Assignment 1	Project Summary
Course	Fullstack Application Development with Node.js + Express.js + React.js - 2020

Project author		
№	Pseudonym	Face-to-face/ online
1	earmenov	face-to-face

Project name	Travel Playlist Generator

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

A user travellling from A to B wants to have something to listen to during the duration of the travel. The user wants to generate a track list based on his musical tastes (the user selects genres from a predefined list). An algorithm, which uses external service as track sample data, generates the playlist. Playlist Generator calculates the travel duration time between the starting and destination locations and combines tracks chosen randomly in the specified genres, until the playing time roughly matches the travel time duration. Rounding of +/- 5 minutes is allowed. The generated playlist is saved under this user's profile and they can start listening to it. 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. The main user roles (actors in UML) are:

- *Anonymous User* – can only browse the playlists and see the tracks list and details of them. This includes the application start page, the user login and user registration forms.

- *Registered User* – the web application provides the users the ability to generate new playlists, control the generation algorithm, edit or delete their own existing playlists. Editing existing playlists is limited to changing the title or associated genre tags, but does not include editing of the tracklist.

- *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, as well as *Playlists*.

2. Main Use Cases / Scenarios		
Use case name	**Brief Descriptions**	**Actors Involved**
2.1. Browse generated playlists	The *User* can browse the application start page with generated playlists, the user login and user registration forms.	All users
2.2. Register	*Anonymous User* can register in the system by providing a valid e-mail address, first name, and choosing password. By default, all new registered users have *User* role.	*Anonymous User*
2.3. 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*: *User* or *Administrator*.	*Registered User, Administrator*
2.4. Manage Users	*Administrator* can browse and filter users based on different criteria: first 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*
2.5. Manage Playlists	*Administrator* or *User* can generate new playlists, control the generation algorithm, edit or delete their own existing playlists. Editing existing playlists is limited to changing the title or associated genre tags, but does not include editing of the tracklist (e.g. removing or adding individual songs).	*Administrator, Registered User*
2.6. Browse Playlists	*Registered User and Administartor* can browse her/his *Playlists*, as well as the Playlists of all other users in the app and listen to their tracks.	*Registered User, Administrator*
2.7. Play Tracks	*Users* can open a playlist and start listening to their tracks. A track player is available to skip and pause songs.	*User*
2.8. Filter/Sort Playlists	*User* browses playlists and can sort and filter them using different playlist metadata fields.	*User*

3. Main Views (SPA Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Home and User Registration	Presents login and register form, as well as playlists of users from the web app.	*/*
3.2. Create Playlist	Presents a form for creating a playlist. The form contains auto suggest address fields, 20 genres to choose from with a limit of 5, as well as percentage distribution for each genre, a playlist title and an image cover.	*/new-playlist*
3.3. Browse Playlists	Presents the playlists of all users in the web app. And the ability to filter and sort them. The playlists can be hoovered for additional info and clicked so the user can start listening to their tracks. (playlist like option might be included)	*/all-playlists*
3.4. Edit Playlist	Presents a form for playlist edit. Editing existing playlists is limited to changing the title, cover image or associated genre tags, but does not include editing of the tracklist.	*/edit-playlist*
3.5. User Data	Presents ability to view and edit personal *User Data*.	*/personal*
3.6. User Profile	Presents ability to browse, filter and sort *User's playlists*.	*/profile*
3.7. Playlist Data	Presents the tracks of a playlist with ability to start listening to them, as well as playlist info card and a track player that can play and skip tracks and a volume and a track duration sliders.	*/playlist*
3.8. Users	Presents ability to manage (CRUD) Users and their User Data (available for *Administrators* only).	*/users*

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. Users	GET *User Data* for all users. Available only for *Administrators*.	*/api/users*
4.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*.	*/api/users/{userId}*

4.3. Sign in	POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests.	*/api/auth/signin*
4.4. Sign up	POST a signup request for registering a user and saving him in the database.	*/api/auth/signup*
4.5. Generate Playlist	POST Playlist, and return list of all Playlists of current user.	*/api/playlists/generate*
4.6. Playlist	GET, PUT, DELETE *Playlist*.	*/api/playlists/{playlistId}*
4.7. Paginate Playlists	GET page size amount of playlists.	*/api/playlists/page*
4.8. All Genres	GET all genres.	*/api/genres/get-all*
4.9. User	GET, PUT, DELETE *User* with specified user*Id*.	*/api/users/{userId}*
4.10. Filter Playlists	GET *Playlists by the given filter criteria.*	*/api/playlists/filter/{criteria}*
4.11. Sort Playlists	GET *Playlists by the given sort criteria.*	*/api/playlists/sort/{criteria}*