|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | Konstantin Netsov | | knetsov91@gmail.com | face-to-face |

|  |  |
| --- | --- |
| Project name | Favourizer |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| With the growth of techologies users have access to images, videos and other web resources from various devices. **Favourizer** gives the user possibility to organize in one place its preferable images, videos, articles that access from different devices. 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 see other users public resources.  • *Registered User* – can organize its resources(add, remove, modify)  • *Administrator* – can manage (create, edit user data and delete) all *Registered Users*  • *Friend(same as Registered User)* – can *view public and private resources of its friends.* |
|  |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse profile information** | *Registered User* can browse the information view and edit its profile | *Registered User* |
| * 1. **Login** | *Registered* User can login with facebook or google account or with site account. | *Anonymous User* |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data*. | *Registered User* |
| * 1. **Manage Users** | *Administrator* can create a new user using *Register UC*.  *Administrator can delete user.* | *Administrator* |
| * 1. **Organize resources** | User can group resources by different criteria (image, video etc)  User can upload resource to permanent storage(e.g Google drive) | *Registered User* |
| * 1. **Add/Edit Resource** | *User can add resources of different types(links, etc )*  *User can edit resource(e.g make it public/private )*  *User can remove resource.* | *Registered User* |
| * 1. **Manage friends** | *Registered User can add friend(other user) who can see private and public resources of the user*  *Registered User can remove friend.*  *Registered User can browse its friends resources.* | *Registered User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Registered *User see its “Board”(place for user's resources), Anonymous User see others users public resources* | / |
| * 1. **Resources** | Presents resources of Registered *User*. Offers abilities to create, read, update, delete (CRUD). | */resources* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data.* | */profile* |
| * 1. **Users** | Presents ability to manage (CRUD) Users and their User Data (available for Administrators only, as described in UCs). | */users* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **User** | GET, PUT, DELETE *User Data* for *Registered User* with specified *userId*. | */api/users/{userId}* |
| * 1. **User Friends** | GET users's friends by userId | */api/users/{userId}/friends* |
| * 1. **Friends** | POST,DELETE friend(with UserId2) of users with userId1 | */api/users/{userId1}/friends/{userId2}* |
| * 1. **Resource** | GET, PUT, DELETE Resource *Data*  for Resource with specified *resourcesId*. | */api/resources/{resourcesId}* |
| * 1. ***Resource Type*** | 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).  GET and POST resource type by its resourceId | */api/resources/{resourcesId}/{type}* |