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

|  |  |
| --- | --- |
| Project name | University courses manager |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| With the digitalization of the modern world every educational institution needs a good and reliable system for management of the courses and the communication between the teachers and students.  This system for University courses management will give the following abilities to the each user by a role:   * Firstly, the **administrator** will be granted with the ability to manage other users` accounts. * Secondly, it will give the **teachers** the ability to add courses * Thirdly, i will give the **students** the ability to enroll in them. * Fourthly, it will provide the **teachers** with the ability to give final grades to the students. * Also, there will be a chat, where they will be able to communicate with each other.   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. There will be also a real-time event streaming from the server to the web client using ***Socket.IO*** and ***Server Sent Events (SSE)/WebSocket*** in order to allow the Instructor to monitor test completion progress of students in real time. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse through the available courses** | All the user roles can access the courses data read-only. | All users |
| * 1. **Add a course** | *Teacher will be able to add courses, which have fields such as:  - Students limit*   * *Name* * *Description* * *Targeted specialty* * *Entrance test - Y/N etc.* | *Teacher* |
| * 1. **Edit a course** | *Teachers will be able to edit courses which are theirs.* | *Teacher* |
| * 1. **Remove a course** | *Teachers will be able to remove courses which are theirs if the course has not started yet.* | *Teacher* |
| * 1. **Send messages through chat** | *Every user will have the ability to chat with each other.* | *All users* |
| * 1. **Set grades** | *Teacher will have the ability to give final grades to the students.* | *Teacher* |
| * 1. **Enroll course** | *Students will be able to enroll in available courses.* | *Student* |
| * 1. **Cancel course enrollment** | *Students will be able to cancel their course enrollments* | *Student* |
| * 1. **Cancel enrollment of a student** | *Teacher will be able to remove students from his courses before the course has started.* | *Teacher* |
| **2.10. Add user** | *Administrators will be able to add teachers and students.* | *Admin* |
| **2.11. Delete user** | *Administrator will be able to remove a user which removes all the courses if user`s role is “Teacher” and the enrollment if user`s role is “Student”.* | *Admin* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Main page where you users can choose to login or to register. | / |
| * 1. **All courses** | Page where are the courses are shown, with options for enrollment(student) or edition/removal(teacher) | */courses* |
| * 1. **Course edition** | Course edition | */editCourse* |
| * 1. **Course addition** | Course addition | */addCourse* |
| * 1. **User Registration** | Form for user registration | */register* |
| * 1. **Login** | Form for login | */login* |
| * 1. **List of the students currently in a specified course** | Show all students enrolled in a course. | */courseStudents* |
| * 1. **Grade addition** | Presents a form for a grade addition | */addGrade* |
| * 1. **Grade edition** | Presents a form for a grade edition | */editGrade* |
| * 1. **Chat window** | Presents a list with the users and a chat currently open. | */chat* |

|  |  |  |
| --- | --- | --- |
| 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 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*. | */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,and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Courses** | GET and POST courses. | */api/courses* |
| * 1. **Courses modification** | GET, PUT, DELETE *Courses* by specified groupId | */api/courses/{groupId}* |
| * 1. **Grades** | GET, POST grades for the students in the course by specified courseId and studentId. | */api/grades/{studentId}/{courseId}* |
| * 1. **Grades modification** | GET, PUT, DELETE *for grades modification or deletion by studentId and courseId.* | */api/tests/{courseId}/{studentId}* |
| * 1. **All grades** | GET *All grades for a course by courseId* | */api/grades/{courseId}* |
| * 1. **Messages** | GET, POST messages between users by {loggedId} and {addresseeId} | */api/chat/{loggedId}/{addresseeId}* |