|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assignment 1 | | Project Summary | | |
| Course | | Fullstack Application Development with Node.js + Express.js + React.js - 2017 | | |
|  | | | | |
| Project author | | | | |
| № | First name, last name | | E-mail | Face-to-face/ online |
| 1 | Bojidar Milanov | | bojidar\_milanov@mail.bg | face-to-face |

|  |  |
| --- | --- |
| Project name | Bulls and Cows (Extended Edition) |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Messiah is generating some number and you must recognize it by typing different numbers and look the results. The number can be with 4,5 or more digits – depends on the game type you choose. When your number contains digit that is same on the same position like in my number - you have a BULL. When your number contains a digit that is contained in mine too, but on different position - is a COW. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies.  The main user roles (actors in UML) are: • *Anonymous Player* – can play every type of game, but his scores are not saved. Can read the “How to play” section.  • *User* (extends *Registered User*) – can play every type of game, read the manual and his scores are saved in database for the “HIGHSCORE” page. *User* can leave comments about the game and maintenance.  • *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, as well as *Game types* and *Game Results*. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Read the “how to play” section** | The *User* can read the “How to play” section. | 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. All new registered users have *Player* role.  *Administrator* can register new by entering *User Data* and choosing a Role (*Player* or *Administrator*). | *Anonymous User, Administrator* |
| * 1. **Change User Data** | *Registered User* can view and edit her personal *User Data*.  *Administrator* can view and edit *User Data* of all *Users* and assign them *Roles*: *Player* 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 Game Types** | *Administrator* c add/edit/delete game types, can hide/show them from users too. Can set timers for different types. | *Administrator* |
| * 1. **Start the GAME** | *Everyone can start the “Bulls and Cows” game. Depend on the game type boxes for digits are shown on the screen and timers are set. Pretty front end part for clients. Working on mobile devices with no problems. On every turn the attempt and result are shown on the screen.* | *Anonymous Player, Registered User, Administrator* |
| * 1. **Game Over** | *When the time* *specified by the administrator runs out the game is lost, alert on the screen appears and after that the player is returned to the home view.* | *Anonymous Player, Registered User, Administrator* |
| * 1. **Complete The GAME** | *When player recognize the generated number in time there is a “Success” message with buttons “HIGHSCORE” and “NEW GAME”. If he is not Registered User the screen offers him to register for saving his scores too. If he is Registered User his scores (attempts + time) are saved.* | *Anonymous Player, Registered User, Administrator* |
| * 1. **HIGHSCORES view** | *Everyone can enter the “HIGHSCORES” view and can see the best 10 successful games.* | *Anonymous Player, Registered User, Administrator* |
| * 1. **Tournaments** | *Registered users can play in tournaments and can see the tournament’s score* | *Registered User, Administrator* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents a page with buttons that refers to different screens and information about the creator and game. | / |
| * 1. **Game** | Presents page of the game itself. | */* |
| * 1. **High scores** | Presents page with information about the best successful games. | */* |
| * 1. **Tournaments** | Presents page of all active tournaments | */* |
| * 1. **Admin** | Presents page with sections where administrator can enter – *Game* and *Users*. If administrator is not logged in that page returns the client to Login screen. | */* |
| * 1. **Login** | Presents login page where registered users can enter or access the register screen | */* |
| * 1. **Register** | Presents page with personal information fields – first name, last name, email (unique) and passwords | */* |
| * 1. **Editing games** | Page where administrator can add/edit/delete game types and set timers | */* |
| * 1. **Editing users** | Page where administrator can edit/delete users | */* |
| * 1. **User information** | Page where registered user can edit his information (except email) and can see his game history | */* |

|  |  |  |
| --- | --- | --- |
| 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 *the system* and modified entity is returned as result from POST request). Available only for *Administrators*. | */*getAllUsers |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*. | */*delete\_user, /update\_user, /register\_user |
| * 1. **Game types** | GET all game types generated by *Administrator* and return them for players. | */*getGameTypeByName, /delete\_game\_type, /update\_game\_type, /insert\_game\_type, /getAllGameTypes, /game\_type, /getAllActiveGames |
| * 1. **Generate number** | POST game type and client IP, generate number for the game and store it. | */*generate\_number |
| * 1. **Check number** | Client POST number and compare it with the generated one. Return success (0,1), bulls (number of bulls) and cows (number of cows). Update tries and time for this game (for further score storing) | */*check\_number |
| * 1. **Tournaments** | Operations with tournaments | */*getAllTournaments , /getAllActiveTournaments, /insert\_tournament, /update\_tournament, /delete\_tournament, /getTournamentScores |
| * 1. **Highscore** | GET top 10 successful games | */*getHighscore |

Maybe there will be more when I make it…  
Best regards.