01/2019

Parking Ticket System Report

**Author id: Q13699474**

**Author name: Antons Lelis**

**Course title: Bsc Software Enginering**

**Tutor’s name** **Dr Craig Gallen**

# Diagrams

The skeleton of domain diagram was provided to us in the archetype, however all methods were renamed and changed where needed.

In terms of decisions, the main one was to store the price timetable in the parkingMeter itself, because the array list from 0 to 23 fit our needs, as number of the object in the array list can be viewed as hour. This makes the implementation of the new class that will store timetable unnecessary.

Also, another major decision, was to remove the retriveMatching method from the rest interface, as we will only need to work with one meter at the time, so the retriveMatching method will be unused, so I decided to remove it completely.

No major decisions were made during the creation of use-case and robustness diagrams as they were drawn after implementation of the code.

# Unmatching Design

Unfortunately, due to bad time management I was unable to complete purchase section of the work, so the client side of the project doesn’t work and is not included in the submitted version, because the unfinished work was causing issues at the building stage of the project. Apart from that, everything else matches the design.

# Test plan

Each module has its own tests which are implemented using Junit. Tests are implemented in all modules to simplify the error handling, as if the tests pass in DAO, for example, and the error occurs later on, we will know, that the problem is not with DAO.

In the model module, test was implemented to check if ParkingMeter is being properly Marshaled and Un-marshaled. Tests in DAO are used to check if DAO can work with ParkingMeter object. Tests on service layer make sure that service factory properly generates service facade and checks that service façade methods can be accessed. Rest interface test are there to check the retrieve function.

# Critical evaluation

I personally think that I could have done a better job if I started work on the project earlier, as the lack of time is the main reason why the project is unfinished. I was also forced to use silly and unprofessional method on the web part of the project, as I could not come up with the proper solution in time, and despite the final version being silly, at least it works.

Despite failing to fully complete the project, this unit and project gave me probably best lesson of my life, because before I usually was able to do everything at the last moment. The unit also gave me a good look of what programmers work is actually is about, as we mostly were working with pre-written code and were only making changes and implementing certain methods.