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

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

Project name	DormShare

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

Living in a dorm can be substantially easier if there's a sufficient level of communication between tenants. DormShare is intended to make it possible through just a few clicks to create "events" or "requests" that anyone in the same building can see and choose whether to respond. 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 view the registration page.
- Confirmed *Tenant* (extends *Tenant*) – can create events or make requests - an event could be "going to the shop", which could be useful for picking up groceries for a fellow tenant; a request could be a plea for a particular thing, for example wine opener or a cup of milk. Each tenant has a personal score, reflecting how many times he's made a request and how many times he's responded to a request in order to prevent exploiting the system.
- *Tenant* (extends *Registered User*) – same as Confirmed Tenant, except that he has a limited time to use the application until confirmed by a Building Administrator.
- *Building Administrator (for short BA)* (extends *Confirmed Tenant*) – can set up specific rules for the building he's administering, for example "no tenant can have a score of less than -500" or "you can't make more than 1 request a day". Every tenant has to choose upon registration a building he's registering to. After registration he'll have n-day access to the application, unless his registration is confirmed by the specific Building Administrator - when the tenant is confirmed, he'll have unlimited access to the application. Building Administrators also have to register and their registrations have to be confirmed by the Administrator. Until then, the Building Administrator's registration is put on hold. To register as a Building Administrator, the user must provide information about the building he's Administering. He can also give up his role as BA, promoting another regular tenant and demoting himself to a regular tenant.

• *Administrator* (extends *Registered User*) – the Administrator answers Building registrations - a new Building is created when a user attempts to register as a Building Administrator. The Administrator can choose to confirm or reject a BA registration query.

2. Main Use Cases / Scenarios		
Use case name	**Brief Descriptions**	**Actors Involved**
2.1. Register	*Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing password. To register as a Tenant, one must also provide the name of his building, while to register as a Building Administrator, one must also provide data about the building, such as location and name. Tenants have to be confirmed by BAs, while BAs have to be confirmed by the Administrator	All users
2.2. Change User Data	A tenant can view and change his profile picture, view his rating and view other tenants' rating. A BA can do all a tenant can do and change his building's name and sharing rules. *The Administrator* can change building names and set global rules.	*All users*
2.3. Manage Users	*BAs can ban tenants in their respective building from using the application and pass their role as BA to another tenant. They can also view all tenants in their building's rating.* *The administrator can change a building's administrator, choosing from the pool of tenants in that building*	*All users*
2.4. Manage Building Rules	*BA's can create rules for the building for events and requests - examples of such rules were given above.* The Administrator can set up global rules that apply to every building.	*Administrator, BA*

2.5. Create a request	*Tenants can create requests, asking for something. When creating a request, the tenant's personal score will go down, but only when the request is fulfilled by someone. The tenant should specify how will the request's criteria be satisfied (i.e. "meet on floor 5 to receive").*	*Tenant, Confirmed Tenant, BA*
2.6. Answer a request	*Tenants can answer requests, by marking them. They then have to follow through with the request withing reasonable timeframe. Once marked by one tenant, the request cannot be marked by another.*	*Tenant, Confirmed Tenant, BA*
2.7. Complete a request	*After a request is answered, the tenant who created it has to either finish a request, or unmark it in case the responding tenant doesn't follow through. Upon completion, the requesting tenant has score subtracted and the answering tenant has score added.*	*Tenant, Confirmed Tenant, BA*
2.8. Create an event	*Tenants can set up events (i.e. "going to the shop"). They can set up a limit for how many other tenants can answer to the event and can kick tenants from the event. The tenant can give a time frame for joining.*	*Tenant, Confirmed Tenant, BA*
2.9. Join event	*All tenants can join in on another tenant's event. They have to specify their particular request concerning the event.*	*Tenant, Confirmed Tenant, BA*
2.10. Complete event	*Upon completion of the event, all tenants who have joined in have to mark their request to the event fulfilled. This subtracts points from the tenants who have joined and adds points to the tenant who created the event.*	*Tenant, Confirmed Tenant, BA*
2.11. Report user	*Both sides in events and requests can be reported to the BA - if they promised they'd do something but didn't or if they didn't mark an event completed even though it was. Also tenants can report their BA to the administrator.*	*Tenant, Confirmed Tenant, BA, Administrator*
2.12. Review report	*A BA reviews reports and may mark users. When a user has been reported too many times, he can be banned temporarily or permanently.* *The administrator can review reports of a BA and appoint a new one.*	*Tenant, Confirmed Tenant, BA, Administrator*

3. Main Views (SPA Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Home	When logged in, shows the user profile, created events and requests, options to create events or requests. When not logged in, redirects to the login page.	*/*
3.2. Login	Allows users to login. Has link to the registration page.	*/login*
3.3. Register	Allows users to register as regular tenants or as BAs.	*/register*
3.4. Confirm registration	The administrator can confirm registration of BAs and BAs can confirm registrations of users in the building they administer.	*/confirm*
3.5. Manage rules	The administrator can set up global rules. Every BA can set up rules for his own building.	*/rules*
3.6. Review reports	The administrator and every BA can view reports they've received and take action.	*/reports*
3.7. Create event	Allows a tenant to create an event.	*/event*
3.8. Create request	Allows a user to create a request.	*/request*
3.9. Manage tenants	Allows a BA to view all tenants with their scores and to ban them.	*/tenants*
3.10. Manage BAs	Allows the Administrator to view all buildings and their BAs and to appoint new BAs.	*/bas*

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. Users	GET *User Data* for all users, and POST new *User Data*. Available only for the *Administrator and BA*.	*/api/users*
4.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*.	*/api/users/{userId}*
4.3. Login	POST *User Credentials* (e-mail address and password) and receive a valid *Security Token* to use in subsequent API requests.	*/api/login*
4.4. Logout	POST a logout request for ending the active sessio, and invalidating the issued *Security Token*.	*/api/logout*

4.5. Events	GET *events*, and POST new *events.*	*/api/events*
4.6. Event	GET, PUT, DELETE *specific event*.	*/api/events/{eventId}*
4.7. Requests	GET *requests*, and POST new *requests.*	*/api/requests*
4.8. Request	GET, PUT, DELETE *specific request*.	*/api/requests/{requestId}*
4.9. Reports	GET *reports*, and POST new *reports.*	*/api/reports*
4.10. Report	GET, PUT, DELETE *specific report*.	*/api/reports/{reportId}*
4.11. BAs	GET *Ba list*, and POST data for new *BAs.*	*/api/bas*
4.12.BA	GET, PUT, DELETE *specific BA*.	*/api/bas/{basId}*