Yellow background text = Not Yet Implemented

|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2021 | |
|  | | | |
| Project author | | | |
| F№ | Name | | Email |
| 61971 | Kaloyan Dochev | | Kaloyan\_dochev4fmi@abv.bg |

|  |  |
| --- | --- |
| Project name | Content Management System |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Platform for easy management of content displayed to user and used by publishers.  The product visible to outside users should be something akin to a news page.  The product visible to the clients should essentially be a system with which to choose what content types (basically different pages and sections of pages) to be displayed. Information views are dynamically managed to be visible or not and are comprised of modules such as main menu, article feed, main page text etc. Said modules can be reorganized in a page by admins. Admins can edit the functionality of modules.  The main user roles (actors in UML) are:  • *Anonymous User* – can only view the information pages and read articles  • *Registered User* (extends *Anonymous User*) – can comment on articles.  • *Publisher* (extends *Registered User*) – can create articles.  • *Administrator* (extends *Registered User, Publisher*) – can manage content. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information and articles** | The *User* can browse the information views (The ones that are set to visible by the Admin) in *CMS*, and cancan browse articles on them. | All users |
| * 1. **Register** | *Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password and will become a Registered User.  *Administrator* can register new User by entering *User Data* as a Registered User or as a Publisher | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Registered User* can view and edit their personal *User Data* as well as enter more*.*  *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, Role.  *Administrator* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC) or delete.  *Administrator* can create a new user using *Register UC*. | *Administrator* |
| * 1. **Manage Content** | *Administrator* can choose which Information Views to be accessible.  *Administrator* can choose which modules Information Views are comprised from. | *Administrator* |
| * 1. **Write Comment** | Users can write comments with the specified role next to their name. | *Registered User*  *Publisher*  *Administrator* |
| * 1. **Write Article** | Write an article in a certain page. | *Publisher*  *Administrator* |
| * 1. **Manage visible pages** | Admins can reorganize what pages are visible in the main/top/side menu module and as a whole. | *Administrator* |
| * 1. **Manage page content** | Admins can manage what content is posted on a page as in make an article visible or not or even delete article. | *Administrator* |
| * 1. **Edit page** | Admins can edit what modules a page is made of and how there are laid out. Such modules can be main menu, article feed, comment section etc. | *Administrator* |
| * 1. **Edit component** | Admins can edit how a component behaves. | *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. **View Page** | View a section if it is marked as visible by administrator. Navigable through a main menu if such a module links to them. | */{Section}* |
| * 1. **View Article** | View certain section content. | */{Section }/{Content}* |
| * 1. **User Registration** | Presents a view allowing the *Anonymous Users* to register in *CMS*, as well as to enroll to a *Student Group* by presenting a *Token Key* for that group. | */register* |
| * 1. **Login** | Presents a view allowing the users to login. | */login* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data*, as well as deregister from *CMS*. | */personal* |
| * 1. **Dashboard** | For Admins only.  Overall control  Links to edit | */dashboard*  */Manager* |
| * 1. **Reorder Main Nav** | Allows reordering of the navigable sections and adding new sections to the navigation menu. | */Manager/sections-nav-preview* |
| * 1. **List All Sections** | Lists all sections with edit buttons that take to the Section Form view. | */Manager/list-sections* |
| * 1. **Add Section** | Add a new section from the Section Form view. | */Manager/add-section/{Section}* |
| * 1. **Edit Section** | Edit an existing section from the Section Form view | */Manager/edit-section/{Section}* |
| * 1. **Add Component** | Add a new section from the Component Form view. | */Manager/addt-*component*/{*Component*}* |
| * 1. **Edit Component** | Edit component from the Component Form view. | */Manager/edit-*component*/{*Component*}* |
| * 1. **Add Content** | Allows adding content through the Content Form View.  Uses “What You See Is What You Get”(WYSIWYG) to achieve editing like in MS Word. | */Manager/add-content/{Content}* |
| * 1. **Edit Content** | Allows editing content through the Content Form View. | */Manager/edit-content/{Content}* |
| * 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 *CMS* 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 *CMS* 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 *CMS,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Sections** | GET *User Data* for all users, and POST new *User Data* (Id is auto-filled by *CMS* and modified entity is returned as result from POST request).  Available only for *Administrators*. | */api/sections* |
| * 1. **Section** | GET, PUT, DELETE *Section Data* for *Section* with specified *sectionId*, according to restrictions.  Available only for *Administrators*. | */api/sections/{sectionId}* |
| * 1. **Components** | GET *Component Data* for all components, and POST new *Component Data* (Id is auto-filled by *CMS* and modified entity is returned as result from POST request).  Available only for *Administrators*. | */api/components* |
| * 1. ***Component*** | GET, PUT, DELETE *Component Data* for *Component* with specified *componentId*, according to restrictions.  Available only for *Administrators*. | */api/components /{componentId}* |
| * 1. **Content** | GET *Content Data* for all users, and POST new *Content Data* (Id is auto-filled by *CMS* and modified entity is returned as result from POST request).  Available only for users depending on role restrictions of said content. | */api/content* |
| * 1. **Content** | GET, PUT, DELETE *Content Data* for *Content* with specified content*Id*, according to restrictions.  Available only for users depending on role restrictions of said content. | */api/content/{*content*Id}* |