“The **Vehicle Rental Management System** is designed to manage the rental process of different types of vehicles including **Cars**, **Motorcycles**, and **Trucks**. The system incorporates key principles of **Object-Oriented Programming (OOP)** such as **Abstraction, Inheritance, Polymorphism, Encapsulation, and Composition**.”

“The main goal is to demonstrate how these principles are applied to build a robust, flexible, and maintainable system that facilitates the management of vehicle rentals efficiently.”

1. Problem Domain

* “Core Requirements: The system needs to manage a fleet of vehicles, handle customer rentals, calculate rental costs, and track rental transactions.”
* “Vehicle Types: Three main types of vehicles are supported Car, Motorcycle, and Truck each having unique rental characteristics.”

2. Key OOP Principles Demonstrated:

* Abstraction:
  + “An abstract Vehicle class is created, defining common attributes and abstract methods for calculating rental costs and checking availability.”
  + “Concrete implementations (Car, Motorcycle, Truck) extend the Vehicle class to implement specific behaviors.”
* Inheritance:
  + “The Car, Motorcycle, and Truck classes inherit from the abstract Vehicle class, ensuring a clear hierarchy and reusable code structure.”
  + “Each subclass provides specific implementations for calculating rental costs and checking vehicle availability.”
* Polymorphism:
  + “A Rentable interface is defined with methods rent and return Vehicle, which are implemented by the Vehicle subclasses.”
  + “The RentalAgency class demonstrates polymorphism by processing rentals and returns using Vehicle or Rentable types.”
* Encapsulation:
  + “Private fields for critical information like vehicleId, model, baseRentalRate, and isAvailable are encapsulated within the Vehicle class.”
  + “Getter and setter methods ensure controlled access, with input validation for setters.”
* Composition:
  + “The Rental Agency class maintains a collection of Vehicle objects to manage the fleet.”
  + “The Customer class keeps track of rental history by storing a list of rented vehicle IDs.”

***Testing & Evaluation***

* **“Unit Testing**: Each class and method is unit-tested to ensure correctness of functionality.”
* **“Polymorphism & Inheritance**: Tested with various combinations of vehicle types and rental scenarios.”
* **“Abstraction & Encapsulation**: Ensured through method abstraction, private fields, and controlled access.”

***Conclusion***

“This system provides a clear and scalable design using key OOP principles. It supports adding new vehicle types or rental rules with minimal changes, adhering to a well-structured and maintainable system.”