|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Assignment 1 | | Online Chat | | |
| 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 | Georgi Petkov | | georgi.petkov@sap.com | face-to-face |

|  |  |
| --- | --- |
| Project name | Online Chat |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| Nowadays with all current social networks there are many problems and risks arising. To name a few:   * there is access from many and different types of devices which all should be secured * tons of private information are being stored and need protection * your private information may be used for unclear purposes * the need to easily access the network requires installing applications on each device with tons of features you may not need * these solutions are not lightweight and have big consumption of both memory and bandwidth (and battery for mobile devices)   All this can be avoided for the most frequent use case - simple communication between users. It can be achieved with simple online chat storing only credentials and if demanded - chat history. It requires nothing but a browser so it’s available to all devices without any additional applications. The private information being stored and resource consumptions are reduced to minimum.  The system will be developed as a *Single Page Application (SPA)* using ***React.js*** as front-end, and ***Node.js + express*** as backend technologies. ***MongoDB*** will be used for persistence.  The main user roles are:   * *Anonymous User* – can only view the information page and register * *Standard User* – can maintain friends list, send messages and adjust privacy options * *Administrator* – can manage users |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Manage users** | *Administrator* can list, create, delete and ban/unban users. | *Administrator* |
| * 1. **Register & Login** | *Anonymous User* can register and login into the system by providing a valid username and password. | *Anonymous User* |
| * 1. **Logout** | *Any User* can logout from the system. | *Any User* |
| **2.3. Send Message** | *Standard User* can send messages to any other user. | *Standard User* |
| **2.4. Manage Friends List** | *Standard User* can add/remove friends and accept/reject friend requests | *Standard User* |
| **2.5. Broadcast system message** | *Administrator* can broadcast system message to all users. | *Administrator* |
| **2.6. Receive notifications** | *Administrator and Standard User* can receive notification while being online or after they login. These notifications can be for friend’s login/logout, received message from allowed user, receiving a system message from administrator and so on. | *Administrator, Standard User* |

|  |  |  |
| --- | --- | --- |
| 1. Main Views (SPA Frontend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Home** | Presents the introductory information for the system as well as detailed instructions how to start using it. Prominently offers ability to login or register. | / |
| * 1. **Chat** | Presents the chat and the friends list together with the notifications list (available to standard users only). | */chat* |
| * 1. **Users** | Presents ability to manage (CRUD) users and send messages (available for administrators only, as described in UCs). | */administration* |
| * 1. **About** | Presents information about the project and his owner. | */about* |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (Node.js Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Logged User** | GET the current user if an already created session exists. Used to determine if login is required.  POST Tries to authenticate the user with the given credentials. | */login* |
| * 1. **Logout** | DELETE Logouts the current user by deleting the current session. | */logout* |
| * 1. **Register** | POST Registers a new user with the given credentials. | */register* |
| * 1. **Users** | GET list of all users, POST new user. Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE user with specified *userId*. | */api/users/{userId}* |
| * 1. **Friend** | POST and DELETE for respectively send/accept request and remove friend/reject request | */api/users/{userId}/friends-list/{userId}* |
| * 1. **Chat History** | GET, DELETE chat by id. | */api/chat/{chatId}* |
| * 1. **Messages** | POST for a sending a new message . | */api/chat/{chatId}/messages* |
| * 1. **Notifications Channel** | URL for web sockets connection for real-time notifications from the backend. | */api/notifications* |

Used packages in backend:

express

express-session

express-ws

path

webpack

webpack-dev-middleware

webpack-hot-middleware

bcryptjs

mongoose

http-status-codes

body-parser

Used packages in frontend:

react

prop-types

react-bootstrap

react-router-dom

react-bootstrap-tabs

redux

react-redux

react-toastify

axios

Used packages for deployment/development:

babel

webpack

css-loader

eslint

jsbeautify