SW2 – Project Evaluation Form

- **Each team must submit the following Documentation that contains:**
  - **Project Description in detail.**
  - **Class Diagram. And Database Schema.**
- **Each team must submit the project via GitHub:**
  - **Source Code.**
  - **Video Demo for running (2 – 5 Minutes).**
  - **Documentation and Evaluation Form.**
- **The Evaluation will start with giving all teams 30 marks then check the following criteria:**

Violation Level	Full	Medium	Small	Grade
Documentation	-5	-2	-1	
Not Apply MVC (it does not Separate Business logic from GUI). Example of violation: write the implantation for a method such as an inset item into the database inside the Button Action method)	-6	- 3	-1	
Violate clean code – Variables	-2	-1	-.05	
Violate clean code – Functions	-2	-1	-.05	
Violate Single-responsibility Principle	-2	-1	-.05	
Violate Open-closed Principle	-2	-1	-.05	
Violate the Liskov Substitution Principle	-2	-1	-.05	
Violate Interface Segregation Principle	-2	-1	-.05	
Violate Dependency Inversion Principle	-2	-1	-.05	
Not Upload code to GitHub	-1			
Only One Branch Without Merge (GitHub)	-2			
Only One Contribution (GitHub)	-2			
Total Minus from Grade				

Design Pattern Bounce	+4	
Bounce on Overall Work	+2	
Total Team Grade / 30		

Name (Arabic)	ID	Individual Bounce +2	Grade
مينا عماد ميخائيل رياض	201900879		
مونيكا نادر صدقي جيد	201900861		
ميرنا عاطف مرزوق صليب	201900872		
موريس عايد يوسف حنا	201900859		

Parking Guidance System

1-Customer Module :

a. Enables customer to print ticket for entry station with entry id, plate number, transaction date

b. Enables customer to pay in exit station for parking hours with entry id.

2-Operator in Entry Station :

a. Enables operator to monitor free spots in Parking

b. Enables operator to advise customer with free spot.

3-Operator in Exit Station :

a. Enter ticket id to calculate total parking hours .

4- Admin Module :

a. Enables admin to add spots in parking.

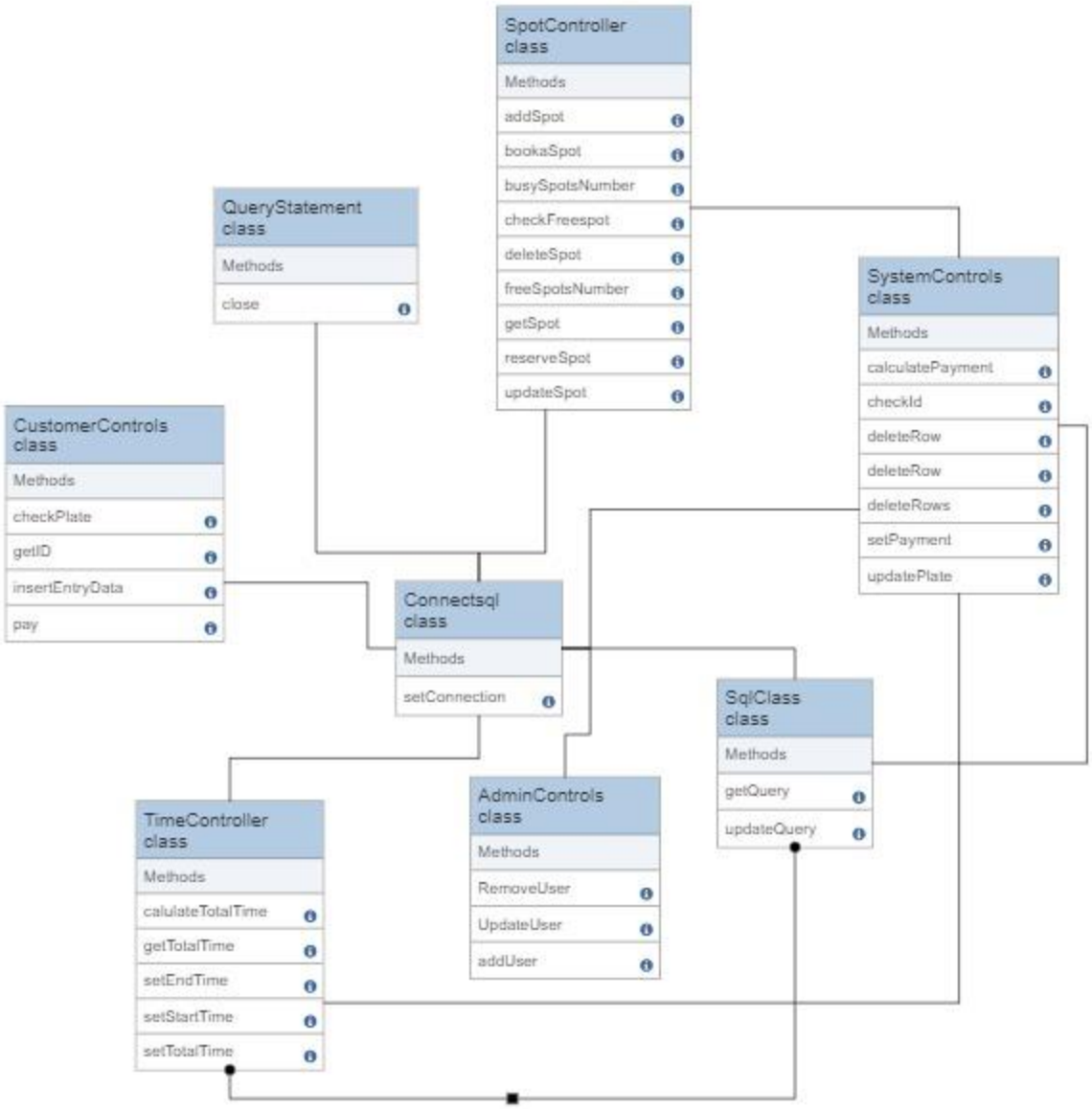
b. Enables admin to view total spots in parking.

c. Enables admin to add / update / delete users with different roles

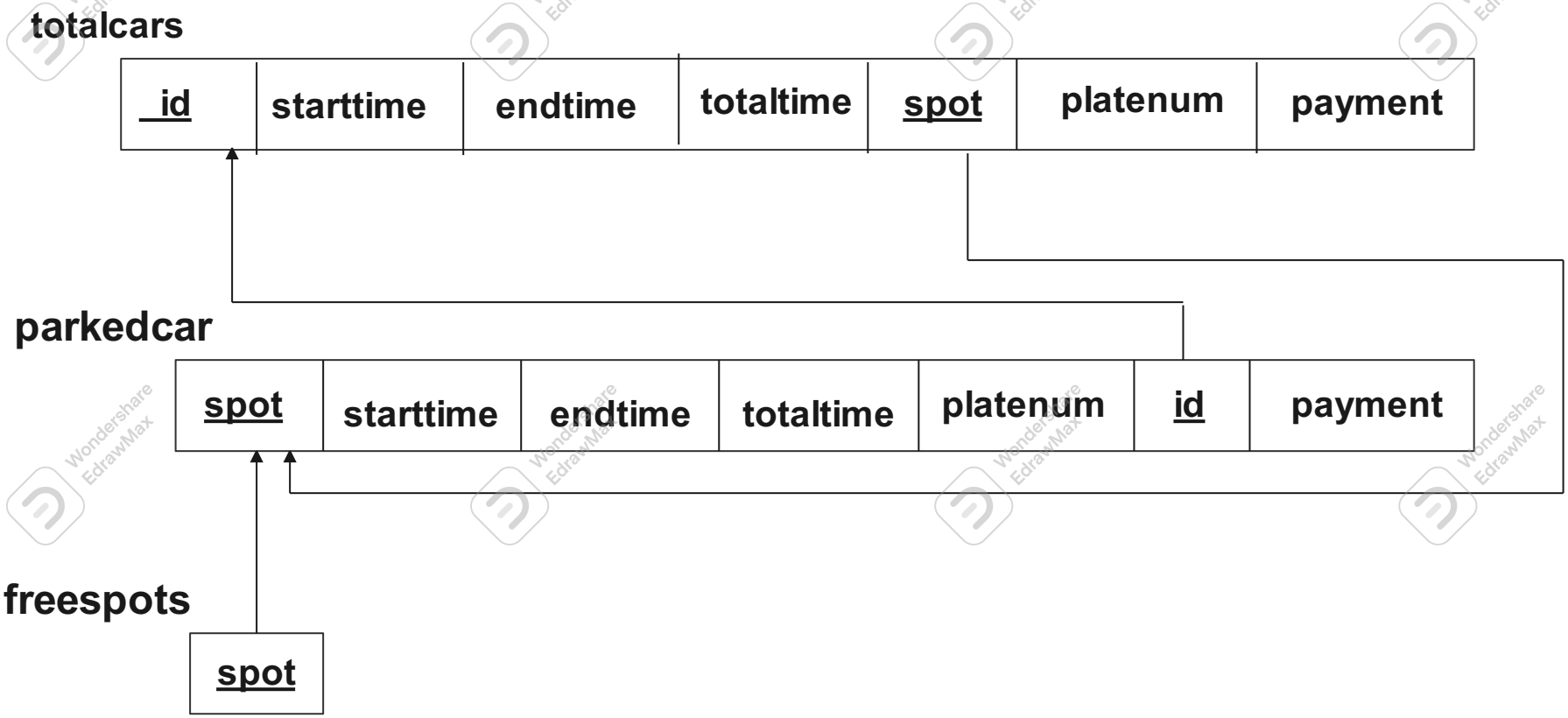
d. Enables admin to view shifts report with payment

e. Enables admin to view parked cars report

Class diagram :



DataBase Schema :



Use Observer Pattern

Observer pattern:

To understand observer pattern, first you need to understand the subject and observer objects.

The relation between subject and observer can easily be understood as an analogy to magazine subscription.

A magazine publisher(subject) is in the business and publishes magazines (data).

If you(user of data/observer) are interested in the magazine you subscribe(register), and if a new edition is published it gets delivered to you.

If you unsubscribe(unregister) you stop getting new editions.

Publisher doesn't know who you are and how you use the magazine, it just delivers it to you because you are a subscriber(loose coupling).

Definition:

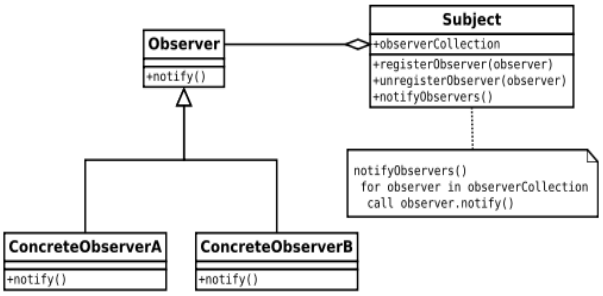
The Observer Pattern defines a one to many dependency between objects so that one object changes state, all of its dependents are notified and updated automatically.

Explanation:

One to many dependency is between Subject(One) and Observer(Many).

There is dependency as Observers themselves don't have access to data. They are dependent on Subject to provide them data.

Class diagram:o2



Here Observer and Subject are interfaces(can be any abstract super type not necessarily java interface).

All observers who need the data need to implement observer interface.

notify() method in observer interface defines the action to be taken when the subject provides it data.

The subject maintains an observerCollection which is simply the list of currently registered(subscribed) observers.

registerObserver(observer) and unregisterObserver(observer) are methods to add and remove observers respectively.

notifyObservers() is called when the data is changed and the observers need to be supplied with new data.

Advantages:

Provides a loosely coupled design between objects that interact. Loosely coupled objects are flexible with changing requirements. Here loose coupling means that the interacting objects should have less information about each other.

Observer pattern provides this loose coupling as:

1)Subject only knows that observer implement Observer interface.Nothing more.

2)There is no need to modify Subject to add or remove observers.

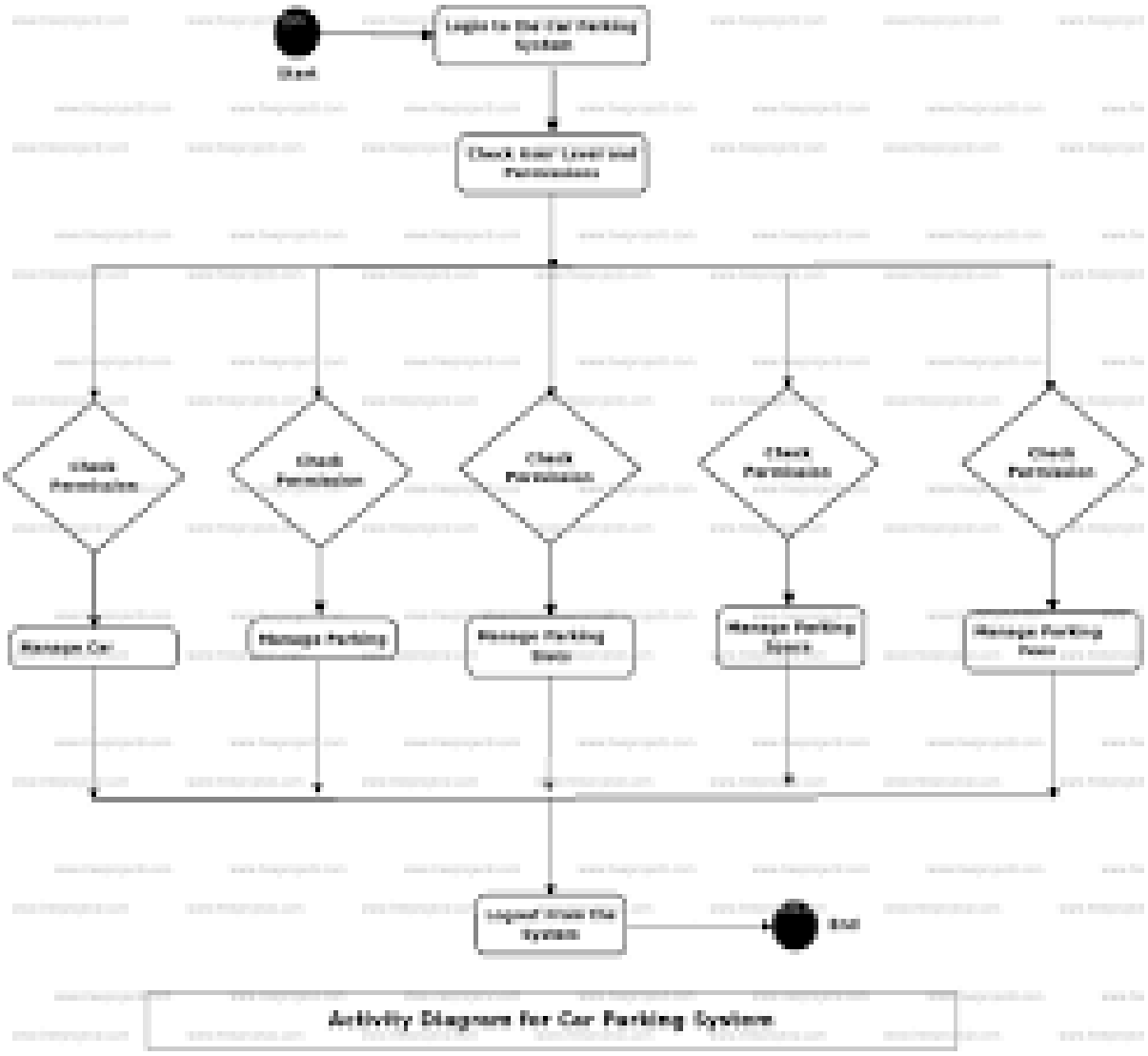
3)We can reuse subject and observer classes independently of each other.

Disadvantages:

1)Memory leaks caused by Lapsed listener problem because of explicit register and unregistering

of observers.

Activity diagram:



Activity diagram: