**Applied Solid principles on the project:**

1. Yes, I applied on this project most of the Solid principles:
2. Single Responsibility: applied on Most of the classes where all the classes in this project has only one functionality to do like
3. Garage class which has used to manage the garage
4. Slot which stores all slots data
5. Customer who stores all slots data
6. Interface Segregation: applied on interface authenticate and abstract class loggin

Where authentication hold the same function that will be inherited and loggin hold the function which will be override

1. Open closed: also applied on interface authenticate where I inherit from it

loginOwner and LoginAdmin and Later if I want to add new user to the system, I will create new class and override the function in the authenticate class

1. Dependency inversion: in this project I have two different login methods on for the owner and the admin in last phase put them in one class, but in this phase I made interface called authenticate to take the function which will be override and Abstract class called loggin to hold the function both have with same input and same output and already implemented in loggin class.
2. Yes, this project contains design patterns
3. Adapter: like I said in Dependency inversion I spilt the class to two classes to make them reusable.

Where authenticate class contain authentication function which take two parameters

String username and string password.

1. Strategy: in Person class it holds username and password and Owner and Admin only inherit from it
2. singleton: used in Display Class.

Also, I Designed the project with all object oriented concepts like:

Abstract class, Interface, inheritance, polymorphism and packing.