Car Rental System

Team Members:

1. **Harsh Vora: 001098451**
2. **Haoran Xu: 002680841**
3. **Malav Ashish Patel: 002965322**

Overview:

Our Purpose for the Project is to provide a robust and efficient database management system to make the car rental process smooth and streamlined. The primary objective is to provide a dependable and efficient system for managing the car rental business by offering a large selection of automobiles at reasonable prices, recording customer service, and utilizing technology to make the car rental process as simple and hassle-free as possible.

Functionalities Provided:

Car Rental System that offers a range of functionalities to both customers and employees of a car rental company. Here is a top-level description of the application's functionality:

1. **User Registration and Authentication:**
    - Customers and employees can log in to their accounts securely.
2. **Car Rental Services:**
    - Customers can browse and search for available rental cars.
    - They can view car details, including brand, type, model, and pricing.
    - Customers can make car reservations, specifying pickup and drop-off locations and times.
3. **Customer Profile Management:**
    - Customers can update their personal information, such as phone number and email address.
    - They can view their reservation history and payment records.
4. **Employee Management:**
    - Employees can access the system with different roles and designations.
    - They can assist customers with reservations, add penalties for a reservation, and make changes in available discounts and insurance.
    - Employees can also view the real-time analytics based on the bookings.
5. **Reservation Handling:**
    - Employees can manage reservations, including creating, modifying, and canceling bookings.
    - Employees can track the availability of vehicles and send vehicles to the garage for service based on the service due date.
6. **Billing and Payment:**
    - The system calculates the billing amount based on reservation details, including fixed costs, variable costs, taxes, and discounts.

    - Customers can make payments through various methods, and the system records payment information.
7. **Location Management:**
    - The system maintains a list of rental locations with addresses and availability information.
8. **Discount and Insurance Options**:
    - Users can explore available discounts and choose insurance options to customize their rental experience.
9. **Penalty Tracking**:
    - Users can view penalties associated with their reservations and billing, facilitating transparency and resolution.

Overall, the Car Rental System aims to streamline the rental process, enhance the customer experience, and efficiently manage reservations, billing, and customer support for a car rental company. Customers can easily book cars, manage their profiles, and receive support, while employees can effectively handle reservations and provide assistance.

README:

This guide will walk you through setting up and running the Car Rental System project on your computer.

Prerequisites:
To run this project, you will need:

- **Python**: Version 3.6
- **MySQL**: Latest version compatible with Python 3.6
- **MySQL Connector for Python**: Required for database connectivity
- **Matplotlib**: Required for visualizations
- **IDE/Text Editor**: Such as Visual Studio Code, PyCharm, etc.

Installation Steps:

1. **Install Python**:
    - Download Python 3.6 from the Python Downloads Page.
    - Run the installer and follow the instructions.
    - Ensure Python is added to your system's PATH.
2. **Install MySQL**:
    - Download MySQL from the MySQL Downloads Page.
    - Run the MySQL installer and follow the setup wizard.
    - Set up a root password for your MySQL server (remember this for later use).
3. **Install MySQL Connector for Python**:
    - Open your command prompt or terminal.
    - Run **pip install mysql-connector-python**.

4. **Install required libraries for Python visualization:**
    - If you don't have matplotlib, run **pip install matplotlib**
5. **Setting Up the Database**:
    - Open MySQL Workbench or the MySQL command-line tool.
    - Create a new database by running the given scripts to set up tables, procedures, triggers, etc. in the following sequence:
        1. Car_Rental_System_Create_Queries.sql
        2. Car_Rental_System_Functions_Triggers.sql
        3. Car_Rental_System_Insertion.sql
        4. Car_Rental_System_Procedures.sql
6. **Configure Database Connection**:
    - Open the **db_connection.py** file in the project directory.
    - Replace the placeholders in the **create_db_connection** function with your MySQL server details (host, username, and password).
7. **Run the Application**:
    - Open the terminal or command prompt.
    - Navigate to the project directory.
    - Run **python main_gui.py** to start the application.

Running the Application:

Once everything is set up, you can run the application using the following command:
python main_gui.py
This will open the main window based on role: customer or employee, choose the login from radio button, and then enter the credentials.

Demo Credentials for Customer:
User Name: qwe
Password: qwe

Demo Credentials for Employees:
User Name: employee1_username
Password: password1

Additional Notes:

- **IDE/Text Editor**: You can use an IDE or text editor of your choice to view or modify the source code. Visual Studio Code or PyCharm are recommended.
- **MySQL Workbench**: For managing the MySQL database, MySQL Workbench is a useful tool.
- **Python Libraries**: Ensure all the required Python libraries are installed. If additional libraries are used in the project, install them using **pip install library-name**.
- **File Structure**: Maintain the file structure as provided in the project repository for smooth functioning.

Troubleshooting:

- If you encounter any issues with Python or MySQL connectivity, ensure that both are correctly installed and configured.
- For issues related to Python libraries, verify that all required libraries are installed.
- Database connection issues are often due to incorrect credentials or server settings in **db_connection.py**.

Support:

For further assistance, consult the official documentation of Python, MySQL, and related libraries, or reach out to the community forums for support.

Database Technical Specifications:

Database Management System (DBMS):

- **DBMS Used**: MySQL
- **Version**: Latest stable release compatible with Python 3.6 (as of your Python version)
- **Character Encoding**: UTF-8 for internationalization support

Database Structure:

- **Database Name**: **Car_Rental_System**
- **Tables**:
    - Location
    - Person
    - Employee
    - Customers
    - Car_Category
    - Cars
    - Car_Reservation
    - Discount
    - Insurance
    - Penalty
    - Billing_Penalty
    - Payment
    - Garage

Data Integrity and Relationships:

- **Primary and Foreign Keys** are used to maintain referential integrity.
- **Normalization**: Up to 3NF to ensure data is stored efficiently without redundancy.
- **Constraints**: Including **NOT NULL**, **UNIQUE**, and **CHECK** constraints for data validation.
- **Cascading**: Including **ON UPDATE** and **ON DELETE** to maintain integrity

Stored Procedures and Functions:

- Multiple stored procedures and functions for encapsulating complex SQL queries and business logic.
- Some procedures make use of complex JOINS between more than 4 tables to provide valuable information to the end user.
- Functions are used to simplify complex calculations involved in calculating the total cost of the reservation after the employee checks in the car by taking into consideration fixed cost, taxes, insurance, penalties, and discounts.

Triggers:

- Implementing triggers to automatically manage specific data changes and maintain data integrity.

Indexes:

- Creating indexes on frequently accessed fields to improve query performance.

Python Application Technical Specifications:

Development Environment:

- **Programming Language**: Python
- **Version**: 3.6
- **IDE/Editor**: Any compatible IDE or text editor (e.g., Visual Studio Code, PyCharm)

Python Libraries:

- **tkinter**: For GUI development.
- **mysql-connector-python**: For MySQL database integration.
- **matplotlib**: For visualization

Application Structure:

- **Modules**:
  - **db_connection**: Manages database connections.
  - **create_operations**: Functions for inserting new records into the database.
  - **read_operations**: Functions for reading data from the database.
  - **database_procedures**: Wrapper functions for calling stored procedures and functions in the database.
  - **main_gui**: The graphical user interface module which provides the option for loginbased on role
  - **employee_gui**: Based on login, if correct credentials for employee are entered, this window will be shown which can perform various CRUD operations.
  - **customer_gui**: Based on login, if correct credentials for customers are

entered, this window will be shown which can perform various CRUD operations.

GUI Features:

- **Main Window**: Central interface with a menu bar for navigation.
- **Menu Options**: Managing different entities like Reservations, Cars, Customers, Employees, Locations, Insurances, Discounts, Penalties, and Payments.
- **Forms and Dialogs**: For adding, editing, and deleting various records.
- **Data Display**: Using Treeview widgets to display lists of entities like cars, customers, reservations, etc.
- **Error Handling**: User-friendly error messages for handling exceptions.

Backend Integration:

- **Database Connection**: Secure and efficient connection handling with the MySQL database.
- **Data Fetching and Updating**: Real-time data interaction with the database for CRUD operations.
- **Business Logic**: Implementation of business rules in Python, complementing stored procedures and triggers in the database.

Testing and Debugging:

- **Debugging**: Regular debugging sessions to ensure code reliability and efficiency.

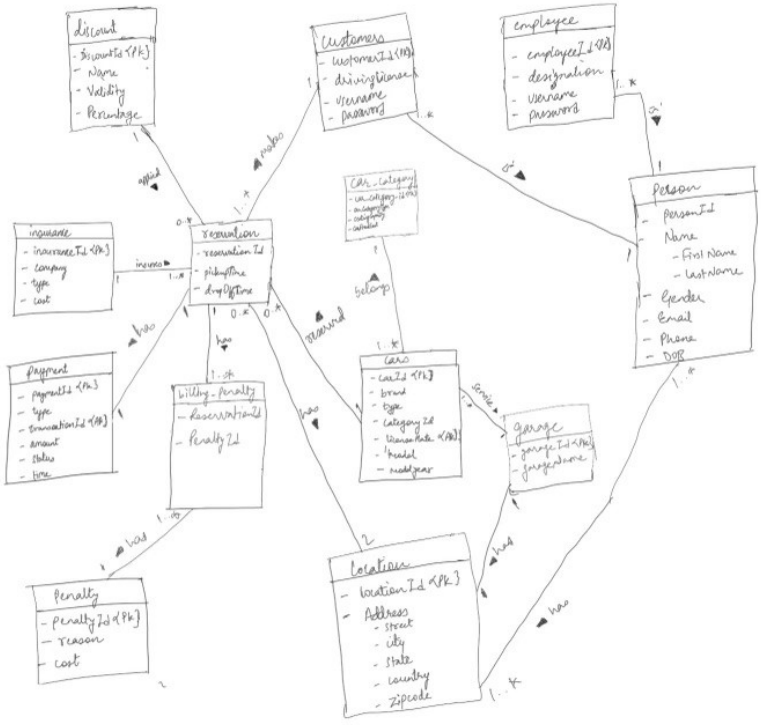
Project Deployment:

- **Local Deployment**: The project can be run on a local machine with Python 3.6 and MySQL installed.
- **Portability**: The application should be easily portable across systems with similar environments.

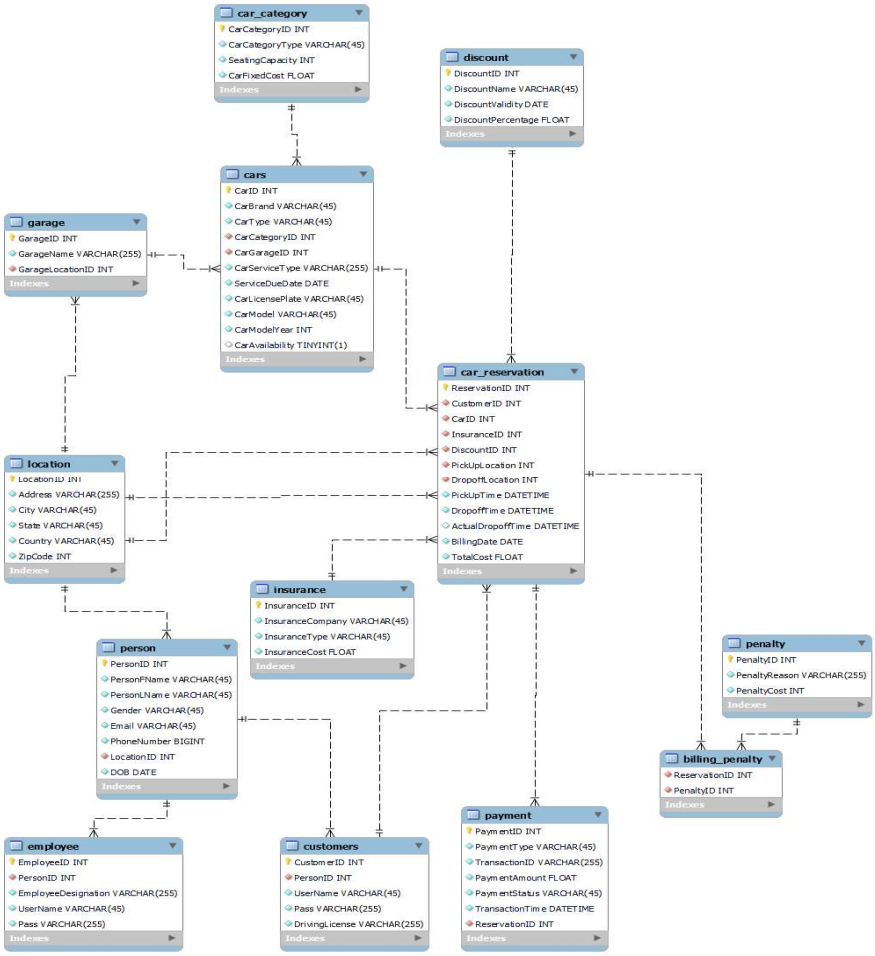
Documentation and Code Maintenance:

- **Code Documentation**: In-line comments and docstrings for major functions and classes.
- **User Manual**: A simple user manual for operating the application.

UML diagram:


discount
- DiscountId (Pk)
- Name
- Validity
- Percentage
customers
- CustomerId (Pk)
- drivingLicense
- Username
- password
employee
- employeeId (Pk)
- designation
- username
- password
Person
- PersonId
- Name
- First Name
- LastName
- Gender
- Email
- Phone
- DOB
insurance
- insuranceId (Pk)
- company
- type
- cost
reservation
- reservationId
- pickupTime
- dropOffTime
Car_Category
payment
- paymentId (Pk)
- type
- transactionId (Pk)
- amount
- status
- time
Billing_Penalty
- ReservationId
- PenaltyId
cars
- CarId (Pk)
- brand
- type
- category Id
- VinNumber (Pk)
- brand
- modelyear
garage
- garage_Id (Pk)
- garageName
Penalty
- penaltyId (Pk)
- reason
- cost
Locations
- locationId (Pk)
- Address
- street
- city
- State
- country
- Zipcode

Logical Design:



User Flow:

Based on roles, the user who is a customer can perform the following operations:

- Can perform CRUD operations on reservation
- Can view only the available cars
- Can view insurance
- Can view discounts
- Can view locations for pickup and drop-off
- Can make a payment using the desired mode towards a reservation
- Can update profile information

Based on role, the user who is an employee can perform the following operations:

- Can perform CRUD operation on reservations
- Can perform CRUD operation on cars
- Can toggle availability of car based on service due date.
- Can check-in the car to generate the final bill

- Can view all the customers
- Can view all the employees
- Can view all the locations available
- Can perform CRUD operation on Insurance
- Can perform CRUD operation on Discounts
- Can perform CRUD operation on Penalties
- Can view payments
- Can see the location-based graphical real-time analytics

Lessons Learned:

Technical Expertise Gained:

1. **Database Management**: Learned intricate details of SQL, including schema design, writing complex queries, stored procedures, and triggers. Gained proficiency in MySQL.
2. **Python Development**: Improved skills in Python, especially in integrating Python with SQL databases using **mysql-connector-python**.
3. **GUI Development**: Enhanced understanding of Tkinter for Python GUI development, including handling events, widgets, and interface layout.
4. **Software Engineering Principles**: Gained experience in modular design, keeping code organized and maintainable.

Insights:

- **Time Management**: Managing a project of this scale highlighted the importance of effective time management, especially in segmenting tasks and prioritizing critical components.
- **Data Domain Insights**: Gained a deeper understanding of the car rental business, including the variety of entities involved and their complex interrelationships.
- **Testing and Debugging**: Realized the importance of thorough testing, which helped catch bugs early and ensured smoother integration.

Alternative Design / Approaches:

- **Web-based Interface**: While a desktop-based GUI was chosen, a web-based interface using frameworks like Flask or Django could provide more accessibility and scalability.
- **ORM (Object-Relational Mapping)**: Instead of raw SQL queries, using an ORM like SQLAlchemy could have simplified database interactions and made the code more Pythonic.
- **Rest API**: Implementing a RESTful API for the backend could have separated concerns more cleanly, making it easier to switch frontends (web, mobile apps) without altering the backend logic.

Future Work:

Planned Uses of the Database:

- **Expansion to Online Platform**: Integrating the database with a web application to provide online booking services.
- **Analytics and Reporting**: Utilizing the data for generating additional detailed analytics and reports, aiding in business decisions and customer insights.

Potential Areas for Added Functionality:

- **Dynamic Pricing**: Developing algorithms for dynamic pricing based on demand, season, and car availability.
- **Maintenance Scheduling**: Automating car maintenance schedules based on usage and time intervals.
- **Mobile Application**: Developing a mobile app for customers to make reservations on the go.
- **Feedback and Rating System**: Implementing a system for customers to provide feedback and ratings for services.