

**CIS020-1 – Introduction to Software Development**

**CIS093-1 – Mathematics and Concepts for Computational Thinking**

**Assignment 2 – Group / Individual Project –CaseStudy**

**(Taxi Booking System)**

**Student Name: Baibhav Paudel - 2146504X - University ID: 012345**

**Student Name: XXXX XXXXX - University ID: 012345**

**OR**

**Student Name: XXXX XXXXX - University ID: 012345**

**Table of Contents**

[Introduction/Overview 3](#__RefHeading___Toc42041_3358107187)

[Task Description 3](#__RefHeading___Toc42043_3358107187)

[Assumptions about the system : 4](#__RefHeading___Toc42045_3358107187)

[Project Plan/Schedule 4](#__RefHeading___Toc42047_3358107187)

[Requirements Analysis 5](#__RefHeading___Toc42153_3358107187)

[Functional Requirements 5](#__RefHeading___Toc42155_3358107187)

[Non-functional Requirements 5](#__RefHeading___Toc42261_3358107187)

[Usability Requirements 6](#__RefHeading___Toc42263_3358107187)

[Design 6](#__RefHeading___Toc42265_3358107187)

[UML Diagrams 6](#__RefHeading___Toc42267_3358107187)

[Uses Case Diagrams(s) 7](#__RefHeading___Toc42269_3358107187)

[Activity Diagram(s) 8](#__RefHeading___Toc42461_3358107187)

[9](#__RefHeading___Toc43107_1759767037)

[Class Diagram(s) 11](#__RefHeading___Toc42463_3358107187)

[Database Design 13](#__RefHeading___Toc42475_3358107187)

[13](#__RefHeading___Toc43109_1759767037)

[Logical Database Design 14](#__RefHeading___Toc42477_3358107187)

[Physical Database Design 14](#__RefHeading___Toc42485_3358107187)

[Skeleton Tables (with Primary Keys and Foreign Keys) 14](#__RefHeading___Toc42487_3358107187)

[Data Dictionary 15](#__RefHeading___Toc1517_2908959286)

[User Interface Design 18](#__RefHeading___Toc42491_3358107187)

[Algorithm Design 23](#__RefHeading___Toc42493_3358107187)

[Implementation 26](#__RefHeading___Toc1519_2908959286)

[Testing 42](#__RefHeading___Toc43111_1759767037)

[Test No 42](#__RefHeading___Toc42495_3358107187)

[Test Date 42](#__RefHeading___Toc42497_3358107187)

[Purpose of test 42](#__RefHeading___Toc42499_3358107187)

[Input data or action 42](#__RefHeading___Toc42501_3358107187)

[Expected result 42](#__RefHeading___Toc42503_3358107187)

[Actual result 42](#__RefHeading___Toc42505_3358107187)

[Action if the above are not the same 42](#__RefHeading___Toc42507_3358107187)

[Screenshot of output 42](#__RefHeading___Toc42509_3358107187)

[Discussion / Reflection / Critical Analysis 42](#__RefHeading___Toc43137_1759767037)

[Conclusion 43](#__RefHeading___Toc43139_1759767037)

[References 43](#__RefHeading___Toc43141_1759767037)

[Appendix 43](#__RefHeading___Toc43143_1759767037)

CIS020-1 – Introduction to Software Development - 2021-2022

CIS093-1 – Mathematics and Concepts for Computational Thinking – 2021-2022  
Assignment 2 – Individual Project – Case Study (Taxi Booking System)

# Introduction/Overview

The Taxi Booking System project, together with the project brief and requirements to execute this assignment, was the final task handed to us. Our primary objective in order to finish Assignment 2 was to create a system for booking taxis. In order to book a cab, a user must first register and sign in using login information (username and password), hence a user registration system and a user login system were created. There must be a way for the system user to reserve a taxi for their journey, thus a booking system was developed. A mechanism for admin control was created to allocate cabs to user booking requests. A driver dashboard was established where he or she could monitor assigned bookings, mark them as completed when the journey was finished, and view forthcoming bookings. A database was made and the system was connected to it using a database connector in order to keep all of the user information and details in the system.

# Task Description

The main aim was to create software for a taxi company so that customers could schedule cabs and view their upcoming journeys online using a desktop application's graphical user interface. All the important data regarding user, booking, taxi, and the driver had to be kept in an external database system.

The user must first register as a user with accurate information in order to make a cab reservation. After registering, the user must be able to input the location, date, and time of their pickup as well as the destination of their journey throughout the booking process.

The system must have a capability that allows users to update or cancel their reservations and view forthcoming trips through a graphical user interface. Admin from the business side must

be allowed to approve the user's booking once they submit it and choose a taxi driver to complete the booking. Each driver must have a single forthcoming trip.

## Assumptions about the system :

* The cost of a journey is determined during the booking process based on the distance between the pickup place and the destination.
* Credit card payments will be added in the driver's account balance.
* The driver is paid immediately once the journey is finished.
* Even after cabs have been assigned, reservations can still be canceled, but only with the admin's permission if an issue occurs.
* Taxi assignments cannot be changed.
* The driver keeps his or her earnings throughout the month, but at the end of the month, he or she is required to pay the corporate service fee which has already been computed and is shown to him in his driver dashboard in order to continue using the system the following month.

# Project Plan/Schedule

|  |  |  |
| --- | --- | --- |
| Week No. | Tasks | Priority |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | Submit Group Report, Project Code and Video Recording (if required) |  |
|  | Project Presentation (if required) |  |

# Requirements Analysis

## Functional Requirements

TTBS = Turbo Taxi Booking System

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | A signup system allows customers to register. | MUST |
| 2 | To access the system, users must log in. | MUST |
| 3 | Request booking from booking form. | MUST |
| 4 | User and reservation information is saved to a separate file. | MUST |
| 5 | An administrator must be able to add a cab to a user's reservation. | MUST |
| 6 | All booking requests will be visible to the administrator. | MUST |
| 7 | Bookings must be able to be canceled by admin. | MUST |
| 8 | The user must be able to see information about forthcoming bookings. | MUST |
| 9 | A booking must be able to be cancelled by the user. | MUST |
| 10 | Booking specifics must to be editable by the user. | SHOULD |
| 11 | he ability to log into the system is required for cab drivers. | MUST |
| 12 | An forthcoming journey must be visible to taxi drivers. | MUST |
| 13 | It should be possible for taxi drivers to see their earnings. | SHOULD |

## Non-functional Requirements

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | The system's processing speed should be high. | MUST |
| 2 | Must be screen-adaptive for devices. | SHOULD |
| 3 | System upgrades must be possible. | MUST |
| 4 | must be free of crashes or run-time problems. | MUST |

## Usability Requirements

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | System GUI must be user friendly. | MUST |
| 2 | All forms for entering data should be brief, simple to complete, and include entry validation. | MUST |

# Design

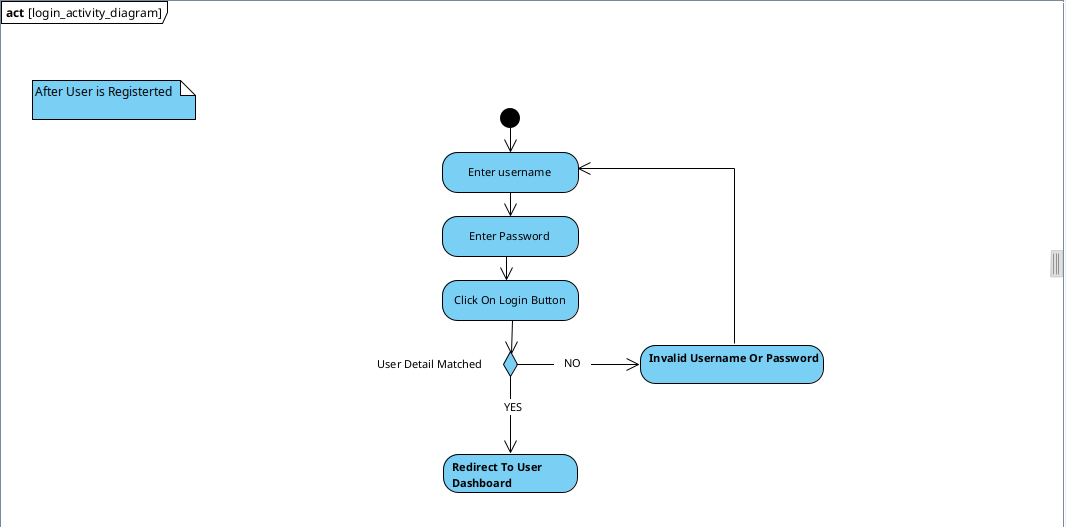
## UML Diagrams

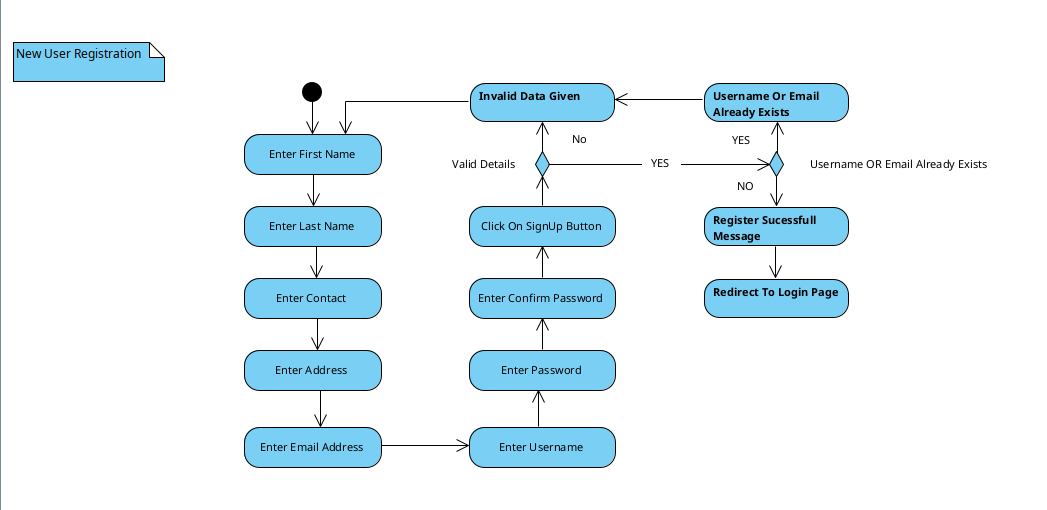
### Uses Case Diagrams(s)

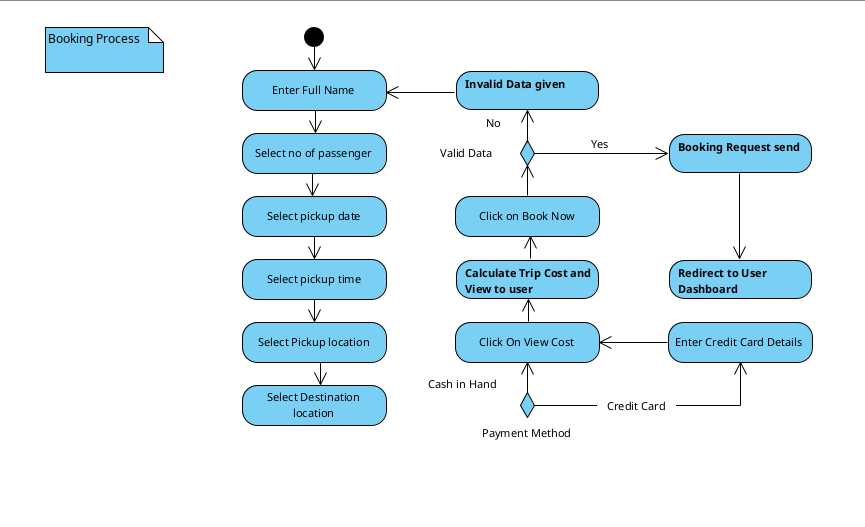
Diagram

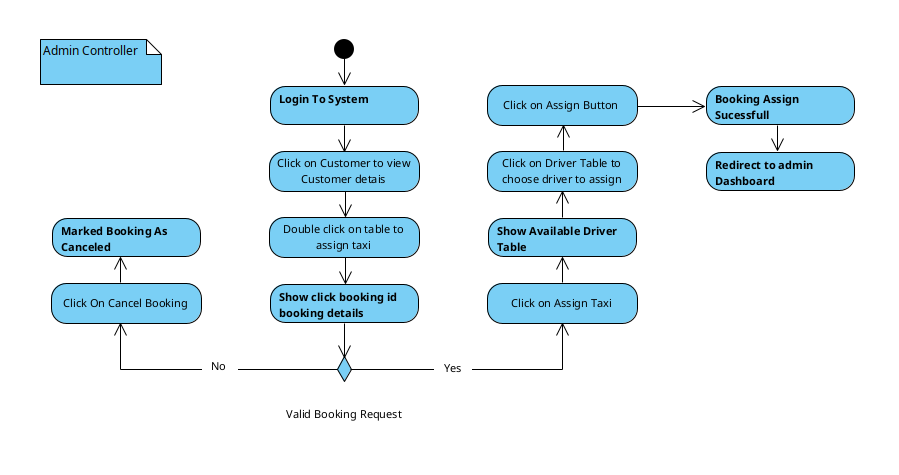
Description automatically generatedFigure 1: Taxi\_Booking\_System\_Usecase\_diagram

### Activity Diagram(s)

Figure 2: Login\_System\_Activity\_Diagram

Figure 3: Register\_System\_Activity\_Diagram

Figure 4: Booking\_System\_Activity\_Diagram

Figure 5: Admin\_Control\_System\_Activity\_Diagram

### Class Diagram(s)

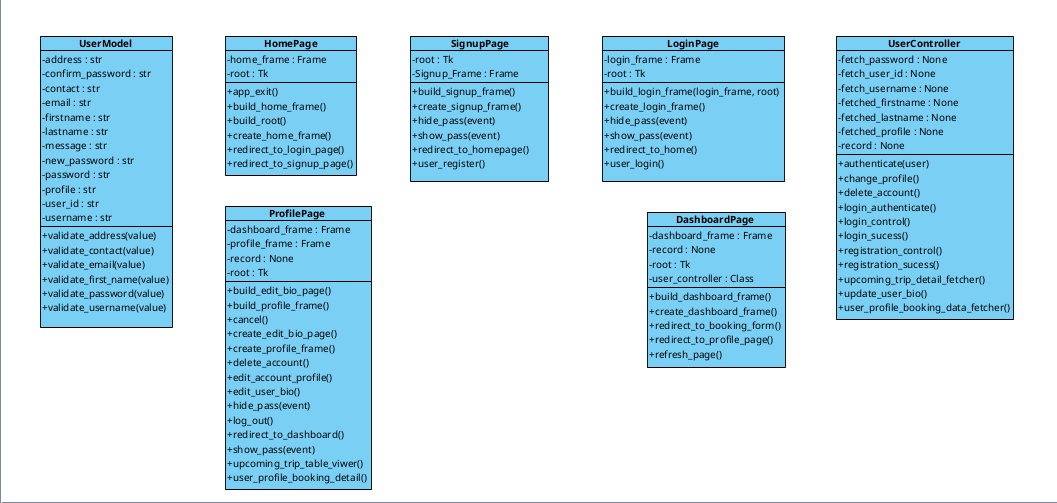
Figure 6: user\_class\_diagram

Figure 7: admin\_class\_diagram

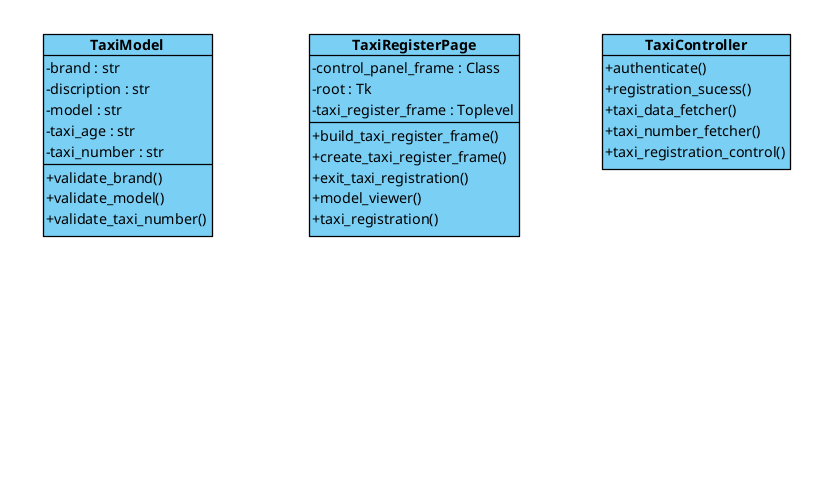
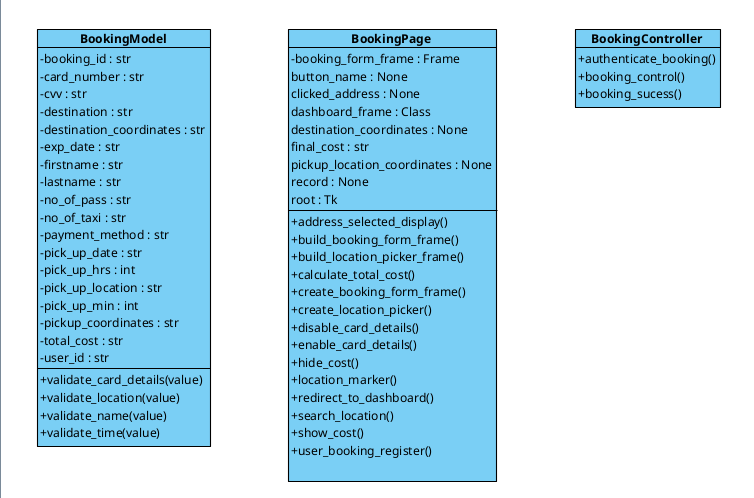
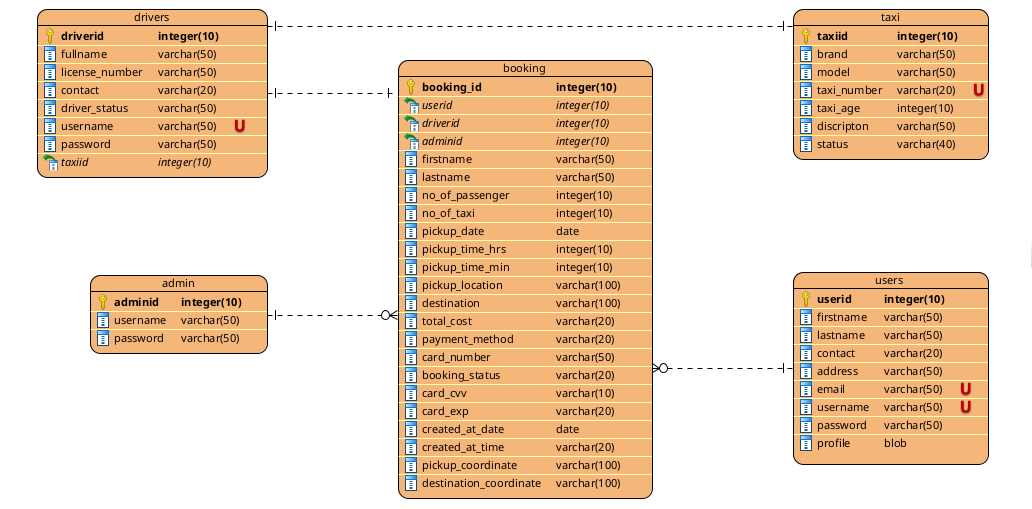
Figure 8: taxi\_class\_diagram

Figure 9: driver\_class\_diagram

Figure 10: booking\_class\_diagram

## Database Design

Figure 11: turbo\_tb\_database\_ERD

### Logical Database Design

Diagram

Description automatically generatedFigure 12: databae\_ERM\_figure

### Physical Database Design

#### Skeleton Tables (with Primary Keys and Foreign Keys)

Users(**userid**, firstname, lastname, contact, address, email, username,

password, profile)

Booking (**booking\_id**, userid\*, driverid\*, adminid\*, firstname,lastname, no\_of\_passenger, no\_of\_taxi, pickup\_date, pickup\_time\_hrs, pickup\_time\_min, pickup\_location, destination, total\_cost, payment\_method, card\_number, booking\_status, card\_cvv, card\_exp, created\_at\_date, created\_at\_time, pickup\_coordinates, destination\_coordinates)

Taxi (**taxiid**, brand, model, taxi\_number, taxi\_age, discription, status)

driver (**driverid**, fullname, license\_number, contact, driver\_status, username, password, taxiid\*)

Admin (**adminid**, username, password)

#### Data Dictionary

Table 1: user\_data\_dicrionary\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Users** | | | | | |
| **Discription :** Users Details | | | | | |
| **Field Name** | **Data Type** | **Length** | **Index** | **Null** | **Discription** |
| **Userid**  **(Primary)** | **int** | **10** | **PK** | **No** | **Auto Increment** |
| **firstname** | **varchar** | **50** |  | **No** | **Firstname of user** |
| **lastname** | **varchar** | **50** |  | **N0** | **Lastname of user** |
| **contact** | **varchar** | **20** |  | **No** | **Contact detail of user** |
| **address** | **varchar** | **50** |  | **No** | **User’s location detail** |
| **email** | **varchar** | **50** | **U** | **No** | **Valid email address of user** |
| **username** | **varchar** | **50** | **U** | **No** | **Username for login** |
| **password** | **varchar** | **50** |  | **No** | **Password for personal security** |
| **profile** | **blob** |  |  | **No** | **User’s profile picture** |

Table 2: booking\_data\_dictionary\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Booking** | | | | | |
| **Discription :** Booking details made by user | | | | | |
| **Field Name** | **Data Type** | **Length** | **Index** | **Null** | **Discription** |
| **booking\_id**  **(Primary)** | **int** | **10** | **PK** | **No** | **Auto Increment** |
| **userid** | **int** | **10** | **Fk** | **No** | **User table foreign key** |
| **driverid** | **int** | **10** | **Fk** |  | **drivers table foreign key** |
| **adminid** | **int** | **10** | **Fk** |  | **admin table foreign key** |
| **firstname** | **varchar** | **50** |  | **No** | **User’s first name** |
| **lastname** | **varchar** | **50** |  | **No** | **User’s last name** |
| **no\_of\_passenger** | **int** | **50** |  | **No** | **No of passenger in a trip** |
| **pickup\_date** | **date** | **50** |  | **No** | **Date for the pickup** |
| **pickup\_time\_hrs** | **int** | **10** |  | **No** | **Time for the pickup HRS** |
| **pickup\_time\_min** | **int** | **10** |  | **No** | **Time for the pickup MIN** |
| **pickup\_location** | **varchar** | **100** |  | **No** | **Location to pickup** |
| **destination** | **varchar** | **100** |  | **No** | **Final destination of trip** |
| **total\_cost** | **varchar** | **20** |  | **No** | **Total cost to be paid** |
| **payment\_method** | **varchar** | **20** |  | **No** | **Method of payment** |
| **card\_number** | **varchar** | **50** |  |  | **Credit card number** |
| **card\_cvv** | **varchar** | **20** |  |  | **Credit card cvv** |
| **card\_exp** | **varchar** | **10** |  |  | **Credit card exp date** |
| **created\_at\_date** | **date** |  |  | **No** | **Booking created date** |
| **created\_at\_time** | **varchar** | **20** |  | **No** | **Booking created time** |
| **pickup\_coordinates** | **varchar** | **100** |  | **No** | **Pickup location coordinates** |
| **destination\_coordinates** | **varchar** | **100** |  | **No** | **Destination coordinates** |
| **booking\_status** | **varchar** | **20** |  | **No** | **Status of booking request** |

Table 3: admin\_data\_dictionary\_table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Admin** | | | | | |
| **Discription :** Admin Details | | | | | |
| **Field Name** | **Data Type** | **Length** | **Index** | **Null** | **Discription** |
| **adminid**  **(Primary)** | **int** | **10** | **PK** | **No** | **Auto Increment** |
| **username** | **varchar** | **50** |  | **No** | **Username to login to system** |
| **password** | **varchar** | **50** |  | **N0** | **Account password** |

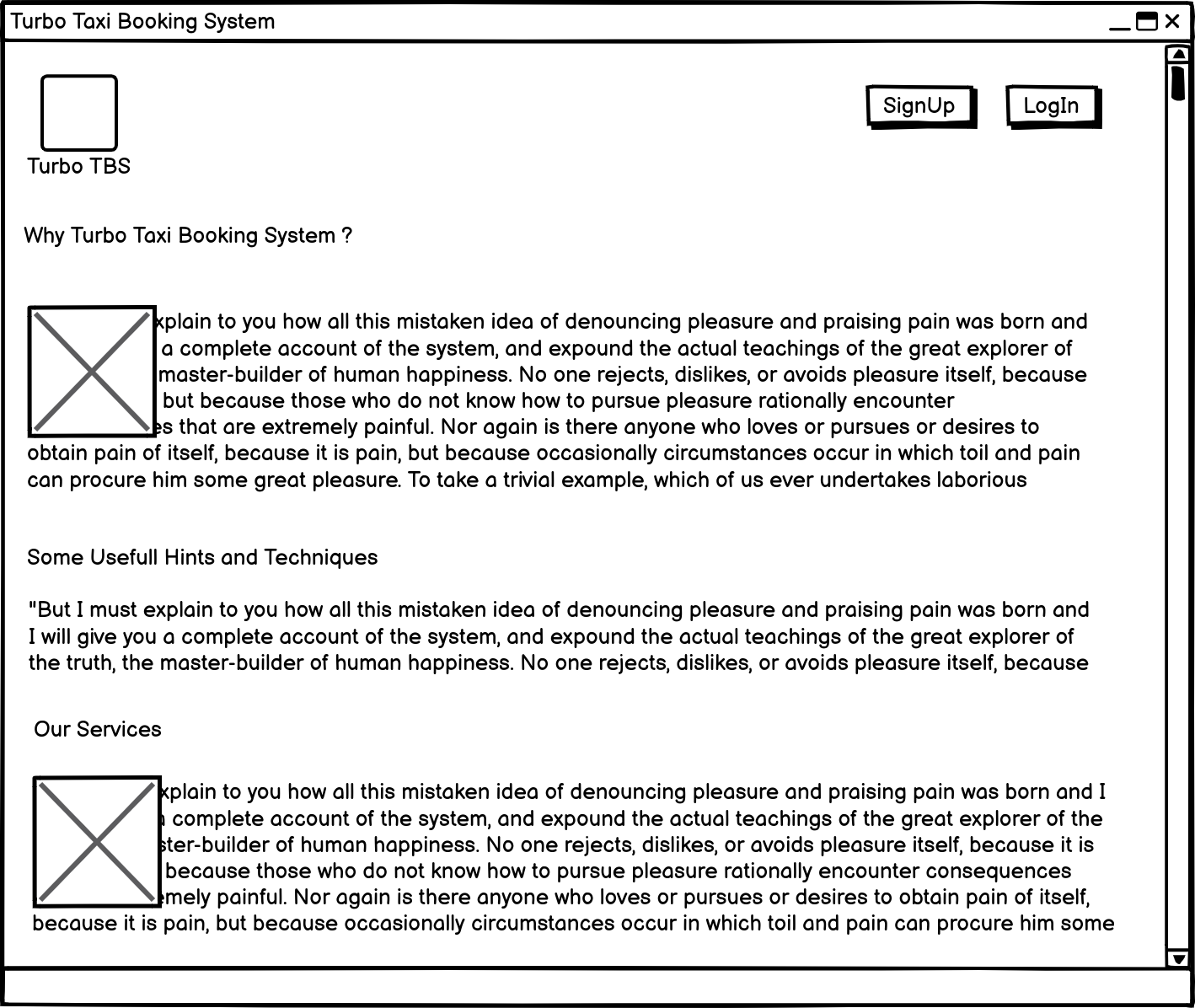
Table 4: taxi\_data\_dictionary\_table

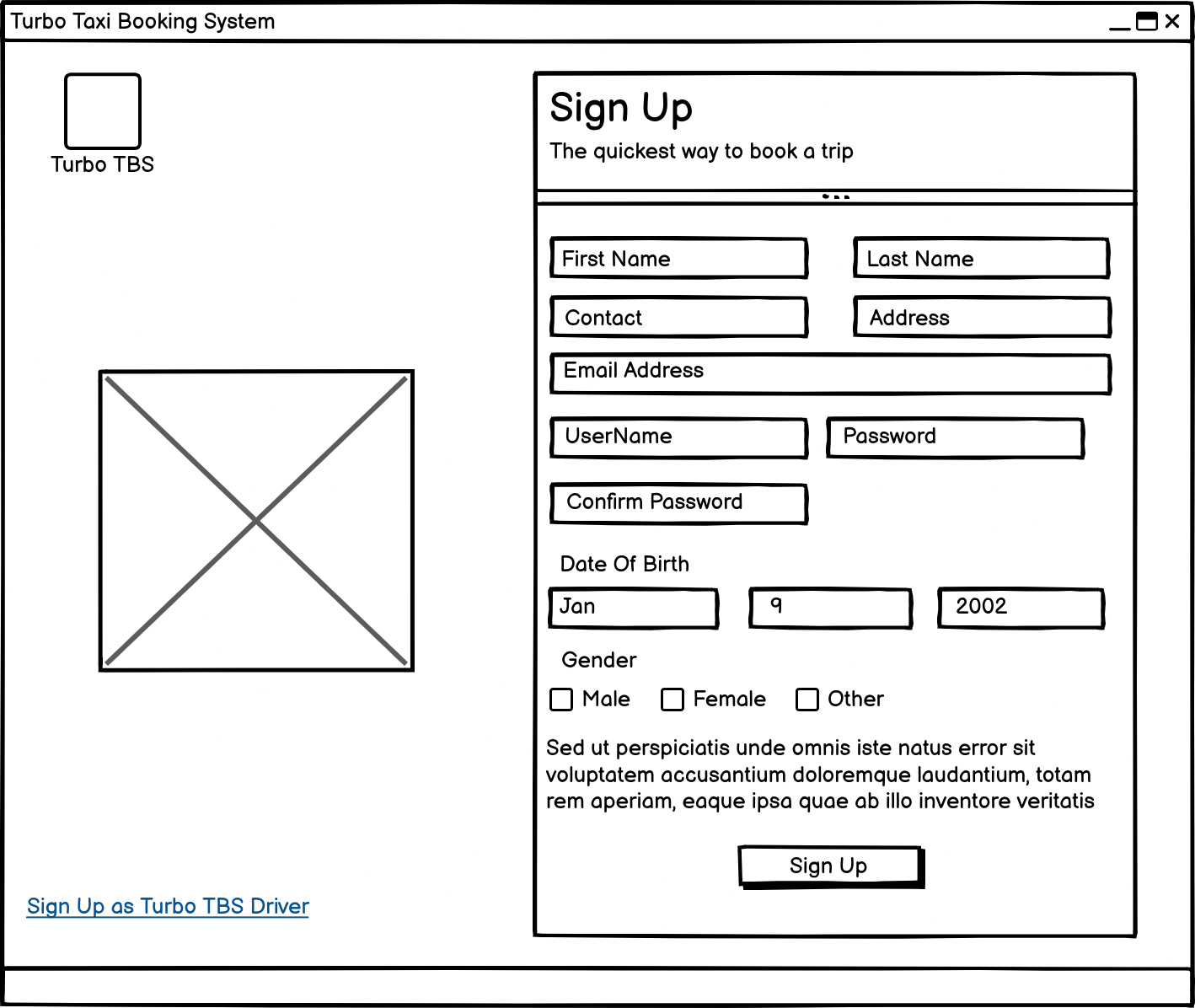
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Taxi** | | | | | |
| **Discription :** Taxi Details | | | | | |
| **Field Name** | **Data Type** | **Length** | **Index** | **Null** | **Discription** |
| **taxiid**  **(Primary)** | **int** | **10** | **PK** | **No** | **Auto Increment** |
| **brand** | **varchar** | **50** |  | **No** | **Brand of a taxi** |
| **model** | **varchar** | **50** |  | **N0** | **Model of a taxi brand** |
| **taxi\_number** | **varchar** | **20** |  | **No** | **Taxi number** |
| **taxi\_age** | **int** | **10** |  | **No** | **Used time of taxi** |
| **discription** | **varchar** | **50** |  | **No** | **Details about taxi** |
| **status** | **varchar** | **40** |  | **No** | **Taxi assign status** |

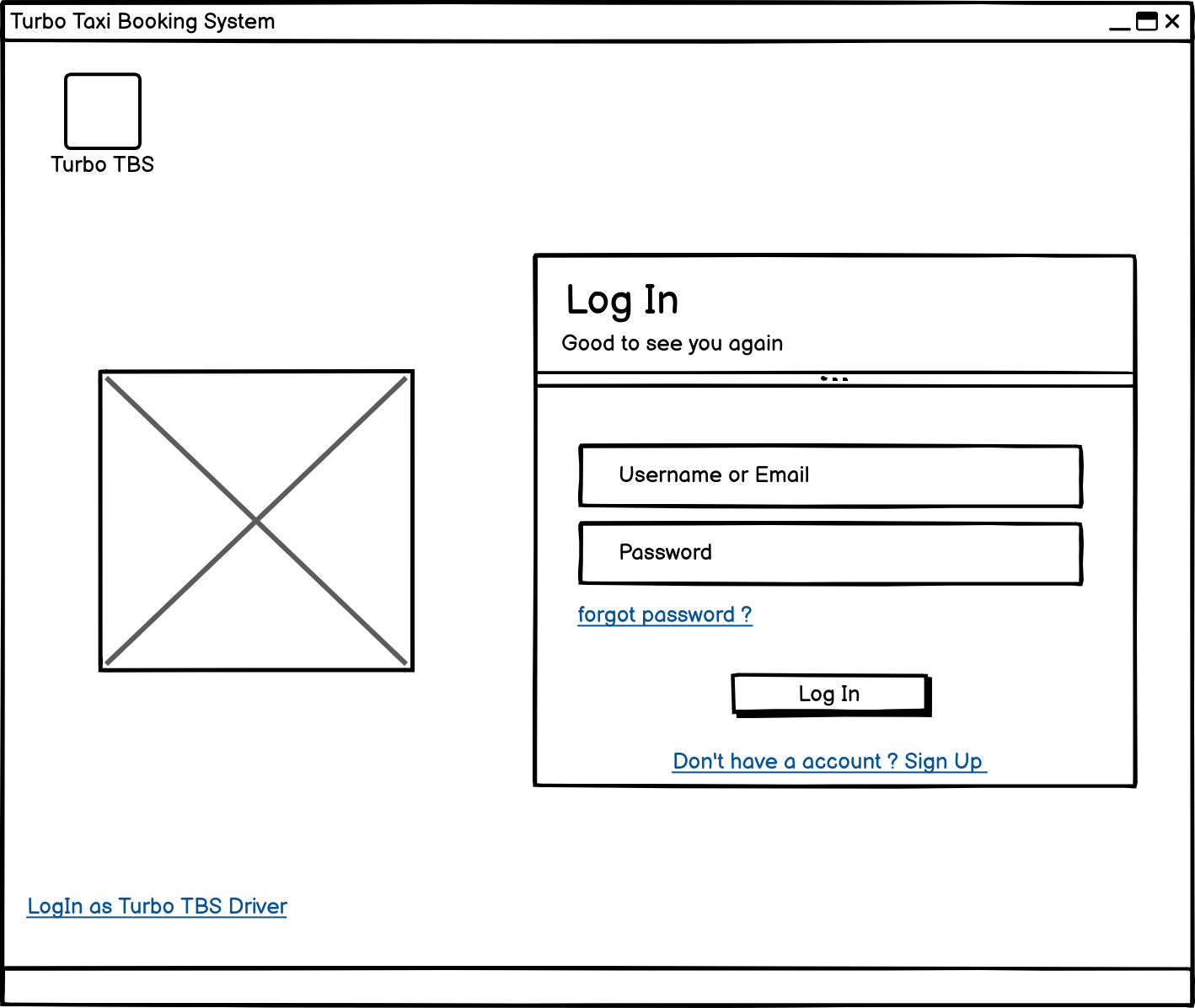
Table 5: drivers\_data\_dictionary\_table

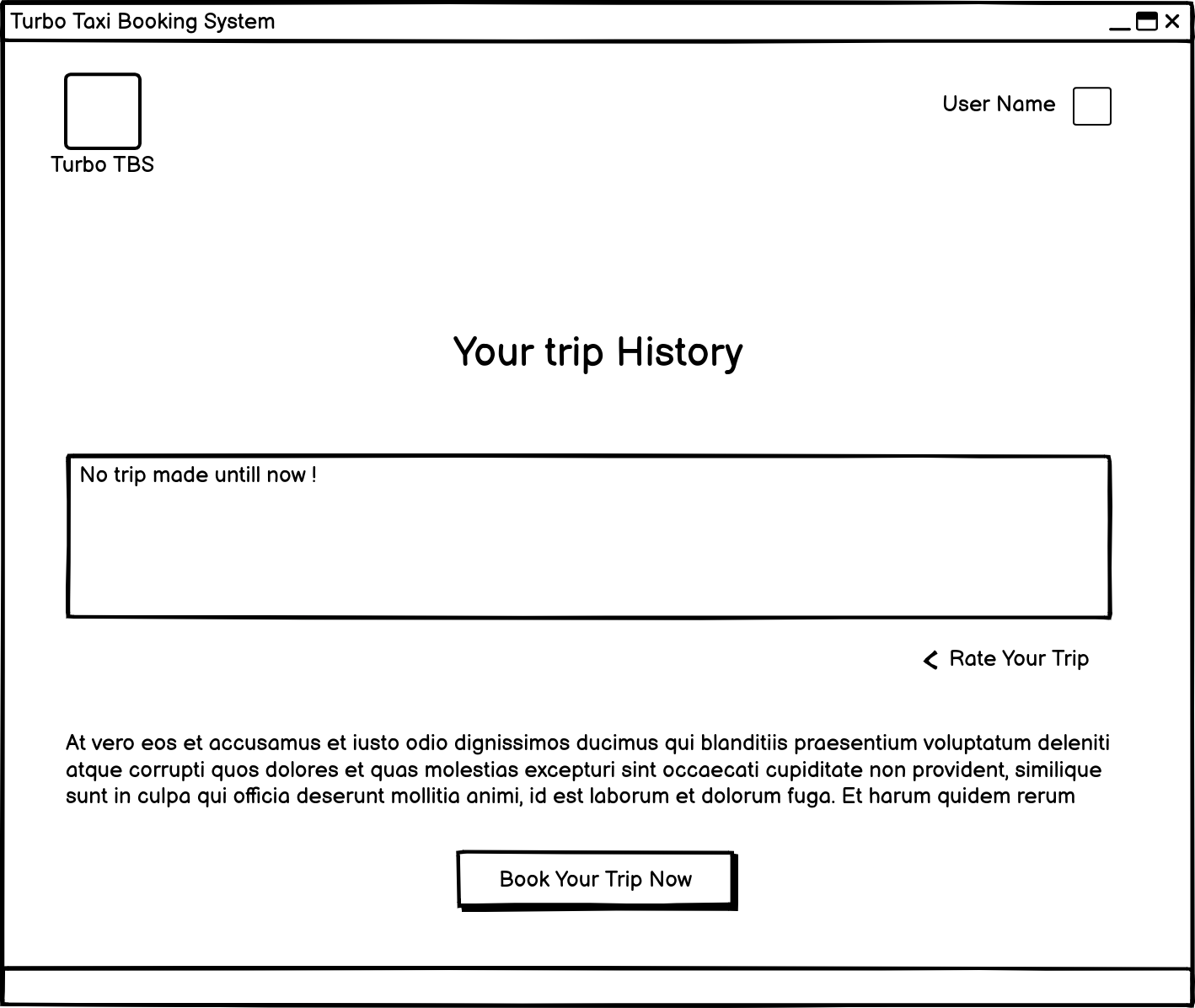
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Drivers** | | | | | |
| **Discription :** Drivers Details | | | | | |
| **Field Name** | **Data Type** | **Length** | **Index** | **Null** | **Discription** |
| **driverid**  **(Primary)** | **int** | **10** | **PK** | **No** | **Auto Increment** |
| **fullname** | **varchar** | **50** |  | **No** | **Firstname of driver** |
| **license\_number** | **varchar** | **50** |  | **N0** | **Lastname of driver** |
| **contact** | **varchar** | **20** |  | **No** | **Contact detail of driver** |
| **driver\_status** | **varchar** | **50** |  | **No** | **Driver’s assign status** |
| **username** | **varchar** | **50** | **U** | **No** | **Username for login** |
| **password** | **varchar** | **50** | **U** | **No** | **Password for personal security** |
| **taxiid** | **int** | **10** | **FK** | **No** | **Foreign key from taxi table** |

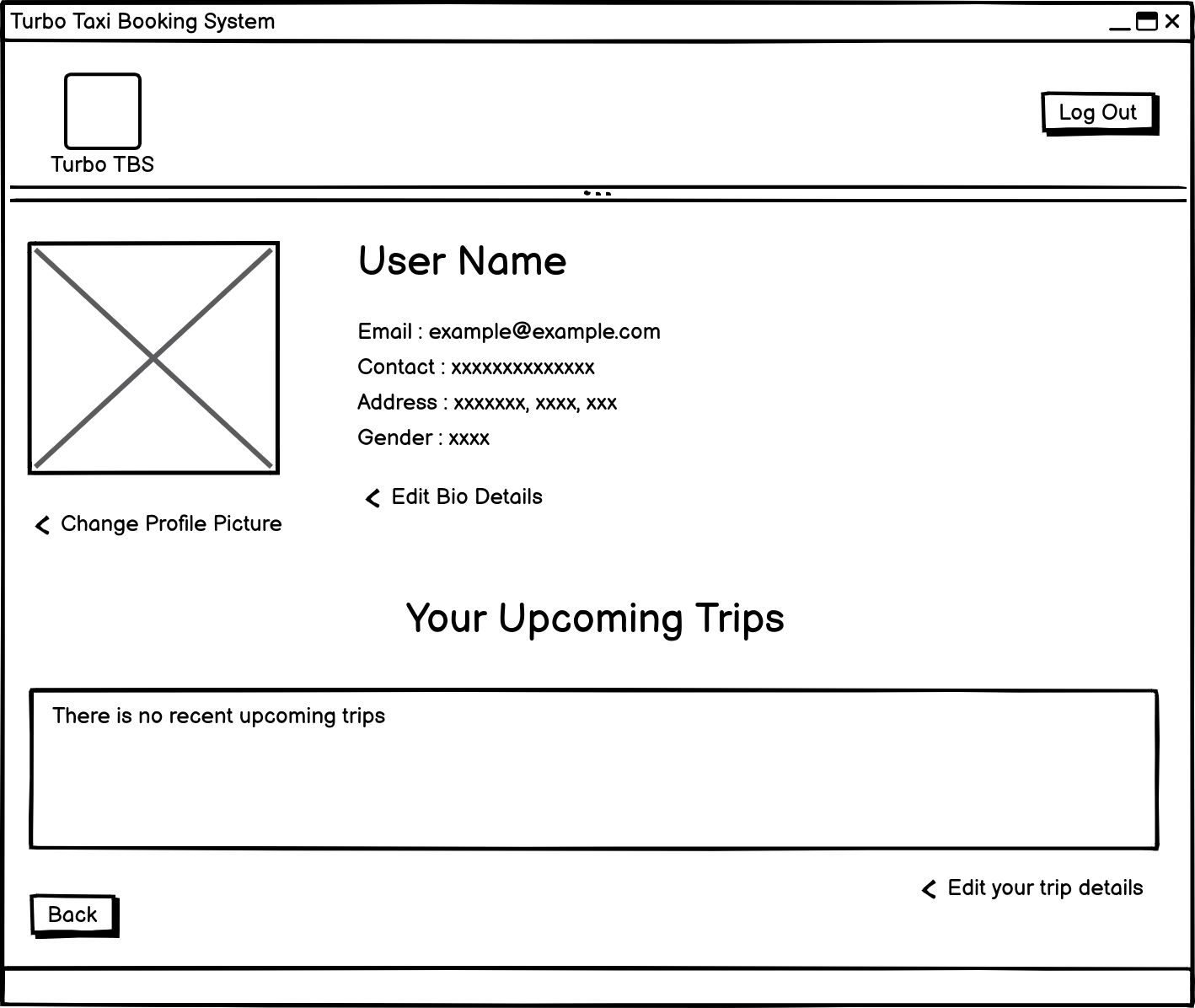
## User Interface Design

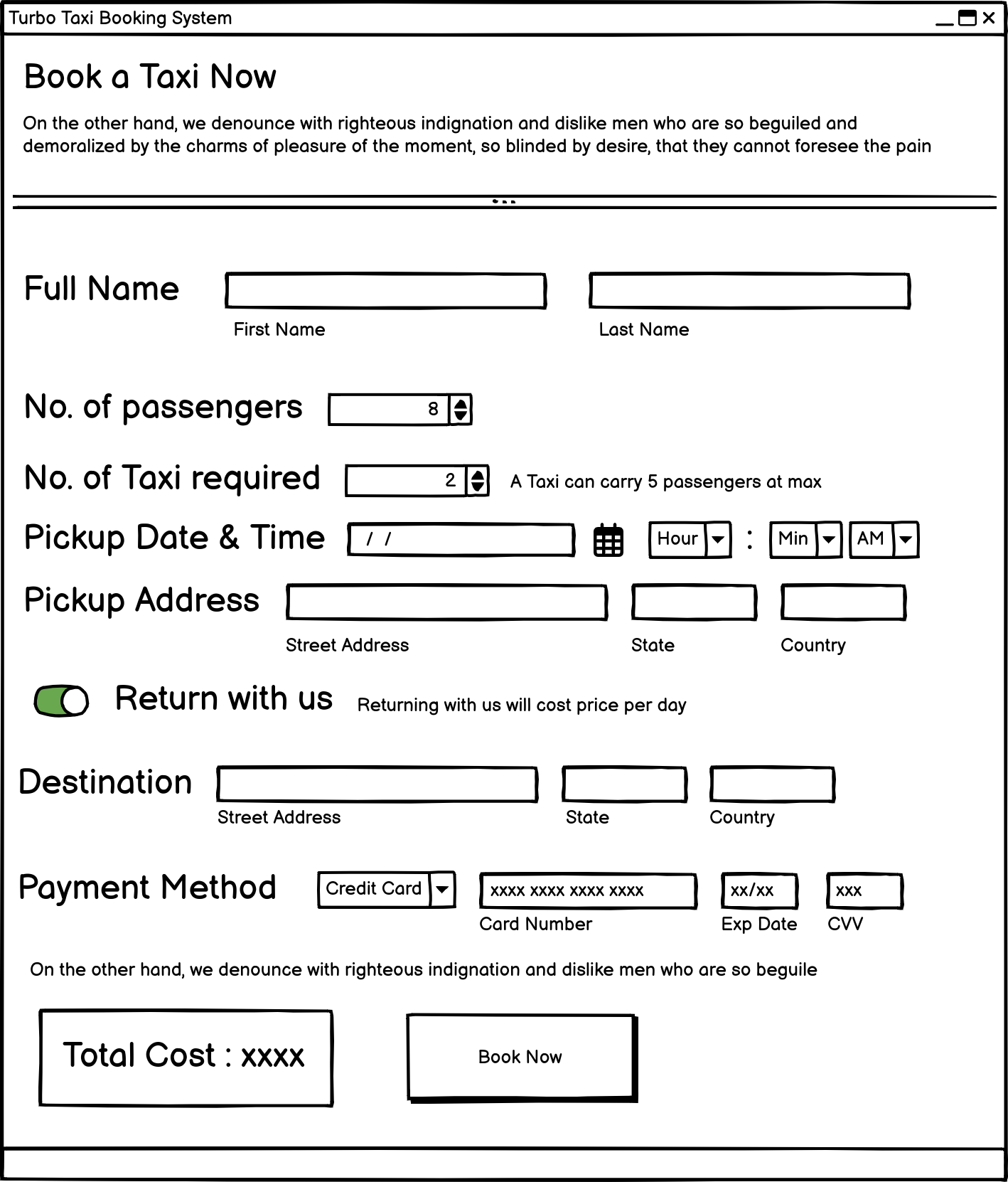
Figure 13: homepage\_interface\_design

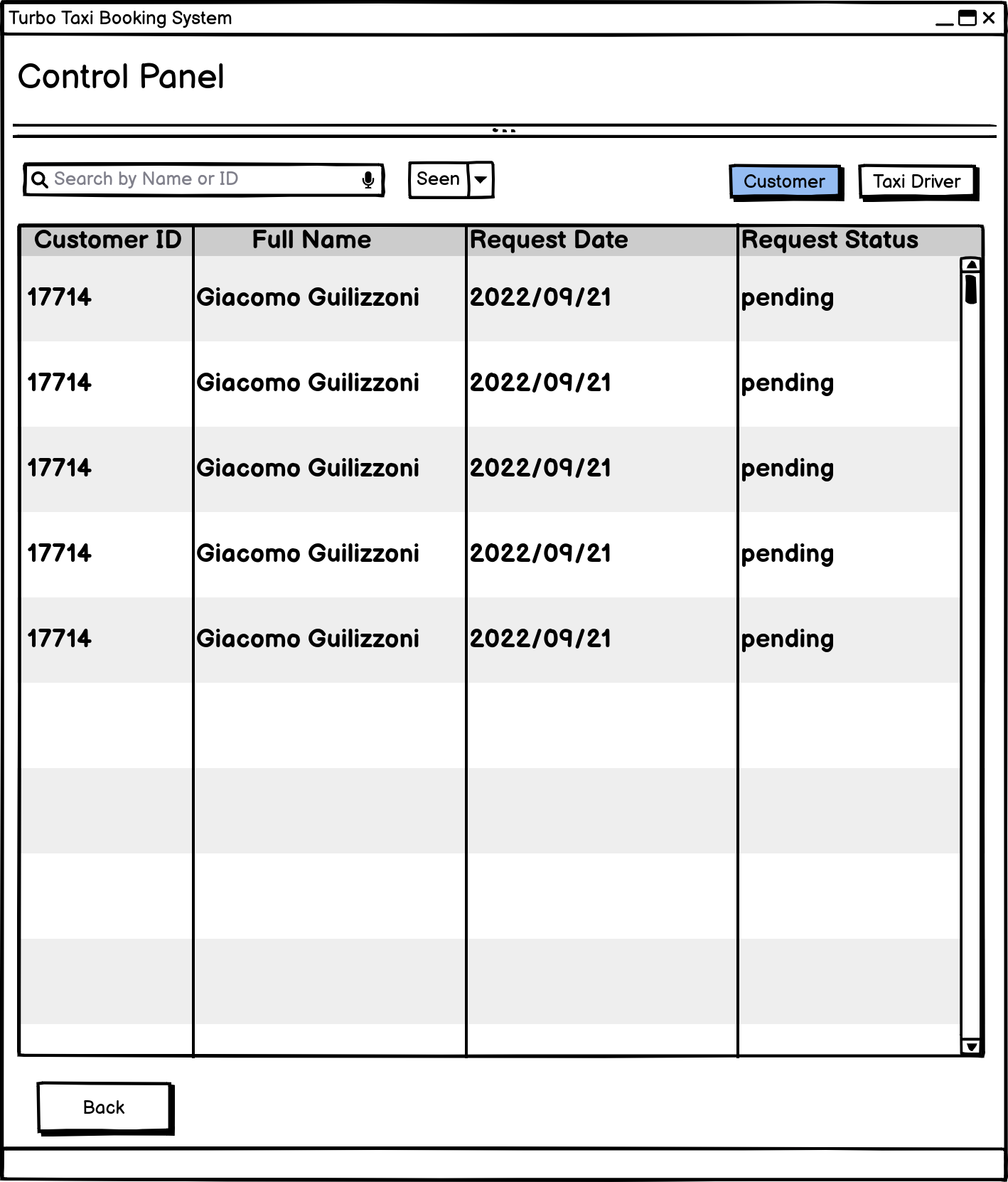
Figure 14: user\_signup\_interface\_design

Figure 15: user\_login\_interface\_design

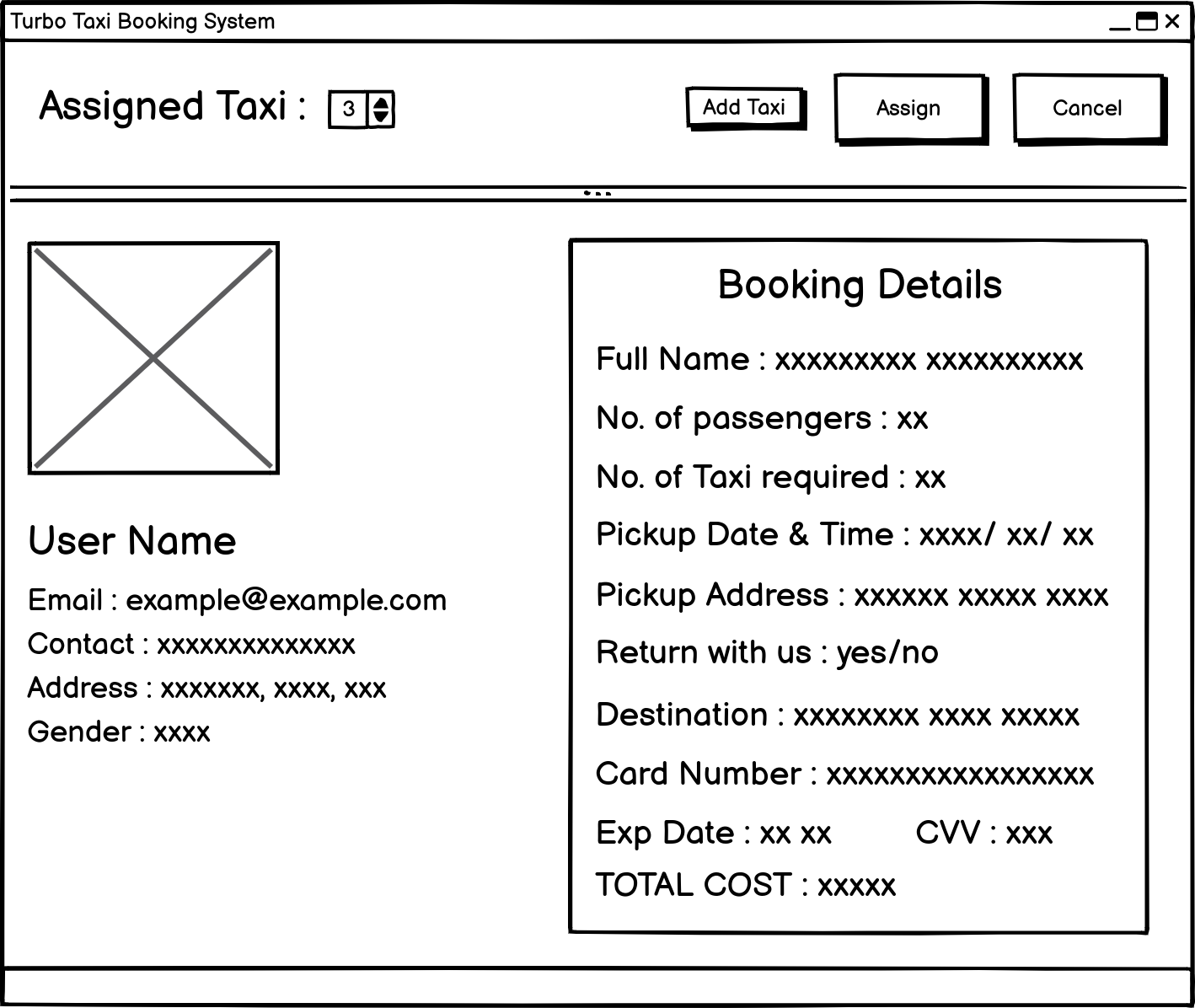
Figure 16: user\_dashboard\_interface\_design

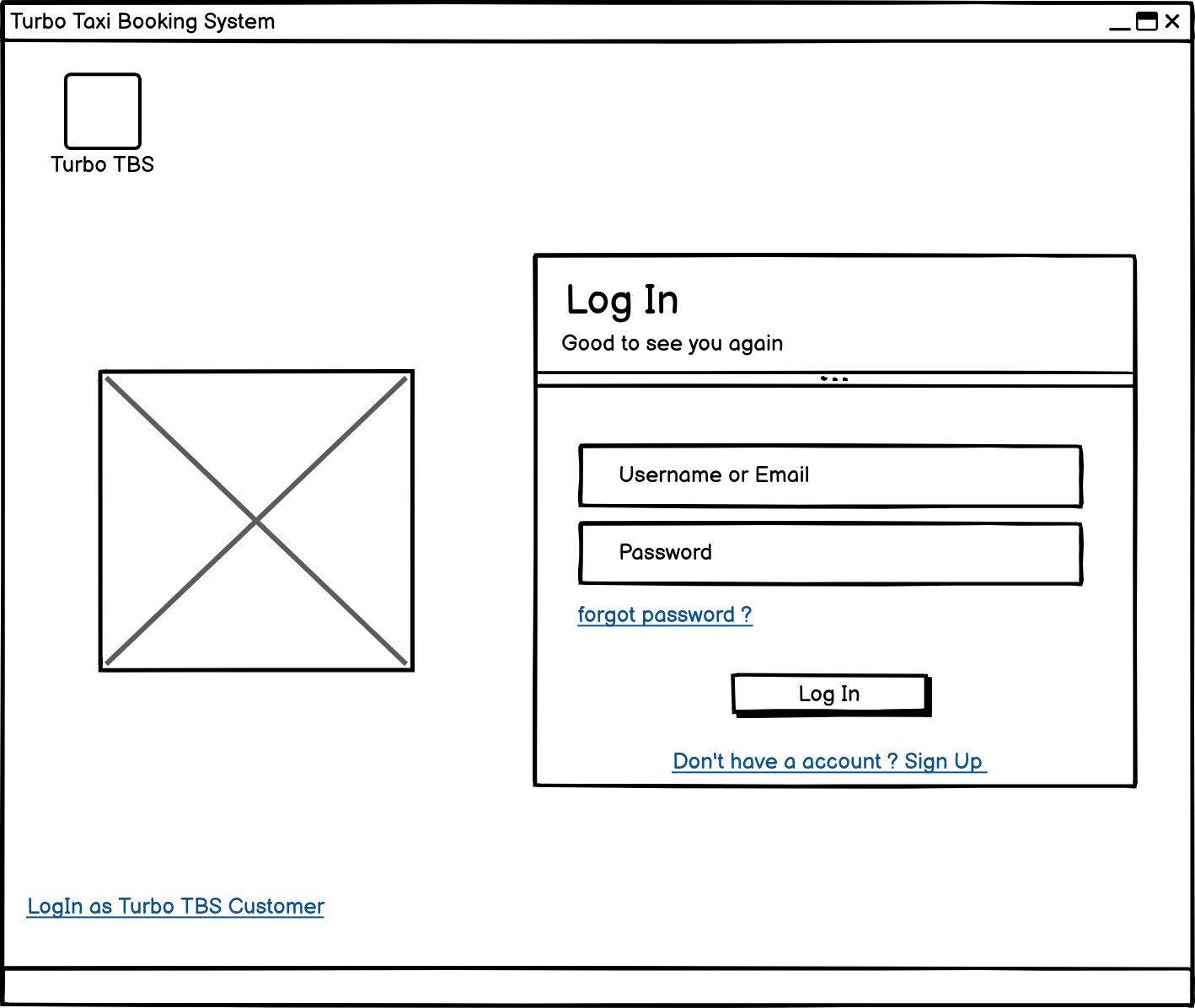
Figure 17: user\_profile\_interface\_design

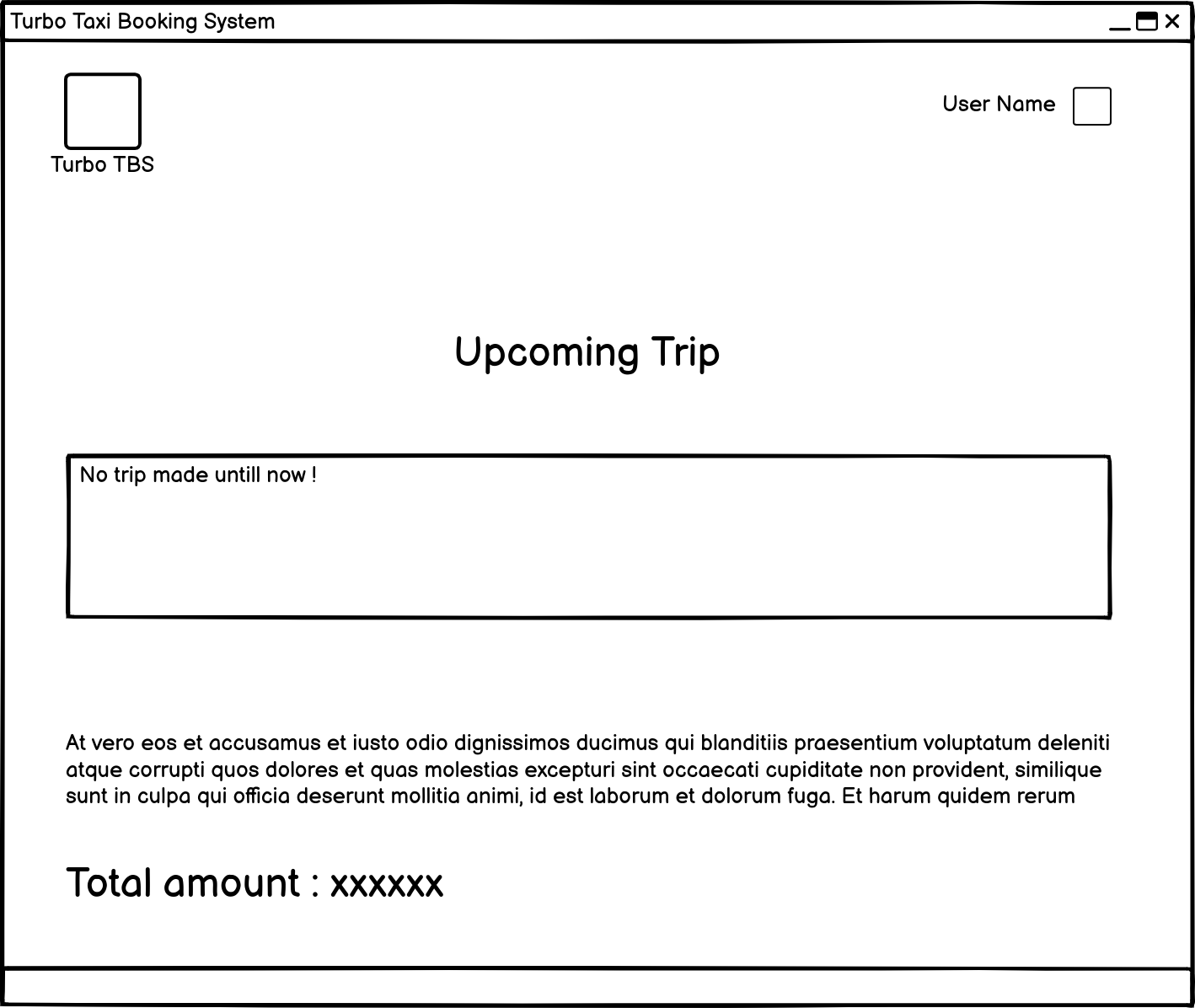
Figure 18: booking\_form\_interface\_design

Figure 19: admin\_interface\_design

## 

Figure 20: booking\_detail\_admin\_view\_interface\_design

Figure 21: driver\_login\_interface\_design

Figure 22: driver\_daashboard\_interface\_design

## Algorithm Design

Flow of the development process

• Flow of the system hierarchy

Taxi booking system

• The system :

Main dashboard : SignUp and SignIn on the main dashboard, with a visible working gallery

1. After clicking SignUp, "SignUp as Customer" will open up and also provide a link for "SignUp as Worker."
2. Both forms will require different information and will be saved on different tables of the database.
3. After clicking Sign In, the Sign In as Customer page will appear, along with a link to Sign In as Worker.
4. The same information will be required for both forms. username, email, and password

type of user

i. Customer ( The one who does the booking action ) :

Working of a customer :

1. Register as a customer

2. Sign up with all the required details.

3. If all the validation is completed, the user will be redirected to the login page.

4. The user will login with the given login email or username and password.

5. The user will be redirected to the user dashboard, which contains a "book now" button, a profile picture, a profile name, and some other details about the system (a hint).

6. Clicking on "Book Now" will redirect the user to the booking form.

7. After entering all of the necessary information, the user can click the Book button, which sends a booking request to the taxi company for approval.

8. This will redirect the user to their dashboard.

9. By clicking on the profile picture or the username on the dashboard, users will be redirected to the profile page.

10. Profile page will include all the previous booking history and ongoing booking information

11. The profile page will allow you to edit the booking details as well as your personal information.

12. The GUI must be good, as each function should change the pages for each different function.

13. Bookings can be canceled from the profile page until the trip date is more than one day away.

14. The completed trip can be seen, and the driver can be rated afterward.

ii . Admin (The one who assign the taxi and driver to the request trip)

Working of a Admin :

1. Log in as an administrator with the provided username and password.

2. Admin should be redirected to the admin dashboard.

3. The portal should contain two options (a taxi driver control panel and a customer booking request control panel).

4. Clicking on "Taxi Driver" (this will bring up a list of all registered taxi drivers as well as all taxi drivers who wanted to register with valid information).

5. It will also provide all the information about the taxi driver after clicking the taxi driver's name on an ID-sized card.

6. It will give the administrator full authority to accept or decline a registration request for a taxi driver after documentation review.

7. Admin will be able to go back to the previous page where they can select between the driver and the customer control panel with a back button.

8. Clicking on "customer" (it will provide a list of all the customers, and you can view their details by clicking their name, which will pop up an ID-type UI) 9. You can view the booking request made by the customer.

10. You can see the drivers that are already booked for a trip and the available taxi drivers that are suitable for the trip.

11. To accept a trip, click on a requested customer name, an Id type UI pops up, and then click on assign taxi after checking the requested booking detail (number of taxi will allow you to input number of taxi vehicle number with add button).

12. The "booking for Customer XXXXX has been completed" message will be shown, and then the pop-up will close on its own.

13. Admin have full authority to accept or decline an offer if it seems reasonable.

14. Admin can ban a registered user or a driver if they are found breaking the rules of an organization.

iii . Driver (The One who are assigned for customer to complete the trip)

Working of Driver :

1. Sign up as a driver

2. Sign up with all the details (full name, address, contact information, gender, date of birth, username, email address, experience, upload of a license photo up to 25 MB, vehicle number, vehicle registration number).

3. If all the validation is completed, the user will be redirected to the login page.

4. The user will login with the given login email or username and password.

5. The user will be redirected to the user dashboard.

6. The driver dashboard contains total income, upcoming trips, some information for easy use, Profile Page

7. By clicking on the profile picture or the username on the dashboard, the user will be redirected to the profile page.

8. The profile page will include all the previous booking history and personal details.

9. After the trip is assigned to the driver, the driver must take the trip as the company requests, but if the driver is not available, the driver can change the status from "available" to "not available."

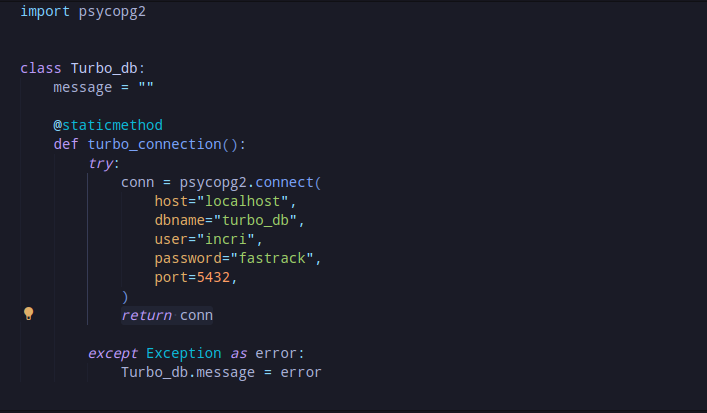
10. Changing the status will affect the admin page, so the filter won't show the driver as an available driver, and assigning the tax trip to such a driver is not allowed.

# Implementation

Using a Tkinter-based GUI, users of the Turbo Cab Booking System can request a taxi and calculating fares for the journey. Users may transmit or request data via a graphical user interface and a client server infrastructure, which allows for easy data sharing. With only one click, the user may use an external database system rather than using user storage to save data in the booking system. In addition, drivers have access to a list of their allocated travels, which are marked as finished when finished.

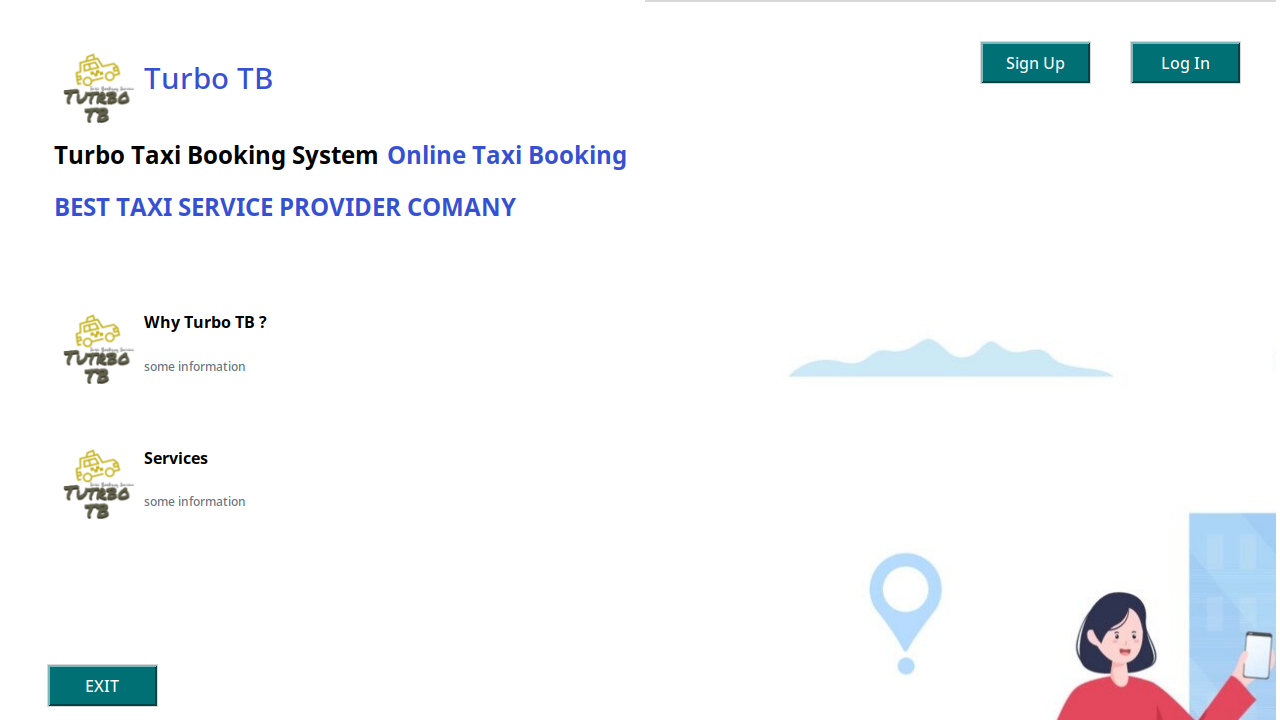
The planning and prototype phases of the development process were the first. All the criteria were initially put down after creating a weekly timetable. Visual Paradigm was used to create ER diagrams for the database design, and Balsamiq was utilized as a prototyping tool for the GUI concept design.

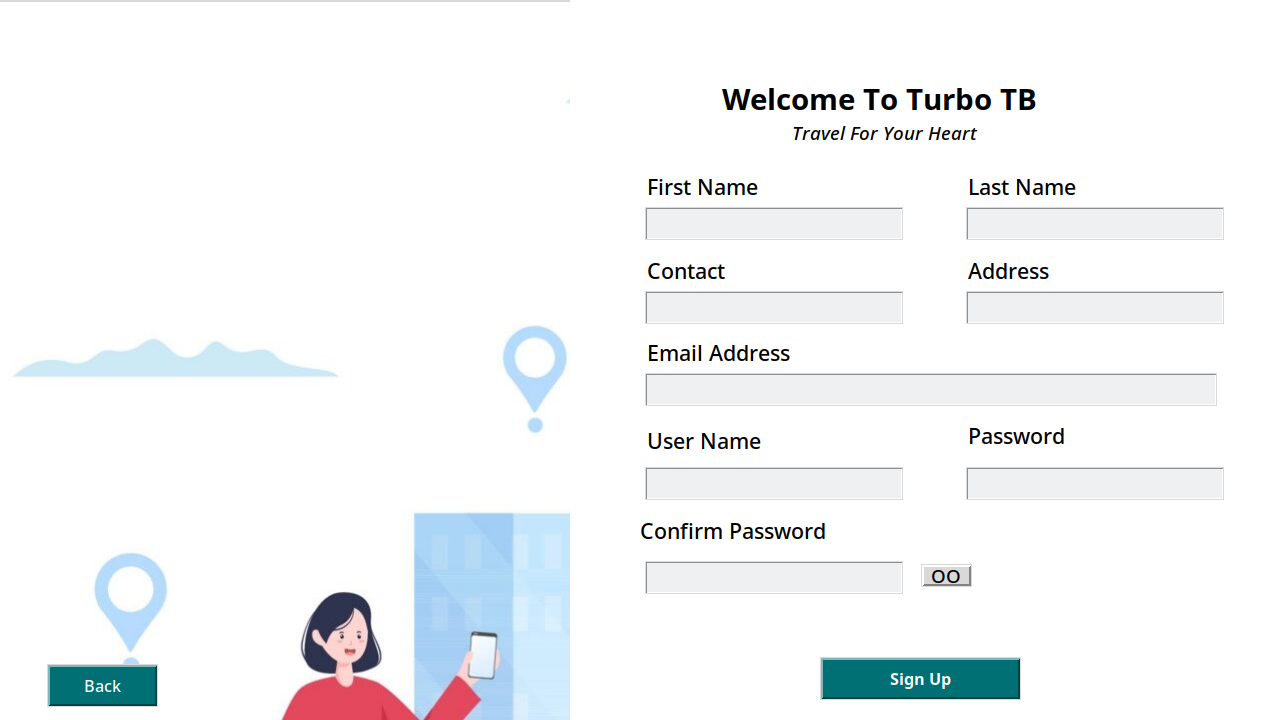
In order to learn and test PostgreSQL as we all know Postgres is a powerful and reliable database that is well-suited to a wide range of applications, Data was stored in a PostgreSQL database, and psycopg2 was utilized as a connector between the system and the database.

Figure 23: database\_connection

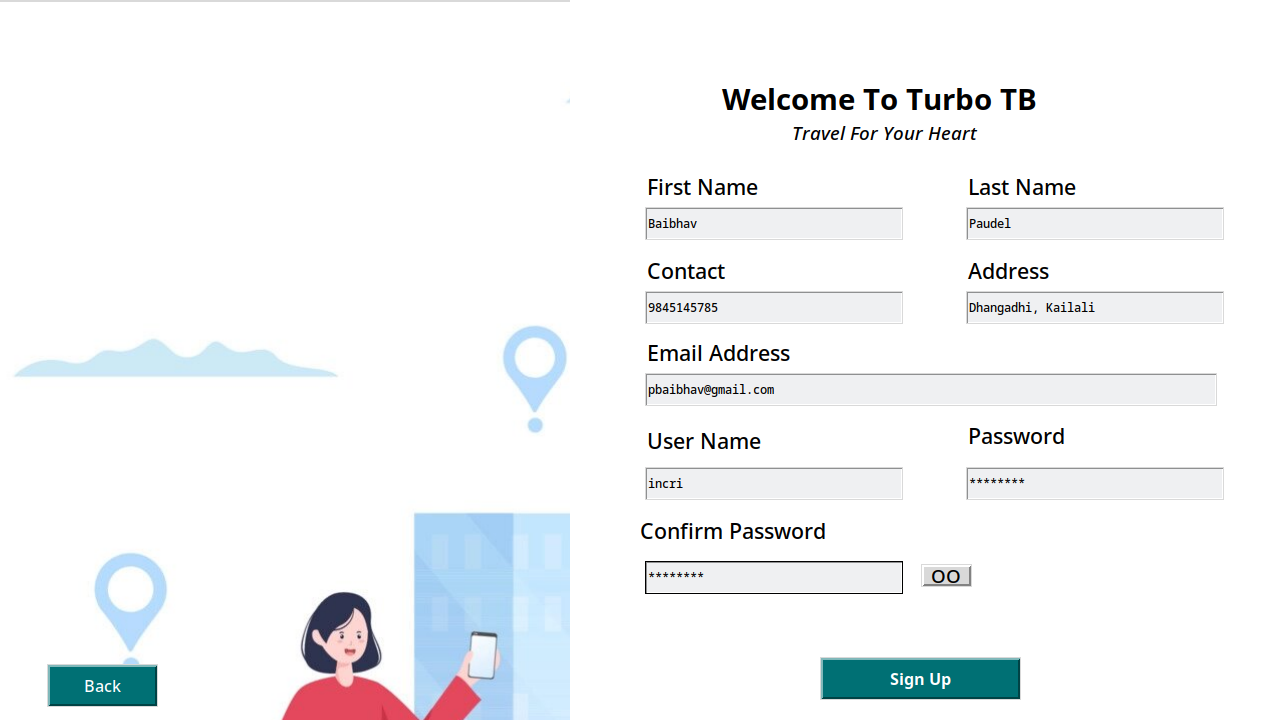
For frontend development, Tkinter, a built-in Python module, was used. The non-visual, or backend, portion of the code was likewise created in Python utilizing the MVC design pattern, where the model manages all of a module's important getter and setter and carries out validation. The frontend, or visual representation, of a program and function that provides data to the model layer's getter for validation is managed by the view layer of a project and The function that establishes a connection to the database system and puts data in the appropriate table according to the SQL command is called by the view layer from controller layer when the validation is finished.

Home page of taxi booking system .

Figure 24: home\_page

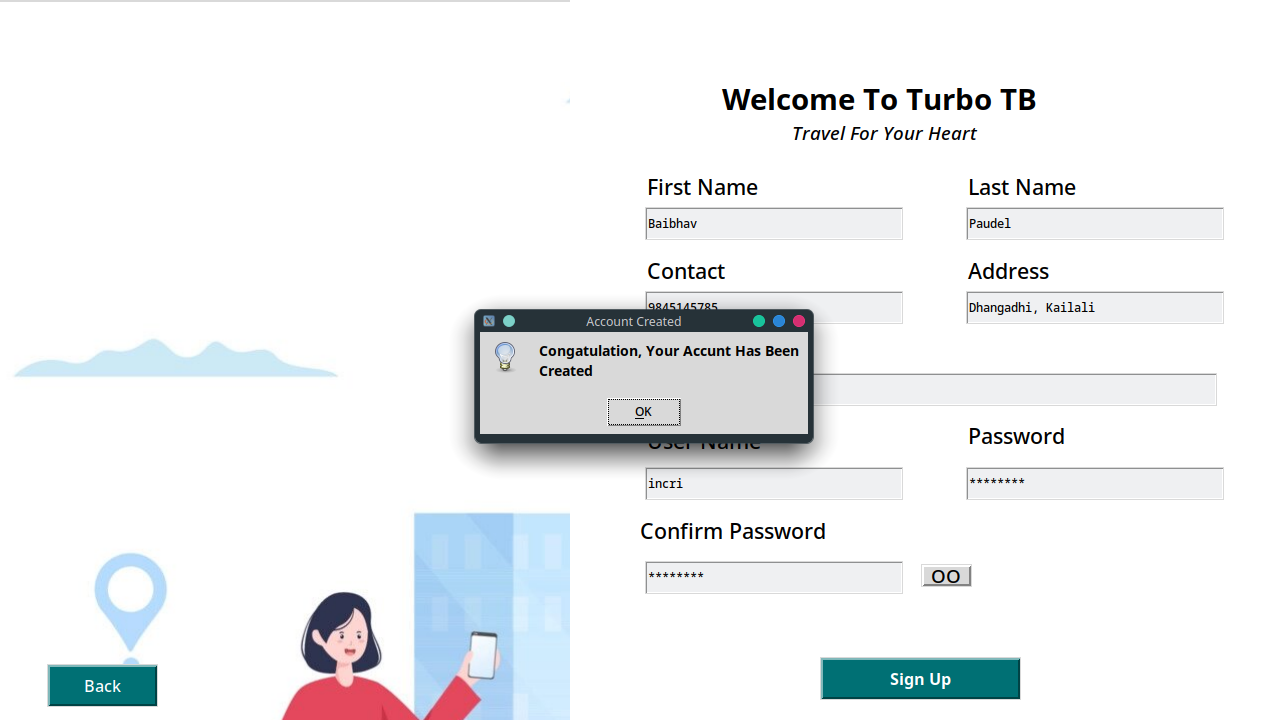
Figure 25: signup\_page

Signup page for the user where unregistered user register for the first time.

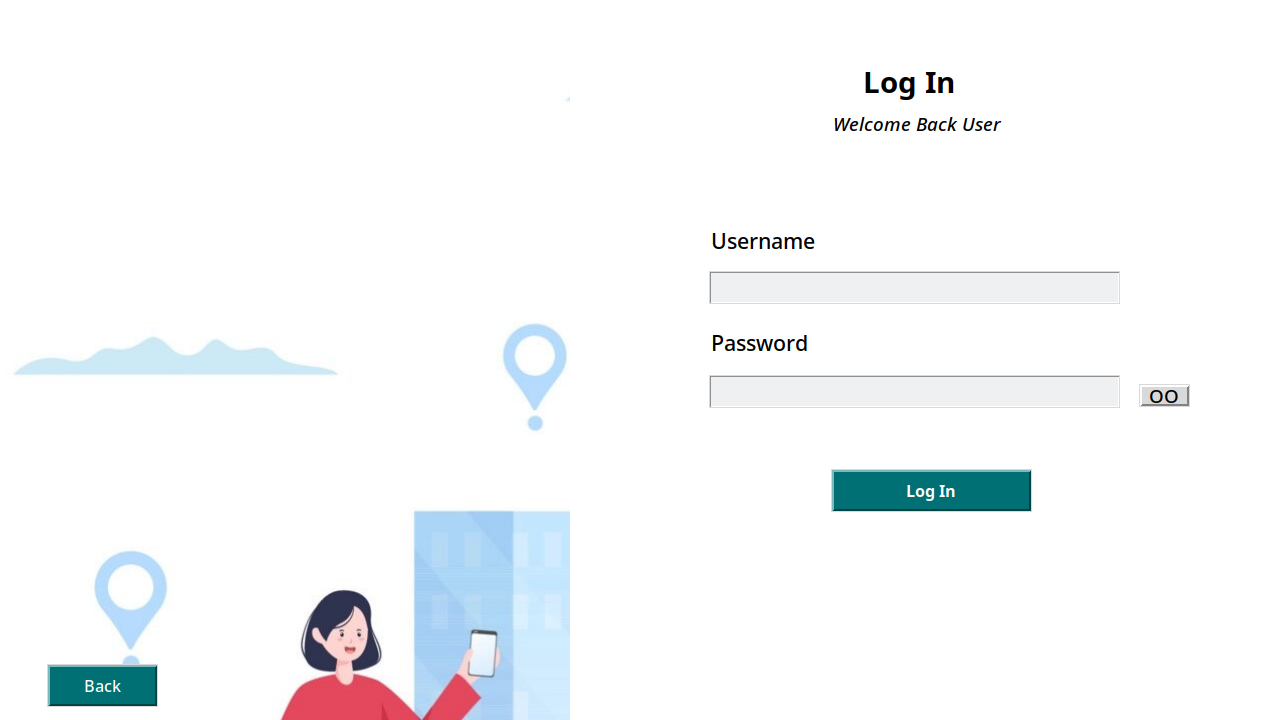
Figure 26: signup\_page\_with data

Registering with valid data.

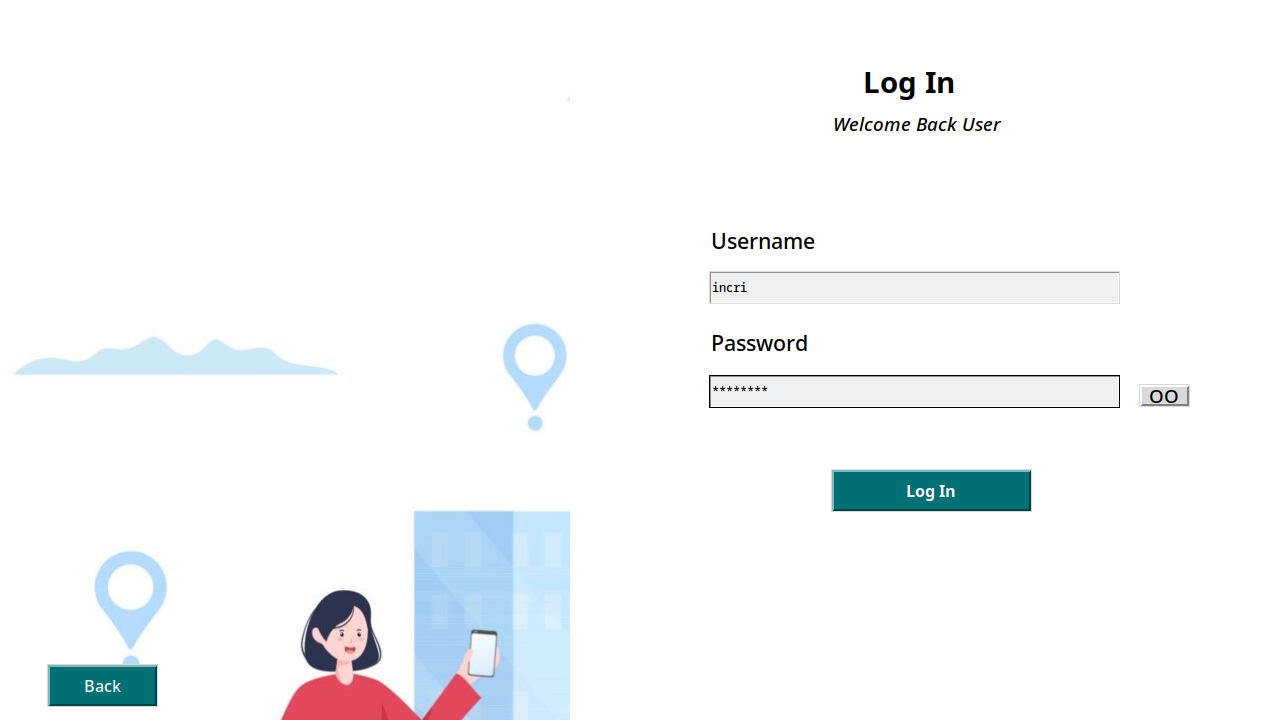
After entering proper information and pressing the signup button, a messagebox with a successful account creation message appears.

Figure 27: after\_clicking\_signup\_button

The login page is used to log into the system if the user is already registered.

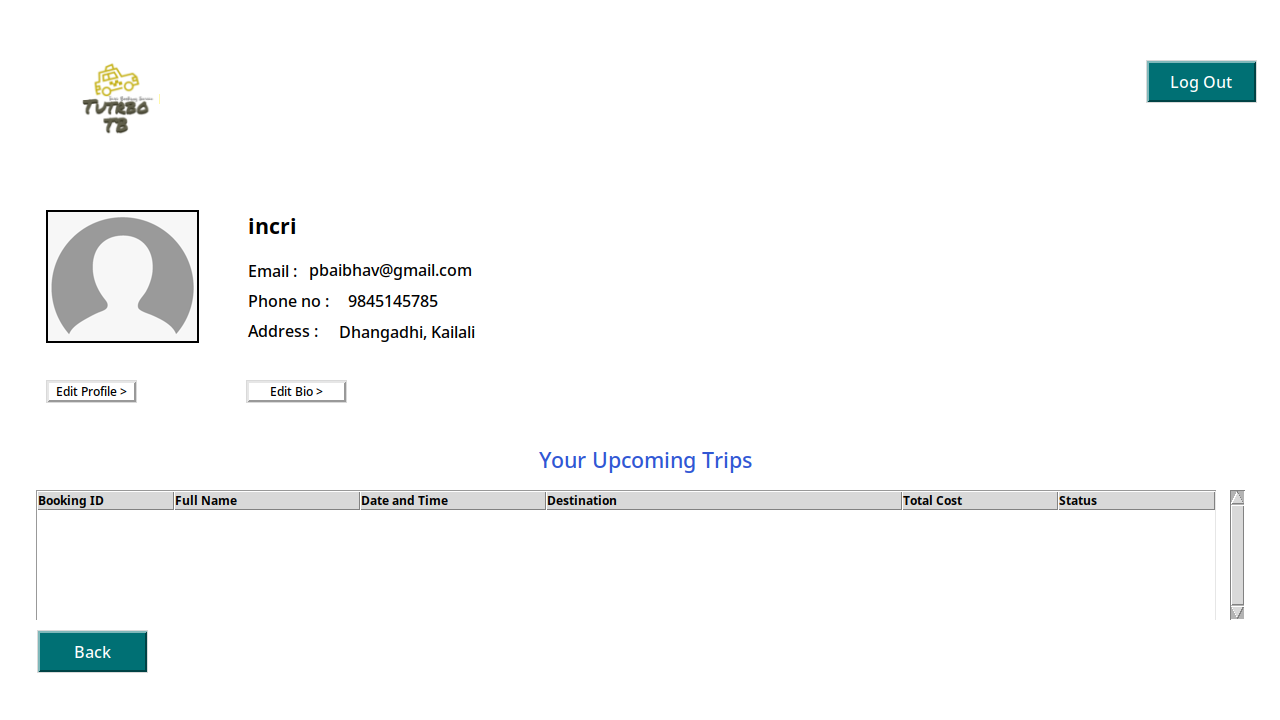
Figure 28: login\_page

Logging in with valid registered data.

Figure 29: login\_page\_with\_data

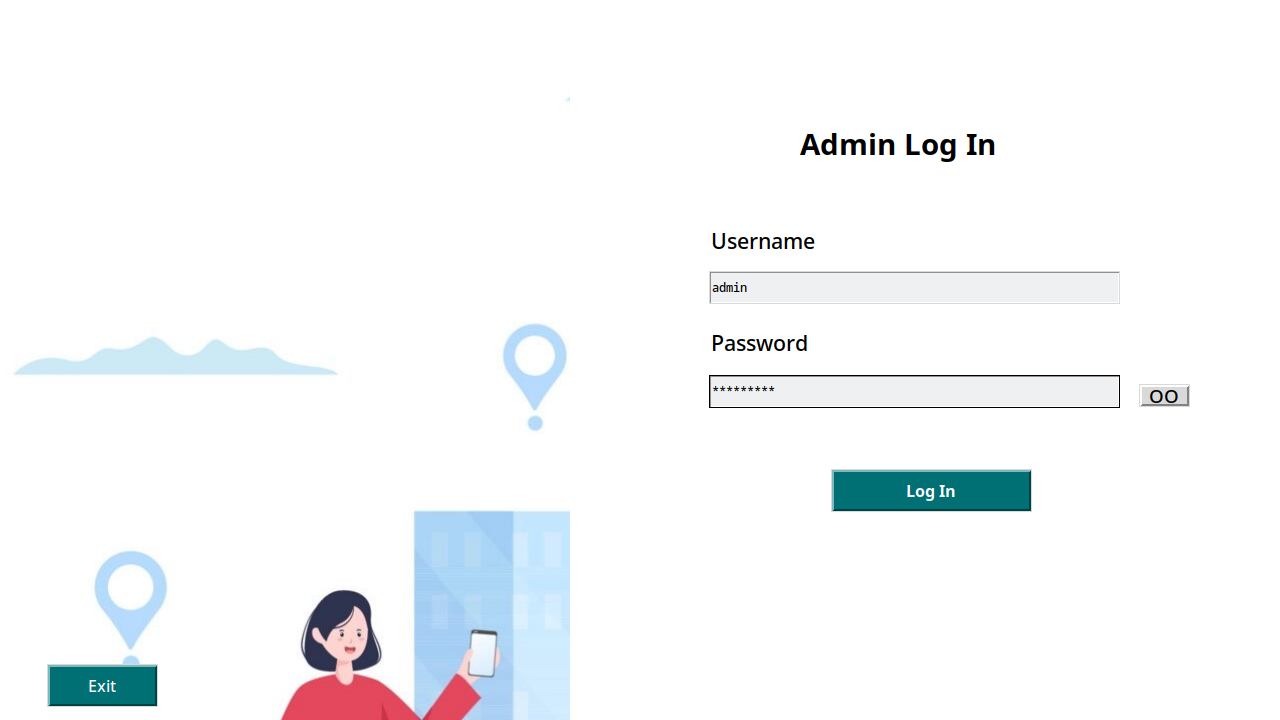
Valid login information enables user dashboard redirection.

Figure 30: user\_dashboard

Figure 31: profile\_page

Users may access their profile pages, which include personal information and information about planned trips.

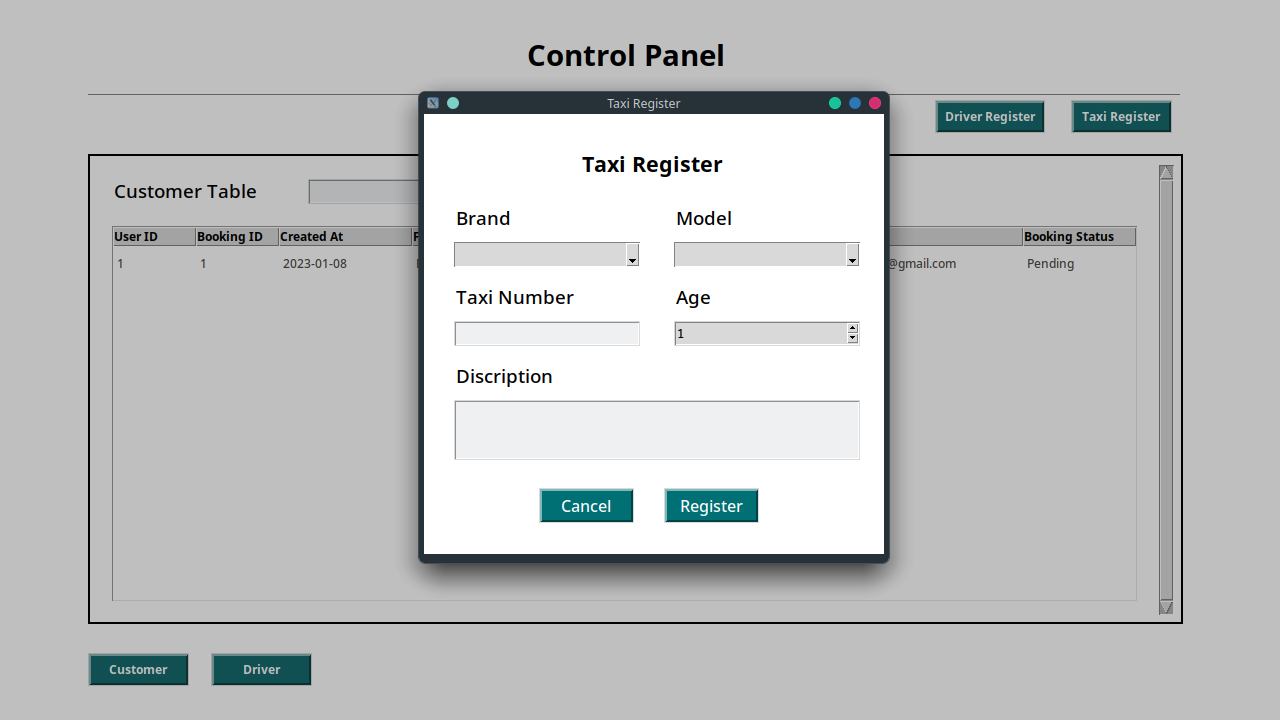
Admin can redirect to the Admin dashboard using valid login information.

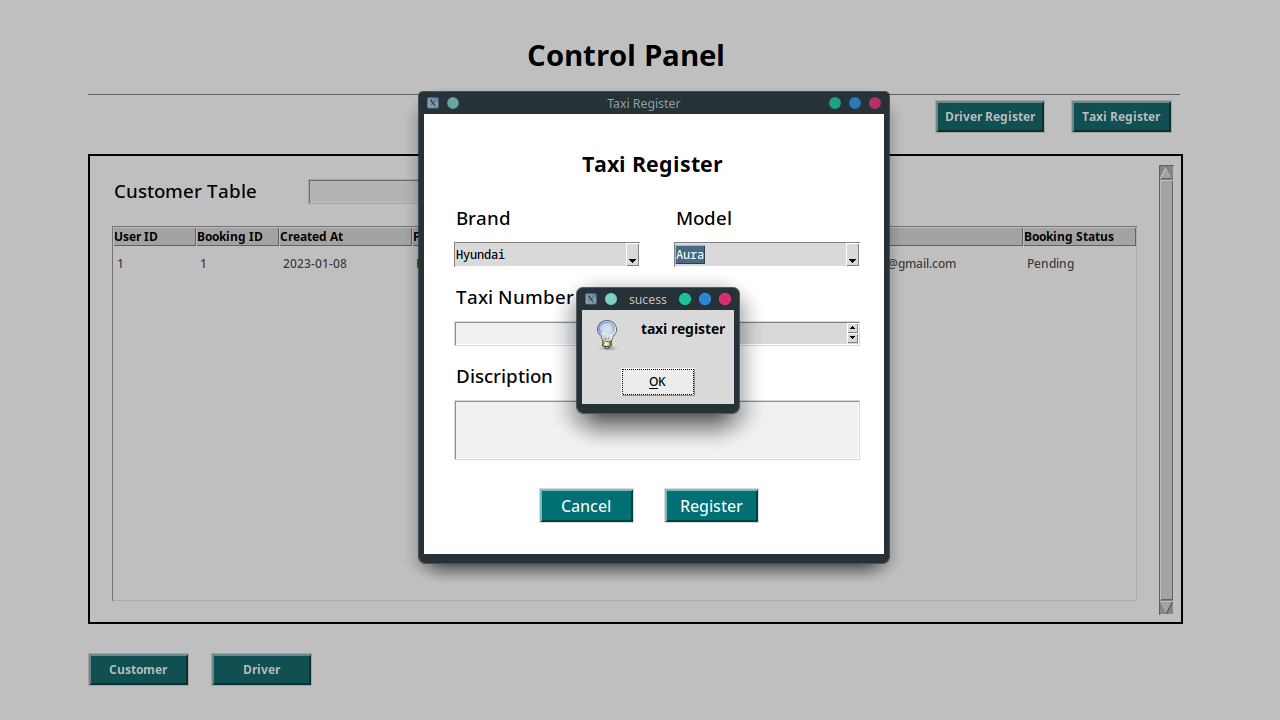
Figure 32: admin\_login

Admin dashboard with search and data filter functionality for seeing booking request information.

Figure 33: admin\_dashboard

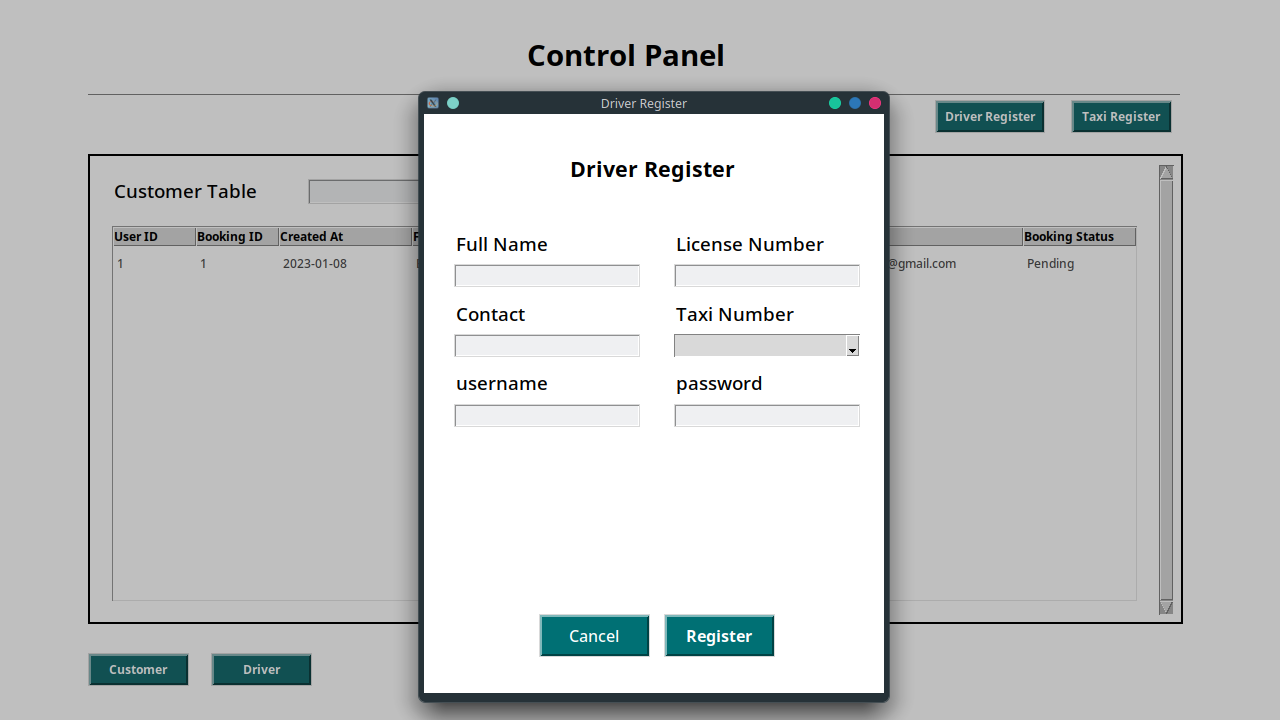
Additionally, the admin dashboard enables admin to register new taxis with relevant information.

Figure 34: register\_taxi

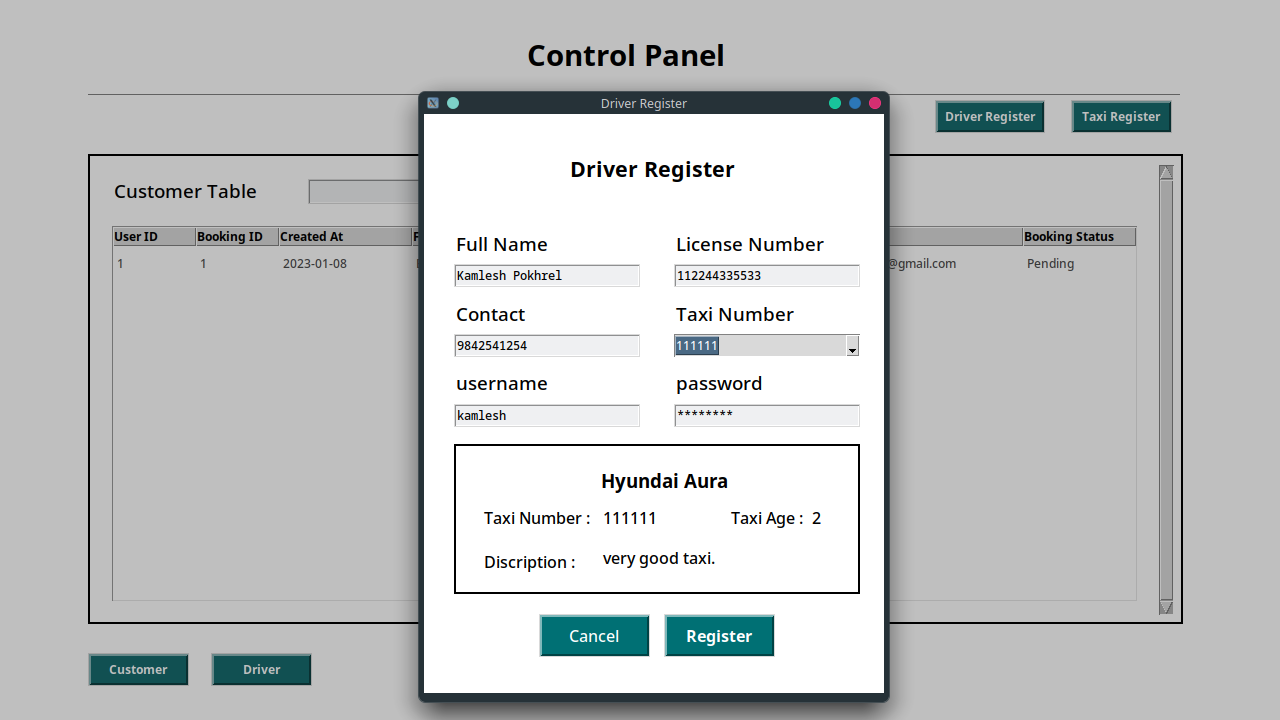
Figure 35: after\_clicking\_on\_taxi\_register

After taxi is registered message box pop up and inserted data get deleted but the toplevel remain there allowiing admin to add more taxi detail without extra clicking work.

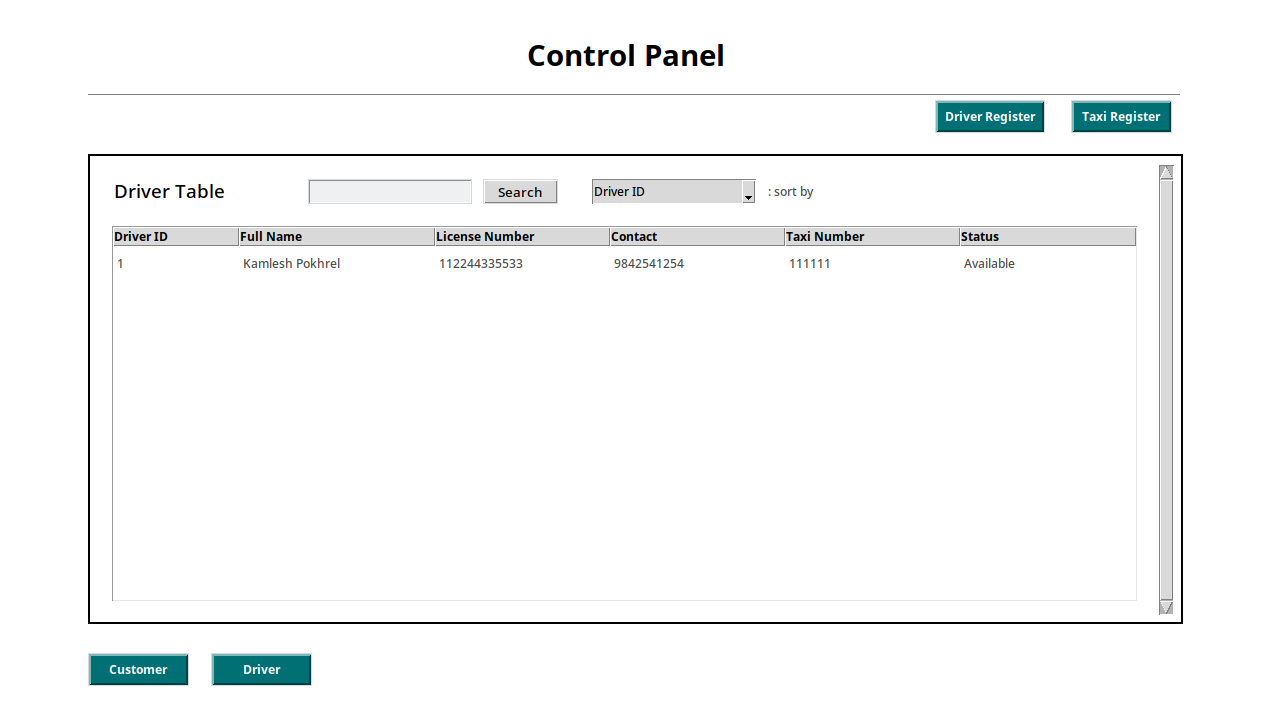
A new driver's personal information can be registered on the admin dashboard as well.

Figure 36: driver\_register

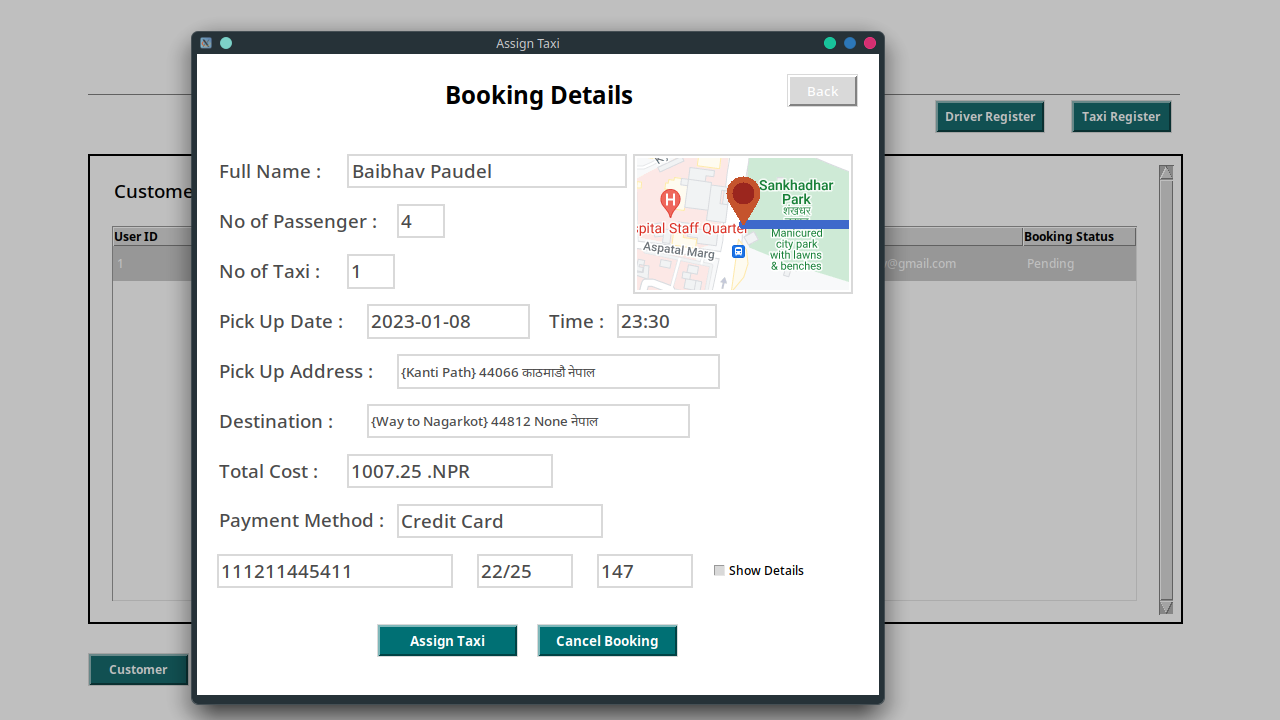
Admin must select a taxi number to provide to the driver of the taxi by using a tkinter combobox, which displays all the available taxi numbers to assign to the driver. After selecting on a taxi number, the details of the taxi are displayed in the frame below.

Figure 37: driver\_register\_with\_data

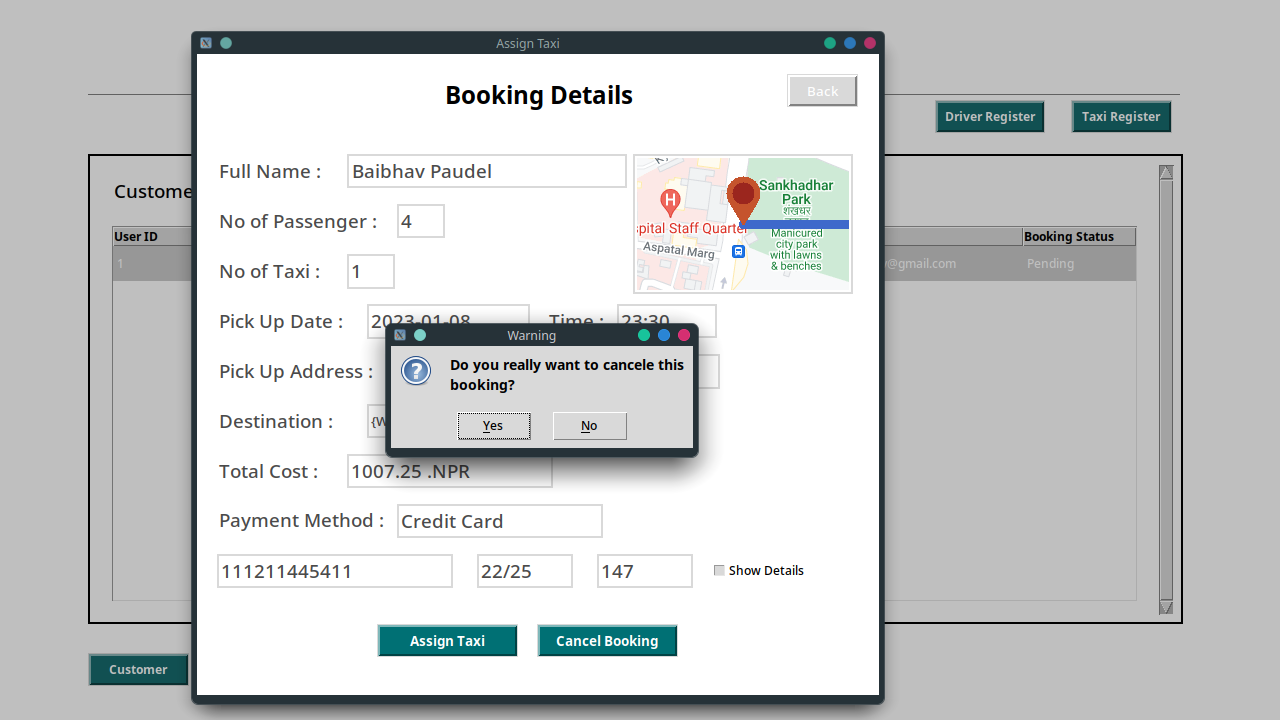
Administrators may view registered drivers and their status on the driver table after registering them.

Figure 38: after\_driver\_is\_registered

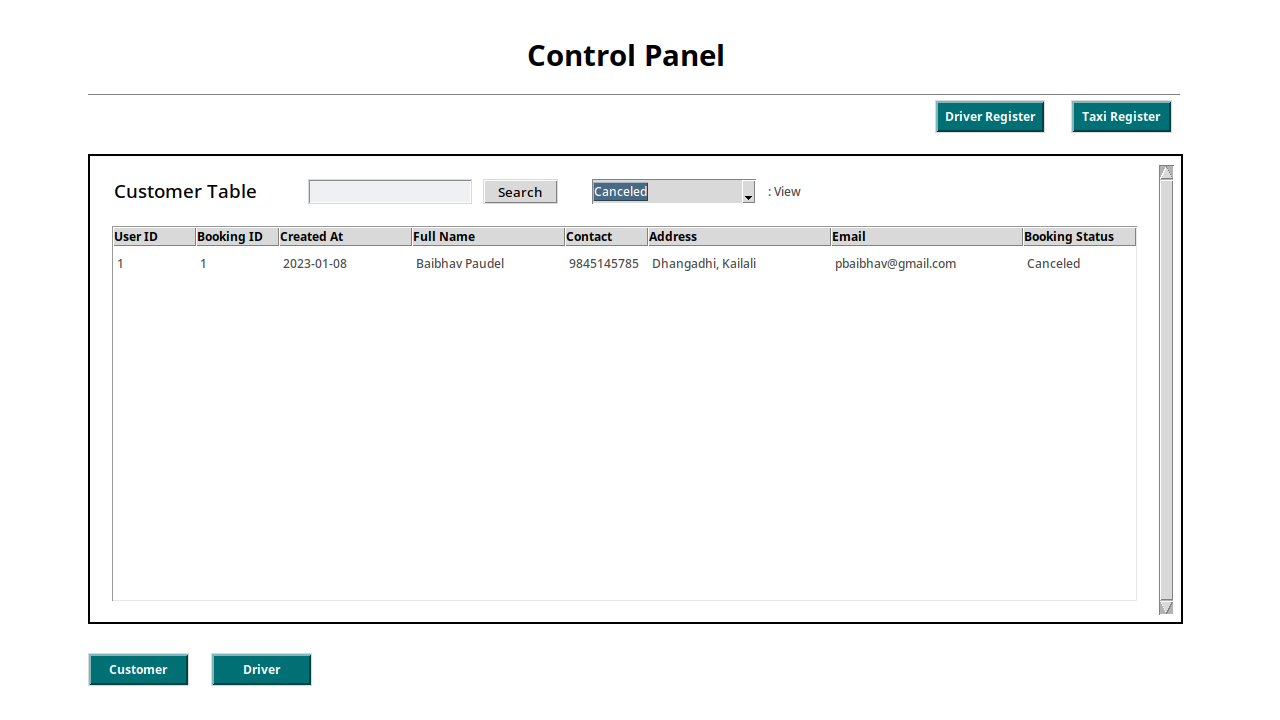
Double clicking on the top level of the booking request will bring up a pop-up with all the booking details, a minimap showing the pickup and drop-off locations, and a button to assign or cancel the service.

Figure 39: double\_clicking\_on\_booking\_request

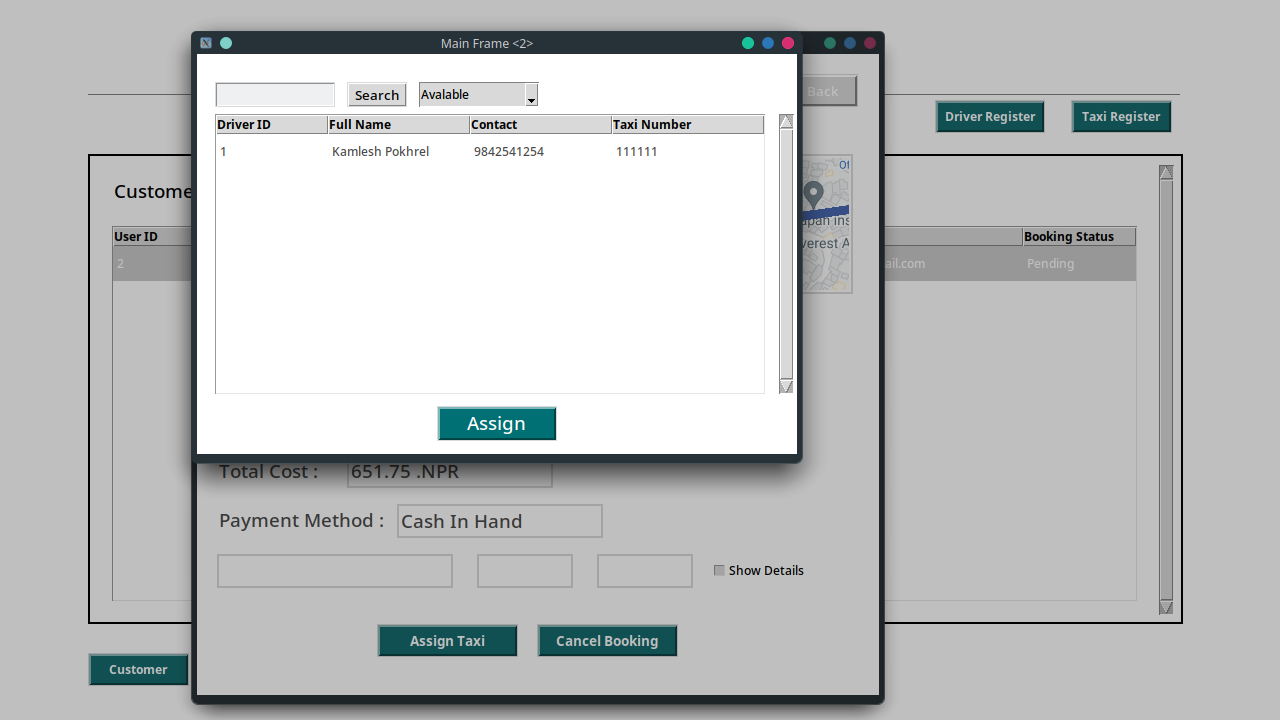
A message box with a security warning appears when a booking request is canceled.

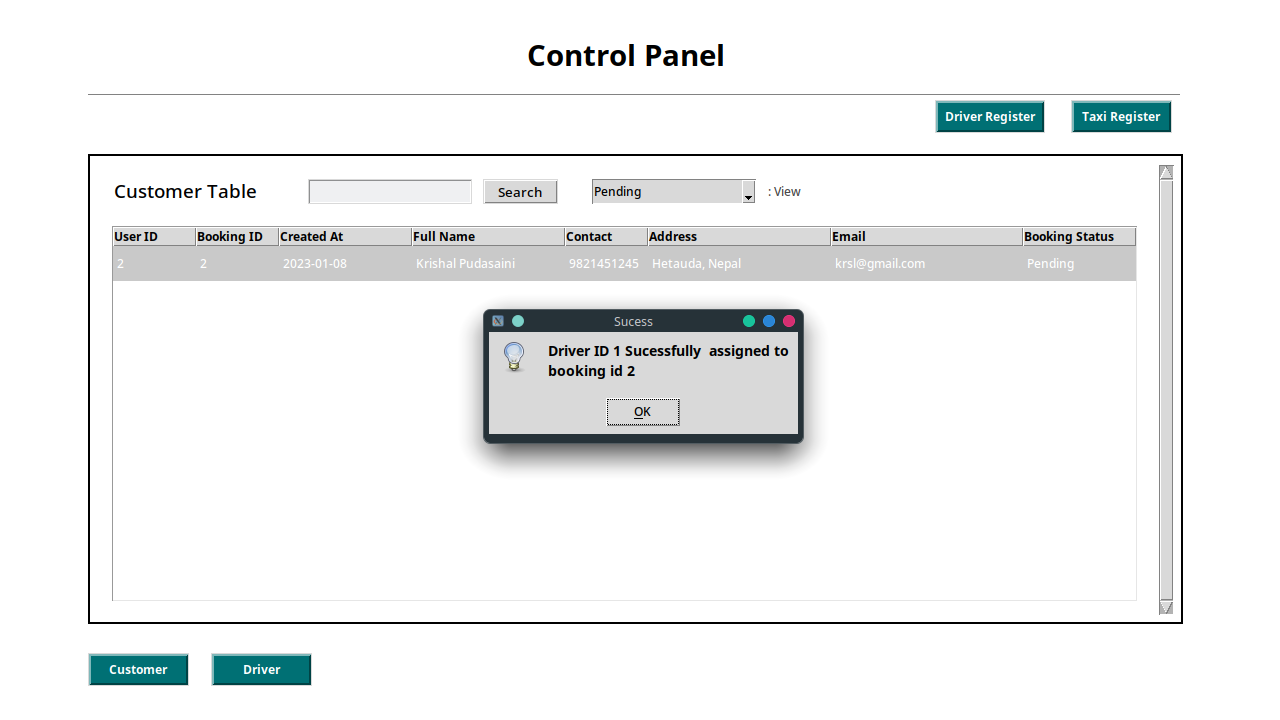
Figure 40: after\_clicking\_on\_cancel\_booking

When a booking request is canceled, it can be kept track of by using the canceled filter, which displays status as canceled.

Figure 41: after\_booking\_is\_canceled

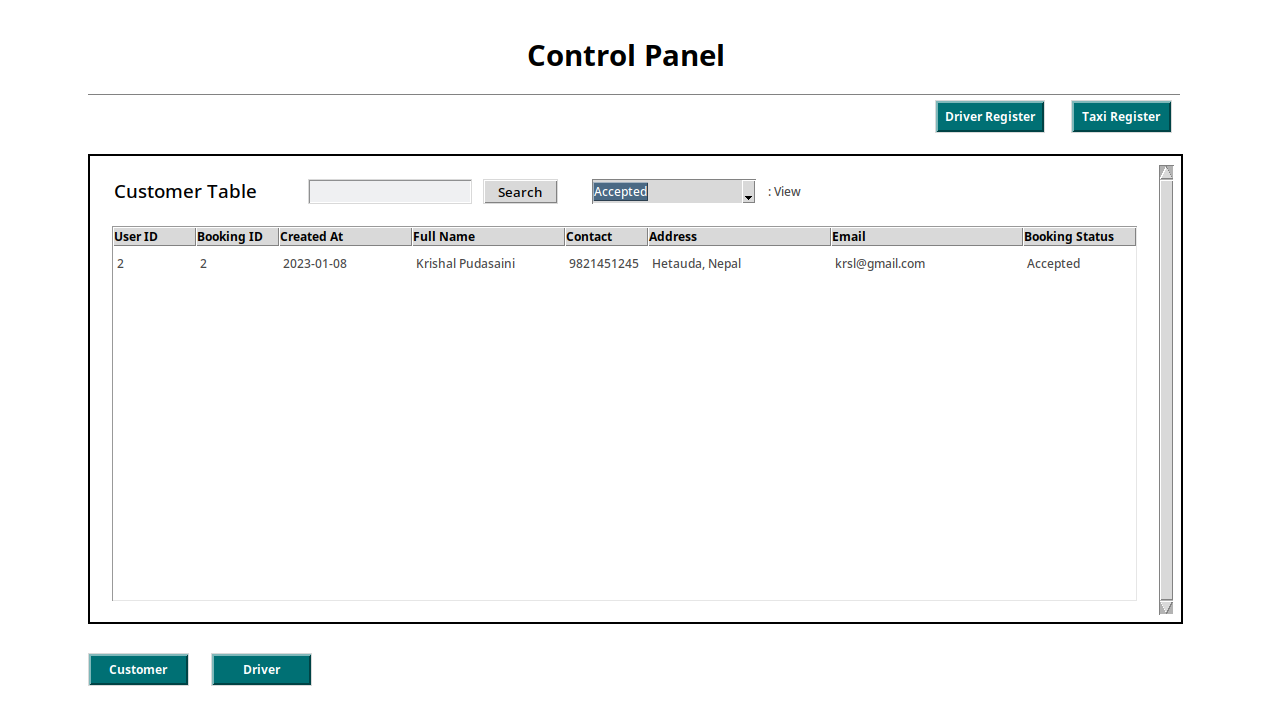
A new top level with a list of all available taxi drivers that may be assigned to the booking request appears when the assign taxi button is pressed.

Figure 42: after\_clicking\_on\_assign\_button

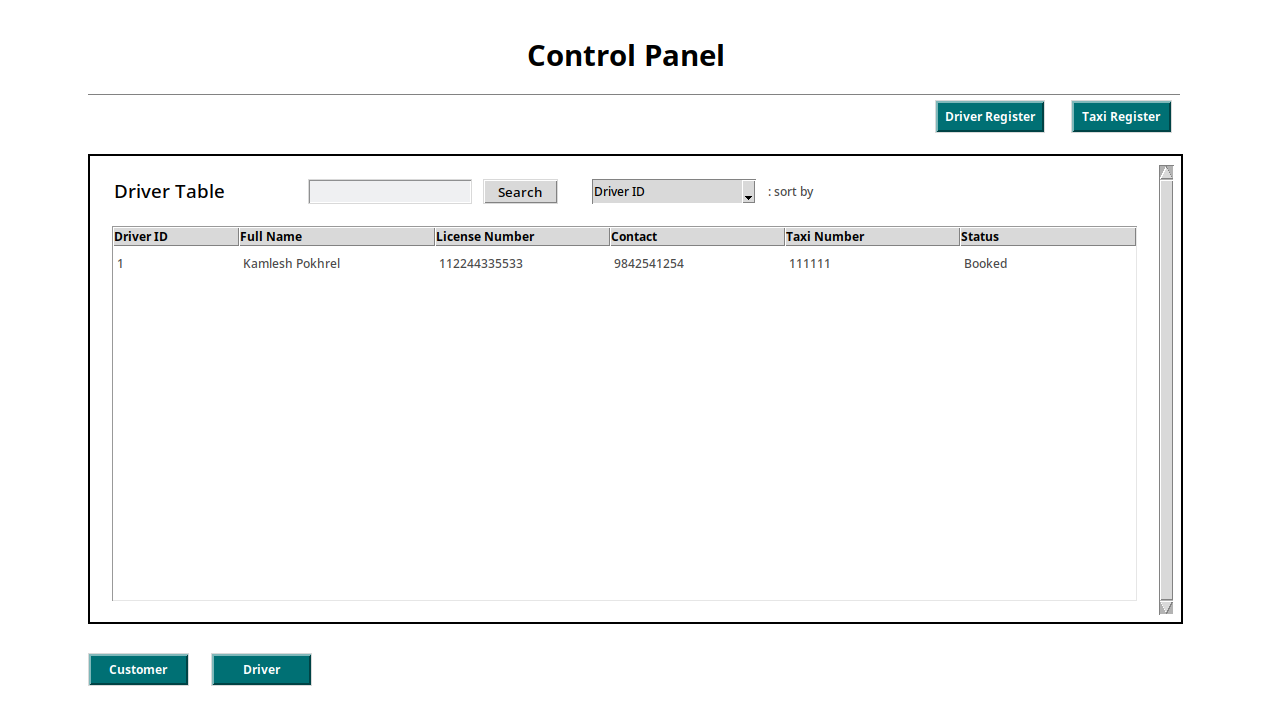
Figure 43: after\_clicking\_on\_register

A success message box appears with the necessary information after selecting the driver to assign to the booking.

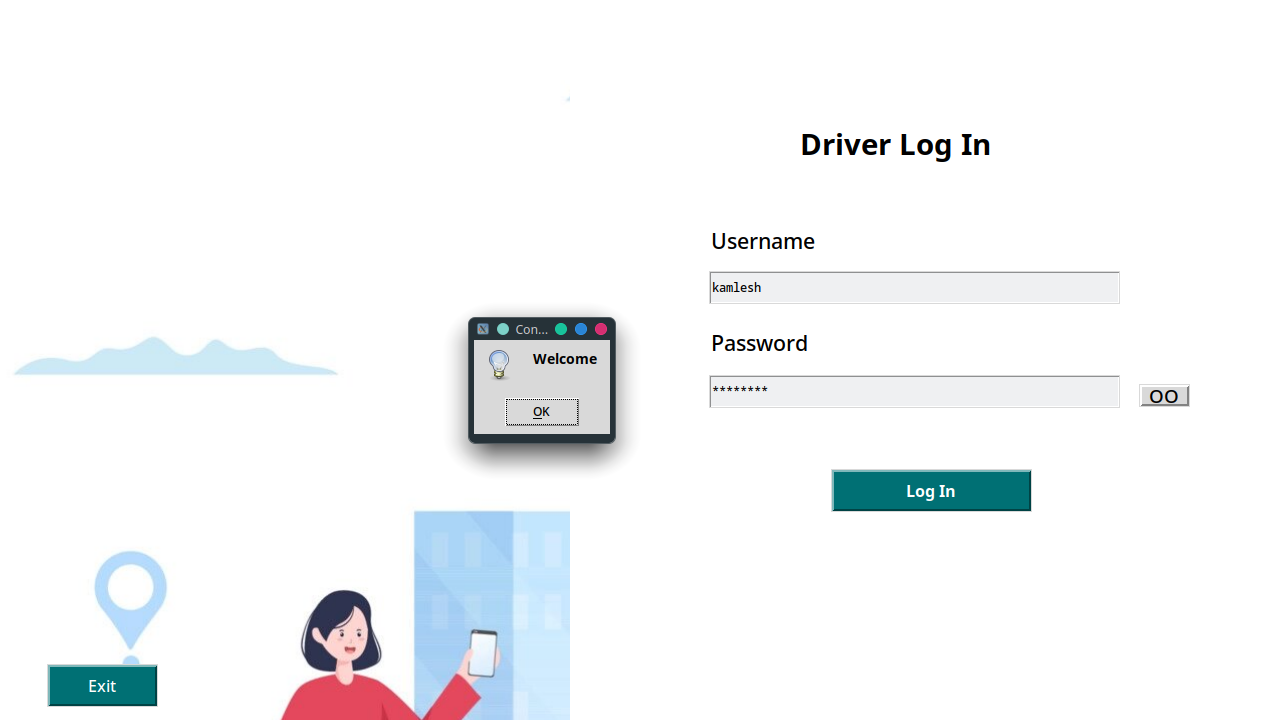
The booking request status changes to accepted when the taxi is assigned, and the Accepted filter may be used to view it.

Figure 44: booking\_detail\_after\_booking\_is\_accepted

Driver status changes to booked once a driver is allocated to a booking in order to prevent the driver from being assigned to any other booking requests until the trip is over.

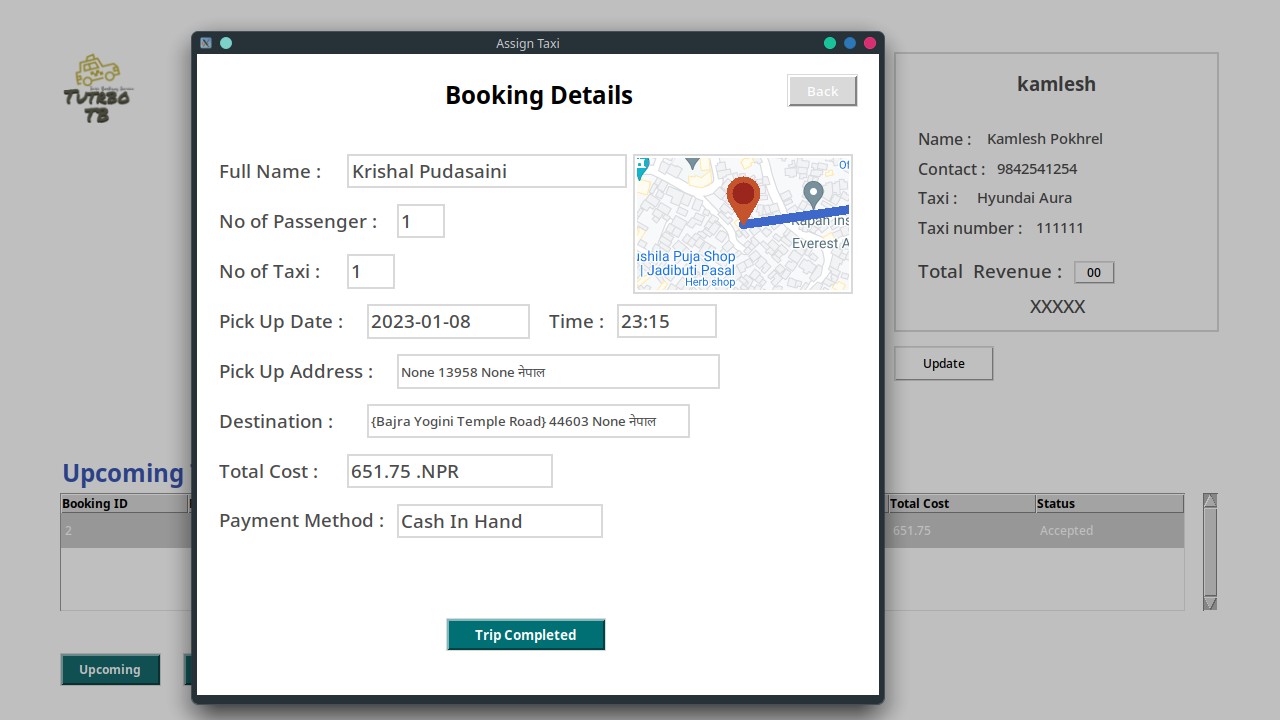
Figure 45: driver\_detail\_after\_booking\_is\_accepted

Driver can access the driver dashboard by using the admin's provided valid login information.

Figure 46: driver\_login\_page

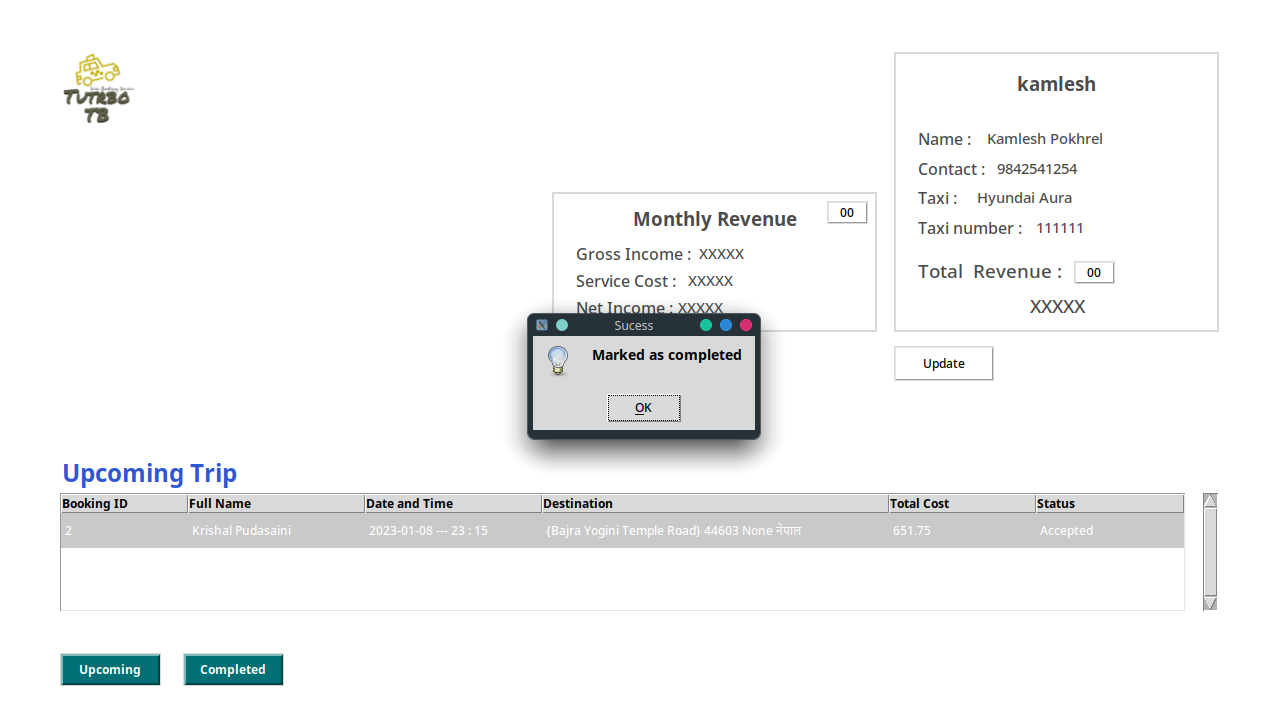
Driver dashboards enable drivers to see the trips that have been allocated to them in the near future.

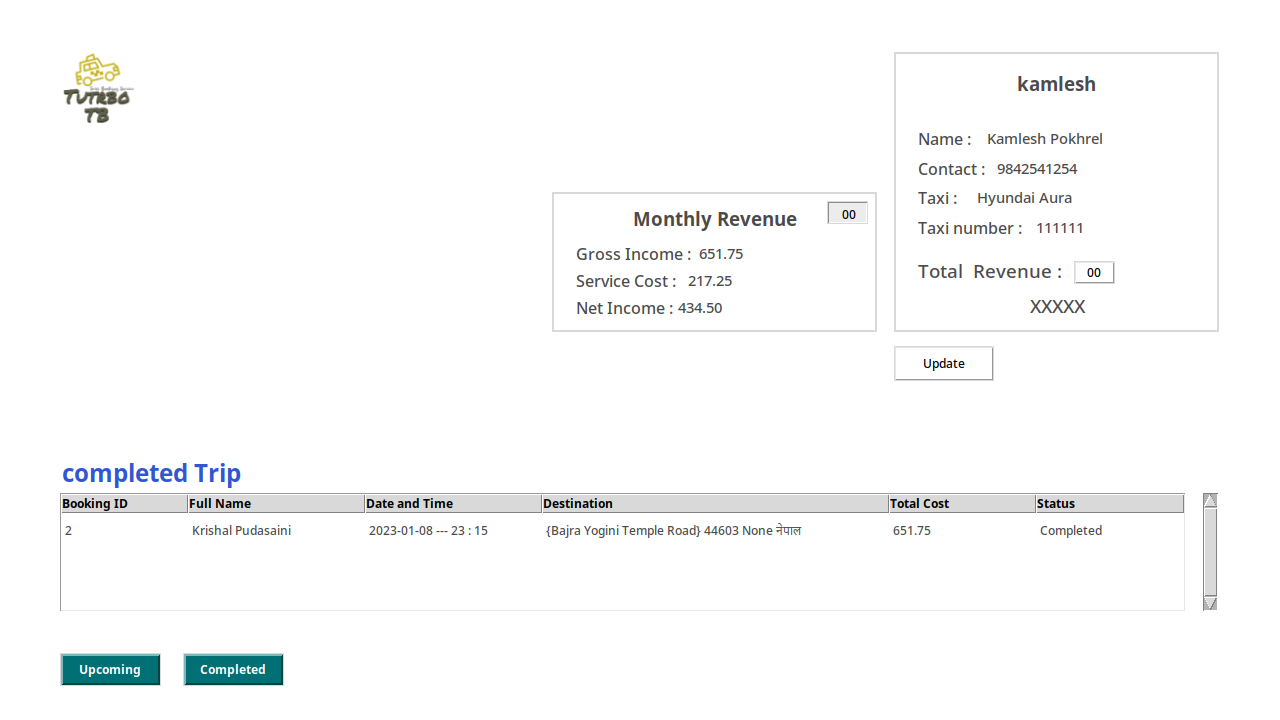
Figure 47: driver\_dashboard

Figure 48: driver\_assigned\_booking\_detail

When you double-click a booking request, the full details of the booking are displayed on the top level along with a trip finished button that enables you to declare a trip complete after the trip date has begun.

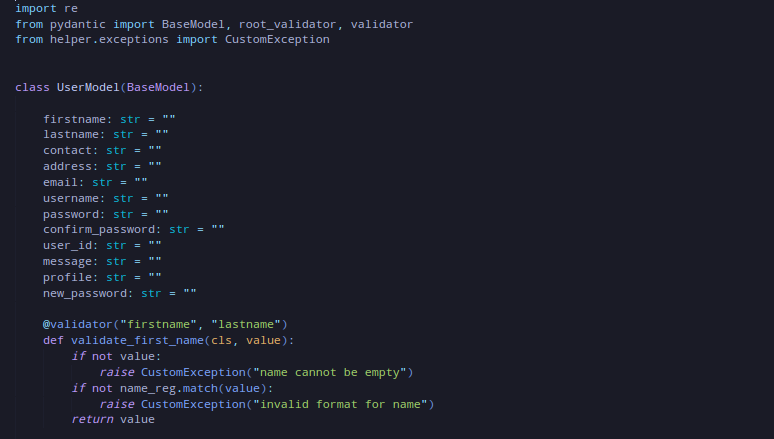
The system notifies the driver via a message box that the journey has been designated as completed after clicking on Trip completed.

Figure 49: Trip\_marked\_as\_completed

Figure 50: completed\_trip\_details

Completed trip money is contributed to total revenue as well as monthly revenue calculations that calculate gross income, service cost, and net income and are reset every month for fresh month data. Completed trip data is presented in the completed trip table.

Pydantic, a Python package that offers a set of tools for developing data validation and data parsing code, was utilized to tackle a problem that arose during the development process regarding verifying my user-given data before putting it in a database.

Figure 51: pydantic\_code\_snippets

A tile-based interactive map renderer widget for Python, TkinterMapView The Tkinter library was also utilized to resolve my other encounter issue, which included giving the system exact pickup and departure locations so that it could determine the trip's overall cost.

Figure 52: TkinterMapView\_code\_snippets

# Testing

Documentation of each test – one after the other

You need to include:

* Test No
* Test Date
* Purpose of test
* Input data or action
* Expected result
* Actual result
* Action if the above are not the same
* Screenshot of output

# Discussion / Reflection / Critical Analysis

As an individual working on a taxi booking software project, here are some things that have gone well or wrong during the development process:

What went well:

* In order to achieve the project's objectives, I was able to describe them precisely and lay out a plan.
* I was able to effectively research and learn any new technologies or tools necessary for the project.
* I was successful in keeping progress on schedule and meeting project deadlines.

What went wrong:

* I struggled with feeling overwhelmed or burnt out while working on the project alone.
* I had difficulty finding the necessary resources to complete the project.
* You may have encountered technical challenges that were difficult to resolve on your own.

As an individual, I had to cultivate strong time management skills throughout the development process. I had to be mindful of deadlines and ensure that I was making steady progress towards completing my tasks on schedule. I used tools such as project management software to help me track my progress and identify any potential bottlenecks or delays.

Basically, you are evaluating what you have done. Did you achieve all the required specification of the Assignment brief? If not, why not? Discuss the problems you encountered and how you overcame them.

What stopped you achieving what you had hoped to achieve and to the standard you know you are capable of achieving? Remote learning, lack of access to BREO, COVID, lack of suitable devices remotely, Internet connection issues etc.

Did you give yourselves enough time to accomplish this work? Was it harder than you expected? What have you learned from tackling this Assignment?

If working as a group, did your group work well together? What did you learn about the dynamics of group work?

Did this work enhance your understanding of the Python programming language? Did it motivate you to learn more Python, and other programming languages?

If you had to undertake this or something similar again would you tackle it differently?

What improvements or enhancements would you recommend for a future iteration of this project?

# Conclusion

What were you asked to do? Did you achieve it?

# References

Use the Harvard Referencing System. Any reference must be cited in text -otherwise it should go in a Bibliography section.

# Appendix

Make sure that all content has a Fig. No., caption and explanation.

You MUST include:

Complete project code.

Include file/class names.

Make sure you code is fully commented and well presented with correct indentation and colour coding.

You could include:

Details of you database/table structure and content (from PHPMyAdmin / SQLite Studio)

Screen shots of your GUI or text-based interfaces

Anything else that you might want to include which does not fit well into the main report body.