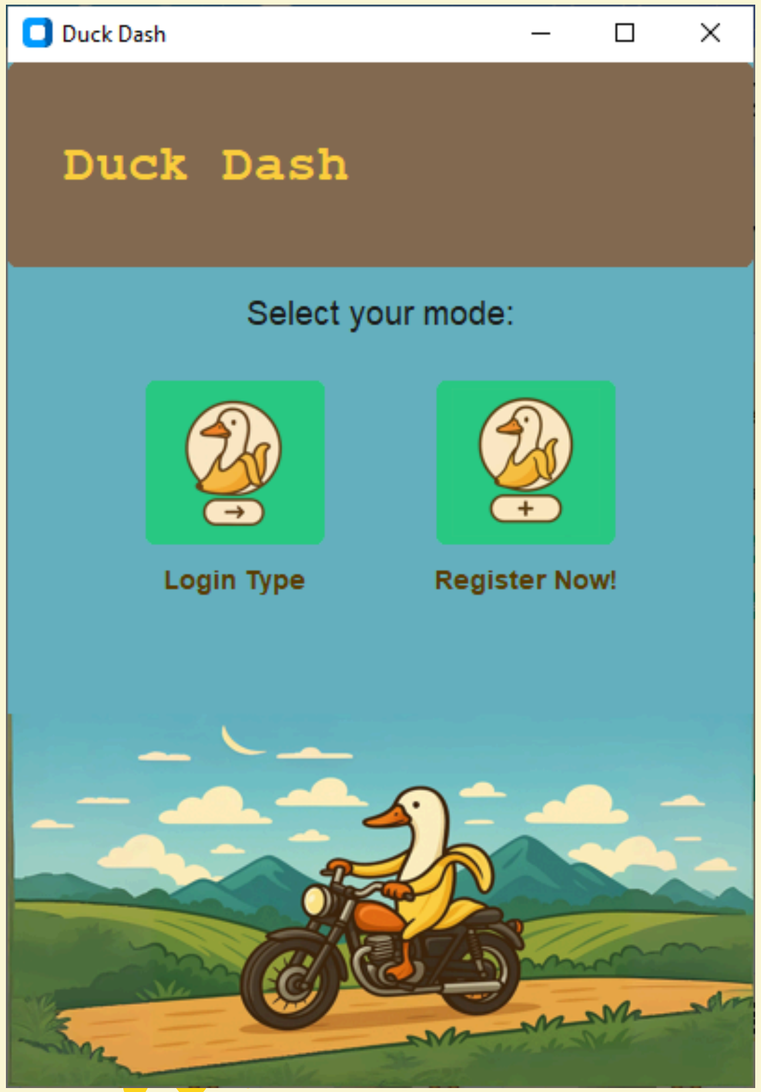
- Main class that will initiate the start of the user interface of Duck Dash
- It will run functions in succession based on what the user has clicked

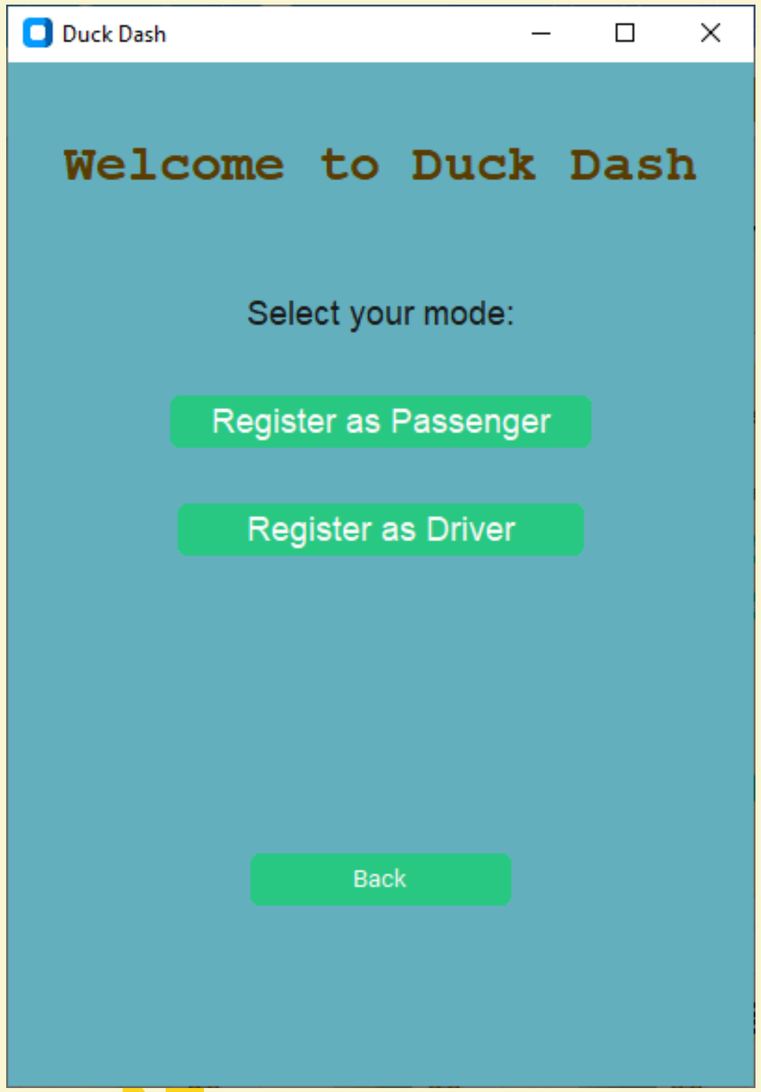
Starting Menu after the load-up animation from the class DuckDashApp instance

STARTING MENU AFTER THE LOAD-UP ANIMATION FROM THE CLASS DUCKDASHAPP INSTANCE

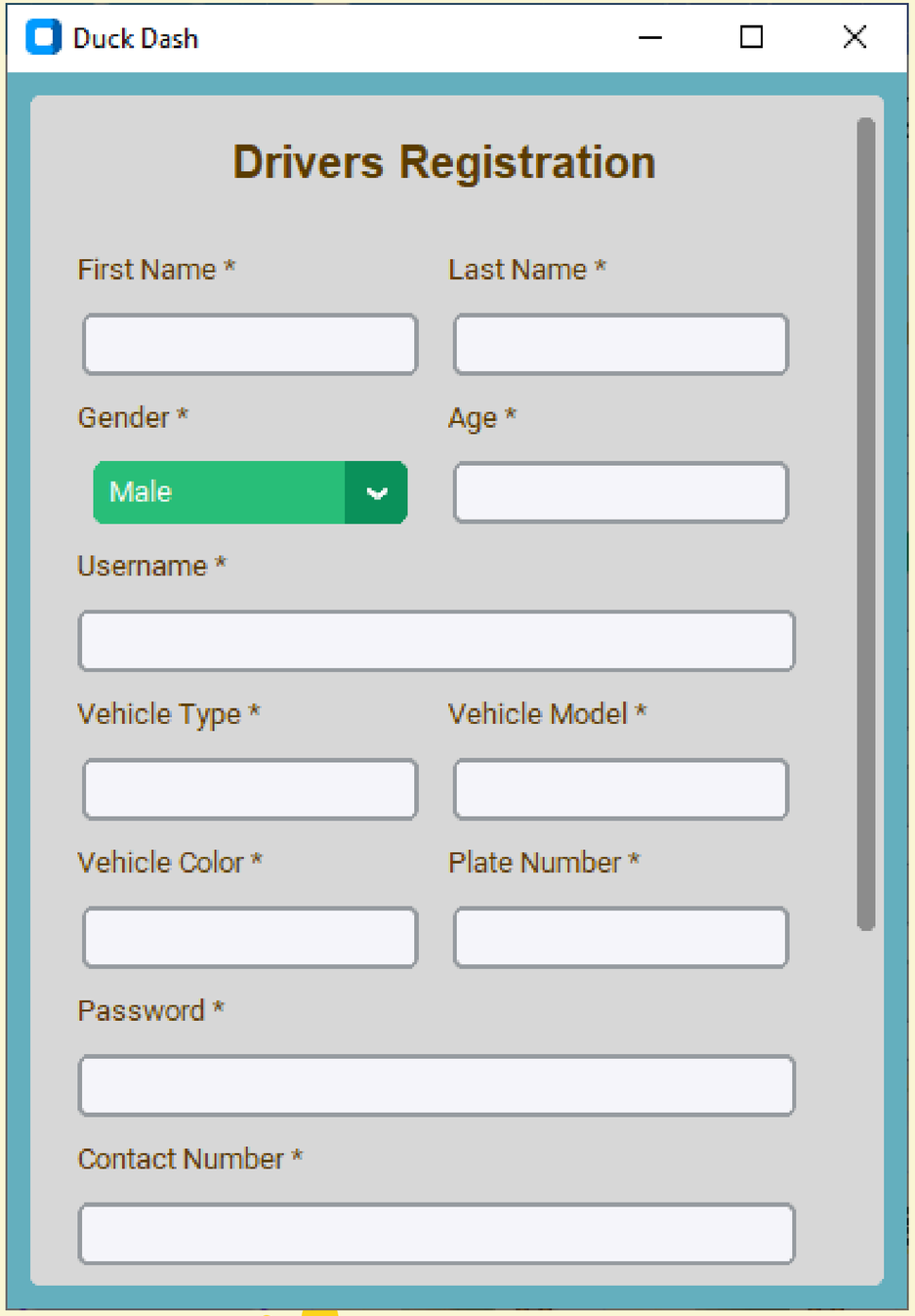


CLICK REGISTER NOW! WILL PROMPT YOU TO CHOOSE TO REGISTER EITHER AS A DRIVER OR A PASSENGER.

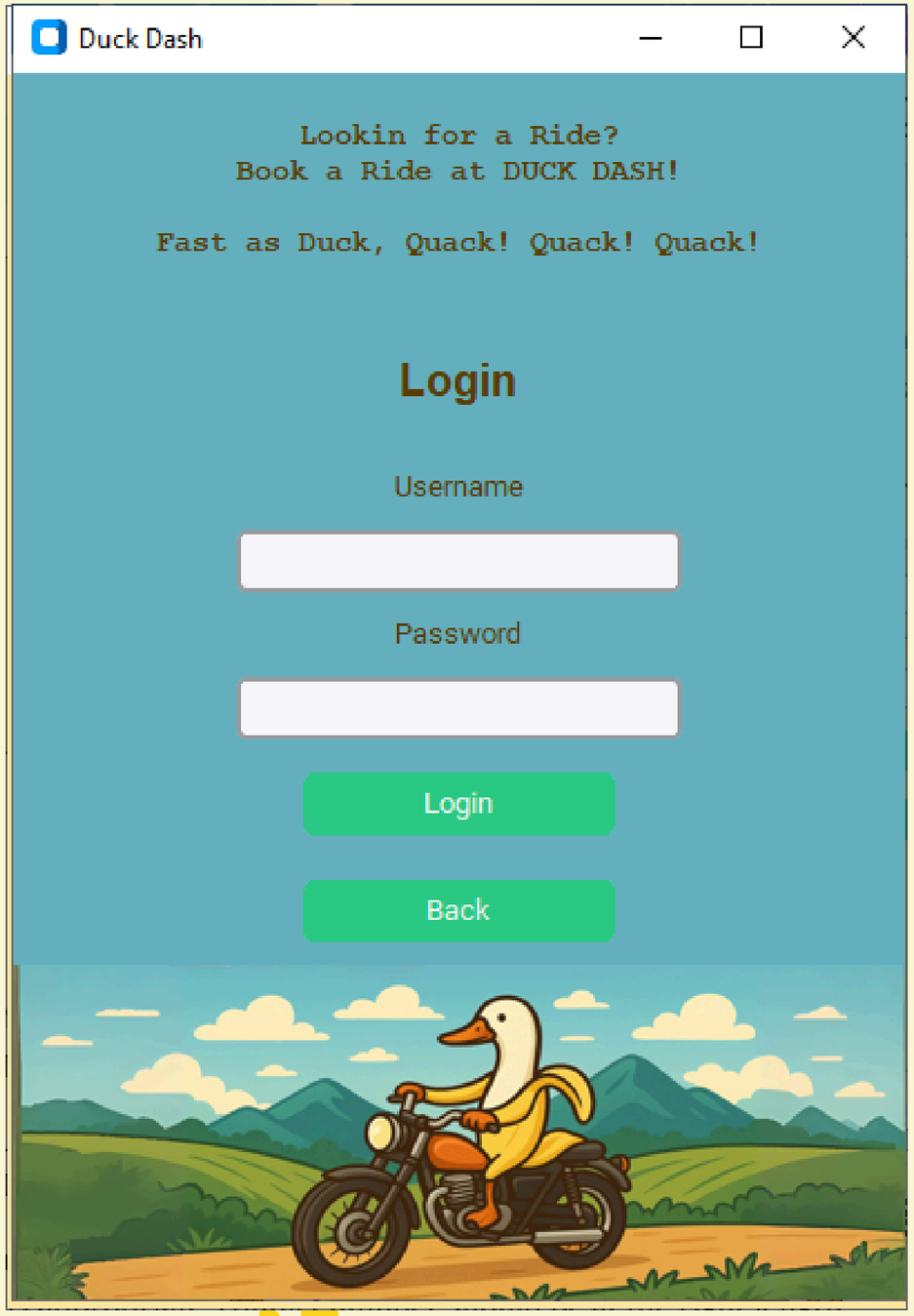




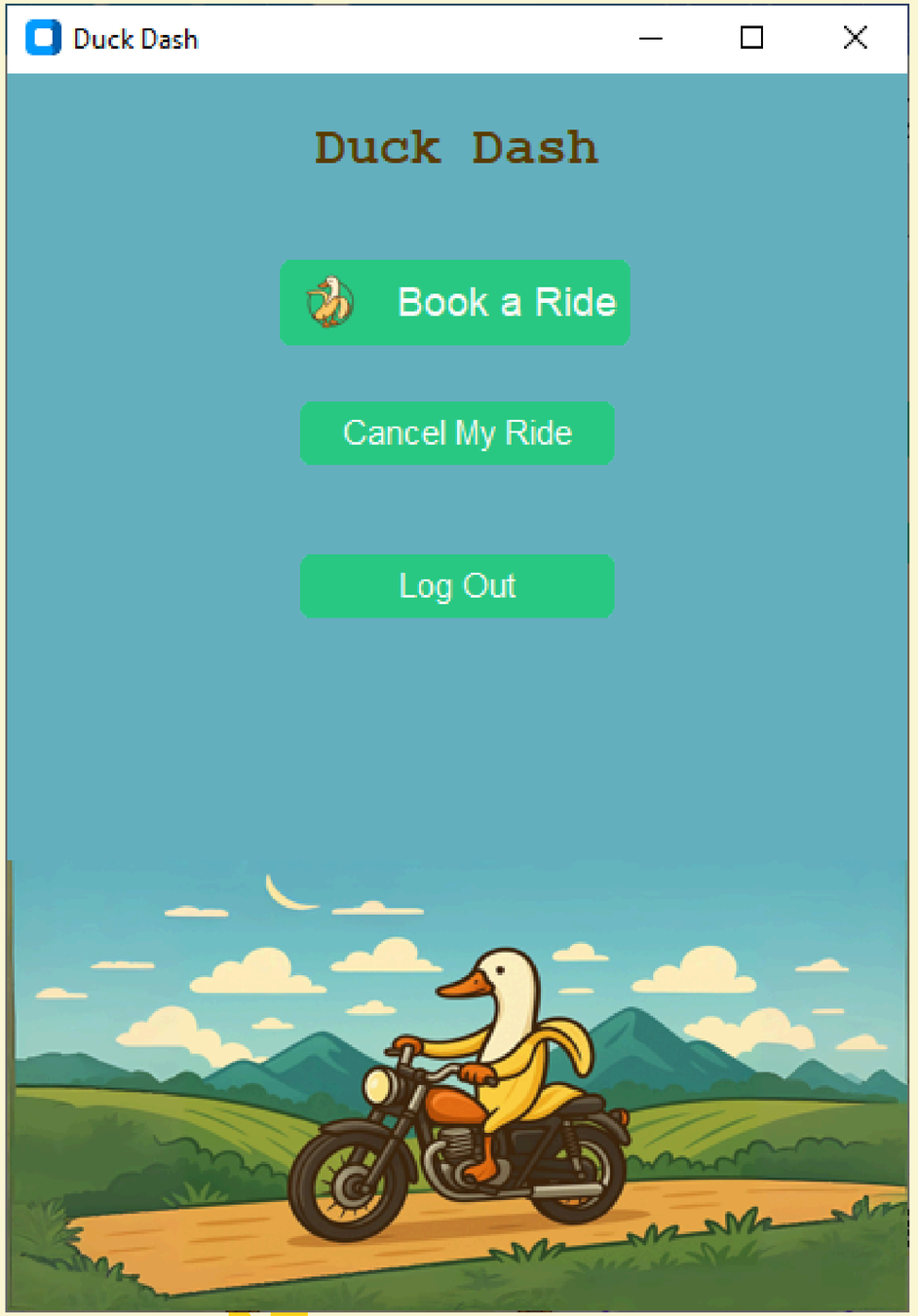
IN THIS EXAMPLE WE'LL REGISTER AS A DRIVER AND IT WILL ASK AS TO FILL UP THE FORMS FOR REGISTRATION WHICH WILL USE THE CLASS DRIVER(USER) TO MAKE AN OBJECT AND BE PUT IN THE DRIVERS.CSV IN THE SAME FOLDER



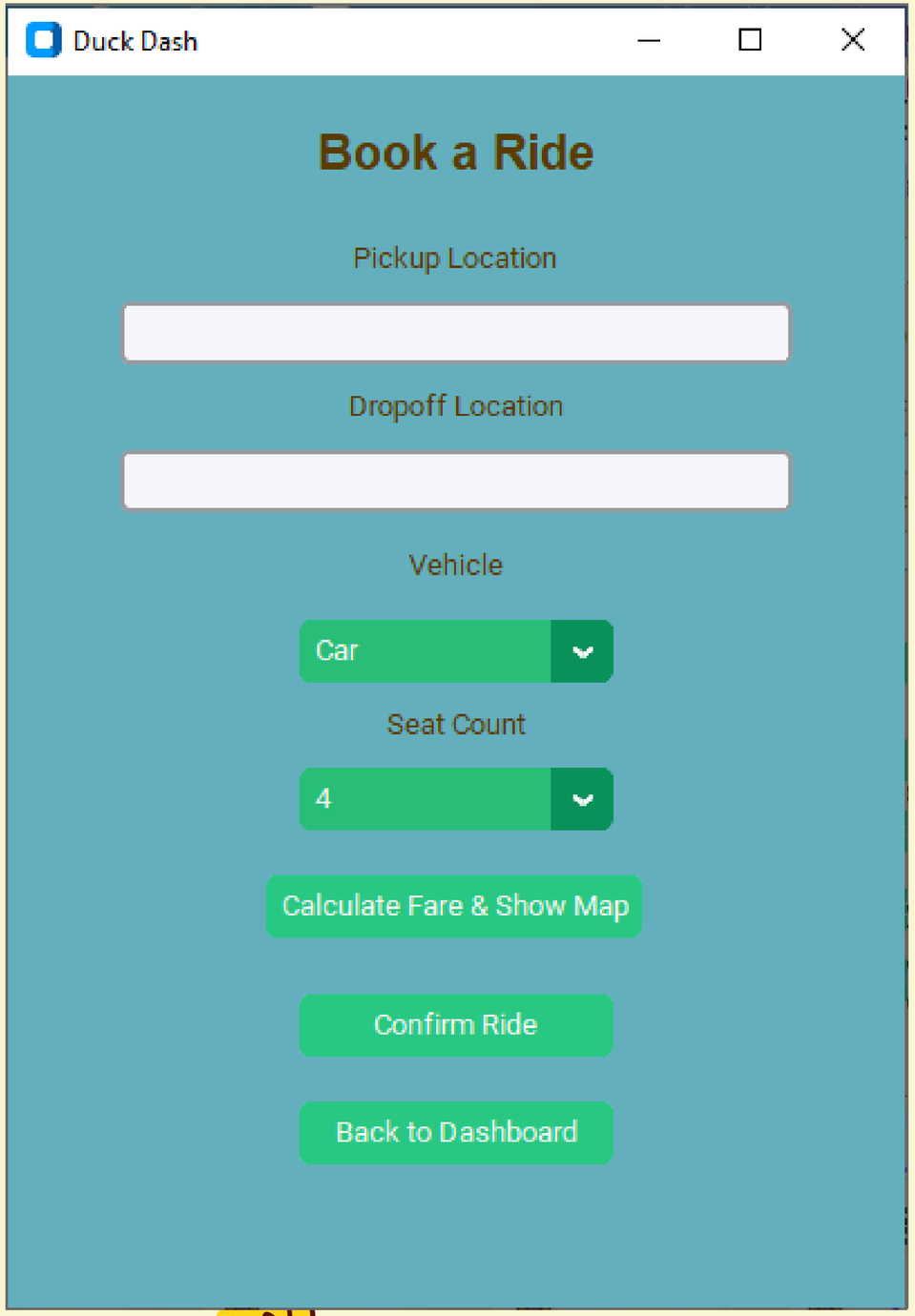
GOING BACK FROM THE START AND NOW CLICKING LOGIN TYPE AND LOGIN AS DRIVER WE WILL BE PROMPTED TO ENTER THE LOGIN CREDENTIALS



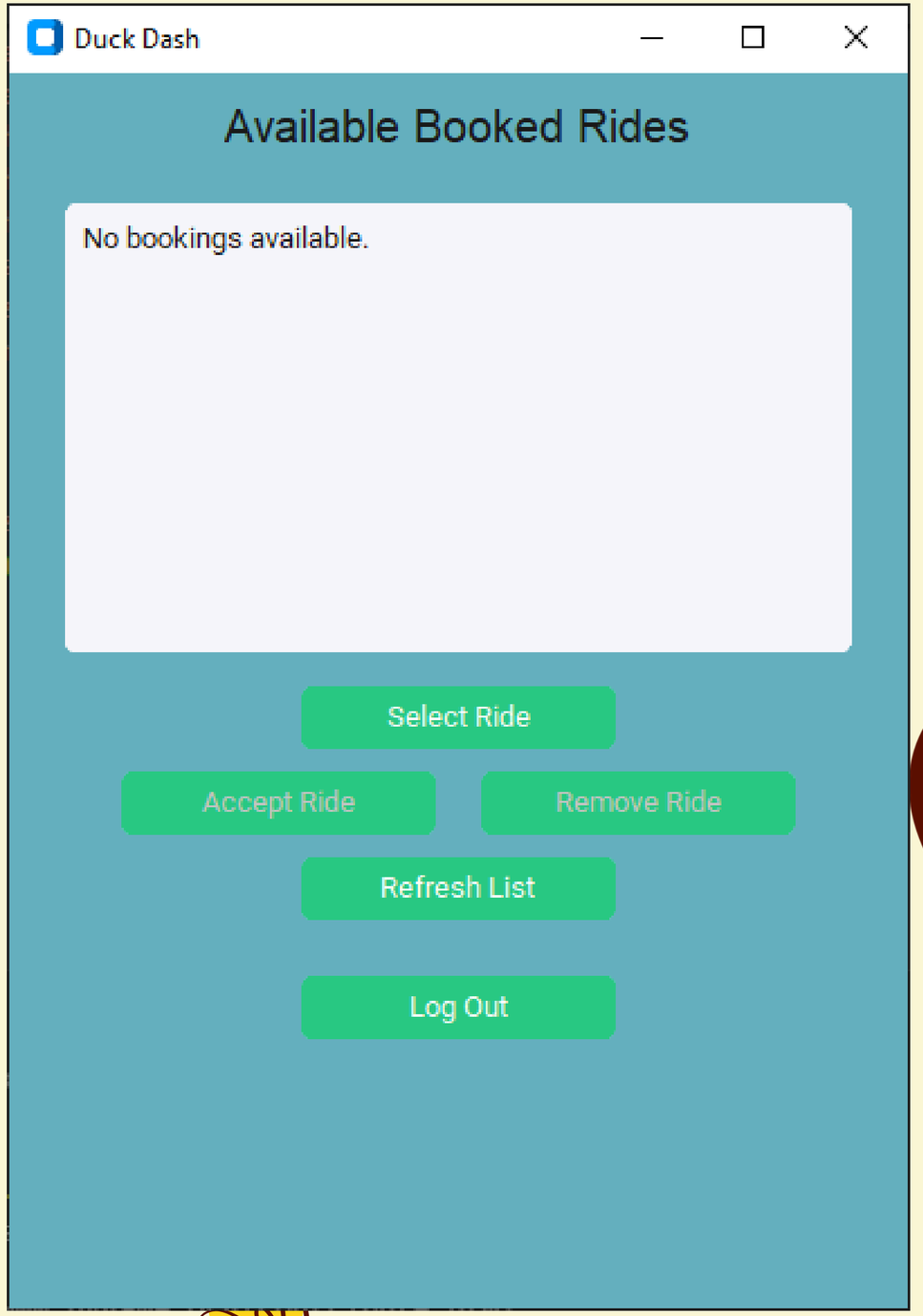
AFTER LOGGING IN WITH USING THE CREDENTIALS THAT YOU'VE PUT IN REGISTERING, YOU CAN NOW USE IT TO BOOK A RIDE



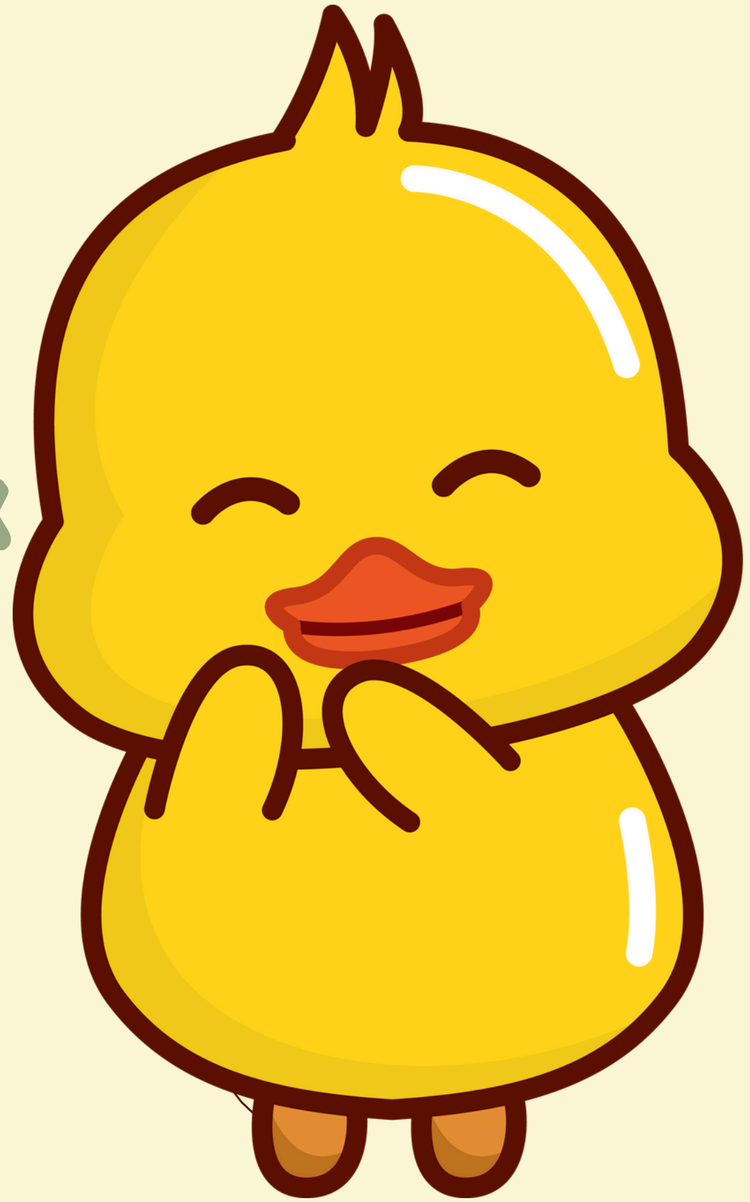
AFTER LOGGING IN WITH USING THE CREDENTIALS THAT YOU'VE PUT IN REGISTERING, YOU CAN NOW USE IT TO BOOK A RIDE.

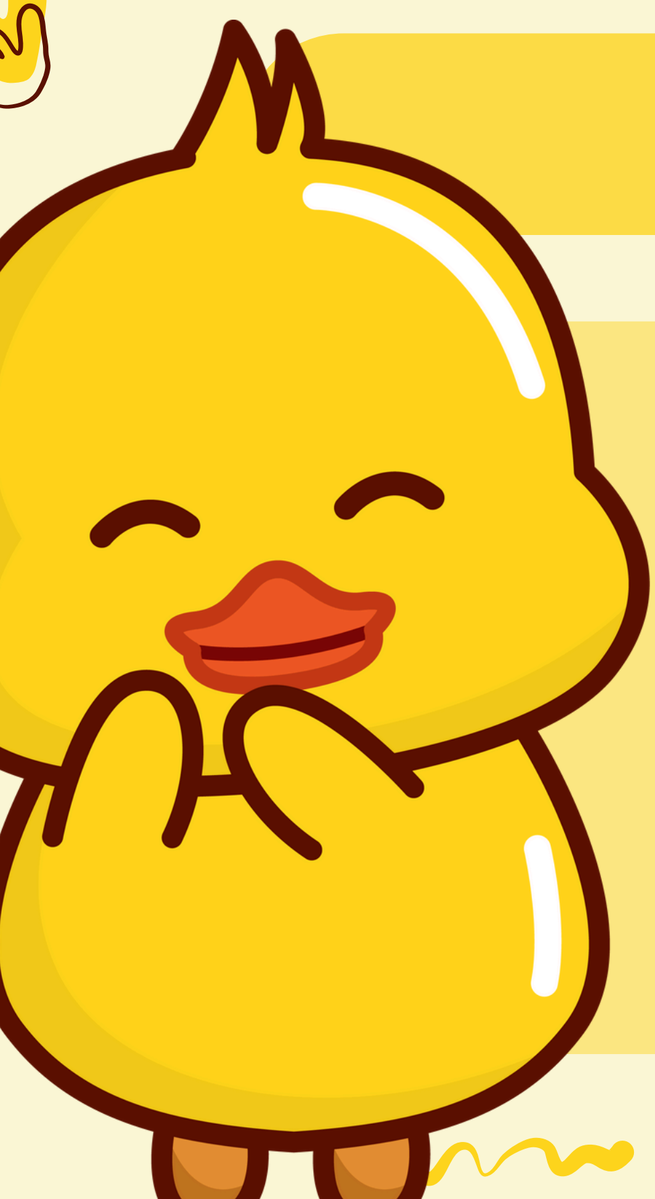


PICKING THE LOGIN AS DRIVER FROM THE OTHER SIDE WILL PROMPT YOU WITH A DIFFERENT SCREEN WHICH IS A DASHBOARD FOR DRIVERS



TESTING & EVALUATION





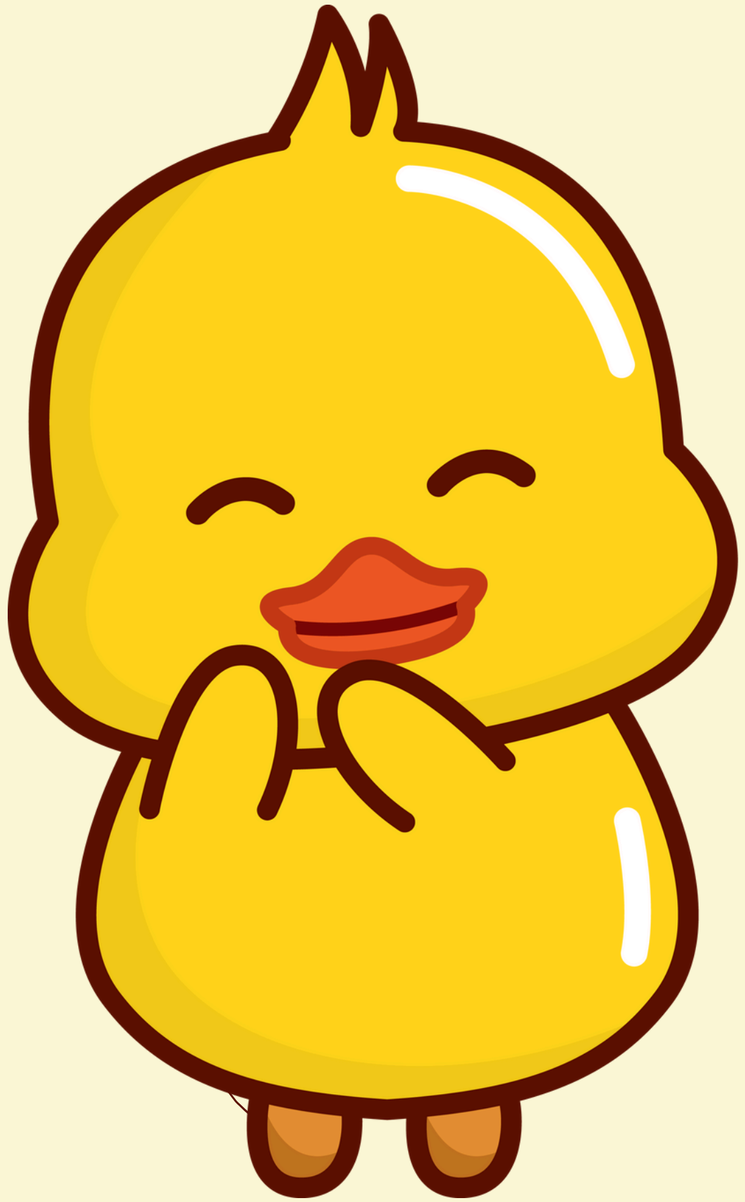
TESTING AND EVALUATION

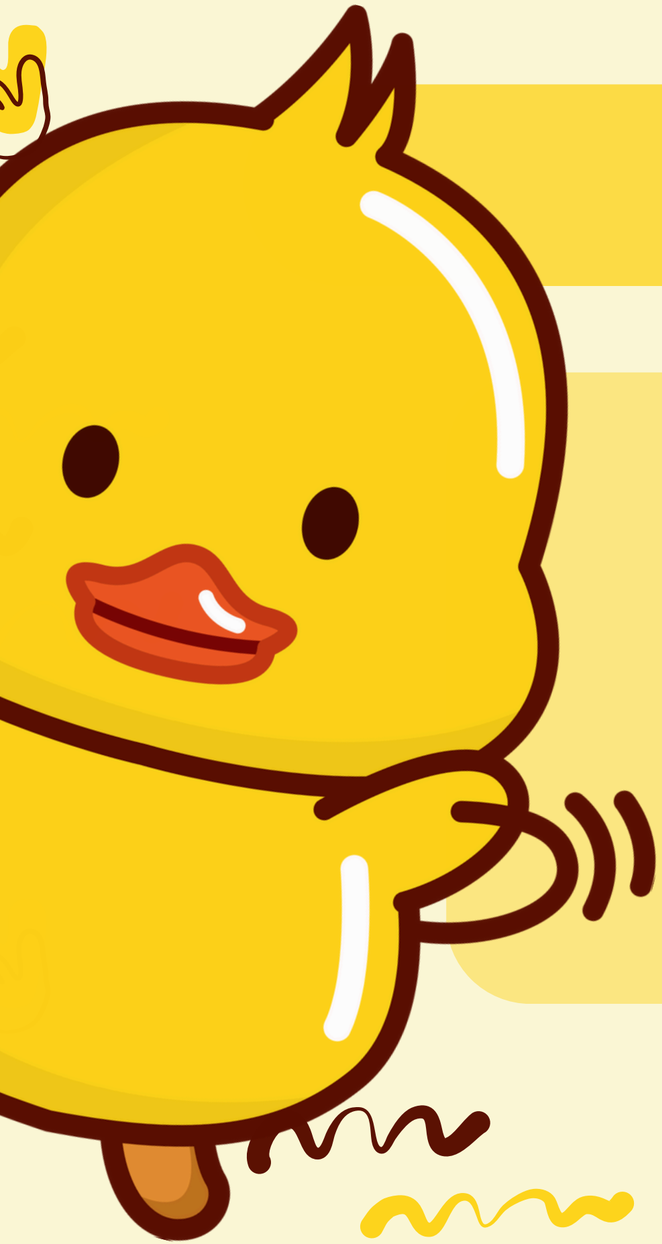
- We have mostly done manual testing as the logic and the possible things that we can do with the program are very small and manageable.
- We also tested common error occurring inputs such as not putting anything in the entry spaces, checking if the age is appropriate, contact number should be at the right length, logging in with the wrong password, and even using the login credentials from the drivers to the passengers and vice versa.

TESTING AND EVALUATION:

Bugs	Description	Fix
Saving data into .csv problems	Not properly saving into their respective csv files	Using from os import * is different from import os since the open() function is being overriden

RESULTS & DISCUSSION





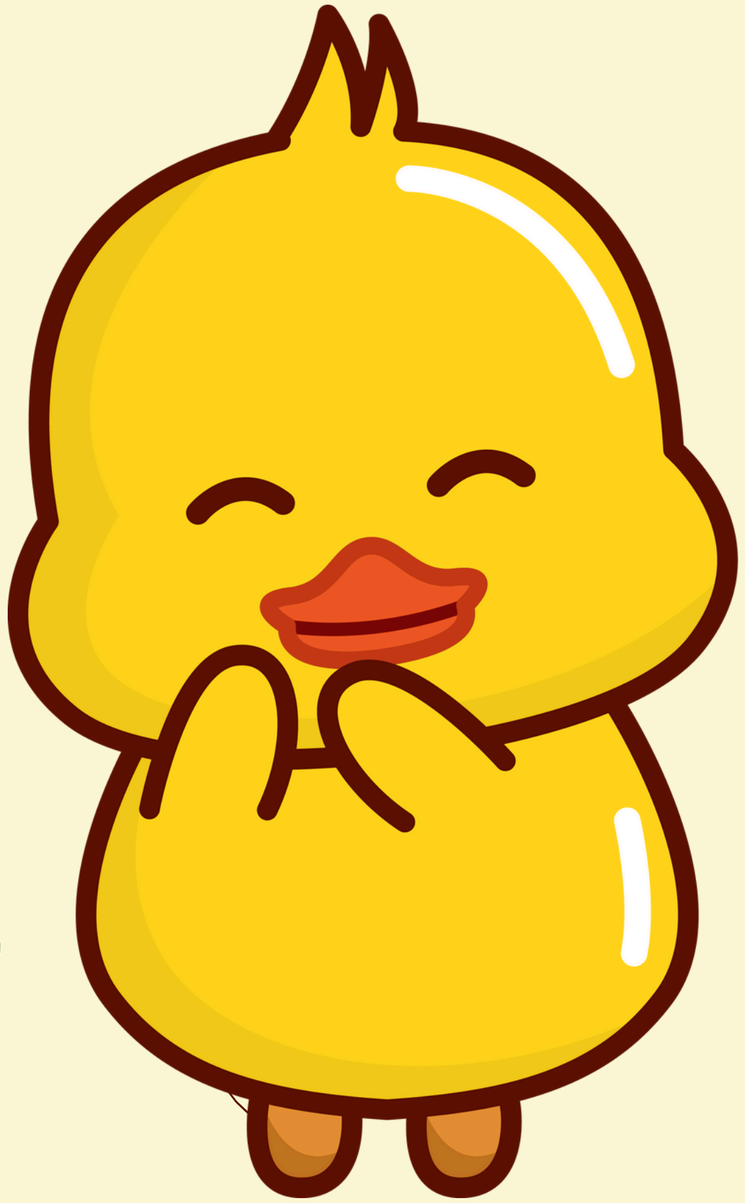
RESULTS & DISCUSSION

The final program meets most of the plans we set out to implement. It manages to create and read profiles for both Passenger and Driver users, with their own separate dashboards. The main purpose of the project, the ability to book a ride, receive its distance and fare and have someone accept and take in a passenger works well, through its simple but effective database system.

RESULTS & DISCUSSION

Though the product could work better with a robust GPS implementation, that feature is just beyond our means at the moment, the distance calculation could also use improvement as it calculates distance in a straight line and doesn't take into account any street routes.
All in all the program works well, and is a small glimpse of the potential of what this program could be with further development.

CONCLUSION & FUTURE WORK



CONCLUSION

During the program, as a group we value our teamwork and cooperation all throughout, this project has brought us to learn about how everyone approaches each of their responsibilities and their strengths and weaknesses in programming. By the end we managed to end with a product that matches our wants while still finding fun in the process. Though for all of us, there will still be things to improve in many aspects like planning, implementing and cooperating, this project will be a learning experience and we'll use it to create an even greater product that will please more.

FUTURE WORK

Additional features to include

- Mobile Port (Android and iOS)
- A robust GPS system
- A stronger security system
- Distance and route calculations that take into account routes, roads blockages, and traffic
- Multi-lingual support

THANK YOU FOR LISTENING!

MEMBERS:

Group 6:
Baltazar, Lord Van Suva
Boncales, Kirsten Caryl
De Jesus, Maria Teresa Mae
Gabriel, Matthewe
Gozon, Ezekiel Zio
Malabuyoc, Ron Andre
Soon, Jd Angelo

BSCPE 1-7