|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | Kristiyan Krumov | | kristiyan.boyanov@gmail.com | face-to-face |

|  |  |
| --- | --- |
| Project name | Gramophone - social media player |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| **Gramophone** is a social media player that lets you share music across limitless people, devices and speakers, whether they're in the same channel or halfway across the world. The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express + socket.io*** as backend technologies.  The main user roles (actors in UML) are:  • *Anonymous User* – can create, share & join channels. Add songs to the playlist.  • *Registered User* – can create, share & join channels. Can also send messages in the channel, save songs to their personal playlist and add songs to the channel playlist. |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Creating a room** | The user can create music channels determining the option weather they will be public or local. Local channels are only visible to users within the same network. | All users |
| * 1. **Register & Login** | *Anonymous User* can register in the system by SSO (Facebook and Soundcloud auhentication). | *Anonymous User* |
| * 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*: *Student, Instructor*, or *Administrator*. | *Registered User, Administrator* |
| * 1. **Manage Users** | *Administrator* can browse and filter users based on different criteria: first and last name, email.  *Administrator* can choose a *User* to manage, and can manage the chosen User - edit (using Change User Data UC) or delete. | *Administrator* |
| * 1. **Join public channel** | *All users can join public music channels, shared by url. User has joined music channel, the current song is automatically played in sync with all users currently listening on the channel.* | *All users* |
| * 1. **Pause song, mute, change volume** | *All users can control their volume, mute or pause the song. When unpaused, the song continues from the synced position.* | *All users* |
| * 1. **Add song to the channel playlist** | *All users in a channel can add songs to the playlist.* | *All users in a channel* |
| * 1. **Send message to the channel** | *Registered users can send text and sound messages to the channel. Sound messages are played with the current song on the moment of receiving.* | *Registered users* |
| * 1. **Inviting people to the channel** | *People could be invited in channel by sending URL, Sharing the channel on facebook or in Slack channel.* | *All users in a channel* |

|  |  |  |
| --- | --- | --- |
| 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 and create room. | / |
| * 1. **Channel page** | Presents the music player with information about the current song, the playlist, chat and list with the people currently listening on that channel. Channel sharing options. | */channel* |
| * 1. **User Data** | Presents ability to view and edit personal *User Data*, saved songs, joined channels. | */me* |
| * 1. **Users** | Information for selected user – saved songs. | */user/{id}* |
| * 1. **About** | Presents information about the *system* 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*. | */api/users/{userId}* |
| * 1. **Channels** | GET channels returns all available channels in the user’s network. POST creates a channel | */api/channels* |
| * 1. **Selected channel** | GET, PUT, DELETE provide RUD operations for selected channel. | */api/channels/{channelId}* |
| * 1. **Fetch channel playlist** | GET gives list of all songs in the channel’s playlist. POST adds new song to the playlist. | */api/channel/{channelId}/songs* |
| * 1. **Fetch song from the playlist** | GET, PUT, DELETE *requests provide RUD operations for selected song.* | */api/channel/{channelId}/songs/{songId}* |