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	Georgi Bojinov	georgi.bojinov@hotmail.com	face-to-face
2	Dragomir Proychev	dragproychev@gmail.com	face-to-face

Project name	Jitter – next generation social network

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

Social media has become an important factor in people’s lives, reaching the point where they can’t live without it. Everything from communication to sharing content, keeping up with all kinds of people and even job hunting, social media can do. As a result, their complexity has risen exponentially, and the number of features can be overwhelming for users. The need for more intuitive and straightforward social networks has arisen and as such, Jitter aims to focus on just that – a simple, intuitive UI, having the essential features but building them in a simple, easy to understand way – a chat system that is fast and efficient and doesn’t use external applications, a like and comment system that makes sure you want to like what you’re liking, a post system that allows for full editing and deletion of content, and a simple user system without multiple role splitting. Jitter is simple, straightforward and intuitive – just what a social network has to be.

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved

2.1. News feed	The *User* can browse his news feed and see all recent posts by friends and his/her own posts.	All users
2.2. Register	*Anonymous User* can register in the social network by providing a valid e-mail address, first and last name, choosing a password and providing profile data – location, education, interests.	*Anonymous User*
2.3. Change Profile	*User* can view and edit his/her personal profile and choose what is private, can only be seen by friends or is public.	*User*
2.4. Add/Edit Post	*User* can make a post on his/her personal page, update, delete, and view it and select whether it is private or not.	*User*
2.5. Add/Edit Comment	*User* can add a comment to a post if commenting is allowed and if the user can view the post. The user can also edit and/or delete the comment.	*User*
2.6. Add/Remove Like	*User* can like a comment or a post and remove his/her own likes on posts and comments.	*User*
2.7. Add/Remove Friend	*User* can add or remove other users from his/her friends list. He/she can also accept or reject friend requests from other users.	*User*
2.8. Send chat message	*User* can send a message to their friends or to another user if he/she has permitted that.	*User*
2.9. Permit chat message	*User* can specify who can send him/her chat messages – only friends or everybody.	*User*
2.10. View/Archive messages	*User* can view all his/her messages and message history with anybody. He/she can also choose to archive any messages that are obsolete.	*User*

3. Main Views (SPA Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Home	Displays the news feed of the currently logged in user, including the posts and recent activity of the user's friends.	/
3.2. Login	Allows an anonymous user to log into the system	*/user/login*
3.3. Register	Allows an anonymous user to create a new account	*/user/register*
3.4. Profile	Displays the profile and the activity of the user with the given id	*/profile/{id}*
3.5. Messages	Displays a list of all conversations for the currently logged in user	*/messages*

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. Users	POST Create a new user with the given credentials	*/api/users*
4.2. User	GET, PUT, DELETE Information about user. Available only for the current user.	*/api/users/{id}*
4.3. Profile	GET Information about a user's profile. PUT Update your own profile by modifying the existing information.	*/api/profile/{id}*
4.4. Posts	POST Create a new post	*/api/posts*
4.5. Post	GET A post with the given id. PUT, DELETE A post the logged in user has created.	*/api/posts/{id}*
4.6. Comments	POST Create a new comment for a post with the given id. GET All comments for the corresponding post	*/api/comments/{postId}*
4.7. Comment	PUT, DELETE A comment the logged in user has posted.	*/api/comments/{commentId}*

4.8. Likes	POST Give a like to a post with postId. GET All likes for an existing post. DELETE The user's like of a post if the logged in user has already liked it	*/api/likes/{postId}*
4.9. Messages	GET All messages for the conversation between the logged in user and user with userId. POST Send a new message for this conversation.	*/api/messages/{userId}*
4.10. Message	PUT Modify an existing message. DELETE Delete a message	*/api/messages/{messageId}*