|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2019 | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Misterski | | face-to-face |
| 2 | Musakichka | | face-to-face |
| 3 | Tiramisu | | face-to-face |

|  |  |
| --- | --- |
| Project name | Mandzhichka |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Nowadays people often wonder what to cook, thus they mostly search for cooking recipe online in hope to come across the perfect madzhichka for them. Mandzhichka.bg provides them with a big variety of mandzhichki to choose from. In addition it allows the users to include their own cooking recipes to share with the hungry world, as well as view, rate, comment and add to favorites other’s cooking recipes. 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:  • *Unlogged User* – can only view the cooking recipes, without the ability to comment, rate, create recipes or add to favorites.  • *Logged User* – can view, comment, rate, create recipes and add to favorites, alongside with managing their profile and adding products to their shopping list. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse cooking recipes** | All users can browse cooking recipes | All users |
| * 1. **Search recipes** | *All users can search recipes by name and key words in description.* | *All users* |
| * 1. **Filter recipes** | *All users can filter the recipes by products, rating, preparation time or type.* | *All users* |
| * 1. **Login** | *Users can login with a created account or by using their google profiles.* | *Unlogged User* |
| * 1. **Register** | *Unlogged User* can register in the system by providing a valid e-mail address and choosing username and password | *Unlogged User* |
| * 1. **Forgotten password** | *Users will get an e-mail containing a new password sent to them, in case they have forgotten their password.* | *Unlogged User* |
| * 1. **Change User Data** | *Logged User* can view and edit their username and password. | *Logged User* |
| * 1. **Add/Delete recipes** | *Logged User can add/delete their own cooking recipes, as well as add/remove recipes to their favorites.* | *Logged User* |
| * 1. **Comment recipe** | *Logged User can add a comment to other recipes.* | *Logged User* |
| * 1. **Rate recipe** | *Logged User can rate recipes.* | *Logged User* |
| * 1. **Edit recipe** | *The recipe owner can edit their recipe, by changing the description, products or adding a picture* | *Recipe Owner* |
| * 1. **Add products to shopping list** | *Logged Users can add the products from a recipe to their personal shopping list, alongside with changing it. The shopping list is like a ToDo list and items can be scratched out of it.* | *Logged User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Contains a list of paged cooking recipes, which have a title, picture, summary and rating, options to filter them alongside with search-bar and option to log in | / |
| * 1. **Recipe info** | Contains the recipe name, long description, products, rating, comments. The products can be added to the shopping list, if the user has logged in | */recipes/{recipe.id}* |
| * 1. **Shopping list** | Has the products, added by the user. The products can be scratched out by the user, if they have obtained them. | */shopping-list* |
| * 1. **Edit recipe** | The owner of the recipe will be able to change the recipe data. | */recipes/{recipe.id}/edit* |
| * 1. **User Profile** | Lets the user view and change his user data. | */profile* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **Registration** | Presents a view allowing the users to register. | */register* |
| * 1. **Favorites** | Allows the user to view their favorite recipes. | */recipes/favorites* |
| * 1. **My Recipes** | Presents the user with a list of their personal recipes | */recipes/my* |
| * 1. **Add new recipe** | Provides the ability to add new recipe. | */recipes/my/add* |
| * 1. **Forgotten Password** | Presents a view allowing the users to request a new password. | */forgotten-password* |
| * 1. **About** | Presents information about the project and his owner. | */about* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | GET List all recipes sorted by rating. | */api?page={pageNum}* |
| * 1. **Recipe info** | GET, PUT, DELETE Recipe with specified *recipeId*, according to restrictions decribed in UCs. | */api/recipes/{recipe.id}* |
| * 1. **Shopping list** | GET Products in the shopping list. | */api/shopping-list* |
| * 1. **Edit recipe** | PUT Update recipe with specified recipe | */api/recipes/{recipe.id}/edit* |
| * 1. **User Profile** | GET User information and PUT to update the user information | */api/profile* |
| * 1. **Login** | POST Form to fill user’s data needed for login. | */api/login* |
| * 1. **Registration** | POST Form to fill user’s data needed for registration. | */api/registration* |
| * 1. **Favorites** | GET List of ordered by name user’s favorite recipes. | */api//recipes/favorites* |
| * 1. **My Recipes** | GET List of ordered by name user’s recipes. | */api/recipes/my* |
| * 1. **Add new recipe** | POST Form to fill in order to create new recipe. | */api/recipes/my/add* |
| * 1. **Forgotten Password** | POST Form to fill in order to request new password | /api/*forgotten-password* |