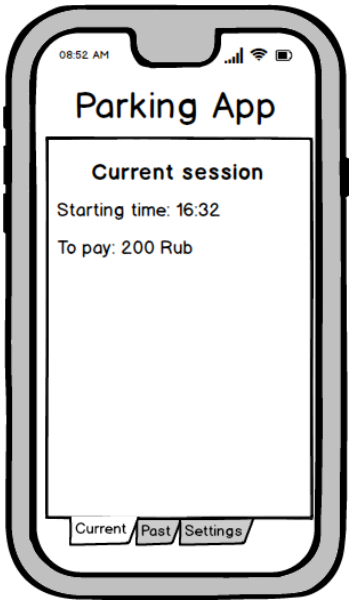
Programming BI. Part 2
Homework assignment 2.
Smart parking: GUI applications



1 Tested skills

- OOP
- GUI design
- Multi-layer application design

2 Introduction

In this assignment, you will continue working on the Smart Parking application. Please read the HW1 requirements to refresh the original task. You can continue working in your solution or take the provided sample as the starting point.

You will need to implement two graphical interfaces: for the client of the parking system and for its owner. Note that usage of WPF is not required, you can utilize any other framework that enables building graphical desktop (or mobile) applications in C#.

3 Description of tasks

Project structure - 1 point

You need to build a solution structure with several projects. Two GUI projects (client and owner apps) need to reuse the same logic of the ParkingManager class and connected classes. The ParkingManager class may need to be extended to support all the functionality required in the two graphical applications.

Client application - 5 points

For the client application, the following functionality is required:

- Registration in the system: the user enters the fields contained in the User class. Add a password field to the User class, the phone number may be used as the login. When registering a new user, make sure that a user with the same login does not already exist.
- Authorization in the system (the user enters login and password)
- Viewing details of any completed session (that is connected to the authorized user)
- Viewing current session (if the user entered the parking, i.e. there is an active session with the user's car plate number, the application should show the starting time and the cost of the session).

Owner application - 4 points

The owner application needs to provide the following functionality:

- Viewing current state of a parking lot: percent of filled spaces
- Viewing details of any parking session, both current and past
- Calculating total profit over an arbitrary period (defined by the start date and end date). If the period contains active sessions (that are not finished), do not take them into account.

Bonus task: + 2 points

In the bonus task, you need to analyze the parking lot occupancy in a given period (entered as two dates: start date and end date), hour by hour.

You need to calculate the maximum number of cars present at the parking lot simultaneously for each of the 24 hour slots, i.e. 00:00 - 01:00, 01:00 - 02:00, ..., 23:00 - 00:00. Don't forget to average the result over the number of days in a given period.

Consider a parking lot with 4 total spaces and a period of three days (start date = 01.11.2019, end date = 03.11.2019) that has the following completed sessions:

01.11.2019	**02.11.2019**	**03.11.2019**
06:15 - 06:20	07:50 - 08:40	13:45 - 14:10
	08:15 - 09:02	14:50 - 15:20

Then the following distribution table can be constructed:

Time of day	**Maximal number of cars**			
	01.11.2019	**02.11.2019**	**03.11.2019**	**Occupancy (%) over 3 days**
00:00:00 - 00:59:59	0	0	0	0
01:00:00 - 01:59:59	0	0	0	0
...				
06:00:00 - 06:59:59	1	0	0	8.3
07:00:00 - 07:59:59	0	1	0	8.3
08:00:00 - 08:59:59	0	2	0	16.7
09:00:00 - 09:59:59	0	1	0	8.3
...				
13:00:00 - 13:59:59	0	0	1	8.3
14:00:00 - 14:59:59	0	0	1	8.3
15:00:00 - 15:59:59	0	0	1	8.3
...				
23:00:00 - 23:59:59	0	0	0	0

You only need the result from the right column. The way of displaying this result to the user is left up to you. As a simple option, you can use a DataGrid, but you can use an external library that plots graphs, you can also export the distrubution to Excel.

4 Submission

Submit your solution following these steps:

1. Delete two temporary folders - “bin” and “obj” from the project folder.
2. Add the whole solution folder to a ZIP **(not RAR or 7Z)** archive.
3. Upload the archive to the Canvas LMS in the corresponding section

5 Grading policy

In general, the grade is determined by the quantity and quality of completed tasks taking into account the maximal possible marks for each of the three parts. The overall grade can be lowered in the following cases:

- Inefficient implementation of an algorithm (-1 point)
- Poor programming style (-1 point)