Assignment 1	Project Summary
Course	Angular 2 and TypeScript Web Application Development

Project author			
№	First name, last name	E-mail	Face-to-face/ online
1	Stilyan Mladenov	stilyan.mladenov@gmail.com	face-to-face
2	Daniela Postolova	daniella.postolova@gmail.com	face-to-face

Project name	Chess game

1. Short project description (Business needs and system features)

The project is a client-server application for real-life chess playing.

It offers individual game rooms and matches history.

The main functionality of the system is accessible only for registered users. A user that is not registered can only see the home page, which describes the application.

The system uses Auth0 identity server for authentication and offers its users email-password registration as well as registration using external provider (Facebook, Google).

Registered users can view a list with existing rooms, join one and play. They also can create their own rooms and wait others to join. Each room provides a chessboard for the game and chat for conversations between players. The app makes the game easy and intuitive with allowing only valid moves. The real-life client-server communication is realized with socket.io library.

The user also has the ability to see a history of all played matches. He can analyze all past games by going back and forward through the selected match, and follow all moves.

The application provides user statistics for past matches like count of loses and winnings.

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
2.1. App description	The *User* has the ability to see the home page containing brief description of the application.	All Users
2.2. Register	An *Anonymous User* can register in the system using email-password authentication or an external provider like Facebook and Google.	*Anonymous User*
2.3. Rooms	*Registered User* has access to the room scene where he can find a list with all currently open rooms.	*Registered User*
2.4. Join Room	*Registered User* can join one of the existing rooms, which are listed in the rooms scene.	*Registered User*
2.5. Create Room	*Registered User* can create a new room and wait other player to join.	*Registered User*
2.6. Play Game	*Registered User* can play chess game after joining a room.	*Registered User*
2.7. Chat	*Registered User* can send and receive messages after joining a room through the chat provided by the system. Only the other player in the room sees the messages.	*Registered User*
2.8. See history	*Registered User* can past matches and the results of them.	*Registered User*

2.9. Go through a past game	*Registered User* can go back and forward through a past match to analyze the game while following each move.	*Registered User*
2.10. Statistics	*Registered User* can see statistics for played matches like loses and winnings count.	*Registered User*

3. Main Views (SPA Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Home	Presents the introductory information for the purpose of the system as well as detailed instructions how to start using it. Offers ability to register.	/home
3.2. Rooms	Presents a list of currently open rooms and gives opportunity to join or create one.	*/rooms*
3.3. Room	Presents the main game view with chessboard where both players make their moves. Provides a real-time chat between both players.	*/rooms/:id*
3.4. Matches	Presents matches history and allows selecting a match for detailed view.	*/matches*
3.5. Match	Presents a past match and allows going back and forward through the game.	*/matches/:id*
3.6. Profile	Presents information for user's loses, winnings and other matches statistics.	*/profile*
3.7. About	Presents information about the project and its creators.	*/about*

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. User	GET, POST, PUT, DELETE, PATCH, BULK *User Data* for user with the given id.	*/api/users/:id*
4.2. User statistics	GET, POST, PUT, DELETE statistics information about user with given id.	*/api/users/:id/statistics*
4.3. Matches	GET, POST, PUT, DELETE, PATCH, BULK all past matches.	*/api/matches*
4.4. Matches per User	GET, POST, PUT, DELETE, PATCH, BULK all matches played by given user.	*/api/matches?user=:id*
4.5. Match	GET, POST, PUT, DELETE, PATCH the match with the given id.	*/api/matches/:id*
4.6. Rooms	GET, POST, PUT, DELETE, PATCH, BULK all currently open rooms waiting for someone to join.	*/api/rooms*
4.7. Join Room	PUT existing room by adding a player.	*/api/rooms/:id*
4.8. Live Games	GET/POST/PUT/PATCH/BULK all currently live games.	*/api/livegames*
4.9. Move Events	GET/POST/PUT/PATCH/BULK all move events for a match with given id.	*/api/moves?matchId=…*
4.10. Message Events	GET/POST/PUT/PATCH/BULK all message events for a match with given id.	*/api/messages?matchId=…*

4.11. Clock Events	GET/POST/PUT/PATCH/BULK all timer events for a match with given id.	*/api/timerevents?matchId=…*