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	Veselin Stoyanov	stoyanov.veseline@gmail.com	face-to-face
2	Denitsa Slavcheva	denica.slavcheva@gmail.com	face-to-face
3	Emil Dudev	emildudev@gmail.com	face-to-face

Project name	Carpooling System with automated Messenger Bot

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

Travelling is one of the most important things for people nowadays. The Carpooling System with automated Messenger Bot aims to solve one common issue in a modern and interactive way. Drivers usually travel with no passengers in their cars. Travellers can easily join drivers to get to the desired destination. This way the passengers and the driver travel cheaper, releasing less CO2 in the atmosphere. Classic carpooling apps require tedious registration, large user base and more time to setup a ride or find one. For this reason our project will introduce a unique Messenger Bot making the whole process easier for both sides.

The system will be developed as a Singe Page Application (SPA) using React.js as front-end, and Node.js + Express.js as back-end technologies.

The main user roles (actors in UML) are:

- Anonymous User – can browse dynamic travel pages (if they know the unique URL)
- Messenger User - can act as driver or passenger
  - Driver – can start travels, dis/approves passengers
  - Passenger – can participate in travels
- Admin User - can manage all users (ban), travels (inspect, cancel, kick people, etc.), can browse statistics

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
2.1. Create a carpool	Messenger users can create a carpool by chatting with a Bot which asks questions. (from, to, date, time, free seats, food, cigarettes, animals)	Messenger User (Driver)
2.2. Find a carpool	*Messenger bot helps Messenger users to find a carpool by asking them questions. (from, to, date, time)*	*Messenger User (Passenger)*
2.3. Accept Passengers	*Messenger bot messages the drivers when a passenger is found. The driver can accept or decline a passenger. If a passenger is accepted he/she is also informed by the bot. Both sides get their mobile phones.*	*Messenger User (Driver)*
2.4. Driver Profile	*Messenger Bot allows driver to fill/edit their personal profile. (car make, model, common preferences - food, cigarettes, animals) etc.*	*Messenger User (Driver)*
2.5. Members Feedback	*12 hours after a carpool has finished, all participants will be asked to provide feedback for the others. This way a trusty*	*Messenger User*

	network will be formed.	
2.6. Dynamic Travel Pages	*Every travel will have its own unique web page with driver, passenger, dynamic map (Google Maps API) and other info. This page won't be indexed by search engines and only the people with the URL will have access to it.*	*Anonymous User, Admin User*
2.7. Informative Website	*The entire system will be explained in an informative SPA website.* • *Homepage* • *About* • *How to use* • *The Idea* • *Stats Page*	*All roles*
2.8. Admin Interface	*Administrators will have a web interface in which they will manage users, travels, access restrictions, etc. They will also have access to various travel statistics.*	*Admin User*

3. Main Views (SPA Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Homepage	Presents the introductory information for the purpose of the system. Prominently offers ability to use via Messenger.	/
3.2. About	Presents information about the functionalities of the system and the authors.	/about

3.3. How to use	Presents detailed information on how to use the platform for all the different roles.	/how-to-use
3.4. The Idea	Explains to users why to carpool.	/the-idea
3.5. Stats	Shows various statistics from the system usage, encouraging people to use the system.	/stats
3.6. Admin	Presents all admin related functionalities as described in the UCs.	*/admin*
3.7. Travel Info Page	Presents all the related information for a particular travel. • Travel info (from, to, date, time, additional preferences) • Dynamic Route Map (Google Maps API) • Participants (profile information, rating, etc.)	*/travel/{{ id/hashed key }}*
3.8. Users	Presents ability to manage (CRUD) Users and their User Data (available for Administrators only, as described in UCs).	*/admin/users*
3.9. User Profile	Allows users to manage their profiles from the web. (requires Facebook login)	/authorize /user/{ id }
3.10. Allow web interactions (if we have enough time to finish the other tasks)	The main idea of the project is that all interactions with the system happen in Facebook's Messenger App via a Messenger Bot. However some users might want to do the same through the web. For this reason the same functionality will be available online if it is more convenient to the user. It will require Facebook authorization though.	/authorize (POST) /profile (GET) /profile/update (PUT) /travels (GET)

		/travel/add (GET) /travel/create (POST) /travel/{{ id }} (GET) /travel/{{ id }}/edit (GET) /travel/{{ id }}/update (PUT) /travel/{{ id }}/delete (DELETE) /travel/{{ id }}/join (POST) /travel/{{ id }}/leave (POST) /user/{{ id }} (GET) etc.

4. API Resources (Node.js Backend)		
View name	**Brief Descriptions**	**URI**
4.1. Messenger Bot	Initiates conversations, handles user responses, adds passengers to queues, notifies users for various events, etc.	*/api/travel/create* */api/travel/{{ id }}/passenger/create* */api/travel/{{ id }}/passenger/delete* */api/travel/{{ id }}/passenger/approve* */api/travel/ {{ id }}/passenger/disapprove* */api/travel/{{ id }}/notify/host*

		/api/travel/{{ id }}/notify/passenger */api/user/create* */api/user/{{ id }}/profile/update* */api/travel/{{ id }}/passenger/ {{ id }}/vote* */api/user/{{ id }}/votes* *etc.*
4.2. Admin	Admin related endpoints	*/api/admin/* */api/admin/travel/{{ id }}/cancel* */api/admin/travel/{{ id }}/update* */api/admin/travel/{{ id }}/delete* */api/admin/user/{{ id }}/update* */api/admin/user/{{ id }}/ban* */api/admin/stats* *etc.*

