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

Project author		
№	Pseudonym	Face-to-face/ online
1	sis0k0	face-to-face

Project name	PupCare

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

PupCare provides 24/7 on-demand access to people seeking dog walking jobs as well as sitters and boarding services in your area. 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. There might also be an Android/iOS app (**React Native**) accompanying the website. The main user roles (actors in UML) are:

- *Dog Carer* (extends *Registered User*) – can search for dog walking/sitting jobs in their area, apply for a job, submit post-walk reports.
- *Dog Owner* (extends *Registered User*) – can register a job for dog sitting/walking, see profiles of dog walkers, has access to post-walk reports.
- *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, access walk reports, jobs.

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,	*Anonymous User,*

	and choosing password. By default, all new registered users have *Dog Owner* role. *Administrator* can register new by entering *User Data* and choosing a Role (*Dog Owner, Dog Carer*, or *Administrator*).	*Administrator*
2.2. 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*: *Dog Owner, Dog Carer*, or *Administrator*.	*Registered User, Administrator*
2.3. Manage Users	*Administrator* can browse and filter users based on different criteria: first and last 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*
2.4. Manage Jobs	*Dog Owner* can browse her/his *Jobs*, add new *Job* using *Add/Edit Job UC*, and delete a *Job*, as well as view the *Post-walk Reports* attached to their own *Jobs.* *Administrator* can browse all jobs, edit and delete them.	*Dog Owner,, Administrator*
2.5. Add a Job	*Dog Owner* can browse her/his *Jobs*, add new *Job* using *Add/Edit Job UC*, and delete a *Job*, as well as view the *Post-walk Reports* for their own *Jobs*. *Administrator* can browse jobs of all *Dog Owners*, edit and delete them.	*Dog Owner, Administrator*
2.6. Add/Edit Job	*Dog Owner* or *Administrator* specifies/edits *Job* meta-data such as: Job name, subject, description, tags. *Dog Owner* or *Administrator* saves the edited *Job.*	*Dog Owner, Administrator*
2.7. Apply for a Job	*Dog Carer* browses Jobs available and can sort and filter them using different Job metadata fields. *Dog Carer* chooses a Job to complete.	*Dog Carer,*
2.8. Select Dog Carer	*Dog Owner* can select one of the *Dog Carer*s applying for a Job.	*Dog Owner*

2.9. Browse Job Results	*Dog Carer* can browse her own completed *Job Results*. *Dog Owner* can browse the *Job Results* for *Jobs* created by her/him and for particular *Dog Carer Groups* assigned to the *Job*.	*Dog Carer, Dog Owner*
2.10. Complete Job	*Dog Carer* completes a Job by submitting a post-walk report: - questionnaire - uploaded photo of the walked pup	*Dog Carer*

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. Prominently offers ability to register.	*/*
3.2. Jobs	Presents Jobs available according to *User's Role* and identity. Offers abilities to browse, choose, create, read, update, delete (CRUD) Jobs, as defined by *UCs* (for *Administrators* and *Dog Owners* only).	*/jobs*
3.3. Apply Job	Presents a Job if available to particular *Dog Carer* allowing them to apply for it.	*/jobs/:id*
3.4. Job Results	Presents *Job Results* available according to *User's Role* and identity as defined by *UCs*.	*/job-results*
3.5. User Registration	Presents a view allowing the *Anonymous Users* to register in *OKTS*.	*/register*
3.6. Login	Presents a view allowing the users to login.	*/login*
3.7. User Data	Presents ability to view and edit personal *User Data*.	*/profile*
3.8. Dashboard	Dog Carer - Presents in real time the Dog Carer's progress on applied Jobs. Dog Owner - Presents in real time the Dog Owner's progress on created jobs.	*/dashboard*
3.9. Users	Presents ability to manage (CRUD) Users and their User Data (available for *Administrators* only, as described in UCs).	*/users*

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* (Id is auto-filled by *OKTS* and modified entity is returned as result from POST request). Available only for *Administrators*.	*/api/users*
4.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions decribed in UCs.	*/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 session with *OKTS,* and invalidating the issued *Security Token*.	*/api/logout*
4.5. Jobs	GET users (according to *User's Role* and identity) and POST new *Job* (Id is auto-filled and modified entity is returned as result from POST request).	*/api/jobs*
4.6. Job	GET, PUT, DELETE *Job Data* (including *Questions* and *Answers*) for *Job* with specified *jobId*.	*/api/jobs/{jobId}*
4.7. Job Report	GET *Job Result* (according to *User's Role* and identity)) for *Job* with specified *jobId*, and POST new *Job Result* (Id is auto-filled and modified entity is returned as result from POST request).	*/api/jobs/{jobId}/report*
4.8. Job Result	GET, PUT, DELETE *Job Result* (according to *User's Role* and identity) for *Job* with specified *jobId* and *Job Result* with specified *jobReportId.*	*/api/jobs/{jobId}/results/ {jobReportId}*