**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**SCHOOL OF TECHNOLOGY**

****

**Course: Database Management Systems**

**Course Code: 20CP208P**

**Project**

**B.Tech. (Computer Science and Engineering)**

**Semester 4**

|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| Dr. Hargeet Kaur | Vandeet Shah |
|  | Adit Shah |
|  | Ravi Makwana |

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Content** | **Pg.no.** |
| 1) | Introduction & Use Cases | 3 |
| 2) | Database Schema | 4 |
| 3) | Normalization | 12 |
| 4) | Entity - Relationship Diagram | 13 |
| 5) | Relational Schema | 14 |
| 6) | Python GUI DEMO | 15 |
| 7) | Conclusion | 20 |

horizontal line

# 

# Introduction

In this project, we have designed a database schema for a ride/cab application which we have named ‘CABoom’. We have created a schema consisting of multiple tables to store user data, vehicle data, trip details, payments, and ratings. We have also modelled an E-R Diagram. Additionally, we have also created a GUI using Python to interact with the database.

## Use Case (Application)

The project deals with booking of rides for people to travel short or long distances.

The application provides an interface to the driver as well as the user for booking and accepting rides as and when requested.

This has a wide application in the real world, and is highly useful for providing travel assistance just with an ease of a device. Rides can be booked readily, and details will be recorded in the database accordingly.

This application helps reduce the time delay and efforts for the customers to seek for a ride.

## Database Schema

Here's the schema we have designed for our application:

***Users Table:***

This table stores user data, including username, password, email, and creation date. It has a primary key of user\_id.

**Query:**

a primary key of user\_id.

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone\_number VARCHAR(15) NOT NULL,

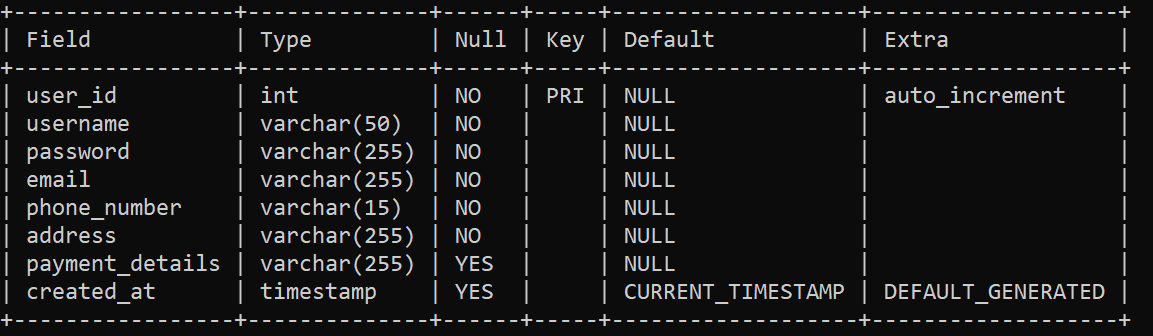
address VARCHAR(255) NOT NULL,

payment\_details VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Table:**



***Vehicles Table:***

This table stores vehicle data, including make, model, year, color, license plate, and capacity. It has a primary key of vehicle\_id.

**Query:**

CREATE TABLE Vehicles (

vehicle\_id INT PRIMARY KEY AUTO\_INCREMENT,

make VARCHAR(50) NOT NULL,

model VARCHAR(50) NOT NULL,

year INT NOT NULL,

color VARCHAR(50) NOT NULL,

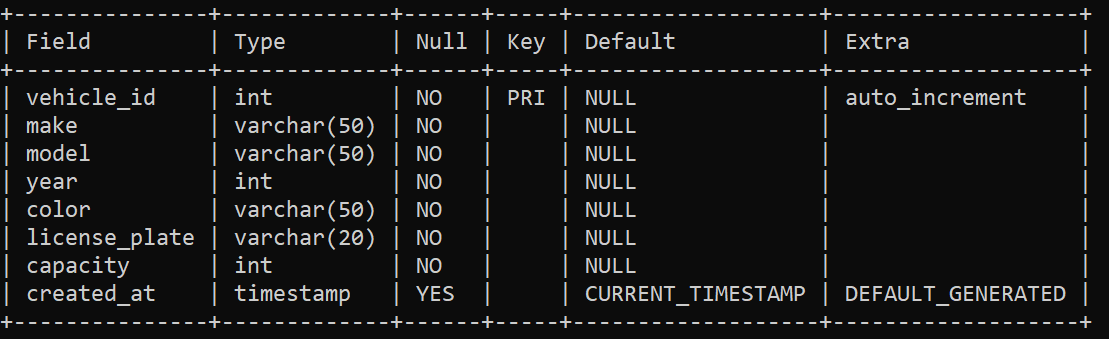
license\_plate VARCHAR(20) NOT NULL,

capacity INT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

**Table:**



***Drivers Table:***

This table stores driver data, including username, password, email, phone number, name, address, and vehicle\_id. It has a primary key of driver\_id and a foreign key of vehicle\_id, referencing the Vehicles table.

**Query:**

CREATE TABLE Drivers (

driver\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

password VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

phone\_number VARCHAR(15) NOT NULL,

name VARCHAR(100) NOT NULL,

address VARCHAR(255) NOT NULL,

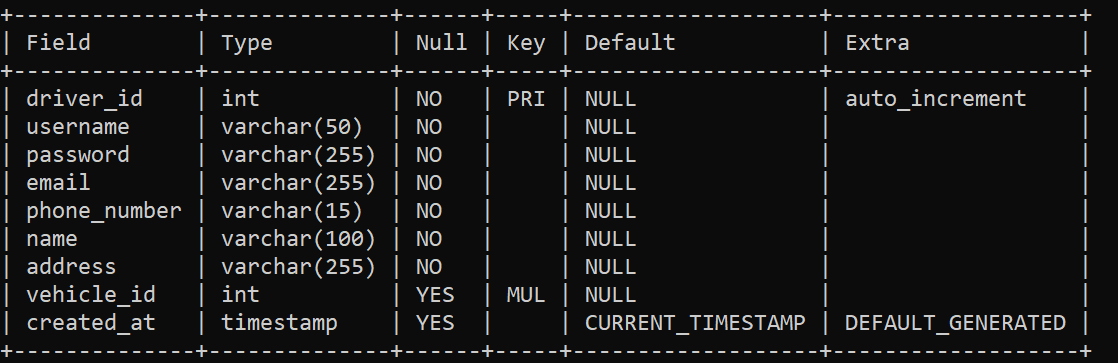
vehicle\_id INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (vehicle\_id) REFERENCES Vehicles(vehicle\_id)

);

**Table:**



***Fare\_ref Table:***

This table stores fare details based on the start and end locations.

**Query:**

CREATE TABLE fare\_ref (

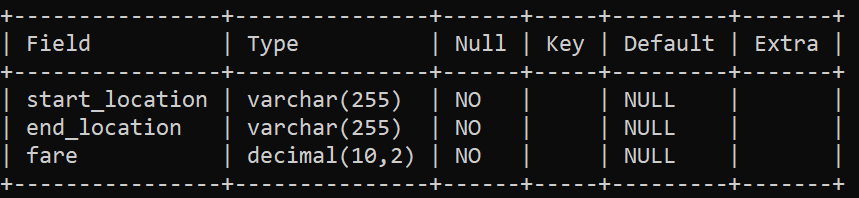
start\_location VARCHAR(255) NOT NULL,

end\_location VARCHAR(255) NOT NULL,

fare DECIMAL(10 ,2) NOT NULL

);

**Table:**



***Trips Table:***

This table stores trip data, including user\_id, driver\_id, vehicle\_id, start\_location\_id, end\_location\_id, start\_time, end\_time, fare, status, and creation date. It has a primary key of trip\_id and multiple foreign keys.

**Query:**

CREATE TABLE Trips (

trip\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

driver\_id INT NOT NULL,

vehicle\_id INT NOT NULL,

start\_location VARCHAR(255) NOT NULL,

end\_location VARCHAR(255) NOT NULL,

start\_time DATETIME NOT NULL,

end\_time DATETIME,

fare DECIMAL(10, 2) NOT NULL,

status ENUM('scheduled', 'ongoing', 'completed', 'cancelled') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

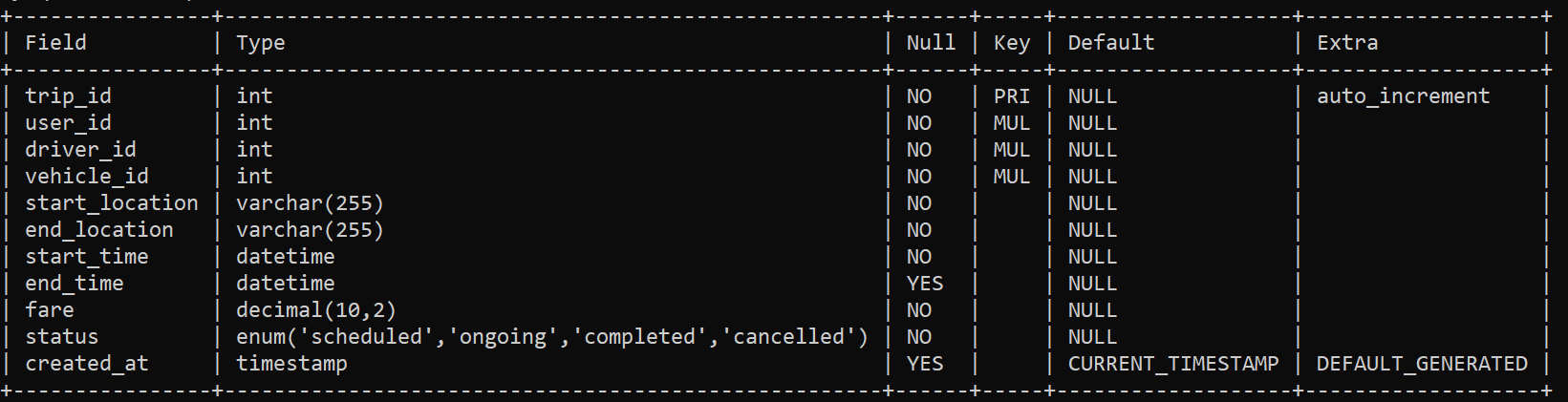
FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (driver\_id) REFERENCES Drivers(driver\_id),

FOREIGN KEY (vehicle\_id) REFERENCES Vehicles(vehicle\_id)

);

**Table:**



***Payments Table:***

This table stores payment data, including user\_id, driver\_id, trip\_id, amount, status, and creation date. It has a primary key of payment\_id and multiple foreign keys.

**Query:**

CREATE TABLE Payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

driver\_id INT NOT NULL,

trip\_id INT NOT NULL,

amount DECIMAL(10, 2) NOT NULL,

status ENUM('pending', 'completed', 'cancelled') NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

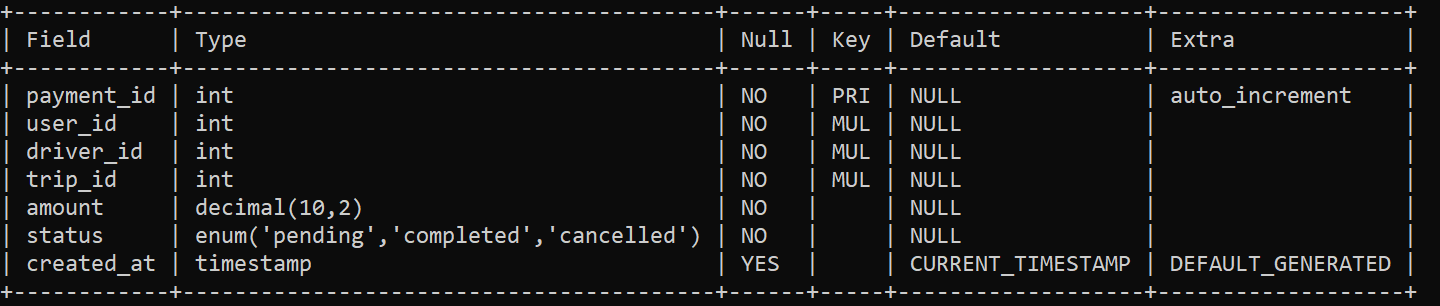
FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (driver\_id) REFERENCES Drivers(driver\_id),

FOREIGN KEY (trip\_id) REFERENCES Trips(trip\_id)

);

**Table:**



***Current\_location Table:***

This table stores the current location of drivers. It has a foreign key of driver\_id, referencing the Drivers table.

**Query:**

CREATE TABLE current\_location (

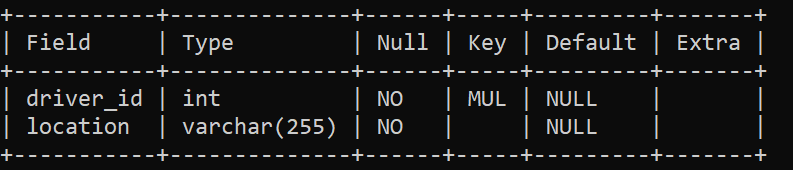
driver\_id INT NOT NULL,

location VARCHAR(255) NOT NULL,

FOREIGN KEY (driver\_id) REFERENCES Drivers(driver\_id)

);

**Table:**



***Ratings Table:***

This table stores rating data, including user\_id, driver\_id, trip\_id, rating, comments, and creation date. It has a primary key of rating\_id and multiple foreign keys.

**Query:**

CREATE TABLE Ratings (

rating\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT NOT NULL,

driver\_id INT NOT NULL,

trip\_id INT NOT NULL,

rating DECIMAL(3, 2) NOT NULL,

comments VARCHAR(255),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (user\_id) REFERENCES Users(user\_id),

FOREIGN KEY (driver\_id) REFERENCES Drivers(driver\_id),

FOREIGN KEY (trip\_id) REFERENCES Trips(trip\_id)

);

**Table:**

### 

**Normalization:**

Normalization is a process of organizing data in a database to reduce redundancy and dependency. We have followed the normalization rules to design the schema for our project. Here's a brief overview of the normalization process we have followed:

***First Normal Form (1NF):***

The 1NF rule states that a table must have a primary key, and all attributes must be atomic. We have applied this rule to all tables in our schema.

***Second Normal Form (2NF):***

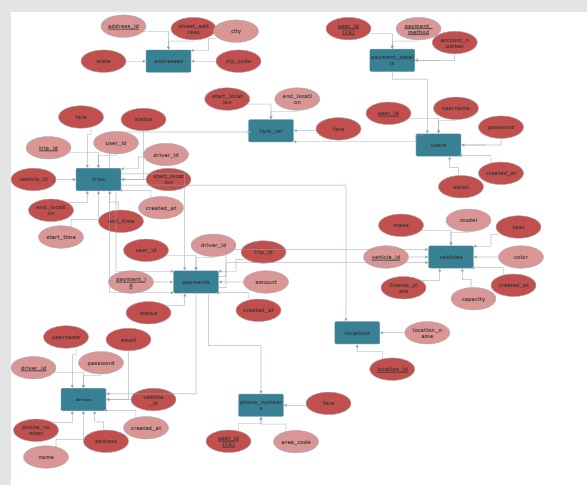
The 2NF rule states that all non-key attributes must be dependent on the primary key. We have achieved 2NF by splitting the Users table into Phone\_Numbers, Addresses, and Payment\_Details tables. This way, we have eliminated redundant data.

***Third Normal Form (3NF):***

The 3NF rule states that all non-key attributes must be independent of each other. We have achieved 3NF by creating separate tables for Trips and Payments.

Hence, in our project, all the tables have at least 3rd Normalization Form (3NF).

**Entity - Relationship Diagram**



**Relational Schema:**

Users (user\_id, username,password, email, phone\_number, address, payment\_details, created\_at)

Drivers (driver\_id, username, password, email, phone\_number, name, address, vehicle\_id, created\_at)

Current\_location (driver\_id, location)

Fare\_ref (start\_location, end\_location, fare)

Payments (payment\_id, user\_id, driver\_id, trip\_id, amount, status, created\_at)

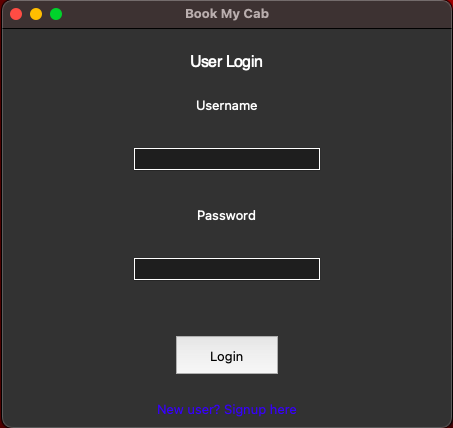
Ratings (rating\_id, user\_id, driver\_id, trip\_id, rating, comments, created\_at)

Trips (trip\_id, user\_id, driver\_id, vehicle\_id, start\_location, end\_location, start\_time, end\_time, fare, status, created\_at)

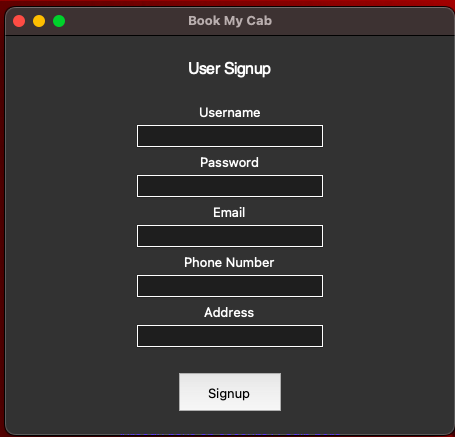
Vehicles (vehicle\_id, make, model, year, color, license\_plate, capacity, created\_at)

**Python GUI DEMO (Screenshots)**

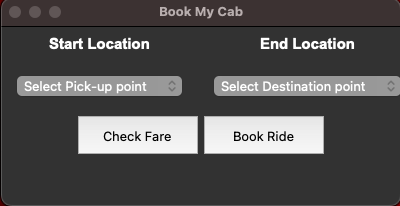
**User Login Portal:**

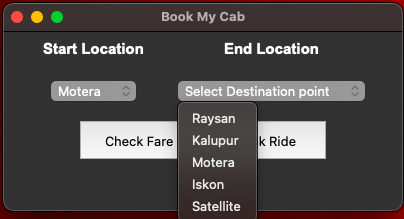


**User Signup Portal:**

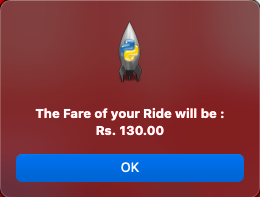


**User Interface:**

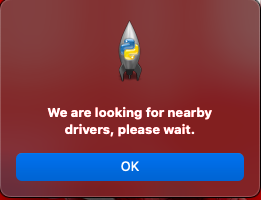




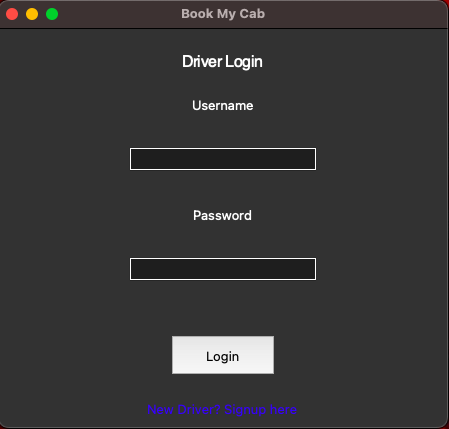
**(on clicking the “Check Fare” button):**



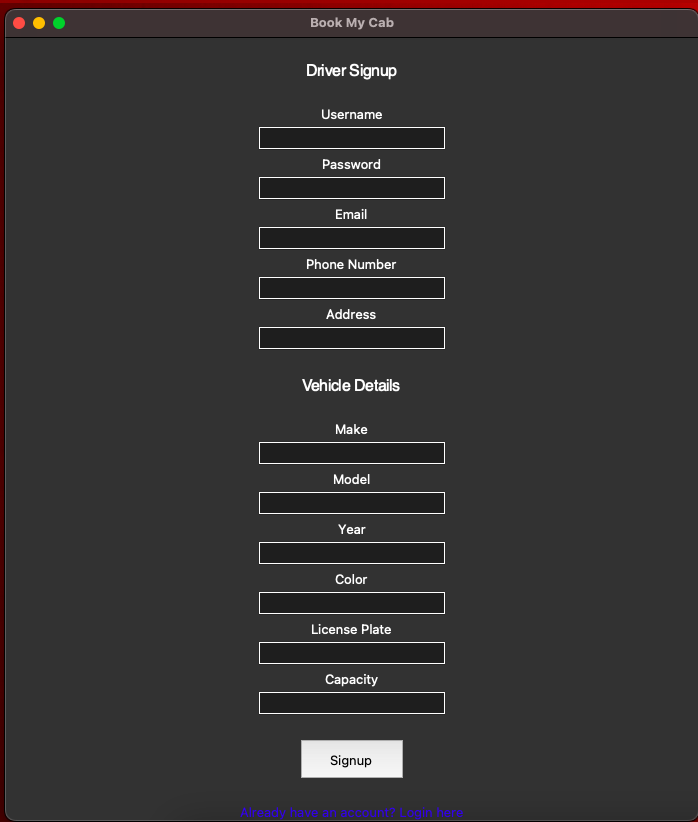
**(on clicking the “Book Ride” button):**



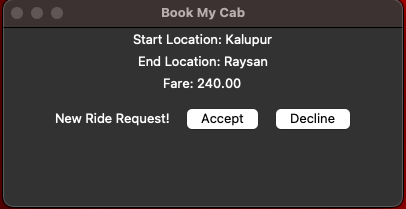
**Driver Login Portal:**



**Driver Signup Portal:**



**Driver Interface:**



**Conclusion**

In conclusion, the DBMS project we have presented is a comprehensive system that allows users to book rides with drivers, make payments, and provide ratings for their experiences. The project uses a MySQL database with several tables to store and manage the relevant data, including user information, trip details, payment information, driver information, and ratings.

Normalization was applied to the database design to eliminate data redundancy, improve data consistency and accuracy, and simplify the database management. The tables were organized into different forms, from the first normal form (1NF) to the third normal form (3NF), to ensure data integrity and minimize data anomalies.

The graphical user interface (GUI) built using Python provides a user-friendly experience for interacting with the database and accessing the various functionalities of the system. The GUI was built with the help of the Tkinter library in Python.

Overall, this project demonstrates the importance of database management systems in modern applications, and the advantages of proper normalization in database design.