|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Web | |
|  | | | |
| Project author | | | |
| № | Name | | Face-to-face/ online |
| 1 | Иван Арабаджийски | | Online |
| 2 | Теодор Кънев | | Online |

|  |  |
| --- | --- |
| Project name | IT-Library |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Уеб приложение, което цели да помогне на хората от ФМИ да намират учебниците и рефератите, от които имат нужда. Могат да ги взимат, за да ги четат за определен период от време, да ги цитират, за периода от време, в който са били взети или да качват нова колекция от книги, автор на която е потребителят. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information** | The *User* can browse the information views (Home, About, Contacts). | *All users* |
| * 1. **Register** | *Anonymous User can register in the system by providing a valid e-mail address, first and last name, and choosing a password.*  *Administrator* can register new users by entering *User Data*. | *Anonymous user*  A*dministrator* |
| * 1. **Login/Logout** | *Registered user /Admin* can login and logout from the system. | *Registered user* |
| * 1. **Modify info** | *Registered users* can view books and borrow them for a limited period of time. | *Registered user* |
| * 1. **Import/export** | *Registered users* can import/export books. | *Registered user* |
| * 1. **Manager profile** | *Registered users* can manage their own profiles and look at their books. | *Registered user* |

|  |  |  |
| --- | --- | --- |
| 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 *the IT-Library.*  Sends a POST request to “signup.php” with the form data:  username: string, firstName: string, lastName: string, password: string | */signup* |
| * 1. **Login** | Presents a view allowing the users to login.  Sends a POST request to “login.php” with the form data:  username: string, password: string | */login* |
| * 1. **Library** | Presents a view allowing the user to browse books, papers and other users’ profiles.  Sends a GET request to “root.php” with query parameters:  title: string, titleExactMatch: Boolean, author: string, authorExactMatch: Boolean, tags: string, sortBy: [“author”, “title”, “date”], order: [“asc”, “desc”] | */library* |
| * 1. **Add collection** | Sends a POST request to “addBooks.php” to add a collection of resources (books and papers).  username: string, csv: CSV file | */addBooks* |
| * 1. **About** | Presents information about the *IT-Library* project and its owners. | */about* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (PHP Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | ???  I prefer to leave that after we are done with the project at all.  GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *the backend* and modified entity is returned as result from POST request). Available only for *Administrators*. | */api/users* |
| * 1. **User** | ???  Same as 4.1  GET, PUT, DELETE *User Data* for *User* with specified *userId.* | */api/users/{userId}* |
| * 1. **Login** | POST *User Credentials* (username and password). Will confirm that such user exists with the same password and will return the following   * {success:true} if everything is alright and session variables will be updated. May be redirected to profile page after that. * {success:false, message:’sample error message’} in case password is wrong or username doesn’t exists. | */api /signin* |
| * 1. **Logout**   **4.5 Sign up** | POST a logout request for ending the active session. Session will be closed and it can be redirected to home screen without any further communication.  POST a singup request with all the data in the field – VALIDATED (the passwords are same) and backend will validate if such username is free and if it is, then registration is done with returning :   * If everything is fine then {success:true} * If username is taken then {success:false,message:’simple error message’} | */api /logout* |