|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | **Web Application & Service Development with Spring Framework v5** **- 2020** | |
|  | | | |
| Project author | | | |
| № | Pseudonym | | Face-to-face/ online |
| 1 | Buba | | online |

|  |  |
| --- | --- |
| Project name | Cookery Book Manager |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The idea of the project is to provide each user the possibility to maintain their own cookery books with recipes. Each registered user can have their own cookery book, where he can write his own recipes with products. For convenience, he would also be able to list all products that are available in his house(in our Refrigerator), in this way he will always know which products he needs to buy, for example. Every user has access to everyone’s public cookery books.  The system will be developed using ***Spring 5 Application Development Framework***.It will implement web-based front-end client using ***Thymeleaf***  templates and ***jQuery*** or ***React.js/Vue.js/Angular*** JavaScript/TypeScript libraries for asynchronous page data updates. Each page will have a distinct URL, and the routing between pages will be done server side using ***SpringMVC/ WebFlux***. The backend will be implemented as a ***REST API***. The main user roles (actors in UML) are:  • *Anonymous User* – Can see other people’s cookery books with recipes, if they are public  • *Registered* *User(extends Anonymous User)* – Can have a Cookery book with Recipes(private or public). Can have also a refrigerator with available products.  • *Administrator* (extends *Registered* *User*) – manages (create, edit user data and delete) all *Users*, as well as all recipies and products |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse public cookbooks with recipes** | All users, including anonymous, can browse public user cookbooks and recipes | All users |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, username, and choosing password  *Administrator* can register new user by entering *user data.* | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data*  *Administrator* can view and edit *User Data* of all *Users* | *Registered User, Administrator* |
| * 1. **Manage Users** | *Administrator* can browse and filter users based on different criteria: first and last name, email.  *Administrator* can choose a *User* to manage, and can manage the chosen User - edit or delete.  *Administrator* can create a new user using the Administrator Panel. | *Administrator* |
| * 1. **Manage Cookery books** | Registered User can create their own cookery books  Administrator can manage other users’ cookery books | *Registered User, Administrator* |
| * 1. **Manage Recipes** | *Registered User* can create, read, update, delete his own recipes.  Administrator can manage other users’ recipes | *Registered User, Administrator* |
| * 1. **Manage Products** | *Registered User* can create, read, update, delete his own products  Administrator can manage other users’ products | *Registered User, Administrator* |
| * 1. **Manage Refrigerator** | *Registered User can manage his own Refrigerator with his currently available products* | *Registered User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents a list of all public books to all users (both anonymous and registered) | /books |
| * 1. **View public book** | Everyone can view certain public book | /books/{id} |
| * 1. **View public recipe** | Everyone can view certain public book’s recipe | /books/{id}/recipes/{id} |
| * 1. **View My Books** | Every registered user has access to their private own panel, where their cookery books are | */user/books* |
| * 1. **View certain book from my books** | User will be able to open a single cookery book from their page | */user/books/{id}* |
| * 1. **Cookery Book edit** | User can edit his own cookery book | */user/books/{id}/edit* |
| * 1. **View Recipe** | View certain recipe from your cookbook with id:=id | */user/books/{id}/recipes/{id}* |
| * 1. **Edit recipe** | Edit certain recipe from your cookbook with id:=id | */user/books/{id}/recipes/{id}/edit* |
| * 1. **Refrigerator** | Every user has Refrigerator – the place where he adds the products that he currently has in his own real refrigerator | */user/refrigerator* |
| * 1. **User Registration** | Presents a view allowing the *Anonymous Users* to register. | */register* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **User Profile** | Provides ability to view and edit personal data, edit personal information and provides fast access to user’s cookery books with recipes and refrigerator | */user* |
| * 1. **Users** | Presents ability to manage (CRUD) Users and their User Data (available for *Administrators* only). | */users* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId* | */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 sessionand invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Cookery Books** | GET,PUT,POST,DELETE cookery books (according to *User's Role* and identity) | */api/books* |
| * 1. **Recipe** | GET, PUT, POST, DELETE *Recipe* with specified *id*. | */api/recipes/{id}* |
| * 1. **Refrigerator** | GET, PUT, POST, DELETE *products in the refrigerator.* | */api/refrigerator* |