|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2020 | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Ryuk | | face-to-face |

|  |  |
| --- | --- |
| Project name | Booking App |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| In near past, when we want to visit some city or spa resourt we use to call to a hotel’s keepers and ask the to reserve a room or two for us. The we need to write our personal data in some paper sheets that the keepers collect and it was a lot of administration. In our days, new technologies are very helpful for our daily routine. We will be blessed if we can plan our vacation and reserve some calm and chill place where we can stay. And this can be possible with our **Booking App.** System with simple user interface that allow users to make their “booking wish” come true. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies. Each view will have a distinct URL, and the routing between pages will be done client side using ***React Router***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization.  The main user roles (actors in UML) are:  • *Anonymous User* – can only view the information pages, prices, hotels and photos.  • Client (extends *Registered User*) – can choose offers, save them for later review, make books for a place and the keepers of the hotel can use it’s information to book a room and give the comfort when the User go to the hotel, the receptionist can just give him/her a key for the room after just see the personal id or digital sheet that confirm reservation.  • *Hotel keeper – can post photos, offers and change prices*  • *Administrator* (extends *Registered User*) – can manage all places and photos, like positioning in the app’s screen |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information and try sample tests** | The *User* can browse the information views (Home, Places’ List, About) in *Booking App*. | All users |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, payment methods and choosing password. By default, all new registered users have *Client* role.  *Administrator* can register new user by enetring its data. This user can be Hotel keeper and its offers or Administrator. | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Client* can view and edit her personal *User Data*.  *Administrator* can view and edit *User Data* of *Hotel keepers* and assign them *Roles*: *Hotel keeper*, or *Administrator*. | *Client Administrator* |
| * 1. **Manage Users** | *Administrator* can browse and filter users based on different criteria: first and last name, email, Role.  *Administrator* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC) or delete. | *Administrator* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents the introductory information for the purpose of the system as well as detailed instructions how to start using it. Prominently offers ability to register. | / |
| * 1. **User Registration** | Presents a view allowing the *Anonymous Users* to register in *Booking app*, as well as to take a look at the offers. | */register* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data.* | */personal* |
| * 1. **Hotel keeper Data** | Here Hotel keeper fill a form where they add their hotel, count of rooms and create page for a hotel | */post* |
| * 1. **Dashboard** | Presents in real time all Hotel keepers cards. | */dashboard* |
| * 1. **Users** | Presents ability to manage (CRUD) Users and their User Data (available for *Administrators* only, as described in UCs). | */users* |
| * 1. **About** | Presents information about the *Booking app* project and his owner. | */about* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions decribed in UCs. | */api/users/{userId}* |
| * 1. **Login** | POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests. | */api/login* |
| * 1. **Logout** | POST a logout request for ending the active session with Booking app. | */api/logout* |
| * 1. **Hotel Keeper** | Post information about it’s new offer in a hotel | */api/post* |