Thank you for sending this application, it was very interesting to solve.

I will be happy to discuss the code in greater details however I thought of sharing some design

decision for the application with some other notes.

* As given in the instruction the Starting Application is a console, we can enter the **Charging options** in the console itself. I thought it will be easier than passing via command line to console. Generally each user should have one parking session hence console will close after one execution. Please refer to Unit Test Case for more extensive scenarios
* I have implemented **Strategy Pattern** for calculating the Parking Fee. I believe this will help to maintain Open-Close Principle as we can have more charging options in future . Since each calculation is very different than other (like in One Weekend is considered in other its not) it will be better to keep those logic to respective classes.
* I have also used **Factory Pattern** as a creational pattern used bit of reflection for one time use so that Adding new ParkingCalculator is seamless. Storing it into collection so we don’t perform this expensive operation again. (At the moment, the application support only one instance)
* Since Short Stay uses Minutes for calculation the methods are using the same across.

In ParkingConfig **ShortStayPerMinFee** uses calculation instead of fixed value the reason is for clarity of the reader if they try searching by Charges in Project.

* Since Parking System are generally local based system hence Globalisation or different Time Zone are not considered in implementation.
* Used Decimal up to 2 decimal points as this is most user friendly for person paying the parking fee.
* I have ignored UK Bank Holiday since it was not mentioned in requirement.
* For Short Stay calculation, have used the same approach as I would have used while calculating the Fee manually. If parked overnight split charges into 3 components
  + Fee on the day vehicle was parked (Parking Date)
  + Fee for full chargeable days (Non-Weekend)
  + Fee on the last day (Exit Day)

For case where Parking Date and Exit Day are same its handled accordingly

* Some of the names are big it could be shortened like by using mins instead of minutes. However it is left as it is as some code analysers will prompt them as spelling mistake and it also helps to make its use very clear, hence less dependence on code comments.
* I was told not to give lot on emphasis on **DI** (Dependency Injection) hence I have not implemented it and also left the test cases for the similar classes like ParkingManager, CalculatorFactory as I think more emphasis was told to be given on core module, which has relevant code coverage / Unit Test Cases.
* CalculatorFactory core logic are quite lean and specific hence I have left them in the same project as ParkingCalculator as we use it to create only ParkingCalculator. However, if new features are added I would ideally move them into another project so that CalculatorFactory can create other type of Calculators as well and add one more level of abstraction on top of existing generic design.

Kind Regards,

Kumar Shantanu