|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2017 | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | honik | | face-to-face |

|  |  |
| --- | --- |
| Project name | Online Income/Expenses Management System |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The Internet age changed the way we earn and spend money. The financial management can be the difference between poor and very successful people. **The Online Income/Expenses Management System** provides its users with the ability to keep track on their account balance. In addition to that it allows users to create a saving plan in order for them to save even more money. 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:  • *Registrated User –* can add earnings and expenses and see reports about his account balance. Can also create saving plans  • *Premium Registrated User –* has all of the benefits of the normal users, but in addition to that, they can administrate users |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. By default all new registered users have *Registated User* role. | *All users* |
| * 1. **Log In** | All of the users can log in. | *All users* |
| * 1. **Change user data** | Every user can change his personal information | *All users* |
| * 1. **Add income and expenses** | Every user can add income and expenses | *All users* |
| * 1. **Add periodical transactions** | Every user can add periodical transactions | *All users* |
| * 1. **Edit/Delete income and expenses** | Every user can edit and delete income and expenses | *All users* |
| * 1. **Account history** | Every user can see the history of his spendings/earnings | *All users* |
| * 1. **Create/Edit/Delete savings plans** | The user can create his own savings plan | *All users* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Dashboard** | Presents the reports about all the account information | */* |
| * 1. **Add expenses/earnings** | Presents the interface to add expenses/earning | */transactions* |
| * 1. **Savings plan** | Presents the plan’s detail | */plans* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data*, as well as deregister from the system | */personal* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **User** | GET, POST, PUT, DELETE *User Data* for *User* with specified *userId*. | */api/users* |
| * 1. **Transactions** | GET users and POST Transaction(Id is auto-filled by the system and modified entity is returned as result from POST request). | */api/transactions* |
| * 1. **Authentication** | POST method for login authentication | */api/auth* |
| * 1. **Plan** | GET, PUT, DELETE *Savings Plan* (according to *User's Role* and identity) ) and POST new *Plans* (Id is auto-filled by the system and modified entity is returned as result from POST request). | */api/planDetail* |