**CarConnect – Car Rental Management System**

**Author**: Afreen Sultana A

**Overview**

CarConnect is a full-stack car rental platform built as part of Hexaware’s foundation training.  
The project manages customers, vehicles, reservations, and administrative tasks using **Python** and **MS SQL Server**.

**Features**

* **Customer Management**: Register, authenticate, update profile, and view booking history.
* **Vehicle Management**: Add, update, search, and list vehicles with availability and daily rates.
* **Reservation Handling**: Create, update, and cancel reservations; compute total cost and track status.
* **Admin Module**: Manage fleet, view reports, and handle user roles and access.
* **Reporting**: Generate summaries of reservations, vehicle availability, and revenue.

**Technology Stack**

* **Backend Language**: Python
* **Database**: Microsoft SQL Server
* **Database Connectivity**: pyodbc
* **Testing**: Python unittest framework

**Database Schema**

Four main tables power the application:

1. **Customer**
   * customer\_id, first\_name, last\_name, email, phone\_number,  
     address, username, password, registration\_date
2. **Vehicle**
   * vehicle\_id, model, make, year, color, registration\_number,  
     availability, daily\_rate
3. **Reservation**
   * reservation\_id, customer\_id, vehicle\_id,  
     start\_date, end\_date, total\_cost, status
4. **Admin**
   * admin\_id, first\_name, last\_name, email, phone\_number,  
     username, password, role, join\_date

**Project Structure**

CarConnect/

│

├─ EntityModel/ # Core entity classes: Customer, Vehicle, Reservation, Admin

├─ DAO/ # Data Access Objects & service provider interfaces/abstract classes

├─ Exception/ # Custom exceptions (e.g., AuthenticationException, VehicleNotFoundException)

├─ Util/ # Database context and utility classes

├─ Main/ # Application entry point and menu-driven logic

└─ Tests/ # Unit test cases

**Key Python Classes**

* **Entities**: Customer, Vehicle, Reservation, Admin
* **Services**: CustomerService, VehicleService, ReservationService, AdminService
* **Interfaces**: ICustomerService, IVehicleService, IReservationService, IAdminService
* **Utilities**: DatabaseContext, AuthenticationService, ReportGenerator
* **Exceptions**: AuthenticationException, ReservationException,  
  VehicleNotFoundException, AdminNotFoundException, InvalidInputException

**Unit Testing**

Implemented using Python’s unittest to ensure correctness and reliability.  
Key test scenarios include:

* Customer authentication with invalid credentials
* Updating customer information
* Adding and updating vehicle details
* Listing available vehicles and all vehicles
* Validating reservation creation and status updates

**How to Run**

1. **Database Setup**:
   * Create the database and run SQL scripts to create the four tables.
   * Update db\_util.py with your SQL Server credentials.
2. **Install Dependencies**
3. **Run Application**
4. **Execute Tests**

**Key Learning Outcomes**

* Designed normalized SQL tables with foreign key relationships.
* Applied Python OOP principles (inheritance, abstraction, and interfaces).
* Implemented robust error handling with custom exceptions.
* Practiced unit testing for reliability and maintainability.