|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Multimedia Technologies with Angular and TypeScript - 2021 | |
|  | | | |
| Project author | | | |
| № | Name | | FN |
| 1 | Galya Dodova | | 45616… |

|  |  |
| --- | --- |
| Project name | Trivia Maker |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| **Trivia** is information and data that are considered to be of little value. It can be contrasted with general knowledge and common sense. Actually, trivia quizzes are not for educational purpose, they are just for fun. It does not mean that people cannot learn something interesting from them. This app gives a chance to users to solve tests, have fun and win prices. In addition to that it allows users to register, and administrators to manage them. The system will be developed as a *Single Page Application (SPA)* using ***Angular*** 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 ***Angular 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 Playerss to see how people solve their tests in real time during competitions. The main user roles (actors in UML) are:  • *Anonymous User* – can only view the information pages and try few sample tests without saving test results. No references after finishing the test.  • *Ordinary User* or Player (extends *Registered User*) – can choose a *Test* to complete – *Test Results* are saved automatically on test completion. List of references to learn about this topic. Every user can create tests, *put references,* see the *Test Results,* and monitor progress on tests in real time.   * Premium user- have access to the premium tests and competitions.   • *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, as well as *Player Competitions*, *Tests* and *Test Results*. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Browse information and try sample tests** | The *User* can browse the information views (Home, Tests, About) in *OKTS*, and can choose to try sample tests using the *Complete Test UC* . | 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. By default, all new registered users have *User* role. The user can provide a *Token Key* in order to get premium access. Happens when they pay the tax.  *Administrator* can register new by entering *User Data* and choosing a Role (*Player, Premium player*, or *Administrator*). | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data* as well as enter *Token Keys*.  *Administrator* can view and edit *User Data* of all *Users* and assign them *Roles*: *Player, Players*, or *Administrator*. | *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 Player Competitions** | *Administrator* or *Players* can browse and filter *Quiz Competitions* based on different criteria: name of category*,* creation date, etc.  *Administrator* or *Players* can create a new *Quiz Group,* and assign a *Token Key* for automatic *Quiz* enrollment*.*  *Administrator* can choose a *Quiz Group* to manage, and can add or remove *people* to the *Group,* or delete it if necessary. | *Administrator, Players* |
| * 1. **Manage Tests** | *Players* can make her/his *Tests*, add new *Test* using *Add/Edit Test UC*, and delete a *Test*, as well as view the *Player's Test Results* for her/his own *Tests* and for particular *Quiz Competitions* assigned to the *Test*. Tests are interactive and answers could be pictures or even videos. A lot of the test are can be games.  *Administrator* can browse test of all *Players*, edit and delete them. | *Players, Administrator* |
| * 1. **Schedule a Test** | *Players* can make her/his *Tests*, add new *Test* using *Add/Edit Test UC*, and delete a *Test*, as well as view the *Player's Test Results* for her own *Tests*.  *Administrator* can browse test of all *Players*, edit and delete them.  There will have public tests for everyone and protected tests for users and premium users. | *Players, Administrator* |
| * 1. **Add/Edit Test** | *Players* or *Administrator* specifies/edits *Test* meta-data such as: test name, category, description, tags, and chooses/creates a *Competitions* for the *Test* using *Manage Competitions UC*. *Players* or *Administrator* creates/edits *Questions* using *Create/Edit Question UC,* and can delete chosen *Question. Players* or *Administrator* saves the edited *Test.* | *Players, Administrator* |
| * 1. **Add/Edit Question** | *Players* or *Administrator* specifies/edits *Test's Question Type, Question Content*, and *Answers* providing weights for each answer (default weight is 0). | *Players, Administrator* |
| * 1. **Complete Test** | *Players* browses tests available and can sort and filter them using different test metadata fields. *Player* or *Players* chooses a test to complete. *OKTS* presents the *Test Questions* one by one to the *Player* or *Players*, who answers them. A full view of the test is possible if the player wants to see all questions at once. The *Answers* of each *Question* are saved in real time on the server*.* On *Test* completion the *Test Results* are saved by *OKTS.* | *Player, Players* |
| * 1. **Monitor Test Progress** | *Players* or *Administrator* monitors in real time the progress of *Players during competitions* on completion of test questions using a *Dashboard* that presents Player*’* progress on active *Tests* in real time. | *Players, Administrator* |
| * 1. **Browse Test Results** | *Player* can browse her own completed *Test Results*.  *Players* can browse the *Test Results* for *Tests* created by her/him and for particular *Player Competitions* assigned to the *Test*. After finishing the test, the player get can see references added by the Players. There are animations congratulating for right answer or making sad noises for wrong answer. Also the winners of a competition can receive badges and specialized avatars to their profile. | *Player, Players* |

|  |  |  |
| --- | --- | --- |
| 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. **Tests** | Presents tests available according to *User's Role* and category. Offers abilities to browse, choose, create, read, update, delete (CRUD) Tests, as defined by *UCs* (for *Administrators* and *Playerss* only). | */tests* |
| * 1. **Complete Test** | Presents tests available to particular Player (according to *Player Group* participation) allowing him/her to choose a test and complete it. | */complete-test* |
| * 1. **Test Results** | Presents *Test Results* available according to *User's Role* and identity as defined by *UCs*. | */test-results* |
| * 1. **User Registration** | Presents a view allowing the *Anonymous Users* to register in *OKTS*, as well as to enroll to a *Player 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*, enroll to more *Player Competitions*, subscribe for premium version as well as deregister from *OKTS*. Also shows earned badges and avatars. | */personal* |
| * 1. **Player Competitions** | Presents ability to browse, filter and manage *Player Competitions (*for *Administrators* and *Players),* and assign Players to them (for *Administrators* only), as described in UCs. | */competitions* |
| * 1. **Dashboard** | Presents in real time the Players’ progress on active Tests, subject to Role restrictions described in UCs. | */dashboard* |
| * 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 *OKTS* 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 *OKTS* 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 *OKTS,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Player Competitions** | GET *Player Competitions*, and POST new *Player Group* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request), according to *User's Role* and identity security restrictions. | */api/competitions* |
| * 1. **Player Groups**   **+collection level** | GET, PUT, DELETE *Player Group* (including assigned Players) for *Player Group* with specified *groupId*. | */api/groups/{groupId}* |
| * 1. **Tests** | GET users (according to *User's Role* and identity) and POST new *Test* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). | */api/tests* |
| * 1. **Test** | GET, PUT, DELETE *Test Data* (including *Questions* and *Answers*) for *Test* with specified *testId*. | */api/tests/{testId}* |
| * 1. **Test Results** | GET *Test Results* (according to *User's Role* and identity) ) for *Test* with specified *testId*, and POST new *Test Result* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). Will have a chart diagram. | */api/tests/{testId}/results* |
| * 1. **Test Result** | GET, PUT, DELETE *Test Result* (according to *User's Role* and identity) for *Test* with specified *testId* and *Test Result* with specified *testResultId.* | */api/tests/{testId}/results/ {testResultId}/* |
| * 1. **Active Tests** | SSE event streaming of Players’ progress on currently active *Tests* (separate event pushed for each *Question Answer*), according to *User's Role* and identity security restrictions. | */api/active-tests* |