|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assignment 1 | | Project Summary | | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2017 | | |
|  | | | | |
| Project author | | | | |
| № | First name, last name | | E-mail | Face-to-face/ online |
| 1 | Denis Angelov | | denis.a.angelov@gmail.com | face-to-face |

|  |  |
| --- | --- |
| Project name | Web Interactive Map |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Driving a car pollutes the environment. More and more of its advocates and lovers solve this problem by replacing the car with bicycle. The Web Interactive Map was created to facilitate this group of people and their driving. This is a online system in which cyclists can share information about the status of the routes they run daily, explore new roads and signal irregularities. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies.  The main user roles (actors in UML) are:  • *Anonymous User* – can only view the map, find your location and view existing markers  • *Registered User* – can describe markers  • *Administrator - ?!?* |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse map** | The *users* can browse the information- map, routes and view described markers. Can measures distance between points on the map and can view its location. | All users |
| * 1. **Register** | *Anonymous User* can register(sign up) in the system by providing a valid e-mail address, first and last name, and choosing password. | *Anonymous User* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data*. | *Registered User* |
| * 1. **Add/Edit Marker** | *Registered User* can add/edit marker on the map*.* | *Registered User* |
| * 1. **View cycling paths** | *Anonymous User* and *Registered User* can view cycling routes on the map. | *Anonymous User, Registered User* |
| * 1. **Find location** | *Anonymous User* and *Registered User* can find location on the map with accuracy ?!? | *Anonymous User, Registered User* |
| * 1. **Calculate distance/area** | *Anonymous User* and *Registered User* can calculate distance between two point or area by a set of points. | *Anonymous User, Registered User* |
| * 1. **Change view of the map** | *Anonymous User and Registered User* can change view of the map- night view/ landscape etc. | *Anonymous User, Registered User* |
| * 1. **Simulate route** | *Registered User* can set route and visualize/simulate it on the map. | *Registered User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents welcome message to the user- anonymous or registered. | / |
| * 1. **Log in** | Page to log in. User must indicate username and password to login to the system. | */login* |
| * 1. **Sign up** | Allows an anonymous user to register to the system. | */signup* |
| * 1. **Map** | Presents the map of the system with information about cycling routes and described markers on it. | */map* |
| * 1. **About** | Presents information about the *Web Interactive map* project and his owner. | */about* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET all users and their’s data. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*. | */api/users/{userId}* |
| * 1. **Tests** | GET users (according to *User's Role* and identity) and POST new *Test* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). | */api/tests* |
| * 1. **Test** | GET, PUT, DELETE *Test Data* (including *Questions* and *Answers*) for *Test* with specified *testId*. | */api/tests/{testId}* |
| * 1. **Test Results** | GET *Test Results* (according to *User's Role* and identity) ) for *Test* with specified *testId*, and POST new *Test Result* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). | */api/tests/{testId}/results* |
| * 1. **Test Result** | GET, PUT, DELETE *Test Result* (according to *User's Role* and identity) for *Test* with specified *testId* and *Test Result* with specified *testResultId.* | */api/tests/{testId}/results/{testResultId}/* |