This document describes the **project assignment** for Telerik School Academy students in season "JavaScript development 2013/2014"

Project Description

Design and implement a SPA application with a client and a server:

- **Server exposing a REST API** implemented with Node.js and deployed to a public cloud
- **Client SPA application** implemented with AngularJS, KendoUI or any other JavaScript framework providing a MVC/MVVM architecture

General Requirements

Please define and implement the following assets in your project:

Requirements for the *REST API*

- Use **Node.js for the REST API**
  - Expose **RESTful** endpoints, i.e. with Express or any other framework
  - Host the application in **a public cloud**. The cloud can be **Heroku**, **Microsoft Azure**, or any cloud provider with **hosting for Node.js applications**
- Use a **NoSQL database**
  - The data storage of the application must be **a NoSQL database**. It can be **MongoDB**, **Apache CouchDB**, **Redis**, etc.
- The REST API should expose **only CRUD** operations over the database and **user login/register**
  - The business logic of the application **should be implemented on the SPA applications**

Requirements for the *SPA application*

- **Implement a MVVM/MVC** architecture with any framework that provides it
  - For MVVM you can use **KendoUI**, **Knockout.js**, **Knockback**, etc.
  - For MVC you can use **AngularJS**, **Ember.js**, **Sammy.js**, **Backbone.js**, etc…
- **Implement a UI for your application**
  - The UI must be **usable**
  - The US must be **responsive** for tablets/smartphones
- The **client application** should communicate with the **REST API from the server application** using **HTTP** requests
- The application must work on **Google Chrome 34**, **Mozilla Firefox 29, Internet Explorer 9 or above** and **Apple Safari 7**
- The **client application** must have **at least 5 pages** (routes)
  - Excluding **login**, **register** and **home** pages

Additional Requirements

- Follow the **best practices for OO design**: use data encapsulation, use exception handling properly, use inheritance and abstraction properly and follow the principles of strong cohesion and loose coupling
- Use a **source control system** by choice
  - Choose between Git, SVN or TFS

Optional Requirements

If you have a chance, time and a suitable situation, you might add some of the following to your project:

- **Unit** and **integration** testing
- **Backward compatibility** (make the application usable on browsers like IE8, IE7 and IE6)

Deliverables

Put the following in a **ZIP archive** and submit it (each team member submits the same file):

- The complete **source code**.
- Brief **documentation** of your project (2-3 pages). It should provide the following information (in brief):
  - Team name and list of team members
  - Project purpose – what problem do you solve?
  - The URL of your repository
  - The URL of the public cloud where the application is deployed
  - Any other information (optionally)
- Optionally provide a **PowerPoint presentation** designed for the project defense

Public Project Defense

Each team will have to deliver a **public defense** of its work to the other students and trainers. You will have **only 15 minutes** for the following:

- **Demonstrate** the application (very shortly).
- Show the **class diagram** (just a glance).
- Show the **commits logs** to confirm that team member have contributed.
- Optionally you might prepare a PowerPoint presentation (3-4 slides).

Please be **strict in timing**! Be **well prepared** for presenting maximum of your work for minimum time. Bring your own laptop. Test it preliminary with the multimedia projector. Open the project assets beforehand to save time. You have **15 minutes**, no more.