UTCN
Computer Science Department
Software Design 2019

ASSIGNMENT A2

==========

1. Objective

The main objective of this assignment is to allow students to become familiar with one of the MV* architectural patterns: MVC, MVP or MVVM.

2. Application Description

Use JAVA/C# to design and implement an application for an online Parking Request System. The main goal of the application is to provide clear and transparent way of requesting and assigning parking spots in the parking lots of Cluj-Napoca. Each citizen can make a new request for a parking spot. The request can be done only for one car that the citizen owns, but he can select multiple parking lots that would suit him good. Each parking lot has a certain number of parking spots. If the citizen has multiple cars, he must file multiple parking requests.

The application should have two types of users: a regular user represented by the citizen and an administrator user, represented by the city clerk responsible for assigning the free parking spots. Both kinds of uses have to provide an email and a password in order to access the application.

The citizen (regular user) can perform the following operations:

- Register a new account.
- Add/Remove owned Car.
- View all request made by him.
- Make a new request.
- Update/Delete request.

The clerk (administrator) user can perform the following operations:

- See list of parking lots.
- See a list of parking requests for a parking lot (ordered by request date).
- Assign Parking Spots to citizens.
- Retract Parking Spots from citizens who did not pay the parking tax.
  (Mark parking spot as free)

3. Application Constraints / Requirements

- Implement the client side of the application
- Use an MV* architectural pattern to organize your code.
- The Model will simulate a connection to the server and will return hard-coded data. NOTE: The data will be updated as the application runs, but cannot be saved (i.e. restarting the application returns again the hard-coded data). For example:
    - After creating a new request the list of all requests is updated with the new element.
    - Updating a request will be visible both in the details view and the list of all requests.
- The user must be able to navigate to all required views.
- Use the **Command Design Pattern** to handle all user input.
    - Example: When the user clicks the login button, you should trigger a Login Command. You do not need to implement commands for key-press events.
- Integrate the **Builder Design Pattern** when implementing the *Make new request* view.
- Methods/Commands with business logic can be stubbed.
    - Example: The LoginCommand redirects to the admin page if the username is *admin* or to the user page if the username is *user*

4. Grading

Grade	**Functionality**
5	Login View List of Requests Create Request (Builder Pattern included) Update Delete Requests
6	Register account Add Delete Car
8	View List of Parking Lots View Single Parking Lot ● View List of Parking Spots ● View List of Parking Requests for the Parking Lot Assign/Retract Parking spot
10	Analysis and Design Document

Note:

1. **Applications that do not respect the imposed architectural and design patterns will not be accepted and will be graded with 2, irrespective to other implementations.**
2. **Operations implemented without a Command design Pattern will receive only up to 50% of the allocated points (if any).**