|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | **Web Application & Service Development with Spring Framework v5** **- 2022** | |
|  | | | |
| Project author | | | |
| № | Names of Participants | | FN |
| 1 | Georgi Boychev | | 62434 |

|  |  |
| --- | --- |
| Project name | Agile Sprints |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The main goal of the project is to implement a web application allowing the Users (Developers, Product Owners, and Administrators) to define and manage new Projects, Sprints and Tasks, as well as to report their progress. The system will be developed using the ***Spring 5 Application Development Framework***.It will implement a web-based front-end client using ***Thymeleaf*** templates. Each page will have a distinct URL, and the routing between pages will be done server side using ***SpringMVC***. The backend will be implemented as a ***REST/JSON API*** using JSON data serialization. There will be also a real-time event streaming from the server to the web client using ***WebFlux*** and ***Server Sent Events (SSE)/WebSocket*** in order to allow the users to receive and update information about the progress of the projects, sprints and tasks in real time. The main user roles (actors in UML) are:  • *Administrators* – able to manage all User's data, except their passwords; able to add new Projects, as well as to complete them after the ProjectResult has been reported by ProductOwner  • *Product Owner* – able to manage their own personal data and change their passwords; able to add new Sprints and Tasks to existing Projects, as well as to verify/report completion of TaskResults and SprintResults for different Tasks and Sprints completed by Developers  • *Developer - able to manage their own personal data and change their passwords; able to add new Sprints and Tasks to existing Projects, as well as to verify/report completion of TaskResults and SprintResults for different Tasks and Sprints completed by Developers* |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **See all ongoing projects** | Any user should be able to see details of ongoing projects | All users |
| * 1. **Add new project/ mark project as completed** | *The administrators can add new projects and mark projects as completed after they have been reported as completed by the Product Owner* | *Administrator* |
| * 1. **Update project** | *A product owner can update the status of a project he is managing* | *Product Owner* |
| * 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. | *Administrator* |
| * 1. **Add new Sprint/Task** | *Product Owners and Developers can add new Sprints and Tasks to existing Projects* | *Product owner, Developer* |
| * 1. **Update Sprint/Task status** | *Product Owners and Developers can update the status of the Sprints and Tasks they are assigned to.* | *Product owner, Developer* |
| * 1. **Report completion of Sprint/Task** | *Developers can report that they have completed a task that they have been assigned.* | *Developer* |
| * 1. **Verify completion of Sprint/Task** | *Product Owners can verify completion of Sprints/Tasks that developers have reported as completed.* | *Product Owner* |
| * 1. **Change names/password** | *All users can change their first names, last names and passwords.* | *All users* |
| * 1. **Add/remove own contact** | *All users can add or remove contacts from their contact lists.* | *All users* |
| * 1. **Edit name/contact list of other user** | *The Administrator can edit other users’ information, except for their passwords*. | *Administrator* |
| * 1. **Change Status of a User** | *The administrator can change the status of a user - ACTIVE, CHANGE\_PASSWORD*, *SUSPENDED, or DEACTIVATED* | *Administrator* |
| * 1. **Add/remove users** | *The administrator can add or remove users* | *Administrator* |
| * 1. **Find Project by title/ tag** | *All users can search for projects and filter them by tag and/or title* | *All users* |
| * 1. **Check list of assigned/ completed tasks of developer** | *All users can see which developers have been assigned which tasks and the record of completed tasks of each developer.* | *All users* |
| * 1. **Check list of assigned/ completed projects of a product owner** | *All users can see which product owners have been assigned which projects and the record of completed projects of each product owner.* | *All users* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data* (Id is automatically generated and a modified entity is returned as result from a POST request). Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *parameters*, according to restrictions described in the use cases listed above. | */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. **Projects** | GET *Projects*, and POST new *Project* (Id is automatically generated and modified entity is returned as result from POST request), according to *User's Role* and identity security restrictions. | */api/projects* |
| * 1. **Project** | GET, PUT, DELETE *Project* with specified projectId. | */api/projects/{projectId}* |
| * 1. **Tasks** | GET Tasks and POST new *Task* (Id is automatically generated and a modified entity is returned as result from a POST request). | */api/tasks* |
| * 1. **Task** | GET, PUT, DELETE *Task* for *Project* with specified *taskId*. | */api/tasks/{taskId}* |
| * 1. **Sprints** | GET *Sprint*s and POST new *Sprint* (Id is automatically generated and a modified entity is returned as result from POST request). | */api/sprints* |
| * 1. **Sprint** | GET, PUT, DELETE *Sprint* with *sprintId.* | */api/sprints/{sprintId}* |
| * 1. **Sprint Result** | Get Sprint Results with specified sprintId | */api/sprint-results/{sprintId}* |
| * 1. **Task Result** | Get Task Results with specified taskId | */api/task-results/{taskId}* |
| * 1. **Project Result** | Get Project Results with specified projectId | */api/project-results/{projectId}* |