

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

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

**Assignment 2 – Group / Individual Project – Case Study**

**(Taxi Booking System)**

**Student Name: Nitesh Hamal**

**University ID: 2126474**

Table of Contents

List of Group Members (if Group Work)………………………………………………………5

Introduction / Overview………………………………………………………………………… 5

Task Description………………………………………………………………………………... 5

Project Plan / Schedule….

Tasks:

Requirements Analysis

Overview of Functional, Technical (Non-Functional) and Usability Requirements

Design:

* UML Diagrams
* Use Case Diagrams
  + Use Case Specifications/Description
  + Use Case Scenario
* Activity Diagrams
* Class Diagram
* Database Design
* Logical Database Design
* Entity Relationship Model
  + ERM Diagram
  + List of Entities
* Physical Database Design (including Data Dictionary)
  + Skeleton Tables
  + Data Dictionary
* User Interface Design

Implementation

Testing

List of Group Members – Roles and Responsibilities (if Group Work)

Discussion / Reflection / Critical Analysis

Conclusion

References

Appendix

* Complete code with comments and class / filenames.

You will need to complete the Table of Contents and Page Nos manually

PLEASE NOTE THAT THIS IS A PROFESSIONAL REPORT

ALL SCREENSHOTS, DIAGRAMS, etc SHOULD BE GIVEN A FIGURE NUMBER, A CAPTION AND AN EXPLANATION

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

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

# University ID and Name

University ID 2126474 Full name Nitesh Hamal

# Introduction/Overview

The university assigned students the task of developing a system that would allow taxis to be booked online. Taxi booking systems are online portals where customers may order cabs. Drivers can register and bid on jobs using several taxi booking platforms. They link passengers with drivers by supplying available cabs via the phone or over the internet. People who are seeking for a private taxi service or do not have access to one in their region typically utilize these services. Prior to beginning the implementation process, it is essential that you conduct a requirement analysis. It is significant to note that the taxi booking system has three different types of users: the customer, the driver, and the administrator. As a prelude to the development process, a Class Diagram, Use Case Diagram, and Activity Diagram are prepared in order to visualize the operation mechanism of the project. A Relational Database Management System (RDMS) is constructed based on an Entity-Relationship Model (ERM) that structures the relationship between the entities in the database. A taxi booking system has been built using the PyCharm IDE. It is an IDE specifically designed for software development, which provides Python libraries for writing Python programs, and it presents the idea of Object-Oriented Programming in the process of development. It is built with the Python GUI toolkit, Tkinter, in order to give it a professional look and feel. A database is used in order to store data related to taxi bookings that are stored in the database using MySQL. Completing a taxi booking system is a difficult process. A difficulty is unavoidable. Solving such challenges and creating a fantastic application is possible with the assistance of teachers, friends, and the internet.

# Task Description

The task is about a taxi company's taxi booking system. A taxi firm would want to provide an online service for the convenience of individuals traveling about, as well as for drivers to monitor their forthcoming journeys. Create a graphical user interface (GUI) for the online booking system that includes a login page for users who have already registered, a registration page for new users, and a page for drivers. Put in place a method for keeping driver and customer information in an external file (e.g., a text file or database). Fields for saving client information, such as names, addresses, phone numbers, and payment methods, as well as travel details, should be included (pick up and drop off address, and pickup time and date). Create a system that will allow you to assign one specific driver to each trip while also storing driver information (name and license number). Set up a system that enables logged-in users to schedule trips via the GUI, including entering the pickup and drop-off addresses as well as the pickup time and date. Create a system that will enable clients to see their planned travel plans and, if necessary, cancel them. Create a system that enables the manager of the taxi business to confirm fresh trip reservations and assign a driver to each confirmed reservation. Create a system that enables taxi drivers to sign in and view a list of their scheduled rides. To check that the online reservation system is functional and complies with taxi business specifications, test it out.

# Project Plan/Schedule

|  |  |  |
| --- | --- | --- |
| Week No. | Tasks | Priority |
| 7 | Case Study  System Analysis  Use Case-V1  UI/UX Design-V1  Activity Diagram-V1 | MUST |
| 8 | Database Connectivity  ER-Diagram-V1  CRUD Version-1  OOP Implementation-V1  Class Diagram-V1 | MUST |
| 9 | Use Case-V2  UI/UX Design-V2  Activity Diagram-V2  Class Diagram-V2  ER-Diagram-V2  OOP Implementation-V2  CRUD Version-2 | MUST |
| 10 | **Final Version**  Use Case  UI/UX Design  Activity Diagram  Class Diagram  ER-Diagram  OOP Implementation  CRUD Implementation  Final Report-V1  Final App-V1 | MUST |
| 11 | **Demo Demonstration and Presentation**  Final Report- Final Version  Final App- Final Version | MUST |
|  |  |  |
|  |  |  |
|  | Submit Group Report, Project Code and Video Recording (if required) |  |
|  | Project Presentation (if required) |  |

# Requirements Analysis

## Functional Requirements

TBS = Taxi Booking System

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | A customer must be able to register on the TBS | MUST |
| 2 | A customer must be able to log in to the TBS | MUST |
| 3 | A customer must be able to log out of the TBS | COULD |
| 4 | An administrator must be able log in to the TBS | SHOULD |
| 5 | An administrator must be able to log out of the TBS | COULD |
| 6 | An administrator must be able to confirm a booking | SHOULD |
| 7 | An administrator must be able to allocate a driver to a booking | SHOULD |
| 8 | An administrator must be able to view all bookings | SHOULD |
| 9 | An administrator must be able to cancel a booking | SHOULD |
| 10 | A customer must be able to view his/her bookings | MUST |
| 11 | A customer must be able to make a booking | MUST |
| 12 | A customer must be able to cancel a booking | MUST |
| 13 | A taxi driver must be able to log in to the TBS | MUST |
| 14 | An administrator must be able to log out of the TBS | SHOULD |
| 15 | A taxi driver must be able to view his bookings | MUST |

## Non-functional Requirements

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | The TBS should process input and return results within 10 seconds |  |
| 2 | The TBS should run on a range of devices from PCs to mobile phones |  |
| 3 | The TBS design should be sufficiently scalable and flexible to allow for further future enhancements |  |
| 4 | The TBS users should not experience critical system failures. 99.99% ‘uptime’ should be achieved. |  |

## Usability Requirements

|  |  |  |
| --- | --- | --- |
| Req. No | Requirement | Priority\* |
| 1 | The TBS should incorporate a user-centric design |  |
| 2 | The design should demonstrate evidence of a good understanding of interface design issues – for example, a consistent design for each form, layout of content, use of colour schemes and images, navigational methods, usability when viewed at various screen resolutions and various monitor dimensions. |  |
| 3 | All data entry forms should be short and easy to complete and there should be entry validation. |  |
| 4 | The TBS should have clear and intuitive navigation |  |
| 5 | The TBS should comply with WW3 Web Accessibility Standards (WCAG)  Text easy to read and language and language style should be appropriate with absence of grammar / spelling errors |  |
|  | There should be a clear layout which remains consistent throughout the application. Style, layout and content should be appropriate for the purpose of the application. |  |

\*MOSCOW Notation:

M = MUST

S = SHOULD

C = COULD

W = WON’T

# Design

## UML Diagrams

### Uses Case Diagrams(s)

Diagram

Description automatically generated

Diagram

Description automatically generated

A picture containing text, sky, map

Description automatically generated

Fig 1: Taxi Booking System - Use Case Diagram (Sea Level)

#### Use Case Specifications/Description

#### Use Case Scenario

Allocate Accommodation

A specific room is allocated to each guest.

Change Booking

An unguaranteed booking may become guaranteed on provision of credit card details. Other changes (for example to the composition of the party or to the dates booked) may be made on the customer’s request.

Calculate Bill Total

The total bill for a room comprises the cost of the room plus any items charged to it (for example, from the minibar, hotel bar or restaurant and any additional services provided) plus the appropriate rate of VAT.

Cancel Booking

A customer may cancel a booking any time before checking in. The booked accommodation is de-allocated.

Charge Item to Room

A guest may use items from the minibar in the room, make external telephone calls, make purchases from the hotel bar or restaurant or order additional services such as laundry. These items are charged to the guest’s room. An external service ordered by the guest may also be charged to their room (although some services, such as taxis, will be paid for directly by the guest).

Check In Guest

When a party (one or more guests) arrives at the hotel each guest is checked in and the booking becomes registered. Details of all the guests are confirmed, or if they have not been given at the time of booking they are recorded. The method of payment (by which guest(s) or whether to be charged to a corporate account) is confirmed. Accommodation is allocated if not already done at the time of booking and special requests are recorded.

Check Out Guest

Before the guest leaves the hotel, they need to check out and the booking becomes completed. The receptionist provides them with the total room bill, which an individual customer must pay. If the bill is to be charged to a corporate customer the guest is asked to check and sign the bill, it is then added to the appropriate corporate account.

Make a Booking

When a customer contacts the receptionist with a booking request the receptionist checks the room occupancy for the required period and if the customer's needs can be met, makes a booking.

A guaranteed booking is only made for corporate customers or for individual customers who supply credit card details.

Accommodation may be allocated at this stage; alternatively where it is a group booking (for example, for 6 twin-bedded rooms) the allocation of each guest in the party to a specific room may be done on checking in. Either way, occupancy records are updated to avoid double booking.

Order External Service

Services not offered by the hotel itself can be ordered, once their availability has been checked. The guest name, room number and details of the service required are taken when the service is ordered.

Provide Guest with Current Bill

On request a guest may be provided with the current total of their bill.

### Activity Diagram(s)

### Class Diagram(s)



Classes (omitting methods / operations)

* + admin (adminid, fullname, address, email, password)
  + booking (bookingid, pickup\_address, drop\_address, pickup\_date, pickup\_time, status, cid, did)
  + customer (cid, fullname, address, email, , number, password, payment)
  + driver (did, fullname, address, email, licenseno, status, password)

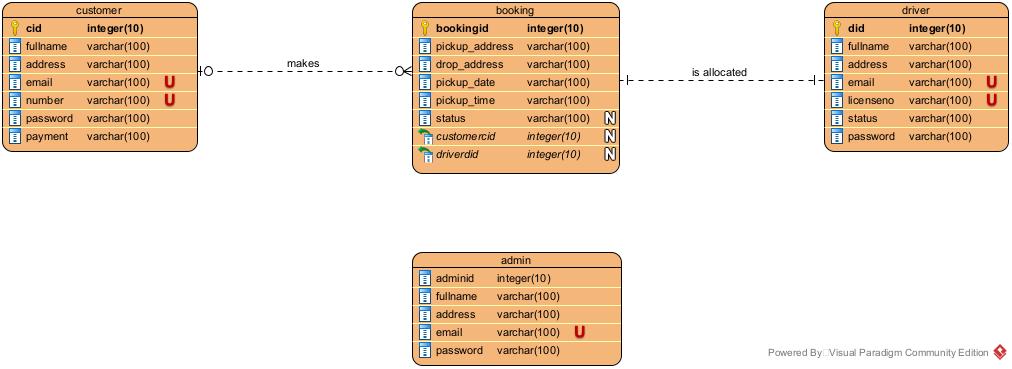
## Database Design

### Entity Relationship Model (ERM)

#### ERM Diagram

Diagram

Description automatically generated



#### List of Entities

customer (**cid**, fullname, address, email, number, password,payment)

booking (**bookingid**, cid\*, did\*, pickup\_address, drop\_address, pickup\_date, pickup\_time, status)

driver (**did**, fullname, address, email, licenseno, status, password)

admin (**adminid**, fullname, address, email, password)

Primary Key = Bold and Underlined

Foreign Key = \*

### Physical Database Design

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

customer (**cid**, fullname, address, email, number, password,payment)

booking (**bookingid**, cid\*, did\*, pickup\_address, drop\_address, pickup\_date, pickup\_time, status)

driver (**did**, fullname, address, email, licenseno, status, password)

admin (**adminid**, fullname, address, email, password)

Primary Key = Bold and Underlined

Foreign Key = \*

#### Data Dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Customer | | | | | | | |
| Description: Customer details | | | | | | | |
| Field Name | **Datatype** | **Length** | **Index** | **Null** | **Default** | **Validation rule** | **Description** |
| customerid *(Primary)* | int (10)  unsigned | 10 | PK | No |  |  | Autoincremented Uniquely identifies every customer |
| title | varchar (10) | 10 |  | No |  |  | Title of customer |
| firstname | varchar (20) | 20 |  | No |  |  | First name of customer |
| lastname | varchar (30) | 30 |  | No |  |  | Last name of customer |
| email | varchar (50) | 50 |  | No |  | Must be email format containing an @ and a ‘.’  Regex expression used | Email of customer |
| telno | varchar (20) | 20 |  | No |  |  | Telephone number of customer |
| password | varchar (20) | 20 |  | No |  |  | TBS password |
| address1 | varchar (30) | 30 |  | No |  |  | First line of address |
| town | varchar (30) | 30 |  | No |  |  | Customer town |
| county | varchar (30) | 30 |  | No |  |  | Customer county |
| postcode | varchar (15) | 15 |  | No |  |  | Customer postcode |
| paymentmethod | varchar (20) | 20 |  | No |  |  | Customer payment method |

Indexes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Keyname** | **Type** | **Unique** | **Column** | **Null** |
| PRIMARY | BTREE | Yes | customerid | No |

## User Interface Design

1. **Sign in GUI**  
   This is the interface that the client, administrator, and driver will use to access their separate dashboards by entering their email address and password. If the client does not already have an account or is not registered, they must first go to the sign-up GUI page to register.  
   Graphical user interface

   Description automatically generated

Figure Sign in GUI

1. **Sign up GUI**  
   The consumer may submit their information, register themselves, and create an email and password for themselves using this interface.  
   

Figure Sign up GUI

1. **Customer Dashboard**  
   This is the Customer Dashboard; it opens when the customer puts on their login email and password in the Sign in page and clicks sign in. Here a customer can request a booking by entering the pickup address, drop address, pickup date, and pickup time in their respective fields. Here a customer can see their pending booking and also their booking history. Customer can view and update their profile, change their password, logout and exit the whole program.  
   Graphical user interface, application

   Description automatically generated

Figure Customer Dashboard

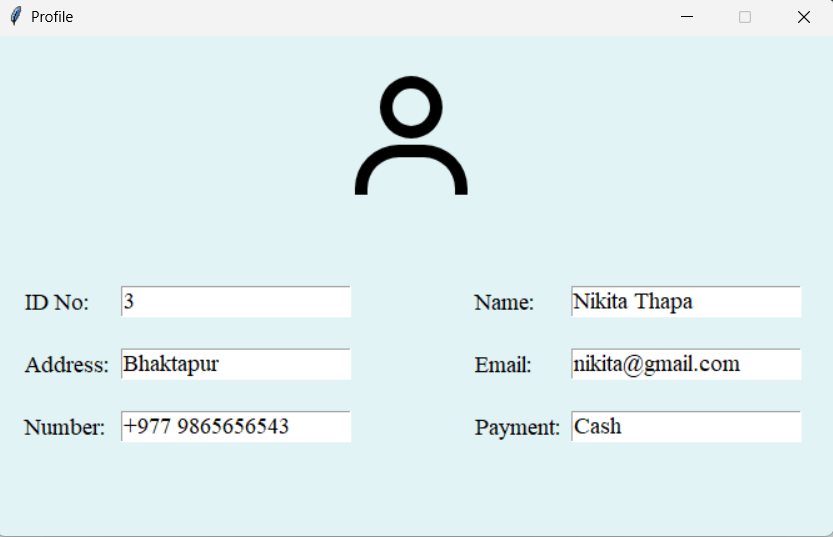
1. **My Profile GUI**  
   By clicking my profile button on customer dashboard, a customer opens the profile GUI to see their own information.  
   

Figure My Profile GUI

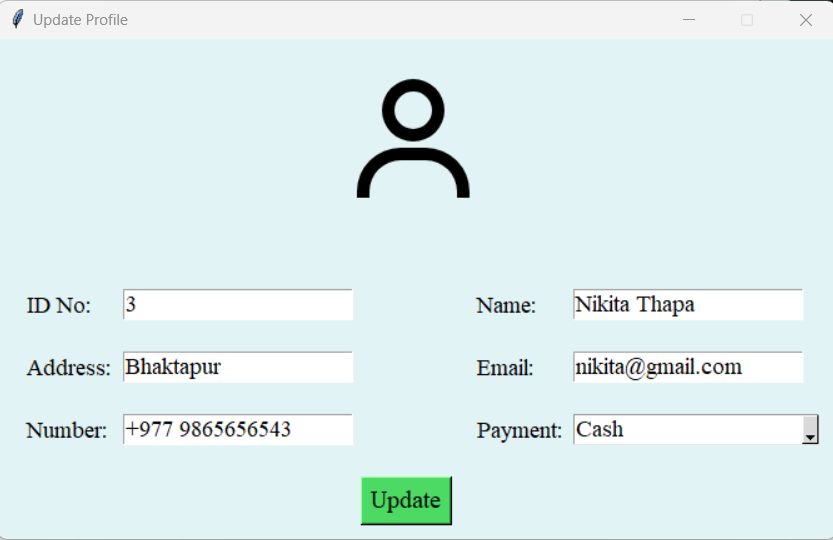
1. **Update Profile GUI**  
   By clicking update profile button on customer dashboard, a customer opens the update profile GUI where they can update their information.  
   

Figure Update Profile GUI

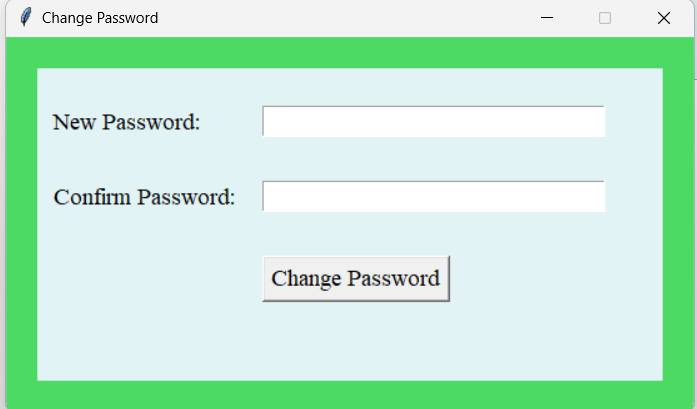
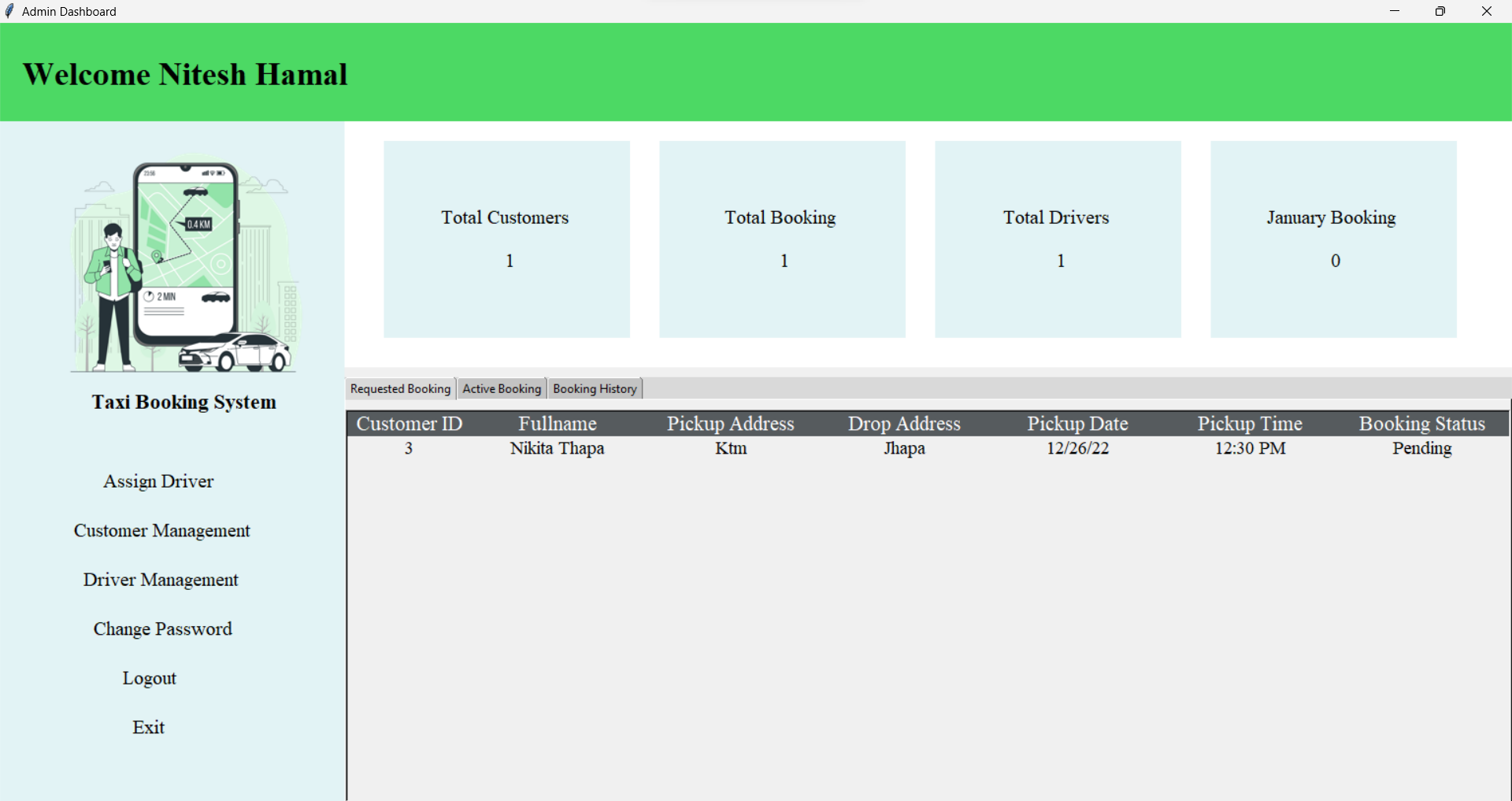
1. **Change password**  
   This is the password change interface; it is the same for all three users, customer, admin, and driver. They must input their new password twice. If the new password does not match the confirm password, an error will be generated, and they will be unable to change their passwords.  
   

Figure Change Password GUI

1. **Admin Dashboard**  
   The Admin Dashboard appears when the administrator enters their login email and password on the Sign in page and clicks Sign in. An administrator can assign a driver to the desired booking and confirm it here. The administrator may examine the overall number of clients, bookings, drivers, and total bookings for this month. He or she may also view all requested bookings, ongoing bookings, booking history, and completed bookings. They may also manage customers by clicking the customer management button, drivers by clicking the driver management button, and their own passwords by clicking the change password button. They can also logout and quit the system.  
   

# Implementation

The task was to use the Python programming language to develop a taxi booking system. First and foremost, I printed a copy of the assignment and conducted a case study on it. Based on what I had learned from the assignment, I then created a Use Case diagram. Following it, I created an activity diagram and a UI/UX design in accordance with the Clam level Use Case. Everything was completed in the first week. I began working on the database connectivity, entity relationship diagram, Object Oriented Programming Implementation, and Class diagram in the second week.

Following this, I discovered that I had to make some adjustments to my program, which required me to update some of the diagrams I had created. On the third week, I took that action. The OOP implementation and further CRUD improvements were made to my code over the same week. Moreover, I completed the final versions of the Use Case, UI/UX Design, Activity Diagram, Class Diagram, ER-Diagram, OOP Implementation, and CRUD Implementation during the fourth week. I worked on my report after finishing everything else, and I also started to add validation to the text field on my codes.

It was an individual project. I accomplished everything by myself. However, I rely on professors and friends for assistance.

**Python Programming Language**

Python is a widely used programming language that aids in the creation of applications by developers. The desktop application I created for this taxi booking system was written in the Python programming language. Python is a programming language that I found to be a little bit simpler than Java, which I had already studied in my previous semester. Different sorts of programming paradigms, including structured, functional, and object-oriented programming, are supported by the Python programming language. In order to create this taxi booking system, I employed object-oriented programming.

**PyCharm IDE and Libraries**

I selected PyCharm as the IDE because the teacher taught us Python using it and I had grown accustomed to it. This project was completed using the Tkinter Python framework. To ensure that my program works properly, I utilized supplementary libraries such as messagebox from tkinter, DateEntry from tkcalander, ImageTK from PIL, and many more.

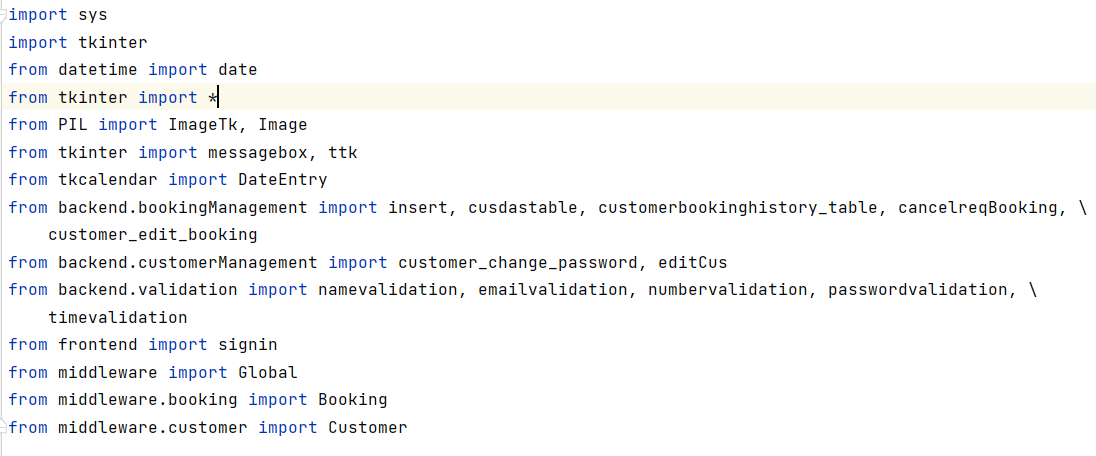


Figure Example of Libraries Used

**Client Server Architecture**

As more people use computers and rely on their networks for daily tasks, the number of computer users is increasing. The term "client" or "host" refers to a computer or other device that makes use of the service or receives the data. Client devices, which include laptops, workstations, and Internet of Things gadgets, are anything that can connect to a network. A server, which is a remote computer, offers access to information and services. Ordinarily, servers take the shape of rack servers or other physical items. The server controls numerous processes, such as hosting applications, controlling Internet connections, printing, and more. A client-server architecture is a configuration of a system that hosts, provides, and manages the majority of the resources and services that the client requests. I finished this project using a client-server architecture. I made use of the local host to store the information in the MySQL database.

**OOP (Object-Oriented Programming)**

The Object-Oriented Programming System (OOPs), which is built on the concepts of abstraction, encapsulation, inheritance, and polymorphism, is the most widely used programming methodology. Users are given the ability to create custom objects and methods for dealing with such items. Construction of objects and their subsequent manipulation to achieve desired outcomes is the main concept underpinning OOPs.

**Inheritance**

Inheritance is a technique for creating a new class that is a modified version of an existing one. The new class is referred to as the derived class, whereas the previous class is referred to as the base class. The characteristics and behaviours of the base class may be extended or overridden by the derived class. This is advantageous since it enables you to modify and reuse code from the base class for the derived class.

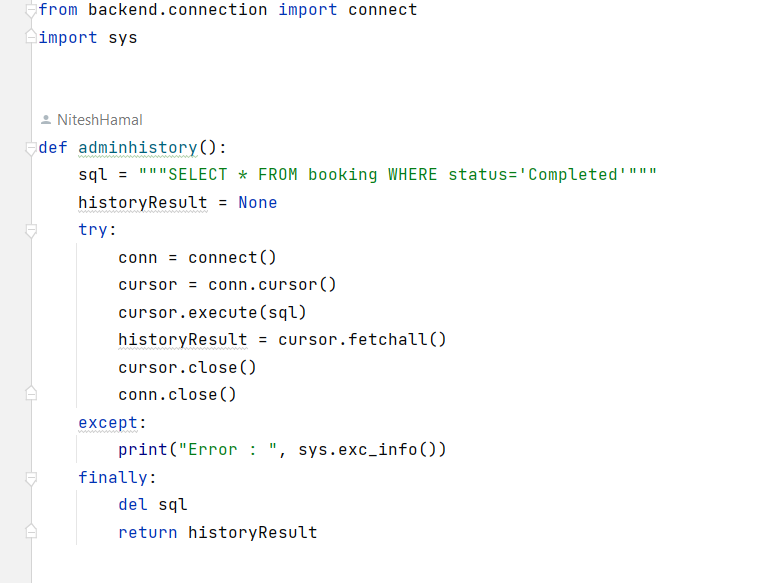


Figure Example of Inheritance

**Polymorphism**

One of the important OOPS ideas is polymorphism. You can have different or numerous kinds of objects, variables, or methods using polymorphism. Thanks to polymorphism, the same function might have several implementations depending on the requirements of the class. It can also be said as parameters with default value. The ability to build flexible code that can handle several object types in a consistent manner is made possible by polymorphism, which is a useful characteristic of object-oriented programming.

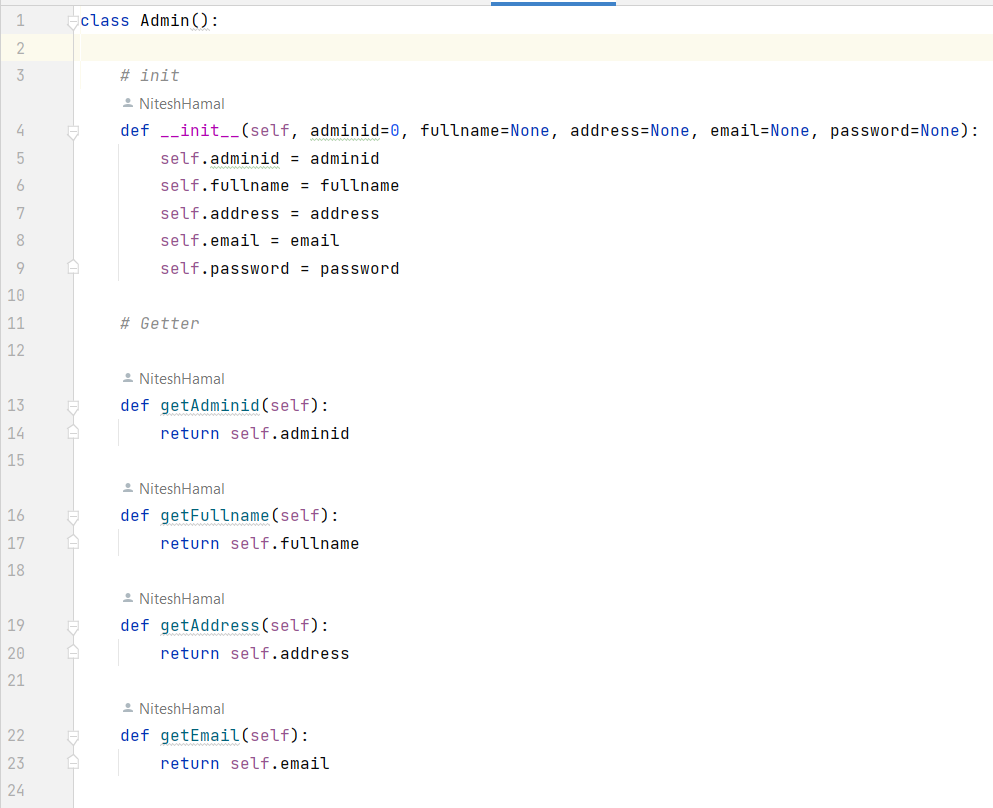


Figure Example of Polymorphism

**Abstraction**

Object-oriented programming uses the term abstraction to describe the idea of concealing implementation-specific information and just displaying an object's core capabilities. In order to concentrate on an object's function rather than its mechanism, abstraction is a beneficial tool. This facilitates both the usage and comprehension of the object as well as the ability to modify the implementation without having an adverse impact on other parts of the code. In Python, you may utilize abstraction by creating a class that has methods that carry out specific functions while obscuring the specifics of how those functions are carried out. Abstraction is a key idea in object-oriented programming, and it is frequently used in conjunction with inheritance and polymorphism to produce a program with a flexible and modular architecture.

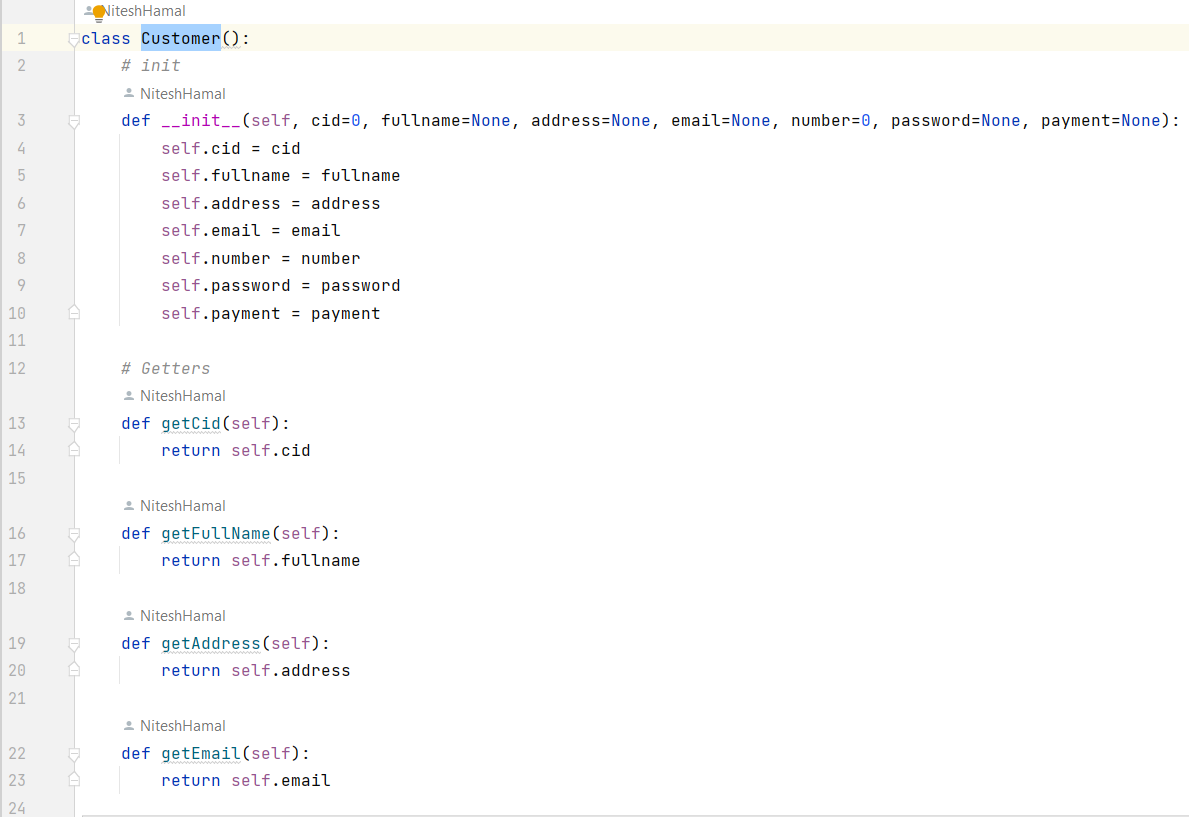


Figure Example of Abstraction

**Encapsulation**

In essence, encapsulation is all about delivering a solution to a problem without forcing the end user to completely comprehend the domain of the problem. In order to prevent users from directly accessing all of an object's variables' state values, encapsulation is a technique for limiting direct access to certain of the object's components. Data members and related data functions or methods for an instantiated class or object can be concealed via encapsulation. In Python, encapsulation refers to the principle of combining data and methods into a single unit. Encapsulation is therefore implemented, for instance, when you create a class. Encapsulation may be seen in the way that classes link all of the data members and methods together into a single entity.



Figure Example of Encapsulation

**Problem Faced**

**Relational Database Management System (RDBMS)**

In order to create, maintain, and manage a relational database, a program known as a relational database management system (RDBMS) must be employed to make all of this possible. Relational databases are a type of database where data is stored in rows and columns arranged in tables that are arranged in a relationship between them. There is a relational database in which every table corresponds to a particular entity, such as a client or an employee, that is stored in that database. Tables are used to display an entity's traits or qualities, while rows are used to display occurrences of the entity, whereas columns are used to display the entity's values. There is an additional benefit to implementing a relational database. This is because it will enable you to connect data contained in different tables to one another by means of keys that have been assigned to each table. For instance, you might have a customer table and an orders table. You could use a customer ID as a key to connect a single customer to their orders. RDBMS software comes in numerous forms, including MySQL, Oracle, and SQL Server. These applications offer a variety of functions and resources to assist you in building and maintaining a relational database, including support for SQL (Structured Query Language), a standard language for communicating with relational databases. When data needs to be organized and retrieved in a systematic fashion, an RDBMS is a helpful tool to store and manage the data. It is frequently used in corporate and enterprise applications, as well as in scientific and research environments. For data storage, I rely on a MySQL database. A free and open-source relational database management system (RDBMS) that is widely used in the world today is MySQL. For the construction and deployment of web-based applications, MySQL has gained the most popularity as the most widely implemented relational database management system (RDBMS). In addition to its use in log analysis, e-commerce, data warehouses, and e-commerce applications, it is also employed in a number of other applications as well. Some of the key benefits of MySQL are its easy installation and use, as well as its vibrant and enduring user and development community. This makes it one of the most popular databases in the world. Additionally, a large number of programming languages, such as PHP, Python, C++, and Java, are supported. Data in MySQL is divided into databases and tables. A database consists of a number of tables, each of which has rows and columns.

# Testing

Include evidence of detailed and robust testing.

There are various ways to document this…

Test Plan / Test Log (tabular format) with date, input/expected output/actual output and screen shot of result.

OR

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

* What went well and what went wrong?
* What would you have done differently?
* How would you improve your application in the future?

This part is very important. You must discuss the project, group work, and time management skills.

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.